

Министерство просвещения Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский педагогический государственный университет»
Кафедра теории и методики обучения математике и информатике



«ИТ-КЛАСС В МОСКОВСКОЙ ШКОЛЕ»

ИНФОРМАТИКА

(БАЗОВЫЙ КУРС)

11 КЛАСС

Электронное издание сетевого распространения

МПГУ
Москва • 2021

УДК 372.800.4(063)
ББК 74.263.2я431
А437

к.п.н. Павлов Д.И., к.п.н. Ткач Т.В., Ремизова Е.Г.,
Бутарев К.В., Комбарова Н.Ф., Шамасова Т.Л.
под научной редакцией д.п.н. Босовой Л.Л.

Ведущий редактор: к.п.н. Горячев А.В.
Редактор-корректор: Каплан А.В.

Информатика (11 класс): материалы базовой программы по
Информатике проекта «IT-Класс в московской школе» –
Москва: МПГУ, 2021. – 816 с.

А437

Настоящее издание содержит учебные материалы в соответствии с рабочей программой базового курса информатики для 11 класса проекта «IT-класс в московской школе», в частности, теоретическую информацию, разборы типовых задач, практические задания, задания для самостоятельной работы и рефлексии.

Будет полезно учителям информатики и методистам, реализующим базовый курс информатики в 11 классе в рамках программы «IT-Класс в московской школе», а также ученикам школ.

УДК 372.800.4(063)
ББК 74.263.2я431

© МПГУ, 2021
© Коллектив авторов, 2021

Оглавление

Элементы оглавления не найдены.

I. Графика и мультимедиа

§1.1 Классификация и устройство вычислительной техники. Установка и подключение исполняемых библиотек

ВСПОМИНАЕМ

Уроки информатики – не просто работа с данными. Они тесно связаны с использованием компьютерной техники и прочего электрооборудования. Современные компьютерные классы оборудованы не только персональными компьютерами и ноутбуками, но и интерактивной доской и мультимедийным проектором. В кабинете информатики могут находиться принтеры, ксероксы и сканеры, роутеры и источники бесперебойного питания с подключенными к ним кабелями и розетками. Иначе говоря, в кабинетах информатики много электрооборудования, что в свою очередь требует особых условий проведения занятий. Особое внимание мы должны уделить вопросам соблюдения техники безопасности, охраны труда и пожарной безопасности.

С целью сохранения здоровья и минимизации возможных негативных последствий использования электротехники при проведении занятий в кабинете информатики **рекомендовано**:

- Сидеть вертикально, с прямой спиной, опущенными и расслабленными плечами;
- Руки держать на подлокотниках или поверхности стола, не допуская провисания локтя;
- Сохранять расстояние 60-70 см от глаз до монитора. Центр монитора рекомендовано располагать на 10-15 градусов ниже уровня глаз;
- Планировать свою работу так, чтобы не допускать непрерывной работы с динамическими или статическими изображениями на экране монитора больше 35 минут.

Также при проведении занятий в кабинете информатики **не рекомендуется**:

- Приходить в грязной одежде, с грязными руками, а также в верхней одежде и без сменной обуви;
- Использовать компьютеры одноклассников, в том числе клавиатуру, манипулятор “мышь”, отвлекаясь от выполнения собственной работы;
- Вставать или передвигаться по классу без указания учителя или не обратившись к учителю за разрешением заранее;
- Прикасаться к экрану монитора руками.

При проведении занятий **необходимо:**

- Включать и выключать технику только после получения указания от учителя;
- Аккуратно использовать оборудование, не допуская его повреждения;
- При обнаружении любой неисправности, не трогая компьютер, провода, соединения, прекратить работу и оповестить учителя;

ОБСУЖДАЕМ

Сегодня компьютеры окружают нас повсюду и большинство видов деятельности человека связаны с их использованием. Но какие они? Все одинаковые? Разные? В чём различия между компьютерами?

ЧИТАЕМ

Существуют разные классификации вычислительной техники в зависимости от выбранного основания. Первая классификация – *принцип действия*. По принципу действия выделяют три вида компьютеров:

1. **Аналоговые компьютеры.** Представляют числовые данные при помощи аналоговых физических переменных.
2. **Цифровые компьютеры.** Производят вычисление в цифровой форме.
3. **Гибридные компьютеры.** Совмещают в себе обработку сигналов как в аналоговой, так и в цифровой форме.

Следующее основание для классификации – *габариты и вычислительная мощность*. Эта классификация представлена на рисунке 1.

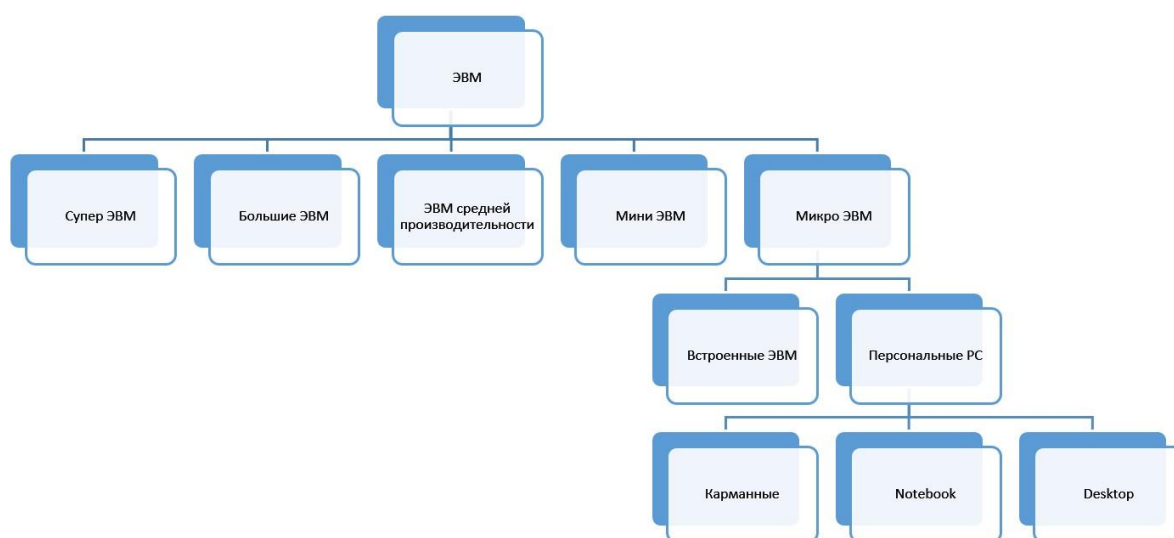


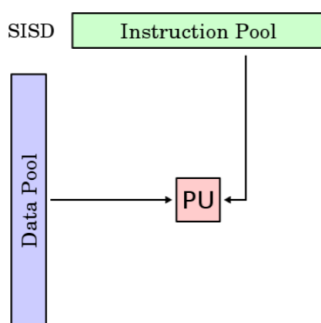
Рис. 1 Классификация по вычислительной мощности и габаритам

Существует также общая классификация архитектур ЭВМ по признакам наличия параллелизма в потоках команд и данных. Была предложена Майклом Флинном в 1966 году и расширена в 1972 году.

Одиночный поток команд (Single Instruction)

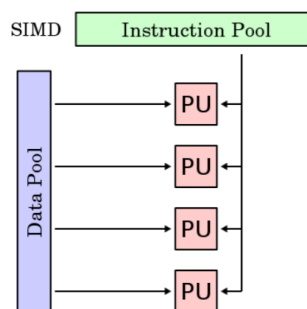
Одиночный поток данных
(Single Data)

SISD (ОКОД) – это традиционный компьютер с одним процессором, который выполняет последовательно одну инструкцию за другой, работая с одним потоком данных.



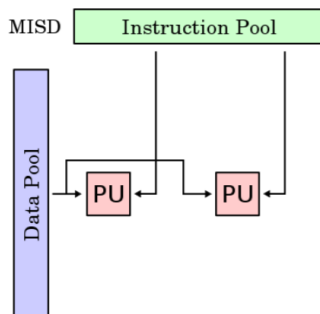
Множество потоков команд (Multiple Instruction)

MISD (МКОД) – компьютеры состоят из одного командного процессора и нескольких модулей обработки данных, называемых процессорными элементами. Командный процессор принимает, анализирует и выполняет команды. Если в команде встречаются данные, контроллер рассылает на все процессорные элементы команду, и эта команда выполняется на нескольких или на всех процессорных элементах. SIMD архитектура используется в графических ускорителях (Технология CUDA, AMD APP, OpenCL).

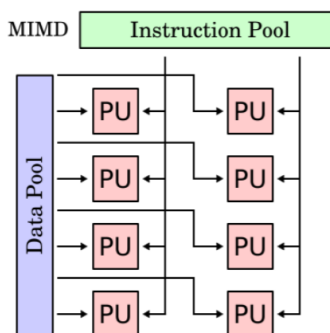


Множество потоков данных
(Multiple Data)

SIMD (ОКМД) – тип архитектуры параллельных вычислений, где несколько функциональных модулей (два или более) выполняют различные операции над одними данными.



MIMD (МКМД) – в этой архитектуре компьютеры имеют несколько процессоров, которые функционируют асинхронно и независимо. В любой момент различные процессоры могут выполнять различные команды над различными частями данных.



Основы архитектуры современных компьютеров заложены в середине XX века двумя выдающимися учёными своего времени – американцем венгерского происхождения Джоном фон Нейманом и советским конструктором вычислительных машин Сергеем Алексеевичем Лебедевым. Эти принципы были разработаны независимо друг от друга, но оказались настолько похожи, что сегодня их принято называть принципами Неймана-Лебедева, хотя это и не очень корректно. Принципы Неймана и Лебедева, имея общие черты, разнятся по глубине и актуальности, если говорить о дне сегодняшнем. Вот они:

Принципы Дж. фон Неймана

1. Компьютеры на электронных элементах должны работать в двоичной системе счисления.
2. Программа должна размещаться в памяти.
3. По форме представления команды и числа одинаковы.
4. Так как физически реализовать запоминающее устройство, обладающее одновременно высоким быстродействием и большой емкостью сложно, то память следует организовывать иерархически.
5. Арифметическое устройство компьютера конструируется на основе сумматоров – устройств, выполняющих операцию сложения.
6. Операции над двоичными кодами осуществляются одновременно над всеми разрядами.

Принципы С.А. Лебедева

1. Представление всей информации в двоичном виде и обработка ее в двоичной системе счисления.
2. Программный принцип управления и размещение программ в памяти машины; иерархическая организация памяти с применением разнофункциональных ее ступеней.
3. Операционно-адресный принцип построения команд в программах и возможность текущего изменения команд путем выполнения операций над ними, как над числами.
4. Иерархическая система машинных действий, состоящая из базовых операций, управляемых аппаратным способом, и составных процедур, реализуемых с помощью стандартных подпрограмм.
5. Построение базовых операций на основе элементарных операций, выполняемых одновременно над всеми разрядами слов.
6. Применение и центрального и местного управления вычислительным процессом.

Сравните принципы Дж. фон Неймана и принципы С.А. Лебедева. Обсудите, какие из них используются сегодня при разработке вычислительных машин (компьютеров, смартфонов и т.д.)?

ЧИТАЕМ

Обобщённо структуру всех современных вычислительных машин можно представить себе в виде схемы (рисунок 2).

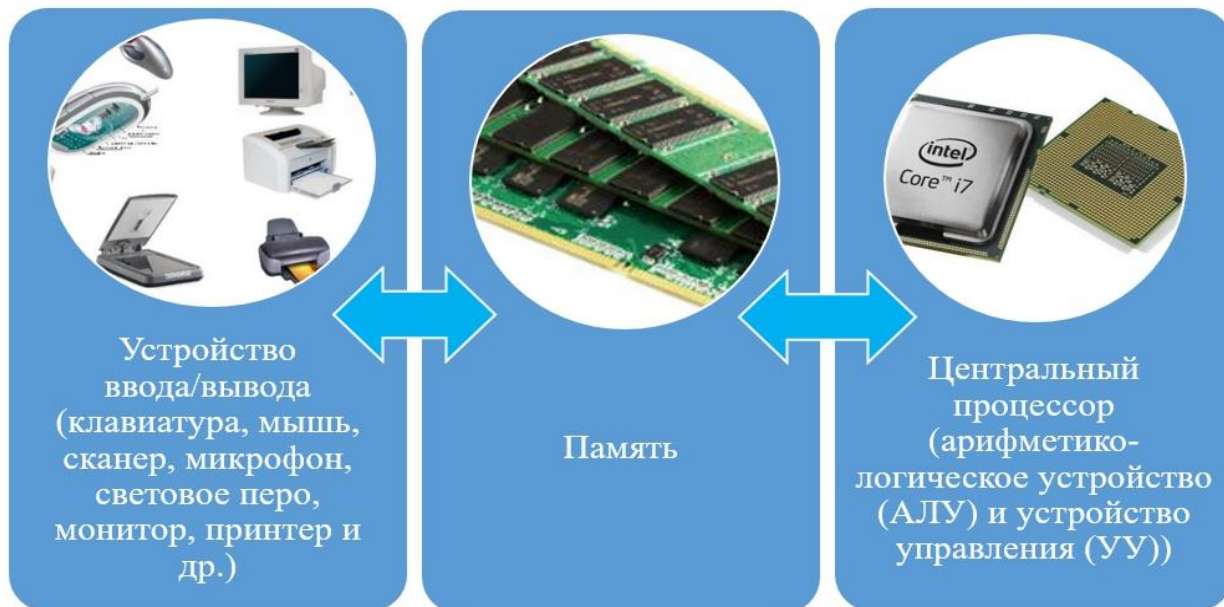


Рис. 2 Обобщённая структура компьютера

Для организации взаимосвязи (обмена информацией) между компонентами компьютера используется специальная подсистема, которая называется «системная шина» или просто «шина». Схема работы системной шины представлена на рисунке 3.

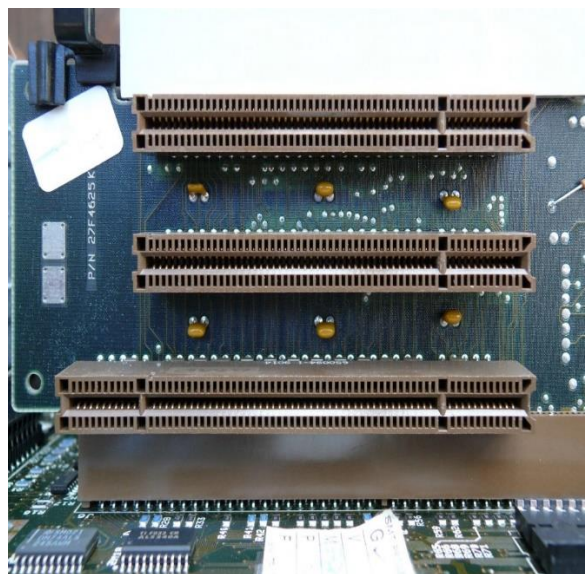
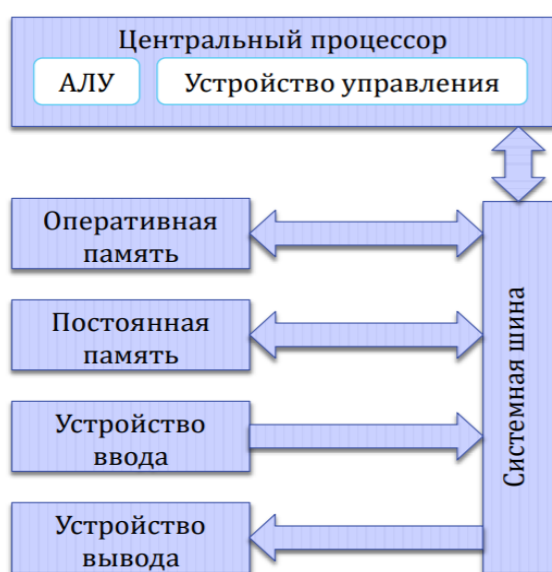


Рис. 3 Взаимодействие через системную шину

Большинство сведений об архитектуре и назначении компонентов компьютера Вам знакомы по обучению в 7–9 классе, а остальные сведения вы

можете найти в дополнительных источниках (некоторые ссылки в конце учебника).

Мы же начнём новый учебный год с освоения темы «Компьютерная графика». Обсудим, как именно аппаратные и программные компоненты компьютера работают с компьютерной графикой и как создаются графические объекты средствами программирования.

Для этого нам необходимо провести некоторые подготовительные мероприятия.

ГОТОВИМСЯ К РАБОТЕ НА ПК

В 10-м классе мы уже освоили некоторые возможности языка Python и интегрированной среды разработки для языка программирования Python – PyCharm. Вы уже наверняка знаете, что для решения отдельных задач с помощью языка Python используются специальные модули и библиотеки.

Для работы с графикой мы будем использовать модуль **graph**. Он представляет собой набор функций для создания учебных графических программ на языке Python на основе виджета **Canvas** библиотеки **Tkinter**. Вы уже знаете, что **Tkinter** это кроссплатформенная событийно-ориентированная графическая библиотека, входящая в стандартную библиотеку Python.

Для запуска графических программ модуль **graph** нужно поместить в ту же папку, где находится файл с программой. Программа с использованием модуля **graph** имеет следующий вид:

```
from graph import *  
... # текст программы  
run()
```

Среду разработки PyCharm (если вы не установили её в прошлом учебном году) и модуль **graph** можно скачать с сайта <https://kpolyakov.spb.ru/school/pycpp.htm>

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №1

Скачайте библиотеку **graph**. Запустите среду программирования PyCharm и создайте новый проект и новый файл. Установите и подключите графическую библиотеку, а после этого введите следующий код программы:

```
from graph import *  
penColor("yellow")  
brushColor("blue")  
rectangle(100, 100, 300, 200)  
brushColor("yellow")  
polygon([(100, 100), (200, 150), (300, 100), (100, 100)])  
penColor("blue")
```

```
brushColor("brown")
circle(200, 150, 20)
run()
```

Сравните результат с изображением на рисунке 4.

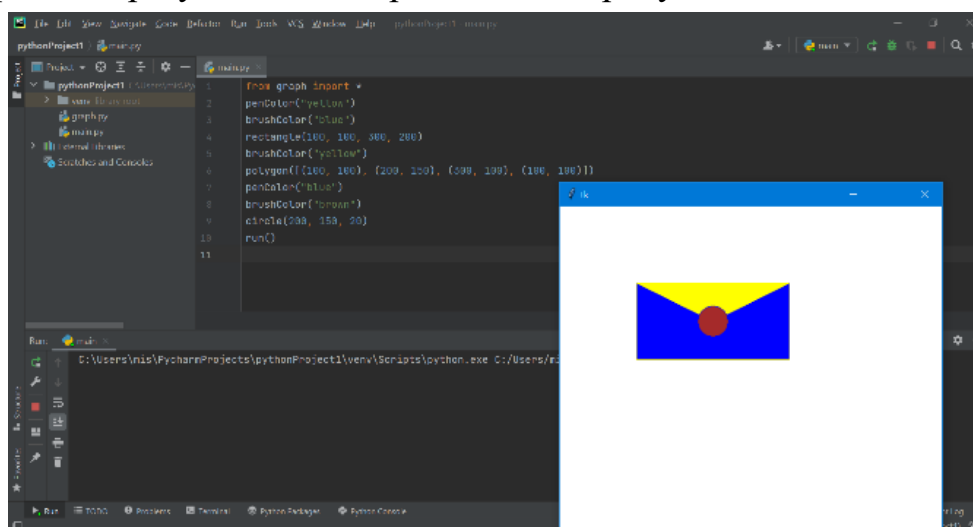


Рис. 4 Результат работы программы

ГОТОВИМСЯ К РАБОТЕ НА ПК

Кроме модуля *graph* для работы с готовыми изображениями на языке Python будем использовать библиотеку Pyllow. Для ее подключения необходимо установить настройки в созданном вами проекте. Для этого в меню File среды PyCharm выберите раздел Settings. Там найдите свой проект, войдите в меню Python Interpreter и нажмите на значок «+».

Затем в поисковой строке наберите Pyllow и нажмите кнопку Install Package (для более поздних версий PyCharm в поисковой строке наберите PIL и в выпавшем списке выберите Pyllow).

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №2

Создайте новый проект. Подключите библиотеку Pyllow. В папку со своим проектом поместите любое фото с именем foto.png и напишите следующий код программы:

```
from PIL import Image
image = Image.open('foto.png')
```

Данная программа откроет файл foto.png для дальнейшей работы с ним.

ГОТОВИМСЯ К РАБОТЕ НА ПК

Мы также будем использовать библиотеку Qt, которая является одной из самых мощных библиотек GUI (графического интерфейса пользователя). Для её запуска также необходимо произвести настройки в созданном вами проекте.

Для этого в меню File среды PyCharm надо выбрать раздел Settings. Там найти свой проект, войти в меню Python Interpreter и нажать на значок «+». После этого в поисковой строке набрать PyQt5 и нажать кнопку Install Package.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Подготовьте сообщение (в виде доклада, презентации, деловой графики) о возможностях и сферах применения аналоговых, цифровых и гибридных вычислительных машин.
2. Подготовьте сообщение (в виде доклада, презентации, деловой графики) о понятии, возможностях и сферах применения суперкомпьютеров.
3. Подготовьте сообщение (в виде доклада, презентации, деловой графики) о возможностях и сферах применения параллельных вычислительных систем.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Охарактеризуйте принцип работы компьютера, опираясь на обобщенную структуру его компонентов.
2. Охарактеризуйте принцип работы компьютера, опираясь на взаимодействие его компонентов через системную шину.

§1.2 Растровая и векторная графика

ВСПОМИНАЕМ

Вспомните, в чём заключаются основные принципы работы компьютера? Кем они сформулированы, и в чём различия этих формулировок?

ОБСУЖДАЕМ

В 7-9 классах вы познакомились с компонентами персонального компьютера. Какие из них предназначены для обработки графических изображений? Сравните свои представления об этих компонентах с представлениями одноклассников.

ЧИТАЕМ

Ближайшие уроки посвящены компьютерной графике. Под термином **компьютерная графика** мы можем понимать область человеческой деятельности, в которой с помощью компьютеров и программного обеспечения создаются и редактируются изображения.

Принято выделять четыре вида компьютерной графики – растровая, векторная, фрактальная и псевдотрёхмерная. Самыми распространёнными сегодня являются первые два вида.

Растровая графика

Вид компьютерной графики при котором изображение формируется путём построения матрицы пикселей (точек), а каждый символ окрашивается в тот или иной цвет путём смешения цветов той или иной палитры.

Самый органичный для компьютера способ представления изображения, так как на экране монитора изображение формируется из точек, подсвечивание которых создает видимое отображение текста и рисунков (рис. 5).

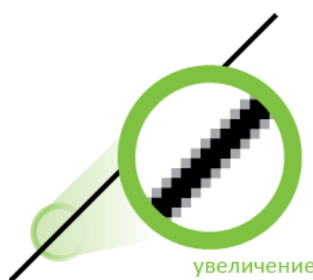


Рис. 5 Растровый способ отображения графики

При работе с растровыми изображениями приходится иметь дело со всеми точками рисунка. Например, если цветная фотография, полученная с помощью цифровой фотокамеры, имеет размеры 1800×1400 точек, то в памяти компьютера необходимо хранить информацию (цвет, яркость) о каждой из K ($K = 1800 \times 1400$) точек фотографии. Информационный объем растрового изображения (I) определяется как произведение числа входящих в изображение точек (K) на информационный объем одной точки (i – глубина цвета), который зависит от количества возможных цветов, т. е.

$$I = K \cdot i$$

При черно-белом изображении $i = 1$ бит/пиксель. Поэтому для хранения черно-белого изображения размером, например, 100×100 точек требуется 10 000 бит. А как определить, сколько информации требуется для хранения изображений большей цветности? Если принять за аксиому, что каждая точка изображения может быть с равной вероятностью окрашена в любой из доступных цветов, то для подсчёта количества информации необходимого для хранения растрового графического изображения мы можем использовать формулу Хартли:

$$I = \log_2 N,$$

где I – количество информации (объём памяти для хранения информации) и N – число равновероятных событий (число возможных цветов для окраски одного пикселя).

Так, если для окраски каждой точки выделено 8 возможных цветов, информационный объём точки равен 3 битам ($\log_2 8 = 3$). Информационный объём такого изображения размером 100×100 точек увеличивается в три раза: $I = 10\,000$ точек \times 3 бита = 30 000 бит. Цветное изображение получается за счет различной яркости трех основных цветов — красного, синего и зеленого (рис. 6).



Рис. 6 Цветовая модель RGB

Цветные изображения могут отображаться в различных режимах (использовать разное возможное количество цветов для окрашивания любой точки), соответственно изменяется и информационный объём точки (табл. 1).

Таблица 1. Информационный объём точки

| Режим | Информационный объём точки |
|-------------------|---|
| 2 цвета | $i = 1$ бит, черно-белое изображение |
| 8 цветов | $i = \log_2 8 = 3$ бита |
| 16 цветов | $i = \log_2 16 = 4$ бита |
| 256 цветов | $i = \log_2 256 = 8$ бит = 1 байт |
| 65 536 цветов | $i = \log_2 65536 = 16$ бит = 2 байта |
| 16 777 216 цветов | $i = \log_2 16777216 = 24$ бита = 3 байта |

Чем больше цветов содержит изображение, тем большую величину дает умножение информационного объема точки (i) на количество точек (K). Поэтому задача расчета объема видеопамати, необходимой для хранения изображения со всего экрана дисплея становится актуальной.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Условие:

Представим, что у Виктора есть телефон «Витязь», у которого осталось 200 Кбайт видеопамати. Достаточно ли этого места для того, чтобы сохранить фотографию размером 640×480 точек, при возможной палитре каждой точки 256 цветов (изображение формата gif)?

Решение:

Информационный объем точки для 256 цветов составляет $i = 8 \text{ бит} = 1 \text{ байт}$. Объем видеопамати должен быть не менее $I = 640 \times 480 \times 1 \text{ байт} = 307200 \text{ байт} = 300 \text{ Кбайт}$.

Ответ:

Нет, данного объема видеопамати не достаточно.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ**Условие:**

Сколько фотографий можно поместить на USB-накопителе объемом 2 Гбайт, делая снимки телефоном с разрешением экрана, равным 1280×1024 при количестве цветов 16 777 216.

Решение:

Информационный объем точки для 16 777 216 цветов составляет $i = 24 \text{ бита} = 3 \text{ байта}$. Вычислим размер одного изображения $I_1 = 1280 \times 1024 \times 3 \text{ байт} = 3932160 \text{ байт} = 3932160 / 1024 = 3840 \text{ Кбайт} = 3840 / 1024 = 3,75 \text{ Мбайт}$.

Объем накопителя $I = 2 \text{ Гбайт} = 2 \cdot 1024 = 2048 \text{ Мбайт}$.

Количество фотографий $2048 / 3,75 \approx 546$.

Ответ:

546 фотографий.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ**Условие:**

Размер экрана iPhone5 равен 3264×2448 пикселей. Какого максимального размера можно напечатать фотографию идеального качества разрешением 300 dpi?

Решение:

Вспомним, что означает «300 dpi». Dpi расшифровывается как dots per inch (точки на дюйм). Для перевода в метрическую систему нужно вспомнить, что $1 \text{ дюйм} = 2,54 \text{ см} = 25,4 \text{ мм}$. Само по себе значение dpi ничего не скажет, но будет полезно, если мы будем знать еще одну величину, чтобы найти третью:

$$\text{dpi} = \text{px} * 25,4 / \text{mm},$$

$$\text{px} = \text{dpi} * \text{mm} / 25,4,$$

$mm = px * 25,4 / dpi$, где px – количество пикселей по горизонтали или вертикали, mm – размер фотографии по горизонтали или вертикали, $25,4$ – перевод дюймов в миллиметры.

Итак, нам известны два значения: количество пикселей и количество dpi .
Найдем размеры фотографии:

$$mm = 3264 px * 25,4 / 300 = 276,$$

$$mm = 2448 px * 25,4 / 300 = 207$$

Ответ:

276 x 207 мм.

ЧИТАЕМ

Векторная графика

Векторное изображение состоит из набора элементарных деталей — графических примитивов: линии, прямоугольника, окружности, дуги и т.п. Положение этих элементов на экране определяется координатами точек и их характеристиками: цветом отображаемых деталей, толщиной линий и др. Векторный рисунок хранится так же, как хранятся тексты, формулы, числа, т.е. хранится не графическое его изображение, а только координаты и характеристики изображений его деталей. Поэтому для хранения векторных изображений требуется существенно меньше памяти, чем для растровых изображений. При запуске программы с векторным рисунком он создается каждый раз вновь, после чего в растровом виде может сохраняться в видеопамяти (рис. 7).



Рис. 7 Векторный способ отображения графики

РЕШАЕМ В ТЕТРАДИ

Задание 1.

Сколько точек должно быть в изображении, если его нужно напечатать размером 150x100 мм с разрешением 300 dpi ?

Задание 2.

Растровый графический файл был преобразован так, что его объем уменьшился в 2 раза. Сколько цветов было первоначально в палитре, если после преобразования было получено растровое изображение того же разрешения с глубиной цвета 8 бит/пиксель?

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Подготовьте сообщение (в виде доклада, презентации, деловой графики) о том, что такое фрактальная графика.
2. Подготовьте сообщение (в виде доклада, презентации, деловой графики) о том, что такое псевдотрёхмерная графика.
3. Воспользуйтесь дополнительными источниками информации и найдите, какие еще графические библиотеки можно использовать в Python и каковы способы их установки.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. В чём заключаются достоинства и недостатки растровой графики?
2. В чём заключаются достоинства и недостатки векторной графики?

§1.3 Основы работы с изображениями. Изменение формата, размеров и поворот изображения

ВСПОМИНАЕМ

На прошлом уроке мы познакомились с понятием «компьютерная графика» и её видами. Вспомните:

1. Какие виды компьютерной графики существуют и каковы особенности их применения?
2. Как определить размер файла размером 200 x 300 пикселей, если каждый пиксель может быть окрашен 10 цветами?
3. Какие модули и библиотеки мы можем использовать при обработке графических изображений с помощью языка программирования Python?

ЧИТАЕМ

Компьютерная графика прочно вошла в нашу жизнь. Вот неполный список тех отраслей, в которых она уже используется: реклама и маркетинг, разработка компьютерных и мобильных игр, киноиндустрия, разработка программ, оформление книг, архитектура.

Рано или поздно, но всем нам придется работать с графикой. В Python для этого есть несколько библиотек, и одна из них — библиотека Pillow.

Библиотека Pillow поддерживает следующие типы файлов: BMP, EPS, GIF, IM, JPEG, MSP, PCX PNG, PPM, TIFF, WebP, ICO, PSD, PDF. Некоторые типы файлов возможно открыть в режиме «только для чтения», в то время как другие возможно корректировать.

ГОТОВИМСЯ К РАБОТЕ НА ПК

Рассмотрим пример, как осуществляется работа с библиотекой Pillow. Будем знакомиться с некоторыми возможностями библиотеки на примере работы с изображением, которое на диске имеет имя fotos1.jpg:



Рис.8 Весенний пейзаж

Основные функции находятся в модуле Image. Вы можете создавать экземпляры этого класса несколькими способами: путем загрузки изображений из файлов, обработки других изображений, либо создания изображений с нуля.

Для импорта модуля Image используется запись:

```
from PIL import Image
```

После успешно выполненного импорта модуля вы можете получить доступ к функциям следующим образом:

```
Image = Image.open("fotos1.jpg")
```

Как только изображение будет загружено, с ним можно начинать работать. Для загрузки изображения с локального диска используется блок **try except** при работе с файлами. Чтобы загрузить изображение с помощью **try except** выполняют следующую комбинацию:

```
from PIL import Image
try:
    original = Image.open("fotos1.jpg")
except FileNotFoundError:
    print("Файл не найден")
```

Если же вы хотите загрузить изображение из сети Интернет (используя его URL-адрес), нужно дополнительно подключить библиотеку requests. Обратите внимание на синтаксис.

```
from PIL import Image
import request
url=`тут URL адрес изображения в сети`
resp = requests.get(url, stream=True).raw
img = Image.open(resp)
```

Теперь, когда у вас есть объект Image, вы можете использовать доступные атрибуты для проверки файла. Например, если вы хотите увидеть размер изображения, вы можете использовать атрибут format.

```
print("Размер изображения:")
print(original.format, original.size, original.mode)
```

Атрибут size — это tuple (кортеж), содержащий ширину и высоту (в пикселях). Обычные mode: L для изображений с оттенками серого, RGB для изображений с истинным цветным изображением и CMYK для печати изображений.

Для нашего изображения будет выведена следующая информация:

```
Размер изображения:
JPEG (400, 275) RGB
```

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №3

Попробуем на практике загрузить графический файл и определить его атрибуты. Придерживайтесь следующего плана:

1. Загрузите любую черно-белую фотографию с расширением .png в ту же папку, где находится ваш проект.
2. Импортируйте модуль Image и загрузите фотографию в буфер проекта.
3. Определите атрибуты вашего файла.

ГОТОВИМСЯ К РАБОТЕ НА ПК

При работе с изображениями нам может потребоваться изменить расширение файла. В Python Pillow метод save() позволяет конвертировать изображение в другой формат следующим образом:

```
from PIL import Image
import sys
try:
    original = Image.open("fotos1.jpg")
except IOError:
    print("Unable to load image")
    sys.exit(1)
original.save('fotos1.png', 'png')
```

Программа считывает изображение JPG и с помощью строки:

```
original.save('fotos1.png', 'png')
```

конвертирует его в PNG формат.

Второй параметр метода `save()` нужен для уточнения итогового формата изображения. С помощью метода `Image.convert()` можно сделать оригинальное изображение черно-белым. Для чего это может понадобиться? К примеру, для уменьшения занимаемого в памяти места. Если само изображение сохранить необходимо, но цвет не важен, можно использовать:

```
from PIL import Image
import sys
try:
    original = Image.open("fotos1.jpg")
except IOError:
    print("Unable to load image")
    sys.exit(1)
grayscale = original.convert('L')
grayscale.show()
```

Программа читает изображение и трансформирует его в **черно-белое**. За это отвечает следующая строка:

```
grayscale = original.convert('L')
```

Параметр метода `convert()` является модом. Мод 'L' представляет черно-белый вариант. Получается следующий результат:



Рис. 9 Демонстрация работы метода `Image.convert()`

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №4

Продолжите работу с изображением.

1. Осуществите конвертирование формата файла из PNG в JPG.
2. Преобразуйте черно-белую фотографию в цветную и сохраните ее на диске под другим именем.

Если затрудняетесь выполнить задание – используйте информационные материалы (ссылки в конце урока).

ГОТОВИМСЯ К РАБОТЕ НА ПК

Если вы определили размеры изображения и Вам необходимо изменить их, так как стоящая перед Вами задача требует изображения с другими значениями, можно использовать метод `resize()`.

Разберём три примера изменения размера:

- Изменение размера изображения с корректировкой ширины и высоты;
- Изменение ширины с учетом пропорций для высоты;
- Изменение высоты пропорционально ширине.

Изменение размера изображения с корректировкой ширины и высоты:

```
from PIL import Image
original = Image.open("fotos1.jpg")
foto2 = original.resize((100, 100), Image.ANTIALIAS)
foto2.show()
foto2.save("foto2.jpg")
```

При выполнении этой программы изображение нового размера сохраняется в файл с новым именем.

Изменение ширины с учетом пропорций для новой высоты изображения:

```
from PIL import Image
original = Image.open("fotos1.jpg")
width, height = original.size
new_width = 800 # ширина
new_height = int(new_width * height / width)
foto3 = original.resize((new_width, new_height),
Image.ANTIALIAS)
foto3.show()
```

Изменение высоты изображения пропорционально ширине:

```
from PIL import Image
original = Image.open("fotos1.jpg")
width, height = original.size
new_height = 600 # Высота
new_width = int(new_height * width / height)
foto4 = original.resize((new_width, new_height),
Image.ANTIALIAS)
foto4.show()
```


Возможны и другие действия над изображением, например, создание миниатюры или поворот изображения.

Создание миниатюры:

Создать миниатюру – уменьшенную версию изображения со всеми его наиболее важными аспектами – поможет следующий код:

```
from PIL import Image
size = (128, 128)
saved = "fotos5.jpg"
original = Image.open("fotos1.png")
original.thumbnail(size)
original.save(saved)
original.show()
```

Поворот изображения:

Поворот изображения осуществляется методом `rotate()` Pillow. `Image.rotate()` возвращает развернутую копию изображения.

```
from PIL import Image
original = Image.open("fotos1.jpg")
rotated = original.rotate(180)
rotated.save('fotos1_rotated.jpg')
```

ЧИТАЕМ

Дополнительную информацию по работе с графическими библиотеками в Python можно найти на сайтах:

<https://pythonru.com/biblioteki>

<https://python-scripts.com/pillow>

Не забывайте использовать ссылки и искать ответы на возникающие вопросы.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Загрузите новое чёрно-белое изображение. Преобразуйте черно-белую фотографию в цветную (используйте справочные материалы) и сохраните ее на диске под другим именем. Получившуюся цветную фотографию поверните на 90°, измените ее размеры.
2. Изучите справочные материалы и найдите ответ на вопрос: какие еще библиотеки, кроме Pillow, можно использовать для работы с графикой в Python?

3. С помощью справочных материалов выясните, какие форматы (BMP, EPS, GIF, IM, JPEG, MSP, PCX PNG, PPM, TIFF, WebP, ICO, PSD, PDF) возможны только для чтения, а какие доступны и для редактирования?

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Какие методы мы использовали для работы с изображением? Какие действия с их помощью выполняли?
2. В каких случаях (при создании программы) Вам может понадобиться использование функций изменения цветности и формата изображения?

§1.4 Графические эффекты и фильтры

ВСПОМИНАЕМ

Давайте вспомним:

1. Какие элементы библиотеки Pillow вы уже опробовали? Для чего их используют?
2. Как можно действовать, если вы столкнулись с задачей по обработке изображения, которую не разбирали ранее?

ЧИТАЕМ

Вы уже овладели некоторыми навыками работы с изображением: умеете его поворачивать, изменять масштаб и формат. Но бывает так, что изображению нужно придать, например, вид наброска или изменить его художественный стиль. Для этих целей в библиотеке Pillow используются фильтры. С помощью фильтров можно также очистить или отретушировать фотографию, провести ее трансформацию с использованием эффектов искажения и освещения.

ГОТОВИМСЯ К РАБОТЕ НА ПК

Библиотека Pillow предоставляет следующий набор фильтров для улучшения изображения:

- BLUR (размытие)
- CONTOUR (контур)
- DETAIL (деталь)
- EDGE_ENHANCE (улучшение края)
- EDGE_ENHANCE_MORE (большее улучшения края)

- EMBOSS (тиснение)
- FIND_EDGES (найти края)
- SMOOTH (гладкий)
- SMOOTH_MORE (более гладкий)
- SHARPEN (резкость)

Для применения того или другого фильтра нужно подключить модуль **ImageFilter**.

Кроме того, для увеличения резкости фотографий в Python можно использовать модуль **ImageEnhance**.

Чтобы открыть изображение, мы используем метод `show()`. Если это не срабатывает, то сначала установите модуль `ImageMagick`, затем попробуйте снова.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Рассмотрим применение некоторых фильтров на следующей фотографии (рис. 10)



Рис. 10 Оригинал

ПРИМЕР 1. Загрузим фото с именем **flower.jpg** с диска и размоем его с помощью фильтра **BLUR**.

```
# импортируем необходимые модули
from PIL import Image, ImageFilter
# загружаем изображение с диска
original = Image.open("flower.jpg")
# размываем изображение
blurflower = original.filter(ImageFilter.BLUR)
# открываем оригинал и размытое изображение
original.show()
blurflower.show()
# сохраняем изображение
blurflower.save("blurflower.jpg")
```

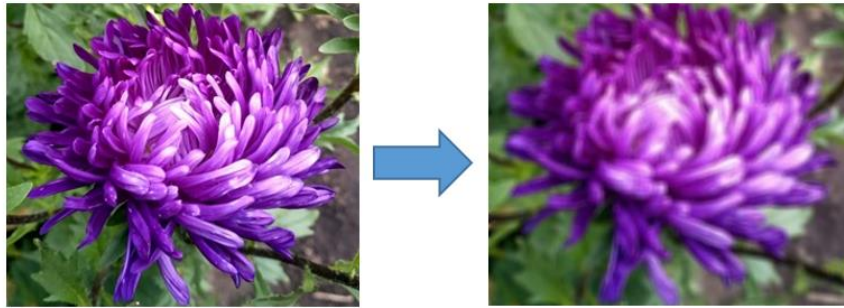


Рис. 11 Результат работы фильтра *BLUR* (размытие)

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 2. Применим к ранее загруженному фото с именем **flower.jpg** фильтр **CONTOUR**.

```
from PIL import Image, ImageFilter
original = Image.open("flower.jpg")
contourflower = original.filter(ImageFilter.CONTOUR)
contourflower.show()
contourflower.save("contourflower.jpg")
```

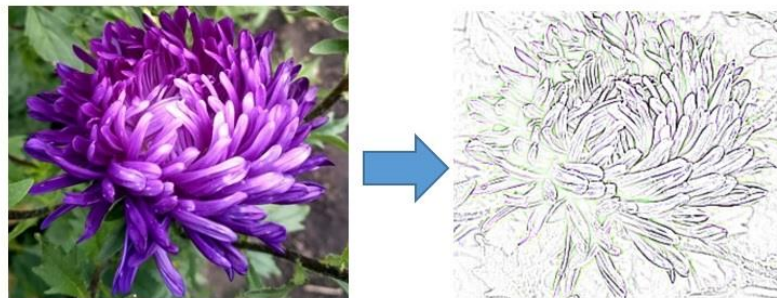


Рис. 12 Результат работы фильтра *CONTOUR* (контур)

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 3. Разместим текст «FLOWER» на ранее загруженном фото с именем **flower.jpg** с помощью модулей **ImageDraw** и **ImageFont**.

```
from PIL import Image, ImageDraw, ImageFont
original = Image.open("flower.jpg")
idraw = ImageDraw.Draw(original)
text = "FLOWER"
font = ImageFont.truetype("arial.ttf", size=24)
idraw.text((20, 10), text, font=font, fill='white')
original.save('flower_text.jpg')
```



Рис. 13 Размещение текста на изображении

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №5

Загрузите цветное изображение в формате PNG. Примените к нему эффект EMBOSS и сохраните новое изображение в формате JPG.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Загрузите цветное изображение, пропорционально измените его размеры так, чтобы ширина была 400 пикселей. Затем дважды выведите его на экран так, чтобы на одном изображении был использован фильтр `EDGE_ENHANCE`, а на другом – фильтр `EDGE_ENHANCE_MORE`.
2. Загрузите цветное изображение. Затем дважды выведите его на экран так, чтобы на одном изображении был использован фильтр `SMOOTH`, а на другом – фильтр `SMOOTH_MORE` и подпишите каждое изображение, чтобы было понятно, какой фильтр применён.
3. С помощью дополнительных источников найдите информацию о модулях ImageDraw, ImageFont.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. В каких случаях может понадобиться применение фильтров к изображениям?
2. Возможно ли применение нескольких фильтров к одному изображению? Все ли фильтры можно удачно сочетать?

§1.5 Рисование графическими примитивами: создание 2D рисунка в Pillow

ВСПОМИНАЕМ

Давайте вспомним:

1. Для чего используются фильтры при обработке фотографии?
2. С какими фильтрами вы уже успели поработать? С какими еще нет?

ЧИТАЕМ

У Pillow есть базовые возможности для создания 2D графики. Модуль ImageDraw предоставляет простую 2D графику для объектов Image. Можно создавать новые изображения, аннотации к ним, а также сразу генерировать графику для Web, например, в виде смайликов.



Рис. 14 Смайлики

ГОТОВИМСЯ К РАБОТЕ НА ПК

Перед тем, как начать создавать свои рисунки, необходимо, используя **объект Image**, создать **фоновое изображение**, на котором и будут рисоваться наши фигуры при помощи **объекта Draw**.

```
# Создается оранжевый фон
img = Image.new('RGBA', (400, 400), 'orange')
# Создается объект, на котором можно рисовать
idraw = ImageDraw.Draw(img)
```

Данное фоновое изображение создается в цветовой модели RGBA. Буква «А» в аббревиатуре означает «Альфа». Ее значение отвечает за величину прозрачности цвета. В отличие от RGB модели, где можно установить необходимое сочетание красного, зеленого и синего цветов, в данной модели мы можем также определять, насколько цвет будет прозрачным.

Фоном также может быть и любое готовое изображение.

```
from PIL import Image, ImageDraw
img = Image.open('IMG.jpg')
idraw = ImageDraw.Draw(img)
```

Перечислим основные методы, с помощью которых можно создавать 2D графику в Pillow Python:

1. **point((x, y), fill='color')** – точка, где:
 - x, y – координаты

- fill='color' – цвет точки
- 2. **line((x1,y1,x2,y2), fill='color', width=<значение>)** – линия, где:
 - x1, y1, x2, y2 – координаты начала и конца
 - fill='color' – цвет линии
 - width=<значение> – толщина линии
- 3. **rectangle((x1,y1,x2,y2), fill='color', outline='color', width=<значение>)** – прямоугольник, где:
 - x1, y1, x2, y2 – координаты диагонали
 - fill='color' – цвет заливки
 - outline='color' – цвет контура
 - width=<значение> – толщина контура
- 4. **ellipse((x1,x2,y1,y2), fill='color', outline='color', width=<значение>)** – эллипс, где:
 - x1, x2, y1, y2 – координаты, отвечающие за диаметры эллипса по оси X и по оси Y
 - fill='color' – цвет заливки
 - outline='color' – цвет контура
 - width=<значение> – толщина контура
- 5. **arc((x1,x2,y1,y2), start, end, fill='color', width=<значение>)** – дуга, где:
 - x1, x2, y1, y2 – координаты, отвечающие за диаметры дуги по оси X и по оси Y
 - параметры start и end – это указание угла дуги в градусах
 - fill='color' – цвет контура
 - width=<значение> – толщина контура
- 6. **chord((x1,x2,y1,y2), start, end, fill='color',outline='color', width=<значение>)** – хорда, где начальные и конечные точки дуги связываются прямой линией
- 7. **pieslice((x1,x2,y1,y2), start, end, fill='color',outline='color', width=<значение>)** – пирог, где начальные и конечные точки дуги связываются прямой линией с центром круга
- 8. **polygon([(x1,y1),(x2,y2),...], fill='color',outline='color', width=<значение>)** – многоугольник
- 9. **text((x,y),'TEXT', fill='color')** – вывод текста

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Пример 1

Условие:

Используя графические методы библиотеки Pillow, создайте на экране изображение точки, линии, квадрата.

Решение:

```
# Рисуем точку
idraw.point((300,30), fill='black')
# Рисуем линии
idraw.line((110, 210, 350, 350), fill="gray", width=3)
idraw.line(xy=((80, 200),(180, 100),(130, 50)), fill='cyan',
width=10)
# Рисуем квадрат
idraw.rectangle((200, 150, 300, 250),
fill='blue',outline='yellow', width=5)
```

Результат:

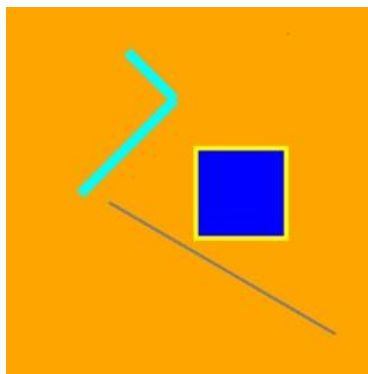


Рис. 15 Изображение точки, линий, квадрата

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Пример 2

Условие:

Используя графические методы библиотеки Pillow, создайте на экране изображение окружности, эллипса, дуги, хорды и пирога.

Решение:

```
# Рисуем окружность и эллипс
idraw.ellipse((0,0,200,200), fill='red',outline='green')
idraw.ellipse((300,100,330,300), fill='yellow',outline='green')
# Рисуем дугу
box=(50, 300, 150, 350)
idraw.arc(box, 30, 180, fill='magenta', width=3)
# Рисуем хорду
idraw.chord(box, 180, 340, fill='magenta', outline=(140,30,50),
width=3)
# Рисуем пирог
idraw.pieslice((100,350,300,390), 80, 340, fill="#FFFF00",
outline=(140,30,150), width=3)
```

Результат:

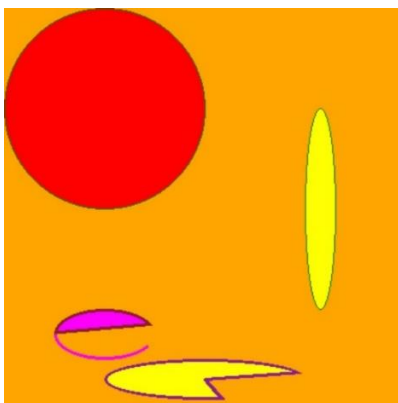


Рис. 16 Изображение окружности, эллипса, дуги, хорды, пирога

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Пример 3

Условие:

Используя графические методы библиотеки Pillow, создайте на экране изображение многоугольника с выводом текста.

Решение:

```
# Рисуем многоугольник
idraw.polygon([(150,150), (250,150), (250,200), (150,250)],
fill="green", outline=None)
# Выводим текст
idraw.text((300, 145), 'POLIGON', fill="green")
img.show()
```

Результат:

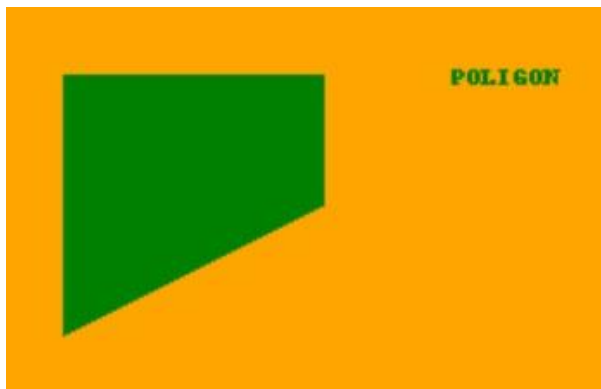


Рис. 17 Изображение многоугольника и вывод текста

ЧИТАЕМ

Параметры ('color') и (<значение>) по умолчанию None (не заполнено).

Заметим, что есть три способа указания цвета:

- текстовый формат: red, green, blue и др.;
- CSS формат (Шестнадцатеричный): "#FF0000", "#00FF00", "#AA6B1C" и др.;
- RGB: (255, 0, 0), (0, 0, 0), (255, 255, 255) и др.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №6

Нарисуйте один из смайликов (см. рис. 14) и сохраните изображение на диске с расширением .png.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №7

Нарисуйте плакат с изображением трех любых геометрических фигур (см. рис. 18)

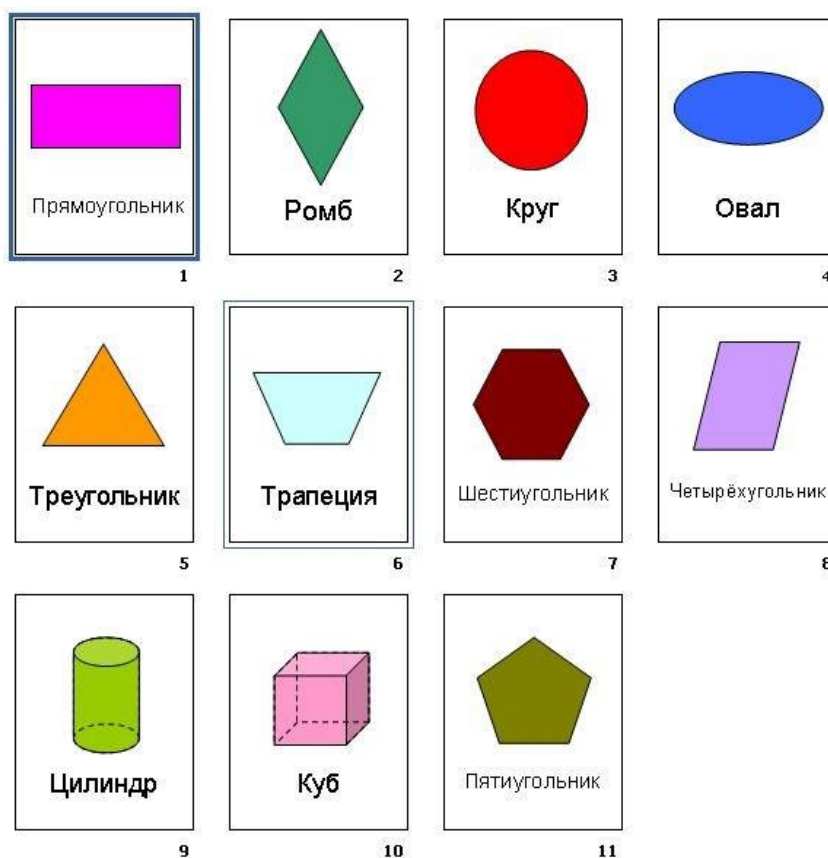


Рис. 18 Геометрические фигуры для задания

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. С помощью дополнительных источников информации найдите, какие еще есть возможности для создания различных эффектов в 2D рисунках.
2. С помощью дополнительных источников информации найдите, какие еще возможности можно использовать для вывода текста на изображении.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. С какими базовыми возможностями для создания и редактирования 2D графики вы познакомились?
2. Какие шаги необходимо предпринять перед созданием рисунков?
3. С какими методами для создания и редактирования 2D графики вы уже познакомились? С какими планируете познакомиться?

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. Рисуем геометрические фигуры в Python с помощью Pillow – <https://python-scripts.com/draw-circle-rectangle-line>
2. 10 инструментов Python для работы с изображениями – <https://medium.com/nuances-of-programming/10-%D0%B8%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%B2-python-%D0%B4%D0%BB%D1%8F-%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%8B-%D1%81-%D0%B8%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%D0%BC%D0%B8-1478612e7ade>
3. Параллельная обработка изображений – <http://onreader.mdl.ru/MasteringConcurrencyInPython/content/Ch08.html>

§1.6 Рисование графическими примитивами: циклы и графика

ВСПОМИНАЕМ

Давайте вспомним:

1. Какие объекты и модули нам нужны для создания и редактирования 2D рисунка?
2. Что называют графическими примитивами? Перечислите их.

ЧИТАЕМ

На предыдущем уроке мы научились создавать графическое изображение с помощью графических элементов (точки, прямой линии, эллипса, прямоугольника и др.).

Но достаточно часто встречается ситуация многократного повторения одних и тех же графических элементов:

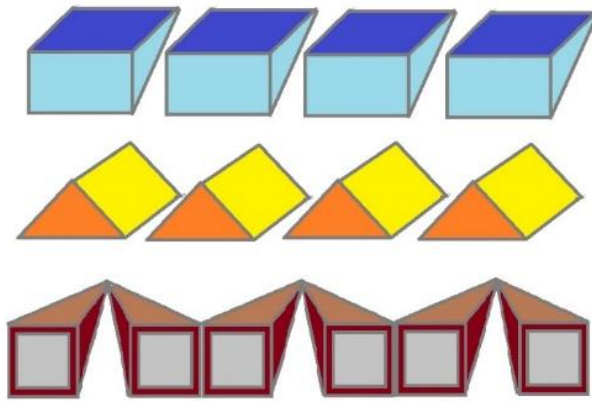


Рис. 19 Повторяющиеся элементы

Из подобных элементов можно, например, составить орнамент.

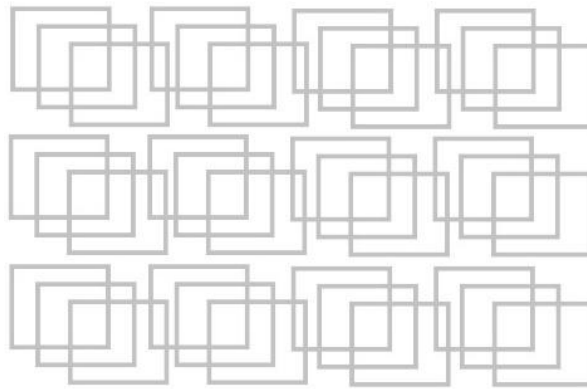


Рис. 20 Орнамент_1

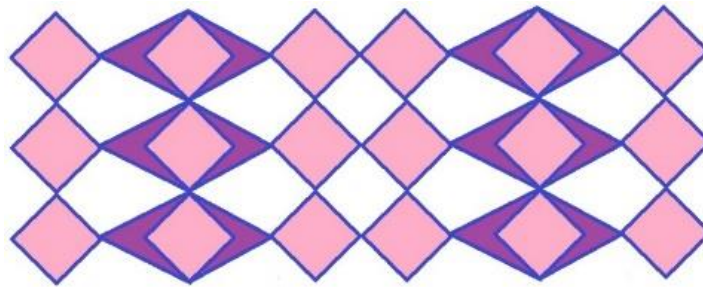


Рис. 21 Орнамент_2

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Пример 1

Условие:

Начните изучение данного вопроса с построения так называемого «звездного неба»: постройте на экране множество разноцветных точек. Для этого нужно строить один и тот же элемент – точку, но постоянно менять ее координаты и цвет.

Решение:

```
from PIL import Image, ImageDraw
import random
img = Image.new('RGBA', (400, 400), 'lightgray')
idraw = ImageDraw.Draw(img)
for i in range(1000):
```

```

x = random.randint(0,400)
y = random.randint(0,400)
idraw.point((x,y),fill=(random.randint(0,255),random.randint
(0,255),
random.randint(0,255)))
img.show()

```

Результат:

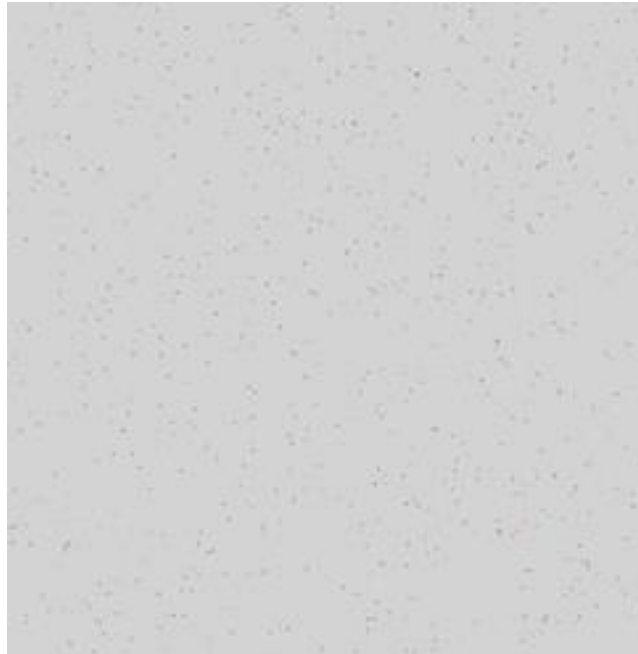


Рис. 22 Результат работы программы

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Пример 2

Условие:

В данном примере нужно построить следующее изображение (см. рис. 23)

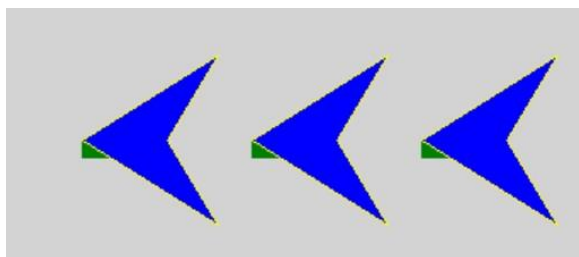


Рис. 23 Заготовка

Решение:

Понимаем, что вначале нужно построить один объект и скопировать его, изменяя некоторые параметры этого изображения. В данном примере мы возьмем начальную координату x_0 первого объекта за начало отсчета и будем рисовать весь объект относительно x_0 . Следовательно, чтобы построить второй объект, нам нужно будет только изменить значение x_0 .

```

from PIL import Image, ImageDraw
img = Image.new('RGBA', (400, 400), 'lightgray')
idraw = ImageDraw.Draw(img)
x0=100
def ptichka (x0):
    idraw.rectangle((x0,200,x0+50,210), fill='green')

idraw.polygon([(x0,200), (x0+80,150), (x0+50,200), (x0+80,250)],
    fill='blue',outline='yellow')
for i in range(3):
    ptichka(x0)
    x0 = x0 + 100
img.show()

```

Посмотрите внимательно. Данный код можно отредактировать и получить простую анимацию. Для этого достаточно по очереди выводить объекты на экран и стирать их.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №8

Напишите две программы, которые выполняют следующие задачи:

Первая: построение «мыльных пузырей». Иначе говоря, вывод на экран множества разноцветных окружностей разного диаметра.

Вторая: построение узора из 10 вложенных друг в друга прямоугольников или окружностей (на выбор).

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Напишите программу для рисования любой из повторяющихся фигур (см. рис. 19).
2. Напишите программу для рисования одного из орнаментов (см. рис. 20, рис. 21).
3. Придумайте свой повторяющийся рисунок и напишите для него программный код.
4. Придумайте своего персонажа и создайте с ним анимацию.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Приведите примеры многократного повторения одних и тех же элементов в жизни, науке, искусстве и др.
2. Сформулируйте алгоритм действий для создания 2D анимации с помощью графических примитивов.

§1.7 Фрактальная графика

ВСПОМИНАЕМ

Давайте вспомним:

1. Для чего вы использовали циклическую конструкцию при создании графических объектов?
2. Какие особенности многократного воспроизведения объектов нужно помнить?

ЧИТАЕМ

Иногда для построения изображения требуется многократно повторить малый объект того же типа. Этот приём часто применяется в компьютерной графике для построения изображений природных объектов, таких как деревья, кусты, горные ландшафты, поверхности морей и так далее. Такой вид графики, при котором самоподобные фигуры повторяются конечное число раз, называется **фрактальной графикой**.

Для реализации фрактальной графики используется такой приём как рекурсия. **Рекурсия** – это определение объекта через такой же объект, только с другими параметрами. Чтобы определить рекурсивный объект, необходимо выделить базовый объект и задать правило, по которому новый объект строится из уже известных объектов.

Этот приём используется не только для графики. Разберём пример рекурсивной функции.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Пример 1

Условие:

Необходимо вычислить значение заданной рекурсивной функции на 100 шаге итерации:

$$F(n) = 1, \text{ при } n = 1$$

$$F(n) = n + 1 + F(n-1), \text{ при } n > 1.$$

Решение:

В данной задаче понятен и базовый объект и правило, по которому находятся все последующие объекты.

```
def F( n ) :  
    if n == 1:  
        return 1  
    else:  
        return n + 1 + F(n-1)
```

Выведем значение функции при $n = 100$.

```
print (F(100))
```

Результат:

5149

Можно было, конечно, эту задачу просчитать вручную. Но мы понимаем, что решение рекурсивных алгоритмов эффективнее и быстрее выполнять с помощью компьютера.

ЧИТАЕМ

Основой изображения, выполненного с использованием фрактальной графики, является **фрактал**. Фрактал – это геометрическая фигура, составленная из меньших фигур того же типа, т.е. обладающая самоподобием. Фрактал реализуется с помощью рекурсии. Наиболее наглядно рекурсия проявляется во фрактальных изображениях, где придется самостоятельно находить и базовый объект, и правило построения следующего элемента.

К известным фрактальным изображениям относятся дерево Пифагора (ветка), снежинка, кривая Коха, снежинка Коха, множество Кантора, треугольник Серпинского и др.

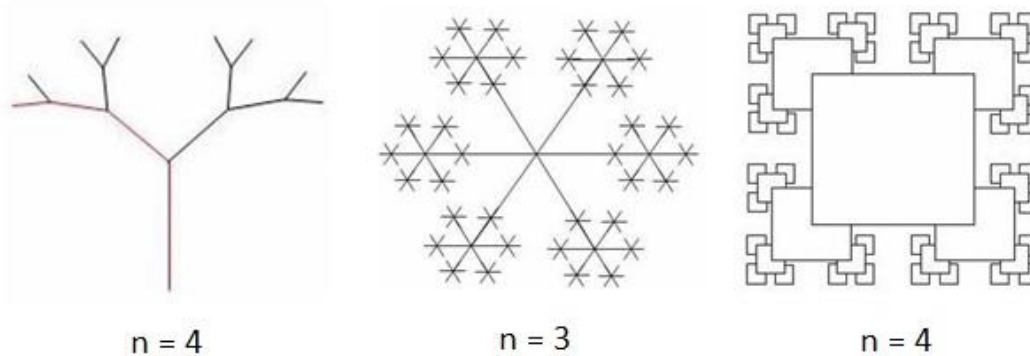


Рис. 24 Ветка, снежинка, множество Кантора

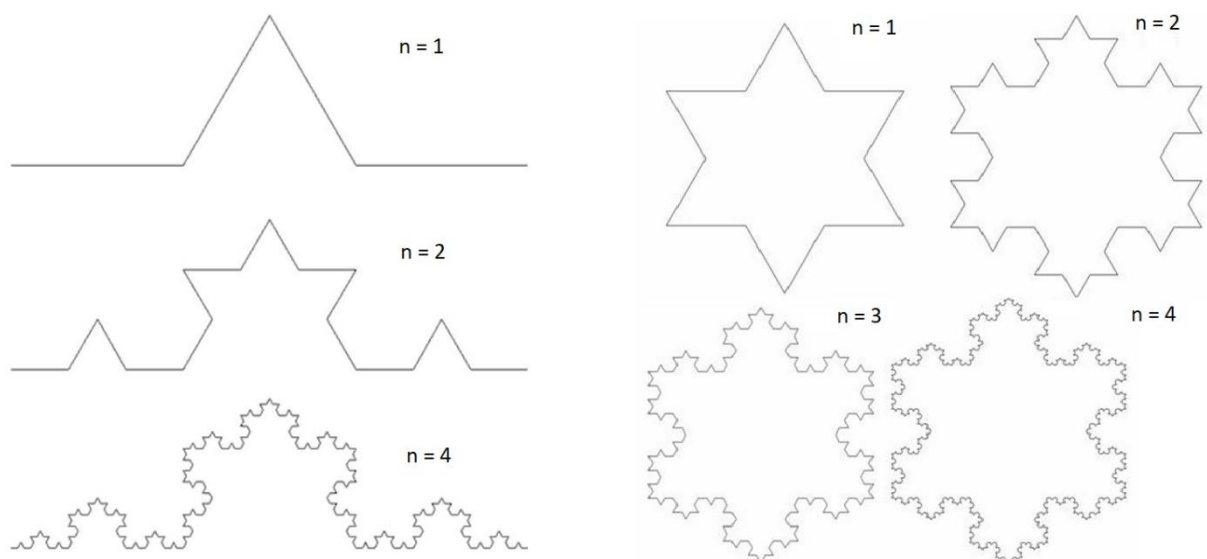


Рис. 25 Кривая Коха, снежинка Коха

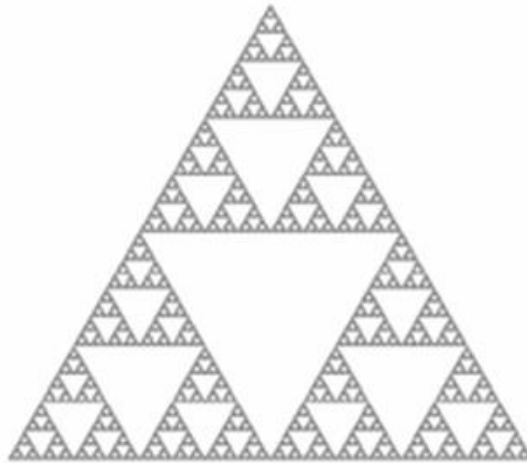


Рис. 26 Треугольник Серпинского

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Пример 2

Условие:

Построить треугольник Серпинского.

Рассуждение:

Определим базовый объект – это “перевернутый” треугольник, построенный внутри “прямого” треугольника, вершины которого лежат на серединах сторон “прямого” треугольника. В результате первой итерации наш “прямой” треугольник разбивается на три составляющие.

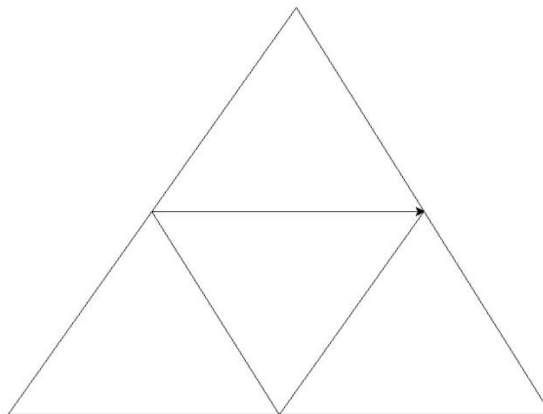


Рис. 27 Базовый объект в треугольнике Серпинского

На следующих итерациях нам нужно найти середины этих “прямых” треугольников и построить в них “перевернутые” треугольники, только с другими координатами.

Итак, мы определили рекурсивный объект и задали правило, по которому новый объект строится из уже известных объектов.

Решение 1 (с помощью turtle (черепашки) с количеством итераций $n = 3$)

```
import turtle
turtle.reset()
```

```

turtle.speed(10)
def TRY(x1, y1, x2, y2, x3, y3, N):
    if N > 0:
        x12 = (x1 + x2) // 2
        y12 = (y1 + y2) // 2
        x23 = (x2 + x3) // 2
        y23 = (y2 + y3) // 2
        x31 = (x3 + x1) // 2
        y31 = (y3 + y1) // 2
        turtle.pu()
        turtle.goto(x31, y31)
        turtle.pd()
        turtle.goto(x12, y12)
        turtle.goto(x23, y23)
        turtle.goto(x31, y31)
        TRY(x1, y1, x12, y12, x31, y31, N - 1);
        TRY(x2, y2, x12, y12, x23, y23, N - 1);
        TRY(x3, y3, x31, y31, x23, y23, N - 1)
x1 = 0
y1 = 200
x2 = -339
y2 = -279
x3 = 300
y3 = -279
n = 3
turtle.pu()
turtle.goto(x1, y1)
turtle.pd()
turtle.goto(x2, y2)
turtle.goto(x3, y3)
turtle.goto(x1, y1)
TRY(x1, y1, x2, y2, x3, y3, n);
turtle.exitonclick()

```

Результат:

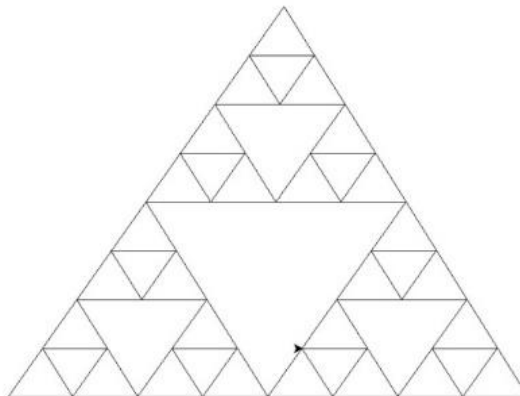
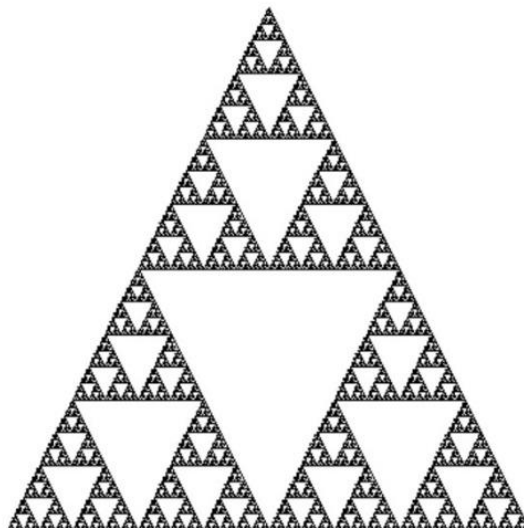


Рис. 28 Результат работы программы с помощью turtle

Решение 2 (с помощью графических методов с количеством итераций n
= 7)

```
from PIL import Image, ImageDraw
img = Image.new('RGBA', (400, 400), 'white')
idraw = ImageDraw.Draw(img)
def TRY(x1, y1, x2, y2, x3, y3, N):
    if N <= 0: return
    x12 = (x1 + x2) // 2
    y12 = (y1 + y2) // 2
    x23 = (x2 + x3) // 2
    y23 = (y2 + y3) // 2
    x31 = (x3 + x1) // 2
    y31 = (y3 + y1) // 2
    idraw.polygon([(x31, y31), (x12, y12), (x23,
y23)], fill=None, outline='black')
    TRY(x1, y1, x12, y12, x31, y31, N - 1);
    TRY(x2, y2, x12, y12, x23, y23, N - 1);
    TRY(x3, y3, x31, y31, x23, y23, N - 1)
x1 = 200
y1 = 20
x2 = 20
y2 = 380
x3 = 380
y3 = 380
n = 7
idraw.polygon([(x1,y1), (x2,y2), (x3,y3)], fill=None,
outline='black')
TRY(x1, y1, x2, y2, x3, y3, n);
img.show()
```

Результат:



ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №9

С помощью компьютера средствами языка Python определите, чему равно значение функции $F(16)$, если алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$$F(n) = 1 \text{ при } n = 1$$

$$F(n) = 2*n + F(n-1), \text{ если } n \text{ чётно,}$$

$$F(n) = F(n-2)+3, \text{ если } n > 1 \text{ и } n \text{ нечётно.}$$

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №10

Проанализируйте данный программный код:

```
def F( n ) :  
    global s  
    s=s+(3*n)  
    if n > 1:  
        s=s+(n-2)  
        F(n-1)  
        F(n-2)  
n=???  
s=0  
F(n)  
print( n, s )
```

Определите наименьшее значение n , при котором сумма чисел, которые будут выведены при вызове $F(n)$, будет больше 100000. Запишите в ответе сначала найденное значение n , а затем через пробел – соответствующую сумму выведенных чисел.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №11

Определите рекурсивный объект и правило, по которому строится новый объект для следующих фигур:

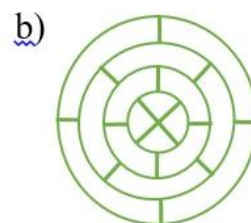
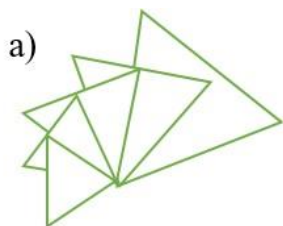


Рис 30 Задание для учащихся

Разработайте алгоритмы и напишите функции для построения фрактальных изображений, изображенных на рис. 30. Исходные данные для функций определите самостоятельно, учитывая, чтобы у пользователя было как можно больше параметров для определения фрактального изображения.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Найдите дополнительную информацию в сети Интернет об известных фрактальных рисунках и напишите к ним программный код, в котором n – число итераций – будет вводиться случайным образом.
2. Придумайте свой фрактальный узор и напишите к нему программный код.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Что такое фрактальное изображение? Как вы это понимаете?
2. Что такое рекурсия? Как рекурсия связана с фракталами?
3. Перечислите основные шаги для решения любой рекурсивной задачи.

§1.8 Научная графика, типы диаграмм

ВСПОМИНАЕМ

Перед началом занятия давайте вспомним:

1. Что такое фрактальная графика?
2. Что относится к известным фрактальным изображениям?
3. Как зависит фрактальный узор от числа итераций?

ЧИТАЕМ

К научной графике относятся таблицы, схемы, диаграммы, иллюстрации, чертежи. Их мы используем при создании иллюстраций, при подготовке отчетной документации, статистических данных и др. Это можно сделать в программах пакета **Office** или **Google Docs**. Но если диаграммы нужно строить регулярно, значит, предпочтительнее делать это автоматически. Вот тут-то на помощь и приходит **Python** с его потрясающей библиотекой **Matplotlib**.

Библиотека поддерживает двумерную (2D) и трехмерную (3D) графику, а также анимированные рисунки.

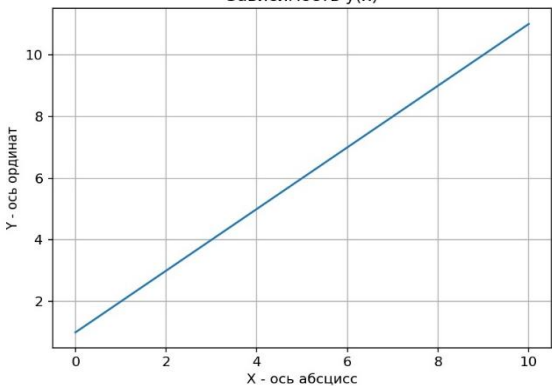
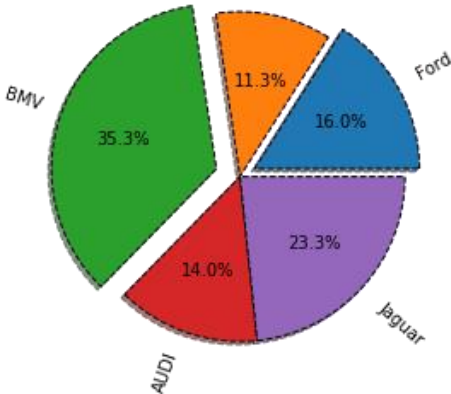
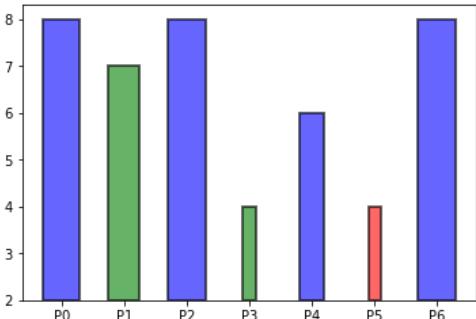
Все функции построения диаграмм ожидают аргументы следующих типов: **np.array**, **np.ma.masked_array** – набор или массив значений из пакета

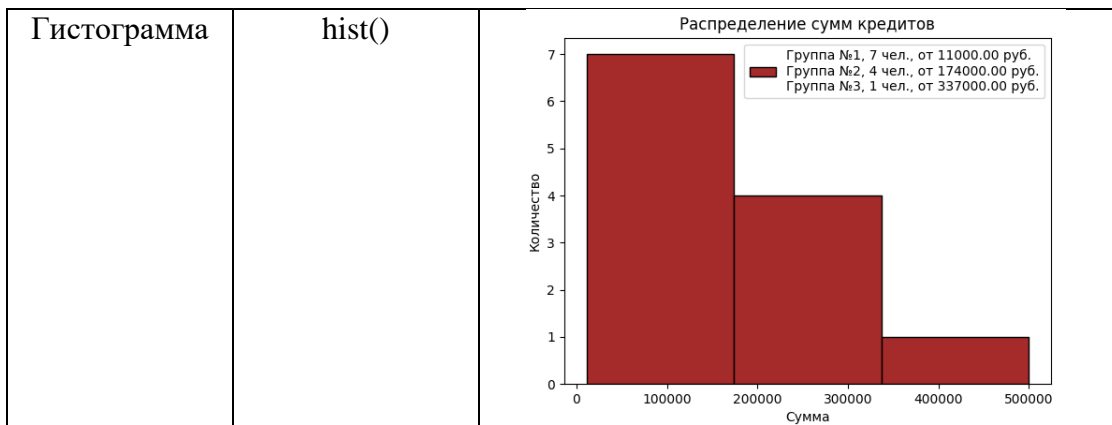
NumPy (устанавливается вместе с **matplotlib**). **NumPy** – фундаментальный пакет для научных вычислений на **Python**.

Пакет поддерживает многие виды диаграмм:

- графики;
- диаграммы разброса;
- столбчатые диаграммы и гистограммы;
- круговые диаграммы;
- ствол-лист диаграммы;
- контурные графики;
- поля градиентов;
- спектральные диаграммы
- и др.

Таблица 2 Основные типы диаграмм

| Тип | Метод | Пример |
|----------------------|------------------|--|
| График | plot() |  <p>Зависимость $y(x)$</p> |
| Круговая диаграмма | pie() |  |
| Столбчатая диаграмма | bar(), barh() |  |



При построении можно указать оси координат, сетку, добавить аннотации, использовать логарифмическую шкалу или полярные координаты. Созданные изображения могут быть легко сохранены, в частности, в популярные форматы (JPEG, PNG и др.).

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Задача:

Построить график функции $y = x + 1$.

Порядок выполнения:

Подключаем необходимые библиотеки:

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
```

Определяем зависимость $y(x)$:

```
x = np.linspace(0, 10, 50)
y = x + 1
```

Строим график и выводим его на экран:

```
plt.title("Зависимость y(x)") # заголовок
plt.xlabel("X - ось абсцисс") # подпись к оси абсцисс
plt.ylabel("Y - ось ординат") # подпись к оси ординат
plt.grid() # включение отображения сетки
plt.plot(x, y) # построение графика
plt.show()
```

Результат выполнения:

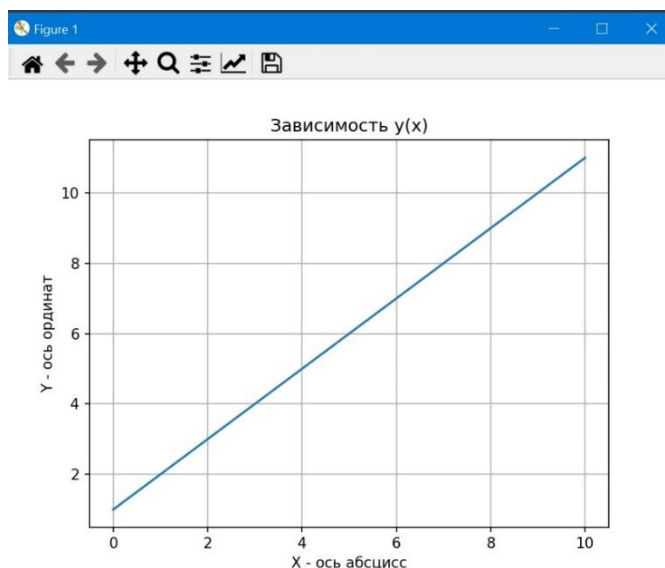


Рис. 31 Построение графика функции $y = x + 1$

ЧИТАЕМ

Обратите внимание, что кроме диаграммы графическое окно, которое открылось при выполнении программы, разобранной в примере №1, содержит специальные функциональные кнопки – интерактивную навигацию (Рис. 32),

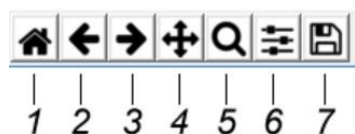


Рис. 32 Специальные функциональные кнопки

а их назначение и возможности приведены в таблице 3.

Таблица 3 Интерактивная навигация графического окна

| Номер | Назначение |
|-------|--|
| 1 | Возвращает график к первоначальному состоянию (вид при первом запуске) |
| 2, 3 | Перемещает назад/вперед по истории изменения графика |
| 4 | Перетаскивает (левая кнопка мыши) или масштабирует (правая кнопка мыши) график |
| 5 | Масштабирует заданную прямоугольную область графика |
| 6 | Открывает дополнительные настройки окна |
| 7 | Вызывает диалоговое окно сохранения графика в файл |

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 2

Условие:

Построить одновременно два графика: $y = x + 1$ и $y = x^2$.

Возможное решение:

```
# Подключение библиотек
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
# Независимая (x) и зависимые (y) переменные
x = np.linspace(-5, 5, 50)
y1 = x+1
y2 = x**2
# Построение графика
plt.title("Зависимость y(x)") # заголовок
plt.xlabel("X - ось абсцисс") # ось абсцисс
plt.ylabel("Y - ось ординат") # ось ординат
plt.grid() # включение отображения сетки
plt.plot(x, y1) # построение графика 1
plt.plot(x, y2) # построение графика 2
plt.show()
```

Результат выполнения:

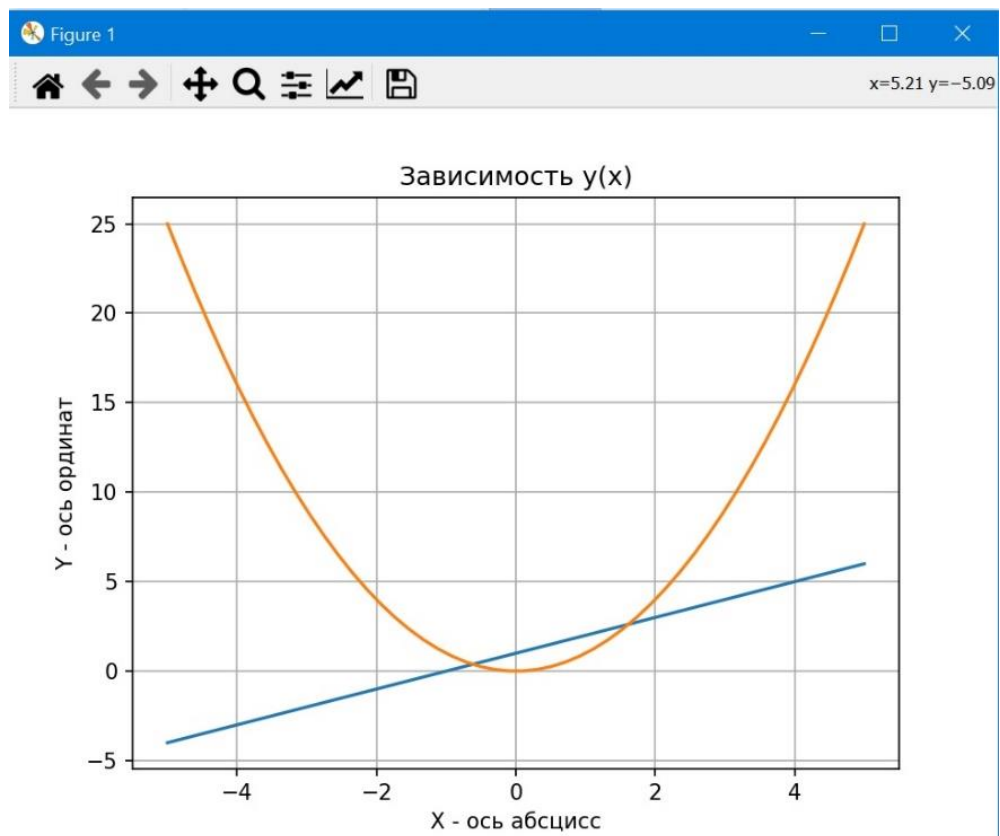


Рис. 33 Построение графиков функций $y = x+1$ и $y=x^2$

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 3

Условие:

Построить графики функций $y = x+1$ и $y=x^2$ в разных координатных плоскостях.

Возможное решение:

```
# Подключение библиотек
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

# Линейная зависимость
x = np.linspace(-10, 10, 50)
y1 = x+1

# Квадратичная зависимость
y2 = [i**2 for i in x]

# Построение графиков
plt.figure(figsize=(8, 8))
plt.subplot(2, 2, 1)
plt.plot(x, y1, color='red')          # построение графика
plt.title("Зависимость:  $y_1 = x+1$ ", loc='left') # заголовок 1
plt.ylabel("y1", fontsize=12)        # ось ординат
# включение отображения сетки
plt.grid(which="major", linestyle="-", color="gray",
          linewidth=1)
plt.grid(which="minor", linestyle="--", color="gray",
          linewidth=0.5)
plt.minorticks_on()
plt.subplot(2, 2, 2)
plt.plot(x, y2)                      # построение графика
plt.title("Зависимость:  $y_2 = x^2$ ", loc='right') # заголовок 2
plt.xlabel("x", fontsize=14)         # ось абсцисс
plt.ylabel("y2", fontsize=14)       # ось ординат
plt.grid(True)                      # включение отображения сетки
plt.show()
```

Результат выполнения:

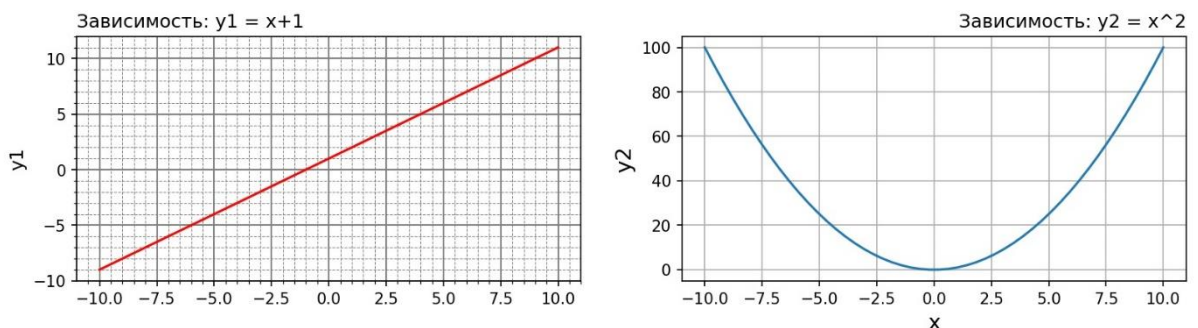


Рис. 34 Графики в разных координатных плоскостях

ЧИТАЕМ

Опираясь на разобранные примеры, мы можем говорить, что типичный ход построения изображения включает следующие этапы:

- 1) Создать объект **Figure**.
- 2) Используя объект **Figure**, добавить координатную плоскость (одну или несколько).
- 3) Добавить графические примитивы.
- 4) Отобразить и/или сохранить изображение.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №12

Выполните построение двух графиков $y = x/2$ и $y = 5 - x^2$.

Постройте эти графики сначала на одной плоскости, а затем на двух разных.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Используя дополнительные источники, найдите информацию о возможностях библиотеки Matplotlib: методах Figure и Axes.
2. Проанализируйте код программы и ее результат работы.

```
import matplotlib.pyplot as plt
from matplotlib.ticker import (MultipleLocator,
FormatStrFormatter, AutoMinorLocator)
import numpy as np
import math
x = np.linspace(0, 5, 50)
y1 = 2*x
y2 = x**2
fig, ax = plt.subplots(figsize=(8, 6))
ax.set_title("Графики зависимостей", fontsize=18)
ax.set_xlabel("Ось ординат", fontsize=14)
ax.set_ylabel("Ось абсцисс", fontsize=14)
ax.grid(which="major", linewidth=1.2)
ax.grid(which="minor", linestyle="--", color="gray",
linewidth=0.5)
ax.scatter(x, y1, c="red", label="y1 = 2*x")
ax.plot(x, y2, label="y2 = x^2")
ax.legend()
ax.xaxis.set_minor_locator(AutoMinorLocator())
ax.yaxis.set_minor_locator(AutoMinorLocator())
ax.tick_params(which='major', length=10, width=2)
ax.tick_params(which='minor', length=5, width=1)
plt.show()
```

Соотнесите основные термины и понятия с элементами изображения, отмеченными числами на данном изображении (см. рис 35).

- a) Дополнительные тики (minor ticks)
- b) Фигура (Figure)
- c) Линейный график (plot)
- d) Заголовок (title)
- e) Основная сетка (major grid)
- f) Легенда (legend)
- g) Подпись оси Y (y label)
- h) Дополнительная сетка (minor grid)
- i) Подпись оси X (x label)
- j) Основные тики (major ticks)
- k) Точечный график (scatter)

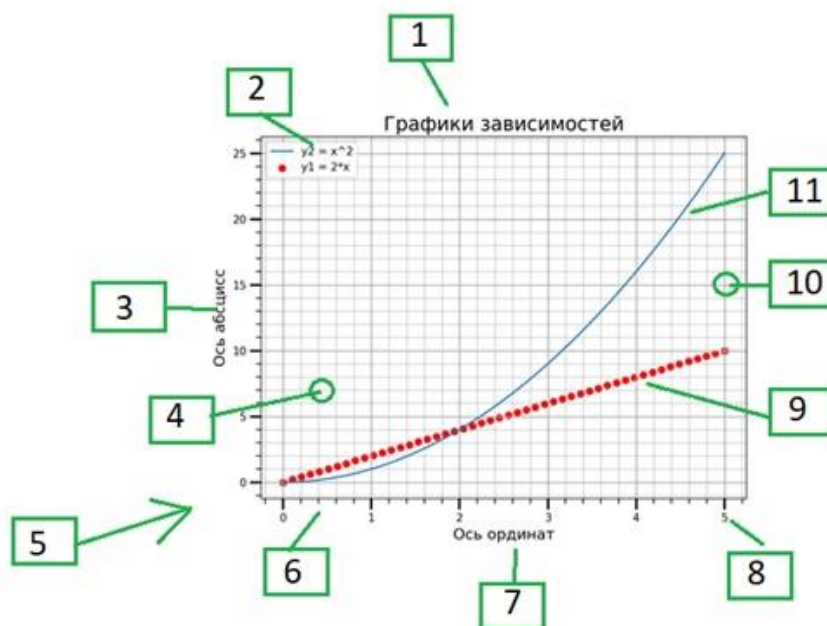


Рис. 35 Основные термины и понятия

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Какие изображения можно отнести к научной графике?
2. С помощью какой библиотеки можно создавать диаграммы и графики в Python?
3. Какие виды диаграмм поддерживает пакет NumPy?

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

- 1) Python – официальный сайт. URL: <https://www.python.org/>
- 2) Python – FAQ. URL: <https://docs.python.org/3/faq/programming.html>

- 3) Саммерфилд М. Программирование на Python 3. Подробное руководство. — М.: Символ-Плюс, 2009. — 608 с.: ISBN: 978-5-93286-161-5.
- 4) Лучано Рамальо. Python. К вершинам мастерства. — М.: ДМК Пресс, 2016. — 768 с.: ISBN: 978-5-97060-384-0, 978-1-491-94600-8.
- 5) Уроки работы в Matplotlib. URL: <https://devpractice.ru/matplotlib-lesson-1-quick-start-guide/>

§1.9 Графики простых функциональных зависимостей

ВСПОМИНАЕМ

Вспомним, что мы узнали о построении элементов деловой графики в Python:

1. Какие этапы включает в себя обычный ход построения изображения?
2. Какие библиотеки необходимо подключить для построения графиков функций?

ЧИТАЕМ

Графики являются одним из самых простых способов отображения зависимостей в графическом виде и могут быть нарисованы с использованием метода **Axes.plot()**.

Библиотека **Matplotlib** предоставляет огромное количество инструментов для построения различных видов графиков, а также способов для изменения их внешнего вида, т.к. линии графика могут иметь различные стили: тип, толщину, цвет, форму и т.д.

Выделим некоторые атрибуты линии:

- 1) **color** — цвет линии, который может быть задан:
 - именем "green";
 - аббревиатурой "g";
 - в шестнадцатеричном формате "#00FF00";
 - RGB (0, 1, 0) или RGBA (0, 1, 0, 1) кортежем;
 - оттенком серого в виде строки "0.8".

Если не указывать цвет, matplotlib выберет его самостоятельно.

- 2) **label** — наименование линии
- 3) **linestyle** — стиль линии:
 - "solid" — сплошная — “-”;
 - "dashed" — пунктирная — “--”;

- "dotted" – точечная – “.”;
 - "dashdot" – точечно-пунктирная – “-.”;
 - и др.
- 4) **linewidth** – толщина линии (вещественное число).
- 5) **marker** – тип маркера для точки графика (по умолчанию отсутствуют):

- "." – точка;
- "+" – плюс;
- "o" – окружность;
- "*" – звезда;
- и др.

Стили маркеров определяются атрибутами **markeredgecolor**, **markeredgewidth**, **markerfacecolor**, **markerfacecoloralt**, **markersize**.

Текстовые элементы графика включают в себя следующие составляющие:

- заголовок поля (title);
- заголовок фигуры (suprtile);
- подписи осей (xlabel, ylabel);
- тестовый блок на поле графика (text), либо на фигуре (figtext);
- аннотация (annotate) – текст и указатель.

К параметрам текста можно отнести:

- Размер шрифта:

fontsize или **size**: число либо значение из списка: {'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'}.

- Стил шрифта:

fontstyle: значение из списка: {'normal', 'italic', 'oblique'}.

- Толщина шрифта:

fontweight: число в диапазоне от 0 до 1000 либо значение из списка: {'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'}.

- Цвет шрифта:

color: один из доступных способов определения цвета.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Задача:

Построить в одной координатной плоскости несколько графиков разных стилей.

Порядок выполнения:

Задать параметры плоскости с помощью методов `Figure` и `Axes`:

```
fig, ax = plt.subplots(figsize=(10,8))  
fig.canvas.set_window_title("Графики функций")
```

Настроить параметры диаграммы и осей:

```
ax.set_title("Графики функций различных стилей")  
ax.grid(True)  
ax.set_xlim(-10, 10)  
ax.set_ylim(-10, 10)
```

При выводе на экран получится вот такое изображение:

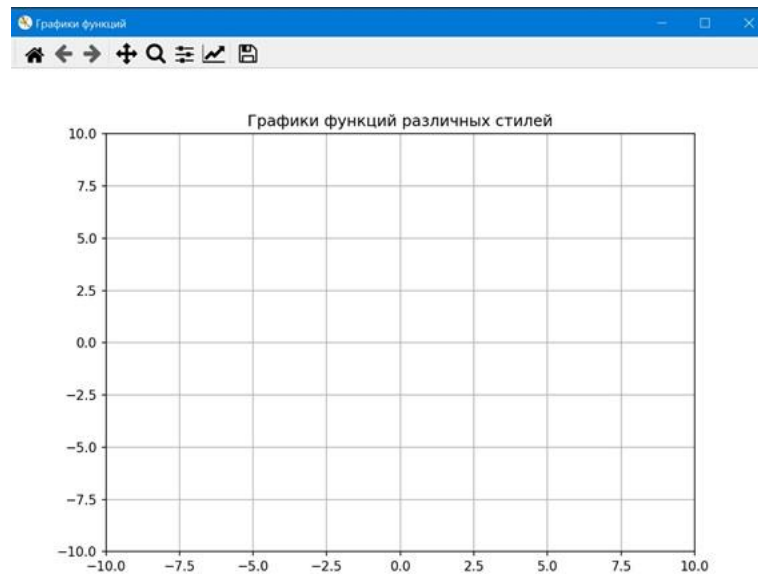


Рис. 36 Настройка параметров плоскости

Получилось, что осью X является нижняя граница окна, а осью Y – левая граница. Переместим оси в центр и уберем верхнюю границу и правую:

```
ax.spines["left"].set_position("center")  
ax.spines["bottom"].set_position("center")  
ax.spines['top'].set_visible(False)  
ax.spines['right'].set_visible(False)
```

Новый результат выглядит так:

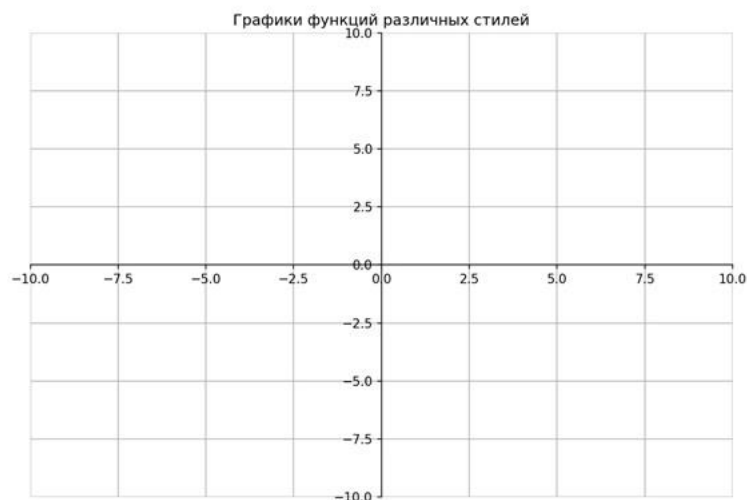


Рис. 37 Настройка осей

Задать диапазон значений для X от -10 до 10 с шагом 0,2:

```
x = np.arange(-10, 10, 0.2)
```

И построить графики функций $y = |-x|$, $y = x^2$, $y = x - 4$ и $y = -5$:

```
ax.plot(x, abs(-x), label="Розовая сплошная линия",
color="pink", linewidth=3)
ax.plot(x, x, label="Фиолетовая пунктирная
линия",color="#D490CC", linewidth=3, linestyle="--")
ax.plot(x, x - 4, label="Линия с точками-маркерами *",
marker="*")
ax.plot([-5] * len(x), x, label="Точки-маркеры
o",linestyle="none", marker="o", markersize=2)
```

И, наконец, вывести легенду так, что программа сама подбирает наилучшее ее расположение относительно всех графиков:

```
ax.legend(loc="best")
```

Результат выполнения:

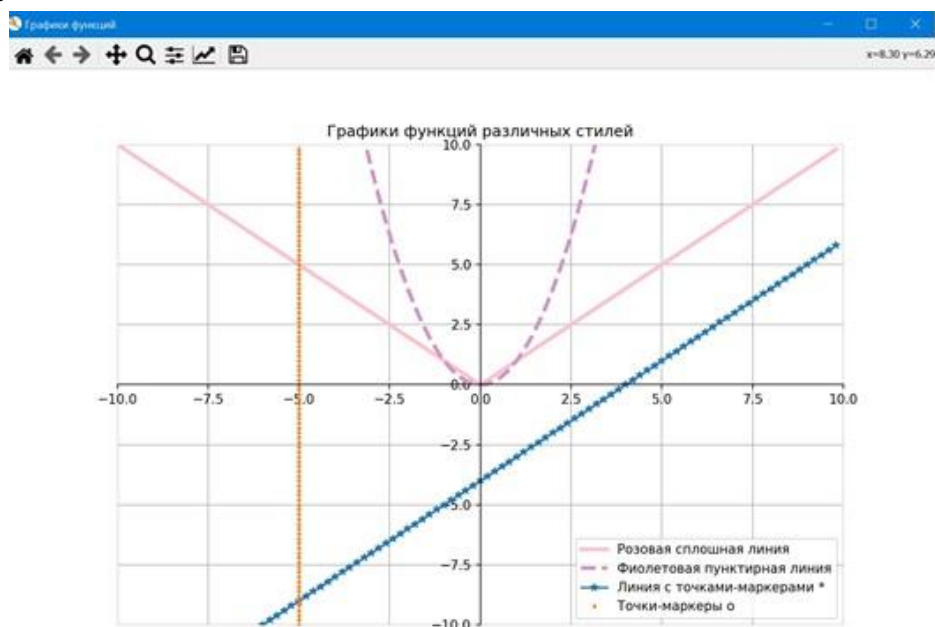


Рис. 38 Графики функций различных стилей

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 2

Задача: Построить график функции $y = \sin(x)$ и залить области между графиком и осью.

Подготовка к решению:

Для заливки областей используется следующая функция:

fill_between(x, y1, y2=0, where=None, interpolate=False), где

x – массив длины N

y1, y2 – массивы длины N или скалярное значение

where – массив bool элементов (длины N), optional, значение по умолчанию: None – задает заливаемый цветом регион, который определяется координатами $x[where]$: интервал будет залит между $x[i]$ и $x[i+1]$, если $where[i]$ и $where[i+1]$ равны True.

Возможное решение:

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(6,4))
fig.canvas.set_window_title("Графики функций")
# Настройки диаграммы и осей
ax.set_title("Заливка областей графиков функций")
ax.grid(True)
ax.set_xlim(-5, 5)
ax.set_ylim(-2, 2)
ax.spines["left"].set_position("center")
ax.spines["bottom"].set_position("center")
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
x = np.arange(-4, 4, 0.01)
y = np.sin(x*np.pi)
plt.plot(x, y, c = "r")
plt.fill_between(x, y)
plt.show()
```

Результат выполнения:

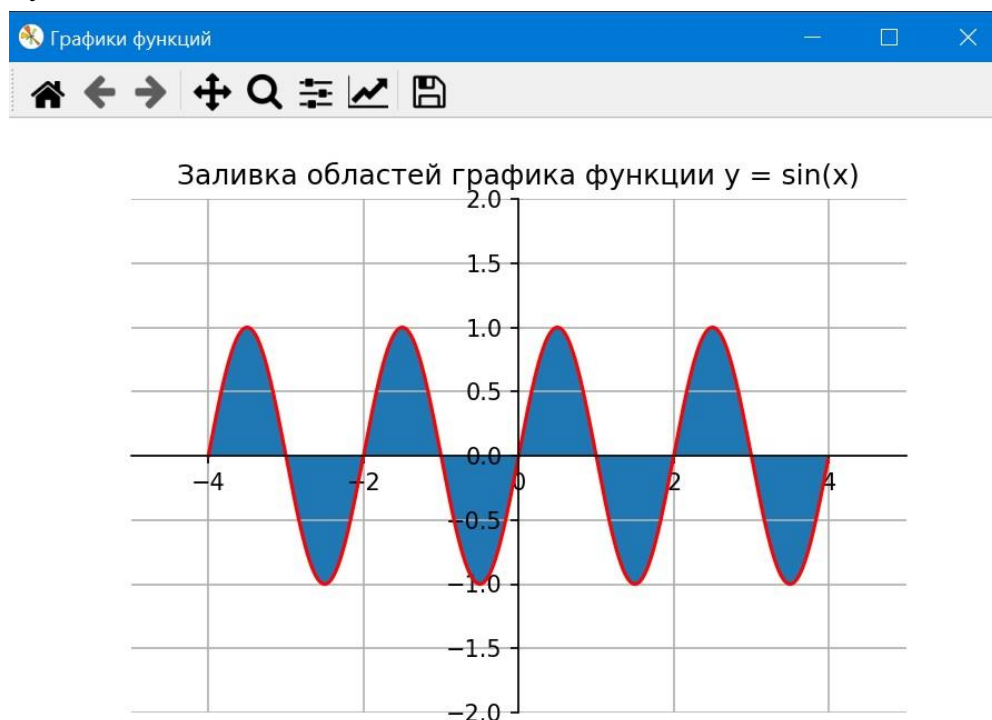


Рис. 39 Заливка областей графика функции

В зависимости от задачи мы можем добавить условие заливки в функцию `fill_between`:

```
plt.fill_between(x, y, where = (y > 0.8) | (y < -0.5))
```

И получить результат:



Рис. 40 Заливка областей графика функции с условием

Также можем задать условие для двуцветной заливки:

```
plt.fill_between(x,y, where = (y > 0),  
                color="green",alpha=0.6)  
plt.fill_between(x,y, where = (y < 0),color="yellow")
```

Обратите внимание, что, используя параметры y_1 и y_2 , можно формировать более сложные решения.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №13

Постройте одновременно на одном поле четыре графика разных стилей, зная, что:

```
x = [1, 5, 10, 15, 20],  
y1 = [1, 7, 3, 5, 11],  
y2 = y1*1,2 + 1,  
y3 = y2*1,2 + 1,  
y4 = y3*1,2 + 1.
```

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №14

Отредактируйте код программы, созданной при выполнении **Практической работы 13** так, чтобы каждый график выводился в отдельном окне.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Постройте график функции $y = 1/x$. Залейте области так, как показано на рисунке 41:

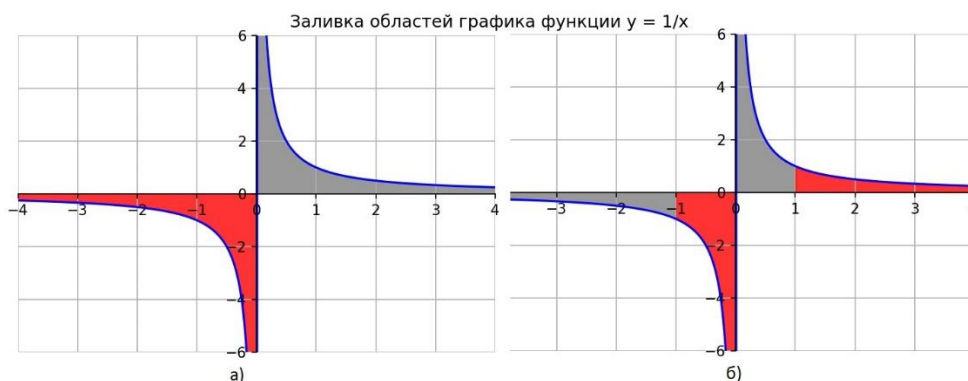


Рис. 41 Заливка областей графика функции $y(x)$

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Какой метод используется для рисования графиков функций?
2. Перечислите известные вам инструменты для построения различных видов графиков.

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. Matplotlib. Урок 2. Работа с инструментом pyplot. URL: <https://devpractice.ru/matplotlib-lesson-2-work-with-pyplot/>
2. Matplotlib. Урок 3. Настройка графиков. Работа с легендой. URL: <https://devpractice.ru/matplotlib-lesson-3-1-work-with-legend/>
3. Matplotlib. Урок 4. Визуализация данных. Линейный график. URL: <https://devpractice.ru/matplotlib-lesson-4-1-viz-linear-chart/>
4. <https://devpractice.ru/matplotlib-lesson-4-1-viz-linear-chart/>
5. “Python. Визуализация данных: Matplotlib, Seaborn, Mayavi”. URL: <https://devpractice.ru/book-python-data-viz/>

§1.10 Столбчатые и круговые диаграммы

ВСПОМИНАЕМ

Давайте повторим:

1. Графики каких стилей можно построить с помощью библиотеки Pillow?
2. Какие параметры плоскости можно настроить с помощью методов Figure и Axes?

ЧИТАЕМ

Столбчатые диаграммы используются для сравнения отдельных характеристик. В диаграммах этого типа категории обычно располагаются на вертикальной оси, а величины – на горизонтальной.

Столбчатая диаграмма может быть нарисована с использованием методов: **bar()** (вертикальная) или **barh()** (горизонтальная).

Рассмотрим некоторые параметры для метода **bar()**:

- **left** (набор числовых значений) – координата x левого края каждого прямоугольника;
- **height** (набор числовых значений) – высота каждого прямоугольника;
- **width** (набор числовых значений) – ширина каждого прямоугольника;
- **bottom** (набор числовых значений) – координата y нижнего края каждого прямоугольника;
- **color** (набор значений) – цвет каждого прямоугольника;
- **tick_label** (строка или набор значений) – подпись по оси OX для каждого прямоугольника.

Укажем также некоторые параметры для метода **barh()**:

- **bottom** (набор числовых значений) – координата y каждого прямоугольника (центр);
- **width** (набор числовых значений) – ширина каждого прямоугольника;
- **height** (набор числовых значений) – высота каждого прямоугольника;
- **left** (набор числовых значений) – координата x правого края каждого прямоугольника;
- **color** (набор значений) – цвет каждого прямоугольника;
- **tick_label** (строка или набор значений) – подпись по оси OY для каждого прямоугольника.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Задача: представить в графическом виде сравнительную характеристику мировых океанов по их площади и средней глубине. Целесообразно это сделать с помощью столбчатых диаграмм для наглядности.

Возможное решение:

```
import matplotlib.pyplot as plt
title = "Сравнительная характеристика океанов"
fig, (ax) = plt.subplots(ncols=1)
```

```

fig.canvas.set_window_title(title)
fig.suptitle(title+":" + "\nпо их площади")
# Настройки диаграммы и осей
ax.set_xlabel("Океан")
ax.set_ylabel("Площадь (млн. кв. км)")
data = [
    ['Тихий', 179],
    ['Атлантический', 92],
    ['Индийский', 76],
    ['Северный Ледовитый', 14.8]
]
size = [x[1] for x in data]
nums = [x + 1 for x in range(len(size))]
tick_label = [x[0] for x in data]
ax.bar(nums, size, tick_label=tick_label, width=0.6,
color="#4590f3")
plt.show()

```

Результаты выполнения:

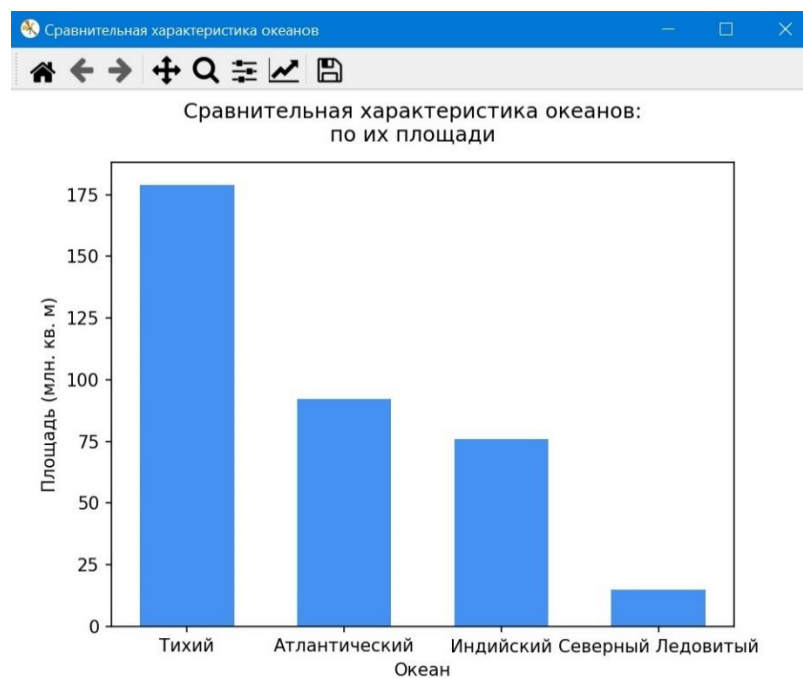


Рис. 42 Вертикальная столбчатая диаграмма

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 2

Задача: сравнить среднюю глубину океанов и построить горизонтальную столбчатую диаграмму.

Порядок выполнения:

Внесем некоторые корректировки в предыдущую программу. Новый код:

```
fig.suptitle(title+": " + "\nпо их средней глубине")
# Настройки диаграммы и осей
ax.set_ylabel("Океан")
ax.set_xlabel("Средняя глубина (м)")
data = [
    ['Тихий', 3980],
    ['Атлантический', 3600],
    ['Индийский', 3710],
    ['Северный'+'\nЛедовитый', 1220]
]
```

Изменим некоторые настройки для горизонтальной диаграммы:

```
ax.barh(nums, size, tick_label=tick_label, height=0.4,
color="#2000c0")
```

Результат выполнения:

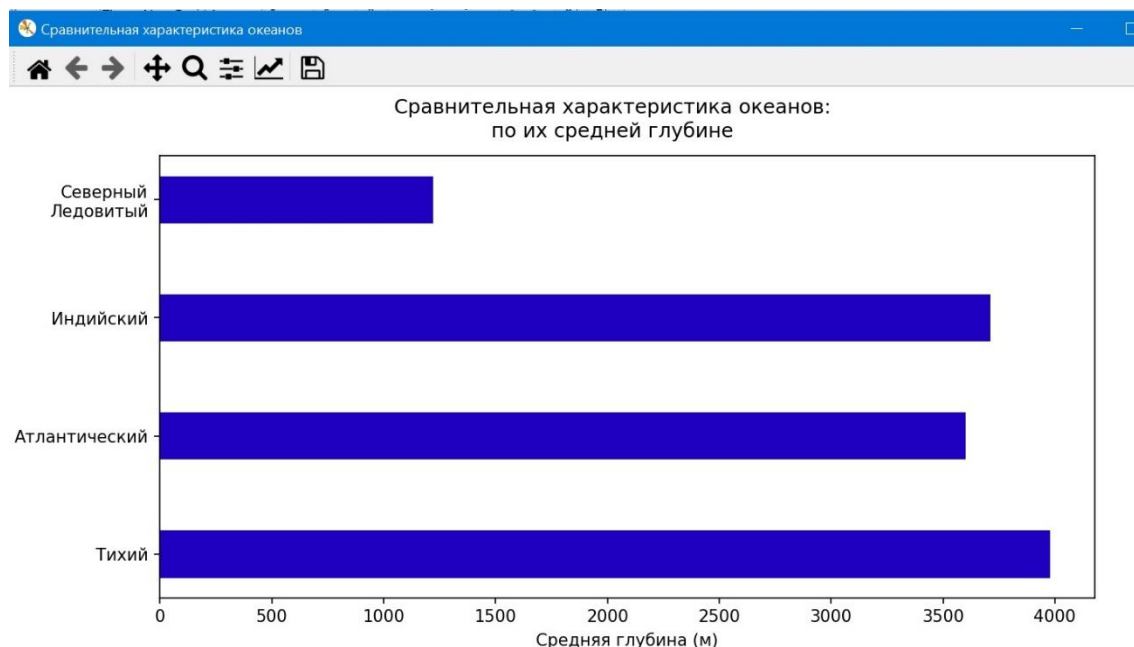


Рис. 43 Горизонтальная столбчатая диаграмма

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №15

Найдите значение курса доллара по дням недели за прошлую неделю календарного года. Можете использовать сайт Центрального Банка России (cbr.ru) или любой другой ресурс. Постройте столбчатую диаграмму с обозначением динамики курса доллара по отношению к рублю РФ.

ЧИТАЕМ

Круговая диаграмма отображает размер значений характеристик некоторых категорий относительно суммы значений по всем категориям, а сектора отдельных категорий на круговой диаграмме отражают процентное

соотношение доли их значений по отношению к сумме значений по всем категориям. Для построения круговой диаграммы может быть использован метод **pie()**.

Некоторые параметры метода **pie()**:

- **x** – набор значений (обязательный параметр);
- **colors** – набор значений цветов секторов;
- **labels** – набор подписей для секторов;
- **autopct** – форматная строка или функция для отображения значения доли на секторах круга;
- **shadow (bool)** – рисует тень, если True;
- **explode (массив)** – если параметр не равен None, то часть долей, которые перечислены в передаваемом значении будут вынесены из диаграммы на заданное расстояние;
- **wedgeprops** – определяет внешний вид долей;
- **radius** – радиус круга (по умолчанию: 1).

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 3

Задача:

Построить классическую круговую диаграмму, представляющую сведения о содержании белков, жиров и углеводов в 100 г клубники: белков – 0,8 г, жиров – 0,4 г, углеводов – 6,2 г.

Возможное решение:

```
import matplotlib.pyplot as plt
vals = [0.8, 0.4, 6.2]
labels = ["Белки", "Жиры", "Углеводы"]
fig, ax = plt.subplots()
ax.pie(vals, labels=labels)
ax.axis("equal")
plt.show()
```

Результат выполнения:

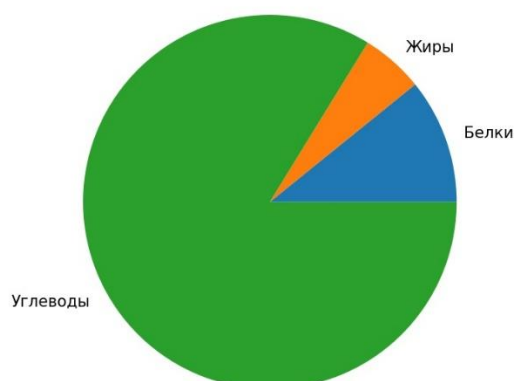


Рис. 44 Классическая круговая диаграмма

Изменим некоторые параметры в строке с методом **pie()**:

```
ax.pie(vals, labels=labels, wedgeprops=dict(width=0.5))
```

Получим следующий вид:

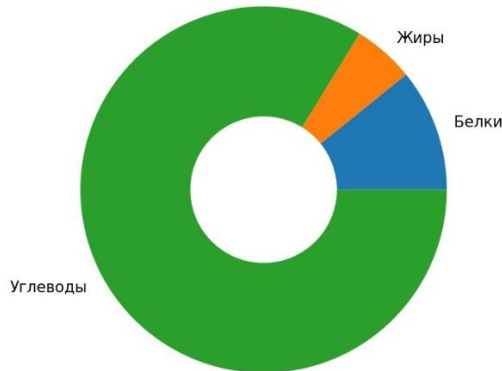


Рис. 45 Кольцевая диаграмма

Добавим к нашей диаграмме тень, вынесем отдельные доли, добавим подписи на сами доли:

```
explode = (0.1, 0, 0.1)
fig, ax = plt.subplots()
ax.pie(data, labels=labels, autopct='%1.1f%%', shadow=True,
colors=('r','g','y'), explode=explode, wedgeprops={'lw':1,
'ls':'-','edgecolor':"k"}, rotatelabels=True, startangle=5)
```

Результат выполнения:

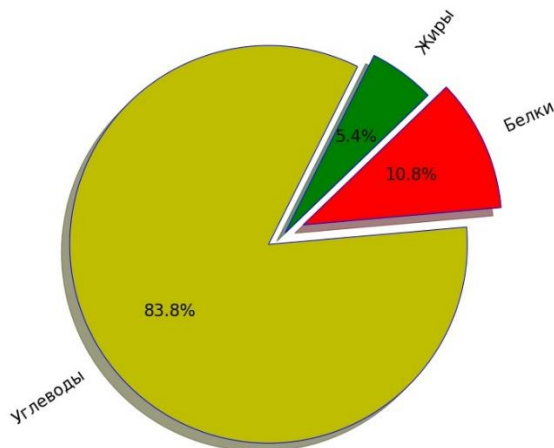


Рис. 46 Круговая диаграмма с выделением секторов (долей) и тенью

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 4

Задача:

Построить комбинированную круговую диаграмму, которая состоит из двух компонентов: внутренняя ее часть является детальным представлением информации, а внешняя – суммарным по заданным областям.

Возможное решение задачи:

```
import numpy as np
import matplotlib.pyplot as plt
```



```

fig, ax = plt.subplots()
offset=0.4
data = np.array([[3, 15, 6], [8, 15, 5], [11, 9, 7]])
cmap = plt.get_cmap("tab20b")
b_colors = cmap(np.array([5, 8, 2]))
sm_colors = cmap(np.array([1, 12, 3, 9, 10, 11, 3, 4, 5]))
ax.pie(data.sum(axis=1), radius=1.15, colors=b_colors,
wedgeprops=dict(width=offset, edgecolor='gray'))
ax.pie(data.flatten(), radius=1-offset, colors=sm_colors,
wedgeprops=dict(width=offset, edgecolor='gray'))
ax.axis("equal")
plt.show()

```

Результат выполнения задания:

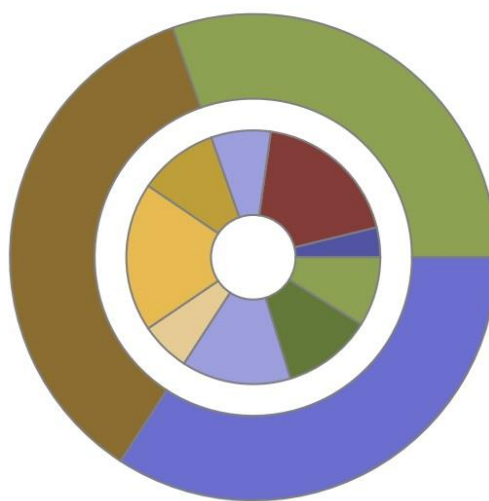


Рис. 47 Вложенная круговая диаграмма

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №16

1. Постройте круговую диаграмму, показывающую данные о соотношении итогов по всем видам коммунальных платежей за 1 квартал текущего года (см. таблица 4)

Таблица 4. Коммунальные платежи

| Месяц | Квартплата (в руб.) | Плата за электроэнергию (в руб.) | Плата за телефон (в руб.) | Всего (в руб.) |
|--------------|---------------------|----------------------------------|---------------------------|-----------------------|
| Январь | 1296 | 235 | 200 | 1731 |
| Февраль | 1305 | 110 | 250 | 1665 |
| Март | 1297 | 183 | 125 | 1605 |
| Апрель | 1298 | 106 | 210 | 1614 |
| Май | 1300 | 147 | 191 | 1638 |
| Июнь | 1321 | 121 | 188 | 1630 |
| Итого | 7817 | 902 | 1164 | 9883 |

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

В строке `cmr = plt.get_cmr("tab20b")` ПРИМЕРА 4 параметр "tab20b" отвечает за цветовую палитру. Постройте вложенную круговую диаграмму и поэкспериментируйте с цветовыми палитрами из следующего списка: ['Pastel1', 'Pastel2', 'Paired', 'Accent', 'Dark2', 'Set1', 'Set2', 'Set3', 'tab10', 'tab20', 'tab20c'].

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Приведите примеры демонстрации информации с помощью столбчатых и круговых диаграмм.
2. Изучите дополнительный материал и узнайте, как можно построить 3D графики с помощью mplot3d Toolkit.

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. Matplotlib. Урок 4.3. Визуализация данных. Столбчатые и круговые диаграммы. URL: <http://devpractice.ru/matplotlib-lesson-4-3-bar-pie/>
2. Галерея графических изображений в Matplotlib. URL: <https://matplotlib.org/stable/gallery/>
3. Matplotlib. Урок 5. Построение 3D графиков. Работа с mplot3d Toolkit. URL: <https://devpractice.ru/matplotlib-lesson-5-1-mplot3d-toolkit/>

§1.11 Обработка звука

ВСПОМИНАЕМ

Вспомним:

1. Для чего используются столбчатые диаграммы?
2. С использованием каких методов может быть нарисована столбчатая диаграмма, а с помощью каких круговая?

ЧИТАЕМ

Современные цифровые компьютеры передают и воспроизводят цифровую аудио- и видеоинформацию.

Если ранее при передаче радио- и телевизионных программ использовалась амплитудно-частотная модуляция несущей частоты радиосигнала низкочастотными звуковыми сигналами, то сейчас звуковой сигнал передается в оцифрованном виде.

Процесс перевода звуковых сигналов из непрерывной формы представления в дискретную, цифровую форму называют оцифровкой.

Оцифровка звуковых сигналов заключается в их временной дискретизации и амплитудном квантовании.

Временная дискретизация – это измерение амплитуды звуковых сигналов через небольшие равные промежутки времени. Величина этих промежутков выбирается из соображений наименьших допустимых искажений звука.

Амплитудное квантование – это определение в цифровом виде значений амплитуды звукового сигнала в моменты времени. При этом выбором шага квантования амплитуды минимизируются возможные искажения звука.

Преобразование сигналов из непрерывной (аналоговой) формы в цифровую выполняют аналого-цифровые преобразователи (АЦП – Analog/Digital Converter – ADC), а обратный процесс выполняют устройства, называемые цифро-аналоговыми преобразователями (ЦАП – Digital/Analog Converter – DAC).



Рис. 48 Место ЦАП и АЦП в цепочке преобразований и передаче сигнала

Этапы преобразования аналогового звукового сигнала в цифровой:

1. аналоговый звуковой сигнал подается на аналоговый фильтр, который ограничивает полосу частот сигнала и устраняет помехи и шумы;
2. для качественного преобразования аналогового сигнала в цифровой с определенной периодичностью осуществляется достаточно большое

- количество запоминаний (отсчетов) мгновенного уровня аналогового сигнала;
3. отсчеты поступают в АЦП, который преобразует мгновенное значение каждого запомненного сигнала в цифровой код или число;
 4. полученная таким образом последовательность бит преобразует звуковой сигнал в цифровой – дискретный как по времени, так и по величине.

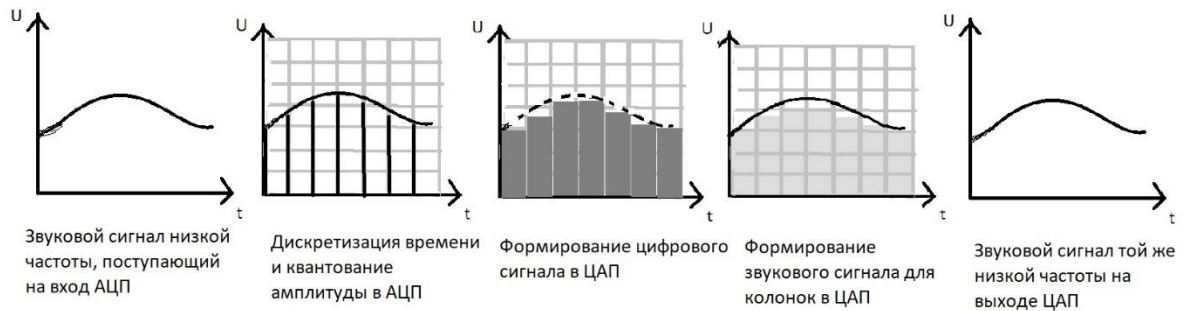


Рис. 49 Процесс оцифровки звукового сигнала

Важнейшим этапом аналого-цифрового преобразования является дискретизация («выборка», «сэмплирование») аналогового сигнала.

Дискретизация – это процесс взятия отсчетов непрерывного во времени сигнала в равноотстоящих друг от друга по времени точках.

Значение частоты дискретизации не может быть произвольным значением, так как фактически определяет ширину полосы частот сигнала, который может быть записан с помощью используемой цифровой системы.

Ширина этой полосы определяется теоремой отсчетов (Котельникова-Найквиста): сигнал, спектр частот которого занимает область от F_{\min} до F_{\max} (низкочастотный звуковой сигнал), может быть полностью представлен своими дискретными отсчетами с интервалом T_d , если T_d не превышает $1/2F_{\max}$.

Другими словами, частота дискретизации $f_d = 1/T_d$ в процессе преобразования должна быть, как минимум, вдвое больше наивысшей частоты звукового сигнала F_{\max} .

Если учесть, что человек способен слышать звуковые колебания, частота которых находится в диапазоне от 16 Гц до 20 кГц, и с позиций теоремы отсчетов взглянуть на требования к частотным характеристикам высококачественной звукотехники (например, проигрывателей аудио компакт-дисков), становится ясно, что частота дискретизации исходного звукового сигнала должна составлять не менее 40 кГц. Реально для подобных систем частота дискретизации выбирается не менее 44,1 кГц. Стандартное значение частоты дискретизации большинства звуковых карт составляет 44,1 и 48,0 кГц.

Итак, чем выше частота дискретизации, тем более точно будет восстановлен звуковой сигнал.

После дискретизации происходит квантование отсчетов. В процессе квантования производится измерение мгновенных значений уровня сигнала, полученных в каждом отсчете, причем осуществляется оно с точностью, которая напрямую зависит от количества разрядов, используемых для записи значения уровня.

Если длина кодового слова N , то количество уровней квантования будет равно $2N$. Если, например, значение амплитуды отсчета представляется 16-разрядным кодовым словом, то максимальное количество градаций уровня сигнала (уровней квантования) будет равно 65536 (2^{16}) и т.д.

Давайте оценим затраты памяти на запись звука в цифровой форме:

- если умножить число разрядов кодового слова (N) на частоту дискретизации сигнала (f_d), выраженную в герцах, то получится скорость передачи данных (v), которую должен обеспечивать цифровой канал записи/воспроизведения звука;
- если полученную скорость передачи данных умножить на общую длительность звукового сигнала в секундах (t), то получится объем памяти, который потребуется для хранения звуковых данных.

В случае записи стереосигнала, когда запись идет по двум (левому и правому) стереоканалам, скорость передачи данных и необходимый объем памяти удваиваются.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Задача:

В формате DVD-Audio производилась двухканальная (стерео) звукозапись с частотой дискретизации 192 кГц и 24-битным разрешением. В результате был получен файл размером 65 Мбайт, сжатие данных не производилось. Определите приблизительно, сколько времени (в минутах) проводилась запись. В качестве ответа укажите ближайшее к времени записи целое число.

Решение:

Чтобы определить время записи, нам нужно поделить размер файла на скорость звукозаписи: $t = I / v$.

Найдем скорость записи звука: $v = 192000 \text{ Гц} \cdot 24 \text{ бит} \cdot 2 = 9216000 \text{ бит/с}$.

Находим время: $t = (65 \cdot 1024 \cdot 1024 \cdot 8 \text{ бит}) / (9216000 \text{ бит/с}) \approx 59 \text{ с} = 1$
мин

Ответ: 1 мин.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 2

Задача:

Музыкальный фрагмент был оцифрован и записан в виде файла без использования сжатия данных в формате квадро (четырёхканальная запись). Известно, что размер получившегося файла равен 2,35 Мбайт, частота дискретизации 44,1 кГц при количестве уровней квантования 65536. Получившийся файл был передан в город А по каналу связи.

Затем тот же музыкальный фрагмент был оцифрован повторно в формате стерео (двухканальная запись) с разрешением в 2 раза выше и частотой дискретизации в 1,5 раза выше, чем в первый раз. Сжатие данных не производилось. Полученный файл был передан в город Б; пропускная способность канала связи с городом Б в 3 раза выше, чем канала связи с городом А.

Сколько секунд длилась передача файла в город А и в город Б?

В ответе запишите в какой город и во сколько раз быстрее был передан данный фрагмент. В качестве ответа сначала напишите букву, соответствующую городу, а после нее цифру, например, А3.

Решение:

Найдем время передачи файла в город А:

- переведем Мбайты в биты;
- зная количество уровней квантования $65536 = 2^{16}$, найдем разрешение $N = 16$ бит, тогда $t_A = (2,35 \cdot 1024 \cdot 1024 \cdot 8 \text{ бит}) / (4 \cdot 16 \cdot 44100 \text{ бит/с}) = 7 \text{ с}$

Найдем время передачи файла в город Б:

Объём файла прямо пропорционален разрешению файла, количеству каналов и его частоте дискретизации, следовательно, объём файла во втором случае в $I_B = (2 \cdot 1,5) / 2 = 3/2 I_A$.

Длительность передачи обратно пропорциональна пропускной способности канала связи, откуда получаем, что длительность передачи файла во второй раз равна: $t_B = t_A \cdot 3/2 \cdot 1/3 = 3,5 \text{ с}$.

Ответ: Б2.

РЕШАЕМ В ТЕТРАДИ

Задание 4.

В таблице 5 приведены сравнительные данные некоторых звуковых форматов. Часть данных была стерта. Найдите эти недостающие данные и заполните таблицу полностью.

Таблица 5. Сравнение звуковых форматов

| Название формата | Квантование, бит | Частота дискретизации, кГц | Число каналов | Величина потока данных с диска, кбит/с | Степень сжатия/упаковки |
|---------------------|------------------|----------------------------|---------------|--|-------------------------|
| CD | 16 | ??? | 2 | 1411,2 | 1:1 без потерь |
| Dolby Digital (AC3) | 20 | 48 | 6 | ??? | ~12:1 с потерями |
| DTS | ??? | 48 | 4 | 1536 | ~3:1 с потерями |

РЕШАЕМ В ТЕТРАДИ

Задание 5.

Используя данные заполненной таблицы 5, определите размер файла в Мбайтах, полученного при начальной записи, если запись длилась полминуты и музыкальный фрагмент был записан в формате Dolby Digital, оцифрован и сохранён в виде файла без использования сжатия данных. Ответ запишите целым числом, кратным 10, единицы измерения писать не нужно.

РЕШАЕМ В ТЕТРАДИ

Задание 6.

Используя данные заполненной таблицы 5, определите возможную частоту дискретизации, если известно, что тот же музыкальный фрагмент был записан повторно в формате моно и оцифрован с разрешением в 4 раза меньше, чем в первый раз. При этом производилось сжатие данных, объем сжатого фрагмента стал равен 20% от начального. Ответ запишите целым числом, кратным 10, единицы измерения писать не нужно.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Какой процесс называется оцифровкой звуковых сигналов?
2. Перечислите основные этапы преобразования аналогового звукового сигнала в цифровой.

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. Мультимедиа технологии: цифровое представление звуковых сигналов. Сборник бесплатных электронных лекций. URL: http://window.edu.ru/catalog/pdf2txt/751/71751/49019?p_page=4
2. Кодирование звуковой информации. URL: <https://dic.academic.ru/dic.nsf/ruwiki/1524417>

§1.12 Контроль по теме "Графика и мультимедиа"

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Вариант 1

1. Для экономии памяти изображение сжимают. После сжатия изображения его размер стал равен 2 Мбайт. Известно, что разрешение уменьшили вдвое, а цветовую палитру с 32768 цветов сократили до 1024 цветов. Сколько Мбайт занимал исходный файл?

2. В папке «Галерея» корпорации «Питон» хранятся фотографии сотрудников размером 2048×1600 пикселей. При кодировании используется алгоритм сжатия изображений, позволяющий уменьшить размер памяти для хранения одного изображения в среднем в 8 раз по сравнению с независимым кодированием каждого пикселя. Каждая фотография дополняется служебной информацией, которая занимает 64 Кбайт. Хранение 32 изображений занимает 12 Мбайт памяти. Какое максимальное количество цветов можно использовать в палитре каждого изображения?

3. Музыкальный фрагмент был оцифрован и записан в виде файла без использования сжатия данных. Получившийся файл был передан в город А по каналу связи. Затем тот же музыкальный фрагмент был оцифрован повторно с разрешением в 2 раза выше и частотой дискретизации в 3 раза меньше, чем в первый раз. Сжатие данных не производилось. Полученный файл был передан в город Б за 15 секунд; пропускная способность канала связи с городом Б в 2 раза выше, чем канала связи с городом А. Сколько секунд длилась передача файла в город А? В ответе запишите только целое число, единицу измерения писать не нужно.

4. Производится звукозапись музыкального фрагмента в формате стерео (двухканальная запись) с частотой дискретизации 32 кГц и 32-битным разрешением. Результаты записываются в файл, сжатие данных не производится; размер полученного файла – 32 Мбайт. Затем производится повторная запись этого же фрагмента в формате моно (одноканальная запись) с частотой дискретизации 16 кГц и 16-битным разрешением. Сжатие данных не производилось. Укажите размер файла в Мбайт, полученного при повторной записи.

Вариант 2

1. С помощью фотокамеры было получено растровое изображение размером 640×1280 пикселей. Для кодирования цвета каждого пикселя используется одинаковое количество бит, коды пикселей записываются в файл один за другим без промежутков. Объем файла этого изображения не превышает 500 Кбайт без учета размера заголовка файла. Заголовок файла занимает 1,5 Кб. Какое максимальное количество цветов можно использовать в палитре?

2. В папке «Галерея» корпорации «Питон» хранятся фотографии сотрудников размером 1600×1200 пикселей. При кодировании используется алгоритм сжатия изображений, позволяющий уменьшить размер памяти для хранения одного изображения в среднем в 5 раз по сравнению с независимым кодированием каждого пикселя. Каждая фотография дополняется служебной информацией, которая занимает 100 Кбайт. Хранение 32 изображений занимает 10 Мбайт памяти. Какое максимальное количество цветов можно использовать в палитре каждого изображения?

3. Музыкальный фрагмент был оцифрован и записан в виде файла без использования сжатия данных. Получившийся файл был передан в город А по каналу связи. Затем тот же музыкальный фрагмент был оцифрован повторно с разрешением в 3 раза выше и частотой дискретизации в 2 раза меньше, чем в первый раз. Сжатие данных не производилось. Полученный файл был передан в город Б за 20 секунд; пропускная способность канала связи с городом Б в 2 раза ниже, чем канала связи с городом А. Сколько секунд длилась передача файла в город А? В ответе запишите только целое число, единицу измерения писать не нужно.

4. Производится четырехканальная (квадро) звукозапись с частотой дискретизации 96 кГц. Запись длится 4 минуты 15 секунд, её результаты записываются в файл без сжатия данных, причём каждый сигнал кодируется минимально возможным и одинаковым количеством бит. Информационный объём полученного файла не превышает 376 Мбайт. Определите максимальную битовую глубину кодирования звука (разрешение), которая могла быть использована в этой записи. В ответе запишите только число.

ПРАКТИЧЕСКАЯ ЧАСТЬ

Вариант 1

Определите наименьшее значение n , при котором сумма чисел, которые будут выведены при вызове $F(n)$, будет больше 1000000. Запишите в ответе

сначала найденное значение n , а затем через пробел – соответствующую сумму выведенных чисел.

| Python | Паскаль | C++ |
|---|---|---|
| <pre>def F(n): print(2*n+4) if n > 1: print(3*n-2) F(n-1) F(n-4)</pre> | <pre>procedure F (n: integer); begin writeln(2*n+4); if n > 1 then begin writeln(3*n-2); F(n-1); F(n-4); end; end;</pre> | <pre>void F(int n) { cout << 2*n+4 << endl; if(n > 1) { cout << 3*n-2 << endl; F(n-1); F(n-4); } }</pre> |

Вариант 2

Определите наименьшее значение n , при котором сумма чисел, которые будут выведены при вызове $F(n)$, будет больше 2000000. Запишите в ответе сначала найденное значение n , а затем через пробел – соответствующую сумму выведенных чисел.

| Python | Паскаль | C++ |
|---|---|---|
| <pre>def F(n): print(4*n+1) if n > 1: print(2*n+5) F(n-1) F(n-2)</pre> | <pre>procedure F (n: integer); begin writeln(n+1); if n > 1 then begin writeln(n+5); F(n-1); F(n-2); end; end;</pre> | <pre>void F(int n) { cout << n+1 << endl; if(n > 1) { cout << n+5 << endl; F(n-1); F(n-2); } }</pre> |

II. Файлы и файловая система

§2.1 Классификация и устройство вычислительной техники. Установка и подключение исполняемых библиотек

ВСПОМИНАЕМ

Вспомним:

1. Что мы знаем о файловой системе компьютера?
2. Какие компоненты файловой системы вам известны?

ЧИТАЕМ

Для эффективной работы с информацией необходим порядок, определяющий способ организации, правила хранения и именования данных на носителях информации. Такой порядок называется **файловой системой**.

Файловая система определяет размер имени файла (папки), максимальный возможный размер файла и раздела носителя, набор атрибутов (свойств, признаков) файла. Некоторые файловые системы предоставляют сервисные возможности, например разграничение доступа пользователей или шифрование файлов. Файловая система поддерживает структуру файлов на логическом и физическом уровнях. Давайте выясним, что такое файл и чем он отличается от папки (каталога).

Файл – это именованная область данных на долговременном носителе информации. **Папка** (иногда её ещё называют **каталог** или **директория**) – объект в файловой системе, упрощающий организацию файлов. Это что-то вроде контейнера для файлов, в который файлы и другие папки группируются по какому-либо принципу.

Имя файлу придумывает тот, кто его создает. Имя файла (не папки) состоит из двух частей. Расширение обычно автоматически задается приложением, в котором создается файл, и указывает на его тип. Для удобства поиска файлы организуются в многоуровневую систему, которая имеет «древовидную» (иерархическую) структуру. Начальный, корневой, каталог содержит вложенные каталоги 1-го уровня, в свою очередь, в каждом из них бывают вложенные каталоги 2-го уровня и т. д.

Для того, чтобы найти файл в иерархической файловой структуре, необходимо указать путь к файлу. В путь к файлу входят записываемые через разделитель \ логическое имя диска и последовательность имен вложенных друг в друга каталогов, в последнем из которых находится данный нужный файл. Например: *C:\Школа\Информатика*. Путь к файлу вместе с именем файла называют **полным именем файла**: *C:\Школа\Информатика\видео.mp4*

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Условие:

Пользователь работал с файлом *C:\www\img\weather\rain.png*. Сначала он поднялся на один уровень вверх, затем создал каталог *pic*, в нём создал ещё один каталог *wallpaper* и переместил туда файл *rain.png*. Каким стало полное имя файла после перемещения?

Модель рассуждения:

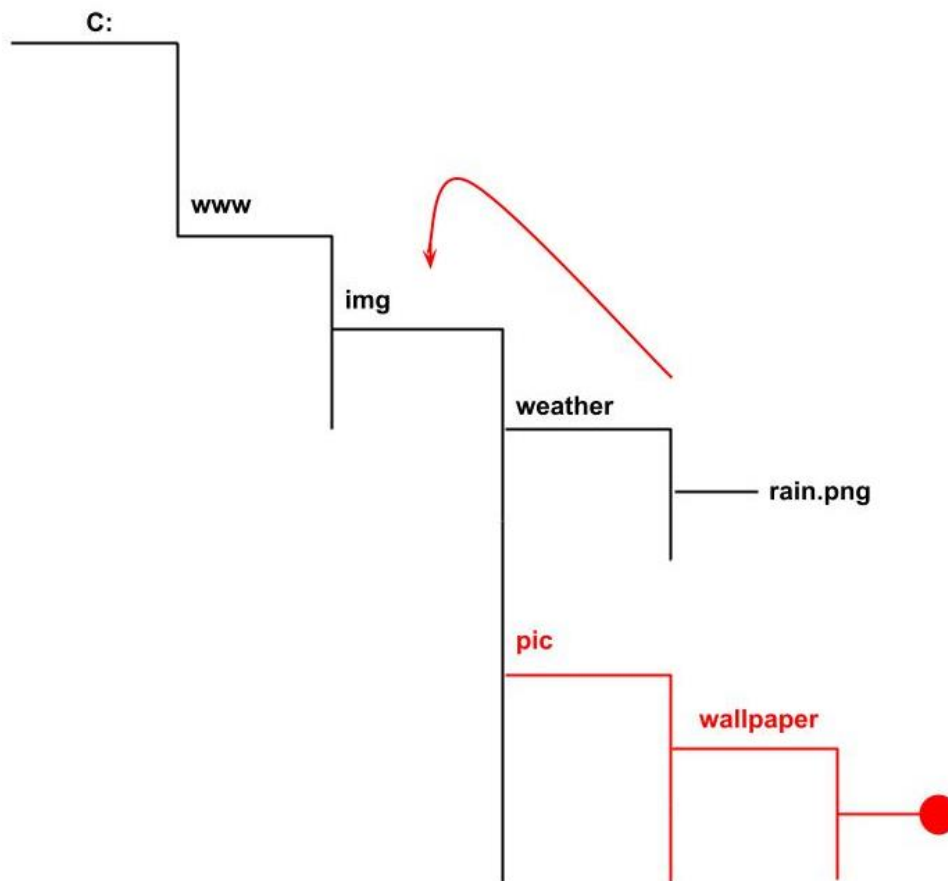


Рис. 50 Дерево папок

Ответ:

C:\www\img\pic\wallpaper\rain.png (понятен по рис. 50, изменения выделены красным цветом).

РЕШАЕМ В ТЕТРАДИ

Задание 7.

Сдав доклад по биологии на "отлично", ученик перенёс папку, полный путь до которой был *D:\Учёба\Биология\Млекопитающие*, в папку Сданные, расположенную в корне диска C. Укажите полный путь к файлу *Виды слонов.txt*, расположенному в папке Млекопитающие.

РЕШАЕМ В ТЕТРАДИ

Задание 8.

Дима хотел послушать музыку на компьютере, для этого ему нужно было включить музыкальный файл *Белые_розы.mp3*. Он начал работу с каталога *C:\Музыка\Хиты*. Сначала он спустился на один уровень вниз, в каталог *Ретро*, затем поднялся на один уровень вверх, потом спустился на один уровень в каталог *Лучшие*, после чего спустился в каталог *Про_цветы* и нашёл там нужный музыкальный файл. Запишите полный путь к данному файлу.

РЕШАЕМ В ТЕТРАДИ

Задание 9.

Пользователь работал с файлом *C:\Class\9b\Pascal\task.pas*. Затем он поднялся на один уровень вверх, создал каталог *Homework*, в нём создал ещё один каталог *Program* и переместил в него файл. Каким стало полное имя этого файла после перемещения?

РЕШАЕМ В ТЕТРАДИ

Задание 10.

В некотором каталоге хранился файл с именем *ex.pas*. После того как в этом каталоге создали подкаталог *Pascal* и переместили в него файл *ex.pas*, полное имя файла стало *C:\Olimp\Ivanov\Pascal\ex.pas*. Каким было полное имя этого файла до перемещения?

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №17

Структурировать информацию в виде папок и файлов по заданному тексту, объединив по смыслу информационные объекты. Представьте, что к данному тексту необходимо найти много дополнительной информации. Как бы вы назвали папки и файлы? Отобразите полученную файловую структуру в виде дерева.

Текст*:

Бактерии живут всюду: в почве, в море, в воздухе. Это очень маленькие, микроскопические организмы. Они существуют на коже человека и даже свободно парят в воздухе. В точке, которую мы поставим карандашом, может поместиться около сотни бактерий. Внешний вид у разных видов бактерий сильно отличается. На земле бактерий столько, что их совокупный вес больше, чем вес всех живых существ, вместе взятых. Бактерии размножаются, постоянно делясь надвое. Это происходит так быстро, что за сутки одна бактерия может превратиться в тысячи миллионов. Бактерии принадлежат к числу древнейших обитателей земли. На севере, во льдах учёные нашли бактерии, жившие три миллиарда лет назад. Некоторые бактерии могут питаться, используя

солнечную энергию. Этот процесс называется фотосинтезом. Но большинство живут на живых растениях и животных или в их останках, когда те умирают. Некоторые бактерии уничтожают мертвые растения и животных. Так они способствуют возвращению ценных питательных веществ в почву, откуда растения могут использовать их для своего роста. Однако некоторые бактерии бывают вредны для нас, потому что вызывают пищевые отравления и наиболее опасные болезни, такие, как туберкулез, пневмония, холера и тиф. Вредоносные бактерии называются также микробами. Их можно убить антисептиками или лекарствами, которые называются антибиотиками. Знаменитый ученый Луи Пастер открыл, что многие болезни вызываются бактериями. Он показал, что микробы переносят инфекцию от одного человека к другому. И он создал спасающие жизнь прививки.

Вирусы намного меньше бактерий. Их вред заключается в том, что они внедряются в клетки и мешают их нормальной работе. Вирусы вызывают целый ряд заболеваний, от легких, таких как самая обыкновенная простуда, до смертельных, таких как бешенство и Covid. Как правило, против вирусов нельзя использовать обычные антибиотики. Когда вирусы попадают в организм, иммунная система сама производит антитела, особого рода защитную систему, которая старается их уничтожить. Вакцина содержит мертвые или обезвреженные вирусы, подобные болезнетворным, она стимулирует иммунную систему производить антитела, так что в случае, если позднее на организм нападут вредоносные микробы, он будет отчасти подготовлен.

**источник: Читаем технические тексты по-русски: учебное пособие по научному стилю речи / Л.И. Ярица; Томский политехнический университет. – Томск:*

Изд-во Томского политехнического университета, 2012. – 196 с.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Используя описания путей к файлам, постройте схему (дерево каталогов):

Z:\Главная\Информатика\Документы\Рисунки\картинка.bmp

Z:\Главная\Информатика\Документы\Файлы.doc

Z:\Главная\Информатика\Документы\Файловая система.doc

Z:\Главная\Информатика\Документы\Новый год\Открытка.doc

Z:\Главная\Информатика\Документы\Рисунки\схема.jpg

Z:\Главная\Информатика\Документы\Новый год\Сценарий.wav

2. Кроссворд по теме «Файл и файловая структура»

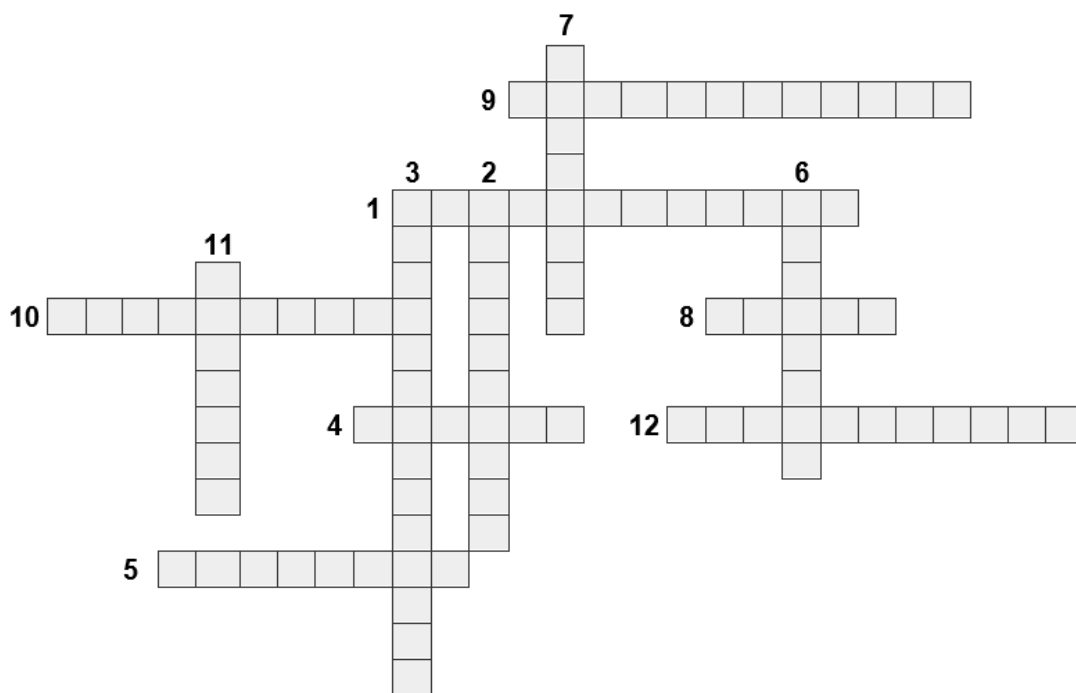


Рис. 51 Кроссворд

1. Перенос файла в другой каталог или другой носитель, при этом исходный файл уничтожается.
2. Часть имени файла, показывающая, в какой программе создан файл.
3. Изменение имени файла.
4. Графическое изображение иерархической файловой структуры.
5. Структура диска, являющаяся совокупностью файлов на диске и взаимосвязей между ними.
6. Расположение частей (элементов) целого в порядке от высшего к низшим.
7. Файл, содержащий графические, текстовые (рисунки, тексты) данные.
8. Поименованная область внешней памяти.
9. Создание копий файлов в другом каталоге или на другом носителе.
10. Уничтожение объекта в исходном каталоге.
11. Поименованная совокупность файлов.
12. Файлы, содержащие прикладные программы.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Что такое файл? Из каких частей состоит имя файла?
2. Какое имя для файла называется полным?

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. [Файловые системы](#)
2. [Типы файловых систем](#)
3. [Форматы файлов](#)
4. [Атрибуты файлов](#)

§2.2 Регулярные выражения. Маска файла.

ВСПОМИНАЕМ

Вспомним:

1. Какие задачи решает файловая система?
2. Что такое папка?

ЧИТАЕМ

Регулярные выражения — это своеобразный фильтр для текстовых данных. Это очень мощный инструмент для поиска текстовых фрагментов и проверки их соответствия заданному шаблону. Как применяются регулярные выражения по отношению к файлам? Три самых востребованных операции: поиск файлов на внешнем носителе – по имени и содержимому; выделение и отмена выделения определённых типов файлов; групповое переименование файлов для определенной цели (нумерация файлов, преобразование символов имени в верхний/нижний регистр, добавление текущих даты/времени).

Для поиска используется строка-образец (англ. pattern, по-русски её часто называют «шаблоном», «маской»), состоящая из символов и метасимволов и задающая правило поиска. Большинство символов в регулярном выражении представляют сами себя за исключением специальных символов [] \ / ^ \$. | ? * + () { }

Рассмотрим их подробнее:

- в масках, кроме «обычных» символов (допустимых в именах файлов) используются два специальных символа: звездочка «*» и знак вопроса «?»;
- звездочка «*» обозначает любое количество любых символов, в том числе может обозначать пустую последовательность;
- знак вопроса «?» обозначает ровно один любой символ;
- при выводе списка имен файлов они могут быть отсортированы по имени, типу (расширению), дате последнего изменения, размеру; это не меняет их размещения на диске;
- если установлена сортировка по имени или типу, сравнение идет по кодам символов, входящих в имя или в расширение.

Так, символ «?» обозначает место, на котором обязательно находится какой-то символ. Так, по запросу `р?ка` могут быть обнаружены слова «река» или «рука». Но не может быть найдено слово «речка», так как за маской «?» скрывается лишь один символ. Чтобы найти слова «речка» и «ручка» необходима запись `р??ка`.

Кроме того, при поиске используется символ «*». Он означает, что на указанном месте может скрываться сколько угодно символов, включая ноль.

То есть по запросу «р*ка» может найтись и «река», и «ручка», и даже «рка». Есть и другие примеры:

- w*.*|*.bak *.old Эта запись означает выделить все файлы, которые начинаются с w и не заканчиваются .bak или .old.
- |*.exe Это выражение помогает выделить все файлы, кроме программ.
- *.ini | windows\ Находит все INI-файлы, за исключением тех, которые содержатся в каталогах windows и их подкаталогах.
- *.htm? | _vti*\ Находит все файлы HTML, за исключением тех, которые содержатся в подкаталогах, начинающихся на _vti (используются Frontpage).
- windows\ system32\ *.ini Находит INI-файлы только в каталогах windows\ и system32\.

РЕШАЕМ В ТЕТРАДИ

Задание 11.

Для групповых операций с файлами используются маски имен файлов. Маска представляет собой последовательность букв, цифр и прочих допустимых в именах файлов символов, в которых также могут встречаться следующие символы: Символ «?» (вопросительный знак) означает ровно один произвольный символ. Символ «*» (звездочка) означает любую последовательность символов произвольной длины, в том числе «*» может задавать и пустую последовательность. Определите, какое из указанных имен файлов удовлетворяет маске: ?ba*r.?xt

- 1) bar.txt 2) obar.txt 3) obar.xt 4) barr.txt

РЕШАЕМ В ТЕТРАДИ

Задание 12.

Находясь в корневом каталоге только что отформатированного диска, ученик создал 3 каталога. Затем в каждом из них он создал еще по 4 каталога. Сколько всего каталогов оказалось на диске, включая корневой?

- 1) 12 2) 13 3) 15 4) 16

РЕШАЕМ В ТЕТРАДИ

Задание 13.

В каталоге находятся файлы со следующими именами:

- file.mdb

- file.mp3
- ilona.mpg
- pile.mpg
- miles.mp3
- nil.mpeg

Определите, по какой из масок будет выбрана указанная группа файлов:

- file.mp3
- pile.mpg
- miles.mp3
- nil.mpeg

1) ?il*.m* 2) ?il*.mp* 3) *il?.mp* 4) ?il*.mp?

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

№ 1

Для групповых операций с файлами используются маски имен файлов. Маска представляет собой последовательность букв, цифр и прочих допустимых в именах файлов символов, в которых также могут встречаться следующие символы: символ «?» (вопросительный знак) означает ровно один произвольный символ; символ «*» (звездочка) означает любую последовательность символов произвольной длины, в том числе «*» может задавать и пустую последовательность. Определите, какое из указанных имен файлов удовлетворяет маске: F??tb*.d?*

1) **Fructb.d** 2) **Feetball.ddd** 3) **Football.mdb** 4) **Futbol.doc**

№ 2

В каталоге находятся файлы со следующими именами:

- bike.mdb
- bike.mp3
- iks.mpg
- like.mpg
- mikes.mp3
- nike.mpeg

Определите, по какой из масок будет выбрана указанная группа файлов:

- bike.mp3
- like.mpg
- mikes.mp3
- nike.mpeg

- 1) `?ik*.m*` 2) `?ik*.mp?` 3) `*ik?.mp*` 4) `?ik*.mp*`

№ 3

В каталоге находятся файлы со следующими именами:

- `acsacal.db`
- `cassandra.db2`
- `cucsa.db2`
- `curasao.dbr`
- `curasao.dat`
- `masai.db`
- `mcsan.db2`

Определите, по какой из масок будет выбрана указанная группа файлов:

- `acsacal.db`
- `cassandra.db2`
- `cucsa.db2`
- `curasao.dbr`
- `mcsan.db2`

- 1) `*c*sa*.db*` 2) `*c*s*.db*` 3) `*s*.db*` 4) `*c*s*.db?`

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Какое выражение называется регулярным?
2. Где применяются регулярные выражения?
3. Какие специальные знаки применяются в маске файлов, их назначение?
4. Применяли ли вы маски файлов? Расскажите о своем опыте.

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. [Материал из Викиучебника](#)
2. [Кроссворд "Расшифруй регулярное выражение"](#)
3. [Тестер регулярного выражения](#)

§2.3 Алгоритмы работы с текстовой информацией

ВСПОМИНАЕМ

Вспомним:

1. Для чего используются регулярные выражения?

2. Символы * и ? в регулярных выражениях означают...

ЧИТАЕМ

Подготовка различных текстовых документов, пожалуй, самая распространенная функция современного персонального компьютера. Текстовые документы нужны всем, они создаются, копируются, хранятся, изменяются, шифруются, передаются, какую бы область деятельности человека вы не взяли. Текстовые файлы — один из самых популярных типов данных в мире.

Что такое текст с точки зрения информатики? **Текст** (от лат. textus «ткань; сплетение, связь, сочетание») — связная и полная в определенном смысле последовательность символов. **Текстовый файл** хранит текстовые данные. Часто текстовые данные понимаются как текст на каких-либо формальных или естественных языках, который может быть прочитан и понят человеком. Но текстовые данные чаще анализирует не человек, а цифровое устройство. Тогда текстовый файл хранит текстовые данные в специальном формате «двоичные данные», информация в котором закодирована цифровым способом, не подходящим для восприятия человеком.

Цифровые устройства и программы обычно обрабатывают текстовые и двоичные данные по-разному, а некоторые из них предназначены для обработки именно текстовых данных или именно двоичных. Программы, назначение которых — создание и редактирование текстовых данных называются текстовыми **редакторами**.

Самые распространенные текстовые форматы – TXT, RTF, DOCX, ODT, PDF. Чем они отличаются? Различие заключается в их возможностях для форматирования, обработки и распространения текста, а также в их совместимости со множеством программных продуктов. Существует большое количество текстовых форматов для текстовых данных. Ценным навыком является способность разбираться, в каком случае какой формат предпочтительнее!

Однако работать с текстами можно не только в текстовом редакторе. Создавать и обрабатывать текстовые файлы можно в программном коде. Минимальной единицей обработки является символ. Часто единицей обработки является строка. Строка – это последовательность символов, которая может обрабатываться как единое целое. Строка состоит из символов, каждый из которых имеет свой индекс, для обращения к символу с номером i строки S используется запись $S[i]$, это говорит о том, что строка может рассматриваться как список символов. Сохранить длину строки S в целую переменную n можно так же, как длину списка:

```
S = input()
N = len(S)
```

Кроме «прямой» индексации (начинающейся с 0) применяются отрицательные индексы: S [-1] означает последний символ строки S, S[-N] означает первый символ строки S длиной N.

Знак сложения при работе со строками означает склеивание, конкатенацию двух строк в одну (добавление второй строки в конец первой или начало первой), например:

```
S = '23'
S = S + '45'    # получили '2345'
S = '01' + S    # получили '012345'
```

Знак умножения при работе со строками означает повторение части или всей строки заданное количество раз, например:

```
P = '123'
S = P[0] * 3 + P    # получили '111123'
S = P * 5           # получили '123123123123123'
```

Строки можно сравнивать между собой так же, как и числа. Можно определить, какая из строк больше, какая – меньше. При сравнении строк сравниваются коды символов в лексикографическом порядке. Существует множество кодировок символов. Понять это легче всего, разобрав, например одну из них (почитайте unicode-table.com). В современных кодировках и русские, и английские буквы расположены в алфавитном порядке, цифры также идут от меньшей к большей. Посмотрим, как можно получить по символу код, а по коду символ:

- с помощью функции ord можно получить код символа ord('#'), он равен 35
- с помощью функции chr можно сделать обратный переход, chr(35) равен '#'

В задачах со строками часто бывает нужным преобразовать строку в число, а число в строку. Тогда на помощь приходят функции str, int, float:

```
S = '123 '
print(int(S) ** 2, float(S) ** 2)
# результат такой: 15129 15129.0

a, b = float(input()), float(input())
print('\n'.join('*' + str(a) + '*' + str(b) + '*'))
# получили столбик из звездочек и символов (чисел)
```

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Задача:

Задан шестизначный номер автобусного билета. Если сумма первых трех цифр совпала с суммой последних трех, то билет счастливым. Как средствами языка Python определить, является ли билет счастливым?

Порядок рассуждения:

Как решить задачу наиболее коротким способом? Применить срезы (части строки, которые могут быть определены диапазоном и шагом), перейти от строк к числовым спискам, вызвать функцию `sum`.

Пример кода:

```
ticket = input()
print('happy' if sum([int(i) for i in ticket[:3]]) ==
      sum([int(i) for i in ticket[3:]]) else 'unhappy')
```

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 2

Задача: переделать строку 'информатика' в строку 'романтика' путем преобразований, затем получить строку 'семантика'. Составьте короткий код решения.

Порядок рассуждения:

- 1) взять срез 'матика'
- 2) вставить 'н' перед 'т'
- 3) взять срез 'ор', перевернуть
- 4) вставить 'ро' перед 'м', вывести
- 5) заменить 'ро' на 'се', вывести

Пример кода:

```
S = 'информатика'
P = list(S[5:])
P.insert(2, S[1])
P.insert(0, S[4:2:-1])
P = ''.join(P)
print(P, P.replace('ро', 'се'))
# получили - романтика семантика
```

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 3

Задача: необходимо проанализировать частоту вхождения символов. На вход программе подается предложение на английском языке. Требуется написать программу, которая определяет, можно ли переставить английские буквы этого предложения (не учитывая остальные символы) так, чтобы получить палиндром – слово, которое читается одинаково слева направо и справа налево. Строчные и прописные буквы не различаются. Если этого

сделать нельзя, программа должна вывести на экран слово «Нет», а если можно – слово «Да» и в следующей строчке искомое слово прописными буквами. Если таких слов несколько, нужно вывести последнее в алфавитном порядке слово.

| Входные данные | Выходные данные |
|----------------|-------------------|
| Deed drd Neer. | Да REEDDNDDEER |

Возможное решение:

```
alf = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
freq = [0] * 26
for ch in input():
    if ch.isalpha():
        freq[alf.index(ch.upper())] += 1
nech = 0
for i in freq:
    if i % 2 > 0:
        nech += 1
        nnech = freq.index(i)
print('Да' if nech <= 1 else 'Нет')
if nech <= 1:
    for i in range(25, -1, -1):
        if freq[i] > 0 and freq[i] % 2 == 0:
            print(alf[i] * (freq[i] // 2), end='')
    if nech:
        print(alf[nnech] * freq[nnech], end='')
    for i in range(0, 26):
        if freq[i] > 0 and freq[i] % 2 == 0:
            print(alf[i] * (freq[i] // 2), end='')
```

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №18

На вход программе подается последовательность символов, среди которых могут быть и цифры. Требуется написать программу, которая составляет и выводит минимальное число из тех цифр, которые не встречаются во входных данных. Ноль не используется. Если во входных данных встречаются все цифры от 1 до 9, то следует вывести «0».

| Входные данные | Выходные данные |
|----------------|-----------------|
| 1A734B39 | 2568 |

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №19

На вход программе подается текст заклинания латинскими буквами, текст состоит из слов, то есть может содержать пробелы. Гарри Поттеру нужно зашифровать его. Сначала Гарри определяет количество букв в самом коротком слове, обозначив полученное число через K. Затем он заменяет каждую букву в заклинании на букву, стоящую в алфавите на K букв ранее (алфавит считается циклическим, то есть перед буквой A стоит буква Z), оставив другие символы неизменными. Строчные буквы при этом должны остаться строчными, а прописные – прописными, все пробелы сохраняют свои места. Требуется написать программу для Гарри Поттера, которая будет выводить на экран текст зашифрованного заклинания.

| Входные данные | Выходные данные |
|--------------------------------|--------------------------------|
| Bestia Zephyrus phoca Theatrum | Wznodv Uzkctmpn kcjxv Oczvomph |

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

№ 1

Одним из ста наиболее употребительных слов русского языка является слово «это». Дана строка, в которой подстрока 'это' встречается как минимум два раза. Разверните последовательность символов, заключенную между первым и последним появлением 'это', в противоположном порядке. Вывести нужно полученную строку.

| Входные данные | Выходные данные |
|---|----------------------------|
| Вот как это надо, это совсем не сложно. Бери все это! | есв иреБ .онжолс ен месвос |

№ 2

Дана строка, для размещения в колонке текста ее необходимо укоротить. Замените в этой строке все 'ай-яй' на слово 'ой', подсчитайте сколько раз вы это сделали.

| Входные данные | Выходные данные |
|--------------------------|-------------------------|
| ну, ай-яй-яй и еще ай-яй | ну, ой-яй и еще ой 2 |

№ 3

Дана строка для шифровки и передачи. Удалите из нее все символы, коды которых делятся на 3. Вывести необходимо в столбик коды всех удаленных символов и новую полученную строку.

| Входные данные | Выходные данные |
|----------------|--|
| шифровка | 1096 1080 1092 1088 1086 1074 |

| | |
|--|------|
| | 1082 |
| | 1072 |
| | ифов |

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Какие данные хранит текстовый файл?
2. Как называются программы, обрабатывающие текстовые файлы?
3. Какие основные операции можно совершать с текстовыми данными в программном коде?
4. Применяли ли вы специальные функции к тексту или к его элементам в программном коде? Расскажите о своем опыте.
5. Какие существуют функции для перехода от символа к коду, от текста к числу и наоборот?

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

[Дополнительная информация по синтаксису Python](#)

§2.4 Работа с текстовыми файлами

ВСПОМИНАЕМ

Вспомним:

1. Как с помощью программного кода можно преобразовывать текстовые данные?
2. Какие возможности представляют функции `ord` и `chr`?

ЧИТАЕМ

Чтобы получить из внешнего мира информацию для обработки и показать результат, программа должна иметь средства для ввода данных из файлов и сохранения данных в файлах. Работа с файлами всегда происходит в три этапа:

- Файл открывается в одном из выбранных режимов.
- Выполняется считывание, обновление или удаление данных файла.
- Файл закрывается, иначе можно потерять данные, так как файл может не быть изменен сразу из-за буферизации, процесс переноса данных из буфера регулируется ОС. Закрытие файла необходимо для уверенности, что все необходимые данные записаны в файл.

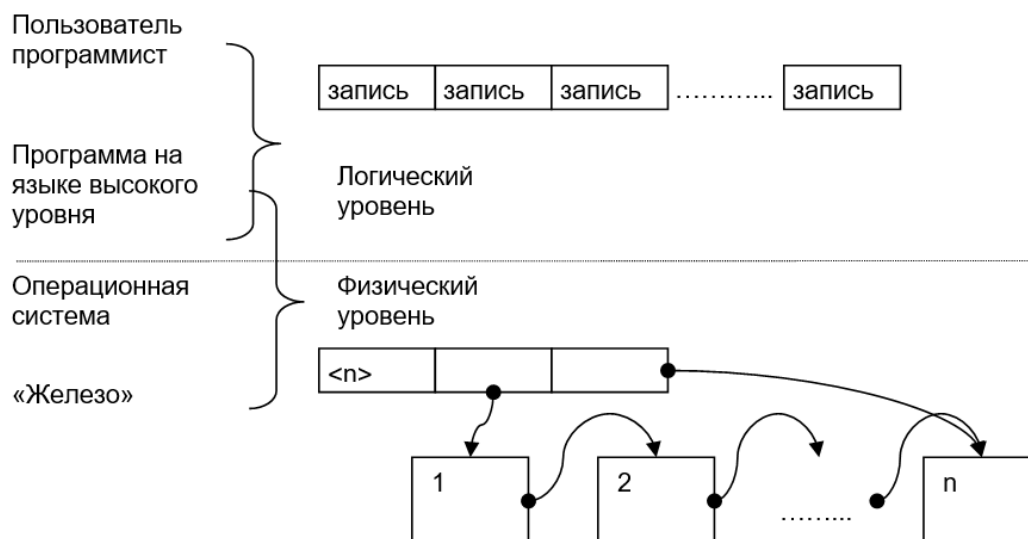


Рис. 52 Структура текстового файла

Для получения возможности чтения данных из файла в Python, его необходимо создать. Это можно сделать стандартными средствами операционной системы, перейдя в нужный каталог и создав там новый текстовый документ. А можно, и с помощью метода `open` в Python, которому надо передать в качестве параметров полное или относительное имя файла и режим его обработки "w" (открытие на запись).

```
f = open("text.txt", "w")
# указано относительное имя,
# файл будет создан в текущей
# папке
... # операции записи в файл
f.close()
```

```
f = open("C:\Users\data\text.txt", "w")
# указано полное имя, файл будет создан в
# папке C:\Users\data
... # операции записи в файл
f.close()
```

Функция `open` возвращает объект `f`, который содержит ссылку на существующий файл. Данный объект несет информацию о файле, представленную в виде четырех основных полей:

| Свойство | Значение |
|--------------------------|---|
| <code>f.name</code> | возвращает полное имя файла |
| <code>f.mode</code> | возвращает режим доступа, в котором был открыт файл |
| <code>f.closed</code> | возвращает <code>true</code> , если файл закрыт |
| <code>f.softspace</code> | возвращает <code>true</code> , если при выводе данных из файла не следует отдельно добавлять символ пробела |

У функции `open` имеется много разных аргументов (<https://pythonz.net/references/named/open/>). Рассмотрим первые три: строка — полное имя файла; строка, обозначающая режим, в котором следует открыть

файл; `encoding` = кодировка символов, он нужен только в текстовом режиме чтения. Рассмотрим возможные значения режима открытия файла:

| Режим | Обозначение |
|-------|--|
| 'r' | открытие на чтение (является значением по умолчанию). |
| 'w' | открытие на запись, содержимое файла удаляется, если файла не существует, создается новый. |
| 'x' | открытие на запись, если файла не существует; исключение <code>FileExistsError</code> , если файл уже существует |
| 'a' | открытие на дозапись, информация добавляется в конец файла. |
| 'b' | открытие в двоичном режиме. |
| 't' | открытие в текстовом режиме (является значением по умолчанию). |
| '+' | открытие на чтение и запись |

После того как файл открыт, мы можем прочитать из него информацию. Есть два способа это сделать:

Метод **`read`**, читает весь файл целиком, если был вызван без аргументов, и `n` символов, если был вызван с аргументом `n`

| Программный код | Что получает переменная <code>t</code> |
|---|--|
| <pre>f = open('text.txt', encoding='utf-8') t = f.read(2)</pre> | 'Пр' |
| <pre>f = open('text.txt', encoding='utf-8') t = f.read()</pre> | 'Привет, МИР!\nЭто я.\n\n' |

Второй способ – прочитать файл по строкам, воспользовавшись циклом **`for`**:

| Программный код | Что получает переменная <code>t</code> |
|--|--|
| <pre>f = open('text.txt', encoding='utf-8') t = [] for line in f: t.append(line)</pre> | ['Привет, мир!\n', '\n', 'Это я.\n', '\n'] |

Запись данных в файл осуществляется следующим образом:

| Программный код | Что получает переменная <code>t</code> |
|-----------------|--|
|-----------------|--|

| | |
|---|---|
| <pre>f = open('text.txt', 'w', encoding='utf-8') f.write('Как прекрасен этот мир!') f.close()</pre> | Как прекрасен этот мир! |
| <pre>f = open('text.txt', 'w') lst = [str(i)+str(i+1) for i in range(10)] for i in lst: f.write(i + '\n') f.close()</pre> | 01 12 23 34 45 56 67 78 89 910 |

Обратим внимание на то, что метод **write** возвращает число записанных символов.

Обработку текстовых файлов можно автоматизировать. Для этого следует использовать конструкцию **with ... as....** Это позволяет не закрывать файл методом **close**. Если файла с указанным именем не обнаружилось, то вложенный в оператор with код выполняться не будет. В следующем примере функция **print** используется для вывода строковой информации на экран. Рассмотрим вывод результата работы программного кода (**rfind** – метод поиска с конца строки, **strip** – метод возвращает копию строки, с обоих концов которой устранены пробельные символы):

| Программный код | Содержимое файла text.txt |
|---|---|
| <pre>with open('text.txt', encoding='utf-8') as inp: for line in inp: last = line[line.rfind(' '):] print(last.strip(), end='')</pre> | У лукоморья дуб зелёный; Златая цепь на дубе том: И днём и ночью кот учёный Всё ходит по цепи кругом; Идёт направо — песнь заводит, Налево — сказку говорит. |

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Задача:

Текстовый файл text.txt содержит только заглавные буквы латинского алфавита (ABC...Z). Определите максимальное количество идущих подряд символов, среди которых нет стоящих рядом букв P и R (в любом порядке).

Возможный вариант решения:

```

with open('text.txt') as F:
    S = F.readline()
max_len = 0
cur_len = 0
for i in range(len(S) - 1):
    if (S[i] == 'P' and S[i + 1] == 'R') or \
        (S[i] == 'R' and S[i + 1] == 'P'):
        cur_len = 1
    else:
        cur_len += 1
        max_len = max(max_len, cur_len)
print(max_len)

```

Результат выполнения программы:

| Исходные данные | Выходные данные |
|-------------------|-----------------|
| RPASPRLKJRPQWERPR | 5 |

Комментарий к программе:

Программа читает содержимое текстового файла как единую строку, затем просматривает пары рядом стоящих символов и, если они не соответствуют заданным сочетаниям 'PR' и 'RP', считает длину подстроки, их содержащей, и изменяет текущий максимум, если в этом есть смысл. Если же сочетания 'PR' или 'RP' обнаружены, то подсчет длины участка начинается заново.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 2

Задача:

В текстовом файле text.txt находится цепочка из символов русского алфавита. Найдите количество подстрок длины 4, удовлетворяющих следующим условиям:

- 1-й символ – прописная буква, обозначающая гласный звук;
- 2-й символ – строчная буква, обозначающая согласный звук;
- 3-й символ – одна из букв, обозначающих шипящий звук;
- 4-й символ – любая буква, не совпадающая с предыдущими.

| Исходные данные | Выходные данные |
|---|-----------------|
| дшпЕршиЦгМАЛЫШИрсрсЁршспшРЩШРЗЩтрвшшрЕршиКарасидлтГ | 3 |

Возможное решение:

```

with open("text.txt", encoding='utf-8') as F:
    S = F.readline()
SL1 = 'АЕЁИОУЫЭЮЯ'

```

```

SL2 = 'бвгджзйклмнпрстфхцчщщ'
SL3 = 'жчшщжчшщ'
k = 0
for i in range(len(S) - 3):
    P = S[i:i + 4]
    if P[0] in SL1:
        if P[1] in SL2:
            if P[2] in SL3:
                if P[3] not in {P[0], P[1], P[2]}:
                    k += 1
print(k)

```

Комментарий к программе:

В программе используются три строки с заданными группами букв и множество из букв для компактной проверки условий выбора подстрок.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №21

Текстовый файл text.txt состоит из цифр и пробелов. Найти сумму всех содержащихся в файле чисел, длина которых не превышает 10 знаков. Числом считать непрерывную последовательность цифр.

| Исходные данные | Выходные данные |
|---|-----------------|
| 12 100 67 33 5 1000000000000000 3 25 958476352435478 | 220 |

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

№ 1

Текстовый файл text.txt состоит строчных латинских букв (a..z). Текст разбит на строки различной длины. Необходимо найти строку, содержащую наименьшее количество подстрок yes (если таких строк несколько, надо взять ту, которая в файле встретилась последней), заменить в ней все 'yes' на 'no'. В качестве результата вывести измененную строку. Гарантируется, что 'yes' присутствует в каждой строке хотя бы раз.

| Исходные данные | Выходные данные |
|--|------------------------------------|
| ufuioh yes klvn xyes jbkkbklnlpk yes bjdbjbks yes esskehojkqhihih oheeuoepr yes ooooooooooooiifkejhe yes w | oheeuoeprnoooooooooooooiifkejhenow |

№ 2

В текстовом файле text.txt находится строка из букв русского алфавита. Найдите количество подстрок длины 3, удовлетворяющих следующим условиям:

- 1-й символ – прописная буква, обозначающая согласный звук;
- 2-й символ – одна из букв, обозначающих гласный звук;
- 3-й символ – одна из букв слова 'селфи' (она может быть как строчной, так и прописной), не встречавшаяся на первом или втором местах.

| Исходные данные | Выходные данные |
|--|-----------------|
| уГГифШщшрвгцшроихршр0роШГРисшрвшощлцджбжФАПРОЕвТЭс | 4 |

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Какие этапы работы с файлами можно выделить?
2. Надо ли всегда предварительно создать файл в текстовом редакторе для того, чтобы открыть его в программном коде?
3. Какие существуют режимы открытия текстового файла?
4. В каких случаях удобно применить использовать конструкцию with ... as...?
5. С какими методами чтения-записи текстовых файлов вы познакомились?

§2.5 Формат CSV. Создание CSV файла, преобразования в Excel формат.

ВСПОМИНАЕМ

Вспомним:

1. Какие особенности обработки текстового файла средствами Python вы знаете?
2. Какие методы чтения и записи текстовых файлов мы использовали?

ЧИТАЕМ

Помимо обычных текстовых документов нередко нам необходимо обрабатывать данные, представленные в виде таблицы. Для нас на экране таблица всегда представляет собой традиционную область со столбцами, строками и границами. Однако мы уже знаем, что компьютер хранит информацию иначе и привычная таблица – просто форма отражения данных.

Хорошим примером работы с такой «настоящей» таблицей служит обработка файлов CSV-формата. CSV расшифровывается как comma-separated values, в переводе на русский – «значения, разделенные запятой». CSV файл –

это простой текстовый файл, который можно открыть даже в простейшем текстовом редакторе и увидеть его содержимое. Значения в таком файле отделяются друг от друга разделителем. Разделителем может быть не только запятая, но и другие символы – точка с запятой, знак табуляции, даже пробел. В России запятая, как правило, используется для разделения целой и дробной части в числах (международный стандарт – точка). Поэтому в файлах CSV в качестве разделителя в России чаще используют знак табуляции или точку с запятой. В результате получаются файлы, которые, строго говоря, имеют формат не CSV, а DSV (англ. delimiter-separated values — значения, разделённые разделителем). Такие файлы открываются в редакторах электронных таблиц как привычная нам таблица, а в текстовых редакторах как текст с разделителем.

Файлы формата CSV позволяют сохранять текстовые данные в формате таблиц, что очень удобно при работе с большими базами данных. Файл CSV можно создать с помощью редакторов электронных таблиц, например таких, как Microsoft Excel, OpenOffice Calc, Google Таблицы. Но специальное ПО не обязательно, подойдет и Блокнот!

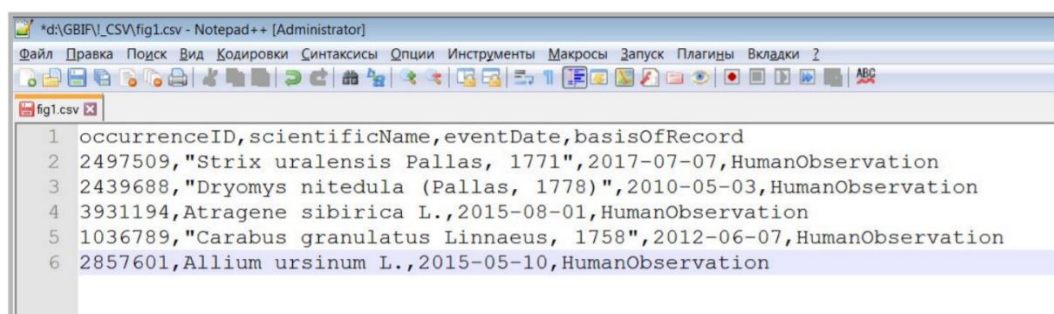


Рис 53* CSV файл открытый в текстовом редакторе

| | A | B | C | D | E |
|---|--------------|-----------------------------------|------------|------------------|---|
| 1 | occurrenceID | scientificName | eventDate | basisOfRecord | |
| 2 | 2497509 | Strix uralensis Pallas, 1771 | 2017-07-07 | HumanObservation | |
| 3 | 2439688 | Dryomys nitedula (Pallas, 1778) | 2010-05-03 | HumanObservation | |
| 4 | 3931194 | Atragene sibirica L. | 2015-08-01 | HumanObservation | |
| 5 | 1036789 | Carabus granulatus Linnaeus, 1758 | 2012-06-07 | HumanObservation | |
| 6 | 2857601 | Allium ursinum L. | 2015-05-10 | HumanObservation | |
| 7 | | | | | |
| 8 | | | | | |
| 9 | | | | | |

Рис 54* Этот же файл, открытый в редакторе электронных таблиц

*Изображения заимствованы http://gbif.ru/files/manuals/best_practice_CSV.pdf

Как выполняется создание CSV файла? Рассмотрим два варианта – в редакторе электронных таблиц и в текстовом редакторе. Рассмотрим первый вариант. Для начала создадим новую электронную таблицу в Excel.

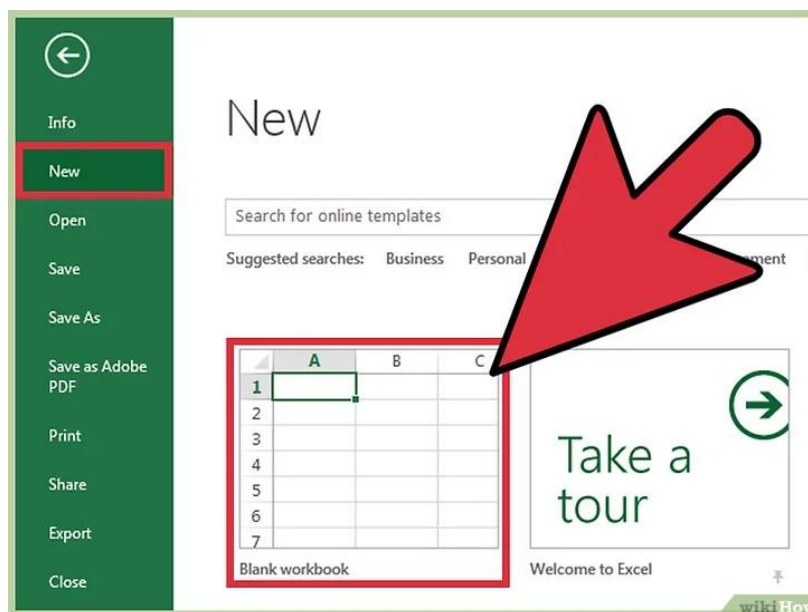


Рис 55** Создание новой электронной таблицы

**Изображения заимствованы <https://ru.wikihow.com>

После этого необходимо создать новые поля и внести в таблицу записи.

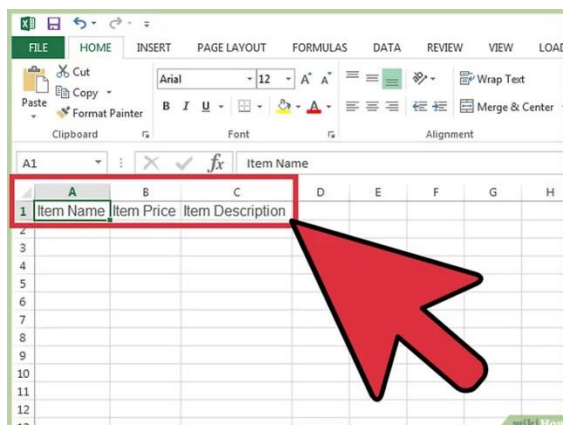


Рис 56** Именованые поля

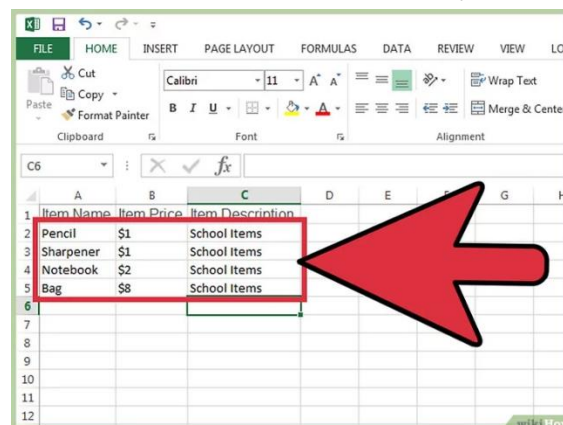


Рис 57** Внесение записей

После того как таблица создана, необходимо сохранить документ, выбрав формат.

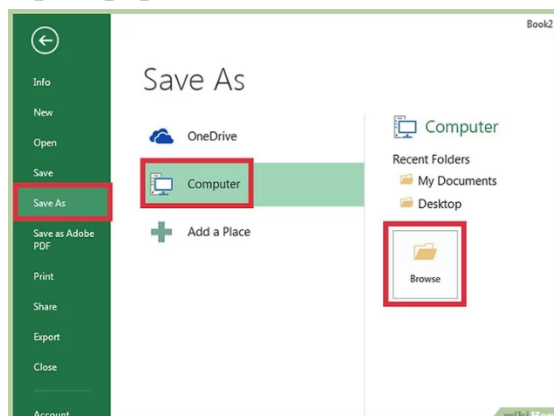


Рис 58** Выбор команды «Сохранить как»

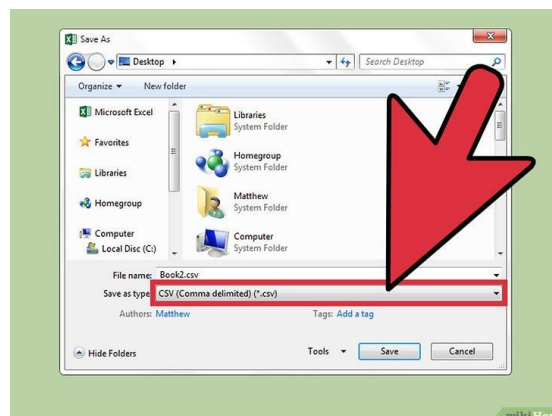


Рис 59** Указать формат файла CSV

Альтернативный способ создания CSV файлов – использование текстового редактора, к примеру, самого простого – Блокнота. Создаём файл, где данные каждой ячейки разделяем запятыми:

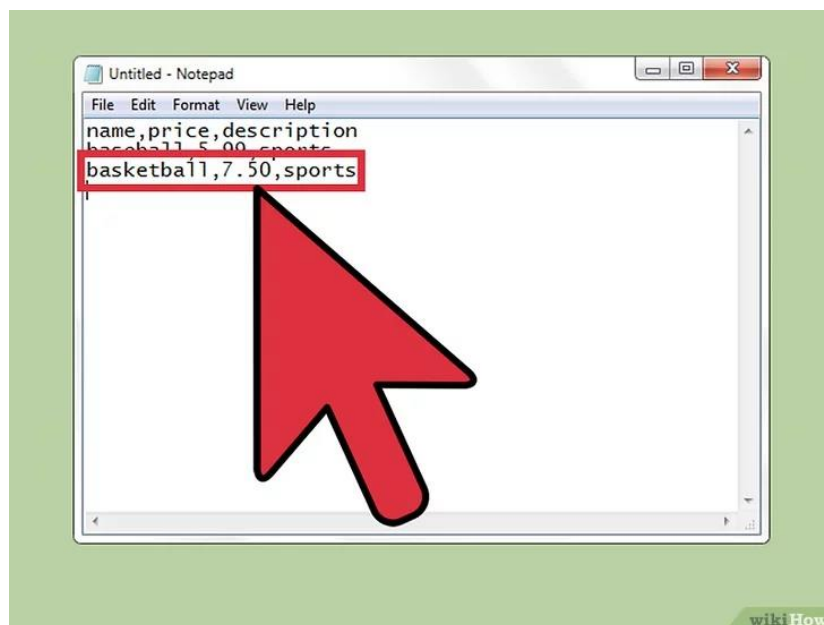


Рис 60** Создание размеченного файла в Блокнот

И так же, как с табличным редактором – сохраните документ, указав формат файла «CSV».

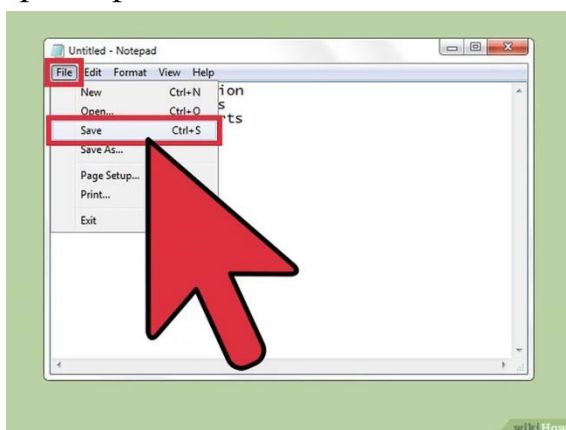


Рис 61** Выбор команды «Сохранить как»

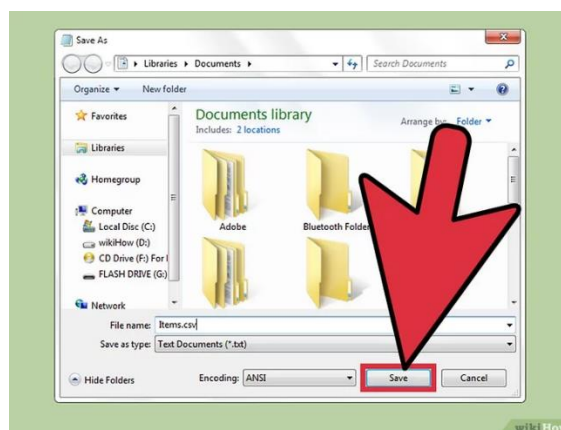


Рис 62** Указать формат файла CSV

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №22

Создайте таблицу 3x3 (любая тематика) в редакторе Блокнот, разделяя данные в строках запятыми. После этого откройте получившийся файл в редакторе электронных таблиц (к примеру, MS Excel).

ЧИТАЕМ

CSV – чрезвычайно популярный формат для хранения табличных данных и обмена ими. Скорее вам придется скачивать откуда-то разные датасеты в CSV, чем самим создавать и сохранять файлы в CSV. Поэтому корректное открытие таких файлов в табличном процессоре является актуальной проблемой для пользователя компьютера. Давайте выясним некоторые нюансы:

- Существует настройка Excel на автоматическое открытие CSV файла, но в этом случае возможно несоответствие кодировки или разделителей.
- Импорт в Excel CSV файла, тогда можно легко установить кодировку, разделители и задать необходимый формат представления данных в электронной книге.

На рисунке 5.11 представлен первый шаг при экспортировании файла CSV:

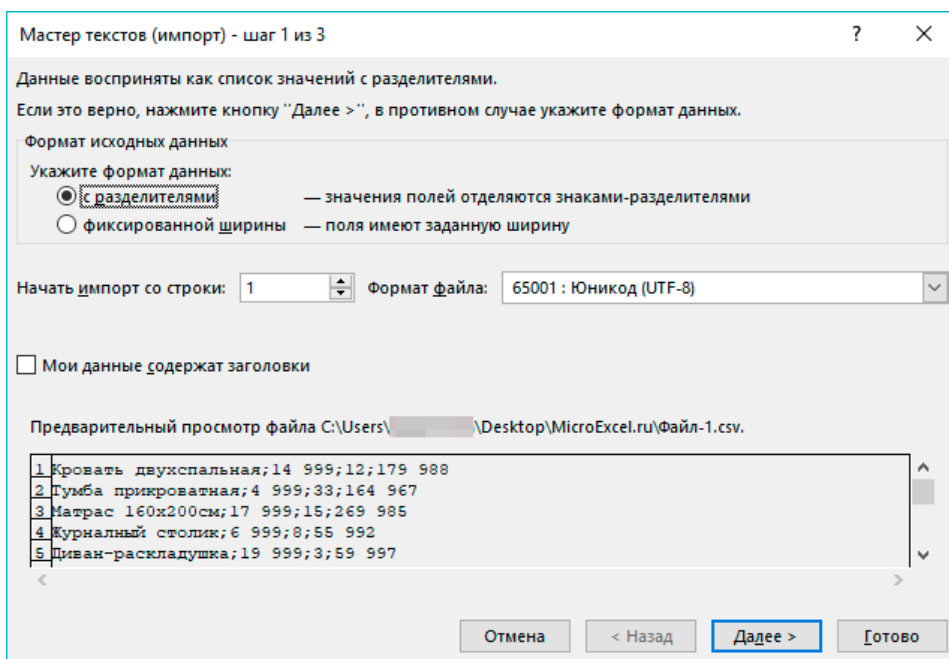


Рис 63*** Экспорт CSV файла, шаг

***Изображения заимствованы <https://microexcel.ru/otkrytie-csv/>

Дальше необходимо выполнить следующие шаги:

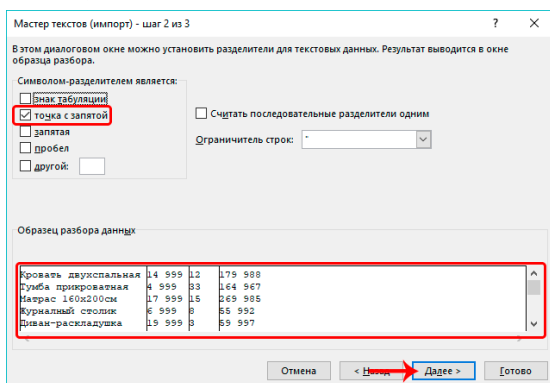


Рис 64** Экспорт CSV файла, шаг 2

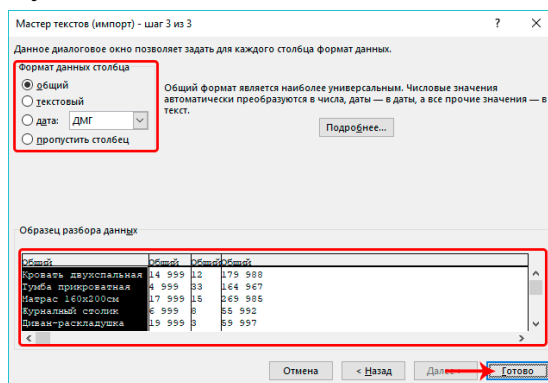


Рис 65** Экспорт CSV файла, шаг 3

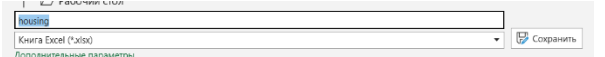

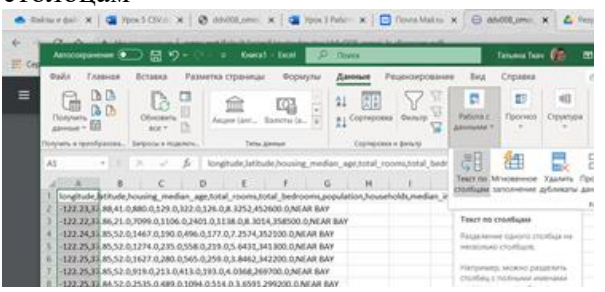
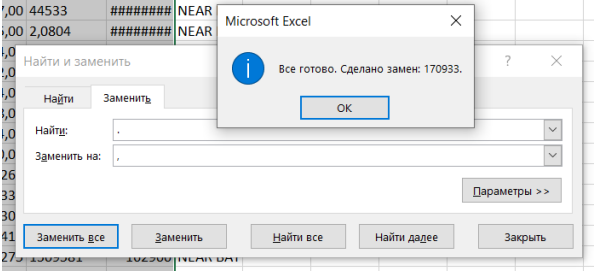
ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №23

Скачайте файл [housing.xlsx](#), необходимый для выполнения задания.

Дальше следуйте инструкции:

| Что сделать? | Как сделать? |
|--------------|--------------|
|--------------|--------------|

| Создайте новый документ, сохраните под именем «housing.xlsx» |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---------------------|--------------|-----------------|------------|--------------------------|-------------|----------|------------|-----------|---------------------|--------------|-----------------|------------|-----------|-------------|-----------|-------------|----------|--------|--------|--------------------------|-------------|----------|---------|---------|---------|---------|--------------------------|-------------|----------|---------|--------|--------|--------|--------------------------|-------------|----------|---------|--------|--------|--------|--------------------------|
| Создайте новый лист, переименуйте его в «housing» |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Из файла housing.csv перенесите содержимое на лист «housing». Закройте файл. | <p>Откройте housing.csv в приложении «Блокнот», выделите весь текст ctrl + A, скопируйте в первую ячейку листа:</p> <table data-bbox="868 613 1474 759"><tr><th>A</th><th>B</th><th>C</th><th>D</th><th>E</th><th>F</th><th>G</th></tr><tr><td>longitude,</td><td>latitude,</td><td>housing_median_age,</td><td>total_rooms,</td><td>total_bedrooms,</td><td>populat</td><td></td></tr><tr><td>-122.23,37,</td><td>88,41.0,</td><td>880.0,</td><td>129.0,</td><td>322.0,</td><td>126.0,</td><td>8.3252,452600.0,NEAR BAY</td></tr><tr><td>-122.22,37,</td><td>86,21.0,</td><td>7099.0,</td><td>1106.0,</td><td>2401.0,</td><td>1138.0,</td><td>8.3014,358500.0,NEAR BAY</td></tr><tr><td>-122.24,37,</td><td>85,52.0,</td><td>1467.0,</td><td>190.0,</td><td>496.0,</td><td>177.0,</td><td>7.2574,352100.0,NEAR BAY</td></tr><tr><td>-122.25,37,</td><td>85,52.0,</td><td>1274.0,</td><td>235.0,</td><td>558.0,</td><td>219.0,</td><td>5.6431,341300.0,NEAR BAY</td></tr></table> <p>Выделите первый столбец</p> | A | B | C | D | E | F | G | longitude, | latitude, | housing_median_age, | total_rooms, | total_bedrooms, | populat | | -122.23,37, | 88,41.0, | 880.0, | 129.0, | 322.0, | 126.0, | 8.3252,452600.0,NEAR BAY | -122.22,37, | 86,21.0, | 7099.0, | 1106.0, | 2401.0, | 1138.0, | 8.3014,358500.0,NEAR BAY | -122.24,37, | 85,52.0, | 1467.0, | 190.0, | 496.0, | 177.0, | 7.2574,352100.0,NEAR BAY | -122.25,37, | 85,52.0, | 1274.0, | 235.0, | 558.0, | 219.0, | 5.6431,341300.0,NEAR BAY |
| A | B | C | D | E | F | G | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| longitude, | latitude, | housing_median_age, | total_rooms, | total_bedrooms, | populat | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| -122.23,37, | 88,41.0, | 880.0, | 129.0, | 322.0, | 126.0, | 8.3252,452600.0,NEAR BAY | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| -122.22,37, | 86,21.0, | 7099.0, | 1106.0, | 2401.0, | 1138.0, | 8.3014,358500.0,NEAR BAY | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| -122.24,37, | 85,52.0, | 1467.0, | 190.0, | 496.0, | 177.0, | 7.2574,352100.0,NEAR BAY | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| -122.25,37, | 85,52.0, | 1274.0, | 235.0, | 558.0, | 219.0, | 5.6431,341300.0,NEAR BAY | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Распределите текст по столбцам соответственно разделителю | <p>Данные -> Работа с данными -> Текст по столбцам</p>  <p>Указать разделитель – «запятая», формат – «текстовый»</p> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Необходимо перевести числовые данные из текстового формата в числовой, заменив точки на запятые. Для этого выделите столбцы A – I | <p>Выделите заголовки, тогда и все столбцы целиком выделяться.</p> <table data-bbox="868 1285 1474 1330"><tr><th>A</th><th>B</th><th>C</th><th>D</th><th>E</th><th>F</th><th>G</th><th>H</th><th>I</th></tr><tr><td>longitude</td><td>latitude</td><td>housing_m</td><td>total_roor</td><td>total_bed</td><td>population</td><td>household</td><td>median_incc</td><td>median_i</td></tr></table> | A | B | C | D | E | F | G | H | I | longitude | latitude | housing_m | total_roor | total_bed | population | household | median_incc | median_i | | | | | | | | | | | | | | | | | | | | | | | | |
| A | B | C | D | E | F | G | H | I | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| longitude | latitude | housing_m | total_roor | total_bed | population | household | median_incc | median_i | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Выполните замену символа «.» на символ «,» |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Измените формат выделенных ячеек на числовой с двумя знаками после запятой | <p>Увидели решетки в столбце? Подвиньте его границу в заголовочной строке вправо.</p> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

После того как действия выполнены, переименуйте *Лист1* в «Графики и диаграммы» и постройте на нем любые три диаграммы разного вида по преобразованным данным. Файл сохраните.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Чем отличается CSV файл от других текстовых файлов? Объясните сходства и различия DSV и CSV форматов.

2. Каково назначение DSV файлов?
3. Какие вы знаете способы создания CSV файла?
4. Какие действия в табличном процессоре надо совершить для правильного сохранения CSV файла?

§2.6 Формат CSV. Обработка средствами программирования

ВСПОМИНАЕМ

Вспомним:

1. Какие средства обработки CSV файлов вы знаете?
2. В чём различия форматов DSV и CSV?

ЧИТАЕМ

Мы убедились ранее, что создавать и представлять в табличном виде DSV-форматы просто. Перенести большой объем данных в Excel не всегда удается, у приложения весьма ограниченные возможности. Данная причина, а также отсутствие четких стандартов в выборе разделителей и [экранировании](#) символов привели к программной обработке CSV файлов при помощи специализированных библиотек. Python для упрощения работы с CSV форматом предоставляет специальный библиотечный модуль csv. В модуле есть два основных объекта: reader и writer. Они необходимы, чтобы читать и создавать CSV-файлы соответственно. Объект reader может обрабатывать различные стили CSV-файлов путем задания дополнительных атрибутов:

- delimiter определяет символ, используемый для разделения каждого поля (элемента столбца). По умолчанию используется запятая (',')
- quotechar определяет символ, используемый для окружения поля. По умолчанию используются двойные кавычки ('"')
- lineterminator определяет последовательность символов, после которой начинается новая строка. По умолчанию используется значение '\r\n'
- quoting, перечислим некоторые его возможные значения:
 - csv.QUOTE_ALL — цитировать (вывести данные полей из таблицы в текстовом виде, то есть в кавычках) все, независимо от типа;
 - csv.QUOTE_MINIMAL — заключить в кавычки поля со специальными символами, такими как delimiter, quotechar;
 - csv.QUOTE_NONNUMERIC — заключить в кавычки все поля, которые не являются числовыми значениями;
 - csv.QUOTE_NONE — ничего не заключать в кавычки.

- `escapechar` указывает символ, используемый для экранирования символа-разделителя, если кавычки не используются.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Задача:

Средствами языка Python открыть и записать новый CSV-файл.

Пример кода:

```
import csv
input_file = open("inp.csv", "rb")
rdr = csv.reader(input_file)
output_file = open("out.csv", "wb")
wrtr = csv.writer(output_file)
for rec in rdr:
    try:
        rec[1] = int(rec[1]) + 1
    except:
        pass
    wrtr.writerow(rec)
input_file.close()
output_file.close()
```

Важно! Если во втором столбце число, то оно увеличится на единицу, иначе ничего не произойдет.

Результат:

Таблица показывает, как изменились данные:

| inp.csv | out.csv |
|-----------------------|-----------------------|
| name,number,text | name,number,text |
| a,1,something here | a,2,something here |
| b,2,"one, two, three" | b,3,"one, two, three" |
| c,3,no commas here | c,4,no commas here |

ПРИМЕР 2

Задача:

Выведем первые 10 строк из таблицы (иначе говоря, выполним цитирование).

Возможное решение:

Используем **with**, чтобы открыть файл `sprint.csv` из папки `files`. Затем, создаем объект **reader**, передаем ему символы-разделители полей и строк. Объект **reader** может служить итератором, т. е. может использоваться в цикле **for**, в котором выполняется перебор строк, каждая представлена в виде списка полей. Функцию **enumerate** добавляем для отсчета десяти строк.

Пример кода:

```
import csv

with open('files/sprint.csv') as f:
    r = csv.reader(f, delimiter=';')
    for index, row in enumerate(r):
        if index > 10:
            break
        print(row)
```

Результат:

| sprint.csv | |
|---|--|
| Name;Country;Medal;Time;Year | |
| "Usain Bolt";"JAM";"GOLD";9.63;2012 | |
| "Yohan Blake";"JAM";"SILVER";9.75;2012 | |
| "Justin Gatlin";"USA";"BRONZE";9.79;2012 | |
| "UsainBolt";"JAM";"GOLD";9.69;2008 | |
| "Richard Thompson";"TRI";"SILVER";9.89;2008 | |
| "Walter Dix";"USA";"BRONZE";9.91;2008 | |
| "Justin Gatlin";"USA";"GOLD";9.85;2004 | |
| "Francis Obikwelu";"POR";"SILVER";9.86;2004 | |
| "Maurice Greene";"USA";"BRONZE";9.87;2004 | |
| "Maurice Greene";"USA";"GOLD";9.87;2000 | |
| "Ato Boldon";"TRI";"SILVER";9.99;2000 | |
| "Obadele Thompson";"BAR";"BRONZE";10.04;2000 | |
| "Donovan Bailey";"CAN";"GOLD";9.84;1996 | |
| "Frankie Fredericks";"NAM";"SILVER";9.89;1996 | |
| "Ato Boldon";"TRI";"BRONZE";9.90;1996 | |
| "Linford Christie";"GBR";"GOLD";9.96;1992 | |
| "Frankie Fredericks";"NAM";"SILVER";10.02;1992 | |
| "Dennis Mitchell";"USA";"BRONZE";10.04;1992 | |
| "CarlLewis";"USA";"GOLD";9.92;1988 | |
| Экран | |
| ['Name', 'Country', 'Medal', 'Time', 'Year'] | |
| ['Usain Bolt', 'JAM', 'GOLD', '9.63', '2012'] | |
| ['Yohan Blake', 'JAM', 'SILVER', '9.75', '2012'] | |
| ['Justin Gatlin', 'USA', 'BRONZE', '9.79', '2012'] | |
| ['UsainBolt', 'JAM', 'GOLD', '9.69', '2008'] | |
| ['Richard Thompson', 'TRI', 'SILVER', '9.89', '2008'] | |
| ['Walter Dix', 'USA', 'BRONZE', '9.91', '2008'] | |
| ['Justin Gatlin', 'USA', 'GOLD', '9.85', '2004'] | |
| ['Francis Obikwelu', 'POR', 'SILVER', '9.86', '2004'] | |
| ['Maurice Greene', 'USA', 'BRONZE', '9.87', '2004'] | |
| ['Maurice Greene', 'USA', 'GOLD', '9.87', '2000'] | |

ПРИМЕР 3

Задача:

Отсортируем строки заданной таблицы.

Возможное решение:

Для корректного результата исключим из обработки первую строку. А названия полей сохраним в виде ключей в словарях. Таким образом, каждой строке табличных данных соответствует свой словарь с данными и все словари хранятся в списке.

Пример кода:

```
import csv

with open('files/sprint.csv') as f:
    r = csv.DictReader(f, delimiter=';')
    sortr = sorted(r, key=lambda x: float(x['Time']))

for record in sortr[:10]:
    print(record)
```

Важно: Обратим внимание на то, что для создания словарей был вызван специальный объект **DictReader**.

Результат:

| sprint.csv |
|---|
| {'Name': 'Usain Bolt', 'Country': 'JAM', 'Medal': 'GOLD', 'Time': '9.63', 'Year': '2012'} |
| {'Name': 'UsainBolt', 'Country': 'JAM', 'Medal': 'GOLD', 'Time': '9.69', 'Year': '2008'} |
| {'Name': 'Yohan Blake', 'Country': 'JAM', 'Medal': 'SILVER', 'Time': '9.75', 'Year': '2012'} |
| {'Name': 'Justin Gatlin', 'Country': 'USA', 'Medal': 'BRONZE', 'Time': '9.79', 'Year': '2012'} |
| {'Name': 'Donovan Bailey', 'Country': 'CAN', 'Medal': 'GOLD', 'Time': '9.84', 'Year': '1996'} |
| {'Name': 'Justin Gatlin', 'Country': 'USA', 'Medal': 'GOLD', 'Time': '9.85', 'Year': '2004'} |
| {'Name': 'Francis Obikwelu', 'Country': 'POR', 'Medal': 'SILVER', 'Time': '9.86', 'Year': '2004'} |
| {'Name': 'Maurice Greene', 'Country': 'USA', 'Medal': 'BRONZE', 'Time': '9.87', 'Year': '2004'} |
| {'Name': 'Maurice Greene', 'Country': 'USA', 'Medal': 'GOLD', 'Time': '9.87', 'Year': '2000'} |
| {'Name': 'Richard Thompson', 'Country': 'TRI', 'Medal': 'SILVER', 'Time': '9.89', 'Year': '2008'} |

ПРИМЕР 4

Задача:

Осуществить запись в CSV-файл с помощью объекта **DictWriter** (по аналогии с DictReader).

Возможное решение:

В данном примере происходит создание файла из списка словарей. Для объекта writer заданы атрибуты для установки разделителя, символов для перехода к новой строке и кавычек для полей, содержащих специальные символы.

Пример кода:

```
import csv
```



```

data = [{
    'язык': 'Питон',
    'автор': 'Гвидо ван Россум',
    'год': 1991,
    'тип': '.py\n'},
    {'язык': 'Джава',
    'автор': 'Джеймс Гослинг',
    'год': 1995,
    'тип': '.java\n'},
    {'язык': 'C ++',
    'автор': 'Бьярн Страуструп',
    'год': 1983,
    'тип': '.cpp\n'}]

with open('files\lang.csv', 'w', encoding='utf-8') as f:
    writer = csv.DictWriter(f, fieldnames=list(data[0].keys()),
        delimiter=';',
                                lineterminator='\n',
                                quoting=csv.QUOTE_MINIMAL)

    writer.writeheader()
    for d in data:
        writer.writerow(d)

```

Результат:

| lang.csv |
|---|
| язык;автор;год;тип Питон;Гвидо ван Россум;1991;".py " |
| Джава;Джеймс Гослинг;1995;".java " |
| C ++;Бьярн Страуструп;1983;".cpp " |

ГОТОВИМСЯ К РАБОТЕ НА ПК

Для выполнения практических заданий Создайте data.csv файл с данными:

| |
|---|
| Programming language,Designed by,Appeared,Extension Python,Guido van Rossum,91,.py Java,James Gosling,95,.java C++,Bjarne Stroustrup,83,.cpp |
|---|

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №24

Создайте программу, которая, используя методы csv.reader() и csv.writer(), считывает данные data.csv файла и обрабатывает их так:

- редактирует годы в третьем столбце: 91, 95, 83 преобразует в 1991, 1995, 1983
- редактирует написание первой колонки, преобразует написание названий языков в написание большими буквами
- добавляет данные о языке javascript:

JAVASCRIPT, Brendan Eich, 1995, .js, C#, Microsoft, 2000, .cs

- выводит результат в новый файл: data1.csv

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №25

Создайте программу, которая, используя методы `csv.DictReader()` и `csv.DictWriter()`, считывает данные data.csv файла и обрабатывает данные:

- удаляет последний столбец
- строку

C++, Bjarne Stroustrup, 1983, .cpp

делает пустой, то есть все поля в строке делает пустыми

- добавляет данные о языке javascript:

JAVASCRIPT, Brendan Eich, 1995, .js, C#, Microsoft, 2000

- выполняет обратную сортировку строк по алфавиту по названию языка
- выводит все текстовые данные из списка словарей в кавычках
- выводит результат в новый файл: data2.csv

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №26

Создайте csv-файл «rows_10.csv», в котором 10 строк и следующие столбцы: имя, возраст, рост, вес, вид спорта. Для этого можно использовать свою программу, электронную таблицу, а можно и простой редактор Блокнот. Заменить разделитель полей ',' на любой другой символ, удовлетворить запросы пользователя на изменение роста или веса по заданному имени и параметрам, вывести в новый файл, дублируя каждую строку с четным номером.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Какой библиотечный модуль используется для создания и обработки csv-файлов?
2. Какие атрибуты имеет объект reader?
3. Какую структуру данных надо использовать для загрузки данных в csv-файл?

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. [Документация по языку Python3](#)
2. [Как читать и писать CSV-файлы в Python](#)
3. [Уроки Python / Работаем с CSV файлами \(считываем и записываем данные\)](#)
4. [Работа с файлами в формате CSV](#)
5. [Файлы CSV](#)

§2.7 Работа с табличными файлами

ВСПОМИНАЕМ

Вспомним:

1. Какие программы обычно используются для работы с таблицами?
2. Какие приёмы работы с данными в табличном редакторе мы помним?

ЧИТАЕМ

Обычно работа с табличными данными ведется в редакторах электронных таблиц, таких как, например, MS Excel. Кое-что о работе с табличным редактором мы уже знаем. К примеру:

- У каждой ячейки есть адрес, состоящий из имени столбца и номера строки, например, F28
- Ячейка может содержать данные разных форматов – число, текст, дату, формулу и др.
- Запись C7:D18 означает диапазон ячеек, то есть все ячейки, заключенные внутри прямоугольника, ограниченного ячейками с адресами C7 и D18
- Любая формула начинается знаком «=», содержит последовательность операндов и операций (+, −, *, / и ^)
- Операндом в формуле может быть не только числовое или текстовое или какое-либо другое константное значение, но и ссылка на ячейку(и) или вызов функции. Например, по формуле =СУММ(A1:A4)^2 вычисляется квадрат суммы значений диапазона ячеек от A1 до A4.

Кроме того, мы наверняка помним, что ссылки на ячейки (способы обращения к ячейкам) бывают трёх видов – относительные, абсолютные и смешанные. Отличие проявляется при копировании формулы в документе и представлено на схеме:



Рис 66 Типы ссылок электронной таблицы

Помимо арифметических действий над данными в таблице можно проводить и логические действия «И», «ИЛИ», «НЕ» а также использовать функции СЧЕТ (количество непустых ячеек), СУММ (сумма), СРЗНАЧ (среднее значение), МИН (минимальное значение), МАКС (максимальное значение) и их вариации, такие как СУММЕСЛИ (сумма с проверкой условия) или СРЗНАЧЕСЛИМН (среднее значение с проверкой нескольких значений). Эти функции можно вписывать вручную или вызывать с помощью мастера функций.

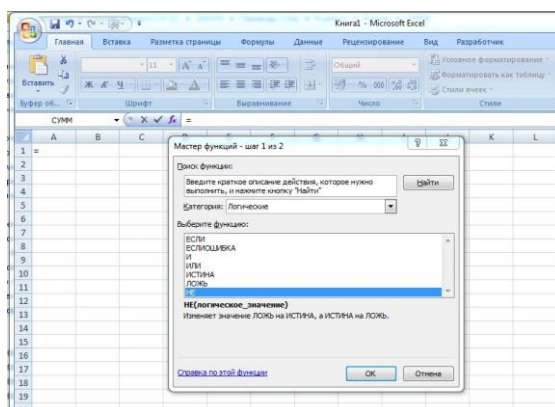


Рис 67 Мастер функций

Встроенный инструмент редактора электронных таблиц позволяет также проводить фильтрацию или сортировку по одному или нескольким критериям.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Задача:

В ячейке A1 электронной таблицы находилась формула $=\$E1*D\3 . Бот обновил ячейки диапазона B1:F6 новыми числами и скопировал по ошибке формулу в ячейку C4, как показано на рисунке. Какое число окажется в ячейке C4?

| | A | B | C | D | E | F |
|---|--------------|---|----|----|----|----|
| 1 | $=\$E1*D\3 | 1 | 10 | 20 | 30 | 40 |
| 2 | | 2 | 11 | 21 | 31 | 41 |
| 3 | | 3 | 12 | 22 | 32 | 42 |
| 4 | | 4 | 13 | 23 | 33 | 43 |
| 5 | | 5 | 14 | 24 | 34 | 44 |
| 6 | | 6 | 15 | 25 | 35 | 45 |

Решение:

Как мы помним, знак \$ закрепляет положение операнда формулы по вертикали или по горизонтали. Следовательно, ссылка \$E1 перейдет в \$E4, а ссылка D\$3 перейдет в F\$3. Ответ ясен – это 1386 ($E4 * F3 = 33 * 42 = 1386$).

ПРИМЕР 2

Задача:

Бот заполнил лист электронной таблицы 1 000 000 случайных вещественных числами. Составьте формулу, которая в диапазоне FA9:ZB525 найдет количество чисел, не принадлежащих отрезку [17; 3000].

Решение:

Для подсчета значений по признаку используется функция СЧЁТЕСЛИ. В данном случае ее следует применить дважды:

=СЧЁТЕСЛИ(FA9:ZB525, "<17")+СЧЁТЕСЛИ(FA9:ZB525, ">3000")

РЕШАЕМ В ТЕТРАДИ

Задание 14.

Дан фрагмент электронной таблицы:

| | A | B | C |
|---|---|---|--------------|
| 1 | 2 | 2 | |
| 2 | 4 | 1 | =A2 + 2*B\$2 |

Чему станет равным значение ячейки C1, если туда скопировать формулу из ячейки C2?

РЕШАЕМ В ТЕТРАДИ

Задание 15.

Дан фрагмент электронной таблицы:

| | A | B | C | D |
|---|---|---|---|---|
| 1 | 6 | 2 | 3 | |
| 2 | 5 | 1 | 2 | |

В ячейку D2 введена формула =A2*B1+C1. Какое значение появится в ячейке D2 в результате вычисления формулы?

РЕШАЕМ В ТЕТРАДИ

Задание 16.

В ячейке B2 записана формула =\$D\$2+E2. Какой вид будет иметь формула, если ячейку B2 скопировать в ячейку A1?

1) =\$D\$2+E1 2) =\$D\$2+C2 3) =\$D\$2+D2 4) =\$D\$2+D1

РЕШАЕМ В ТЕТРАДИ

Задание 17.

В ячейке B3 электронной таблицы записана формула =\$A\$1+B1. Какой вид будет иметь формула, если ячейку B3 скопировать в ячейку C3?

1) = $\$A\$1+C1$ 2) = $\$B\$1+B3$ 3) = $\$A\$1+B3$ 4) = $\$B\$1+C1$

РЕШАЕМ В ТЕТРАДИ

Задание 18.

В электронной таблице значение формулы =СРЗНАЧ(А6:С6) равно (-2). Чему равно значение формулы =СУММ(А6:D6), если значение ячейки D6 равно 5?

1) 1 2) -1 3) -3 4) 7

РЕШАЕМ В ТЕТРАДИ

Задание 19.

В электронной таблице значение формулы =СРЗНАЧ(А6:С6) равно 0,1. Чему равно значение формулы =СУММ(А6:D6), если значение ячейки D6 равно (-1)?

1) - 0,7 2) -0,4 3) 0,9 4) 1,1

РЕШАЕМ В ТЕТРАДИ

Задание 20.

Дан фрагмент электронной таблицы:

| | A | B | C | D | E | F |
|---|---|----|---|---|---|---|
| 1 | 1 | 3 | 4 | 8 | 2 | 0 |
| 2 | 2 | -5 | 5 | 2 | 3 | 4 |
| 3 | 5 | 4 | 5 | 5 | 4 | 2 |
| 4 | 2 | 3 | 1 | 4 | 5 | 3 |

Определите, чему будет равно значение, вычисленное по следующей формуле =СУММ(B1:C4) + F2 * E4 - A3?

ЧИТАЕМ

Табличными данными можно легко оперировать в редакторах электронных таблиц. Но при автоматизации процессов обработки и использовании большого количества данных используют среды программирования. К примеру, мы можем перевести таблицы из формата .xlsx в структуры Python и обрабатывать их программным способом.

Для начала нам понадобится загрузить модуль pandas:

```
import pandas as pd
```

Модуль содержит возможности чтения и выгрузки данных из электронной таблицы. К примеру, вот такой:

| № | Флаг | Буква | Страна | Столица | Площадь км ² | Население | Расположение |
|---|---|-------|-------------|---------|-------------------------|-------------|--------------|
| 1 |  | Р | Россия | Москва | 17 125 191 | 146 748 590 | Европа |
| 2 |  | К | Казахстан | Астана | 2 724 902 | 18 823 490 | Азия |
| 3 |  | У | Узбекистан | Ташкент | 447 400 | 34 532 112 | Азия |
| 4 |  | Б | Беларусь | Минск | 207 600 | 9 504 704 | Европа |
| 5 |  | К | Киргизия | Бишкек | 199 900 | 6 523 500 | Азия |
| 6 |  | Т | Таджикистан | Душанбе | 143 100 | 9 127 000 | Азия |
| 7 |  | А | Азербайджан | Баку | 86 000 | 9 981 500 | Азия |
| 8 |  | А | Армения | Ереван | 29 843 | 2 962 100 | Азия |
| 9 |  | М | Молдова | Кишинёв | 29 680 | 2 681 735 | Европа |

Рис 68 Таблица «Страны СНГ»

Используя библиотеку **pandas**, переведем основную часть таблицы в формат Excel и прочитаем ее средствами Python. Пример выполнения:

Заполнение одного листа

```
In [2]: print("\n\t\tСТРАНЫ СНГ")
df = pd.DataFrame({'Страна': ['Россия', 'Казахстан', 'Узбекистан',
                              'Беларусь', 'Киргизия', 'Таджикистан', 'Азербайджан', 'Армения', 'Молдова'],
                  'Столица': ['Москва', 'Астана', 'Ташкент', 'Минск', 'Бешкек', 'Душанбе', 'Баку', 'Ереван', 'Кишенев'],
                  'Площадь': [17125191, 2724902, 447400, 207600, 199900, 143100, 86000, 29843, 29680],
                  'Население': [146748590, 18823490, 34532112, 9504704, 6523500, 9127000, 9981500, 2962100, 2681735],
                  'Расположение': ['Европа', 'Азия', 'Азия', 'Европа', 'Азия', 'Азия', 'Азия', 'Азия', 'Европа']})
df.to_excel('sng1.xlsx')
df.to_excel('d:\ML\sng2.xlsx', sheet_name='СНГ', index=False)
df
```

СТРАНЫ СНГ

| | Страна | Столица | Площадь | Население | Расположение |
|---|-------------|---------|----------|-----------|--------------|
| 0 | Россия | Москва | 17125191 | 146748590 | Европа |
| 1 | Казахстан | Астана | 2724902 | 18823490 | Азия |
| 2 | Узбекистан | Ташкент | 447400 | 34532112 | Азия |
| 3 | Беларусь | Минск | 207600 | 9504704 | Европа |
| 4 | Киргизия | Бешкек | 199900 | 6523500 | Азия |
| 5 | Таджикистан | Душанбе | 143100 | 9127000 | Азия |
| 6 | Азербайджан | Баку | 86000 | 9981500 | Азия |
| 7 | Армения | Ереван | 29843 | 2962100 | Азия |
| 8 | Молдова | Кишенев | 29680 | 2681735 | Европа |

Рис 69 Заполнение листа формата EXCEL

Рассмотрим для работы еще одну таблицу, она содержит сведения о результатах игр в чемпионате России по футболу:

| Место | Команда | И | В | Н | П | М | О |
|-------|----------------|----|----|----|----|-------|----|
| 1 | Зенит | 30 | 22 | 6 | 2 | 65-18 | 72 |
| 2 | Локомотив | 30 | 16 | 9 | 5 | 41-29 | 57 |
| 3 | Краснодар | 30 | 14 | 10 | 6 | 49-30 | 52 |
| 4 | ЦСКА | 30 | 14 | 8 | 8 | 43-29 | 50 |
| 5 | Ростов | 30 | 12 | 9 | 9 | 45-50 | 45 |
| 6 | Динамо | 30 | 11 | 8 | 11 | 27-30 | 41 |
| 7 | Спартак | 30 | 11 | 6 | 13 | 35-33 | 39 |
| 8 | Арсенал Т | 30 | 11 | 5 | 14 | 37-41 | 38 |
| 9 | Уфа | 30 | 8 | 14 | 8 | 22-24 | 38 |
| 10 | Рубин | 30 | 8 | 11 | 11 | 18-28 | 35 |
| 11 | Урал | 30 | 9 | 8 | 13 | 36-53 | 35 |
| 12 | Сочи | 30 | 8 | 9 | 13 | 40-39 | 33 |
| 13 | Ахмат | 30 | 7 | 10 | 13 | 27-46 | 31 |
| 14 | Тамбов | 30 | 9 | 4 | 17 | 37-41 | 31 |
| 15 | Крылья Советов | 30 | 8 | 7 | 15 | 33-40 | 31 |
| 16 | Оренбург | 30 | 7 | 6 | 17 | 28-52 | 27 |

Рис 70 Таблица «Футбольный чемпионат»

Таблицу легко разделить на три группы по общему количеству очков. Чтобы нагляднее представить информацию о каждой группе участников, выделим для каждой из них отдельный лист рабочей книги и выведем таблицу о второй группе:

Заполнение нескольких листов

```

В [18]: gr1 = pd.DataFrame({'Команда': ['Зенит', 'Локомотив', 'Краснодар', 'ЦСКА', 'Ростов'],
                           'В': [22, 16, 14, 14, 12],
                           'Н': [6, 9, 10, 8, 9],
                           'П': [2, 5, 6, 8, 9],
                           'М': ['65-18', '41-29', '49-30', '43-29', '45-50'],
                           'О': [72, 57, 52, 50, 45]})
gr2 = pd.DataFrame({'Команда': ['Динамо', 'Спартак', 'Арсенал-Т', 'Уфа', 'Рубин', 'Урал', 'Сочи'],
                    'В': [11, 11, 11, 8, 8, 9, 8],
                    'Н': [8, 6, 5, 14, 11, 8, 9],
                    'П': [11, 13, 14, 8, 11, 13, 13],
                    'М': ['27-30', '35-33', '37-41', '22-24', '18-28', '36-53', '40-39'],
                    'О': [41, 39, 38, 38, 35, 35, 33]})
gr3 = pd.DataFrame({'Команда': ['Ахмат', 'Тамбов', 'Крылья Советов', 'Оренбург'],
                    'В': [7, 9, 8, 7],
                    'Н': [10, 4, 7, 6],
                    'П': [13, 17, 15, 17],
                    'М': ['27-46', '37-41', '33-40', '28-52'],
                    'О': [31, 31, 31, 27]})

sheet = {'Group1': gr1, 'Group2': gr2, 'Group3': gr3}
writer = pd.ExcelWriter('football.xlsx', engine='xlsxwriter')
for name in sheet.keys():
    sheet[name].to_excel(writer, sheet_name=name, index=False)
writer.save()

```

Рис 71 Заполнение нескольких листов формата EXCEL

Табличные данные из рабочей книги можно читать частично. Изучим примеры:

Чтение таблиц

```
B [2]: table = pd.read_excel('football.xlsx', sheet_name='Group2', usecols=[0, 1, 3, 6])
table
```

```
Out[2]:
```

| | Команда | В | П |
|---|-----------|----|----|
| 0 | Динамо | 11 | 11 |
| 1 | Спартак | 11 | 13 |
| 2 | Арсенал-Т | 11 | 14 |
| 3 | Уфа | 8 | 8 |
| 4 | Рубин | 8 | 11 |
| 5 | Урал | 9 | 13 |
| 6 | Сочи | 8 | 13 |

```
B [2]: table = pd.read_excel('d:\ML\sng2.xlsx')
table.head()
```

```
Out[2]:
```

| | Страна | Столица | Площадь | Население | Расположение |
|---|------------|---------|----------|-----------|--------------|
| 0 | Россия | Москва | 17125191 | 146748590 | Европа |
| 1 | Казахстан | Астана | 2724902 | 18823490 | Азия |
| 2 | Узбекистан | Ташкент | 447400 | 34532112 | Азия |
| 3 | Беларусь | Минск | 207600 | 9504704 | Европа |
| 4 | Киргизия | Бешкек | 199900 | 6523500 | Азия |

Рис 72 Частичное чтение таблиц из файла EXCEL

В первом примере мы сделали выборку столбцов, во втором – выполнили вывод первых пяти строк.

Далее рассмотрим примеры редактирования выгруженной таблицы и запись отредактированной таблицы в новый файл:

Редактирование таблиц

```
B [4]: headers = table.columns.ravel()
sl = table.to_dict()
for i in sl:
    sl[i] = list(sl[i].values())
print('Excel Sheet to Dict:\n', sl)
# увеличить на 1000 население в Белоруси
# удалить информацию о Казахстане
# добавить еще одну строку с любыми данными
ind1 = sl['Страна'].index('Беларусь')
ind2 = sl['Страна'].index('Казахстан')
sl['Население'][ind1] += 1000
for i in sl:
    del sl[i][ind2]
country, capital, square, population, location = input("Введите страну: "), \
input("Введите столицу: "), int(input("Введите площадь: ")), \
int(input("Введите численность населения: ")), input("Введите континент: ")
sl[headers[0]][ind2], sl[headers[1]][ind2], sl[headers[2]][ind2], sl[headers[3]][ind2], sl[headers[4]][ind2] = \
country, capital, square, population, location
print('Excel Sheet new to Dict:\n', sl)
```

Рис 73 Редактирование таблицы формата EXCEL

```
Excel Sheet to Dict:
{'Страна': ['Россия', 'Казахстан', 'Узбекистан', 'Беларусь', 'Киргизия', 'Таджикистан', 'Азербайджан', 'Армения', 'Молдова'],
 'Столица': ['Москва', 'Астана', 'Ташкент', 'Минск', 'Бешкек', 'Душанбе', 'Баку', 'Ереван', 'Кишинеv'], 'Площадь': [17125191, 27
24902, 447400, 207600, 199900, 143100, 86000, 29843, 29680], 'Население': [146748590, 18823490, 34532112, 9504704, 6523500, 912
7000, 9981500, 2962100, 2681735], 'Расположение': ['Европа', 'Азия', 'Азия', 'Европа', 'Азия', 'Азия', 'Азия', 'Европ
а']}
Введите страну: А
Введите столицу: Б
Введите площадь: 1
Введите численность населения: 2
Введите континент: Д
Excel Sheet new to Dict:
{'Страна': ['Россия', 'А', 'Беларусь', 'Киргизия', 'Таджикистан', 'Азербайджан', 'Армения', 'Молдова'], 'Столица': ['Москва',
'Б', 'Минск', 'Бешкек', 'Душанбе', 'Баку', 'Ереван', 'Кишинеv'], 'Площадь': [17125191, 1, 207600, 199900, 143100, 86000, 29843,
29680], 'Население': [146748590, 2, 9505704, 6523500, 9127000, 9981500, 2962100, 2681735], 'Расположение': ['Европа', 'Д', 'Евр
опа', 'Азия', 'Азия', 'Азия', 'Европа']}
```

Рис 74 Результаты обработки структуры данных

Запись после редактирования

```
B [9]: writer = pd.ExcelWriter('d:\ML\sng3.xlsx', engine='xlsxwriter')
pd.DataFrame(s1).to_excel('d:\ML\sng3.xlsx', index=False)
```

```
B [10]: table = pd.read_excel('d:\ML\sng3.xlsx')
table.head()
```

```
Out[10]:
```

| | Страна | Столица | Площадь | Население | Расположение |
|---|-------------|---------|----------|-----------|--------------|
| 0 | Россия | Москва | 17125191 | 146748590 | Европа |
| 1 | A | Б | 1 | 2 | Д |
| 2 | Беларусь | Минск | 207600 | 9505704 | Европа |
| 3 | Киргизия | Бешкек | 199900 | 6523500 | Азия |
| 4 | Таджикистан | Душанбе | 143100 | 9127000 | Азия |

Рис 75 Запись в EXCEL формат после редактирования и проверка записи

Обратим внимание, что если не нужно использовать заголовочную строку, то можно выгружать данные без нее. А если необходима обработка данных только одного столбца, их легко получить:

С заголовками и без, выбор столбца

```
B [21]: excel_data_df = pd.read_excel('d:\ML\sng3.xlsx', header=None)
print(excel_data_df)
excel_data_df = pd.read_excel('d:\ML\sng3.xlsx')
print()
print(excel_data_df)
print('\n', *excel_data_df['Страна'].tolist())
```

| | 0 | 1 | 2 | 3 | 4 |
|---|-------------|---------|----------|-----------|--------------|
| 0 | Страна | Столица | Площадь | Население | Расположение |
| 1 | Россия | Москва | 17125191 | 146748590 | Европа |
| 2 | A | Б | 1 | 2 | Д |
| 3 | Беларусь | Минск | 207600 | 9505704 | Европа |
| 4 | Киргизия | Бешкек | 199900 | 6523500 | Азия |
| 5 | Таджикистан | Душанбе | 143100 | 9127000 | Азия |
| 6 | Азербайджан | Баку | 86000 | 9981500 | Азия |
| 7 | Армения | Ереван | 29843 | 2962100 | Азия |
| 8 | Молдова | Кишинеу | 29680 | 2681735 | Европа |

| | Страна | Столица | Площадь | Население | Расположение |
|---|-------------|---------|----------|-----------|--------------|
| 0 | Россия | Москва | 17125191 | 146748590 | Европа |
| 1 | A | Б | 1 | 2 | Д |
| 2 | Беларусь | Минск | 207600 | 9505704 | Европа |
| 3 | Киргизия | Бешкек | 199900 | 6523500 | Азия |
| 4 | Таджикистан | Душанбе | 143100 | 9127000 | Азия |
| 5 | Азербайджан | Баку | 86000 | 9981500 | Азия |
| 6 | Армения | Ереван | 29843 | 2962100 | Азия |
| 7 | Молдова | Кишинеу | 29680 | 2681735 | Европа |

Россия A Беларусь Киргизия Таджикистан Азербайджан Армения Молдова

Рис 76 Чтение данных из EXCEL формата с настройками

Следующую задачу решим двумя способами – в электронной таблице и через программный код.

Многие люди говорят, что гендерный разрыв в уровне доходов в стране S преувеличен, а некоторые говорят, что неравенство в рабочей силе осталось в прошлом. Есть ли вообще гендерный разрыв? В одних отраслях он сильнее, чем в других? В таблице представлен набор данных, который показывает средний общий и средний недельный доход для 556 различных профессий. Доходы разбиты на статистику по мужчинам и женщинам, которым предшествует средний доход с учетом обоих полов. В какой отрасли наибольший доход? В какой отрасли средняя разница доходов мужчин и женщин наименьшая? Получить четыре ответа для общего дохода и общего дохода за неделю, а также для этих же видов дохода по гендерному принципу.

Примечание:

Если в некоторых численных ячейках маркировано "Na", это значит, что данные не установлены, в основном это относится к недельному доходу, учитываем только числовые данные. Если получено несколько отраслей, то в качестве ответа надо выбрать ту, в которой доход наибольший.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №27

Для выполнения задания откроем в MS Excel файл «Практическая работа № 27».

Что сделать?

Находим максимальное значение в столбце В

Находим максимальное значение в столбце С

В столбцах Н и I находим название отраслей с общим максимальным доходом и выделяем красным цветом

Находим разницу между доходами мужчин и женщин в среднем общем доходе. Копируем формулу вниз до конца таблицы

Как сделать?

=МАКС(B3:B558)

| | G | H |
|----------|-------|---|
| F_weekly | 1 | 2 |
| | 13894 | 2 |
| 1836 | n | 1 |

=МАКС(C3:C558)

| H | I |
|-------|------|
| 1 | 2 |
| 13894 | 2041 |

=ЕСЛИ(B3=H\$2;A3;0)

| G | H |
|----------|-------|
| F_weekly | 1 |
| | 13894 |
| 1836 | 0 |

=ЕСЛИ(И(F3<>"Na";D3<>"Na");ABS(D3-F3);"

| J |
|----------------------------|
| M_workers - F_workers |
| ответы выделены красным!!! |
| 480 |
| 419 |
| 1 |
| 3 |
| 192 |

Находим разницу между доходами мужчин и женщин в среднем недельном доходе. Копируем формулу вниз до конца таблицы

`=ЕСЛИ(И(G3<>"Na";E3<>"Na");ABS(E3-G3);"")`

| | K | |
|----------|---------------------|---|
| rs | M_weekly - F_weekly | 3 |
| асным!!! | | 0 |
| | 415 | 0 |
| | 345 | 0 |
| | | 0 |
| | | 0 |
| | 345 | 0 |

Находим в полученных столбцах J и K минимальные значения

`=МИН(J:J)`

| J | K | L | M |
|----------------------------|---------------------|---|---|
| M_workers - F_workers | M_weekly - F_weekly | 3 | 4 |
| ответы выделены красным!!! | | 0 | 2 |
| 480 | 415 | 0 | 0 |
| 419 | 345 | 0 | 0 |

Определяем в столбцах J и K, в каких отраслях получился минимум разницы доходов

`=ЕСЛИ(J51=L$2;A51;0)`

| L | M | |
|--------------------------|---|--|
| 0 | 0 | |
| 0 | 0 | |
| 0 | 0 | |
| 0 | 0 | |
| 0 | 0 | |
| 0 | 0 | |
| 0 | 0 | |
| Appraisers and assessors | 0 | |

Фильтруем по столбцу L

L

3

Postmasters and mail superintendents
Appraisers and assessors of personal property
Survey researchers
Sociologists
Miscellaneous social scientists
Biological technicians
Directors religious activities
Animal control workers
Crossing guards
Counter attendants cafeterias
Brokerage clerks
Hunters and trappers
Paperhangers
Model makers and pattern makers
Extruding and forming machine operators
Model makers and pattern makers
Mine shuttle car operators

Сортировать от меньшего к большому

Сортировать от большего к меньшему

Настраиваемая сортировка

Представление листа

Снять фильтр со столбца "3"

Числовые фильтры

Выбрать все

0

Animal control workers

Appraisers and assessors of personal property

Biological technicians

Brokerage clerks

Counter attendants cafeteria

Копируем столбцы D, E, F, G, L на лист “доп” и сортируем по убыванию доходов, красным выделен полученный ответ

Фильтруем по столбцу M, здесь подошел только один ответ, это и есть ответ

Все полученные четыре ответа разместим на листе “ответы”. Чтобы удовлетворить свое любопытство, осталось только перевести их с английского на русский!

| M_worker | M_weekly | F_workers | F_weekly |
|----------|----------|--|----------|
| 31 Na | 31 Na | Directors religious activities and education | |
| 28 Na | 28 Na | Counter attendants cafeteria food concession and coffee shop | |
| 21 Na | 21 Na | Appraisers and assessors of real estate | |
| 19 Na | 19 Na | Miscellaneous social scientists and related workers | |
| 13 Na | 13 Na | Crossing guards | |
| 10 Na | 10 Na | Postmasters and mail superintendents | |
| 10 Na | 10 Na | Biological technicians | |
| 3 Na | 3 Na | Model makers and patternmakers | |
| 2 Na | 2 Na | Animal control workers | |
| 1 Na | 1 Na | Brokerage clerks | |
| 0 Na | 0 Na | Survey researchers | |
| 0 Na | 0 Na | Sociologists | |
| 0 Na | 0 Na | Hunters and trappers | |
| 0 Na | 0 Na | Paperhangers | |
| 0 Na | 0 Na | Extruding and forming machine setters | |
| 0 Na | 0 Na | Model makers and patternmakers wood | |
| 0 Na | 0 Na | Mine shuttle car operators | |

M

4

Bookkeep

Сортировать от меньшего к бо...

Сортировать от большего к ме...

Настраиваемая сортировка

Представление листа

Снять фильтр со столбца "4"

Числовые фильтры

Выбрать все

0

2

Bookkeeping

| A |
|--|
| OFFICE |
| Chief executives |
| Directors religious activities and education |
| Bookkeeping |

2. Выполняем задание в PYTHON с применением модуля pandas (решение в Jupyter Notebook пакета Anaconda)

Что сделать?

Прочитаем данные из таблицы

Как сделать?

```
import pandas as pd

table = pd.read_excel('7_8_9.xlsx')
table
```

| | Occupation | All_workers | All_weekly | M_workers | M_weekly | F_workers |
|---|-------------------------------------|-------------|------------|-----------|----------|-----------|
| 0 | NaN | NaN | NaN | NaN | NaN | NaN |
| 1 | Chief executives | 1046.0 | 2041 | 763.0 | 2251 | 283.0 |
| 2 | General and operations managers | 823.0 | 1260 | 621.0 | 1347 | 202.0 |
| 3 | Legislators | 8.0 | Na | 5.0 | Na | 4.0 |
| 4 | Advertising and promotions managers | 55.0 | 1050 | 29.0 | Na | 26.0 |

Перенесем в данные словарь

```
headers = table.columns.ravel()
sl = table.to_dict()
sl
```

```
{'Occupation': {0: nan,
1: 'Chief executives',
2: 'General and operations managers',
3: 'Legislators',
4: 'Advertising and promotions managers',
5: 'Marketing and sales managers',
6: 'Public relations and fundraising managers',
7: 'Administrative services managers',
8: 'Computer and information systems managers',
9: 'Financial managers',
10: 'Compensation and benefits managers',
11: 'Human resources managers'}}
```

Определяем отрасль с наибольшим средним общим доходом, используем словарь и функцию `max`

```
All_workers = [sl['All_workers'][i] for i in sl['All_workers'].index(max(All_workers[1:]))]
ind = All_workers.index(max(All_workers[1:]))
print(All_workers.count(max(All_workers[1:]))
# первый ответ
sl['Occupation'][ind]
```

1

' OFFICE '

Определяем отрасль с наибольшим средним недельным доходом, используем словарь и функцию `max`

```
All_weekly = [sl['All_weekly'][i] for i in sl['All_w
maxx = -1
for i in All_weekly[1:]:
    if i != 'Na' and i > maxx:
        maxx = i
# введи переменную
ind = All_weekly.index(maxx)
print(All_weekly.count(maxx))
sl['Occupation'][ind]
```

1

'Chief executives'

Определяем отрасль с наименьшей разницей между средними общими доходами мужчин и женщин

```
M = [sl['M_workers'][i] for i in sl['M_workers']]  
F = [sl['F_workers'][i] for i in sl['F_workers']]  
minn = 1000000000000000000  
ind = 0  
MF = []  
indMF = []  
for i in range(1, len(M)):  
    if M[i] != 'Na' and F[i] != 'Na' and abs(M[i] - F[i])  
        if abs(M[i] - F[i]) == minn:  
            MF.append(max(M[i], F[i]))  
            indMF.append(i)  
            minn = abs(M[i] - F[i])  
            ind = i  
if MF:  
    ind = indMF[MF.index(max(MF))]  
  
# mpeмуñ omθem  
sl['Occupation'][ind]
```

'Directors religious activities and education'

Определяем отрасль с наименьшей разницей между средними недельными доходами мужчин и женщин

```
m = [sl['M_weekly'][i] for i in sl['M_weekly']]  
f = [sl['F_weekly'][i] for i in sl['F_weekly']]  
minn = 1000000000000000000  
ind = 0  
mf = []  
indmf = []  
for i in range(1, len(m)):  
    if m[i] != 'Na' and f[i] != 'Na' and abs(m[i] - f[i])  
        if abs(m[i] - f[i]) == minn:  
            mf.append(max(m[i], f[i]))  
            indmf.append(i)  
            minn = abs(m[i] - f[i])  
            ind = i  
if mf:  
    ind = indmf[mf.index(max(mf))]  
  
# четвертый ответ  
sl['Occupation'][ind]
```

'Bookkeeping'

РЕШАЕМ В ТЕТРАДИ

Задание 21.

В ячейке B2 электронной таблицы хранилась формула $= (A2 * 10 + B\$1)^2$, см. данные в таблице. Бот скопировал формулу в ячейки B3 и B4, а также в каждую ячейку диапазона C2:D4. Какое число появится в ячейке D3?

| | A | B | C | D |
|---|---|--------------------|---|---|
| 1 | | 0 | 2 | 4 |
| 2 | 1 | =(\$A2*10+B\$1) ^2 | | |
| 3 | 2 | | | |
| 4 | 3 | | | |
| 5 | | | | |

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

№1

«Впадиной» называется такая ячейка электронной таблицы, значение которой меньше любого из значений соседних ячеек слева, справа, сверху и снизу. Глубиной впадины будем считать разницу между наименьшим значением соседних клеток и значением «впадины». В [электронной таблице](#) (файл *Занятие_2_7_Самостоятельная_работа_1*) определите глубину самой глубокой «впадины» и количество «впадин» максимальной глубины. В ответе сначала укажите максимальную глубину, затем найденное количество.

No2

После проведения олимпиады по информатике жюри олимпиады внесло результаты всех участников олимпиады в электронную таблицу. В столбце А записан код фамилии, в столбце В – код имени и отчества, в столбце С записан класс, в столбцах D, E, F, G записаны баллы по каждой из четырех задач. По

данным результатам жюри хочет определить победителей и призеров. Количество победителей не должно превышать 8% от общего числа участников, общее число победителей и призеров не должно превышать 25% от общего фактического числа участников. Победители и призеры устанавливаются путем ранжирования участников по количеству набранных баллов в общем конкурсе. Найдите количество победителей и призеров, максимальный и минимальный балл победителя, и максимальный и минимальный балл призера. Задачу решите двумя способами – через средства электронной таблицы и составлением программного кода. Файл к заданию (Занятие_2_7_Самостоятельная_работа_2) [здесь](#).

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Чем отличается электронная таблица от обычной?
2. Как связаны понятия «формула» и «ссылки»?
3. Объясните механизм работы функции в формуле. Какие функции наиболее часто используются?
4. Какие средства для работы с форматом Excel имеет библиотека pandas?
5. Какие структуры данных надо использовать для записи в Excel файл?

§2.8 Контроль по теме " Файлы и файловая система "

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Вариант 1

1. Перемещаясь из одного каталога в другой, ученик Вася последовательно посетил папки MEDIA, WIN, C:\, DATA, GR, VIDEO, затем разместил в текущей папке файл CIX – WAVE.mp4. Каково полное имя размещенного файла?

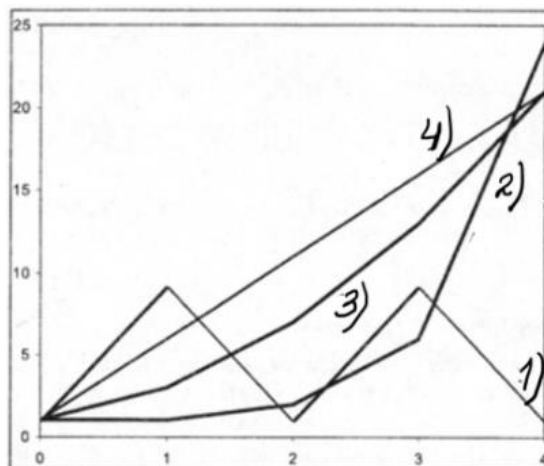
2. Определите, какое из указанных имен файлов удовлетворяет маске: *?e*.doc?*

Примечание: символ “?” означает ровно один произвольный символ; символ “*” означает любую последовательность символов произвольной длины, в том числе и пустую.

3. Дан фрагмент электронной таблицы:

| | A | B | C | D | E |
|---|----------------|------------|-------------------|--------------|----------------|
| 1 | 1 | 5 | | | |
| 2 | 0 | 1 | 1 | 1 | 1 |
| 3 | =A2 +\$A\$1 | =A3 *B2 | =- C2+2*\$B\$1 | =D2 +A3*2 | =E2+\$ B\$1 |

После копирования диапазона ячеек А3:Е3 в диапазон А4:Е6 был построен график по значениям столбца диапазона ячеек А2:А6. Каков номер графика, который соответствует значениям С2:С6?



4. В электронной таблице значение формулы =СУММ(С3:Е3) равно 75. Чему равно значение формулы =СРЗНАЧ(С3:Е3), если значение ячейки Е3 равно 5?

Вариант 2

1. Перемещаясь из одного каталога в другой, ученик Петя последовательно посетил папки AUDIO, MEDIA, WIN, C:\, DATA, GR, VIDEO, MP4, затем оценил размер текущей папки и стал освобождать в ней место. Каково полное имя папки, из которой Петя начал перемещение?

2. Определите, какое из указанных имен файлов удовлетворяет маске: ?a???*

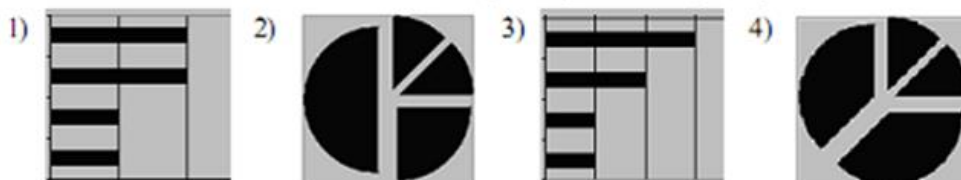
- 1) cad1 2) dad77 3) 3mammy 4) add123

Примечание: символ “?” означает ровно один произвольный символ; символ “*” означает любую последовательность символов произвольной длины, в том числе и пустую.

3. Дан фрагмент электронной таблицы:

| | А | В | С | Д |
|---|--------------|----------|----------|------------|
| 1 | $=(B1+C2)/2$ | $=D2-C1$ | $=D2-C2$ | $=B1+C2+1$ |
| 2 | | | 1 | 4 |

После выполнения вычислений была построена диаграмма диапазона ячеек А1:Д1. Каков номер подходящей диаграммы?



4. В электронной таблице значение формулы =СУММ(B1:B2) равно 5. Чему равно значение ячейки B3, если значение формулы =СРЗНАЧ(B1:B3), если значение равно 3?

ПРАКТИЧЕСКАЯ ЧАСТЬ

Вариант 1

1.

Создайте программу, которая будет в исходной строке заменять 'ma' на 'mama'.

| Входные данные | Выходные данные |
|-------------------------|-------------------------------------|
| Ma ma ma mama mariya ma | Ma mama mama mamamama mamariya mama |

2.

Создайте программу, которая считывает из текстового файла числа и записывает в другой текстовый файл те из них, которые являются полными квадратами. Входной текстовый файл text.txt создайте сами так, чтобы он состоял из двух строк: в первой строке записано число, которое обозначает количество чисел в файле; во второй строке — сами числа, разделенные пробелом.

Вариант 2

1.

Создайте программу, которая будет в исходной строке первую букву слова делать прописной.

| Входные данные | Выходные данные |
|-------------------------|-------------------------|
| А Васька слушает да ест | А Васька Слушает Да Ест |

2.

Создайте программу, которая читает из текстового файла строки и выводит в другой текстовый файл те из них, в которых есть буква "N" или "n". Входной текстовый файл text.txt создайте сами так, чтобы он состоял из нескольких строк: в первой строке записано число k, которое обозначает количество строк в файле, в последующих k строках записаны последовательности латинских букв (a..z).

III. Моделирование и теория игр

§3.1 Введение в теорию систем и системный анализ

ВСПОМИНАЕМ

Вспомним:

1. Знаком ли вам термин «система»? Что вы под ним понимаете?

ОБСУЖДАЕМ

Понятие системы относится к числу основополагающих и используется в различных сферах человеческой деятельности. Известные словосочетания «информационная система», «биологическая система», «система уравнений» и другие показывают распространенность этого термина в разных предметных областях. Приведите известные вам примеры систем.

ЧИТАЕМ

Теория систем и системный анализ имеет особое значение в силу принципиальной возможности использовать системный подход практически в любой решаемой человеком задаче.

При системном подходе объект исследования представляется как система. Что же такое система?

Система — целостный набор элементов, взаимосвязанных и взаимодействующих между собой так, чтобы могла реализоваться функция системы.

Свойство системности присуще и процессу познания. В процесс познания включается анализ и синтез. Анализ – это процесс, состоящий в разделении целого на части, в представлении сложного в виде совокупности более простых частей, но, чтобы познать целое, сложное, необходим и обратный процесс – синтез.

Системный анализ включает:

1. Структурное выделение элементов системы и связей между ними.

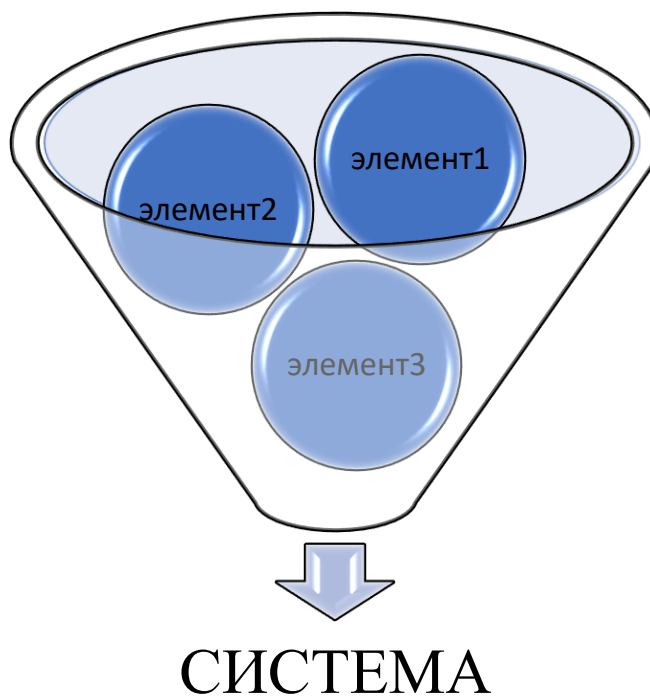


Рис. 77. Структурное выделение системы

2. Функциональное выделение совокупности целенаправленных действий системы, целевой функции системы.

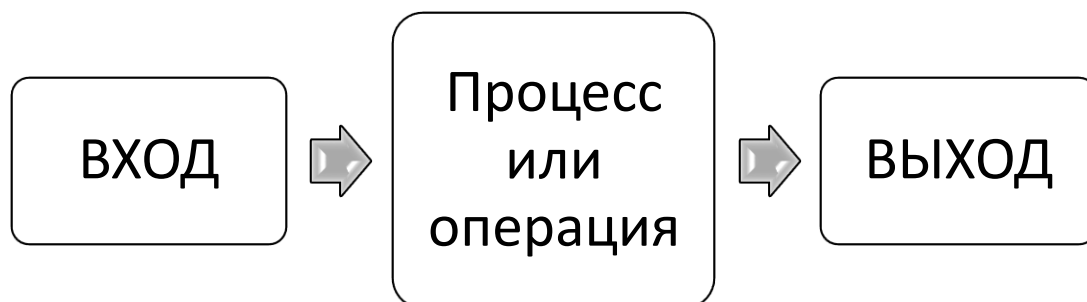


Рис. 78. Функционирование системы:

3. Макроскопическое понимание системы как целого, взаимодействующего с внешней средой.

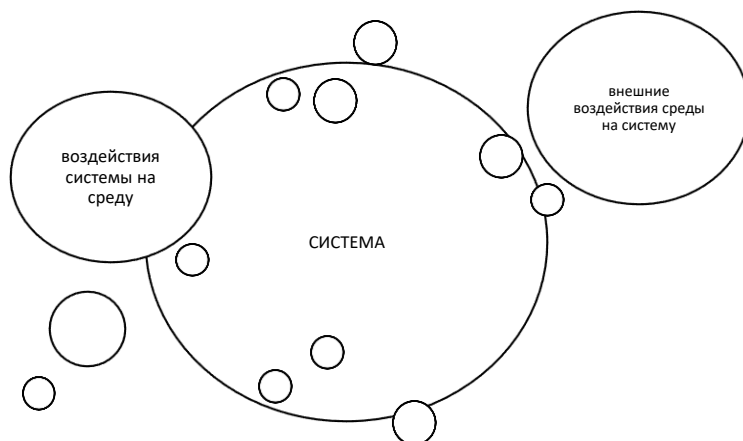


Рис. 79. Взаимодействие с внешней средой системы

4. Микроскопическое рассмотрение системы предполагает раскрытие структуры системы.

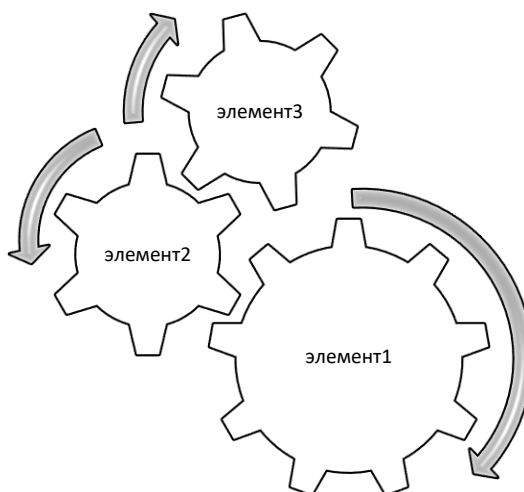


Рис. 80. Структура системы

Под **структурой системы** понимается устойчивое множество отношений, которое в течение интервала наблюдения остается неизменным.

Связи — это элементы, осуществляющие непосредственное взаимодействие между элементами (или подсистемами) системы, а также с элементами и подсистемами окружения.

Связи различают по характеру взаимосвязи как прямые (в направлении основного процесса) и обратные, возвращающие изменение состояния системы в результате управляющего воздействия на нее.

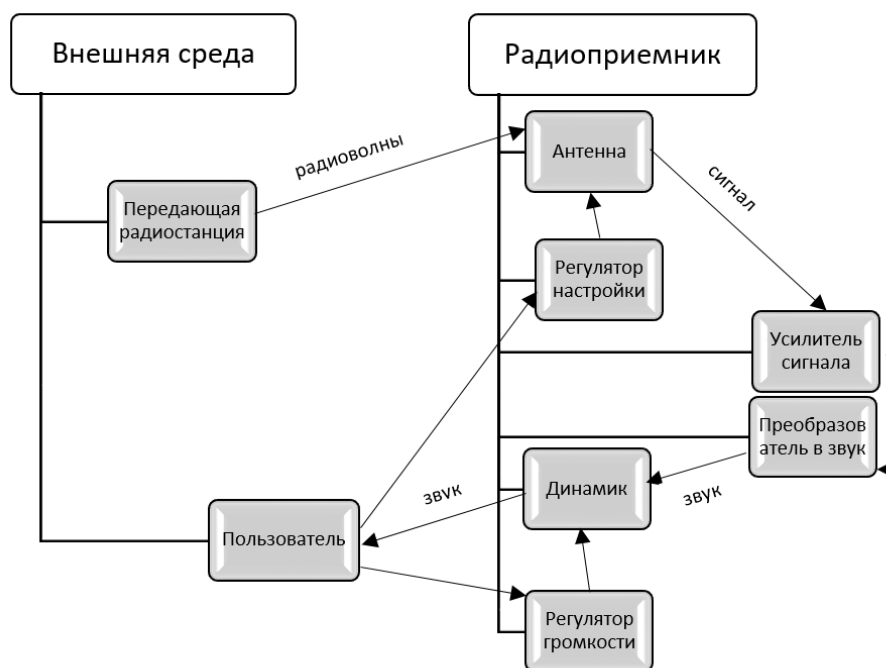


Рис. 81. Схема взаимодействия компонент радиоприемника друг с другом и с окружающей средой

5. Иерархическое представление основано на понятии подсистемы, получаемом при разложении (декомпозиции) системы. Система может быть

представлена в виде совокупностей подсистем различных уровней, составляющую системную иерархию, которая замыкается снизу только элементами.

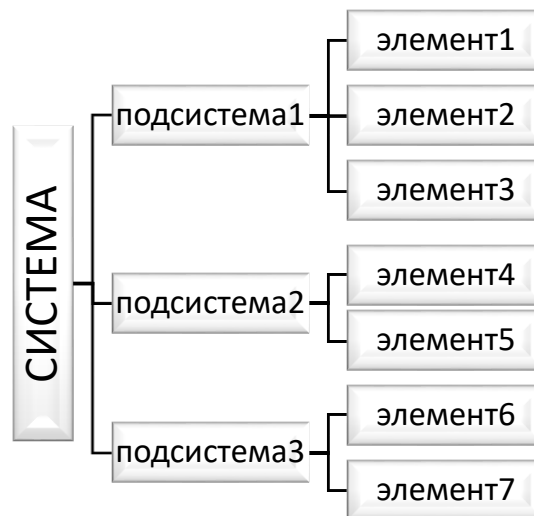


Рис. 82. Иерархическое представление системы

Всякая система может рассматриваться как подсистема системы более высокого порядка (**надсистемы**), а также как надсистема системы более низкого порядка (**подсистемы**). Например, система «отдел» входит как подсистема в систему «предприятие», которая может являться подсистемой «корпорации», система «радиоприемник» состоит из подсистем приема, питания и воспроизведения.



Рис. 83. Иерархия состава системы «Радиоприемник»

6. Процессуальное понимание системного объекта как динамического объекта, характеризующегося последовательностью его состояний во времени.

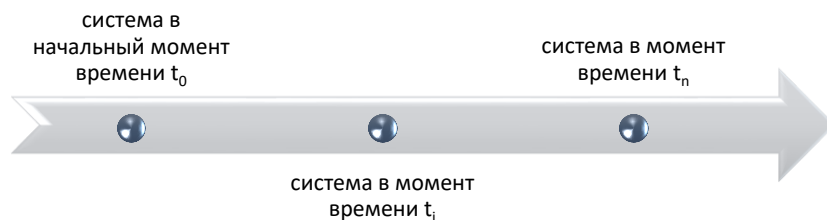


Рис. 84. Представление системы на временной шкале

Определение функционирования системы связано с понятием «проблемной ситуации», которая возникает, если имеется различие между необходимым (желаемым) выходом и существующим (реальным) входом.

Проблема — это разница между существующей и желаемой системами. Решить проблему — значит скорректировать старую систему или сконструировать новую, желаемую.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Пример 1

Условие:

Пусть имеется три предприятия А, В, С – потребителя продукции и предприятие Р – поставщик этой продукции. Причем и поставщик, и потребители выдвигают свои условия, связанные со сроками поставки продукции:

1. А – 1-й или 2-й квартал;
2. В – 1-й или 3-й квартал;
3. С – 2-й или 3-й квартал;

Р – может поставлять продукцию в один квартал только одному потребителю.

Оценить возможность удовлетворения требований сторон участников.

Решение:

Используя законы алгебры логики (в частности, распределительный закон), запишем выражение и преобразуем его:

$$(A1+A2)(B1+B3)(C2+C3) = (A1B1+A1B3+A2B1+A2B3)(C2+C3) = A1B1C2+A1B1C3+A1B3C2+A1B3C3+A2B1C2+A2B1C3+A2B3C2+A2B3C3 = A1B2C3+A3B1C2.$$

Ответ:

1. А – первый квартал, В – третий квартал, С – второй квартал.
2. А – второй квартал, В – первый квартал, С – третий квартал.

РЕШАЕМ В ТЕТРАДИ

Задание 22.

Проходит школьный бал. Первый участник готов выбрать вторую или третью участницу; второй – первую или вторую; третий – первую или третью. На ком должны остановить свой выбор девушки, чтобы все остались довольны?

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 2

Условие: Сколько различных решений имеет система логических уравнений

$$(x_1 \wedge x_2) \vee (x_1 \vee x_3) \wedge (x_1 \vee y_1) = 0 \quad (1)$$

$$(x_2 \wedge x_3) \vee (x_2 \vee x_4) \wedge (x_2 \vee y_2) = 0 \quad (2)$$

$$(x_3 \wedge x_4) \vee (x_3 \vee x_5) \wedge (x_3 \vee y_3) = 0 \quad (3)$$

$$(x_4 \wedge x_5) \vee (x_4 \vee x_6) \wedge (x_4 \vee y_4) = 0 \quad (4)$$

где $x_1, \dots, x_6, y_1, \dots, y_4$, – логические переменные? В ответе не нужно перечислять все различные наборы значений переменных, при которых выполнено данное равенство. В качестве ответа нужно указать количество таких наборов.

Решение:

Преобразуем уравнения, используя законы логики (дистрибутивный и поглощения):

$$(x_1 \wedge x_2) \vee (x_1 \vee x_3) \wedge (x_1 \vee y_1) = (x_1 \wedge x_2) \vee x_1 \vee (x_3 \wedge y_1) = x_1 \vee (x_3 \wedge y_1).$$

Итак, имеем равносильную систему:

$$x_1 \vee (x_3 \wedge y_1) = 0 \quad (1)$$

$$x_2 \vee (x_4 \wedge y_2) = 0 \quad (2)$$

$$x_3 \vee (x_5 \wedge y_3) = 0 \quad (3)$$

$$x_4 \vee (x_6 \wedge y_4) = 0 \quad (4)$$

Из всех уравнений сразу получаем, что x_1, x_2, x_3 и x_4 равны 0, а все переменные y_i независимы. При $x_1 = 0$ и $x_3 = 0$ в первом уравнении y_1 может быть любым – два решения. Аналогично для уравнения (2) получим два решения. В уравнениях (3) и (4) все переменные, кроме x_3 и x_4 , независимы, поэтому подходят по 3 комбинации, когда конъюнкция обращается в 0. Итого, количество решений равно произведению числа независимых комбинаций значений переменных: $2 * 2 * 3 * 3 = 36$.

РЕШАЕМ В ТЕТРАДИ

Задание 23.

При составлении учебного расписания желательно, чтобы урок по истории был первым или вторым, по информатике – первым или

третьим, по физике – вторым или третьим. Как составить расписание?
Сколько вариантов расписания может быть?

РЕШАЕМ В ТЕТРАДИ

Задание 24.

Решить систему логических уравнений:

Сколько различных решений имеет система уравнений

$$X_1 + X_2 * X_3 = 1$$

$$X_2 + X_3 * X_4 = 1$$

...

$$X_8 + X_9 * X_{10} = 1$$

где x_1, x_2, \dots, x_{10} – логические переменные?

Обозначения: “+” – дизъюнкция, “*” – конъюнкция

ЧИТАЕМ

Целостность и эмерджентность системы

Главным свойством системы является целостность. **Целостность** системы означает, что каждый элемент системы вносит вклад в реализацию целевой функции системы, это достигается посредством определенных взаимосвязей и взаимодействий элементов системы и проявляется в возникновении новых свойств, которыми элементы системы не обладают. Это свойство **эмерджентности** — сущностное свойство системы, обуславливающее появление новых свойств и качеств, не присущих элементам, входящим в состав системы. Например, каждая отдельная деталь космического корабля не обладает свойством «полететь в космос».

Для радиоприемника сущностное свойство – способность воспроизводить звук, закодированный в виде радиоволн и посланный радиостанцией. Данное свойство является эмерджентным, т.к. ни один из компонентов радиоприемника по отдельности не обладает им: антенна способна только улавливать радиоволны, преобразователь – преобразовывать радиоволны в звуковые и т.д.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Выберите одну из перечисленных ниже систем. Опишите сущностное свойство системы. Определите, является ли данное свойство эмерджентным.

1. технические устройства и комплексы (автомобиль, компьютер, смартфон);
2. организации (школа, гостиница, кафе);
3. биологические системы (человек, животные, растения);

4. информационные системы (автоматизированные системы, программные комплексы);
5. социально-экономические системы (система безопасности, транспортная система региона).

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Какое свойство системы связано с появлением характеристик, отсутствующих у элементов, образующих систему?
2. Какое значение термина «проблема» мы рассмотрели? Какие способы решения проблемы Вам известны?

§3.2 Модели и моделирование. Этапы моделирования

ВСПОМИНАЕМ

Вспомним:

1. Что такое система?
2. Сколько характеристик есть у системы?

ЧИТАЕМ

На самом деле, любая система обладает бесконечным набором свойств различной природы. В процессе познания окружающего мира взаимодействие осуществляется с ограниченным множеством свойств, лежащих в пределах возможности их восприятия и их необходимости для цели познания.

Решение задач, связанных с исследованием, проектированием, совершенствованием систем, бывает сложно или нерационально проводить на самих этих системах.

В этих и других случаях исходная система заменяется некоторой другой материальной или абстрактной системой. Исходную будем называть «**объект моделирования**» или «**объект-оригинал**» или «**прототип**», а вторую – «**модель**».

Модель — это аналог (заместитель) оригинала, отражающий наиболее существенные для целей моделирования свойства.

Цель отражает причину создания модели и определяет ее назначение.

Изучение любой системы предполагает создание модели системы, позволяющей проанализировать ее и предсказать поведение в определенных условиях, решать задачи анализа и синтеза реальной системы.

Модель в определенных условиях может заменить объект-оригинал и служит для получения информации об объекте-оригинале и других объектах, с ним связанных. Характерные случаи, когда требуется модель:

- объект-оригинал есть сложная система, изучение которой затруднено, невозможно или экономически невыгодно;
- проведение экспериментов с объектом-оригиналом разрушительно для него;
- прогноз состояния или поведения объекта в будущем;
- выбор оптимального решения, связанного с функционированием объекта-оригинала;
- объект-оригинал еще не существует в материальном виде;
- получение информации об объекте-оригинале с целью информационного обеспечения людей, работающих с ним;
- при обучении работе с моделируемой системой, в играх и т. п.

Моделирование – это метод познания окружающего мира, который рассматривается как процесс, включающий разработку модели и ее исследование.

В курсе информатики рассматриваются компьютерные модели и компьютерное моделирование.

Этапы компьютерного моделирования

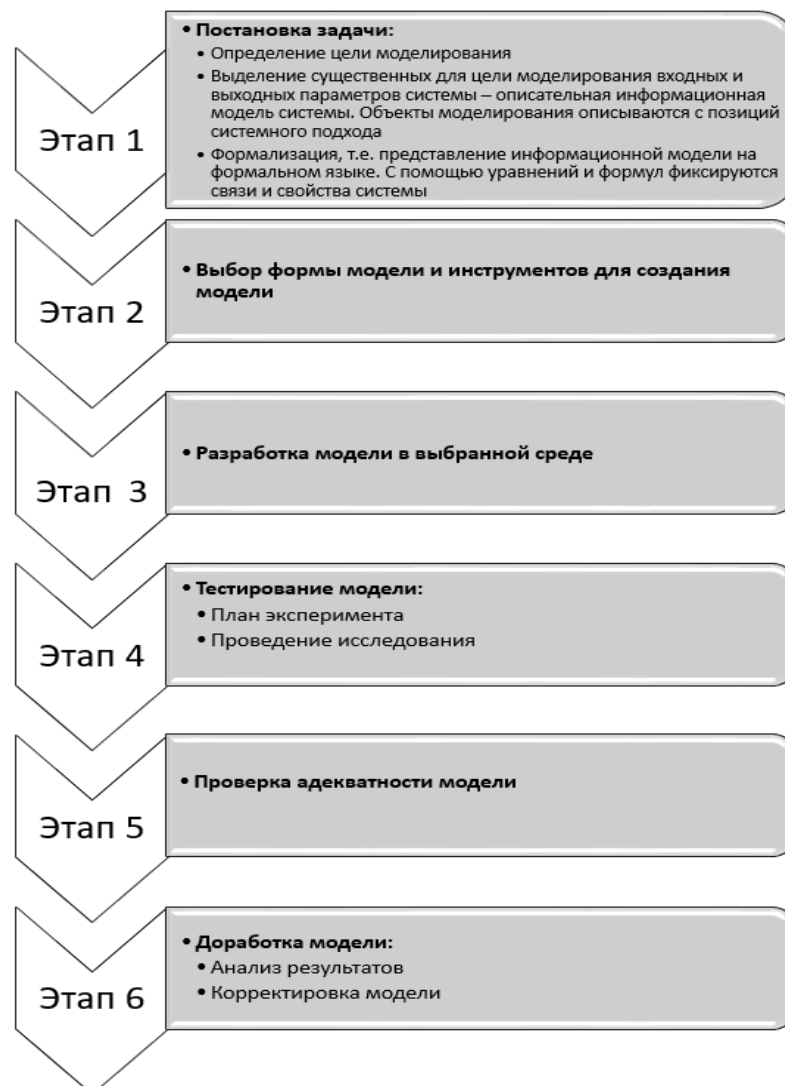


Рис. 1. Этапы компьютерного моделирования

Принципы, которым должна удовлетворять построенная модель:

1. Адекватность – соответствие модели целям исследования и реальной системе относительно выбранного множества свойств.
2. Соответствие модели решаемой задаче.
3. Принцип абстрагирования от второстепенных деталей – упрощение при сохранении существенных свойств системы.
4. Соответствие между требуемой точностью результатов моделирования и сложностью модели.
5. Баланс погрешностей различных видов.
6. Блочное строение.

ОБСУЖДАЕМ

Строится модель полета кометы, вторгнувшейся в пределы Солнечной системы. Какие входные параметры модели могут быть? А какие выходные параметры можно получить?

Например, масса кометы, ее скорость, время появления, масса и расположение планет Солнечной системы являются входными величинами, а наименьшее расстояние, на которое комета приблизится к Земле – выходной величиной.

ЧИТАЕМ

Структурную модель системы называют структурной схемой. На структурной схеме отражается состав системы и ее внутренние связи. Для отображения структурной схемы системы часто используются графы.

Граф отображает элементный состав системы и структуру связей.

Граф состоит из **вершин**, обозначающих элементы системы, и **ребер** – линий, обозначающих связи (отношения) между элементами системы. Граф может быть описан в виде таблицы (матрицы смежности или весовой матрицы). **Степень вершины** – это количество ребер, которые соединены с этой вершиной

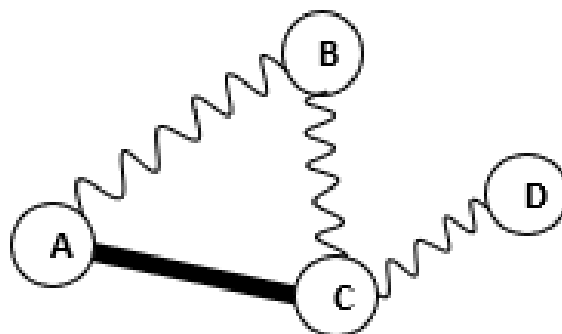
РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Пример 1

Условие:

Грунтовая дорога проходит последовательно через 4 населенных пункта: А, В, С и D. Велосипедисту стало известно, что длина дороги между четвертым и вторым пунктом равна 80 км, между вторым и третьим – 40 км, и между третьим и первым – 10 км. Между четвертым и третьим построили новое асфальтовое шоссе длиной 40 км. Оцените минимально возможное время движения велосипедиста из пункта А в пункт В, если его скорость по

грунтовой дороге – 20 км/час, по шоссе – 40 км/час. Схема дорог у велосипедиста имеется, но неизвестно, под каким номером был каждый населенный пункт в сообщении:



Решение:

Построим табличную модель системы дорог, полученную в сообщении, и соотнесем ее с графом схемы дорог.

Обозначим четыре населенных пункта П1, П2, П3 и П4.

| | П1 | П2 | П3 | П4 |
|----|----|----|----|----|
| П1 | | | 10 | |
| П2 | | | 40 | 80 |
| П3 | 10 | 40 | | 40 |
| П4 | | 80 | 40 | |

По таблице определяем, что П1 имеет степень 1, значит, это вершина D. П3 имеет степень 3, а значит это С. Пункты П2 и П4 имеют одинаковую степень и связаны друг с другом и с С. Они не определяются однозначно, но это не влияет на ответ, т.к. если ехать напрямую из А в В по грунтовой дороге, то потребуется $80 : 20 = 4$ часа, если ехать через С, т.е. по шоссе и по грунтовой, то потребуется $40 : 40 + 40 : 20 = 3$ часа.

Ответ:

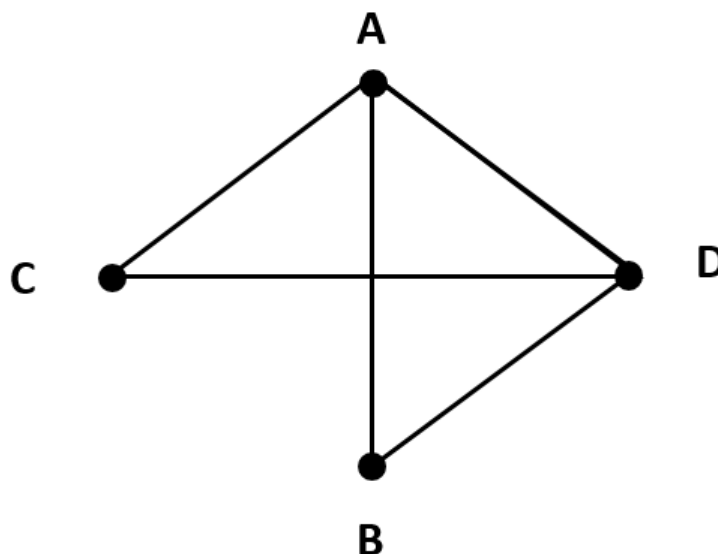
3 часа.

РЕШАЕМ В ТЕТРАДИ

Задание 25.

Грунтовая дорога проходит последовательно через 4 населенных пункта: А, В, D и С. Известно, что длина дороги между четвертым и вторым пунктом равна 40 км, между вторым и первым – 120 км, и между третьим и первым – 30 км, а между третьим и четвертым 25 км. Между четвертым и первым построили новое асфальтовое шоссе длиной 100 км. Известно также, что $AC < BD$ и $CD < AB$. Оцените минимально возможное время движения велосипедиста из пункта А в пункт В, если его скорость по грунтовой дороге

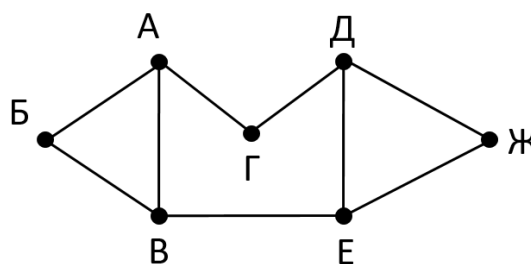
– 20 км/час, по шоссе – 40 км/час. Схема дорог у велосипедиста имеется, но неизвестно, под каким номером был каждый населенный пункт в сообщении:



ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

На рисунке слева изображена схема дорог Н-ского района, в таблице звёздочкой обозначено наличие дороги из одного населённого пункта в другой. Отсутствие звёздочки означает, что такой дороги нет. Определите, какие номера населённых пунктов в таблице могут соответствовать населённым пунктам В и Е на схеме. В ответе запишите эти два номера в возрастающем порядке без пробелов и знаков препинания.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | * | * |
| 2 | | | | * | | | * |
| 3 | | | | * | * | | |
| 4 | | * | * | | * | | |
| 5 | | | * | * | | * | |
| 6 | * | | | | * | | * |
| 7 | * | * | | | | * | |



ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. В каком случае модели одного и того же объекта будут разными?
2. Какие свойства объекта должна отражать модель?
3. Для чего используется моделирование?
4. Какие элементы включает процесс моделирования?
5. Какими могут быть входные и выходные параметры модели для следующих систем:
 - Строительство дома

- Солнечная система
- Метрополитен?

§3.3 Классификация моделей

ВСПОМИНАЕМ

Вспомним, что мы знаем о моделях и моделировании:

1. Что такое классификация? Основание классификации?
2. Что мы называем моделью объекта?

ЧИТАЕМ

Классификация моделей может быть рассмотрена по различным основаниям:

По области использования:

- Учебные (изучение объекта)
- Практические (использование в повседневной жизни)
- Опытные (исследование объекта)
- Научно-технические (исследование процессов и явлений)
- Игровые (моделирование поведения объекта: военные, экономические, деловые игры, игровая терапия)
- Имитационные (исследование свойств лекарственных препаратов на животных)

По отрасли знаний:

- Биологические
- Социологические
- Экологические
- Исторические и т.п.

ОБСУЖДАЕМ

Приведите примеры различных по областям использования и отраслям знаний модели.

ЧИТАЕМ

По фактору времени: статические и динамические.

- Статическая модель отображает единовременный срез по объекту. Примеры: карта местности, схема персонального компьютера, фотография, схема метро, карта медицинского осмотра пациента.
- Динамическая модель позволяет увидеть или прогнозировать изменение объекта во времени.

Динамические модели, например, представимы в виде формул, позволяющих вычислить параметры объекта как функции времени.

Примеры: набор формул, описывающий движение планет Солнечной системы; график изменения температуры в течение суток; медицинская карта пациента в поликлинике.

По множеству значений переменных: непрерывные и дискретные.

- Непрерывные – модели систем, числа состояний которой и число значений выходных характеристик бесконечно.
- Дискретные – модели систем, переменные которых могут принимать конечное число наперед известных значений. Основой формализованного описания дискретных объектов является аппарат математической логики (логические функции, аппарат булевой алгебры, алгоритмические языки). В связи с развитием вычислительной техники дискретные методы анализа получили широкое распространение также для описания и исследования непрерывных объектов. Например, замена непрерывной математической функции на набор ее значений в фиксированных точках.

По характеру изучаемых процессов: детерминированные и стохастические.

- Детерминированная модель отображает процесс, в котором предполагается отсутствие случайных факторов.
- Стохастическая модель учитывает вероятностные процессы и события.

Примеры: уравнение равноускоренного движения объекта – детерминированная модель процесса, модель броуновского движения частиц – стохастическая модель процесса.

По отдельной характеристике объекта:

- Модель внешнего вида
- Модель структуры
- Модель поведения
- Модель функционирования системы

По форме представления: натурные (материальные) и информационные

- **Материальные** — это уменьшенные (увеличенные) копии оригинала, которые отражают его внешние свойства, внутреннее устройство, суть процессов, происходящих с прототипами.

Основными видами материальных моделей являются геометрические и физические, т.е. подобные оригиналу не только с точки зрения геометрических соотношений, но и с точки зрения происходящих в нем физических процессов.

При этом и модель, и оригинал всегда имеют одинаковую физическую структуру.

Примеры: физические, химические опыты, роботы, скелет человека.

- **Информационные** модели — это целенаправленно отобранная информация об объекте, представленная в некоторой форме.

К ним относятся вербальные и знаковые модели.

- **Модель называется вербальной**, если она представима в мыслимой или разговорной форме. Модель получается в результате раздумий и умозаключений человека, представленных в словесной форме
- **Модель называется знаковой**, если она выражает основные свойства и отношения реального объекта или процесса с помощью определенной системы знаков и символов, то есть средствами любого формального языка (литературного, алгоритмического, математического, программирования).

Примеры: списки, схемы, чертежи, графики, математические формулы и т.д.

Знаковые модели бывают структурными, текстовыми и математическими.

- **Структурные** – информация об объекте оригинале (структура системы) представлена в виде схем, чертежей, графиков, таблиц
- **Текстовые** – используют предложения естественного языка для описания объектов, событий, ситуаций, предметных областей с целью их осмысления, анализа и использования опыта
- **Математические** – это совокупность математических выражений, определяющих формальную зависимость между входными и выходными данными моделируемого объекта или явления в виде уравнений, неравенств, систем уравнений и неравенств, логических и числовых выражений

По инструментарию знаковые модели делятся на компьютерные и некомпьютерные.

- Компьютерные модели делятся на структурно-функциональные и имитационные
 - Структурно-функциональными называются компьютерные модели, которые представляют собой условный образ объекта, описанный с помощью

компьютерных сред (трехмерная модель объекта, траектория движения объекта)

- Имитационными называются компьютерные модели, представляющие собой программу или комплекс программ, позволяющий воспроизводить процессы функционирования объекта в разных условиях (распад атома, анимация работы двигателя)

Компьютерные модели можно строить с помощью самых разных программных средств и пакетов: офисных пакетов, например, MSOffice (MSExcel, MSWord, MSAccess), AutoCad, SolidWorks и т.п. и любых систем программирования.

Компьютерная модель, в основе которой лежит математическая модель, называется **компьютерной математической моделью**.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание: Заполните пропущенные ячейки таблицы по образцу:

| Объект оригинал | Модель | По форме представления: материальная или информационная | По фактору времени: статическая или динамическая | По области использования | По отдельной характеристике объекта |
|------------------------------|-----------------------------------|---|--|--------------------------|-------------------------------------|
| Планета Земля | Глобус | материальная | статическая | учебная | внешний вид |
| Человек | Фотография | информационная | статическая | практическая | внешний вид |
| | Расписание уроков в школе | | | | |
| | План эвакуации при пожаре | | | | |
| | Макет корабля | | | | |
| | Чертеж двигателя | | | | |
| | Компьютерная игра «Гонки» | | | | |
| | Химическая формула воды | | | | |
| | Схема метрополитена | | | | |
| | Формула равноускоренного движения | | | | |
| Книги библиотеки | | | | | |
| Кровеносная система человека | | | | | |
| | Медицинская карта пациента | | | | |

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Какие модели являются статическими, а какие – динамическими: рисунок извержения вулкана, видеозапись извержения вулкана?

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. <http://izd-mn.com>

§3.4 Моделирование работы автомата

ВСПОМИНАЕМ

1. Что такое информация с точки зрения вычислительной техники?
2. Какая информация называется непрерывной? Какая информация называется дискретной?

ЧИТАЕМ

Сложные дискретные процессы, такие как протокол передачи информации в компьютерных сетях, и сложные дискретные системы, такие как компьютеры и их отдельные устройства (процессор, устройства ввода и др.) или пульт управления энергосистемой, и простые, такие как устройства, регулирующие работу светофора или управления лифтом, устройства управления роботами – можно считать преобразователями дискретной информации.

Широкий класс таких преобразователей объединяется под общим названием автомат.

Теория автоматов изучает математические модели преобразователей дискретной информации, называемые автоматами, а также задачи, которые они могут решать.

Теория автоматов лежит в основе цифровых технологий и программного обеспечения, так, например, компьютер — это частный случай практической реализации конечного автомата. Математический аппарат теории автоматов применяется при построении компиляторов и разработке языков программирования, а также для нахождения разрешимости и сложности задач.

Автомат — это дискретный преобразователь информации, реакция которого на входные сигналы зависит от состояния входа и входной истории.

Характерной особенностью этих преобразований являются дискретность функционирования и конечность областей значений параметров, их описывающих.

Результат преобразования автомата зависит не только от того, какая информация в данный момент появилась на входе, но и от того, что происходило раньше, от **предыстории** преобразования.

Что это означает, можно понять на примере: реакция человека на чье-либо замечание может варьироваться в зависимости от его настроения, которое определяется событиями, имевшими место ранее.

Автомат представляет собой кибернетическую систему, перерабатывающую дискретную информацию и меняющую свое внутреннее состояние лишь в допустимые моменты времени. В каждый момент времени автомат находится в некотором состоянии и может изменить это **состояние** под действием входного сигнала.

Состояние системы – это ее характеристика, однозначно определяющая ее дальнейшее поведение, все последующие реакции системы на внешние воздействия.

На один и тот же сигнал автомат может реагировать по-разному, в зависимости от того, в каком состоянии он находится в данный момент. Таким образом, в своих состояниях автомат запоминает свою историю.

Число возможных историй бесконечно велико, даже если вариантов входных воздействий не много. Однако на множестве предысторий можно ввести отношение эквивалентности. При этом две предыстории попадут в один класс эквивалентности, если они приводят автомат в одно и то же состояние. Очевидно, автомату не нужно запоминать конкретные входные истории. Достаточно, чтобы он запоминал классы эквивалентности, к которым данные истории принадлежат. Наиболее интересен случай, когда число классов эквивалентности и соответственно состояний конечно. Такой преобразователь называется **конечным автоматом (КА)**.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Условие:

Пусть: $P = \{p_1, p_2, p_3\}$ – входной алфавит, $W = \{w_1, w_2\}$ – выходной алфавит.

Будем рассматривать абстрактный конечный автомат в виде «черного ящика» с одним входом и одним выходом, т.е. анализируемое устройство описывается на уровне задания зависимости значений на его выходе от значений на его входе.

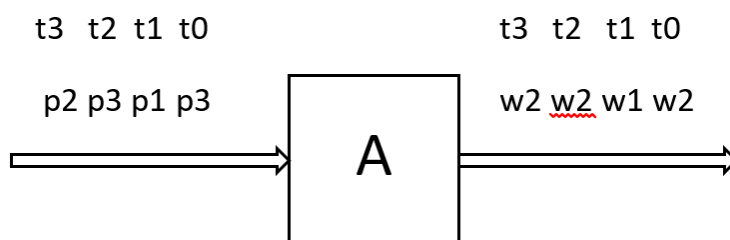


Рис. 86. Конечный автомат в виде «черного ящика»

Пример рассуждения:

Работа автомата рассматривается в дискретные моменты времени t_i , называемые автоматными тактами (i меняется от 0 до n , где n – длина входного слова). В примере на рис.86 в начальный момент времени t_0 автомат входную букву p_3 перерабатывает в выходную букву w_2 . В результате работы автомат входное слово $p_3p_1p_3p_2$ перерабатывает в выходное слово $w_2w_1w_2w_2$.

ЧИТАЕМ

В общем случае конечный автомат определяет устройство, которое реализует отображение множества **слов входного алфавита** во множество **слов выходного алфавита**.

Функционирование конечного автомата состоит в том, что последовательность p_1, p_2, \dots символов конечного алфавита P , поступающая на вход, вызывает на его выходе определенную последовательность $\omega_1, \omega_2, \dots$ символов того же или другого алфавита. Таким образом, конечные автоматы включают набор состояний и переходов между ними, зависящих от входных данных.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Пример 2

Условие:

Простейшим нетривиальным конечным автоматом является переключатель «включено-выключено». Это устройство помнит свое состояние, и от этого состояния зависит результат нажатия кнопки. Из состояния «выключено» нажатие кнопки переводит переключатель в состояние «включено», и наоборот.

Конечно-автоматная модель переключателя представлена на рисунке 87.

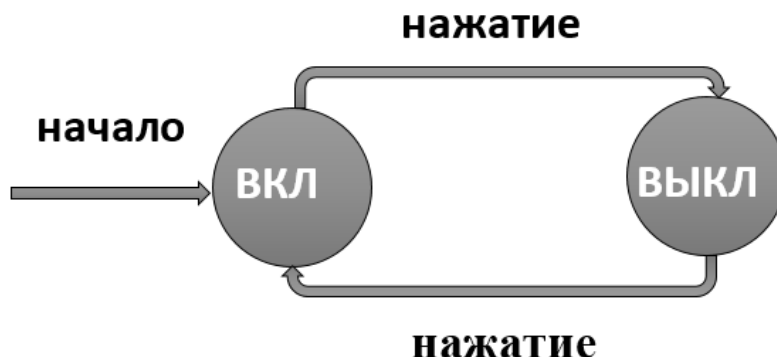


Рис. 87. Конечно-автоматная модель переключателя

Существует несколько способов представления конечных автоматов. Функционирование несложного автомата удобно представлять графически с помощью диаграммы состояний, являющейся ориентированным графом, у

которого состояния представлены вершинами, а дуги соответствуют переходам из одного состояния в другое. Дуги отмечаются «входными символами», задающими внешние воздействия на систему (устройство). В примере это *Нажатие*, что означает нажатие на кнопку переключателя. Стрелки указывают, что всякий раз при *Нажатии* система переходит из одного состояния в другое. Одно из состояний является так называемым «начальным состоянием» (в нашем случае – «вкл.»), в котором система находится изначально. На рисунке оно отмечено словом *Начало* и стрелкой, указывающей на это состояние.

ЧИТАЕМ

Для сложных автоматов более целесообразным представляются способы задания в форме таблиц переходов и выходов либо в форме матриц смежности направленного графа.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 3

Условие:

Рассмотрим пример задания автомата с помощью таблицы и графа:

Пусть $P = \{p1, p2\}$ – входные сигналы, $S = \{s0, s1, s2\}$ – состояния автомата.

Пример рассуждения:

В графическом способе каждому состоянию автомата ставится в соответствие вершина графа, которая помечается символом этого состояния. Если из состояния s_i существует переход в состояние s_j под воздействием входного символа p_k , то вершины s_i и s_j соединяются дугой, исходящей из s_i , а сама дуга помечается символом p_k , под воздействием которого осуществляется данный переход (рис. 88). Направление дуги указывает направление перехода, поэтому граф автомата является ориентированным графом.

Функцию перехода можно задать тремя различными способами:

Перечислением: $s0 = \varphi(s0, p1)$, $s1 = \varphi(s0, p2)$, $s1 = \varphi(s1, p1)$, $s2 = \varphi(s1, p2)$, $s2 = \varphi(s2, p1)$, $s0 = \varphi(s2, p2)$.

Таблицей переходов:

| | p1 | p2 |
|----|----|----|
| s0 | s0 | s1 |
| s1 | s1 | s2 |
| s2 | s2 | s0 |

Графом переходов: (рис. 88).

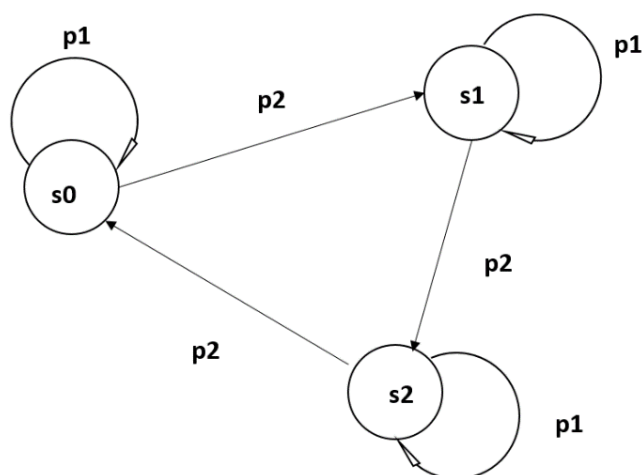


Рис. 88. Граф переходов

РЕШАЕМ В ТЕТРАДИ

Задание 26.

Построить граф переходов по таблице.

| | p1 | p2 |
|----|----|----|
| s0 | s1 | s0 |
| s1 | s2 | s1 |
| s2 | s0 | s2 |

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 4

Условие:

Автомат получает на вход четырёхзначное натуральное число и строит новое число по следующему алгоритму:

- 1) вычисляются суммы первой и второй, второй и третьей и третьей и четвёртой цифр;
- 2) из полученных сумм отбрасывается наименьшая;
- 3) остальные записываются в порядке невозрастания.

Пример. Исходное число: 1294. Суммы: $1 + 2 = 3$; $2 + 9 = 11$; $9 + 4 = 13$. Отбрасывается наименьшая сумма 3. Результат: 1311. Укажите наименьшее и наибольшее число, при вводе которых автомат выдаёт значение 115.

Решение:

Возможны два варианта суммы: 1 и 15 и 11 и 5. Т.к. в порядке невозрастания, значит рассматриваем 11 и 5. 11 получается, как суммы: $5 + 6$, $7 + 4$, $8 + 3$, $9 + 2$. $5 = 0 + 5$, $1 + 4$, $2 + 3$.

Из этих комбинаций наименьшее подходящее число должно начинаться с 1, подходит 1056, наибольшее – 9232.

Примечание:

Эту задачу можно решать как аналитически, проверяя разные варианты, так и составив программный код.

РЕШАЕМ В ТЕТРАДИ

Задание 27.

Автомат получает на вход трёхзначное число. По этому числу строится новое число по следующим правилам.

1. Перемножаются отдельно первая и вторая цифры, а также – вторая и третья цифры.

2. Полученные два числа записываются друг за другом в порядке неубывания без разделителей. Пример. Исходное число: 179. Произведения: $1*7 = 7$; $7*9 = 63$. Результат: 637. Укажите наименьшее число, при обработке которого автомат выдаёт результат 312.

ЧИТАЕМ

Более мощным автоматическим устройством по сравнению с КА является машина Тьюринга, которая является расширением модели КА и позволяет реализовать любой алгоритм.

Тьюринг впервые определил понятие алгоритма, исходя из понятия автоматически работающей машины, и предложил формальную модель вычислительного устройства, в котором смоделировал процесс выполнения произвольного алгоритма человеком, доведя его до самых элементарных операций (запись и стирание символа, использование дополнительной памяти и др.). Это устройство было названо машиной Тьюринга.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 5

Условие:

Вычислительное устройство получает на вход строку символов и преобразовывает её. Устройство может выполнять две команды, в обеих командах v и w обозначают цепочки символов.

заменить (v, w)

Эта команда заменяет в строке первое слева вхождение цепочки v на цепочку w . Если цепочки v в строке нет, эта команда не изменяет строку.

нашлось (v)

Эта команда проверяет, встречается ли цепочка v в строке исполнителя Редактор. Если она встречается, то команда возвращает логическое значение «истина», в противном случае возвращает значение «ложь». Строка при этом не изменяется.

Алгоритм решения:

```
НАЧАЛО
ПОКА нашлось (aaa)
    заменить (aaa, b)
    заменить (bb, a)
КОНЕЦ ПОКА
КОНЕЦ
```

Какая строка получится в результате применения приведённой выше программы к строке вида $a...ab...b$, состоящей из 44 “а” и 21 “b”? В ответе запишите полученную строку.

Решение:

```
s = 44 * 'a' + 21 * 'b'
while "aaa" in s:
    s = s.replace("aaa", "b", 1)
    s = s.replace("bb", "a", 1)
print(s)
```

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №28

Автомат получает на вход строку символов и преобразовывает её. Редактор может выполнять две команды, в обеих командах v и w обозначают цепочки символов.

```
заменить (v, w)
нашлось (v)
Дана программа для автомата:
ПОКА нашлось (sss)
    заменить (sss, d)
    заменить (ddd, e)
    заменить (eee, c)
КОНЕЦ ПОКА
```

Какая строка получится в результате применения приведённой ниже программы к строке, состоящей из 130 символов «с»?

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

Дополнительный материал (конечные автоматы):

3. <https://textarchive.ru/c-2820239-pall.html>
2. https://studopedia.net/13_23418_kak-predstavit-povedenie-konechnogo-avtomata.html
3. https://teccxx.neocities.org/mx1/13_task.html

4. Дискретная математика: учеб. пособие /сост. В.М.Пестриков, В.С. Дудкин, Г.А. Петров. – СПб.: СПб ГТУРП, 2013.- 136 с.

Дополнительный материал (клеточные автоматы):

1. <https://knife.media/cellular-automata/>

§3.5 Моделирование физических процессов

ВСПОМИНАЕМ

Вспомним:

1. Что такое модель?
2. Какую модель называют математической?
3. Что такое компьютерное моделирование?

ЧИТАЕМ

В физике компьютерное математическое моделирование является важным методом исследования. Наряду с традиционным делением физики на экспериментальную и теоретическую сегодня можно выделить третий фундаментальный раздел – вычислительная физика. Часто численное моделирование в физике называют вычислительным экспериментом, поскольку оно имеет много общего с лабораторным экспериментом и является инструментом познания качественных закономерностей природы. Важнейшим его этапом, когда расчеты уже завершены, является представление их в максимально наглядной и удобной для восприятия форме – в виде графиков, диаграмм, траекторий движения динамических объектов, что в силу особенностей человеческого восприятия обогащает исследователя качественной информацией.

Для решения задач вычислительной физики используются специальные математические пакеты прикладных программ, электронные таблицы и языки программирования высокого уровня, например, Python.

Рассмотрим некоторые задачи моделирования физических процессов движения тела в электронных таблицах Excel.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Условие:

Рассмотрим моделирование движения тела с некоторой постоянной скоростью $\vec{v} = \text{const}$.

Поскольку ни одна из характеристик скорости (направление и величина) не изменяется, движение будет происходить вдоль прямой линии, т.е. является

прямолинейным. Совместим с этой прямой ось Ox . Каждую секунду координата x тела будет получать одно и то же приращение, поэтому в любой момент времени может быть найдена как

$$x = v_x * t, \text{ где } v_x - \text{проекция вектора скорости на ось } Ox.$$

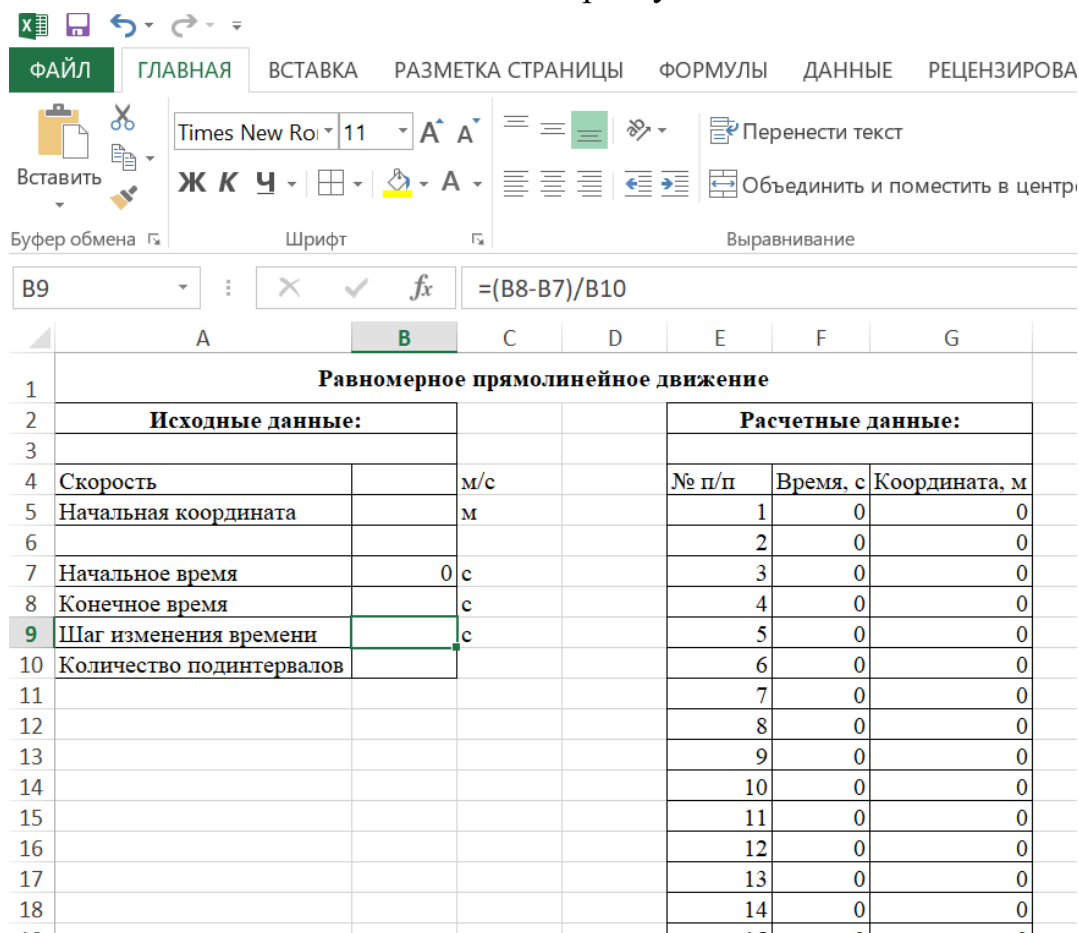
Ход рассуждений:

Если в начальный момент времени ($t_0 = 0$) положение тела не совпадало с началом отсчета, то уравнение будет иметь вид: $x = x_0 + v_x * t$. Проекция вектора скорости может быть и положительной, и отрицательной в зависимости от того, какой угол образует вектор скорости с направлением оси Ox .

Графическое моделирование процесса равномерного прямолинейного движения будет заключаться в построении графика зависимости $x = f(t)$ при различных значениях и направлениях скорости.

Практическая работа №29

1. Запустить Excel.
2. Создать шаблон модели по образцу:



| Равномерное прямолинейное движение | | | | | | |
|------------------------------------|---|-----|-------------------|----------|---------------|--|
| Исходные данные: | | | Расчетные данные: | | | |
| Скорость | | м/с | № п/п | Время, с | Координата, м | |
| Начальная координата | | м | 1 | 0 | 0 | |
| | | | 2 | 0 | 0 | |
| Начальное время | 0 | с | 3 | 0 | 0 | |
| Конечное время | | с | 4 | 0 | 0 | |
| Шаг изменения времени | | с | 5 | 0 | 0 | |
| Количество подинтервалов | | | 6 | 0 | 0 | |
| | | | 7 | 0 | 0 | |
| | | | 8 | 0 | 0 | |
| | | | 9 | 0 | 0 | |
| | | | 10 | 0 | 0 | |
| | | | 11 | 0 | 0 | |
| | | | 12 | 0 | 0 | |
| | | | 13 | 0 | 0 | |
| | | | 14 | 0 | 0 | |

Рис. 89. Шаблон таблицы задачи 1.

Исходными данными для данной модели будут: 1) начальное положение (координата) тела – ячейка B5, 2) проекция скорости на выбранную ось – B4, 3) временной интервал (задаваемый начальным и конечным моментом

времени), в течение которого рассматривается движение (время протекания процесса) – B7 и B8.

Переменные величины: время и координата тела; постоянные – проекция скорости на ось Oх (это означает, что ее числовое значение в процессе движения не изменяется).

Для построения графика $x = f(t)$ необходимо получить определенное число точек (t ; x). Следовательно, необходимо задать шаг изменения переменной $t - \Delta t = (t_{\max} - t_{\min})/n$, где n – число подинтервалов, на которые разбивается весь интервал (ячейка B10). Таблица будет содержать $n + 1$ точек. В ячейку B9 вставить соответствующую формулу (см. рисунок 1).

Начальный момент времени (ячейка B7) принимается равным нулю!

Для примера выбрать следующие значения: $v_x = 5$ м/с; $x_0 = 0$ м; $t_0 = t_{\min} = 0$ с; $t_{\max} = 10$ с, $n = 20$.

3. Заполнить таблицу данных:

3.1. В ячейку F5 занести начальный момент времени из ячейки B7, введя формулу связи.

3.2. В ячейку F6 вводится формула, рассчитывающая следующий (после начального) момент времени: $=F5+\$B\10 , после чего ее следует скопировать в остальные ячейки колонки.

3.3. Ячейку G5 связать с ячейкой B5, содержащей начальную координату.

3.4. Составить и занести в ячейку G6 формулу, позволяющую рассчитать координату в соответствующий ей момент времени. Примечание: При составлении формул не забывайте использовать абсолютные ссылки для постоянных величин.

3.5. Скопировать формулу из ячейки G6 в диапазон ячеек G7:G25.

3.6. Изменить имя листа, содержащего таблицу на *Равномерное движение*.

4. Построение графика зависимости $x(t)$. Данные для построения графика зависимости координаты равномерно движущегося тела от времени находятся в диапазоне ячеек F5:G25 созданной таблицы. Значения в столбце F (диапазон F5:F25) будут откладываться по горизонтальной оси (Ось X (категорий)), значения в столбце G (диапазон G5:G25) – по вертикальной оси (Ось Y (значений)).

4.1. Выделить диапазон данных.

4.2. Выбрать команду *Вставка-Диаграмма...*

4.3. С помощью мастера диаграмм провести построение и оформление графика. Основные требования: тип диаграммы – Точечная, вид – Точечная диаграмма с гладкими кривыми; название диаграммы и наименование осей координат с указанием единиц измерения величин, откладываемых по этим

осям, задать в виде: "Название диаграммы" – График равномерного движения; "Ось X (категорий)" – Время t , с; "Ось Y (значений)" – Координата x , м; – включить основные линии сетки по каждой из осей; – отключить легенду, т.к. в данной задаче используется только один набор (ряд) точек; – расположить диаграмму на отдельном листе.

5. Сохранить данные в файле.

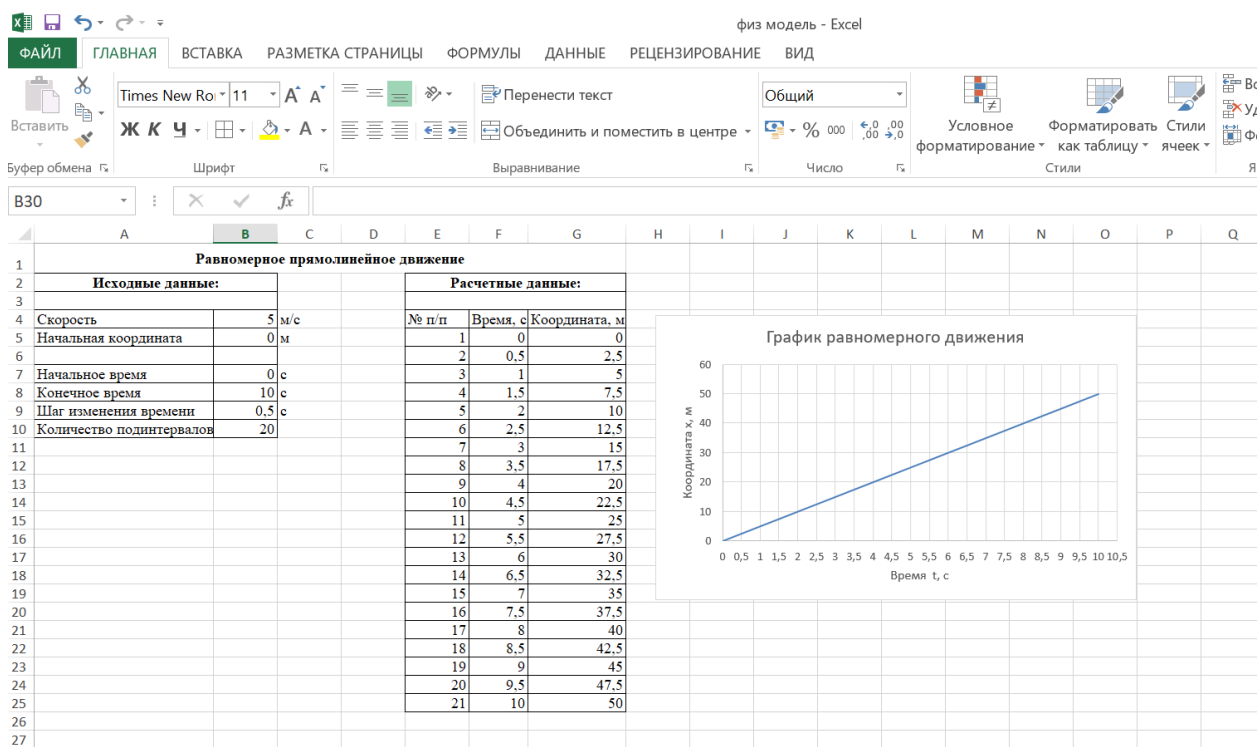


Рис. 90. Пример построения модели

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1.

Используйте таблицу, полученную в рамках выполнения практического задания № 29. После этого выполните следующие шаги:

1. Отформатировать числовые данные в таблице, назначив диапазону F5:G25 числовой формат с двумя десятичными знаками после запятой.
2. Проследить, влияет ли шаг программы (величина Δt) на вид графика.
3. Изменяя начальные данные в ячейках, проследить за изменением вида графика.
4. Задать отрицательное значение проекции скорости. Каков смысл знака "-" в значении проекции скорости? в значении координаты?

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №30

Моделирование равноускоренного движения тела

Постановка задачи:

Рассмотрим прямолинейное равноускоренное движение ($\vec{a} = \text{const}$) тела вдоль прямой.

Математическая модель:

Поскольку движение происходит вдоль прямой, то для его описания достаточно одной координаты. Пусть тело движется вдоль оси Оу. Запишем уравнение равноускоренного движения в проекции на выбранное направление оси Оу: $v_y = v_{0y} + a_y \cdot t$.

Проекции скорости и ускорения могут быть как положительными, так и отрицательными в зависимости от взаимного направления векторов v , a , v_0 и оси Оу. При этом если $a_y > 0$, скорость получает положительное приращение, т.е. с течением времени она увеличивается (ускоренное движение); если $a_y < 0$, скорость с течением времени уменьшается (замедленное движение).

Координата тела при этом будет изменяться по закону:

$$y(t) = y_0 + v_{0y} \cdot t + (a_y \cdot t^2) / 2,$$

где y_0 – начальное положение тела, т.е. его координата в момент времени $t_0 = 0$.

Компьютерное моделирование будет заключаться в построении графиков зависимостей $v_y = f_1(t)$ и $y = f_2(t)$ при различных значениях a_y и v_{0y} .

1. Запустить Excel.
2. Создать шаблон модели по образцу.

| | | | | | | | | | |
|---|------------------------------------|---|------------------|-------------------|----------|---------------|---------------|---|---|
| <div> <div>ФАЙЛ</div> <div>ГЛАВНАЯ</div> <div>ВСТАВКА</div> <div>РАЗМЕТКА СТРАНИЦЫ</div> <div>ФОРМУЛЫ</div> <div>ДАННЫЕ</div> <div>РЕЦЕНЗИРОВАНИЕ</div> </div> | | | | | | | | | |
| <div> <div> <div>Вставить</div> <div>Буфер обмена</div> </div> <div> <div>Times New Ro</div> <div>11</div> <div>A</div> </div> <div> <div>Ж К Ч</div> <div></div> <div>A</div> </div> <div> <div>Шрифт</div> </div> <div> <div>Выравнивание</div> </div> </div> | | | | | | | | | |
| <div> <div>Q2</div> <div> <div>✕</div> <div>✓</div> <div>f_x</div> </div> </div> | | | | | | | | | |
| | A | B | C | D | E | F | G | H | I |
| 1 | Равномерное прямолинейное движение | | | | | | | | |
| 2 | Исходные данные: | | | Расчетные данные: | | | | | |
| 3 | | | | | | | | | |
| 4 | Начальная скорость | | м/с | № п/п | Время, с | Скорость, м/с | Координата, м | | |
| 5 | Начальная координата | 0 | м | 1 | 0 | 0 | 0 | | |
| 6 | Ускорение | | м/с ² | 2 | 0 | 0 | 0 | | |
| 7 | Начальное время | | с | 3 | 0 | 0 | 0 | | |
| 8 | Конечное время | | с | 4 | 0 | 0 | 0 | | |
| 9 | Шаг изменения времени | | с | 5 | 0 | 0 | 0 | | |
| 10 | Количество подинтервалов | | | 6 | 0 | 0 | 0 | | |
| 11 | | | | 7 | 0 | 0 | 0 | | |
| 12 | | | | 8 | 0 | 0 | 0 | | |
| 13 | | | | 9 | 0 | 0 | 0 | | |
| 14 | | | | 10 | 0 | 0 | 0 | | |
| 15 | | | | 11 | 0 | 0 | 0 | | |
| 16 | | | | 12 | 0 | 0 | 0 | | |
| 17 | | | | 13 | 0 | 0 | 0 | | |
| 18 | | | | 14 | 0 | 0 | 0 | | |
| 19 | | | | 15 | 0 | 0 | 0 | | |
| 20 | | | | 16 | 0 | 0 | 0 | | |
| 21 | | | | 17 | 0 | 0 | 0 | | |
| 22 | | | | 18 | 0 | 0 | 0 | | |
| 23 | | | | 19 | 0 | 0 | 0 | | |
| 24 | | | | 20 | 0 | 0 | 0 | | |
| 25 | | | | 21 | 0 | 0 | 0 | | |
| 26 | | | | | | | | | |
| 27 | | | | | | | | | |
| 28 | | | | | | | | | |

Рис. 91. Шаблон таблицы задачи 2.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 2.

- Для практической работы № 30 составьте компьютерную модель движения тела с ускорением в электронных таблицах.
- Для примера можете выбрать следующие значения: $v_{0y} = 5$ м/с; $y_0 = 0$ м; $a_y = 2$ м/с²; $t_0 = t_{\min} = 0$ с; $t_{\max} = 10$ с, $n = 20$.
- Заполните таблицу "Время-Скорость-Координата", вставив расчетные формулы. При составлении формул не забывайте использовать абсолютные ссылки для постоянных величин.
- Выделите диапазон ячеек таблицы расчетных данных и задайте для него наиболее подходящий формат чисел (числовой или экспоненциальный).
- Измените имя листа на *Равноускоренное движение*.

6. Постройте графики зависимости скорости от времени и координаты от времени на отдельных диаграммах, диаграммы должны иметь заголовок (название графика) и подписи каждой из осей с указанием условного обозначения и единиц измерения величин, откладываемых по этим осям.

7. Проверьте, влияет ли шаг изменения времени (т.е. количество временных подинтервалов) на вид графиков.

8. Изменяя начальные данные, проследите за изменением вида графиков. Каков физический смысл отрицательных значений величин скорости, ускорения и координаты?

9. Сравните движение тел с разным ускорением.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 3.

В редакторе электронных таблиц решите графически задачу: из пункта А выехал грузовик с постоянной скоростью 52 км/ч. Одновременно с ним из пункта В, отстоящего от А на расстоянии 2,5 км, начал двигаться мотоциклист. Считая движение мотоциклиста равноускоренным с $a=3 \text{ м/с}^2$, определить с помощью соответствующих графиков время, через которое мотоциклист догонит грузовик, и путь, пройденный каждым из них до встречи. Для более точного определения величин по графику рекомендуется включить, помимо основных, промежуточные линии.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Постановка задачи:

Построить компьютерную модель для задач типа тех, что решались в рамках практических заданий №29, используя язык Python (библиотеки Matplotlib и Numpy).

Возможное решение:

```
In [3]: import numpy as np
import matplotlib.pyplot as plt
v = float(input())
x0 = float(input())
t = np.linspace(0, 10, 20)
x = x0 + v * t
plt.title("Равномерное прямолинейное движение")
plt.xlabel("Время, с")
plt.ylabel("Координата, м")
plt.grid()
plt.plot(t, x)
plt.show()
```

5

0

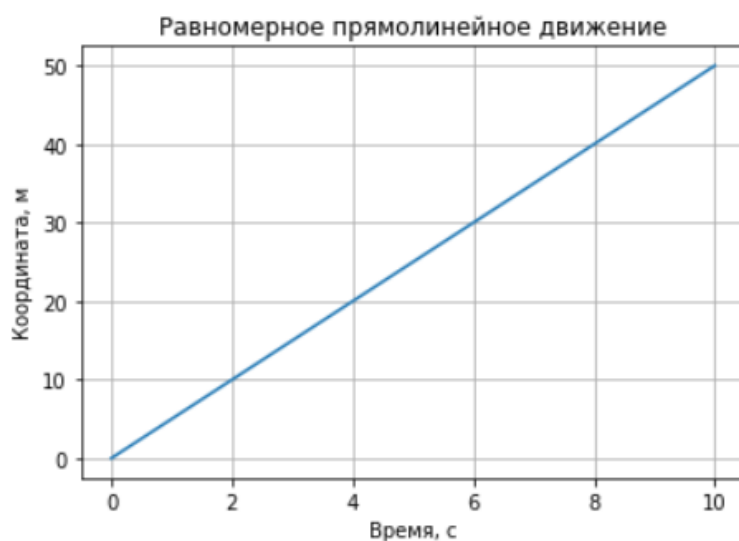


Рис. 92. Компьютерная модель для задачи 1 в Python

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 4.

Привести решение практического задания №30 на Python.

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. <http://window.edu.ru/resource/908/23908/files/excel.pdf>

§3.6 Моделирование биологических процессов

ВСПОМИНАЕМ

Вспомним:

1. Какой язык называется формальным?
2. Какие статистические характеристики набора данных вам известны?

ЧИТАЕМ

Наиболее успешные модели сделаны в содружестве специалистов – математиков, физиков, биологов, хорошо знающих объект моделирования. При этом наиболее трудная часть совместной работы – это формализация знаний об объекте на языке, который может затем быть переформулирован в математическую или компьютерную модель.

Рассмотрим модели, разработанные совместно математиками и биологами.

Среди математических моделей биологических систем можно выделить **регрессионные, качественные и имитационные**.

Регрессионные модели используют формулы, описывающие связь различных характеристик системы, но не объясняющие физического или биологического смысла этих зависимостей.

Для построения регрессионной модели достаточно статистически достоверных корреляций (количественно выраженных взаимосвязей) между переменными или параметрами системы.

Коэффициенты в регрессионных моделях обычно определяются по экспериментальным данным.

Имитационные модели конкретных сложных живых систем, как правило, максимально учитывают имеющуюся информацию об объекте.

По существу, создание имитационной модели включает путь последовательных приближений, в процессе которых получается новая информация об объекте моделирования, совершенствуется система наблюдений, проверяются гипотезы о механизмах тех или иных процессов в рамках общей имитационной системы.

Примеры имитационных моделей

1. *Молекулярная динамика* – в моделях задаются координаты и импульсы всех атомов, составляющих биомолекулу и законы их взаимодействия. Вычисляемая на компьютере картина "жизни" системы позволяет проследить, как физические законы проявляются в функционировании простейших биологических объектов – биомолекул и их окружения. Модели, в которых элементами являются группы атомов, используются в современной технике компьютерного конструирования биотехнологических катализаторов и лекарственных препаратов.

2. *Имитационные модели* созданы для описания физиологических процессов, происходящих в жизненно важных органах: нервном волокне, сердце, мозге, желудочно-кишечном тракте, кровеносном русле. На них проигрываются "сценарии" процессов, протекающих в норме и при различных

патологиях, исследуется влияние на процессы различных внешних воздействий, в том числе лекарственных препаратов.

3. *Имитационные модели* для описания продукционного процесса растений применяются для разработки оптимального режима выращивания растений с целью получения максимального урожая или получения наиболее равномерно распределенного во времени созревания плодов.

4. *Модели для описания экологических систем* – сложных образований, включающих множество биологических, геологических, метеорологических и прочих факторов. Такая модель может быть успешно использована для выбора оптимальной стратегии эксплуатации природной экосистемы или оптимального способа создания искусственной экосистемы. Например, разработка оптимальной стратегии вылова рыбы.

5. *Модели глобальной динамики.* Модели роста послужили предостережением человечеству о том, что Земля – ограниченная система, безудержный прогресс ведет к истощению ее ресурсов, и человечество ждет глобальный экологический кризис.

Модель ядерной зимы наглядно показала, что глобальная ядерная война приведет к уничтожению как побежденных, так и победителей.

Существуют глобальные модели, позволяющие рассчитать «парниковый эффект» и другие процессы, протекающие в глобальном масштабе.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №31

Рассмотрим построение биологических моделей в электронных таблицах.

Условие:

Моделирование эпидемии вирусной инфекции (модель ограниченного роста). Пусть при эпидемии вирусной инфекции число больных N изменяется по формуле $N_{k+1} = N_k + Z_{k+1} - V_{k+1}$, где Z_k – количество заболевших в k -й день, а V_k – количество выздоровевших в тот же день.

Число заболевших рассчитывается согласно модели ограниченного роста: $Z_{k+1} = R * ((L - N_k - I_k)/L) * N_k$,

где L – общая численность жителей, R – коэффициент роста и I_k – число переболевших (тех, кто уже переболел и выздоровел, и поэтому больше не болеет): $I_{k+1} = I_k + V_{k+1}$.

Рассмотрим ситуацию, в которой в начале эпидемии заболел 1 человек, все заболевшие выздоравливают через 6 дней и больше не болеют.

Выполните моделирование развития эпидемии при $L = 2000$ и $R = 0,6$ до того момента, когда количество больных станет равно нулю. Постройте график изменения количества больных.

| | A | B | C | D | E | F | G |
|----|------------------|------|---------|-------------------------|----------------------------|----------------------|--------------------|
| 1 | Коэффициент R= | 0,6 | | Количество заболевших Z | Количество выздоровевших I | Количество переболев | Количество больных |
| 2 | Число жителей L= | 2000 | день 1 | 1 | 0 | 0 | 1 |
| 3 | | | день 2 | 1 | 0 | 0 | 2 |
| 4 | | | день 3 | 1 | 0 | 0 | 3 |
| 5 | | | день 4 | 2 | 0 | 0 | 4 |
| 6 | | | день 5 | 2 | 0 | 0 | 7 |
| 7 | | | день 6 | 4 | 0 | 0 | 10 |
| 8 | | | день 7 | 6 | 0 | 0 | 17 |
| 9 | | | день 8 | 10 | 1 | 1 | 26 |
| 10 | | | день 9 | 15 | 1 | 2 | 40 |
| 11 | | | день 10 | 24 | 1 | 3 | 63 |
| 12 | | | день 11 | 36 | 2 | 4 | 98 |
| 13 | | | день 12 | 56 | 2 | 7 | 151 |
| 14 | | | день 13 | 83 | 4 | 10 | 231 |
| 15 | | | день 14 | 122 | 6 | 17 | 346 |
| 16 | | | день 15 | 170 | 10 | 27 | 506 |

Рис. 93. Модель эпидемии вирусной инфекции

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1.

Ответьте на следующие вопросы по результатам Практического задания №31:

1. Когда закончится эпидемия?
2. Сколько человек переболеет, а сколько вообще не заболеет?
3. Каково максимальное число больных в один день?
4. Изменяя коэффициент R, определите, при каких значениях модель явно перестает быть адекватной.
5. Как изменится ситуация, если считать, что переболевшие могут вновь заболевать? Постройте соответствующую модель.

ЧИТАЕМ

Качественные (базовые) модели

В любой науке существуют простые модели, которые поддаются аналитическому исследованию и обладают свойствами, которые позволяют описывать целый спектр природных явлений. Такие модели называют **базовыми**.

После того, как досконально математически изучена суть процессов на такой базовой модели, по аналогии становятся понятными явления, происходящие в гораздо более сложных реальных системах.

Нелинейность — неотъемлемое свойство базовых систем математической биологии. Несмотря на огромное разнообразие живых систем, можно выделить некоторые важнейшие присущие им качественные свойства: рост, самоограничение роста, существование в двух или нескольких стационарных режимах, автоколебательные режимы (биоритмы) и пр. Все эти свойства можно продемонстрировать на сравнительно простых нелинейных динамических моделях, которые и выступают в роли базовых моделей математической биологии.

Модели роста популяций

Неограниченный рост. Экспоненциальный рост.

Одно из фундаментальных предположений, лежащих в основе всех моделей роста — пропорциональность скорости роста численности популяции, будь то популяция зайцев или популяция клеток. В основе этого предположения лежит тот общеизвестный факт, что важнейшей характеристикой живых систем является их способность к размножению. Для многих одноклеточных организмов или клеток, входящих в состав клеточных тканей — это просто деление, то есть удвоение числа клеток через определенный интервал времени, называемый характерным временем деления. Для сложно организованных растений и животных размножение происходит по более сложному закону, но в простейшей модели можно предположить, что скорость размножения вида пропорциональна численности этого вида.

Математически это записывается с помощью уравнения, линейного относительно переменной x , характеризующей численность (концентрацию) особей в популяции:

$$dx/dt = Rx \quad (1).$$

Здесь R в общем случае может быть функцией как самой численности, так и времени, или зависеть от других внешних и внутренних факторов.

Согласно закону (1), если коэффициент пропорциональности **$R=\text{const}$** , численность будет расти неограниченно по экспоненте. В этих моделях увеличение скорости изменения величины связано с ростом самой этой величины.

При этом производство продовольствия и других товаров растет линейно, и, следовательно, популяция, растущая экспоненциально, обречена на голод.

Для большинства популяций существуют ограничивающие факторы роста популяции прекращается. Единственное исключение представляет человеческая популяция, которая на протяжении всего исторического времени растет даже быстрее, чем по экспоненте. Закон экспоненциального роста

справедлив на определенной стадии роста для популяций клеток в ткани, водорослей или бактерий в культуре.

Ограниченный рост

Базовой моделью, описывающей ограниченный рост, является модель:

$$dx/dt = Rx(1 - x/K) \quad (2).$$

Здесь параметр K носит название "емкости популяции" и выражается в единицах численности (или концентрации). Он не имеет какого-либо простого физического или биологического смысла и носит системный характер, то есть определяется целым рядом различных обстоятельств.

Рассмотрим численность популяции в последовательные моменты времени. Это соответствует реальной процедуре пересчета особей (или клеток) в популяции. В самом простом виде зависимость численности на временном шаге номер $n+1$ от численности предыдущем шаге n можно записать в виде:

$$x_{n+1} = Rx_n(1 - x_n) \quad (3).$$

Решение подобных разностных уравнений лежит в основе моделирования любых реальных биологических процессов.

Интересный факт. Если уравнение ограниченного роста (3) переписать в виде $x_{n+1} = x_n^2 + C$ (4) и построить в комплексной плоскости, то получится известный фрактал – множество Мандельброта.

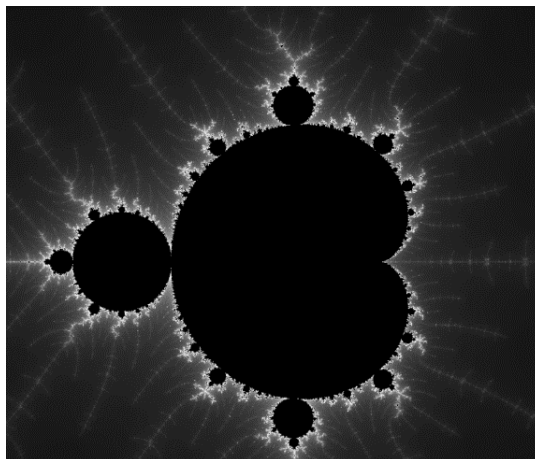


Рис.94. Множество Мандельброта

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №32

Представьте себе, что на Земле останется только один источник пресной воды — озеро Байкал. На сколько лет Байкал обеспечит население всего мира водой?

Для построения математической модели определим исходные данные. Обозначим:

V – объем озера Байкал 23000 км³;

N – население Земли на конец 2020 года 7 830 458 560 чел.;

p – потребление воды в день на 1 человека (в среднем) 300 л.

Так как 1 л. = 1 дм³ воды, необходимо выполнить перевод V воды озера из км³ в дм³.

$$V (\text{км}^3) = V * 109 (\text{м}^3) = V * 1012 (\text{дм}^3)$$

Результат — количество лет, за которое население Земли использует воды Байкала, обозначим Y . Итак, $Y = (V * 1000000000000) / (N * p * 365)$.

| | A | B | C |
|---|--|---------------|-----------------------------|
| 1 | V – объем озера Байкал 23000 км ³ ; | 23000 | На сколько лет хватит воды: |
| 2 | N – население Земли на конец 2020 года 7 830 458 560 чел.; | 7 830 458 560 | 26,82418411 |
| 3 | p – потребление воды в день на 1 человека (в среднем) 300 л. | 300 | |
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |
| 8 | | | |

Рис. 95. Модель обеспечения населения Земли водой озера Байкал

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 2.

Ответьте на вопросы, выполнив вычислительный эксперимент:

1. Сколько лет можно будет пользоваться водами Байкала, если потребление воды увеличится до 400 литров на человека?
2. Сколько лет можно будет пользоваться водами Байкала, если население Земли уменьшится до 6,5 млрд. чел.?

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №33

Для производства вакцины на заводе планируется выращивать культуру бактерий. Известно, что если масса бактерий — x г., то через день она увеличится на $(a-bx)x$ г., где коэффициенты a и b зависят от вида бактерий. Завод ежедневно будет забирать для нужд производства вакцины m г. бактерий. Для составления плана важно знать, как изменяется масса бактерий через 1, 2, 3, ..., 30 дней.

Исходные данные:

a и b – коэффициенты;

x_0 – начальная масса бактерий;

m – масса бактерий, забираемых для нужд производства;

Количество бактерий каждого следующего дня зависит от количества бактерий предыдущего дня и вычисляется по формуле:

$x_{i+1} = x_i + (a - b \cdot x_i) \cdot x_i - m$ – масса бактерий в следующий день.

Результатами являются значения массы бактерий через 1, 2, 3, 4 ... 30 дней.

Введите в компьютерную модель исходные данные (например, $a=1$, $b=0.0001$, $m=2000$, $x_0=12000$) и **постройте график** зависимости массы бактерий от количества дней.

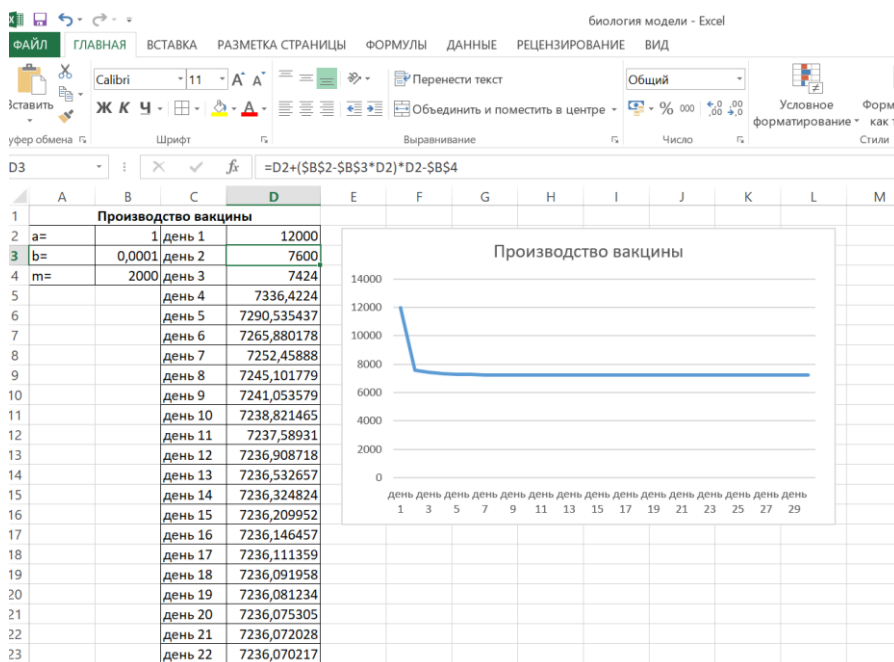


Рис. 96. Производство вакцины

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 3.

Опираясь на результаты Практического задания №33 определите:

1. Как изменяется масса бактерий в течение месяца?
2. Что произойдет к концу месяца, если увеличить начальную массу бактерий? Проведите эксперимент, взяв начальную массу 13000 г., 14000 г., 17000 г., 18000 г. Постройте соответствующие графики зависимости массы бактерий от количества дней. Выполните анализ полученных результатов.

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. https://dspace.kpfu.ru/xmlui/bitstream/handle/net/22207/06_44_001048.pdf
2. <http://www.library.biophys.msu.ru/MathMod/>

3. <http://www.michurin.net/online-tools/mandelbrot.html>

§3.7 Моделирование экономических процессов

ВСПОМИНАЕМ

Вспомним:

1. Вспомните, что такое системный подход?
2. В чем заключается анализ? Синтез?

ЧИТАЕМ

Замечено, что в экономике действуют устойчивые количественные закономерности. Следовательно, возможно формализованное описание этих закономерностей на языке математики и изучение полученных математических моделей экономических процессов с использованием системного подхода.

Математическое моделирование сложных экономических систем включает построение формальных моделей изучаемых процессов, алгоритмизацию полученных моделей и их машинную реализацию, выполнение расчетов, анализ и интерпретацию полученных результатов.

Рассматривая экономику как систему, можно отметить, что основной целью экономики является обеспечение общества предметами потребления, в том числе создающими условия для безопасности общества, и что элементы экономики – это хозяйственные единицы (предприятия, фирмы, банки и т. п.). Надсистему национальной экономики образуют природа, мировая экономика, общество.

Построение и анализ математических моделей – основной метод экономических исследований.

Математическое моделирование используется для решения широкого круга практических задач экономического характера: это задачи о перевозках, складировании, распределении ресурсов, задачи календарного планирования, оценки эффективности и рисков инвестиционных проектов и многие другие.

Широкое распространение математического моделирования в экономике в значительной степени обусловлено развитием информационных инструментальных средств, которые позволяют переводить экономико-математические модели из классической символьной формы представления в компьютерную.

Экономисты в своей практике часто сталкиваются с оптимизационными задачами с наличием ограничений. К таким задачам относятся, например, задача о производстве продукции, на которое накладываются определённые ограничения на материальные и трудовые ресурсы, задачи обеспечения

материалами производства продукта, задача минимизации расходных материалов для производства продукции, задачи на оптимизацию деятельности в максимально короткий срок. При всей разнородности перечисленных задач каждой из них присущи следующие свойства:

1. наличие альтернативных решений
2. наличие цели
3. наличие ограничивающих факторов.

Первое свойство указывает на то, что для каждой задачи существует как минимум 2 варианта её решения, и можно выбрать решение, которое в наибольшей степени способствует достижению цели и является оптимальным. Общую постановку этих задач можно представить следующим формальным образом: найти оптимум целевой функции задачи при заданных ограничениях.

Под оптимизацией процесса понимают поиск экстремумов целевой функции либо поиск значений ее аргументов, при которых функция принимает заданное значение. На эти аргументы, называемые также параметрами оптимизации, могут быть наложены ограничения разного рода. В задачах линейной оптимизации целевая функция и ограничения линейно зависят от переменных. Такие задачи называют также задачами **линейного программирования**.

Универсальными инструментальными средствами создания модели являются языки программирования. Наряду с этим существует множество специализированных средств моделирования, позволяющих быстрее и с меньшими затратами по сравнению с универсальными языками программирования создавать и исследовать модель.

ГОТОВИМСЯ К РАБОТЕ НА КОМПЬЮТЕРЕ

Рассмотрим в качестве среды моделирования табличный процессор Excel, который имеет специальные программные надстройки и развитую библиотеку аналитико-расчётных функций для решения широкого класса экономических задач и экономического анализа.

Выполним построение компьютерных моделей с помощью надстройки в Excel «Поиск решения». Эта надстройка позволяет моделировать и находить решения практически для всех классов оптимизационных задач экономического анализа.

Запускаем Excel. Для того, чтобы установить надстройку, необходимо выполнить последовательность команд: Файл – Параметры – Надстройки – (Управление – Надстройки Excel – Перейти) – Выбрать «Поиск решения» – ОК.

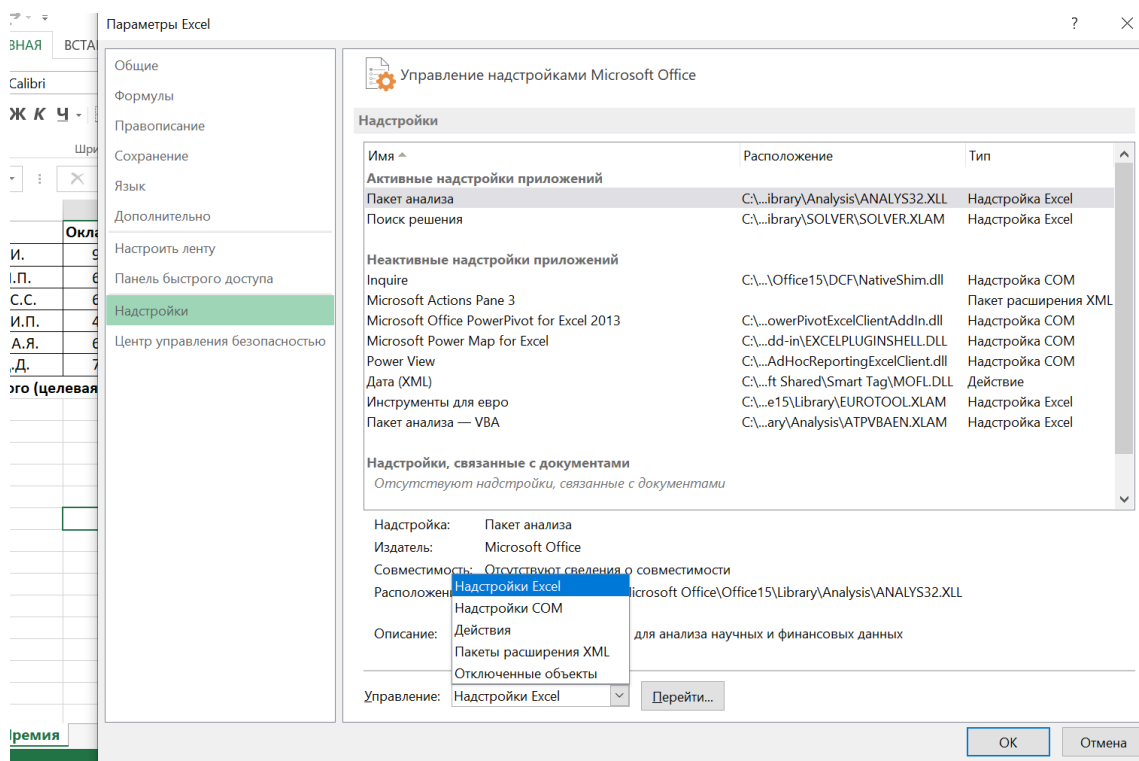


Рис. 97. Выбор надстроек в Параметрах Excel

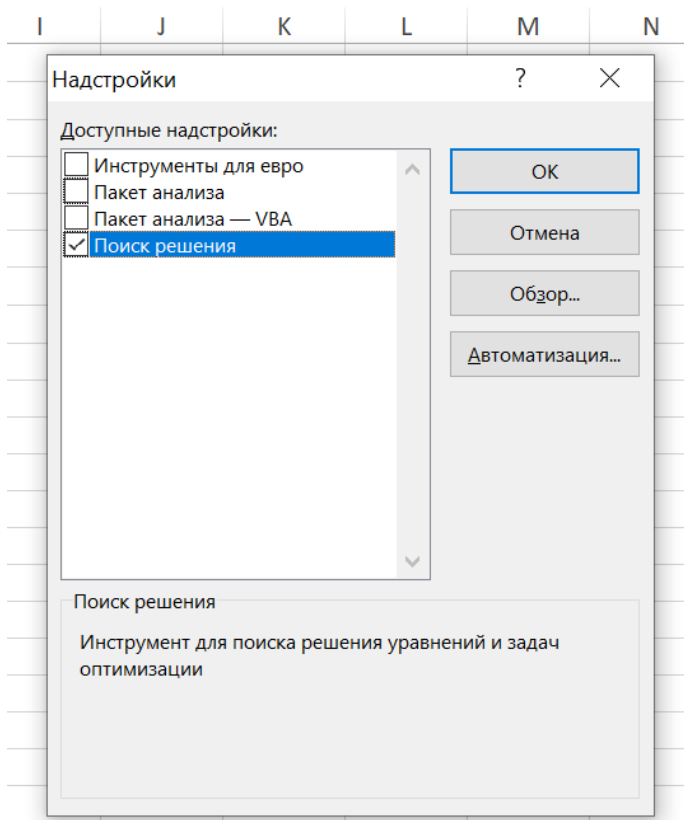


Рис. 98. Выбор надстройки Поиск решения

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №34

Условие:

На предприятии между сотрудниками необходимо распределить премию. Чтобы распределение было максимально справедливым, необходимо каждому сотруднику выделить сумму, пропорциональную его окладу.

Порядок выполнения задания:

Пусть необходимо распределить премию в размере 500000 рублей. Шаблон таблицы приведен на Рисунке 99. Введите данные в соответствующие ячейки. Премия равна окладу, умноженному на единый для всех коэффициент.

| | A | B | C | D | E | F | G | H |
|----|-------------------------|-------------|--------------|---|-------------|---|---|---|
| 1 | Фамилия | Оклад, руб. | Премия, руб. | | Коэффициент | | | |
| 2 | Иванов И.И. | 90 000,00 | 0,00 | | | | | |
| 3 | Петрова П.П. | 68 000,00 | 0,00 | | | | | |
| 4 | Соколова С.С. | 66 000,00 | 0,00 | | | | | |
| 5 | Суворова И.П. | 48 000,00 | 0,00 | | | | | |
| 6 | Хлопонин А.Я. | 62 000,00 | 0,00 | | | | | |
| 7 | Яковлев Д.Д. | 76 000,00 | 0,00 | | | | | |
| 8 | Итого (целевая функция) | | 0 | | | | | |
| 9 | | | | | | | | |
| 10 | | | | | | | | |
| 11 | | | | | | | | |
| 12 | | | | | | | | |
| 13 | | | | | | | | |

Рис. 99. Условия задачи 1.

Во вкладке *Данные* нажмите кнопку *Поиск решения*. Выбираем параметры *Поиска Решения*:

1. Оптимизировать целевую функцию – это ячейка C8, по условию значение премии должно быть равно 500000.
2. Указываем, какие ячейки нужно изменять, чтобы получить оптимальное решение. В нашем случае это коэффициент E2.
3. Указываем ограничения – это кнопка *Добавить*. В нашем случае коэффициент в E2 должен быть неотрицательным.
4. Нажимаем *Найти решение*.

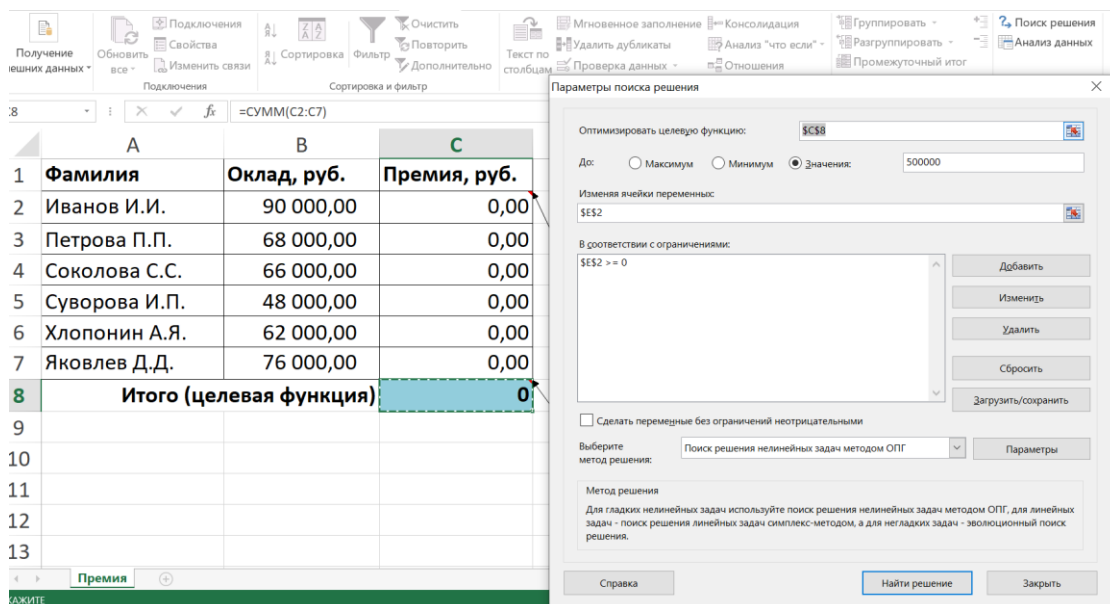


Рис. 100. Заполнение параметров Поиска решения

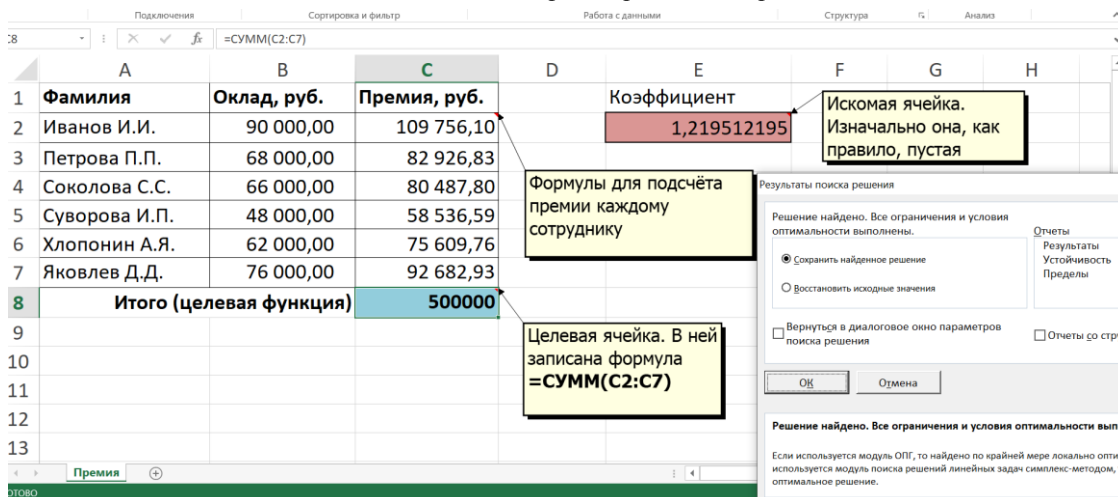


Рис. 101. Найденное решение.

Получаем коэффициент 1,219512195.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №35

Условие:

Предприятие производит две модели изделия из пластика. Для производства модели 1 требуется 2 листа пластика, 0,2 ч машинного времени, прибыль с одного изделия – 60 р. Для производства модели 2 требуется 3 листа пластика, в два раза больше машинного времени, прибыль с одного изделия при этом – 100 р. На производство изделий может быть затрачено не больше 1600 листов пластика и 200 ч машинного времени. Какое количество изделий каждой модели необходимо выпускать, чтобы получить максимальную прибыль?

Порядок выполнения задания:

Заполним таблицу согласно Рисунку 102.

| | A | B | C | D | E | F | G |
|----|-------------------------------|-----------------------------|-----------------------------|------|---|---|---|
| 1 | | Исходные данные | | | | | |
| 2 | | Модель 1 | Модель 2 | | | | |
| 3 | требуется листов пластика, шт | 2 | 3 | | | | |
| 4 | требуется маш. времени, ч | 0.2 | 0.4 | | | | |
| 5 | прибыль, руб. | 60 | 100 | | | | |
| 6 | | Искомые значения | | | | | |
| 7 | | Кол-во 1 модели изделия, шт | Кол-во 2 модели изделия, шт | | | | |
| 8 | | 0 | 0 | | | | |
| 9 | | | | | | | |
| 10 | | Целевая функция | | | | | |
| 11 | | 0 | шт | | | | |
| 12 | | | | | | | |
| 13 | | Ограничения | | | | | |
| 14 | | | | | | | |
| 15 | Всего листов пластика | 0 | <= | 1600 | | | |
| 16 | Всего маш. времени, ч | 0 | <= | 200 | | | |
| 17 | | | | | | | |
| 18 | | | | | | | |
| 19 | | | | | | | |
| 20 | | | | | | | |
| 21 | | | | | | | |

Рис. 102. Условия задачи 2.

Целевая функция – суммарная прибыль по всем изделиям, выпущенным предприятием. Необходимо найти экстремум (максимум) целевой функции. Заполним параметры *Поиска решения* в соответствии с таблицей (Рисунок 103). К ограничениям, указанным в условиях задачи, добавим, что количество изделий должно быть целым и неотрицательным.

Параметры поиска решения

Оптимизировать целевую функцию: \$B\$12

До: ☒ Максимум ☐ Минимум ☐ Значения: 0

Изменяемые ячейки: \$B\$9:\$C\$9

В соответствии с ограничениями:

- \$B\$15 <= \$D\$15
- \$B\$16 <= \$D\$16
- \$B\$9:\$C\$9 = целое
- \$B\$9:\$C\$9 >= 0

☐ Сделать переменные без ограничений неотрицательными

Выберите метод решения: Поиск решения нелинейных задач методом ОПГ

Метод решения: Для гладких нелинейных задач используйте поиск решения нелинейных задач методом ОПГ, для линейных задач - поиск решения линейных задач симплекс-методом, а для негладких задач - эволюционный поиск решения.

Найти решение

Рис. 103. Параметры Поиска решения задачи 2.

Вопрос:

Какое количество изделий каждого вида должно выпустить предприятие согласно построенной модели?

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1.

С помощью сервиса *Поиск решения* решить задачу:

Намечается выпуск двух видов костюмов – мужских и женских. На женский костюм требуется 1 м шерсти, 2 м сатина и 1 день трудозатрат. На мужской костюм – 3,5 м шерсти, 0,5 м сатина и 1 день трудозатрат. Всего имеется 350 м шерсти, 240 м сатина и 150 дней трудозатрат. Требуется определить, сколько костюмов каждого вида необходимо сшить, чтобы обеспечить максимальную прибыль, если прибыль от реализации женского костюма составляет 100 рублей, а от мужского – 200 рублей. При этом следует иметь в виду, что необходимо сшить не менее 60 мужских костюмов.

ВСПОМИНАЕМ

Вспомним:

1. Какие знания называются декларативными?
2. Какие знания называются процедурными?

ЧИТАЕМ

Подбор параметра

Для решения экономических задач часто используют процедуру **Подбор параметра**. Данная процедура относится к технологиям анализа целевой функции или технологиям «Что – если» – анализа. Подбор параметра – это способ поиска определенного значения ячейки путем изменения значения в другой ячейке. При этом значение в ячейке изменяется до тех пор, пока формула, зависящая от этой ячейки, не вернет требуемый результат.

ГОТОВИМСЯ К РАБОТЕ НА КОМПЬЮТЕРЕ

Чтобы воспользоваться процедурой *Подбор параметра*, необходимо выполнить следующую последовательность действий:

1. Выберите команду *Подбор параметра* в меню *Данные/Анализ «Что-если»*.
2. В поле *Установить в ячейке* введите ссылку на ячейку, содержащую необходимую формулу.
3. Введите искомый результат в поле *Значение*.
4. В поле *Изменяя значение ячейки* введите ссылку на ячейку, значение которой нужно подобрать. Формула в ячейке, указанной в поле *Установить в ячейке*, должна ссылаться на эту ячейку.
5. Нажмите кнопку *ОК*.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №36

Условие:

Какое количество товара по цене 300 руб. за шт. необходимо закупить, чтобы после его реализации по цене 500 руб. за шт., уплаты налога 40% с прибыли, уплаты стоимости за место торговли 400 руб. получить прибыль 20000 руб.?

Порядок выполнения задания:

Заполните таблицу по образцу на Рисунке 104.

| | A | B | C | D | E | F | G |
|---|-------------------|---------------------|------------------------|-------|-----------------------|-----------------|---------|
| 1 | Количество товара | Цена закупки за шт. | Цена реализации за шт. | Доход | Налог на прибыль, 40% | Стоимость места | Прибыль |
| 2 | | 300 | 500 | 0 | 0 | 400 | -400 |
| 3 | | | | | | | |
| 4 | | | | | | | |
| 5 | | | | | | | |
| 6 | | | | | | | |
| 7 | | | | | | | |
| 8 | | | | | | | |

Рис. 104. Шаблон таблицы для задачи 1.

Выберите команду *Подбор параметра* в меню *Данные/Анализ «Что-если»*.

В поле *Установить в ячейке* введите ссылку на ячейку G2, содержащую необходимую формулу.

Введите искомый результат 20000 в поле *Значение*.

В поле *Изменяя значение ячейки* введите ссылку на ячейку A2. Нажмите кнопку ОК.

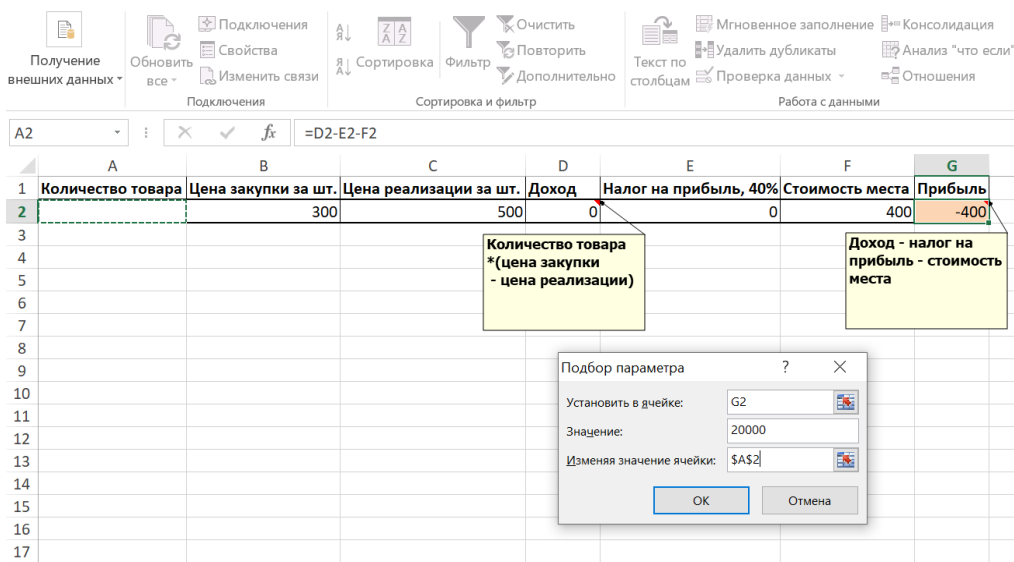


Рис. 105. Подбор параметра

Полученный результат показан на Рисунке 106.

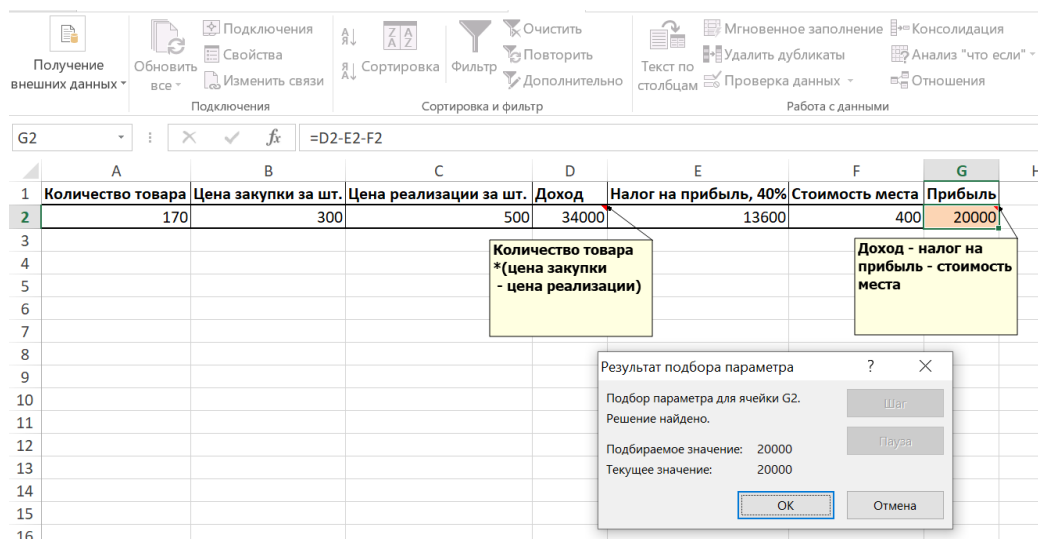


Рис. 106. Результат подбора параметра.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 2.

Какую сумму надо затратить на приобретение товара по 30 руб. за шт., чтобы после его реализации по 40 руб. за шт. получить прибыль 10000 руб., если налог составляет 60% от прибыли?

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 3.

Дворник убирает территорию размером 100 м². За уборку 0,5 м³ снега ему платят 35 руб. Сколько должно выпасть снега (какова должна быть толщина снежного покрова), чтобы дворник смог убрать его и купить себе новые ботинки по цене 1000 руб.?

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. <http://www.lib.uniur.ac.ru/edocs/iuni/20070818.pdf>
2. <https://bibl.nngasu.ru/electronicresources/uch-metod/management/4889.pdf>
3. <http://iside.distance.ru/w/Books/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D1%8D%D0%BA%D0%BE%D0%BD%D0%BE%D0%BC%20%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D0%B2%20%D0%B8%20%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC.pdf>

§3.8 Стохастическое моделирование

ВСПОМИНАЕМ

Вспомним:

1. По какому основанию классификации модели делятся на детерминированные и стохастические?
2. Что такое вероятность события?
3. Как получить случайное число в Python?
4. Как работает генератор случайных чисел?

ЧИТАЕМ

Стохастическая модель – это модель, в которой учитываются случайные факторы.

При исследовании сложных систем со случайными возмущениями строятся вероятностные имитационные модели, в которых учитывается влияние случайных факторов. Результаты, полученные при воспроизведении на такой модели рассматриваемого процесса, являются случайными реализациями. Поэтому для нахождения объективных и более точных характеристик процесса требуется его многократное воспроизведение, с последующей статистической обработкой полученных данных. Такое исследование с помощью имитационного моделирования называется **статистическим моделированием**.

Статистическая модель случайного процесса – это алгоритм, с помощью которого имитируют работу сложной системы, взаимодействие элементов системы, носящих вероятностный характер.

Методика статистического моделирования называется также **методом Монте-Карло**. Название "метод Монте-Карло" (ММК) дано в честь города Монте-Карло, который широко известен своими казино. Генератор случайных чисел, используемый в ММК, является имитацией рулетки в казино.

Возникновение идеи использования случайных явлений в области приближенных вычислений принято относить к 1878 г., когда появилась работа Холла об определении числа π с помощью случайных бросаний иглы на разграфленную параллельными линиями бумагу. Рассматривается событие, вероятность которого выражается через число π , и она приближенно оценивается с помощью вычислительных экспериментов.

К разделам науки, в которых используется метод Монте-Карло, относятся задачи теории массового обслуживания, теории игр и математической экономики, теории передачи сообщений при наличии помех и ряд других.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Пример. Рассмотрим классическую задачу, которую можно решить с использованием метода Монте-Карло: **вычислить значение числа π** .

Идея определения числа π заключается в следующем: при радиусе круга $r = 1$, его площадь равна π . Рассмотрим четверть круга, тогда $4S = \pi$ (1), если S – площадь четверти круга. Если мы определим площадь четверти круга, то мы сможем определить значение числа π по формуле (1). Для определения площади круга используем метод Монте-Карло, который требует применения случайных чисел. Экспериментально случайные числа можно получить как при помощи рулетки, так и при помощи, например, дождя. Для понимания идеи рассмотрим способ нахождения числа π с помощью дождя. Для опыта приготовим кусок картона, нарисует на нем квадрат и впишем в квадрат четверть круга (Рисунок 107).

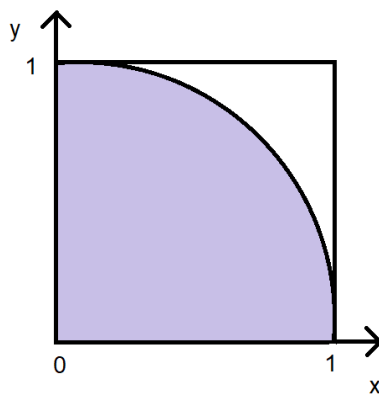


Рис. 107. Чертеж для задачи о нахождении числа π .

Если такой чертеж некоторое время подержать под дождем, то на его поверхности останутся следы капель. Подсчитаем число следов внутри квадрата и внутри четверти круга. Их отношение будет приблизительно равно отношению площадей этих фигур, так как попадание капель в различные места чертежа равновероятно. Т.к. площадь квадрата равна 1, то отношение и будет равно площади четверти круга, откуда по формуле (1) найдем число π .

Таким образом, значение числа π можно найти, определив число капель в кругу и число капель в квадрате.

Реальный дождь можно заменить виртуальной таблицей случайных чисел, которая составляется с помощью компьютера по специальной программе (генератора псевдослучайных чисел). Каждому следу капли поставим в соответствие два случайных числа x и y , характеризующих его положение вдоль осей Ox и Oy . Числа должны попадать в интервал от 0 до 1. В каждом языке программирования предусмотрена возможность получения случайного числа в заданном интервале.

Эти числа будем считать координатами капли, т. е. капля как будто попала в точку $(x; y)$. Аналогично поступаем и со всеми выбранными случайными числами. Если окажется, что для точки $(x_i; y_i)$ выполняется

неравенство $x_i^2 + y_i^2 > 1$, то, значит, она лежит вне круга. Если $x_i^2 + y_i^2 \leq 1$, то точка лежит внутри круга. Чем больше точек смоделировать, тем точнее получится результат.

Широкое применение метода статистических испытаний (Монте-Карло) стало возможным благодаря компьютерам.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Пример реализации метода Монте-Карло для нахождения числа π на языке Питон:

```
from random import random
def MMC(kol_tochek):
    i = 0
    count = 0
    while i <= kol_tochek:
        x, y = random(), random()
        if x ** 2 + y ** 2 < 1:
            count += 1
        i += 1
    Pi = 4 * count / kol_tochek
    print(Pi)

k = int(input('Введите количество точек в опыте:'))
MMC(k)
```

Процедура MMC реализует метод Монте-Карло (ММК).

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1.

С помощью метода Монте-Карло вычислить площади представленных на рисунках фигур, используя язык программирования.

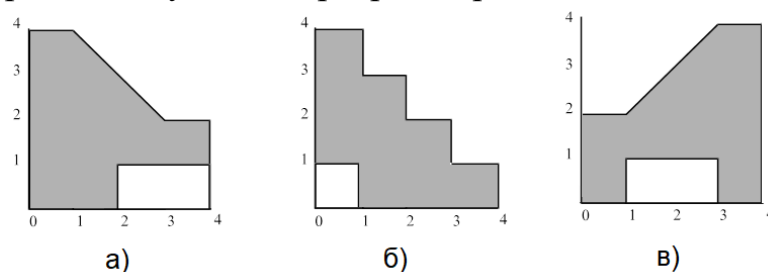


Рис. 108. Вычислите площади фигур методом Монте-Карло.

§3.9 Системы массового обслуживания

ВСПОМИНАЕМ

Вспомним:

1. Что такое структура системы?
2. Как можно показать состав и структуру системы?

ЧИТАЕМ

Магазины, больницы, банки, МФЦ, телефонные компании и т.д. – все это **системы массового обслуживания (СМО)**. В целом эти системы неплохо описываются математическими формулами и моделями, основанными на теории вероятностей и математической статистике.

Каждая СМО состоит из определенного числа обслуживающих единиц, которые называются каналами обслуживания. Элементы СМО:

1. входной поток заявок;
2. очередь;
3. узлы обслуживания;
4. выходной поток.

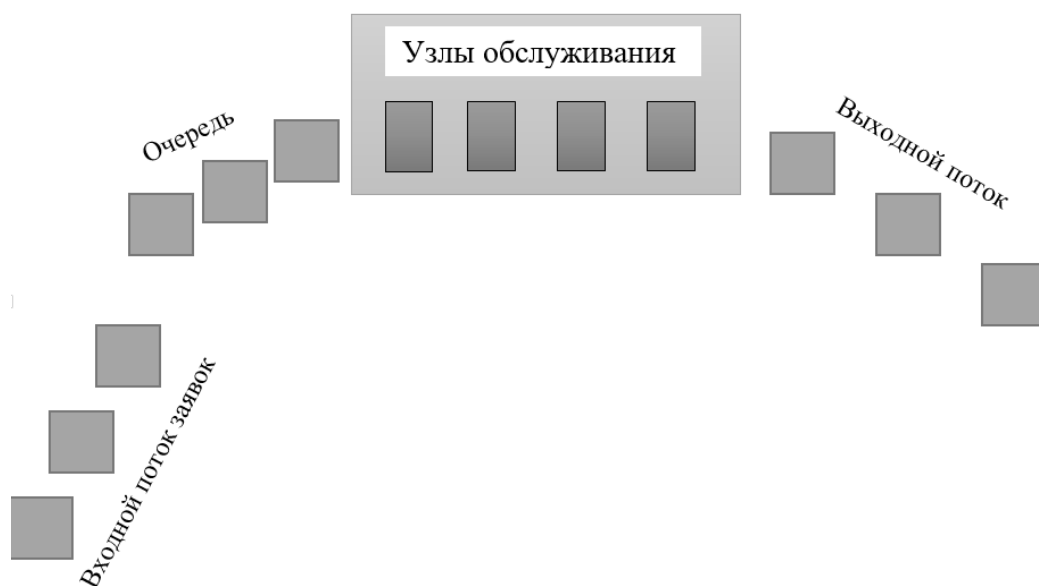


Рис. 109. СМО

Входной поток заявок может быть случайным (время между поступлением двух заявок является случайной величиной) или

детерминированным (поступление происходит в соответствии с определенным графиком).

Очередь – место, где поступившие заявки ждут начала обслуживания. Характеристики очереди:

- **длина** (ограниченная или неограниченная), время ожидания (может быть ограничено),
- **дисциплина** – связана с правилом, в соответствии с которым обслуживаются клиенты: первый пришел – первый ушел (**FIFO** – First in, First out), первый пришел – последний ушел (**LIFO** Last in, First out, структуру с такой дисциплиной доступа называют стек), а также существуют дисциплины с приоритетами.

Для успешного функционирования любой СМО следует оптимизировать уменьшение потерь от длинных очередей с увеличением затрат на содержание дополнительных каналов обслуживания.

Издержки на содержание каналов обслуживания обычно можно считать прямо пропорциональными числу каналов, а издержки от ожидания клиентов (или сотрудников) в очереди – пропорциональными длине очереди или времени ожидания, которые, согласно формулам теории очередей, резко падают с увеличением числа каналов обслуживания. В результате можно оценить оптимальное число каналов обслуживания, минимизирующее полные издержки, связанные с функционированием СМО.

ГОТОВИМСЯ К РАБОТЕ НА КОМПЬЮТЕРЕ

Для оптимизации процесса часто прибегают к имитационному моделированию с помощью специальных программных средств. Электронные таблицы также можно использовать для построения имитационных моделей СМО.

Рассмотрим пример имитационного моделирования СМО в среде ТП Excel с использованием надстройки «Системы массового обслуживания».

Надстройку можно скачать по ссылке: <https://hcx1.net/addins.html>. Для подключения надстройки выполнить команды: Файл – Параметры – Надстройки – Управление надстройками – Перейти – Обзор – Выбрать загруженный файл и нажать ОК.

Надстройка для MS Excel «Системы массового обслуживания» предназначена для расчета параметров и анализа исследуемой СМО со стандартным (пуассоновским) потоком заявок. Нестандартный поток заявок моделируется методом Монте-Карло. Достаточно выбрать вид СМО и ввести ее параметры, и надстройка рассчитает все важные характеристики системы. Результаты выводятся в обычном листе Excel.

Книга1 - Excel

ФАЙЛ ГЛАВНАЯ ВСТАВКА РАЗМЕТКА СТРАНИЦЫ ФОРМУЛЫ ДАННЫЕ РЕЦЕНЗИРОВАНИЕ ВИД **СМО**

Рассчитать

Параметры СМО Модель СМО Метод расчета Потоки заявок и обслуживания Серверы Разное

☒ Неограниченная очередь
☐ Ограниченная очередь
☐ Ограниченная популяция

☒ По формулам ТМО
☐ МК-моделирование

Пуассоновский поток $\lambda = 8$
Экспоненциальное распределение
времени обслуживания $\mu = 10$

$S = 1$ единиц
Количество серверов
в системе

☒ RU
☐ EN
? Справка

| | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | |

Рис. 110. Интерфейс надстройки СМО

Работа с надстройкой начинается с блока "Метод расчета". Здесь имеется два флажка: «Формулы ТМО» и «МК – моделирование».

При выборе «Формулы ТМО» надстройка рассчитывает параметры систем массового обслуживания по известным формулам теории массового обслуживания (ТМО). Достаточно выбрать вид СМО и ввести ее параметры, и надстройка рассчитает все характеристики системы, которые возможно рассчитать в теории.

Монте-Карло-моделирование («МК – моделирование») используется для расчета параметров СМО с помощью простейшего статистического моделирования характеристик в случае, когда требования теории к функции распределения потоков клиентов или обслуживания не удовлетворяются.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Условие:

Компания планирует поставить аппарат по продаже кофе в торговом центре. Она оценивает поток клиентов в 10 человек в час. В среднем автомат тратит 5 минут на обслуживание каждого клиента. Предполагая пуассоновский поток заявок и экспоненциальное распределение для времени обслуживания найти:

- Долю времени, когда автомат загружен;
- Долю времени, когда он бездействует;
- Среднее число клиентов у автомата;
- Среднее число клиентов в очереди у автомата;
- Среднее время, затрачиваемое клиентом для получения напитка;
- Среднее время, которое клиент проводит в очереди;
- С какой вероятностью возле автомата будут стоять более 3 клиентов.

Пример решения:

Программа требует ввода трех характеристик – λ , μ и S :

λ – интенсивность входного потока (среднее число клиентов, приходящих в систему за единицу времени). В нашей задаче $\lambda = 10$.

μ – среднее число клиентов, которых сервер может обслужить в единицу времени, или пропускная способность сервера. Значение μ в нашей задаче рассчитывается так: единица времени 1 час = 60 мин разделить на 5 мин на клиента: $\mu = 12$.

Если величина введенного числа выходит за допустимые смысловые границы, которые устанавливает модель для данного входного параметра, программа выдает предупреждающее сообщение и требует изменить ввод.

S – количество серверов. В модели предполагается, что все серверы совершенно одинаковы, т.е. среднее время обслуживания клиентов строго одинаково (технический предел числа серверов – 30).

Как интенсивность потока, так и скорость обслуживания (или пропускная способность) имеют одинаковую размерность: количество клиентов за единицу времени. Безразлично, какую единицу времени выбрать. Главное, чтобы она была одинакова и для λ , и для μ . В листе результатов расчета будет использована выбранная единица времени.

Обратите внимание, если $\mu * S \leq \lambda$, т.е. входной поток больше или равен суммарному потоку обслуживания, то обслужить всех клиентов невозможно, и очередь неограниченно растет. В этом случае система нестационарная, и расчет не будет выполнен.

После ввода всех параметров, необходимо нажать на кнопку Рассчитать.

Результат вы видите на рисунке 111.

| Параметры СМО | Модель СМО | Метод расчета | Потоки заявок и обслуживания | Серверы | Разное |
|---------------|------------|---------------|------------------------------|---------|--------|
| Р1 | | | | | |

| Результаты | Потоки заявок и обслуживания |
|-------------------|---|
| $\lambda = 10$ | Процент загрузки каждого сервера $\rho = 0,83$ |
| $\mu = 12$ | Среднее число клиентов в системе $L = 5,00$ |
| $S = 1$ | Средняя длина очереди $L_q = 4,17$ |
| $\sigma_\mu = 12$ | Среднее время пребывания в системе $W = 0,50$ |
| | Среднее время ожидания в очереди $W_q = 0,42$ |
| | % времени, когда все серверы свободны $P_0 = 16,67\%$ |
| | Вероятность, что в системе ровно N клиентов: |
| | $P_{10} = 13,89\%$ |
| | $P_{11} = 11,57\%$ |
| | $P_{12} = 9,65\%$ |
| | $P_{13} = 8,04\%$ |
| | $P_{14} = 6,70\%$ |
| | $P_{15} = 5,58\%$ |
| | $P_{16} = 4,65\%$ |
| | $P_{17} = 3,88\%$ |
| | $P_{18} = 3,23\%$ |
| | $P_{19} = 2,69\%$ |
| | $P_{20} = 2,24\%$ |
| | $P_{21} = 1,87\%$ |
| | $P_{22} = 1,56\%$ |
| | $P_{23} = 1,30\%$ |
| | $P_{24} = 1,08\%$ |
| | $P_{25} = 0,90\%$ |
| | $P_{26} = 0,75\%$ |
| | $P_{27} = 0,63\%$ |
| | $P_{28} = 0,52\%$ |
| | $P_{29} = 0,43\%$ |
| | $P_{30} = 0,37\%$ |

Рис. 111. Решение задачи.

Ответы:

a. 0.83 b. 16.67 c. 5 d. 4.17 e. 0.5 f. 0.42 g. 0.46 (сумма вероятностей того, что в системе более 3 клиентов)

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1. (Выполняется по вариантам)

В супермаркете имеется M касс. Посетители занимают места в очереди в кассы. Как правило, очереди к различным кассам примерно одинаковые. Время прихода покупателей и время их обслуживания имеет экспоненциальную зависимость. Среднее время прихода равно T , а среднее время обслуживания равно Z .

| № варианта | M Число касс (штук) | T Время прихода покупателя (минута) | Z Время обслуживания покупателя (минута) |
|---------------|--------------------------|--|--|
| 1 | 3 | 20 | 10 |
| 2 | 6 | 25 | 15 |

Построить имитационную модель работы супермаркета и определить статистические характеристики системы согласно исходным данным, представленным в таблице. Найти значения характеристик:

- Процент загрузки каждой кассы
- Среднее число клиентов в системе
- Средняя длина очереди
- Среднее время пребывания в системе
- Среднее время ожидания в очереди
- % времени, когда все кассы свободны
- Вероятность, с которой возле кассы будут стоять более 3 клиентов

Задание 2.

Подберите такие параметры системы, чтобы процент простоя касс в задании 1 был минимальным.

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

Как решить задачи, в которых длина очереди ограничена или ограничена популяция клиентов, можно посмотреть в дополнительном материале.

<https://hcxl.net/m.html>

§3.10 Основные понятия теории игр

ВСПОМИНАЕМ

Вспомним:

- В каких случаях мы прибегаем к моделированию?

ЧИТАЕМ

Теория игр занимается математическим анализом **конфликтных ситуаций**, представленных в виде математической модели, определяющей некоторую **игру**. Первые математические аспекты и приложения теории игр были изложены в работе Джона фон Неймана и Оскара Моргенштерна «Теория игр и экономическое поведение» в 1944 г.

Моделями теории игр можно описывать разнообразные явления: экономические, военные, правовые конфликты, взаимодействие человека с природой, биологическую борьбу за существование и т. д. Например, в экономике взаимоотношения между поставщиком и потребителем, покупателем и продавцом, банком и клиентом носят конфликтный характер из-за различия целей партнеров и стремления каждого из них принимать решения, которые реализуют поставленные цели в наибольшей степени. При этом каждому приходится считаться не только со своими целями, но и с целями партнера, и учитывать неизвестные заранее решения, которые эти партнеры будут принимать.

Для любой **игры** должны быть известны:

- порядок чередования действий (или «ходов») участников;
- правила выполнения каждого хода;
- количественный результат игры (выигрыш, проигрыш), к которому приводит последовательность ходов.

Конфликтная ситуация — это ситуация, в которой сталкиваются интересы двух или более противодействующих сторон, преследующих различные цели. Стороны стремятся предпринять такие действия, которые приведут к наибольшему для себя успеху. Таким образом, если цели сторон противоположны (антагонистичны), то максимизация выигрыша одной из сторон будет означать максимизацию проигрыша другой стороны, что и порождает конфликтную ситуацию.

Конфликтующие стороны будут осуществлять поиск наилучших для себя решений. При этом каждой из сторон приходится принимать решение в условиях неопределенности поведения противодействующих сторон.

Теория игр позволяет:

- смоделировать процесс игры и ее возможные результаты до ее фактического начала;
- по результатам анализа смоделированной игры принять решение о целесообразности участия и оптимальном поведении в реальном конфликте.

Таким образом, теория игр дает математический прогноз конфликта.

Можно еще отметить, что **теория игр** — это теория математических моделей *принятия решений в условиях неопределенности*, когда принимающий решение субъект («игрок») располагает информацией лишь о множестве возможных ситуаций, в одной из которых он в действительности находится, о множестве решений («стратегий»), которые он может принять, и о количественной мере того выигрыша, который он мог бы получить, выбрав в данной ситуации данную стратегию.

Неопределенность имеет различное происхождение:

- Может являться *следствием сознательной деятельности других лиц*, отстаивающих свои интересы.
- Может быть *следствием появления случайности в игровой ситуации*: сознательные действия игроков, осуществляющих случайный выбор своих стратегий.
- Может быть *следствием действия так называемой «природы»*, характеризуемой обстоятельствами, не зависящими от субъектов игровой ситуации (игры с «природой»). К таким обстоятельствам можно отнести условия внешней среды (в которых принимаются решения): состояние погоды, рыночная конъюнктура, выход из строя техники и др. На основании какой-то, например, статистической информации, можно сделать **n** предположений о возможных условиях обстановки (состояний «природы»), которые трактуются как бы стратегиями «природы» — игрока 2.

Игры классифицируются по различным основаниям, полный перечень которых представлен на рисунке 112.

По количеству игроков

- парные игры (игры с двумя участниками)
- игры n игроков, где $n > 2$

По количеству стратегий

- игры с конечным числом стратегий
- игры с бесконечным числом стратегий

По степени информированности игроков о стратегиях, сделанных ходах и предпочтениях противника

- игры с полной информацией
- игры с неполной информацией

В зависимости от вида функций выигрыша

- матричные игры
- биматричные игры
- непрерывные игры
- выпуклые игры и др.

В зависимости от характера выигрышей

- игры с нулевой суммой (антагонистические игры)
- игры с ненулевой суммой, в которых целевые критерии для игроков различны

По возможности предварительных переговоров и взаимодействий между игроками в ходе игры

- коалиционные (корпоративные)
- бескоалиционные

Рис. 112. Классификация игр

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1

Используя дополнительные материалы и информационные ресурсы сети Интернет подготовьте сообщение по одной из классификаций игр, представленных на Рисунке 112.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Примеры парных игр

Пример 1. «Дилемма заключенного».

В совершении преступления подозреваются двое: А и Б. Есть основания полагать, что они действовали по сговору, и полиция, изолировав их друг от друга, предлагает им одну и ту же сделку: если один свидетельствует против другого, а тот хранит молчание, то первый освобождается за помощь

следствию, а второй получает максимальный срок лишения свободы (10 лет). Однако иных доказательств их вины у следствия нет. Если оба молчат, их деяние квалифицируется как неоказание помощи следствию, и они приговариваются к 6 месяцам. Если оба свидетельствуют друг против друга, они получают минимальный срок (по 3 года). Каждый подозреваемый выбирает, молчать ему или свидетельствовать против другого. Однако ни один из них не знает точно, что сделает другой.

Решение.

Будем рассматривать подозреваемых как игроков в данной игре: игрок А и игрок Б. Сформируем таблицу выигрышей игроков, выбрав в качестве их выигрышей величины, противоположные по знаку их возможным срокам заключения. Цель каждого из игроков — минимизация собственного срока заключения (т. е. максимизация выигрыша).

| Игрок А \ Игрок Б | Хранить молчание | Давать показания |
|-------------------|------------------|------------------|
| Хранить молчание | $(-0.5; -0.5)$ | $(-10; 0)$ |
| Давать показания | $(0; -10)$ | $(-3; -3)$ |

Рассуждения игроков: если партнер молчит, то лучше свидетельствовать против него (стратегия «давать показания») и выйти на свободу (иначе — полгода тюрьмы). Если же партнер дает показания (свидетельствует против него), то лучше тоже свидетельствовать против партнера (опять стратегия «давать показания»), чтобы получить 3 года (иначе — 10 лет).

Итак, стратегия «давать показания» строго доминирует над стратегией «хранить молчание», каждый игрок приходит к этому выводу. Таким образом, в условиях, когда каждый игрок оптимизирует свой собственный выигрыш, не заботясь о выгоде другого игрока, единственное возможное равновесие в игре — взаимное свидетельство обоих участников друг против друга — пара стратегий («давать показания», «давать показания»).

В то же время оптимальной ситуацией в данной игре является пара стратегий («хранить молчание», «хранить молчание»), для которой выигрыши игроков равны $-0,5$ (полгода заключения каждому). С точки зрения группы (этих двух подозреваемых) это наилучшее решение, при котором дальнейшее увеличение выигрыша одного из игроков (т. е. уменьшение его срока заключения) возможно только за счет уменьшения выигрыша другого (увеличение его срока заключения).

Суть дилеммы проявляется именно в том, что подозреваемые (как игроки), ведя себя по отдельности рационально (с позиции индивидуальной рациональности), вместе (как группа) приходят к нерациональному решению — к выбору стратегий, образующих ситуацию с худшим результатом.

Данный пример представляет собой бескоалиционную игру, причем игра парная — два игрока, биматричная, с ненулевой суммой (сумма

выигрышей игроков в каждой ситуации отлична от нуля). Выигрыши каждого игрока задаются соответствующими матрицами.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 2.

Рассмотрим игру, в которой участвуют два игрока. Игрок № 1 выбирает сторону монеты («орел» или «решка»), а игрок №2 пытается угадать, какая сторона выбрана. Если он не угадывает, то платит игроку № 1 10 денежных единиц, если угадывает — игрок № 1 платит ему 10 денежных единиц.

Составьте таблицу выигрышей игроков.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 3

Двое играют в следующую игру. Игрок 1 бросает случайным образом на горизонтальную плоскость игральный кубик, но игроку 2 не сообщает исход бросания. Игрок 2 пытается отгадать, выпало число очков не больше 3-х или больше. Если выпадает число очков больше 3 и игрок 2 угадывает это, то он получает от игрока 1 количество денежных единиц, равное выпавшему числу. Если выпадает число очков не больше 3 и игрок 2 угадывает это, то игроки ничего не платят друг другу. Если игрок 2 не отгадывает, то он платит игроку 1 в размере выпавшего числа.

Составьте таблицу выигрышей игроков.

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. https://elar.urfu.ru/bitstream/10995/43897/1/978-5-7996-1940-4_2016.pdf

§3.11 Инварианты и полуинварианты

ВСПОМИНАЕМ

Вспомним:

1. Чем занимается теория игр?
2. Что называется конфликтной ситуацией?

ЧИТАЕМ

Рассмотрим последовательность объектов или процесс, в котором позиции последовательно сменяются, например, игру.

Полуинвариант — это связанное с позицией число, которое при разрешенных действиях **все время растет или все время убывает** (возможно, нестрого).

Если полуинвариант меняется при каждой операции/ходе, он называется строгим, иначе нестрогим.

Оценив изменение полуинварианта на одном шаге или на группе шагов, можно оценить, за сколько шагов процесс закончится.

Типичные полуинварианты: сумма, произведение, модуль разности, сумма модулей, сумма квадратов и их комбинации (например, суммы).

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Например, при игре в крестики-нолики число заполненных клеток с каждым ходом увеличивается. Число заполненных клеток — это полуинвариант игры, и он растет с каждым ходом. На ограниченной доске из этого следует, что рано или поздно игра закончится. При игре на бесконечной доске игра может не закончиться никогда, но зато мы можем гарантировать, что позиция не повторится, ведь число заполненных клеток каждый раз новое.

ЧИТАЕМ

Если полуинвариант может принимать лишь конечное число значений или убывает, принимая лишь натуральные значения, то он достигнет «крайнего» значения. Это может обеспечить искомую позицию.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Условие:

В двух коробках лежат по 9 шариков. За один ход можно убрать из любой коробки 1 шарик или убрать 1 шарик из левой коробки и положить 9 шариков в правую.

- а) Докажите, что ходы рано или поздно закончатся.
- б) Какое наибольшее число ходов могло быть сделано?

Вариант решения:

а) Т.к. из левой коробки шарики только убираются (это полуинвариант этой игры), то максимум через 18 ходов она опустеет, а тогда придется брать из второй, и она тоже через конечное число ходов опустеет.

б) Наибольшее число ходов – 90. Вначале берем только из левой, а в правую добавляем 9 – 9 ходов. Получаем в итоге в левой – 0, а в правой – 81 шарик.

Убираем по одному из второй – 81 ход. Итого 90 ходов.

ЧИТАЕМ

Инвариантом называется величина или некоторое свойство, которое не меняется при заданных преобразованиях.

Нечисловые инварианты чаще всего связаны с чередованием или с невозможностью уничтожить элемент с каким-то свойством.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Условие:

На доске написаны шесть чисел: 1, 2, 3, 4, 5, 6. За один ход разрешается к любым двум из них одновременно добавлять по единице. Можно ли за несколько ходов все числа сделать равными?

Вариант решения:

Сумма написанных чисел нечетна, она равна 21. Это инвариантное свойство, т.к. за каждый ход эта сумма увеличивается на 2, т. е. всегда остается нечетной. А сумма шести равных чисел всегда четна. Это значит, что сделать числа равными невозможно.

Ответ. Нельзя.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Условие:

100 фишек выставлены в ряд. Разрешено менять местами две фишки, стоящие через одну. Можно ли с помощью таких операций переставить все фишки в обратном порядке?

Вариант решения:

Пронумеруем места, на которых стоят фишки, числами от 1 до 100. Заметим, что после выполнения данной в условии операции номер каждой фишки либо не изменился, либо изменился (увеличился или уменьшился) на 2. Таким образом, фишка, стоящая вначале на месте с четным номером, в любой момент остается стоять на месте с четным номером (это инвариантное свойство). Следовательно, фишка, стоящая на месте под номером 100, никогда не сможет попасть на клетку с номером 1.

Ответ. Нельзя.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Условие:

На доске написано число 12. В течение каждой минуты число либо умножают, либо делят на 2 или на 3, и результат записывают на доску вместо исходного числа. Докажите, что число, которое будет написано на доске ровно через час, не может быть равно 54.

Вариант решения:

После каждой операции меняется четность общего количества двоек и троек в разложении на простые множители числа на доске (это инвариантное свойство). Вначале это число нечетно (равно трем), т. к. $12=2*2*3$. Поэтому через 60 операций (секунд) оно должно быть нечетным, но в разложении $54=2*3*3*3$ четыре множителя. Противоречие показывает, что превращение из 12 в 54 невозможно.

ЧИТАЕМ

Типичная ситуация в игре: есть набор позиций (состояний) и переходы между ними. Это можно рассматривать как граф. Пусть с каждой позицией связана некоторая величина. Если величина при переходах не меняется, она — инвариант. Значения инварианта разбивают граф на компоненты связности, и нет маршрута между позициями с разными значениями инварианта. Соответственно, можно доказывать невозможность действия: например, нельзя доехать на поезде от Нью-Йорка до Москвы, поскольку поезда из Америки ходят только в Америку. Но можно доказать и существование: если добраться удалось, то был либо перелет, либо плавание.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1

В одной клетке квадратной таблицы со стороной 8 стоит знак минус, а в остальных стоят плюсы. Разрешается за один ход в некотором квадрате со стороной 2 изменять знаки на противоположные. Необходимо доказать, что с помощью таких ходов нельзя получить таблицу с одними плюсами.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Условие:

В банке лежат 1100 долларов. Разрешаются две операции: взять из банка 370 долларов или положить в него 111 долларов. Эти операции можно проводить много раз, при этом никаких денег, кроме тех, что первоначально лежат в банке, нет. Какую максимальную сумму можно извлечь из банка и как это сделать?

Пример решения:

Можно смоделировать задачу в Excel с помощью надстройки *Поиск решения*, в качестве целевой функции используем остаток на счете и минимизируем его (Рисунок 113).

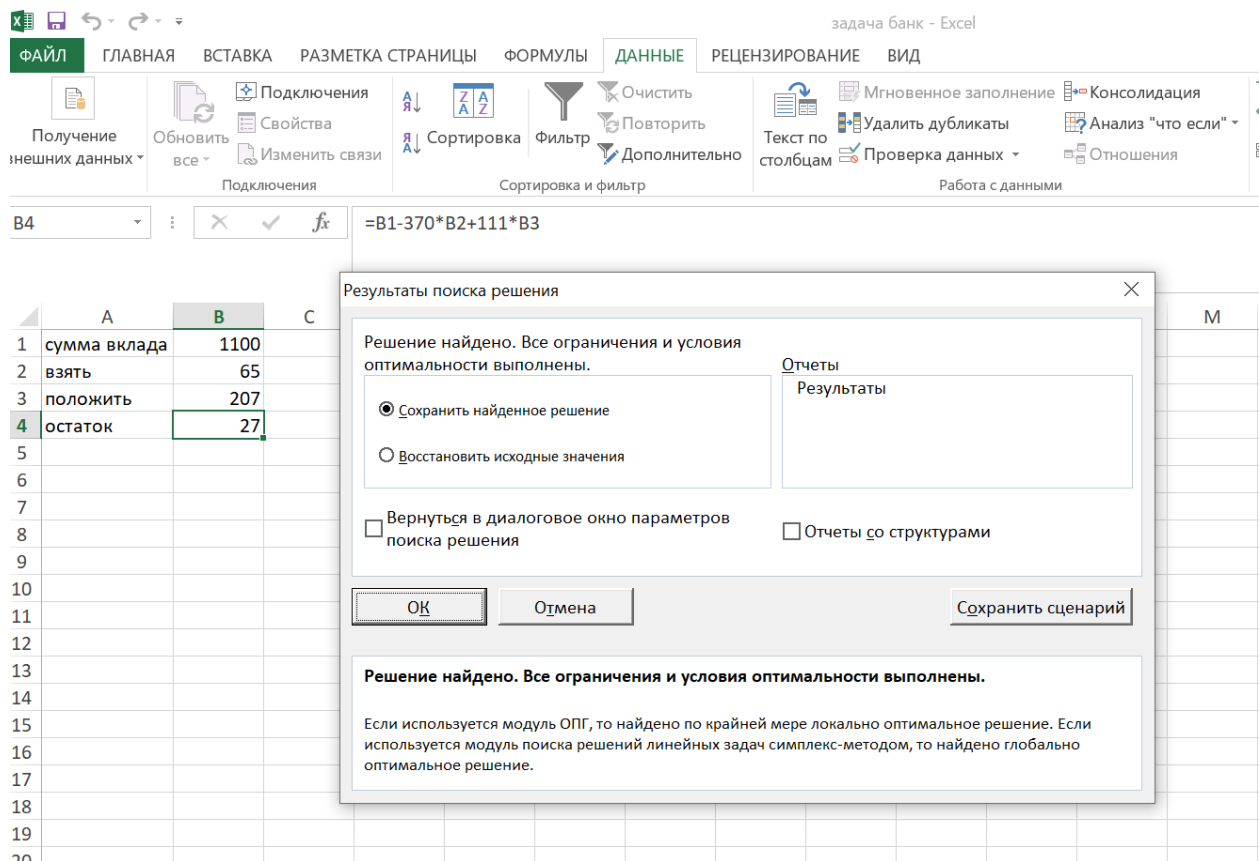


Рис. 113. Параметры Поиска решения.

Получим, что минимум равен 27, значит максимальная сумма – $1100 - 27 = 1073$.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 2

В банке есть 5670 долларов. Разрешаются две операции: взять из банка 490 долларов или положить в него 160 долларов. Эти операции можно проводить много раз, при этом никаких денег, кроме тех, что первоначально лежат в банке, нет. Можно ли извлечь всю сумму из банка и как это сделать?

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. http://school39.tgl.ru/sp/pic/File/SchoolSubjects/informatics/5_class/presentation/Urok_43-44_Viigrishnaya_strategiya_Viigrishnie_i_proigrishnie_pozitsii.pdf
2. <https://math.mosolymp.ru/upload/files/training/2015-04/Integral.linearized.pdf>
3. <https://foxford.ru/wiki/matematika/invariant>

§3.12 Выигрышные стратегии

ВСПОМИНАЕМ

Подумайте:

1. Могут ли в парной игре выиграть оба игрока?

ЧИТАЕМ

Если в игре оба игрока стремятся к победе, то такие партии игры можно называть разумными. Игроков, которые не делают ходы наугад и стараются победить, тоже считаем разумными.

В каждой игре с полной информацией, в которой нет ничьей, существует выигрышная стратегия для одного из игроков.

Выигрышная стратегия – это последовательность ходов, следуя которой один из игроков может выиграть при любых ходах противника.

Выигрышную стратегию может иметь только один из игроков.

Например, шахматы и шашки имеют выигрышную стратегию, но, чтобы ей воспользоваться, нужно перебрать все возможные партии игры. Для этого строится **дерево игры**, на первом уровне которого, например, для шахмат 20 вершин, на втором – уже 400!

В играх, которые допускают ничью, может существовать ничейная стратегия – последовательность ходов, позволяющая каждому из игроков свести партию игры к ничьей. Например, в игре «Крестики-нолики».

ОБСУЖДАЕМ

Как свести партию игры в игре «Крестики-нолики» к ничьей? Есть ли в этой игре выигрышная стратегия?

ЧИТАЕМ

Для поиска выигрышной стратегии необходимо рассмотреть все возможные позиции игры, построив **дерево игры**.

Позиция выигрышная для игрока, если из нее есть ход, который оставит противнику проигрышную позицию.

Позиция проигрышная для игрока, если любой ход из нее ставит противника в выигрышную позицию.

Выигрышную стратегию имеет тот игрок, который первый сможет занять выигрышную позицию. Выигрышная стратегия состоит в том, чтобы после каждого хода оставлять противнику проигрышную позицию.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Условие:

Пусть имеется куча из n камней. Играют два игрока. Каждый по очереди может взять один или два камня. Выигрывает тот, кто забирает последний камень. У кого из игроков есть выигрышная стратегия, если в куче 9 камней? 10 камней?

Возможные решения:

1. В куче 9 камней. 9 кратно 3. В этом случае выигрышную стратегию имеет второй игрок, т.к. он всегда может дополнить число камней, взятое первым, до 3-х. Позиция первого игрока – проигрышная.
2. В куче – 10 камней. Остаток от деления на 3 равен 1. Если первый игрок заберет это количество, то он поставит второго игрока в проигрышную позицию, а сам выиграет (см. п. 1).

РЕШАЕМ В ТЕТРАДИ

Задание 28.

В игре «Камешки» имеется 456 камней. Игроки по очереди могут взять 1, 2 или 3 камешка. Выигрывает тот, кто забирает последний камешек. У кого из игроков есть выигрышная стратегия, в чем она заключается?

ЧИТАЕМ

Методы теории игр находят применение в различных отраслях человеческой деятельности, в частности, международных отношениях, экономике и других общественных науках. Очень важное значение они имеют для искусственного интеллекта и кибернетики, особенно с появлением интереса к интеллектуальным агентам. Именно поэтому задачи на поиск выигрышной стратегии входят в типологию заданий единого государственного экзамена (ЕГЭ). В настоящий момент это задания № 19, 20, 21.

Для решения задачи рассматривается **дерево игры**, которое показывает все возможные варианты, начиная с некоторого начального положения.

В любой ситуации у игрока есть варианты хода, поэтому от каждого узла этого дерева отходят «ветки» по количеству вариантов хода.

Полный перебор вариантов можно выполнить только для простых игр.

Все позиции в простых играх делятся на **выигрышные** и **проигрышные**.

Общая стратегия игры состоит в том, чтобы своим ходом создать проигрышную позицию для соперника.

Игрок имеет выигрышную стратегию, если он может выиграть при любых ходах противника. Описать стратегию игрока – значит описать, какой ход он должен сделать в любой ситуации, которая ему может встретиться при

различной игре противника. В описание выигрышной стратегии не следует включать ходы играющего по этой стратегии игрока, не являющиеся для него безусловно выигрышными, т.е. не являющиеся выигрышными независимо от игры противника.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Условие:

Наташа и Марина играют в следующую игру. Перед ними лежит нитка с бусинами. Девочки по очереди (первой начинает Наташа) либо добавляют на нитку одну бусину, либо удваивают количество бусин на нитке. У девочек есть неограниченное количество бусин. Игра завершается в тот момент, когда количество бусин на нитке становится не менее 29. Победителем считается та, у которой первой на нитке окажется 29 или больше бусин. В начальный момент на нитке было S бусин, $1 \leq S \leq 28$. Укажите такое значение S , при котором Наташа не может выиграть за один ход, но при любом ходе Наташи Марина может выиграть своим первым ходом.

Возможное решение:

Таким значением будет 14, т.к. Наташа может либо прибавить одну бусину и получится нитка из 15 бусин, а значит, Марина, увеличив ее в два раза, получит 30 бусин и выиграет, либо Наташа удвоит и получит 28 бусин, тогда Марина выигрывает любым своим ходом. Позиция 14 – проигрышная для игрока.

Ответ: 14.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Условие:

Для игры, описанной в п. 1, найдите два таких значения S , при которых у Наташи есть выигрышная стратегия, причём одновременно выполняются два условия:

- Наташа не может выиграть за один ход;
- Наташа может выиграть своим вторым ходом независимо от того, как будет ходить Марина.

Найденные значения запишите в ответе в порядке возрастания.

Возможное решение:

Составим таблицу, в которой отметим выигрышные и проигрышные ходы: выигрышная позиция – зеленая, проигрышная – розовая.

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

Позиция 14 – проигрышная для игрока, который в нее попадает (см. п. 1). Если Наташа своим ходом получает это значение, то она выиграет. В 14 она

попадает из 7, удвоив количество бусин, или из 13 ходом +1. Значит, 7 и 13 – выигрышные для нее позиции

Ответ: 7 13.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Условие:

1. Для игры, описанной в п. 1, найдите значение S , при котором одновременно выполняются два условия:
 - у Марины есть выигрышная стратегия, позволяющая ей выиграть первым или вторым ходом при любой игре Наташи;
 - у Марины нет стратегии, которая позволит ей гарантированно выиграть первым ходом.

Если найдено несколько значений S , в ответе запишите минимальное из них.

Возможное решение:

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

Количество бусин должно быть таким, чтобы Наташа никаким своим ходом не попадала в выигрышную позицию, а, наоборот, попадала в проигрышную для себя. Таким числом является 12. Наташа может пойти в 13 и в 24, в первом случае Марина добавит 1 камень и выиграет вторым ходом, т.к. 14 – это проигрышная для игрока (Наташи) позиция. В случае хода в 24 бусины Марина удваивает и выигрывает первым ходом. Начальное число бусин не может быть меньше 12, т.к. Наташа ходом +1 не даст возможность Марине выиграть первым или вторым ходом.

Ответ: 12.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Условие:

Два игрока, Паша и Вася, играют в следующую игру. Перед игроками стоит стержень для пирамидки и кольца двух цветов: красные и белые. Игроки ходят по очереди, первый ход делает Паша. За один ход игрок может добавить на стержень (по своему выбору) одно кольцо любого цвета или увеличить количество колец одного цвета в два раза. Игра завершается в тот момент, когда суммарное количество колец двух цветов на пирамидке становится не менее 77. Победителем считается игрок, сделавший последний ход, т.е. первым получивший такую позицию, при которой на пирамидке будет 77 или больше колец. В начальный момент на пирамидке было 7 колец красного цвета и S колец белого цвета; $1 \leq S \leq 69$.

1. Известно, что Вася выиграл своим первым ходом после неудачного первого хода Паши. Укажите минимальное значение S , когда такая ситуация возможна.
2. Найдите два таких значения S , при которых у Паши есть выигрышная стратегия, причём одновременно выполняются два условия:
 - Паша не может выиграть за один ход;
 - Паша может выиграть своим вторым ходом независимо от того, как будет ходить Вася.

Найденные значения запишите в ответе в порядке возрастания.

3. Найдите минимальное значение S , при котором одновременно выполняются два условия:
 - у Васи есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Паши;
 - у Васи нет стратегии, которая позволит ему гарантированно выиграть первым ходом.

Возможное решение:

1. Очевидно, что для того, чтобы за два хода получить выигрышную сумму, исходя из минимального начального значения, необходимо удваивать количество белых колец на стержне. Вычислим: $77-7=70$, $70:2=35$ – необходимый минимум для выигрыша; $35:2=17.5$. Следовательно, минимальное значение – 18.
2. Обозначим кортежем (x, y) позицию, в которой на пирамидке x колец красного и y колец белого цветов. Кольца должно быть столько, чтобы Паша своим вторым ходом выиграл, но Вася своим не мог выиграть во всех случаях. Значит, максимальное значение y Васи должно быть на 1 меньше выигрыша и при этом, даже при ходе Васи «плюс один» Паша бы выиграл. Значит, значение S не может быть меньше 31. Иначе, даже если Паша сделает ход $(7*2, S)$, он не сможет выиграть в случае хода Васи $(14+1, S)$ (даже при $S=30$). Поэтому первое значение – это **$S=31$** . Ходы: $P(14, 31) \rightarrow B(15, 31) \rightarrow P(15, 62)$ – выигрыш. Остальные первые ходы Васи также ведут к выигрышу Пети (проверить!). Еще одно возможное значение: **$S=34$** . В самом деле, тогда $P(7+1, 34) \rightarrow B(8, 68) \rightarrow P(9, 68)$. (Можно проверить для всех возможных ходов Васи – значение будет меньше 77).

Ответ: $S = 31, 34$.

3. Из п. 2 следует, что для того, чтобы выиграть, Вася должен своим ходом отправить Пашу в проигрышные позиции $(8, 34)$ и $(14, 31)$. Это возможно из позиций $(7, 34)$, $(7, 17)$, $(14, 30)$ и $(7, 31)$ после первого хода Паши. Исключаем вариант $(7, 17)$ как невозможный после первого хода. Рассмотрим остальные: $(7, 34)$ возможен из начальной позиции $(7, 17)$, но остальные ходы из этой позиции не выигрышные.

В (7, 31) Паша может попасть из (7, 30). Рассмотрим эту начальную позицию и все ходы Паши:

- $P(7, 30) \rightarrow B(7, 31) \rightarrow P(14, 31)$ проигрышная позиция
- $P(7, 30) \rightarrow B(8, 30) \rightarrow P(16, 30)$ проигрыш Паши
- $P(7, 30) \rightarrow B(14, 30) \rightarrow P(14, 31)$ проигрышная позиция
- $P(7, 30) \rightarrow B(7, 60)$ выигрыш Васи первым ходом

Ответ: $S = 30$.

РЕШАЕМ В ТЕТРАДИ

Задание 29.

Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежат кольца для пирамидки. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может поставить на стержень одно или два кольца или увеличить количество колец в два раза. Например, имея пирамидку из 15 колец, за один ход можно получить пирамидку из 16, 18 или 30 колец. У каждого игрока есть неограниченное количество колец. Игра завершается в тот момент, когда высота пирамидки становится не менее 35 колец. Победителем считается игрок, сделавший последний ход, т.е. первым получивший пирамидку, в которой будет 35 или больше колец. В начальный момент пирамидка была высотой в S колец; $1 \leq S \leq 34$.

- а) Укажите все такие значения числа S , при которых Петя может выиграть в один ход.
б) Укажите такое значение S , при котором Петя не может выиграть за один ход, но при любом ходе Пети Ваня может выиграть своим первым ходом.
- Укажите два таких значения S , при которых у Пети есть выигрышная стратегия, причём одновременно выполняются два условия:
 - Петя не может выиграть за один ход;
 - Петя может выиграть своим вторым ходом независимо от того, как будет ходить Ваня.
- Укажите значение S , при котором одновременно выполняются два условия:
 - у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Пети;
 - у Вани нет стратегии, которая позволит ему гарантированно выиграть первым ходом.

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. <https://fipi.ru/>

2. <https://kpolyakov.spb.ru/>

§3.13 Компьютерное моделирование игр

ВСПОМИНАЕМ

Подумайте:

1. Что более вероятно: выиграть в лотерею, правильно угадав 5 из 36 номеров или 7 из 49?

ЧИТАЕМ

В некоторых ситуациях кажущееся очевидным решение таковым не является. Проиллюстрируем вышесказанное следующим парадоксом.

Парадокс Монти Холла

Правила игры: представьте, что перед вами три двери, как показано на рисунке ниже. За двумя дверями находятся козы, за одной — приз. Надо угадать дверь с призом, и он — ваш.

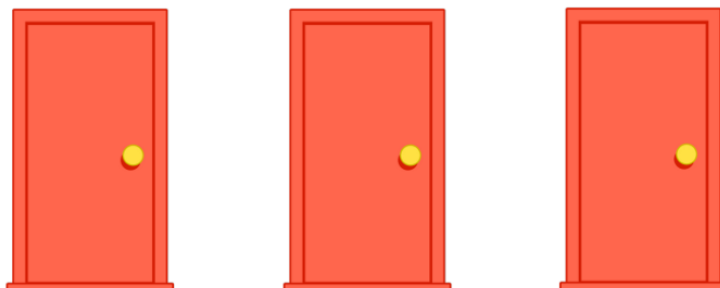


Рис. 114. Двери

После того, как участник выбирает дверь, ведущий открывает одну из дверей с козой и предлагает участнику поменять свой выбор. Менять ли выбор?

Рассуждая, что остались две двери, и приз может с одинаковой вероятностью $1/2$ оказаться как за одной, так и за другой дверью, поэтому выбор менять не нужно, шансы не выросли, участник совершает ошибку. Правильный ответ—всегда менять первоначальный выбор. Поступая так, участник удваивает свои шансы на победу. Почему это так?

В самом деле, при выборе одной из трёх дверей вероятность того, что приз окажется именно за ней, составляет $1/3$. А вероятность того, что он окажется за одной из двух оставшихся дверей, будет $2/3$.

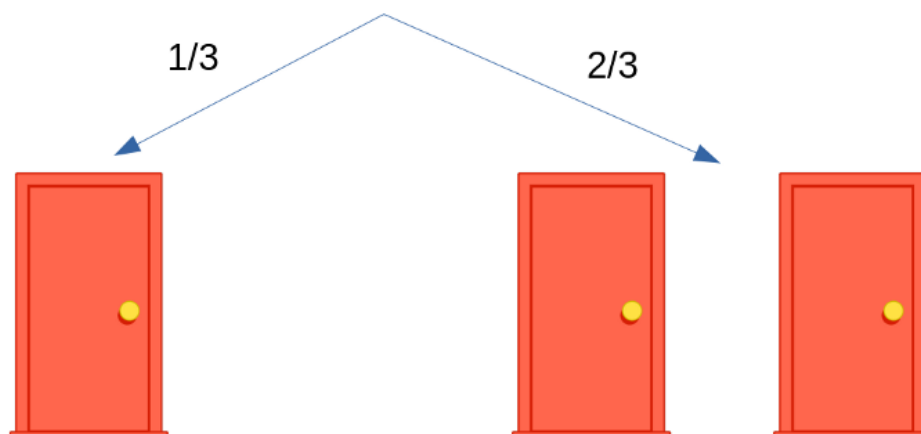


Рис. 115. Вероятности

Ведущий открывает одну из невыбранных дверей—тех, что справа. И открывает он всегда ту, за которой коза.

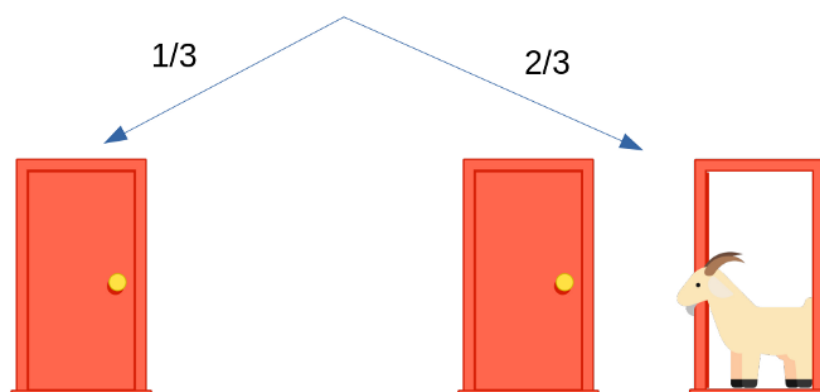


Рис. 116. Результат действий ведущего

Вероятности остаются неизменными: $1/3$ слева и $2/3$ справа. Изменилось лишь то, что справа одна дверь теперь открыта, но вероятность для оставшейся неоткрытой двери здесь та же, что была прежде для обеих, т.е. в два раза выше, чем у двери слева.

Однако важно не забывать, что всегда есть вероятность проигрыша. Верное решение определяется стратегией. Правильная стратегия—делать так, чтобы шансы на победу были максимальными или хотя бы такими, которые позволяют больше выигрывать, чем проигрывать.

РЕШАЕМ В ТЕТРАДИ

Задание 30.

Разберите парадокс Монти Холла для 100 дверей. Какова стратегия игрока должна быть, если открыть 99 дверей с козой, 1 дверь с козой?

ОБСУЖДАЕМ

Надо ли менять выбор в каждом из этих случаев?

ЧИТАЕМ

Проанализировав этот парадокс, можно сделать вывод, что правильно построенная математическая модель во многих ситуациях поможет принять верное решение. Если необходимо перебрать множество вариантов, человеку может помочь компьютерное моделирование. При решении задач на выигрышные стратегии из ЕГЭ также можно воспользоваться компьютерным моделированием.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №36

Рассмотрим **пример 3** из прошлого урока:

Два игрока, Паша и Вася, играют в следующую игру. Перед игроками стоит стержень для пирамидки и кольца двух цветов: красные и белые. Игроки ходят по очереди, первый ход делает Паша. За один ход игрок может добавить на стержень (по своему выбору) одно кольцо любого цвета или увеличить количество колец одного цвета в два раза. Игра завершается в тот момент, когда суммарное количество колец двух цветов на пирамидке становится не менее 77. Победителем считается игрок, сделавший последний ход, т.е. первым получивший такую позицию, при которой на пирамидке будет 77 или больше колец. В начальный момент на пирамидке было 7 колец красного цвета и S колец белого цвета; $1 \leq S \leq 69$.

1. Известно, что Вася выиграл своим первым ходом после неудачного первого хода Паши. Укажите минимальное значение S , когда такая ситуация возможна.

2. Найдите два таких значения S , при которых у Паши есть выигрышная стратегия, причём одновременно выполняются два условия:

- Паша не может выиграть за один ход;
- Паша может выиграть своим вторым ходом независимо от того, как будет ходить Вася.

Найденные значения запишите в ответе в порядке возрастания.

3. Найдите минимальное значение S , при котором одновременно выполняются два условия:

- у Васи есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Паши;
- у Васи нет стратегии, которая позволит ему гарантированно выиграть первым ходом.

Рассмотрим вариант **решения с помощью электронных таблиц:**

1. Построим математическую модель игры в электронных таблицах. Составим таблицу, в которой по вертикали отметим количество красных колец (начиная с 7), а по горизонтали – количество белых колец (начиная с 1). Для каждой пары вычислим наибольшее возможное число колец, которое может получить игрок за один ход. В ячейке B2 запишем формулу $=\text{МАКС}(\$A2*2+B\$1; \$A2+1+B\$1; \$A2+B\$1*2; \$A2+B\$1+1)$ и заполним ей всю нашу таблицу. Можно формулу оптимизировать (подумайте, как). С помощью условного форматирования пометим ячейки, в которых сумма больше 76 (условие выигрыша).

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------|----|----|----|----|----|----------------|----|--|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| AJ2 | | | | | | f _x | | =MAKC(\$A2*2+AJ\$1;\$A2+1+AJ\$1;\$A2+AJ\$1*2;\$A2+AJ\$1+1) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ▲ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | AA | AB | AC | AD | AE | AF | AG | AH | AI | AJ | AK | AL | AM | AN |
| 1 | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 2 | 7 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 23 | 25 | 27 | 29 | 31 | 33 | 35 | 37 | 39 | 41 | 43 | 45 | 47 | 49 | 51 | 53 | 55 | 57 | 59 | 61 | 63 | 65 | 67 | 69 | 71 | 73 | 75 | 77 | 79 | 81 | 83 | 85 |
| 3 | 8 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 26 | 28 | 30 | 32 | 34 | 36 | 38 | 40 | 42 | 44 | 46 | 48 | 50 | 52 | 54 | 56 | 58 | 60 | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 | 82 | 84 | 86 |
| 4 | 9 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 29 | 31 | 33 | 35 | 37 | 39 | 41 | 43 | 45 | 47 | 49 | 51 | 53 | 55 | 57 | 59 | 61 | 63 | 65 | 67 | 69 | 71 | 73 | 75 | 77 | 79 | 81 | 83 | 85 | 87 |
| 5 | 10 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 32 | 34 | 36 | 38 | 40 | 42 | 44 | 46 | 48 | 50 | 52 | 54 | 56 | 58 | 60 | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 | 82 | 84 | 86 | 88 |
| 6 | 11 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 35 | 37 | 39 | 41 | 43 | 45 | 47 | 49 | 51 | 53 | 55 | 57 | 59 | 61 | 63 | 65 | 67 | 69 | 71 | 73 | 75 | 77 | 79 | 81 | 83 | 85 | 87 | 89 |
| 7 | 12 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 38 | 40 | 42 | 44 | 46 | 48 | 50 | 52 | 54 | 56 | 58 | 60 | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 | 82 | 84 | 86 | 88 | 90 |
| 8 | 13 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 41 | 43 | 45 | 47 | 49 | 51 | 53 | 55 | 57 | 59 | 61 | 63 | 65 | 67 | 69 | 71 | 73 | 75 | 77 | 79 | 81 | 83 | 85 | 87 | 89 | 91 |
| 9 | 14 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 44 | 46 | 48 | 50 | 52 | 54 | 56 | 58 | 60 | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 | 82 | 84 | 86 | 88 | 90 | 92 |
| 10 | 15 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 47 | 49 | 51 | 53 | 55 | 57 | 59 | 61 | 63 | 65 | 67 | 69 | 71 | 73 | 75 | 77 | 79 | 81 | 83 | 85 | 87 | 89 | 91 | 93 |
| 11 | 16 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 50 | 52 | 54 | 56 | 58 | 60 | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 | 82 | 84 | 86 | 88 | 90 | 92 | 94 |
| 12 | 17 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 53 | 55 | 57 | 59 | 61 | 63 | 65 | 67 | 69 | 71 | 73 | 75 | 77 | 79 | 81 | 83 | 85 | 87 | 89 | 91 | 93 | 95 |
| 13 | 18 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 56 | 58 | 60 | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 | 82 | 84 | 86 | 88 | 90 | 92 | 94 | 96 |
| 14 | 19 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 59 | 61 | 63 | 65 | 67 | 69 | 71 | 73 | 75 | 77 | 79 | 81 | 83 | 85 | 87 | 89 | 91 | 93 | 95 | 97 |
| 15 | 20 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 | 82 | 84 | 86 | 88 | 90 | 92 | 94 | 96 | 98 |
| 16 | 21 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 65 | 67 | 69 | 71 | 73 | 75 | 77 | 79 | 81 | 83 | 85 | 87 | 89 | 91 | 93 | 95 | 97 | 99 |
| 17 | 22 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 | 82 | 84 | 86 | 88 | 90 | 92 | 94 | 96 | 98 | 100 |
| 18 | 23 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 71 | 73 | 75 | 77 | 79 | 81 | 83 | 85 | 87 | 89 | 91 | 93 | 95 | 97 | 99 | 101 |
| 19 | 24 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 74 | 76 | 78 | 80 | 82 | 84 | 86 | 88 | 90 | 92 | 94 | 96 | 98 | 100 | 102 |
| 20 | 25 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 77 | 79 | 81 | 83 | 85 | 87 | 89 | 91 | 93 | 95 | 97 | 99 | 101 | 103 |
| 21 | 26 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 80 | 82 | 84 | 86 | 88 | 90 | 92 | 94 | 96 | 98 | 100 | 102 | 104 |
| 22 | 27 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 83 | 85 | 87 | 89 | 91 | 93 | 95 | 97 | 99 | 101 | 103 | 105 |
| 23 | 28 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 86 | 88 | 90 | 92 | 94 | 96 | 98 | 100 | 102 | 104 | 106 |
| 24 | 29 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 89 | 91 | 93 | 95 | 97 | 99 | 101 | 103 | 105 | 107 |
| 25 | 30 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 92 | 94 | 96 | 98 | 100 | 102 | 104 | 106 | 108 |
| 26 | 31 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 95 | 97 | 99 | 101 | 103 | 105 | 107 | 109 |
| 27 | 32 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 98 | 100 | 102 | 104 | 106 | 108 | 110 |
| 28 | 33 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 101 | 103 | 105 | 107 | 109 | 111 |
| 29 | 34 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 104 | 106 | 108 | 110 | 112 |
| 30 | 35 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 107 | 109 | 111 | 113 |
| 31 | 36 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 110 | 112 | 114 |
| 32 | 37 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 113 | 115 |
| 33 | 38 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 116 |
| 34 | 39 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 117 | |
| Лист1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Рис. 117. Решение в ЭТ.

Выигрышная позиция первым ходом – это координаты всех розовых ячеек.

По таблице получаем, что Вася выигрывает первым ходом из позиций, возможных после первого хода Паши: $(7, 18) \rightarrow (7, 36)$, $(7, 35) \rightarrow (8, 35)$, $(7, 32) \rightarrow (14, 32)$. Минимальное подходящее значение S равно 18.

ОТВЕТ: 18.

2. Найдём проигрышные значения (при любой игре побеждает следующий игрок). Рассмотрим комбинации, из которых каждый

возможный ход попадает в выигрышные позиции. Это позиции, имеющие сумму 76. Отметим их зеленым цветом.

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | AA | AB | AC | AD | AE | AF | AG | AH | AI | AJ | AK | AL | AM | AN | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1 | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | 7 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 23 | 25 | 27 | 29 | 31 | 33 | 35 | 37 | 39 | 41 | 43 | 45 | 47 | 49 | 51 | 53 | 55 | 57 | 59 | 61 | 63 | 65 | 67 | 69 | 71 | 73 | 75 | 77 | 79 | 81 | 83 | 85 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | 8 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 26 | 28 | 30 | 32 | 34 | 36 | 38 | 40 | 42 | 44 | 46 | 48 | 50 | 52 | 54 | 56 | 58 | 60 | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 | 82 | 84 | 86 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | 9 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 29 | 31 | 33 | 35 | 37 | 39 | 41 | 43 | 45 | 47 | 49 | 51 | 53 | 55 | 57 | 59 | 61 | 63 | 65 | 67 | 69 | 71 | 73 | 75 | 77 | 79 | 81 | 83 | 85 | 87 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | 10 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 32 | 34 | 36 | 38 | 40 | 42 | 44 | 46 | 48 | 50 | 52 | 54 | 56 | 58 | 60 | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 | 82 | 84 | 86 | 88 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | 11 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 35 | 37 | 39 | 41 | 43 | 45 | 47 | 49 | 51 | 53 | 55 | 57 | 59 | 61 | 63 | 65 | 67 | 69 | 71 | 73 | 75 | 77 | 79 | 81 | 83 | 85 | 87 | 89 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | 12 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 38 | 40 | 42 | 44 | 46 | 48 | 50 | 52 | 54 | 56 | 58 | 60 | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 | 82 | 84 | 86 | 88 | 90 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | 13 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 41 | 43 | 45 | 47 | 49 | 51 | 53 | 55 | 57 | 59 | 61 | 63 | 65 | 67 | 69 | 71 | 73 | 75 | 77 | 79 | 81 | 83 | 85 | 87 | 89 | 91 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | 14 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 44 | 46 | 48 | 50 | 52 | 54 | 56 | 58 | 60 | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 | 82 | 84 | 86 | 88 | 90 | 92 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | 15 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 47 | 49 | 51 | 53 | 55 | 57 | 59 | 61 | 63 | 65 | 67 | 69 | 71 | 73 | 75 | 77 | 79 | 81 | 83 | 85 | 87 | 89 | 91 | 93 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | 16 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 | 82 | 84 | 86 | 88 | 90 | 92 | 94 | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | 17 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 63 | 65 | 67 | 69 | 71 | 73 | 75 | 77 | 79 | 81 | 83 | 85 | 87 | 89 | 91 | 93 | 95 | | | | | | | | | | | | | | | | | | | | | | | |
| 13 | 18 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 | 82 | 84 | 86 | 88 | 90 | 92 | 94 | 96 | | | | | | | | | | | | | | | | | | | | | | | | |
| 14 | 19 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 65 | 67 | 69 | 71 | 73 | 75 | 77 | 79 | 81 | 83 | 85 | 87 | 89 | 91 | 93 | 95 | 97 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 15 | 20 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 | 82 | 84 | 86 | 88 | 90 | 92 | 94 | 96 | 98 | 100 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16 | 21 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 | 82 | 84 | 86 | 88 | 90 | 92 | 94 | 96 | 98 | 100 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 17 | 22 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | | | | | | | | | | | | | | | | | | | | | |
| 18 | 23 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | | | | | | | | | | | | | |
| 19 | 24 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | | | | | | | | | | | | | | | |
| 20 | 25 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 |
| 21 | 26 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | | |
| 22 | 27 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | | | | |
| 23 | 28 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | | | | | | |
| 24 | 29 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | | | | | | | | |
| 25 | 30 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | | | | | | | | | | |
| 26 | 31 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | | | | | | | | | | | | |
| 27 | 32 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | | | | | | | | | | | | | | |
| 28 | 33 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | | | | | | | | | | | | | | | | |
| 29 | 34 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | | | | | | | | | | | | | | | | | | |
| 30 | 35 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | | | | | | | | | | | | | | | | | | | | |
| 31 | 36 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | | | | | | | | | | | | | | | | | | | | | | |
| 32 | 37 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | | | | | | | | | | | | | | | | | | | | | | | | |
| 33 | 38 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 34 | 39 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Рассмотрим начальные комбинации, ходы из которых попадают в выигрышные позиции. По таблице видно, что это (7;33) и (7;30). Оба – подходят, выбираем меньшее – 30.

Ответ: 30.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Условие:

Игроки Паша и Вася в следующую игру. Перед игроками стоит стержень для пирамидки и кольца двух цветов: красные и синие. Игроки ходят по очереди, первый ход делает Паша. За один ход игрок может добавить на стержень (по своему выбору) три кольца одного цвета или увеличить количество колец одного цвета в два раза. Игра завершается в тот момент, когда суммарное количество колец двух цветов на пирамидке становится не менее 62. Победителем считается игрок, сделавший последний ход, т.е. первым получивший такую позицию, при которой на пирамидке будет 62 или больше колец. В начальный момент было 7 колец красного цвета, S колец синего цвета; $1 \leq S \leq 54$.

Задание 1

Известно, что Вася выиграл своим первым ходом после неудачного первого хода Паши. Укажите минимальное значение S , когда такая ситуация возможна.

Задание 2

Найдите минимальное значение S , при котором у Паши есть выигрышная стратегия, причём одновременно выполняются два условия:

- Паша не может выиграть за один ход;
- Паша может выиграть своим вторым ходом независимо от того, как будет ходить Вася.

Задание 3

Найдите два значения S , при которых одновременно выполняются два условия:

- у Васи есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Паши;
- у Васи нет стратегии, которая позволит ему гарантированно выиграть первым ходом.

Найденные значения запишите в ответе в порядке возрастания.

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. <https://medium.com/nuances-of-programming/%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%B5%D0%B9%D1>

[%88%D0%B5%D0%B5-%D0%BE%D0%B1%D1%8A%D1%8F%D1%81%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5-%D0%BF%D0%B0%D1%80%D0%B0%D0%B4%D0%BE%D0%BA%D1%81%D0%B0-%D0%BC%D0%BE%D0%BD%D1%82%D0%B8-%D1%85%D0%BE%D0%BB%D0%BB%D0%B0-3732389ed76d](#)

2. <https://kpolyakov.spb.ru/school/ege.htm>

ИТОГОВОЕ ЗАДАНИЕ ПО ТЕМАМ: ГРАФИКА, ФАЙЛЫ И МОДЕЛИРОВАНИЕ

Мини-проект

Реализовать одну из математических моделей из задачника на моделирование физических, биологических или экономических процессов, используя библиотеки языка Python. Построить графики для различных начальных условий, проанализировать и сделать выводы.

IV. Базы данных

§4.1 Большие данные. Понятие и классификация СУБД

ВСПОМИНАЕМ

Вспомним:

1. Что такое данные? Чем отличаются понятия «данные» и «знания»?

ОБСУЖДАЕМ

Обсудите, что вы знаете о больших данных. Какие данные мы называем большими?

ЧИТАЕМ

Это данные из реального мира, совершенно разнообразные, разноформатные, которые необходимо обрабатывать, хранить, передавать. В эпоху быстро развивающихся информационных технологий объем поступающих данных растет с невероятной скоростью.

Примерами больших данных являются:

- Твиты, хранящиеся на серверах Twitter
- Информация, которую Яндекс получает от отслеживания передвижения автомобилей
- Данные медицинских страховых компаний о том, кто, какое и в какой больнице получает лечение
- Информация, которую собирают банки о людях и их финансовых действиях, так называемая клиентская база
- Открытые данные (например, постоянно передаваемые данные об изменениях земли со спутников)

Конечно же большие данные не могут храниться и обрабатываться на простых жестких дисках традиционными способами. Конечно, существуют специальные инструменты, такие как, например, Hadoop, Spark, NoSQL, Data Discovery. Технология обработки больших данных получила название **big data**, она включила в себя специализированные подходы и методы обработки как структурированных, так и неструктурированных данных для получения новой информации, использующейся в конечном итоге для конкретных потребностей и целей информационного общества. В деловом мире технология big data стала способна решать актуальные бизнес-задачи, которые раньше казались слишком сложными. Примеры можно изучить здесь: <https://www.oracle.com/ru/big-data/what-is-big-data/>.

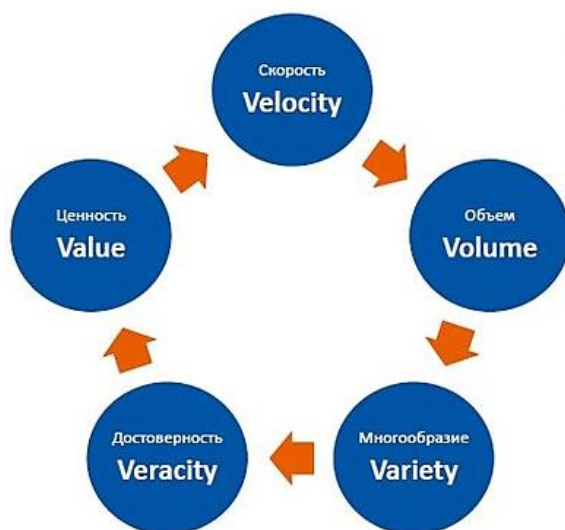
Большие данные – очень востребованная область. Сегодня применение Big Data находится в разных сферах:



Рис. 120. Области применения Big Data

Выделяют пять основных характеристик больших данных:

1. объём (volume) – величина физического объёма данных. Большие данные измеряются в десятках терабайт;
2. скорость (velocity) – скорость постоянного прироста данных, а также необходимости высокоскоростной обработки и получения результатов на их основе;
3. многообразие (variety) – возможность одновременной обработки различных типов структурированных и полуструктурированных данных.
4. достоверность (veracity) – данные должны быть репрезентативны и непротиворечивы;
5. ценность (value) – данные должны обладать полезностью или потенциальной ценностью (многие данные, собираемые сегодня, не всегда подлежат обработке, большинство просто хранится до нужного времени).



| Характеристика | Традиционная база данных | База Больших Данных |
|------------------------------------|--------------------------|---|
| Объём информации | От гигабайт до терабайт | От петабайт до эксабайт |
| Способ хранения | Централизованный | Децентрализованный |
| Структурированность данных | Структурирована | Полуструктурирована или неструктурирована |
| Модель хранения и обработки данных | Вертикальная модель | Горизонтальная модель |
| Взаимосвязь данных | Сильная | Слабая |

Рис. 121. Основные характеристики Big Data

Заметим, что на практике довольно часто приходится иметь дело с небольшими наборами данных, которые также могут быть результатами применения технологии big data. Наборы данных могут быть различным образом структурированы, например, разбиты на разделы и табличные пространства. Наборы данных могут иметь распределенный тип хранения, то есть составные части размещаются в различных узлах компьютерной сети в соответствии с каким-либо критерием. Такой структурированный набор данных, хранящийся в памяти вычислительной системы, называется базой данных (БД). Описанную каким-либо образом логическую структуру базы данных называют моделью организации данных (моделью данных). К основным моделям данных относятся: иерархическая, сетевая, реляционная.

Совершенно понятно, что модель представления данных должна быть создана, должна поддерживаться и быть использована для эффективной работы с данными. Все эти функции выполняет **Система управления базами данных (СУБД)**. Данная система содержит комплекс языковых и программных средств, предназначенный для создания, ведения и совместного использования БД многими пользователями.

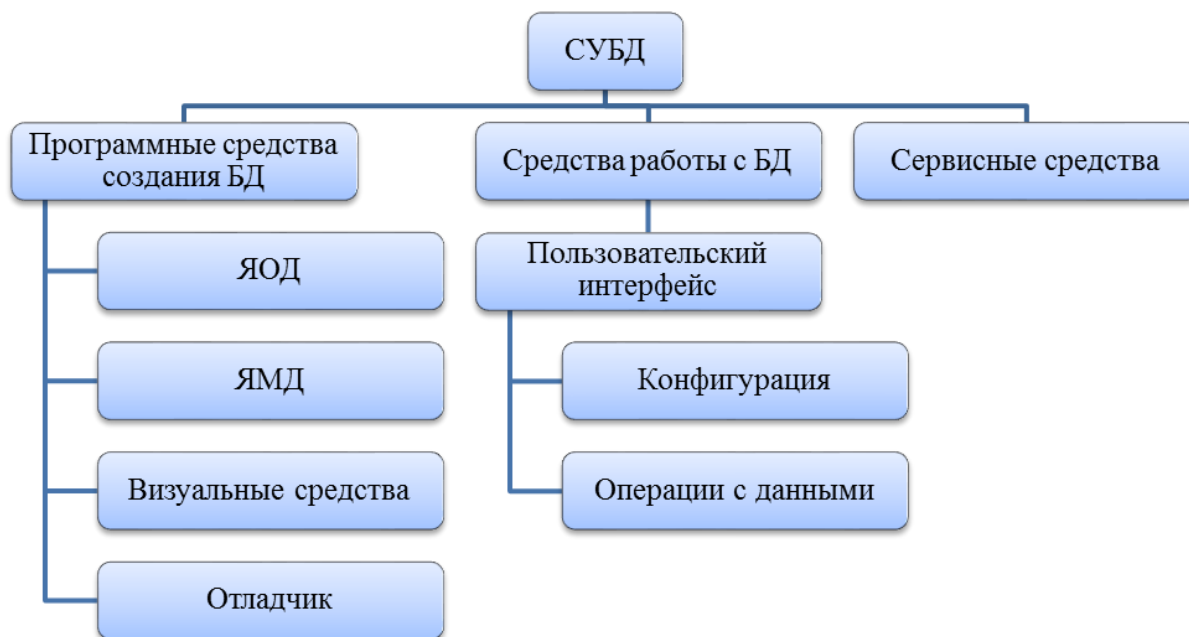


Рис. 122. Структура СУБД

Обратите внимание на аббревиатуры ЯОД и ЯМД. ЯОД расшифровывается как язык описания данных, а ЯМД – как язык манипулирования данными (язык определения схемы БД).

В основе классификации СУБД могут лежать: модель данных; степень распределенности; способу доступа к БД; степени универсальности.



Рис. 4. Классификация СУБД

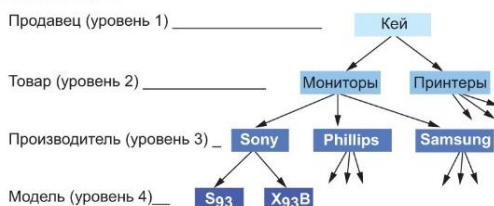
Например, СУБД, основанные на табличном представлении данных, называются реляционными СУБД.

РЕШАЕМ В ТЕТРАДИ

Задание 31.

Установите соответствие между видами СУБД (Реляционные, Распределённые, Сетевые, Локальные, Иерархические) и примерами А-Д.

Прайс-лист



А

Учащийся (номер в списке, фамилия, группа)



Реферат (шифр, руководитель, предмет)

Б



В

| Сеансы | | | |
|-----------|-----------------------|-------|-----------|
| Кинотеатр | Фильм | Время | Стоимость |
| Беларусь | Лови момент | 15:30 | 10,00 р. |
| Пионер | Зеленая книга | 19:15 | 9,50 р. |
| Мир | Как приручить дракона | 16:00 | 10,00 р. |
| Аврора | Капитан Марвел | 16:20 | 9,00 р. |
| Мир | Рифмуется с любовью | 14:00 | 6,00 р. |

Г



Д

РЕШАЕМ В ТЕТРАДИ

Задание 32.

Ответьте на вопросы теста:

1) Что такое база данных?

- а) текстовый файл б) простая таблица в) организованная структура хранения информации г) электронная таблица для хранения информации

2) Какую функцию не выполняет СУБД?

- а) создание структуры БД для хранения информации б) проверка корректности прикладных программ, работающих с БД в) выполнение операций над данными в БД г) загрузка данных в БД

3) Что побуждает пользователя к использованию СУБД?

- а) создание структуры БД для хранения информации б) получение новой информации как результата технологии big data в) большой объем математических вычислений г) эффективная работа со структурированной информацией для решения разных задач

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Создайте однотабличную базу данных с 10 или более записями, используя редактор электронных таблиц. Используйте следующие поля:

| № | Название фильма | Год выхода | Жанр | Длительность | Рейтинг |
|---|-----------------|------------|------|--------------|---------|
| 1 | | | | | |

Информацию о фильмах можно найти в сети Интернет, например, здесь:

<https://www.kinopoisk.ru/>

Выполните следующие операции по управлению данными:

- 1) Отсортируйте данные по возрастанию рейтинга фильма.
- 2) Отсортируйте данные так, чтобы названия были по алфавиту, а годы выхода по убыванию.
- 3) Выберите фильмы, которые относятся к одному жанру, содержащему их в максимальном количестве.

§4.2 Виды хранилищ. Основные понятия реляционных БД

ВСПОМИНАЕМ

Вспомним:

1. Какие типы баз данных Вам известны?
2. Что называют «полем» и что «записью» в реляционной базе данных?

ЧИТАЕМ

Современные бизнес-системы управляют большими объемами разнородных данных. Это приводит к тому, что единственное хранилище данных не является оптимальным решением в информационной системе. В настоящее время становится предпочтительнее применять разные хранилища для различных типов данных, каждое из которых направляют на определенную рабочую функциональность или определенный шаблон использования. При этом стараются сочетать разные технологии реализации хранилища данных.

Хранилища данных обычно отличаются друг от друга методами структурирования данных и типами поддерживаемых операций. Заметим, что хранилища данных одной категории не обязательно предоставляют один и тот же набор возможностей. Какие-то хранилища, и их большинство, выполняют обработку запросов и данных на стороне сервера, какие-то имеют отдельный модуль или несколько модулей для обработки и анализа данных. Обычно хранилища данных поддерживают разные интерфейсы программного доступа и управления. Подробнее о хранилищах вы можете прочитать в дополнительных материалах.

Рассмотрим реляционную СУБД. Данная информационная система поддерживает технологии хранилища пар отношений "ключ — значение". Познакомимся с терминами реляционной модели:

| Термин реляционной модели | Определение |
|---------------------------|--|
| Отношение | Табличная связь |
| Схема отношения | Строка заголовков столбцов таблицы (шапка таблицы или заголовок таблицы) |
| Кортеж | Строка таблицы (запись) |
| Сущность | Описание свойств объекта |
| Атрибут | Столбец таблицы(поле) |
| Домен | Множество допустимых значений атрибута |
| Первичный ключ | Уникальный идентификатор |
| Кардинальность | Количество строк |
| Степень | Количество столбцов |

Реляционная база данных представляет собой хранилище данных, содержащее набор двухмерных таблиц. Данные в таблицах должны удовлетворять следующим принципам:

- Значения атрибутов должны быть атомарными. То есть каждое значение, содержащееся на пересечении строки и колонки, должно не допускать деления на несколько значений
- Значения каждого атрибута должны принадлежать к одному и тому же типу
- Каждая запись в таблице уникальна
- Каждое поле имеет уникальное имя
- Последовательность полей и записей в таблице не существенна

Сущность – это множество объектов реального мира с одинаковыми свойствами. В качестве сущности может выступать личность, место, продукт и т. д., информацию о которых необходимо хранить в БД. Данные о сущности хранятся в отношении (в таблице). Атрибуты представляют собой свойства, характеризующие сущность. В структуре таблицы каждый атрибут именуется и ему соответствует заголовок некоторого столбца таблицы. Ключом отношения называется совокупность его атрибутов, однозначно идентифицирующих каждый кортеж. Другими словами, множество атрибутов K , являющееся ключом отношения, обладает свойством уникальности. Еще одним очень важным свойством ключа является его избыточность. Что означает: никакое из подмножеств множества K не обладает свойством уникальности.

Ключи обычно используют для достижения следующих целей:

- исключения дублирования значений в ключевых атрибутах (остальные атрибуты в расчет не принимаются);
- упорядочения кортежей. Возможно упорядочение по возрастанию или убыванию значений всех ключевых атрибутов, а также смешанное упорядочение (по одним — возрастание, а по другим — убывание);
- организации связывания таблиц.

Очень важным является понятие внешнего ключа. Внешний ключ – это множество атрибутов первого отношения (таблицы R_1), значения которых должны совпадать со значениями возможного ключа второго отношения (второй таблицы R_2). Именно с помощью внешних ключей устанавливаются связи между отношениями.

К отношениям можно применять систему операций (запрос), позволяющую получать одни отношения из других. Например, результатом запроса к реляционной БД может быть новое отношение (новая таблица), вычисленное на основе имеющихся отношений. Поэтому, как правило, можно разделить все обрабатываемые данные на хранимую и вычисляемую части.

Итак, основной единицей обработки данных в реляционных БД является отношение, а связи между отношениями позволяют строить запросы, результатами которых могут быть новые отношения.

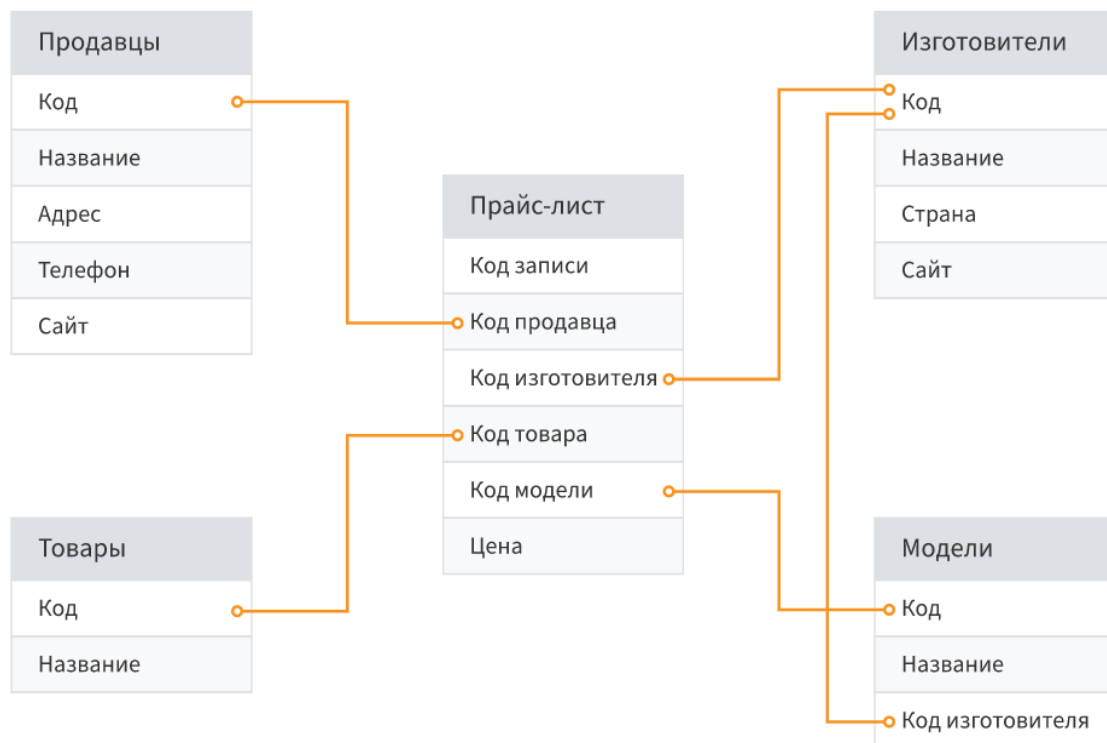


Рис. 124. Пример реляционной базы данных

ГОТОВИМСЯ К РАБОТЕ НА КОМПЬЮТЕРЕ

Для работы с базой данной (БД) удобно использовать менеджер БД, с помощью которого можно управлять базой в визуальном режиме. Для СУБД SQLite самый удобный менеджер — это SQLiteStudio. Для установки можно скачать дистрибутив, перейдя по ссылке: <https://sqlitestudio.pl/>

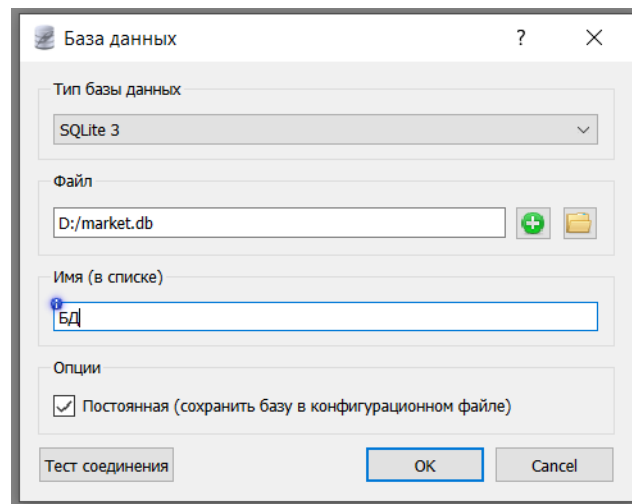
Вот основные функции SQLite:

Что делать?

Создание новой базы данных market

Как делать?

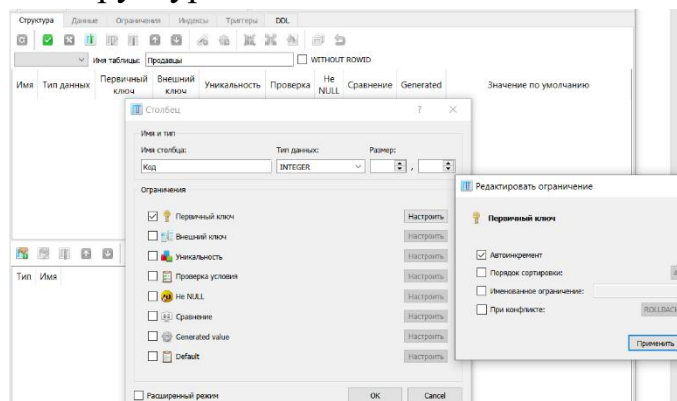
Для создания новой БД SQLite войдите в меню «Database — Add a database» Введите имя файла market и имя базы данных Магазин для отображения в списке



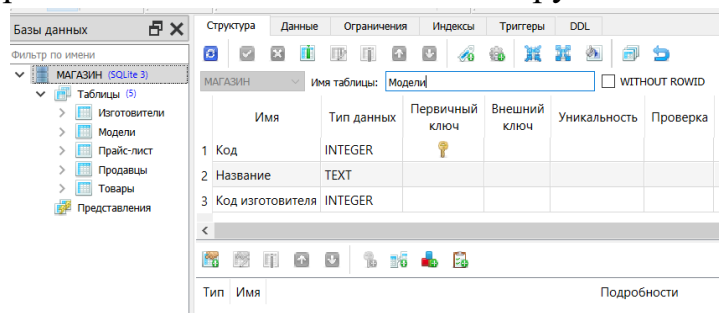
Создание таблиц

Войдите в меню «Structure – Create a table». Появится окно структуры.

Введите имя таблицы, например «Продавцы». Теперь нужно добавить поле. Нажмите «Добавить столбец (Insert)» и также добавьте другие поля. В конце обязательно нажмите зеленую галочку «Подтвердить изменения структуры».

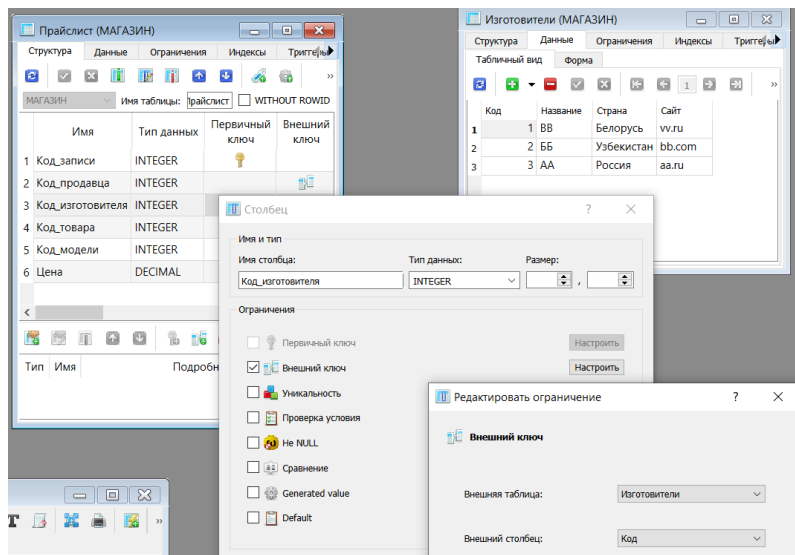


Повторите эти действия для создания других таблиц.



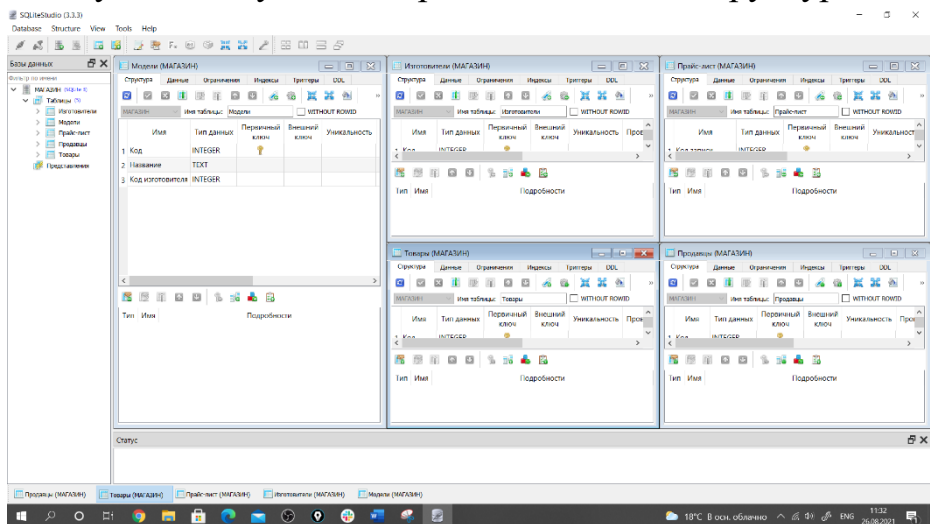
Создание связей

Установите внешние ключи «Структура», выбор поля



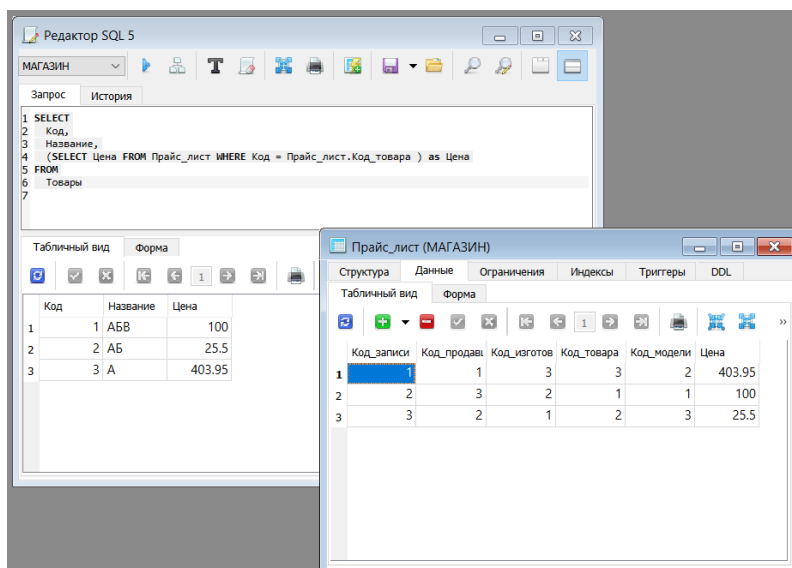
Добавление
данных

Нажмите на вкладку «Данные». Нажмите клавишу «Вставить строку» (зеленый плюс или Insert) и введите данные. Сначала данные вводятся в таблицы без внешних ключей. Порядок ввода данных: Товары, Продавцы, Изготовители, Модели, Прайс-лист. В конце нажмите зеленую галочку «Подтвердить изменения структуры»



Создание запроса

Получите информацию сразу из двух таблиц с помощью запроса



Таким образом с помощью менеджера БД можно быстро создать базу данных, заполнить ее данными и отладить SQL-запросы, которые потом можно вставить в программу.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №37

Условие:

В школе № 3333 проходили предметные олимпиады. В них успешно выступили ученики 9А, 9Б, 10А и 10Б классов. Классный руководитель 9А класса – учитель физики Иванова Н. А. Классный руководитель 9Б класса – учитель математики Петрова А. Н. Классный руководитель 10А класса – учитель химии Сидорова Р. В. Классный руководитель 10Б класса – учитель математики Шмелева Т. Л. В соревновании по истории медаль завоевал ученик 9А класса Захар Васильев; грамоту получил ученик 9А класса Иван Тюкин; почетный приз – ученица 10Б класса Мария Зубкова. В соревновании по математике медаль завоевала ученица 9А класса Елена Котова; грамоту получила ученица 10А класса Арина Репина; почетный приз – ученица 10А класса Анастасия Попова. В соревновании по физике медали получили ученик 9Б класса Алексей Ильин и ученица 10Б класса Грачева Екатерина. В соревновании по химии медаль получил ученик 9А класса Антон Кирилов; приз получил ученик 9А класса Влад Болотов. Возраст победителей: Васильев, Ильин и Кирилов – 15 лет; Болотов и Котова – 16 лет; Грачева – 18 лет; остальным ребятам – по 17 лет. По итогам олимпиады за успехи своих воспитанников дипломами были награждены учителя Иванова, Сидорова и Шмелева.

Задание:

Требуется выполнить следующие действия:

1. Создать базу данных с информацией о результатах олимпиады.

2. Обратиться к БД со следующими запросами:

- Получить список всех ребят, награжденных медалями. В списке указать: фамилию, имя, класс, предмет. Упорядочить список в алфавитном порядке по фамилиям.
- Получить список всех награжденных медалями, классные руководители которых получили дипломы.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Как используются хранилища данных?
2. Какими свойствами обладает реляционная база данных?
3. Что называют записью и полем в таблице базы данных?
4. Назначение SQLiteStudio? Какие возможности предоставляет пользователю?
5. Как создать новую базу данных?
6. Какое расширение имеют файлы баз данных?
7. Какие объекты являются составляющими баз данных?

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. <https://habr.com/ru/post/441538/>
2. <https://www.topobzor.com/obzor-10-oblachnyx-xranilishh-dannyx/.html>
3. https://www.youtube.com/watch?v=29rx5SyXHvE_ol/print/index.php?id=3434
4. <https://www.youtube.com/watch?v=29rx5SyXHvE>

§4.3 Проектирование многотабличных БД

ВСПОМИНАЕМ

Вспомним:

1. Что такое «первичный ключ»?
2. Что называют кардинальностью реляционной базы данных?

ЧИТАЕМ

Многотабличная база данных является реляционной. Понятие «реляционный» означает «основанный на отношениях». Реляционная база данных состоит из сущностей (таблиц), находящихся в некотором отношении друг с другом (название произошло от английского слова relation—отношение).

Создание многотабличных БД начинается с проектирования. Что мы понимаем под проектированием? **Проектирование базы данных** — это деятельность, направленная на получение ее прототипа, который удовлетворял бы всем предъявляемым функциональным требованиям, учитывал возможности доступных технологий при заданных ограничениях информационной системы. Таким образом, **спроектировать** базу данных — значит детально ее описать, затем формализовать описание для уточнения и оптимизации ее структуры. Сам процесс проектирования можно разбить на несколько этапов:

1. сбор информации;
2. определение сущностей;
3. определение атрибутов для каждой сущности;
4. определение связей между сущностями;
5. нормализация;
6. преобразование к физической модели;
7. создание прототипа базы данных.

Первые пять этапов образуют часть деятельности, которую относят к логическому проектированию, а остальные два — относятся к физическому проектированию. Проектирование начинается с момента замысла концепции базы данных и может еще продолжаться далее на этапах ее реализации и тестирования.

Процесс проектирования базы данных должен охватывать несколько важных сфер:

- представление данных предметной области БД
- интерфейс взаимодействия с БД (формы, отчеты и т. д.)
- архитектура, учитывающая особенности конкретной вычислительной среды ("клиент-сервер", параллельные архитектуры, распределенная вычислительная среда и т. д.)

Очень важно учитывать, что приложения, которые должны взаимодействовать с БД, проектируются параллельно с ней и изменяются вместе с изменением ее модели.

Многотабличная база данных является сложным многокомпонентным объектом. Основной задачей проектировщика базы данных является обоснованный выбор такой ее модели, которая обеспечит согласованное взаимодействие всех ее компонентов в соответствии с конкретным назначением информационной системы.

При проектировании таблиц рекомендуется руководствоваться несколькими основными принципами:

1. Информация в таблице **не должна дублироваться**.

Из-за избыточности данных, во-первых, неэкономно используется память компьютера, во-вторых, при дублировании информации нет защиты от ошибок ввода. Например, в таблице библиотечного каталога повторяются

фамилии писателей. Что нужно сделать, чтобы избежать дублирования? Решение простое – внести список авторов книг в отдельную таблицу, а в каталоге хранить только их коды.

2. Должны существовать **отношения и связи** между таблицами.

Связи необходимо создавать, чтобы отдельные таблицы функционировали как единое целое. Таблицы связываются между собой при помощи ключевых полей. В вышеприведенном примере библиотечный каталог и список авторов связываются между собой при помощи ключевого поля код автора. В списке авторов поле код является первичным ключом (Primary Key), а в каталоге код автора является внешним ключом (Foreign Key).

Вообще, между таблицами в реляционной БД существует три вида связи:

Один к одному (1 : 1) – одной записи в первой таблице соответствует ровно одна запись во второй. Например, каждому клиенту библиотеки соответствует ровно один читательский билет. Справочная таблица клиентов и список номеров читательских билетов в отношении один к одному.

Один ко многим (1 : N) – одной записи в первой таблице соответствует многие записи во второй. Например, есть таблица заказов книг и библиотечный каталог. Ведь в одном заказе может быть много книг, поэтому наблюдаем здесь связь один ко многим.

Многие ко многим (N : M) – многим записям в первой таблице соответствует многие записи во второй. В этом случае связь реализуется через дополнительную (одну или более) таблицу через связь «один ко многим». Например, есть таблица, в которой хранится информация об областях знаний и таблица с информацией о хранящихся в библиотеке книгах с их инвентарными номерами (среди книг есть такие, которые присутствуют в нескольких экземплярах). Для установления связи **N : 1 : M** в качестве промежуточной таблицы подойдет каталог книг. Одна книга может относиться сразу к нескольким областям знаний, одной книге может соответствовать несколько инвентарных номеров.

Давайте вспомним лабораторную работу с предыдущего урока, схему БД **Магазин**. Три отдельные таблицы Товары, Продавцы и Изготовители содержали информацию о товарах. Каждая из них содержала ключевое поле Код. Таблицы Изготовители и Модели были связаны отношением **один к одному**, так как каждой модели товара соответствует один изготовитель. В основной таблице Прайс-лист были установлены внешние ключи, с помощью которых она связывалась с другими таблицами. Таблицы Товары, Продавцы, Изготовители, Модели имели тип связи с таблицей Прайс-лист **один ко многим**. В таблице Прайс-лист может быть сколь угодно записей, содержащих информацию об одном и том же товаре, продавце, изготовителе или модели, такую таблицу принято называть **подчиненной**.

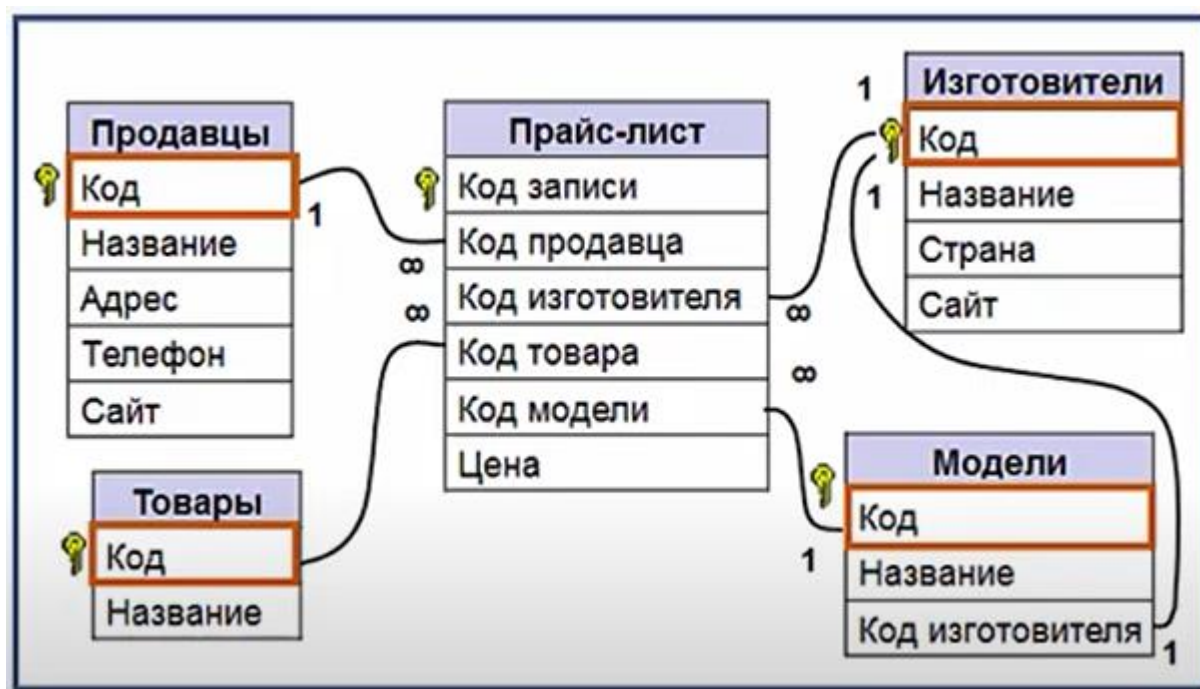


Рис. 125. Пример отношений между таблицами в реляционной базе данных

3. Должна соблюдаться целостность данных

Рассмотрим таблицу библиотечного каталога и таблицу с инвентарными номерами книг. Если мы удалим какую-нибудь запись из первой таблицы, то их инвентарные номера во второй таблице станут недействительными, что может привести к нарушению целостности данных. Проблему можно решить двумя способами:

А. запретить удаление записей из первой таблицы, пока имеются связанные с ней записи в других;

Б. выполнить каскадное удаление данных. Если удаляется запись из первой таблицы, то удаляются все записи в связанной с ней подчиненной таблице, в которой записей об одной и той же книге каталога может быть несколько.

Правильно спроектированные таблицы и связи в базе данных позволяют избежать избыточности данных и сохранять их целостность. Процесс создания таких таблиц и связей между ними называют **нормализацией**. Заметим, что излишняя нормализация может привести к замедлению работы с БД, так как приходится делать выборку из нескольких таблиц вместо одной. Если такая проблема возникает, то применяют денормализацию, которая все данные обратно сводит в единую таблицу.

Для проектирования баз данных чаще всего используют Entity-Relationship model или ER – модель.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1

В сети интернет осуществите поиск информации о ER – моделях. Можете воспользоваться ссылкой <https://www.lucidchart.com/pages/ru/erd-%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0> или использовать другие источники.

Законспектируйте основные правила и особенности построения ER-моделей.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Условие:

Представьте, что вам надо сделать заказ билетов в кинотеатре. Билеты на фильмы можно купить в день показа или забронировать заранее. Известны цены, временные рамки всех сеансов, в том числе и старых. Один и тот же фильм может быть показан на многих сеансах, но цена может быть разной. Спроектируйте базу данных, постройте ER-модель.

Технология решения:

Выделим сущности:

- Сеанс
- Фильм
- Зритель
- Билет
- Бронь
- Цена

Каждой сущности соответствует своя таблица в базе данных. Определим конкретный набор атрибутов для сущности, рассмотрев задачи, в которых они используются:

- Сеанс (*Номер сеанса*, Номер фильма, Дата показа, Номер стоимости);
- Фильм (*Номер фильма*, Название, Продолжительность, Краткое описание);
- Зритель (*Номер зрителя*, ФИО, Дата рождения);
- Билет (*Номер билета*, Номер сеанса, Номер стоимости);
- Бронь (*Номер брони*, Номер сеанса, Дата брони);
- Стоимость (*Номер стоимости*, Номер сеанса, Стоимость билета).

Мы обозначили курсивом первичные ключи – атрибуты сущности, уникально ее характеризующие. Подчеркиванием мы обозначили внешние ключи – атрибуты, уникально характеризующие сущности, на которые они ссылаются.

Для того чтобы определить отношения между сущностями, используем механизм связей. Чтобы было легче разобраться в том, как соотносятся сущности, имя связи должно нести в себе определенный смысл. Например, отношение между сущностями Зритель и Билет можно определить как «Покупает». Каждый Сеанс «Имеет» Стоимость и «Включает» Фильм. Отношение между Зритель и Бронь можно назвать «Заказывает», а между

Бронь и Сеанс – «Выдается на». Построим ER-модель, в которой будем использовать названия связей и стандартные обозначения:

- один-к-одному (1:1)
- один-ко-многим (1:N)

Результат работы:

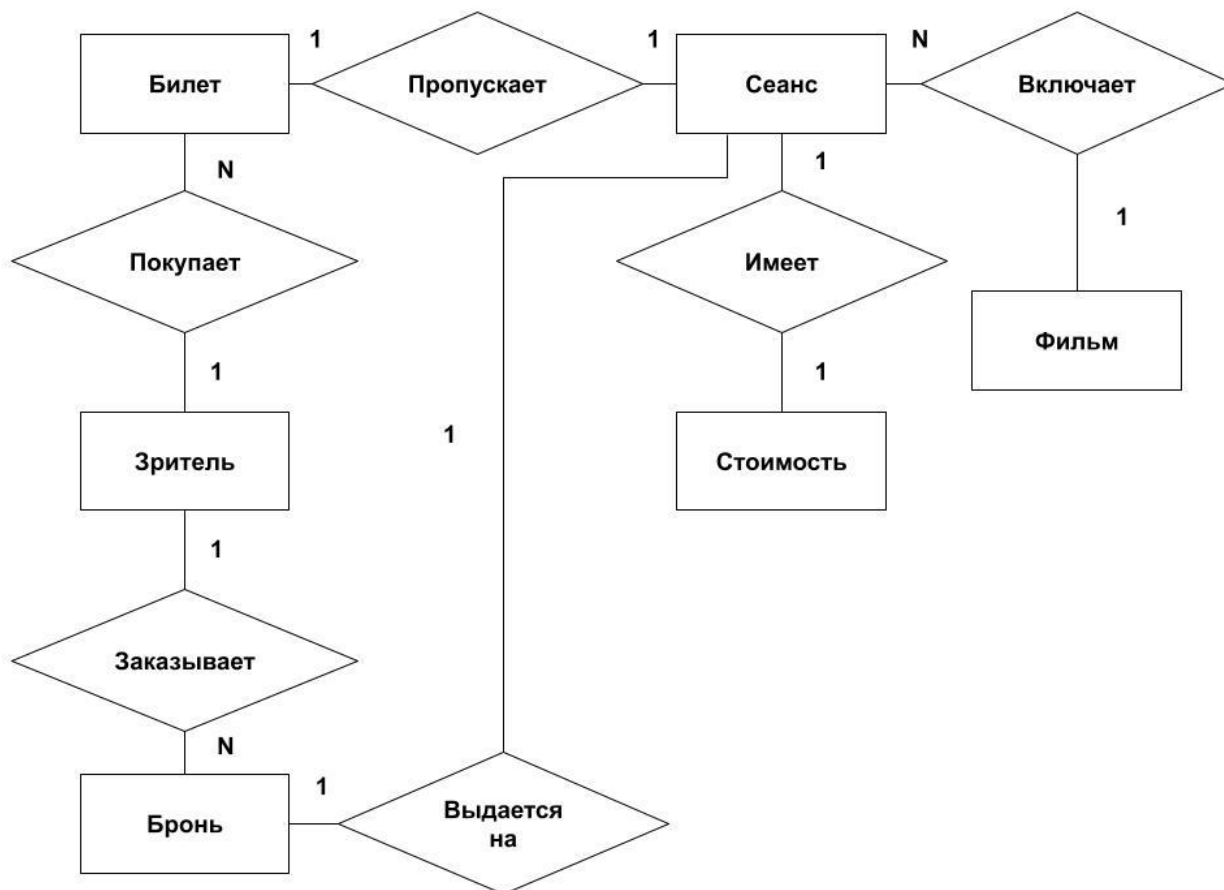


Рис. 126. Механизм связей в ER-модели

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Практическая работа №38

Спроектировать модель БД для аптек, в которой рассматриваются следующие сущности: Аптека; Группа препаратов; Наличие; Дефицит; Сотрудник; Клиент; Корзина; Покупка.

Опишем отношения между сущностями. Сущность *Аптека* связана отношением *Имеет* «один ко многим» с сущностью *Наличие*, отношением *Имеет* «один ко многим» с сущностью *Дефицит*, отношением *Имеет* «один ко многим» с сущностью *Покупка*, отношением *Работает* «один ко многим» с сущностью *Сотрудник*. Сущности *Наличие* и *Дефицит* связаны отношениями *Включает* «многое к одному» с сущностью *Препарат*. Сущность *Препарат* кроме упомянутых связей связано отношением *Включает* «многое к одному» с сущностью *Группа* и отношением *Включает* «один ко многим» с сущностью *Покупка*. Сущность *Покупка* кроме упомянутых связей связана отношением *Включает* «многое к одному» с

сущностью *Корзина*. Сущность *Сотрудник* кроме упомянутой связи с сущностью *Аптека* связан отношением *Регистрирует* «один ко многим» с сущностью *Корзина*. Сущность *Клиент* связан отношением *Имеет* «один ко многим» с сущностью *Корзина*.

Постройте ER-модель.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Практическая работа №39

Спроектировать базу данных «Приемная комиссия в ВУЗе» в логическом и физическом смысле в SqliteStudio по описанию (в качестве справочного материала можно использовать Справочник абитуриента).

Решаемые задачи:

- учет абитуриентов (личные сведения, какое учебное заведение закончил, экзамены, сданные по системе ЕГЭ, количество баллов, на какой факультет и на какую специальность поступает)
- получение данных (абитуриенты, поступающие на факультет X, на специальность Y, количество девушек (юношей), поступающих на факультет X, на специальность Y)
- поиск данных (сведения об абитуриенте по первым буквам фамилии, сведения о специальностях на факультете)
- вычисления (средний балл ЕГЭ для поступающих на факультет X).

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

О проектировании: <https://www.youtube.com/watch?v=GQaTW6N1GY>

О нормализации баз данных:
<https://www.youtube.com/watch?v=1GWx5CZdSCg>
<https://analytics.infozone.pro/vvedenie-v-proektirovanie-baz-dannih/>

§4.4 Многотабличные БД. Решение задач.

ВСПОМИНАЕМ

Вспомним:

1. Какую базу данных называют «реляционной»?
2. Что из себя представляет ER-Модель?

ЧИТАЕМ

База данных – это хранилище больших объемов данных некоторой предметной области, организованное в определенную структуру, то есть хранящихся в упорядоченном виде.

Данные в реляционных БД представлены в виде таблиц. Строки таблицы принято называть записи, а столбцы — поля:

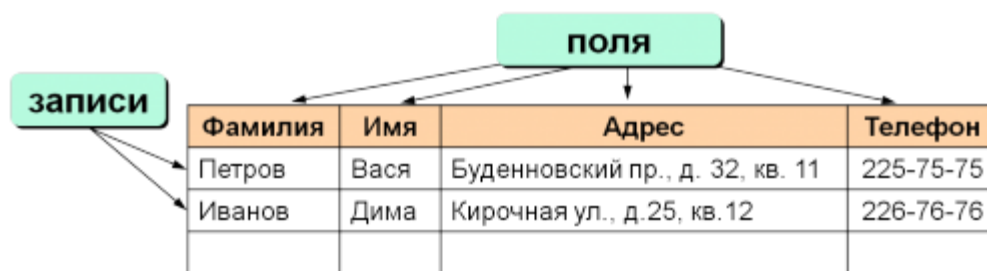


Рис. 127 Структура таблицы

Любая таблица обладает следующими свойствами:

- Все поля должны иметь уникальные имена. В примере: Фамилия, Имя, Адрес, Телефон.
- Поля имеют различные типы данных, в зависимости от их значения (например, символьный, целочисленный, вещественный, денежный, текст, дата, время и др.).
- Поля могут быть как обязательными для заполнения, так и нет.
- Таблица может иметь очень большое количество записей.
- Таблица может иметь ключевое поле, которое однозначно определяет запись.
- В таблице не может быть двух и более записей с одинаковым значением ключевого поля (ключа).
- Для выбора ключевого поля берутся какие-либо уникальные данные об объекте: например, номер паспорта человека (второго такого номера ни у кого нет).
- Если в таблице не предусмотрены такие уникальные поля, то создается поле с названием ID или Код с уникальными номерами (значениями счетчика для каждой записи в таблице).

В реляционной БД между таблицами устанавливается отношение, выраженное через связь, которая создается с помощью ключевых полей.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Посмотрите видеоразбор задания №3 из ЕГЭ:

<https://www.youtube.com/watch?v=NzuSqAmkJrk>

При необходимости просмотрите ещё несколько таких разборов. После чего в рабочей тетради создайте алгоритм решения задания №3 как «вручную», так и с помощью языка программирования.

РЕШАЕМ В ТЕТРАДИ

Задание 33.

Ниже в табличной форме представлен фрагмент базы данных:

| №п/п | Наименование товара | Цена | Количество | Стоимость |
|------|---------------------|------|------------|-----------|
| 1 | Монитор | 7654 | 20 | 153080 |
| 2 | Клавиатура | 1340 | 26 | 34840 |
| 3 | Мышь | 235 | 34 | 7990 |
| 4 | Принтер | 3770 | 8 | 22620 |
| 5 | Аудиоколонки | 410 | 16 | 6560 |
| 6 | Экшн-камера | 2760 | 10 | 27600 |

На какой позиции окажется товар «Экшн-камера», если произвести сортировку данной таблицы по возрастанию столбца «Количество»?

- 1) 5 2) 2 3) 3 4) 6

РЕШАЕМ В ТЕТРАДИ

Задание 34.

На игровом Интернет-портале есть следующая статистика об играх и количестве играющих:

| Аркадные | Логические | Словесные | Спортивные |
|---------------|-----------------|-----------|------------|
| Астероид | Фишдом | Виселица | Шашки |
| Веселая ферма | Филлер | Сканворд | Боулинг |
| Пузырьки | Снежные загадки | Анаграмма | Футбол |

| Игра | Количество играющих |
|-----------------|---------------------|
| Астероид | 536 |
| Шашки | 353 |
| Боулинг | 60 |
| Веселая ферма | 264 |
| Виселица | 981 |
| Анаграмма | 288 |
| Сканворд | 119 |
| Снежные загадки | 93 |
| Пузырьки | 100 |
| Филлер | 458 |

| | |
|--------|-----|
| Фишдом | 437 |
| Футбол | 572 |

Определите, игры какого типа пользуются наибольшей популярностью у игроков (в игры какого типа играет наибольшее количество людей)?

- 1) Аркадные 2) Логические 3) Словесные 4) Спортивные

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Условие:

В фрагменте базы данных представлены сведения о родственных отношениях. На основании приведённых данных определите, сколько прямых потомков (т.е. детей и внуков) Павленко А.К. упомянуты в таблице 1.

Таблица 1

| ID | Фамилия И.О. | Пол |
|------|----------------|-----|
| 2146 | Кривич Л.П. | Ж |
| 2155 | Павленко А. К. | М |
| 2431 | Хитрук П.А. | М |
| 2480 | Кривич А.А. | М |
| 2302 | Павленко Е. А. | Ж |
| 2500 | Сокол Н.А. | Ж |
| 3002 | Павленко И. А. | М |
| 2523 | Павленко Т. Х. | Ж |
| 2529 | Хитрук А.П | М |
| 2570 | Павленко П. И. | М |
| 2586 | Павленко Т. И. | Ж |
| 2933 | Симонян А. А. | Ж |
| 2511 | Сокол В.А. | Ж |
| 3193 | Биба С.А. | Ж |

Таблица 2

| ID_Родителя | ID_Ребенка |
|-------------|------------|
| 2146 | 2302 |
| 2146 | 3002 |
| 2155 | 2302 |
| 2155 | 3002 |
| 2302 | 2431 |
| 2302 | 2511 |
| 2302 | 3193 |
| 3002 | 2586 |
| 3002 | 2570 |
| 2523 | 2586 |
| 2523 | 2570 |
| 2529 | 2431 |
| 2529 | 2511 |

Возможное решение:

1. Сначала находим в таблице 1 Павленко А.К. (ID = 2155)
2. Теперь по таблице 2 ищем его детей – их идентификаторы 2302 и 3002; можно строить генеалогическое дерево:

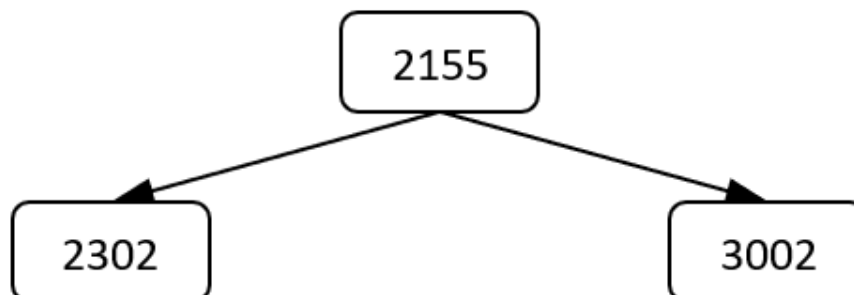


Рис. 128 Дерево 1

3. Далее так же определяем внуков 2155, то есть, детей 2302 и 3002:

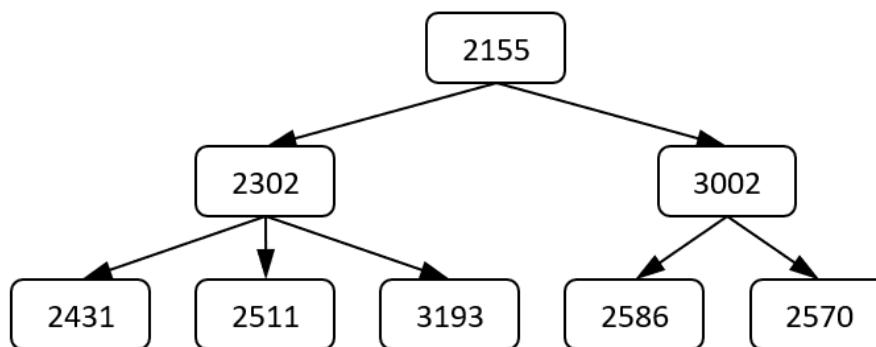


Рис. 3 Дерево 2

4. Как следует из таблицы, данных о правнуках 2155 в таблице нет.
5. Всего прямых потомков 7 – двое детей и 5 внуков.
6. Ответ: 7.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №40

В [файле](#) (Практическая_работа_40) представлены таблицы из базы данных «Продукты» о поставках товаров в магазины районов города. База данных состоит из трёх таблиц.

Таблица «Движение товаров» содержит записи о поставках товаров в магазины в течение первой декады июня 2021 г., а также информацию о проданных товарах. Поле *Тип операции* содержит значение *Поступление* или *Продажа*, а в соответствующее поле *Количество упаковок, шт.* занесена информация о том, сколько упаковок товара поступило в магазин или было продано в течение дня. Заголовок таблицы имеет следующий вид.

| ID операции | Дата | ID магазина | Артикул | Тип операции | Количество упаковок, шт. | Цена, руб./шт. |
|-------------|------|-------------|---------|--------------|--------------------------|----------------|
|-------------|------|-------------|---------|--------------|--------------------------|----------------|

Таблица «Товар» содержит информацию об основных характеристиках каждого товара. Заголовок таблицы имеет следующий вид.

| Артикул | Отдел | Наименование | Ед. изм. | Количество в упаковке | Поставщик |
|---------|-------|--------------|----------|-----------------------|-----------|
|---------|-------|--------------|----------|-----------------------|-----------|

Таблица «Магазин» содержит информацию о местонахождении магазинов. Заголовок таблицы имеет следующий вид.

| ID магазина | Район | Адрес |
|-------------|-------|-------|
|-------------|-------|-------|

На рисунке приведена схема указанной базы данных.



Рис. 4 Схема базы данных Продукты

Используя информацию из приведённой базы данных, определите насколько увеличилось количество упаковок яиц диетических, имеющихся в наличии в магазинах Заречного района, за период с 1 по 10 июня включительно.

В ответе запишите только число.

§4.5 Разработка реляционных БД. Язык запросов SQL

ВСПОМИНАЕМ

Вспомним:

1. Какие объекты являются составляющими баз данных?
2. Как реализуются связи между объектами?

ЧИТАЕМ

Для работы с большими объёмами данных необходимо автоматизировать их обработку, иначе теряется смысл использования компьютера. Для такой автоматизации используются специальные языки управления запросами. Один из самых популярных языков – SQL.

SQL — это язык программирования структурированных запросов, который применяется в качестве эффективного способа хранения данных, поиска их фрагментов, получения результатов вычисления с данными, обновления данных, частичного извлечения их из базы и удаления.



Рис. 130 Схема функционирования СУБД

SQL можно назвать главным инструментом оптимизации и обслуживания базы данных. Чтобы понять, для чего нужен SQL, рассмотрим простой и наглядный пример.

ГОТОВИМСЯ К РАБОТЕ НА КОМПЬЮТЕРЕ

Представим себе таблицу с информацией о школьниках: имена, дата рождения, телефон, пол, класс, фото и другие. В ней есть определённое количество записей и полей:

| Ученики | | | | | | | | | |
|-------------|----------|----------|--------------|------------|----------|-----|-------|---------|-------------|
| код_ученика | фамилия | имя | отчество | дата_рожде | телефон | пол | класс | фото | характерист |
| 1 | Шатилова | Юлия | Игоревна | 12.06.1990 | 2222222 | | 11а | Package | |
| 2 | Яркина | Анна | Владимировна | 30.08.1990 | 3333333 | | 11а | Package | |
| 3 | Брыков | Леонид | Петрович | 01.01.1990 | 44444444 | ✓ | 11б | Package | |
| 4 | Лыков | Владимир | Иванович | 02.02.1990 | 5555555 | ✓ | 11б | Package | |
| 5 | Хорин | Никита | Сергеевич | 03.03.1990 | 6666666 | ✓ | 11в | Package | |
| 6 | Гончаров | Иван | Петрович | 04.04.1990 | 7777777 | ✓ | 11ж | Package | |
| (№) | | | | | | | | | |

Рис. 131 Таблица 1

И представим таблицу с информацией об учителях: имена, дата рождения, телефон, пол, класс, фото и другие.

| Учителя | | | | | | | |
|-------------|-----------|----------|-------------|-------------|------------|--------------|------------|
| код_учителя | фамилия | имя | отчество | предмет | телефон | адрес | e-mail |
| 4 | Ткачева | Татьяна | Васильевна | информатика | 243536476 | опаороро | гнгггггггг |
| 5 | Княженко | Наталья | Геннадиевна | математика | 3448758568 | роолуу98787 | оророр |
| 6 | Бирюков | Владимир | Николаевич | физика | 46465758 | 8667987оорв | ггвгплвар |
| 7 | Пилкина | Ольга | Анатолевна | информатика | 356276868 | 5756845949 | арвовоо |
| 8 | Лобов | Геннадий | Романович | информатика | 7897870707 | hghgglghlg | iuiiytytt |
| 9 | Меньших | Зинаида | Васильевна | математика | 5873878707 | лодлаолаозао | ггге9нгггг |
| 10 | Лобанович | Мария | Петровна | физика | 9808979797 | нррлрлрлрждо | к4466 |
| (№) | | | | | | | |

Рис. 132 Таблица 2

Как построить взаимодействие между этими двумя таблицами и получить информацию о процессах? Путем установления связей через новые таблицы. Какие? Конечно же, учителей и учеников связывают предметы и экзамены. Добавим справочник про классы и установим связи по определенным полям:

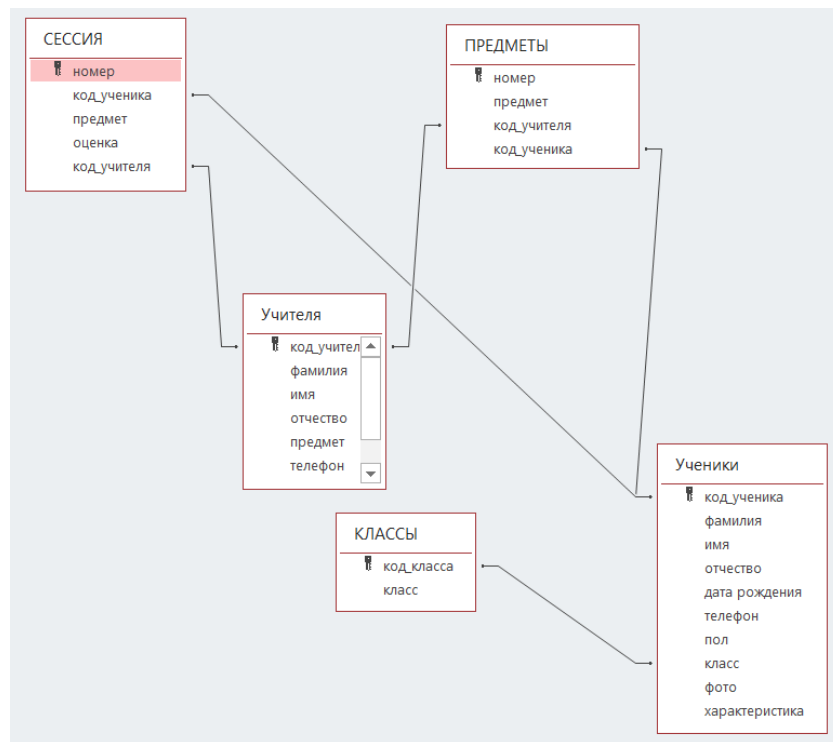


Рис. 133 Схема данных базы данных Школа

Получили Database Schema (схему данных), то есть структуру данных, которой можно управлять на формальном языке, поддерживаемом СУБД. Представьте, что нам нужно получить информацию об учащихся, которые сдали все экзамены на пять, например для того, чтобы поощрить. Необходимо попросить СУБД предоставить информацию о школьниках по заданным условиям. В этом случае создается и выполняется запрос данных на языке SQL. Этот язык является языком структурированных запросов. Инструкция на SQL будет иметь синтаксический вид:

```
SELECT * FROM СЕССИЯ WHERE оценка = 5;
```

Но чтобы оптимизировать запрос и вывести список школьников без повторения имен, придется еще добавить других инструкций. SQL поможет нам также вычислить некоторую статистическую информацию. Например, средний балл учеников, сдавших математику, или количество учеников, имеющих хотя бы одну задолженность в экзаменационную сессию.

Рассмотрим несколько примеров запросов для описанной выше БД:

1) Вывести ФИО и предмет учителей, фамилия которых начинается на «Л» и заканчивается на «ч»:

```
SELECT учителя.фамилия, учителя.имя, учителя.отчество,
учителя.предмет
FROM учителя
WHERE (((учителя.[фамилия]) In (SELECT фамилия
FROM учителя
WHERE фамилия like '*ч';) And
(учителя.[фамилия]) Like 'Л*'));
```

Результат выполнения запроса:

| фамилия | имя | отчество | предмет |
|-----------|-------|----------|---------|
| Лобанович | Мария | Петровна | физика |
| * | | | |

Рис. 134 Запрос 1

2) Вывести всех учеников (с указанием класса) учителя с фамилией, начинающейся на «Л»:

```
SELECT учителя.фамилия AS учитель, ученики.класс AS класс,
ученики.фамилия AS ученик
FROM учителя, ученики, предметы
WHERE учителя.код_учителя=предметы.код_учителя AND
ученики.код_ученика=предметы.код_ученика AND учителя.фамилия
like 'Л*';
```

Результат выполнения запроса:

| Запрос2 | учитель | класс | ученик |
|---------|---------|-------|----------|
| | Лобов | 11в | Хорин |
| | Лобов | 11ж | Гончаров |

Рис. 135 Запрос 2

3) Вывести информацию о всех учениках в порядке имен, противоположном порядку алфавита:

```
SELECT фамилия, имя, отчество, фото
FROM ученики
ORDER BY 1 DESC;
```

Результат выполнения запроса:

| фамилия | имя | отчество | фото |
|----------|----------|--------------|---------|
| Яркина | Анна | Владимировна | Package |
| Шатилова | Юлия | Игоревна | |
| Хорин | Никита | Сергеевич | |
| Лыков | Владимир | Иванович | |
| Гончаров | Иван | Петрович | |
| Брыков | Леонид | Петрович | |
| * | | | |

Рис. 136 Запрос 3

4) Вывести всю информацию об учителях, принявших экзамены у конкретных учеников, и поставленных учителями оценок:

```
SELECT *
FROM учителя, сессия
WHERE учителя.код_учителя=сессия.код_учителя;
```

Результат выполнения запроса:

| учителя.код | фамилия | имя | отчество | учителя.пре | телефон | адрес | e-mail | номер | код_ученики | сессия.пред | оценка | сессия.код |
|-------------|----------|----------|-------------|-------------|------------|-------------|------------|-------|-------------|-------------|--------|------------|
| 1 | Ткачева | Татьяна | Васильевна | информатика | 243536476 | оаоророро | гнгггггггг | 30 | 2 | информатика | 3 | 4 |
| 4 | Ткачева | Татьяна | Васильевна | информатика | 243536476 | оаоророро | гнгггггггг | 33 | 3 | информатика | 2 | 4 |
| 5 | Княженко | Наталья | Геннадиевна | математика | 3448758568 | роолуу98787 | оророр | 29 | 1 | математика | 5 | 5 |
| 6 | Бирюков | Владимир | Николаевич | физика | 46465758 | 8667987оорв | гггггггггг | 28 | 1 | физика | 4 | 6 |
| 6 | Бирюков | Владимир | Николаевич | физика | 46465758 | 8667987оорв | гггггггггг | 32 | 2 | физика | 4 | 6 |
| 6 | Бирюков | Владимир | Николаевич | физика | 46465758 | 8667987оорв | гггггггггг | 34 | 3 | физика | 3 | 6 |
| 7 | Пилина | Ольга | Анатольевна | информатика | 356276868 | 5756845949 | аровооо | 27 | 1 | информатика | 2 | 7 |
| 9 | Меньших | Зинаида | Васильевна | математика | 5873878707 | лодлаолаоао | гггггггггг | 31 | 2 | математика | 5 | 9 |
| 9 | Меньших | Зинаида | Васильевна | математика | 5873878707 | лодлаолаоао | гггггггггг | 35 | 3 | математика | 2 | 9 |

Рис. 137 Запрос 4

5) Вывести фамилии и имена учеников, сдавших экзамен по математике:

```
SELECT ученики.фамилия, ученики.имя, ученики.класс,
сессия.оценка AS математика
FROM сессия, ученики
WHERE (сессия.оценка>2 AND сессия.предмет='математика' AND
ученики.код_ученика=сессия.код_ученика) ;
```

Результат выполнения запроса:

| фамилия | имя | класс | математика |
|----------|------|-------|------------|
| Шатилова | Юлия | 11a | 5 |
| Яркина | Анна | 11a | 5 |

Рис. 138a Запрос 5

SQL, простой и лёгкий в изучении язык из области свободного программного обеспечения, активно применяется с 1974 года:

- разработчиками баз данных (обеспечивают функциональность приложений),
- тестировщиками (в ручном и автоматическом режиме),
- администраторами (выполняют поддержание работоспособности среды).

Взаимодействие с базами данных на SQL происходит быстро даже в ситуациях, когда объёмы данных велики (Big Data).

Области применения SQL:

- SQL DDL – язык определения данных;
- SQL DML – язык управления данными (изменение и извлечение данных);
- SQL DCL – язык контроля данных (защита данных от повреждения и неправильного использования);
- SQL клиент/сервер – настройка единого входа с проверкой подлинности пользователя в нескольких веб-приложениях;
- SQL трёхуровневой архитектуры – защищает БД от несанкционированного использования и копирования.

ЧИТАЕМ

Рассмотрим основные конструкции языка SQL.

Базовый оператор SELECT

SELECT имеет следующий синтаксис:

```
SELECT select_list  
[INTO new_table]  
FROM table_source  
[WHERE search_condition]  
[GROUP BY group_by_expression]  
[HAVING search_condition]  
[ORDER BY order_expression [ASC \ DESC] ]
```

В операторе SELECT указываются столбцы выбираемых данных, место хранения столбцов, критерий отбора данных и порядок сортировки, применяемый к этим данным. Более того, с помощью оператора SELECT можно сгруппировать данные и назначить критерий отбора на уровне группы.

При указании конструкции INTO результат выполнения запроса будет сохранен в новой таблице. Единственный аргумент, new_table, определяет имя таблицы, в которую будут вставлены результаты.

Ключевое слово FROM описывает таблицы или представления, связанные с изменяемой таблицей.

Добавление директивы WHERE

Используя SELECT, можно сузить результаты запроса, добавив директиву WHERE. В директиве WHERE в качестве критерия отбора строк можно использовать несколько столбцов. Для проверки значений полей можно использовать различные операторы сравнения. Описание этих операторов приведено в таблице.

Таблица 5. Операторы сравнения директивы WHERE

| Описание | Оператор |
|----------------|--|
| = | Равно |
| > | Больше чем |
| < | Меньше чем |
| >= | Больше либо равно |
| <= | Меньше либо равно |
| <> | Не равно |
| IN | Входит в заданный список значений |
| between... and | Входит в интервал значений |
| like | Содержит значение, соответствующее заданному шаблону. Шаблон для сравнения – это строка, состоящая из одного |

| | |
|--|--------------------------------------|
| | или нескольких символов подстановки. |
|--|--------------------------------------|

Помимо перечисленных выше операторов сравнения, значения столбцов, включаемые в директиву WHERE, можно проверять с помощью логических операторов AND и OR.

Добавление директивы **ORDER BY**

Достаточно добавить директиву ORDER BY, чтобы отсортировать записи по любому столбцу. Для сортировки в порядке убывания используется ключевое слово DESC.

Использование директивы WHERE для объединения таблиц

Не всегда будет достаточно просмотра данных только из одной таблицы. Данные в таблице могут быть связаны с данными других таблиц базы данных. Для объединения нескольких таблиц вместе в одном операторе SELECT можно использовать директиву WHERE:

```
SELECT author.au_lname, author.au_fname, titles.title
FROM author, titleauthor, titles
WHERE authors.au_id =titleauthor.au_id AND
Titleauthor.title_id=titles.title_id
ORDER BY authors.au_lname, authors.au_fname, titles.title
```

Нужно обратить внимание на способ записи имен столбцов в этом операторе. Во избежание ошибки неоднозначности при отборе данных одним оператором SQL из нескольких таблиц (заданных директивой FROM) для столбцов с одинаковыми именами в различных таблицах необходимо задавать в виде префиксов имена соответствующих таблиц.

Использование оператора **JOIN** для объединения таблиц

Объединение таблиц может быть реализовано и другим способом, а именно – с помощью оператора JOIN. Этот метод объединения таблиц имеет следующий синтаксис:

```
SELECT столбец 1, столбец 2, столбец 3
FROM таблица 1 оператор_объединения таблица 2
ON критерий_объединения
```

Рассматриваемый нами оператор JOIN определяет, каким образом должны быть возвращены строки из объединяемых таблиц. Директива ON действует подобно директиве WHERE, указывая, в каких полях в объединяемых таблицах должно проверяться равенство значений. Различные операторы объединения описаны в таблице 6.

Таблица 6. Операторы объединения таблиц

| Оператор | Описание |
|--------------------|---|
| CROSS JOIN | Возвращает каждую строку из первой таблицы, объединенную с каждой строкой из второй таблицы. Количество строк результата равно произведению числа строк в каждой таблиц |
| INNER JOIN | Возвращает все строки из каждой таблицы, удовлетворяющие критерию отбора, который задан в директиве WHERE, при условии совпадения значений в объединяемых полях, указанных в директиве ON |
| LEFT[OUTER] JOIN | Возвращает все строки таблицы, которые находятся слева в объединении, удовлетворяющие критерию отбора директивы WHERE, и только те строки таблицы, расположенные справа в объединении, в которых существует совпадение по объединяемым полям, заданным в директиве ON |
| RIGHT [OUTER] JOIN | Возвращает все строки таблицы, которые находятся справа в объединении, удовлетворяющие критерию отбора директивы WHERE, и только те строки таблицы, расположенные слева в объединении, в которых существует совпадение по объединяемым полям, заданным директивой ON |
| FULL [OUTER] JOIN | Возвращает все строки из каждой таблицы, удовлетворяющие критерию отбора директивы WHERE, в которых нет совпадения по объединяемым полям, заданным директивой ON |

С помощью оператора объединения команду SQL из предыдущего примера можно записать следующим образом:

```
SELECT authors.au_iname, authors.au_fname, titles.title FROM
(authors INNER JOIN titleauthor
```

```
ON authors.au_id = titleauthor.au_id) INNER JOIN titles
ON titleauthor.title_id = titles.title_id
ORDER BY authors.au_lname, authors.au_fname, titles.title
```

Итоговые функции в операторах SQL

Итоговая функция может возвращать одно значение для всех строк, отобранных запросом. Если же в оператор SQL была добавлена директива GROUP BY, то такое итоговое значение вычисляется на каждом уровне группировки. Список итоговых функций, которые можно использовать в запросах, приведен в таблице 7.

Таблица 7. Итоговые функции

| Итоговая функция | Описание |
|------------------|--|
| AVG | Возвращает среднее из всех значений в столбце путем вычисления их общей суммы, деленной на количество строк. |
| COUNT | Возвращает количество значений, отличных от NULL, в указанном столбце или выражении. Если в качестве выражения задан символ звездочки (например, count(*)), то результатом является число строк, отобранных запросом |
| MIN | Возвращает минимальное значение в заданном столбце или выражении. |
| MAX | Возвращает максимальное значение в заданном столбце или выражении. |
| SUM | Возвращает сумму всех значений в заданном столбце или выражении |

В следующем примере функция count применяется ко всем строкам таблицы titles без группировки:

```
SELECT count (title) 'наименований'
FROM titles
```

Результат представляет собой количество записей в таблице titles.

Использование директивы GROUP BY

Предположим, что вам нужно сгруппировать все похожие строки результата выполнения запроса по значениям одного или нескольких столбцов. Это можно сделать, задав требуемые столбцы в директиве GROUPBY. Если задано несколько столбцов, то строки сначала

группируются по первому столбцу, а затем (внутри этих групп) – по второму столбцу и т.д. В следующем примере директива GROUPBY используется вместе с итоговой функцией, чтобы показать, как эта функция применяется к значениям внутри каждой отдельной группы:

```
SELECT authors.au_iname, authors.au_fname, count (titles.title)
'Названий'
FROM authors, titleauthor, titles
WHERE
    authors.au_id = titleauthor.au_id AND titleauthor.title_id =
titles.title_id
GROUP BY authors.au_iname, authors.au_fname
```

Использование директивы HAVING

Подобно директиве WHERE, директива HAVING используется для задания критерия отбора возвращаемых запросом данных. Различие связано с уровнем, на котором выполняется проверка критерия. В директиве WHERE критерий используется для ограничения количества строк, возвращаемых запросом. Затем директива GROUP BY формирует из этих строк группы и вычисляет заданные значения. После этого критерии директивы HAVING используются для ограничения количества групп в соответствии с данными на уровне группы. В следующем примере директива HAVING используется для выборки имен только тех авторов, которые написали более одной книги:

```
SELECT a.au_iname, a.au_fname, count (c.title) 'Названий'
FROM authors a, titleauthor b, titles c
WHERE a.au_id = b.au_id AND b.title_id = c.title_id
GROUP BY a.au_iname, a.au_fname
HAVING count (c.title) > 1
```

Создание таблиц

Создание таблиц выполняется инструкцией:

```
CREATE TABLE имя_таблицы
(имя_столбца тип [NULL| NOTNULL] [DEFAULT значение_по_умолчанию]
[ограничение_на_столбец]... ;
[, имя_столбца тип [NULL| NOTNULL] [DEFAULT
значение_по_умолчанию]
[ограничение на столбец]...]...
[ограничение на таблицу]...)
```

Удаление базы данных

При удалении базы данных или объекта базы данных удалятся все связанные с ними структуры и данные. Поэтому в большинстве систем

выполнять команду DROP разрешается только либо владельцу соответствующего объекта, либо лицу, наделенному специальными полномочиями. Синтаксис команды DROP DATABASE обычно имеет следующий вид:

```
DROP DATABASE имя_базы_данных
```

Это очень опасная команда, так как при ее выполнении уничтожается все содержимое базы данных.

Добавление, изменение и удаление данных

В SQL для изменения данных используются три основные команды (их часто называют операторами модификации данных).

- Оператор INSERT добавляет новые строки в базу данных;
- Оператор UPDATE изменяет существующие в базе данных строки;
- Оператор DELETE удаляет строки из базы данных.

С помощью каждого оператора модификации (INSERT, UPDATE, DELETE) за один раз можно изменять данные только в одной таблице.

```
INSERT INTO имя_таблицы [(столбец1 [, столбец2]...)]  
VALUES(константа1 [, константа2]...)
```

Для каждой добавляемой строки используется отдельный оператор INSERT.

Для получения данных из одной или нескольких таблиц в команде INSERT можно использовать оператор SELECT:

```
INSERT INTO имя_таблицы [(вставляемый_список_столбцов)]  
SELECT список_столбцов  
FROM список_таблиц  
WHERE условия
```

Оператор SELECT в команде INSERT позволяет взять данные из нескольких или всех столбцов одной таблицы и вставить их в другую таблицу.

Оператор UPDATE предназначен для изменения существующих в таблице данных. В операторе UPDATE нужно указать изменяемые строки и их новые значения. Новые данные могут быть константами или выражениями, или могут быть получены из других таблиц.

```
UPDATE имя_таблицы  
SET имя_столбца = выражение  
[WHERE условие]
```

Ключевое слово UPDATE предшествует названию таблицы или курсора (виртуальной таблицы). Как и в случае с другими операторами

модификации данных, в каждом операторе UPDATE можно изменять данные только одной таблицы.

Не менее важной, чем добавление и изменение строк, является возможность их удаления. Для удаления данных применяется команда DELETE. Подобно INSERT и UPDATE, команда DELETE позволяет манипулировать с одной или несколькими строками. Так же, как и в случае с другими операторами модификации данных, при удалении строк можно пользоваться информацией из других таблиц. Оператор DELETE имеет следующий синтаксис:

```
DELETE FROM имя_таблицы  
WHERE условие
```

В предложении WHERE определяются подлежащие удалению строки. В случае отсутствия предложения WHERE в операторе DELETE будут удалены все строки таблицы.

РЕШАЕМ В ТЕТРАДИ

Задание 35.

Опишите словами действия, выполняемые в запросах:

1.

```
SELECT name, database_id, create_date  
FROM sys.databases;
```

| | name | database_id | create_date |
|---|-----------|-------------|-------------------------|
| 1 | master | 1 | 2003-04-08 09:13:36.390 |
| 2 | tempdb | 2 | 2019-02-13 01:00:31.300 |
| 3 | model | 3 | 2003-04-08 09:13:36.390 |
| 4 | msdb | 4 | 2017-08-22 19:39:22.887 |
| 5 | b_library | 5 | 2019-02-17 04:38:30.220 |

Рис. 1386 Задание 1

2.

```
SELECT * FROM tAuthors;
```

| | AuthorId | AuthorFirstName | AuthorLastName | AuthorAge |
|---|----------|-----------------|----------------|-----------|
| 1 | 1 | Александр | Пушкин | 37 |
| 2 | 2 | Сергей | Есенин | 30 |
| 3 | 3 | Джек | Лондон | 40 |
| 4 | 4 | Шота | Руставели | 44 |
| 5 | 5 | Рабиндранат | Тагор | 80 |

Рис. 138в Задание 2

```
SELECT * FROM tAuthors ORDER BY AuthorId DESC;
```

| Результаты | | Сообщения | | |
|------------|----------|-----------------|----------------|-----------|
| | AuthorId | AuthorFirstName | AuthorLastName | AuthorAge |
| 1 | 5 | Рабиндранат | Тагор | 80 |
| 2 | 4 | Шота | Руставели | 44 |
| 3 | 3 | Джек | Лондон | 40 |
| 4 | 2 | Сергей | Есенин | 30 |
| 5 | 1 | Александр | Пушкин | 37 |

Рис. 138г Задание 3

4.

```
SELECT max(AuthorAge) FROM tAuthors;
```

| Результаты | | Сообщения | | |
|------------|---------------------------|-----------|--|--|
| | (Отсутствует имя столбца) | | | |
| 1 | 80 | | | |

Рис. 138д Задание 4

5.

```
INSERT INTO tAuthors VALUES ('Уильям', 'Шекспир', '51');
```

Проверим:

```
SELECT * FROM tAuthors;
```

| Результаты | | Сообщения | | |
|------------|----------|-----------------|----------------|-----------|
| | AuthorId | AuthorFirstName | AuthorLastName | AuthorAge |
| 1 | 1 | Александр | Пушкин | 37 |
| 2 | 2 | Сергей | Есенин | 30 |
| 3 | 3 | Джек | Лондон | 40 |
| 4 | 4 | Шота | Руставели | 44 |
| 5 | 5 | Рабиндранат | Тагор | 80 |
| 6 | 6 | Уильям | Шекспир | 51 |

Рис. 138е Задание 5

6.

```
UPDATE tAuthors SET AuthorFirstName = 'Лев', AuthorLastName='Толстой', AuthorAge = '82' WHERE AuthorId = '6';
```

Затем, обратимся к БД, чтобы вывести все имеющиеся записи:

```
SELECT * FROM tAuthors;
```

| Результаты | | Сообщения | | |
|------------|----------|-----------------|----------------|-----------|
| | AuthorId | AuthorFirstName | AuthorLastName | AuthorAge |
| 1 | 1 | Александр | Пушкин | 37 |
| 2 | 2 | Сергей | Есенин | 30 |
| 3 | 3 | Джек | Лондон | 40 |
| 4 | 4 | Шота | Руставели | 44 |
| 5 | 5 | Рабиндранат | Тагор | 80 |
| 6 | 6 | Лев | Толстой | 82 |

Рис. 138ж Задание 6

7.

```
DELETE FROM tAuthors WHERE AuthorId = '5';
```

Чтобы увидеть изменения, снова обратимся к базе для вывода всех записей:

```
SELECT * FROM tAuthors;
```

| Результаты | | Сообщения | | |
|------------|----------|-----------------|----------------|-----------|
| | AuthorId | AuthorFirstName | AuthorLastName | AuthorAge |
| 1 | 1 | Александр | Пушкин | 37 |
| 2 | 2 | Сергей | Есенин | 30 |
| 3 | 3 | Джек | Лондон | 40 |
| 4 | 4 | Шота | Руставели | 44 |
| 5 | 6 | Лев | Толстой | 82 |

Рис. 138з Задание 7

8.

```
SELECT BookId, BookTitle
FROM tBooks
WHERE Author = (SELECT AuthorId FROM tAuthors WHERE AuthorFirstName = 'Александр');
```

Получим:

| Результаты | | Сообщения | |
|------------|--------|--------------------|--|
| | BookId | Book Title | |
| 1 | 1 | Руслан и Людмила | |
| 2 | 2 | Кавказский пленник | |
| 3 | 3 | Евгений Онегин | |

Рис. 138и Задание 8

9.

```
SELECT tBooks.BookId, tBooks.BookTitle, tAuthors.AuthorFirstName,
tAuthors.AuthorLastName
FROM tBooks
JOIN tAuthors ON tAuthors.AuthorId = tBooks.Author;
```

| Результаты | | Сообщения | | |
|------------|--------|--------------------|-----------------|----------------|
| | BookId | Book Title | AuthorFirstName | AuthorLastName |
| 1 | 1 | Руслан и Людмила | Александр | Пушкин |
| 2 | 2 | Кавказский пленник | Александр | Пушкин |
| 3 | 3 | Евгений Онегин | Александр | Пушкин |
| 4 | 4 | Радунца | Сергей | Есенин |
| 5 | 5 | Преображение | Сергей | Есенин |
| 6 | 6 | Мартин Иден | Джек | Лондон |
| 7 | 7 | Морской волк | Джек | Лондон |
| 8 | 8 | Белый Клык | Джек | Лондон |

Рис. 138к Задание 9

10.

```
SELECT *
FROM tBooks
WHERE Author != ALL(SELECT AuthorId FROM tAuthors WHERE AuthorFirstName IN ('Александр', 'Сергей'));
```

Рис. 138л Задание 10

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №40

В рамках раздела «подготовка к работе за компьютером» мы разбирали пример базы данных с таблицами Ученики, Учителя, СЕССИЯ, ПРЕДМЕТЫ, КЛАССЫ. Создайте такую базу данных с помощью SQLiteStudio. Заполните таблицы по образцу или внесите больше записей.

Создайте связи между таблицами. Создайте запросы, примеры которых были разобраны.

§4.6 SQL запросы. Создание таблиц

ВСПОМИНАЕМ

Вспомним:

1. Что такое СУБД, и какую функцию она выполняет?

ЧИТАЕМ

SQLite – это библиотека, реализующая легковесную дисковую базу данных (БД), не требующую отдельного серверного процесса и позволяющую получить доступ к БД с использованием языка запросов SQL. SQLite может быть использована в приложениях для внутреннего хранения данных. Если существует необходимость, то можно легко перенести код в более многофункциональную БД, такую как PostgreSQL или Oracle. Документацию по модулю SQLite можно найти в разделе «Дополнительные информационные ресурсы».

ГОТОВИМСЯ К РАБОТЕ НА КОМПЬЮТЕРЕ

Чтобы воспользоваться SQLite3 в Python, необходимо импортировать модуль sqlite3, а затем создать объект подключения к БД.

Объект подключения создается с помощью метода connect():

```
import sqlite3
con = sqlite3.connect('mydatabase.db')
```

Курсор SQLite3

Для выполнения операторов SQL нужен объект курсора, создаваемый методом cursor().

Курсор SQLite3 – это метод объекта соединения. Для выполнения операторов SQLite3 сначала устанавливается соединение, а затем создается объект курсора с использованием объекта соединения следующим образом:

```
con = sqlite3.connect('mydatabase.db')
cursorObj = con.cursor()
```

Теперь можно использовать объект курсора для вызова метода `execute()` для выполнения любых запросов SQL.

`cur = con.cursor` — позволит нам производить операции с БД.

`cur.execute(...)` — создаёт таблицу.

`commit()` — сохранение БД.

ЧИТАЕМ

1) Создание соединения с БД

Импортируем модуль `sqlite3`, затем создадим объект подключения к базе данных с помощью метода `connect()`:

```
import sqlite3
Con = sqlite3.connect('mydb.db')
```

2) Создание курсора

Для выполнения операторов SQL, необходимо создать объект курсора методом `cursor()`:

```
Cur_Obj = Con.cursor()
```

Используем объект курсора для вызова метода `execute()`, который нужен для выполнения запросов SQL.

3) Создание таблицы

Создадим таблицу `T` для хранения, обновления, использования данных об абитуриентах, подавших заявления на специальности вуза. Допустим, она содержит минимальные сведения: имя, возраст, специальность, телефон, дату подачи заявления.

Код будет таким:

```
import sqlite3
from sqlite3 import Error

def sql_connection():
    try:
        return sqlite3.connect('mydb.db')
    except Error:
        print(Error)

def sql_tableT(con, cur, ):
    cur.execute(
        "CREATE TABLE T (id integer PRIMARY KEY, name text, age
integer, speciality text, telephone text, Date text)")
    con.commit()

Con = sql_connection()
```



```
Cur_Obj = Con.cursor()
Cur_Obj.execute('drop table T')
```

Данный программный код содержит следующие шаги: создание объекта подключения; создание объекта курсора; вызов метод execute с запросом CREATE TABLE в качестве параметра; создание таблицы T со следующими полями – id, name, age, speciality, telephone, date; метод commit() сохраняет все сделанные изменения.

4) Вставка данных в таблицу

Чтобы вставить данные в таблицу, воспользуемся оператором INSERT INTO. Рассмотрим следующую функцию и ее вызовы:

```
def sql_insert(con, cur, rec):
    cur.execute('INSERT INTO T(id, name, age, speciality,
telephone, date) VALUES(?, ?, ?, ?, ?, ?)' \
, rec)
    con.commit()
```

```
Con = sql_connection()
Cur_Obj = Con.cursor()
rec1 = (2, 'Иванов Иван Иванович', 18, 'IT', '+79258967123',
'2021-07-06')
rec2 = (1, 'Михайлов Михаил Михайлович', 20, 'TR',
'+79263648599', '2021-06-29')
sql_insert(Con, Cur_Obj, rec1)
sql_insert(Con, Cur_Obj, rec2)
```

Данный программный код содержит следующие шаги: создание кортежа данных для записи; создание объекта курсора; вызов метод execute с запросом INSERT INTO в качестве параметра; вставка данных; метод commit() сохраняет все сделанные изменения.

5) Обновление таблицы

Предположим, что нужно обновить поле name для id = 2. Для обновления будем использовать инструкцию UPDATE, воспользуемся WHERE в качестве условия для выбора нужного абитуриента.

Создадим такую функцию:

```
def sql_update(con, cur):
    cur.execute('UPDATE T SET name = "Петров Петр Петрович"
where id = 2')
    con.commit()
```

И вызовем ее:

```
Con = sql_connection()
Cur_Obj = Con.cursor()
sql_update(Con, Cur_Obj)
```

Это изменит имя 'Иванов Иван Иванович' на 'Петров Петр Петрович'.

6) Выборка данных

Оператор SELECT используется для выборки данных из одной или более таблиц. Если нужно выбрать все столбцы данных из таблицы, можете использовать звездочку (*). SQL синтаксис для этого будет следующим:

```
select * from table_name
```

В SQLite3 инструкция SELECT выполняется в методе execute объекта курсора. Например, выберем все строки и столбцы таблицы T:

```
cur.execute('SELECT * FROM T')
```

Если нужно выбрать несколько столбцов из таблицы, укажем их как показано ниже:

```
select column1, column2 from table_name
```

Например,

```
cur.execute('SELECT id, name FROM T')
```

Оператор SELECT выбирает все данные из таблицы T базы данных.

7) Выборка всех данных

Чтобы извлечь данные из БД, выполним инструкцию SELECT, а затем воспользуемся методом fetchall() объекта курсора для сохранения значений в переменной. При этом переменная будет являться списком, где каждая строка из БД будет отдельным элементом списка. Далее будет выполняться перебор значений переменной и печать значений:

```
def sql_fetch1(con, cur):  
    cur.execute('SELECT * FROM T')  
    # для получения количества строк нужно получить все данные, а  
    # затем получить длину результата  
    rows = Cur_Obj.fetchall()  
    print(len(rows))  
    for row in rows:  
        print(row)
```

Для извлечения конкретных данных из БД воспользуемся предложением WHERE. Например, выберем идентификаторы и имена тех абитуриентов, чей возраст превышает 19:

```
def sql_fetch2(con, cur):  
    cur.execute('SELECT id, name FROM T WHERE age > 19')  
    rows = cur.fetchall()  
    for row in rows:  
        print(row)
```

В приведенном выше операторе SELECT вместо звездочки (*) были указаны атрибуты id и name.

8) Удаление данных

Для удаления конкретных данных из БД, можно воспользоваться WHERE в сочетании с DELETE вместо SELECT. Когда оператор DELETE

используется без каких-либо условий, все строки в таблице будут удалены, а общее количество удаленных строк можно получить методом `rowcount`.

```
print(Cur_Obj.execute('DELETE FROM T').rowcount)
```

Если ни одна строка не удалена, будет возвращено 0.

9) Список таблиц

Чтобы вывести список всех таблиц в базе данных SQLite3, нужно обратиться к таблице `sqlite_master` (это главная таблица в SQLite3, в которой хранятся все таблицы), а затем использовать `fetchall()` для получения результатов из оператора `SELECT`.

```
def sql_fetch3(con, cur):  
    cur.execute('SELECT name from sqlite_master where type=  
"table"')  
    print(cur.fetchall())
```

10) Проверка существования таблицы

При создании таблицы необходимо убедиться, что таблица еще не существует. Аналогично, при удалении таблицы она должна существовать.

Чтобы проверить, существует ли нужная таблица, используем «if not exists» с оператором `CREATE TABLE` следующим образом:

```
import sqlite3  
con = sqlite3.connect('mydb.db')  
def sql_table(con):  
    cur = con.cursor()  
    cur.execute('create table if not exists TABLE(id integer,  
name text)')  
    con.commit()
```

```
sql_fetch(con)
```

11) Удаление таблицы

Удаление таблицы выполняется с помощью оператора `DROP`. Синтаксис оператора `DROP` выглядит следующим образом:

```
drop table table_name
```

Чтобы проверить, существует ли таблица при удалении, мы используем «if not exists» с инструкцией `DROP TABLE` следующим образом:

```
cur.execute('drop table if exists TABLE')
```

12) Одновременное добавление нескольких строк

Для вставки нескольких строк одновременно использовать оператор `executemany`. Рассмотрим, как реализован данный оператор в следующем коде:

```
import sqlite3  
con = sqlite3.connect('my_data_base.db')  
cur = con.cursor()
```

```

cur.execute('create table if not exists PROJECTS(id integer,
name text)')
data = [(1, "Ridesharing"), (2, "Water Purifying"), (3,
"Forensics"), (4, "Botany")]
cur.executemany("INSERT INTO projects VALUES(?, ?)", data)
con.commit()
cur.execute('SELECT * FROM projects')
rows = cur.fetchall()
print(len(rows))
for row in rows:
    print(row)

```

Мы создали таблицу DATA с двумя столбцами, в каждом столбце по четыре значения. Эта переменная передается методу executemany() вместе с запросом.

13) Исключения SQLite3

Исключением являются ошибки времени выполнения программы.

В SQLite3 у есть следующие основные исключения Python:

| | |
|--------------------------|---|
| DatabaseError | Любая ошибка, связанная с базой данных, вызывает ошибку DatabaseError |
| IntegrityError | Является подклассом DatabaseError и возникает, когда возникает проблема целостности данных, например, когда внешние данные не обновляются во всех таблицах, что приводит к несогласованности данных |
| ProgrammingError | Возникает, когда есть синтаксические ошибки или таблица не найдена или функция вызывается с неправильным количеством параметров / аргументов |
| OperationalError | Возникает при сбое операций базы данных, например, при необычном отключении, не по вине программиста |
| NotSupportedError | Возникает при использовании некоторых методов, которые не определены или не поддерживаются базой данных |

Обратите внимание, что использовался заполнитель для передачи значений.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №41

Собираем все упражнения в единый код.

```

import sqlite3
from sqlite3 import Error

```

```

def sql_connection():
    try:
        return sqlite3.connect('mydb.db')
    except Error:
        print(Error)

def sql_tableT(con, cur, ):
    cur.execute(
        "CREATE TABLE if not exists T (id integer PRIMARY KEY,
name text, age integer, speciality text, telephone text, Date
text)")
    con.commit()

def sql_tableN(con, cur, ):
    cur.execute(
        "CREATE TABLE N (id integer PRIMARY KEY, name text, age
integer, speciality text, telephone text, Date text)")
    con.commit()

def sql_update(con, cur):
    cur.execute('UPDATE T SET name = "Петров Петр Петрович"
where id = 2')
    con.commit()

def sql_insert(con, cur, rec):
    cur.execute('INSERT INTO T(id, name, age, speciality,
telephone, date) VALUES(?, ?, ?, ?, ?, ?)', rec)
    con.commit()

def sql_fetch1(con, cur):
    cur.execute('SELECT * FROM T')
    rows = Cur_Obj.fetchall()
    print(len(rows))
    for row in rows:
        print(row)

def sql_fetch2(con, cur):
    cur.execute('SELECT id, name FROM T WHERE age > 19')
    rows = cur.fetchall()
    for row in rows:
        print(row)

def sql_fetch3(con, cur):
    cur.execute('SELECT name from sqlite_master where type=
"table"')
    print(cur.fetchall())

```

```

Con = sql_connection()
Cur_Obj = Con.cursor()
Cur_Obj.execute('drop table if exists T')
sql_tableT(Con, Cur_Obj)
rec1 = (2, 'Иванов Иван Иванович', 18, 'IT', '+79258967123',
'2021-07-06')
rec2 = (1, 'Михайлов Михаил Михайлович', 20, 'TR',
'+79263648599', '2021-06-29')
sql_insert(Con, Cur_Obj, rec1)
sql_insert(Con, Cur_Obj, rec2)
sql_update(Con, Cur_Obj)
sql_fetch1(Con, Cur_Obj)
sql_fetch2(Con, Cur_Obj)
print(Cur_Obj.execute('DELETE FROM T').rowcount)
# создаем еще одну таблицу
Cur_Obj.execute('drop table if exists N')
sql_tableN(Con, Cur_Obj)
sql_fetch3(Con, Cur_Obj)
# Когда работа с БД завершена, рекомендуется закрыть соединение.
# Соединение может быть закрыто с помощью метода close()
Con.close()

```

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №44

Создать базу данных для хранения названий фильмов, а также дополнительной информации о них. Реализовать несколько функций для:

- добавления фильмов;
- получения данных обо всех фильмах;
- получения данных об одном фильме;
- обновления информации об одном фильме;
- удаления одного фильма.

Будем использовать Python для взаимодействия с базой данных SQLite, а фильмы позаимствуем с [сайта](#).

Что сделать?

Импортировать библиотеку SQLite, подключиться к базе данных и создать курсор

Как сделать?

Создание

```

import sqlite3

Con = sqlite3.connect('movies.db')
c = Con.cursor()

```

Объявим функцию, проверяющую наличие таблицы, имя которой будет передано в качестве ее параметра. Если таблица существует, функция вернет True, иначе — False

Воспользуемся объявленной функцией для проверки наличия таблицы, и в случае отсутствия — создадим её

Создадим функцию, добавляющую фильм в таблицу. В ней необходимо выполнить лишь одну инструкцию INSERT и сохранить то, что добавили

Чтение

Объявим функцию для получения данных обо всех фильмах

Объявим функцию для извлечения информации обо одном из фильмов

Обновление

Создадим функцию обновления данных с помощью словаря.

Ключами в update dictionary являются имена в соответствии со столбцами таблицы.

Необходимо

выполнить обновление всех числовых и текстовых полей для заданного ID фильма

```
def table_exists(table_name):
    c.execute(''SELECT count(name) FROM sqlite_master WHERE TYPE = 'table' \
AND name = '{}'''.format(table_name))
    if c.fetchone()[0] == 1:
        return True
    return False
```

```
if not table_exists('movies'):
    c.execute(''
        CREATE TABLE movies(
            movie_id INTEGER,
            name TEXT,
            release_year INTEGER,
            genre TEXT,
            rating REAL
        )
    '')
```

```
def insert_movie(movie_id, name, release_year, genre, rating):
    c.execute('' INSERT INTO movies (movie_id, name, release_year, genre, rating)\
VALUES(?, ?, ?, ?, ?) '', (movie_id, name, release_year, genre, rating))
    Con.commit()
```

```
def get_movies():
    c.execute(''SELECT * FROM movies'')
    data = []
    for row in c.fetchall():
        data.append(row)
    return data
```

```
def get_movie(movie_id):
    c.execute(''SELECT * FROM movies WHERE movie_id = {}'''.format(movie_id))
    data = []
    for row in c.fetchall():
        data.append(row)
    return data
```

```
def update_movie(movie_id, update_dict):
    valid_keys = ['name', 'release_year', 'genre', 'rating']
    for key in update_dict.keys():
        if key not in valid_keys:
            raise Exception('Invalid field name!')
    for key in update_dict.keys():
        fields = ''UPDATE movies SET {} = '{}' WHERE movie_id = {}'''\
.format(key, update_dict[key], movie_id)
        c.execute(fields)
    conn.commit()
```

Удаление

Объявим функцию для удаления информации об одном из фильмов

```
def delete_movie(movie_id):
    c.execute('DELETE FROM movies WHERE movie_id = {}'.format(movie_id))
    Con.commit()
```

Тестирование

Добавим в базу данных несколько фильмов.

```
insert_movie(1, 'Titanic', 1997, 'Drama', 7.8)
insert_movie(2, 'The Day After Tomorrow', 2004, 'Action', 6.4)
insert_movie(3, '2012', 2009, 'Action', 5.8)
insert_movie(4, 'Men in Black', 1997, 'Action', 7.3)
insert_movie(5, 'World War Z', 2013, 'Romance', 10)
```

Проверить, что они добавлены, можно так:

```
print(get_movies())
```

```
print(get_movies())
[(1, 'Titanic', 1997, 'Drama', 7.8), (2, 'The Day After Tomorrow', 2004, 'Action', 6.4), (3, '2012', 2009, 'Action', 5.8), (4, 'Men in Black', 1997, 'Action', 7.3), (1, 'Titanic', 1997, 'Drama', 7.8), (2, 'The Day After Tomorrow', 2004, 'Action', 6.4), (3, '2012', 2009, 'Action', 5.8), (4, 'Men in Black', 1997, 'Action', 7.3), (5, 'World War Z', 2013, 'Romance', 10.0), (1, 'Titanic', 1997, 'Drama', 7.8), (2, 'The Day After Tomorrow', 2004, 'Action', 6.4), (3, '2012', 2009, 'Action', 5.8), (4, 'Men in Black', 1997, 'Action', 7.3), (5, 'World War Z', 2013, 'Romance', 10.0), (1, 'Titanic', 1997, 'Drama', 7.8), (2, 'The Day After Tomorrow', 2004, 'Action', 6.4), (3, '2012', 2009, 'Action', 5.8), (4, 'Men in Black', 1997, 'Action', 7.3), (5, 'World War Z', 2013, 'Romance', 10.0)]
```

Извлечем информацию об одном фильме

```
print(*get_movie(2))
(2, 'The Day After Tomorrow', 2004, 'Action', 6.4) (2, 'The Day After Tomorrow', 2004, 'Action', 6.4)
```

Обновим последнюю запись с ID=5

```
update_movie(5, {'genre': 'Horror', 'rating': 7.0})
print(*get_movies())
(1, 'Titanic', 1997, 'Drama', 7.8) (2, 'The Day After Tomorrow', 2004, 'Action', 6.4) (3, '2012', 2009, 'Action', 5.8) (4, 'Men in Black', 1997, 'Action', 7.3) (5, 'World War Z', 2013, 'Horror', 7.0)
```

Удалим фильм с ID=3

```
delete_movie(3)
print(*get_movies())
[(1, 'Titanic', 1997, 'Drama', 7.8), (2, 'The Day After Tomorrow', 2004, 'Action', 6.4), (4, 'Men in Black', 1997, 'Action', 7.3), (5, 'World War Z', 2013, 'Horror', 7.0)]
```

Творческое задание

Создать запросы для описанной выше БД с информацией о фильмах и их реализацию с применением ORDER BY и JOIN, итоговых функций, GROUP BY, HAVING.

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. <https://pythonru.com/osnovy/sqlite-v-python> — справочная документация
2. https://www.youtube.com/watch?v=Tlc_Mi6bwOQ — видео
3. <https://www.youtube.com/watch?v=H-mSFkCY5ds> — видео
4. <https://fixmypc.ru/post/rabota-s-modulem-sqlite-i-bazoi-v-python-3/> — запросы и многое другое.
5. <https://dev-gang.ru/article/kak-sozdat-bazu-dannyh-sqlite-na-python-010lbkcv7k/> — как создать базу данных SQLite на Python, практические советы
6. <https://nuancesprog.ru/p/10647/> — основы SQLite на примере практической задачи
7. <https://1drv.ms/u/s!Ahq2O7QNuMtCiaNL-6MultXLCQQDTg?e=64BImU> — скачиваемые файлы

§4.7 Кейсы для итогового мини-проекта «Проектирование базы данных»

| | |
|---|--|
| 1 | <p>Телепрограмма</p> <p>Система должна обеспечивать хранение программы телепередач нескольких телекомпаний на неделю (по дням, времени показа) с указанием категорий телепередач: новости, спорт (по видам), художественные фильмы (по жанрам), сериалы и т.д., обеспечивать формирование совокупной программы просмотра по определенному запросу (вкусу) с указанием временных «накладок» телепередач, иметь возможность формировать список «любимых» передач (сериалов, фильмов, телепередач) для обязательного включения в ежедневный (еженедельный и т. д.) просмотр. Желательно иметь возможность хранения дополнительных сведений для художественных фильмов и сериалов: название, страна, год, режиссер, ведущие актеры, краткое содержание.</p> <p>Разработать запросы.</p> |
| 2 | <p>Скорая помощь</p> <p>Информация о вызовах бригад скорой помощи представлена следующими данными:</p> <ul style="list-style-type: none">- номер бригады;- специализация врачей бригады;- дата вызова;- время вызова;- адрес;- фамилия, имя и отчество пациента;- дата рождения;- пол;- действия. <p>Для значений специализации и действий бригады скорой помощи создать отдельные справочники, используемые при заполнении полей данных.</p> <p>Предусмотреть возможность добавления, изменения и удаления записей в справочниках.</p> <p>Необходимо предусмотреть возможность упорядочивания данных (сортировку) по следующим полям:</p> <ul style="list-style-type: none">– по специализации бригады;– по адресу;– по ФИО пациента;– по полу;– по возрасту. |

| | |
|---|---|
| | <p>Сортировка по перечисленным полям данных обеспечит сортировку по любой совокупности перечисленных полей (вложенную сортировку).</p> <p>Критериями поиска для данной системы являются: специализация бригады скорой помощи, номер бригады, дата и время вызова (указываются начало и конец периода поиска).</p> <p>Запросы:</p> <p>Для всех бригад по каждой дате определить количество вызовов данной бригады в данную дату.</p> <p>Для каждой специализации определить количество вызовов, и для каждого действия, проведенного бригадами данной специализации, подсчитать их количество.</p> <p>Для каждой специализации найти средний возраст пациентов каждого пола в отдельности, найти процент от общего количества лиц, обслуженных бригадами скорой помощи.</p> |
| 3 | <p>Турагентство</p> <p>Работники турагентства продают путевки путешествий по разным странам. В каждую страну организуются несколько маршрутов. В сведениях о каждом маршруте указывается цель путешествия (отдых, экскурсия, лечение, тур с целью покупок, обучение и др.).</p> <p>В БД должна храниться информация:</p> <ul style="list-style-type: none"> • о странах: код страны, название страны, стоимость визы (руб.); • маршрутах: код страны, код маршрута, наименование маршрута; • продажах: код маршрута, цель путешествия, цена путевки (руб.), количество проданных путевок по маршруту, дата продажи. <p>При проектировании БД необходимо учитывать следующее:</p> <p>в каждую страну организуются несколько маршрутов. Маршрут имеет отношение только к одной стране;</p> <p>маршрут участвует в нескольких продажах. Продажа связана только с одним маршрутом.</p> <p>Кроме того, следует учесть:</p> <p>по каждой стране обязательно организуется маршрут. Каждый маршрут обязательно имеет отношение к некоторой стране;</p> <p>маршрут не обязательно может участвовать в продаже (может быть невостребован). Каждая продажа обязательно связана с одним маршрутом.</p> <p>Необходимо также разработать возможные запросы к таблицам реляционной БД и формы вывода результатов.</p> |

| | |
|---|--|
| 4 | <p>Учёт литературы</p> <p>Разработайте базу данных для хранения списка учебной и методической литературы по предмету, проверьте количество экземпляров, доступных школьникам, предусмотрите возможность оформления заказа для дополнительной покупки малокомплектной литературы. Учтите различные виды литературы (учебник, статья в периодическом издании или сборнике, задачник, методические указания и т.д.). Предмет содержит ряд изучаемых тем, литература может быть как по всему предмету, так и по теме. Предусмотрите возможность получения списков литературы в соответствии с календарным графиком учебного процесса.</p> <p>Разработайте возможные запросы к таблицам реляционной БД и формы вывода результатов.</p> |
| 5 | <p>Расписание занятий в школе</p> <p>Расписание занятий в школе включает сведения о названиях классов и предметов, фамилиях учителей, обозначениях кабинетов (классов), учебной смене, дне недели, номере урока. В реальной жизни требуется также отслеживать отсутствие «накладок» в расписании («непересечение» занятий для учителей, классов и кабинетов по сменам, урокам и дням недели, отсутствие «окон» в расписании для учителей и учеников), что усложняет задачу. Содержанием запросов должен быть вывод фрагментов расписания занятий для классов, смен, учителей и т. д.</p> |
| 6 | <p>Музей</p> <p>База данных должна содержать информацию:</p> <ul style="list-style-type: none"> - об имеющихся в наличии экспонатах (наименование, автор, источник происхождения, количество экземпляров, принадлежность к тематическому разделу, история происхождения, состояние); - о музейных хранилищах; - о выставочных залах. <p>Каждое хранилище предназначено для хранения экспонатов определенного тематического направления. Содержимое выставочных залов меняется с течением времени. Запросы должны отвечать на вопросы:</p> <ul style="list-style-type: none"> - Какие экспонаты есть по данной тематике? - В хранилище или в зале находится данный экспонат в заданный момент времени? - Каково количество экспонатов по данной тематике, в данном хранилище? - Какие экспонаты требуют реставрации? |
| 7 | <p>Фитнес-клуб</p> |

| | |
|---|---|
| | <p>Разработать БД, которая должна обеспечивать ведение учета предоставления услуг, а также облегчать поиск необходимых данных, таких как справочник тренеров, преискурант, просмотр списка тренеров, предоставляющих каждый вид услуг, и др.</p> <p>Структура таблицы «Абонементы»: код абонемента, название, цена, количество посещений, количество дней.</p> <p>Структура таблицы «Клиенты»: код клиента, фамилия, имя, отчество, адрес, телефон.</p> <p>Структура таблицы «Помещения»: код помещения, название.</p> <p>Структура таблицы «Продажа абонементов»: номер карты, код клиента, абонемент, дата начала, дата окончания.</p> <p>Структура таблицы «Расписание»: № пп, дата, время начала, время окончания, услуга, помещение, сотрудник, примечание.</p> <p>Структура таблицы «Специализация сотрудников»: номер сотрудника, услуга, примечание.</p> <p>Структура таблицы «Список сотрудников»: номер сотрудника, фамилия, имя, отчество, адрес, дата рождения, оклад.</p> <p>Структура таблицы «Услуги»: код услуги, наименование услуги.</p> <p>Структура таблицы «Учёт посещений»: № пп, номер карты, № по расписанию.</p> <p>Разработайте возможные запросы к таблицам реляционной БД и формы вывода результатов.</p> |
| 8 | <p>Салон красоты</p> <p>База данных должна содержать сведения о следующих объектах:</p> <ul style="list-style-type: none"> • сотрудники – фамилия, имя, отчество, адрес, дата рождения, должность, оклад; • сведения о перемещении (должность, причина перевода, номер и дата приказа); • мастера – фамилия, специализация (список услуг, которые может оказывать), список; • материалов, находящихся на руках у мастера (наименование, количество, единица измерения, стоимость); • клиенты – фамилия, имя, отчество, телефон, дата и время заказа, заказ (услуга, стоимость услуги, мастер, оказывающий услугу). <p>Запросы:</p> <ul style="list-style-type: none"> • Дата заказа • Занятость сотрудников в день заказа • Списание материала • Удаление сотрудников |

| | |
|--|--|
| | <ul style="list-style-type: none"> • Счет клиенту • Распределение материалов по мастерам. <p>Примечание о Бизнес-правилах:</p> <ul style="list-style-type: none"> • Каждый мастер специализируется в оказании нескольких услуг. • Клиент делает только один заказ, в котором может заказать несколько услуг. • Повторные заказы этого клиента рассматриваются как заказы нового клиента. • Услугу оказывает мастер, имеющий соответствующую специализацию и свободный в указанное время. После выполнения заказа с мастера списываются использованные материалы. • Сведения о выполненных заказах сохраняются в течение года. • Сведения об уволенных сотрудниках сохраняются в течение 5 лет. • Количество исполнителей – 1. |
|--|--|

Больше кейсов на сайте: <http://access.avorut.ru/load/>

V. Web-программирование

ВСПОМИНАЕМ

Вспомним:

1. Что мы знаем о сети Интернет?
2. Какие ресурсы сети Интернет Вам знакомы?

ЧИТАЕМ

Ресурсами Всемирной паутины или WWW (World Wide Web) мы пользуемся постоянно, так как очень часто ищем информацию и при этом используем возможности распределенной системы WWW, предоставляющей доступ к связанным между собой документам, расположенным на различных компьютерах, подключенных к сети Интернет. WWW представляет совокупность цифровых источников информации, имеющих гипертекст, использующую язык разметки и поддерживающую множество протоколов Интернет.

Важно понимать разницу между WWW и Интернет. Интернет — это огромная сеть, связывающая компьютеры, а Всемирная паутина WWW формирует весь текст, изображения, аудио, видео онлайн.

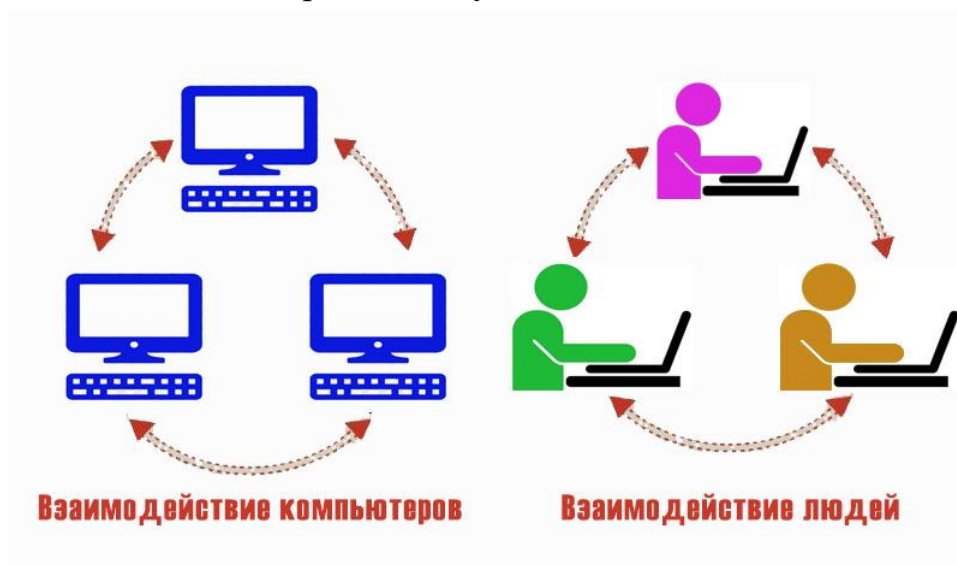


рис.139. Разница между WWW и Интернет

Большую часть этой информации можно получить через веб-сайты. Миллиарды веб-сайтов (web-sites) составляют всемирную паутину. Веб-сайт является набором веб-страниц, связанных между собой гиперссылками, единой системой навигации.

Веб-страница, содержащая код на языке HTML, обрабатывается специальными приложениями, которые отображают документ в его форматированном виде. Такие приложения, называемые Веб – браузер (Web Browser, веб-обозреватель), предоставляют пользователю удобный интерфейс

для запроса веб-страниц, их просмотра и, при необходимости, отправки введенных пользователем данных на сервер. Создано и используется огромное множество различных интересных веб-обозревателей, и каждый из них обладает своими уникальными функциями и возможностями. Самые известные – Яндекс-браузер, Google Chrome, Mozilla Firefox, Opera, Safari и т.п.

А вот для размещения своей информации, создания своего Web-сайта потребуется значительно больше знаний.

Гипертекстовые ссылки – основа работы Всемирной паутины. Любой источник информации состоит из ссылок на другие веб-страницы, рисунки, звуковые файлы, видео и т.п. Эти ссылки называются уникальным указателем ресурсов (Uniform Resource Locator – URL). Каждый файл, документ и другой информационный ресурс имеет собственный URL, что позволяет рассматривать его как адрес.

URL имеет следующий формат:

protocol://hostname/other_information

Каждый адрес имеет три составляющие:

- тип протокола URL. Например, http:// , https://, ftp://
- домен или адрес URL. Например, http://feb-web.ru/ – фундаментальная электронная библиотека “Русская литература и фольклор”
- на страницах Web-сайта feb-web.ru можно найти документы типа htm, html. Например, произведение PUSCHKINIANA, автор Межов В. И. расположена по адресу http://feb-web.ru/feb/pushkin/biblio/meg/meg1001-.htm

Протоколы

Благодаря протоколу HTTP обеспечивается работа Всемирной паутины. Протокол HTTP лежит в основе обмена данными в Интернете. HTTP — HyperText Transfer Protocol, протокол передачи гипертекста, позволяет получать (передавать) различные ресурсы, например гипертекстовые документы.

HTTP-протокол предполагает клиент-серверное взаимодействие передачи данных. Клиент, обычно это Web-браузер, формирует запрос и отправляет его на веб-сервер. Серверное программное обеспечение обрабатывает этот запрос, формирует ответ и отправляет обратно клиенту.

С помощью протокола HTTP решается задача обмена данными между пользовательским приложением и веб-сервером.

Клиенты и серверы взаимодействуют, обмениваясь одиночными сообщениями, а не потоком данных. Сообщения, отправленные клиентом, обычно веб-браузером, называются запросами, а сообщения, отправленные сервером, называются ответами.

HTTP является протоколом прикладного (верхнего 7-го) уровня в соответствии со спецификацией модели OSI. Для пересылки своих сообщений HTTP часто использует возможности другого протокола – TCP (или TLS – защищённый TCP).

Стандарт протокола можно посмотреть, например, по этой ссылке <https://www.w3.org/Protocols/rfc2616/rfc2616.html>

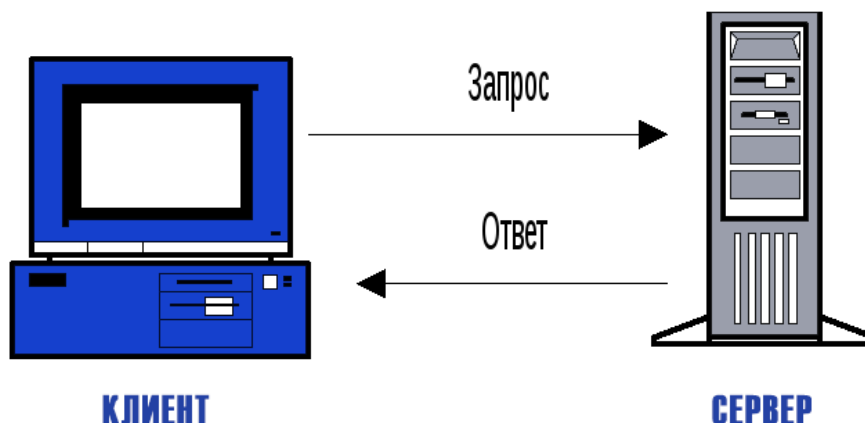


Рис.140 Взаимодействие клиент – сервер.

HTTP часто используется как протокол передачи информации для других протоколов прикладного уровня, таких как SOAP, XML-RPC и WebDAV. В таком случае говорят, что протокол HTTP используется как «транспорт».

ОБСУЖДАЕМ

Обсудите, для каких целей стали использовать протокол HTTP и какие еще протоколы передачи данных вы знаете. Почему используют не один протокол для передачи всей информации, а разные протоколы?

Обсудите, почему текст с гипертекстовыми ссылками стал активно использоваться на веб-страницах. Куда ведут эти ссылки, и как происходит взаимодействие между клиентскими и серверными ресурсами?

ЧИТАЕМ

Для расширения знаний по теме используйте электронные библиотеки и справочники. Для обзора протокола HTTP можно посмотреть <https://developer.mozilla.org/ru/docs/Web/HTTP/Overview>

§5.1 Что такое WWW

ЧИТАЕМ

Все веб-сайты различаются своими выполняемыми задачами, функциональными возможностями, назначением, техническими характеристиками и прочими параметрами. Все многообразие веб-сайтов можно условно классифицировать, но следует учитывать, что часто один сайт может быть симбиозом нескольких типов. Построить классификацию по отдельным признакам сайтов очень проблематично. Можно выделить условно категории и подкатегории, но единой классификации сайтов до сих пор не существует. Приведем пример классификации по основным шести направлениям.



Рис.141. Классификация веб – сайтов

По технологии создания

Статические сайты (HTML-сайт) написаны на языке HTML, верстаются вручную. Исторически – первый способ создания сайтов. Создают минимальную нагрузку на сервер и очень быстрые.

Динамические сайты (CMS, Content Management System, система управления контентом) – страницы выстраиваются из блоков – фрагментов кода. Для каждого блока назначается шаблон – фрагмент кода со вставками переменных. Содержимое, настройки сайта и контент хранится в базе данных

MySQL. Есть бесплатные платформы, например, WordPress, Drupal, Joomla, OpenCart и т.п., платные, например, Bitrix и самописные (разработчик настраивает платформу под определенные требования). Распространёнными языками веб-программирования являются, например, php, perl, asp и т.д.

Для таких сайтов обновление информации реализуется проще. Но сайты, сделанные на CMS, создают дополнительную нагрузку на сервера. Пока динамические сайты являются наиболее востребованным видом технической реализации сайтов.

Сайт-конструктор — это платформа с набором готовых шаблонов и инструментов, например, Ucoz, «Тильда» и т.п. Есть бесплатные и платные с расширенным набором инструментов. Позволяют неспециалисту самостоятельно создать простой проект (влог, визитку и т.п.) Но есть ограничения: невозможно изменить код, настройки минимальны, невозможен перенос на другую платформу.

Flash-сайты (Macromedia Flash, Adobe Flash) имеют возможность создания оригинальных интерактивных веб-страниц со звуком и анимацией. Реализация требует немало времени, усилий и финансовых затрат. Созданные на Flash сайты, имеют, как правило, большой вес и долго загружаются. Отличаются сложностью в управлении содержанием. Такие сайты плохо индексируются поисковиками.

По содержанию

Личные (персональные сайты) – сайтом владеет один человек. Таких сайтов много, и они имеют широкий спектр назначений.

Сайты коммерческих организаций – можно выделить несколько подвидов:

- интернет-магазин – реклама товаров и услуг, взаимодействие с покупателем, прием и подтверждение заказов, информация о продавце. Для каждого клиента предусмотрен личный кабинет с отображением истории покупок;
- сайт-визитка — это дополнительный сайт-инструмент для маркетинга. Обычно состоит из 5-7 страниц, которые содержат небольшую, только главную информацию о фирме;
- Landing Page – обычно это одна страница, которая имеет одну цель – заставить посетителя произвести действие: купить продукцию, записаться на курс и т.п. Такой сайт должен ярко рассказать о товаре или услуге, вызвать желание совершить покупку, собрать информацию о покупателе и его потребностях.

Сайты некоммерческих организаций – сайты общественных организаций, пенсионные фонды, научно-исследовательские институты и т.п.

Для двух категорий выделены специальные домены: gov – государственные службы, edu – образовательные учреждения.

По объему информации и виду решаемых задач

Простые сайты – сайты-визитки, портфолио, домашние странички и т.п., содержат небольшой объем информации.

Тематические сайты – посвящены определенным темам, например, исторического содержания, информации о странах и городах. Могут быть интересны узкому кругу пользователей.

Многофункциональные сайты (порталы) – сайты с большим объемом информации разных направлений: новости, почта, блоги, видео, прогнозы погоды и т.п. Например, крупнейшие порталы: Яндекс, Мэйл.ру, Google.

По типу информационных возможностей, предоставляемых пользователям Интернета

Сайты информационные, предоставляющие контент — это различные новостные сайты, в том числе информационные городские, региональные, различные справочники и энциклопедии и т.п.

Сайты для онлайн-общения и контактов — это социальные сети, блоги, форумы, чаты, доски объявлений, сайты, организующие общение между людьми.

Сайты электронной коммерции – список подобных сайтов очень велик. К этой категории относятся Интернет-магазины, сайт-витрина, сайт-портфолио, агрегаторы, электронные услуги, платежные системы, сайты банков с услугами по обслуживанию клиентов, сайты, обеспечивающие доступ к фондовому рынку и т.п.

Сайты, предоставляющие онлайн-сервисы – сюда можно отнести сервисы для хостинга для сайтов, форумов, блогов, чатов, а также сайты, предоставляющие бесплатную электронную почту, трансляторы, онлайн-редакторы документов, рисунков, видео. Сюда же можно отнести сайты типа WEB 2.0.

Поисковые сайты – специальные поисковые системы для поиска необходимой информации по запросам пользователей. Известные на весь мир поисковики Яндекс, Рамблер, Google, Bing отличаются скоростью и возможностями поиска.

Также приведем примеры других поисковых систем со своими специфическими характеристиками:

- Анонимные поисковые системы, например, Creative Commons Search, StartPage.

- Поисковики видео и других полезных материалов, например, Giphy, SlideShare, Vimeo.
- Поисковые сервисы, которые предустановлены в популярных браузерах, например, DuckDuckGo.
- Поиск ответов на вопросы от реальных людей, например, Quora, WolframAlpha.
- Boardreader – разработан по типу доски объявлений и выполняет поиск на форумах по всему миру.
- Dogpile – агрегатор для сбора найденных данных по трем популярным поисковым системам Google, Yandex и Yahoo.

Неестественные сайты Black SEO

Сайты Дорвей (Doorway) появились в период обострения конкуренции за продвижение сайтов в рейтинге поисковых сервисов. Цель создания подобных сайтов – быстрое продвижение своего сайта или сайта заказчика. Это небольшие сайты, которые ориентированы на несколько ключевых слов, оптимизированных под определенные запросы, и зачастую нацелены на перекидывание пользователя на главный сайт заказчика.

Сателлиты (satellite) – вспомогательные сайты, не являются самодостаточными, а служат для продвижения главного сайта.

Сайты MFA (Made For AdSense) – сайты, сделанные для заработка на контекстной рекламе AdSense. Недобросовестное создание подобных сайтов, копирование информации с других источников создает информационный мусор в интернете.

Сайты MFS (Made For Sale). Sale — это популярная биржа ссылок, т. е. подобные сайты предназначены для продажи ссылок. В отличие от MFA-сайтов, для этих сайтов не нужно привлекать на сайт посетителей, которые кликали бы на контекстную рекламу, достаточно нарастить сайту показатели ТИЦ и PR, чтобы была возможность с него продавать ссылки.

Пояснение:

ТИЦ (тематический индекс цитирования) – показатель, используемый поисковой системой «Яндекс» для определения авторитетности и значимости сайта на основании его ссылочной массы. Именно количество и качество обратных ссылок определяет ТИЦ ресурса.

PR (PageRank) – показатель, используемый поисковой системой Google для определения все той же авторитетности и значимости сайтов с опорой на качество ссылочной массы. Так же как и в случае с ТИЦ, на PR влияет количество и качество обратных ссылок. (<https://www.rookee.ru/learn/tic-i-pr-sajta-chto-eh-to/>)

По доступности

В этом делении все просто. Сейчас на многих сайтах просят регистрироваться, и после авторизации вы имеете доступ к ресурсам сайта. Есть закрытые сайты, например, сайты предприятий, банков, образовательных и т.п., на которые вход доступен только разрешенному кругу пользователей. И продолжают существовать сайты не требующие авторизации, например, новостные, исторические и т.п.

Существует и смешанный доступ, когда часть информации возможно получить без авторизации, а другую часть – после авторизации, причем доступ может быть разного уровня.

ОБСУЖДАЕМ

Обсудите, с какими типами веб-сайтов вам последнее время чаще приходится работать. Для каких целей эти сайты используются?

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Классифицируйте по предложенным выше принципам три наиболее посещаемых вами сайта. Результаты разместите в тетради, заполнив таблицу.

| Название (ссылка) | Технология создания | Принадлежность по типу | Выполняемые задачи | Возможности пользователя |
|----------------------|------------------------|---------------------------|-----------------------|-----------------------------|
| | | | | |
| | | | | |

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Перечислите типы сайтов согласно классификации, приведенной выше.
3. Встречались ли вам сайты, которые не подходят ни под один тип классификации, приведенной выше?
3. Приведите примеры сайтов следующих типов: поисковые системы, почтовые системы, форумы, хостинги, медиагруппы, доски объявлений, социальные сети.

§5.2 Знакомство с популярными CMS

ВСПОМИНАЕМ

Вспомним:

1. Какие технологии используемые при создании сайтов вы знаете?
2. Какие виды интернет-сервисов вам известны?

Сайты можно писать “руками”, т.е. в текстовом редакторе, используя HTML, CSS, JavaScript. Таким образом можно верстать небольшие сайты. Но последнее время все чаще обращаются к конструкторам CMS (Content Management System), где верстка сайта происходит проще, быстрее, имеются разнообразные возможности, да и администрирование таких сайтов проще. Согласно опросам и аналитическим исследованиям, на время написания этого учебника самыми популярными CMS являются (результаты за март 2021г. <https://itrack.ru/research/cmsrate/>):

Бесплатные CMS:

- WordPress ru.wordpress.org/
- Joomla www.joomla.org/
- Google sites – www.sites.google.com
- OpenCart www.opencart.com/
- Drupal www.drupal.org
- Wix www.wix.com/
- Evolution CMS evo.im/
- MODX Revolution modx.com/
- прочие

CMS с платными сервисами:

- 1С-Битрикс <https://www.1c-bitrix.ru/>
- CMS.S3 <https://megagroup.ru/cms?referer=ag9483>
- Tilda <https://tilda.cc/ru/>
- DataLife Engine <https://itrack.ru/rating-cms/datalife-engine/>
- Shop-Script <https://www.shop-script.ru/>
- Adobe Muse <https://helpx.adobe.com/ru/muse/get-started.html>
- UMI.CMS <https://www.umi-cms.ru/>
- NetCat <https://netcat.ru/>

Приведем краткое описание популярных систем управления контентом для создания сайта. Следует иметь ввиду, что во многих бесплатных программах бывают платные дополнения, которые расширяют возможности сервисов.

1. WordPress – самая популярная CMS с большой историей. На ее основе работает огромное количество новостных сайтов, в том числе достаточно известных. WordPress имеет открытый исходный код и благодаря этому гибко настраивается, используя огромное количество готовых тем и плагинов. По статистике самая взламываемая CMS. Подходит для создания промо-сайта, B2B кабинета, корпоративного сайта, сайта с каталогом, интернет-магазина, портала.

2. Joomla – достаточно понятная, простая CMS и предоставляет большие возможности. Даже незнакомые с программированием пользователи могут самостоятельно создать удобный сайт и получить систему управления к нему. Подходит для создания промо-сайта, корпоративного сайта, сайта с каталогом, портала.

3. OpenCart — это платформа для создания интернет-магазина. Имеются удобные шаблоны для создания полезных сервисов: регистрация, личный кабинет, формы с валидацией, отзывы и т.д. Имеется много дополнений которые помогают настроить интернет – магазин разного уровня сложности. Подходит для создания корпоративного сайта, сайта с каталогом, интернет-магазина.

4. Drupal – известная и с большой историей CMS. За время существования этой системы управления сайтами создано много документации, руководство и информации о ее работе, но все равно администрирование этой системы требует специальной подготовки. Подходит для создания промо-сайта, корпоративного сайта, сайта с каталогом, интернет-магазина.

5. Wix – бесплатный HTML5-редактор с возможностью платного расширения функций. Поддерживает функцию drag-n-drop (возможность перетаскивать виджеты). Имеется большое количество виджетов, анимации, шрифтов, фонов и т.д. и позволяет создавать яркие сайты с минимум навыков. Имеется Wix ADI – технология автоматической сборки сайтов на основе предоставленной информации. Подойдет для создания сайтов малого и среднего размеров, промо-сайта, корпоративного сайта.

6. Evolution CMS – система с открытым исходным кодом. Имеется достаточно много информации и системе и развитое сообщество разработчиков. Разработчику сайта для этой системы нужны навыки работы в HTML. Подходит для создания: Интернет-магазина, каталога товаров, сайта компании и прочие.

7. MODX Revolution — распространяется бесплатно по лицензии GPL с открытым исходным программным кодом (Open Source). Система рассчитана на разработчиков, но имеет дружелюбный интерфейс, в административной панели можно работать с drag and drop и это удобно как разработчикам, так и пользователям. Подходит для создания промо-сайта, корпоративного сайта, сайта с каталогом, Интернет-магазина.

ОБСУЖДАЕМ

Обсудите можете ли вы определить, изучая известные веб – сайты, какие популярные CMS использованы для их создания и работы. А может быть не использованы совсем.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Осуществите в сети интернет-поиск информации о том, какие CMS использовались при создании популярных сайтов российского сегмента сети интернет (Рунета). Найденную информацию зафиксируйте в тетради в виде таблицы так, чтобы было проанализировано не менее десяти сайтов.

| Название сайта | URL | Область | CMS |
|----------------|-----|---------|-----|
| | | | |
| | | | |

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Что означает CMS и для каких целей используют эти системы?
2. Какая CMS, по вашему мнению, сегодня самая рекомендуемая непросвещенному в программировании пользователю для создания личного сайта? Обоснуйте ответ.

§5.3 Знакомство с HTML

ВСПОМИНАЕМ

Вспомним:

1. В чём существенные различия между основными CMS?
2. Необходимо ли владеть навыками HTML при создании сайта с использованием CMS?

ЧИТАЕМ

HTML (HyperText Markup Language) — язык разметки (маркировки) гипертекста.

В 1986 году Международная организация по стандартизации (ISO) приняла стандарт под названием SGML (Standard Generalized Markup Language), и этот год принято считать годом рождения этого языка. HTML — это наследник SGML.

В 1991 году появилась потребность в передаче гипертекстовой информации по сети Интернет и на основе SGML был создан новый язык — HTML. Первый в мире веб-сайт создал Тим Бернерс-Ли 6 августа 1991 года, разместил описание проекта World Wide Web и написал первую в истории веб-страничку <http://info.cern.ch/hypertext/WWW/TheProject.html>

К 2012-му году практически все современные браузеры в мире, начинают понимать язык HTML5. Спецификация HTML5 была опубликована 28 октября 2014, и с этого дня официально рекомендуется использовать HTML5.

На данный момент работа над языком HTML5 продолжается, создаются новые теги и технологии, всё это добавляется в спецификацию.

HTML 5 сделала синтаксис более строгим по сравнению с предыдущей, улучшилась поддержка мультимедиа-технологий, появились много новых структурных элементов и код стал более понятным, исключили еще часть устаревших тегов, больше внимания уделяется поддержке скриптов, например, javascript

HTML язык состоит из тегов — это своеобразные команды, которые преобразовываются в визуальные объекты в браузере пользователя. Все браузеры понимают html и могут представлять веб-странички для пользователей в понятном виде. Страницы сайта являются html-кодом, и этот код браузер переводит в понятный вид для пользователя.

ОБСУЖДАЕМ

Чем отличается документ, написанный в текстовом редакторе, например, Word, от документа, написанного с помощью HTML кода?

ГОТОВИМСЯ К РАБОТЕ НА ПК

Создание простой странички следует начать с создания файла в формате .html, например, index.html.

Для редактирования этого файла можно использовать любой текстовый редактор. Мы будем показывать примеры в редакторе NotePad++, который каждый может установить по ссылке <https://notepad-plus-plus.org>

Можно использовать другие бесплатные и платные IDE (Integrated Development Environment) – интегрированная, единая среда разработки для работы.

Самые популярные IDE:

- Brackets – бесплатный редактор, обеспечит большее удобство работы. Brackets ориентирован на работу с HTML, CSS и JavaScript. <https://brackets.download/>
- JetBrains IntelliJ IDEA, доступна в двух версиях: Community Edition — это свободная версия под лицензией Apache 2.0? Для JVM и Android-разработки, Ultimate Edition – коммерческая версия, для корпоративной и веб-разработки. <https://www.jetbrains.com/ru-ru/idea/>
- JetBrains Webstorm – коммерческая IDE, бесплатная пробная версия на 30 дней. <https://www.jetbrains.com/ru-ru/webstorm/>
- Microsoft Visual Studio Code – бесплатная IDE <https://code.visualstudio.com>

- Atom — бесплатный текстовый редактор с открытым исходным кодом для Linux, macOS, Windows с поддержкой плагинов, с возможностью установки огромного количества расширений, в том числе и платных. <https://atom.io/>

- Sublime Text 3 — имеет премиум-версию и бесплатную портативную версию. <http://www.sublimetext.com/3>

ЧИТАЕМ

Каждый HTML-документ имеет команды, называемые теги. Тег (tag — именованная метка) является основной составляющей HTML.

Обычно используются парные теги — открывающий начальный, например, `<tag>`, и закрывающий, конечный, например, `</tag>`. Все, что внутри, называется содержимым.

Если открыто несколько тегов, то закрываются они строго в обратном порядке.

Пример парного тега: `<p> содержимое </p>`

Возможно также применение одиночных тегов.

Пример одиночного тега: `содержимое
`

Для тегов могут быть использованы атрибуты — дополнительные характеристики тега или метатега. Это специальные слова, которые используются внутри начального тега для управления поведением элемента. HTML-атрибуты сообщают браузеру, как должен отображаться элемент на странице. Значение атрибута заключается в кавычки. Существуют глобальные атрибуты, которые могут быть использованы для любого HTML-элемента. В то же время могут существовать атрибуты, которые применяются только к конкретному тегу.

Если для тега не добавлен какой-либо допустимый атрибут, то в этом случае будет подставляться значение по умолчанию.

`<tag атрибут1="значение1" атрибут2="значение2">`

Пример тега с атрибутом: `<p align="center"> содержимое </p>`

Теги можно писать в нижнем и верхнем регистре, и произвольно смешивать, но это будет не красиво и не рекомендуется.

Текст HTML-документа начинается с декларации типа документа, или «доктайпа». При использовании HTML5 пишем `<!DOCTYPE html>`. Это необходимо, чтобы браузер определил версию HTML.

Любая веб-страница имеет структуру:

```
<html> -начало страницы
  <head> -заголовок страницы
  ....
  </head>
```



```
<body>    -тело страницы
.....
</body>
</html>
```

Все последующие практические работы будут дополнять предыдущую так, чтобы получилась небольшая учебная веб-страничка. Для заполнения этой странички мы будем использовать фрагменты тексты известных книг по WEB-программированию.

Существует множество справочных материалов по HTML. Некоторые из них вы можете найти в конце урока в разделе «Дополнительные информационные ресурсы».

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №42

Откройте редактор Notepad++ и создайте документ, в котором есть все элементы основной структуры страницы. Сохраните этот документ с расширением HTML, а затем откройте файл в браузере, например Yandex или Chrome.

Внимание! Пояснения после знака // или /* писать не надо!

```
<!DOCTYPE html>    /* тип документа, определяет версию и
соответствующий DTD-файл в Интернете*/
<html>              /*начало html документа */
  <head>             /*заголовок страницы*/
    <title> создание страницы на HTML </title> /*содержание
заголовка*/
  </head>            /*окончание заголовка*/
  <body>             /*начало тела страницы. Основная часть документа*/
    <h1> страница с использованием тегов HTML </h1>
/*содержание. <h1> – перевод текста в заголовок 1 уровня*/
  </body>            /*окончание тела страницы*/
</html>             /*окончание html документа */
```

Откройте файл Задания 1 в браузере. Должна появиться следующая картинка

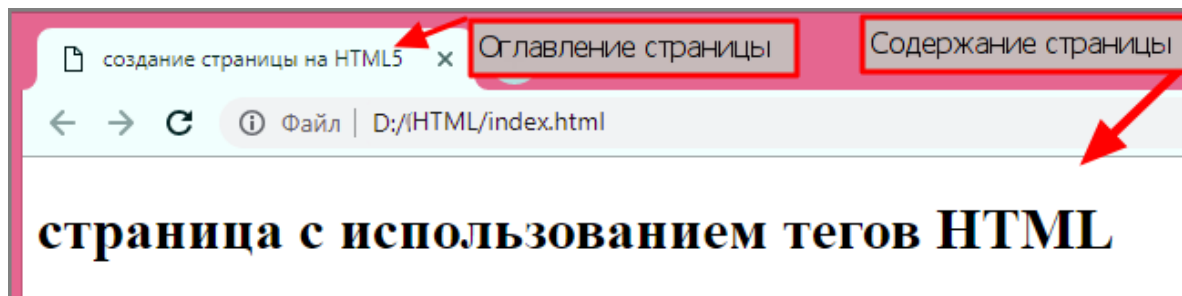


Рис.142. Результат создания веб – страницы основной структуры

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №43

Добавьте несколько строк в первый документ для установки кодировки страницы и откройте этот документ в браузере.

```
<!DOCTYPE html>
<html lang="ru"> /* начало страницы, установка языка контента
*/
  <head>
    <meta charset= "UTF-8"> /* установка кодировки страницы*/
    <title> Создание страницы на HTML </title>
    <meta name = "description" content="Примеры HTML разметки">
    / *описание страницы для поисковых систем*/
    <meta name="keywords" content="html,lesson"> /*ключевые
      слова для поиска*/
  </head>
  <body>
    <h1> Страница с использованием тегов HTML5 </h1>
  </body>
</html>
```

Что изменилось?

Строки, которые ввели, являются служебной информацией и не отображаются в браузере.

В браузере можно посмотреть код страницы, для этого используйте сочетание клавиш Ctrl+U.

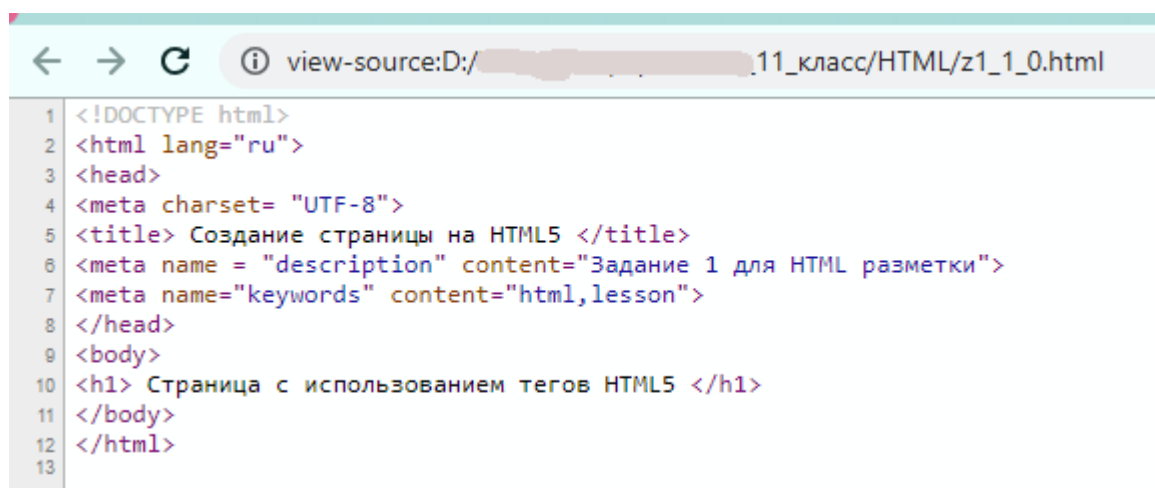


Рис.3.2. Просмотр кода страницы, быстрые клавиши Ctrl+U

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №44

Создайте страницу с использованием тегов <p>... </p> – разделение абзацев, одиночного тега
 – перенос на новую строку, выделение важного слова – одиночный тег .

Получите вид страницы как на рисунке.

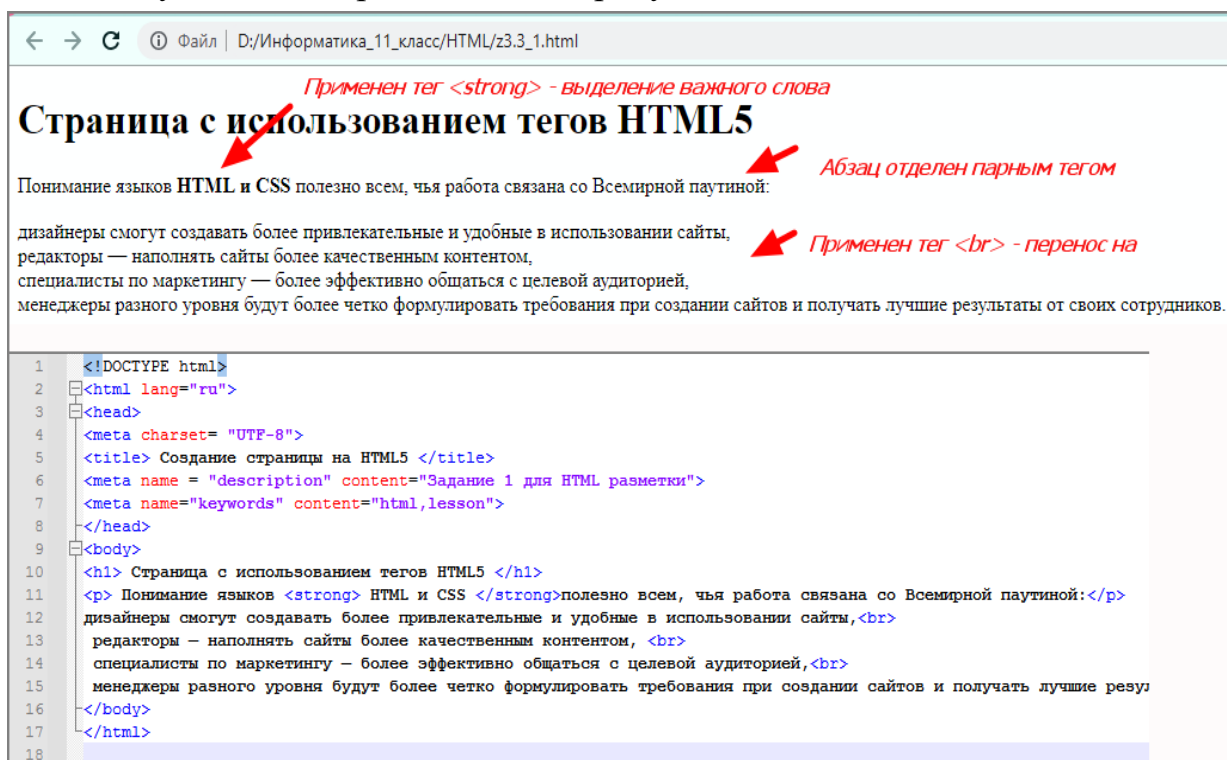


Рис.143. Использование тегов <p> и

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №45

Создайте страницу с использованием заголовков второго и четвертого уровней, используйте тег <p> для разделения двух абзацев, выделите два важных слова с помощью тега .

ОБСУЖДАЕМ

В документе помимо смыслового наполнения должна содержаться некоторая метаинформация, позволяющая определить его структуру и внешнее представление. Такая метаинформация называется разметкой документа.

Обсудите, удобно ли с помощью языка разметки HTML:

- выделять смысловые части (логические элементы) документа и связей между ними?
- указывать действия, которые должны быть осуществлены с этими элементами?

ЧИТАЕМ

В гипертекстовом документе можно создавать списки двух типов: маркированные и нумерованные.

- ТЕКСТ1
- ТЕКСТ2

1. ТЕКСТ3
2. ТЕКСТ4

Для создания списков используется двойной тег `` (неупорядоченный список) или `` (упорядоченный / нумерованный список). Потом следуют двойные теги `` (элементы списка)

Например, для создания маркированного списка осуществляется запись вида:

```
<ul>
  <li> ТЕКСТ1 </li>
  <li> ТЕКСТ2</li>
</ul>
```

При нумерованном списке она выглядит так:

```
<ol>
  <li> ТЕКСТ3 </li>
  <li> ТЕКСТ4</li>
</ol>
```

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №46

В документ, созданный в практической работе 44, вставьте два списка: нумерованный и нenumерованный список. Получите документ по примеру, приведенному на рисунке 144.

Страница с использованием тегов HTML5

Понимание языков **HTML** и **CSS** полезно всем, чья работа связана со Всемирной паутиной:

дизайнеры смогут создавать более привлекательные и удобные в использовании сайты,
редакторы — наполнять сайты более качественным контентом,
специалисты по маркетингу — более эффективно общаться с целевой аудиторией,
менеджеры разного уровня будут более четко формулировать требования при создании сайтов
и получать лучшие результаты от своих сотрудников.

Основные элементы HTML:

1. Форматирование текста – выделение курсивом, жирным шрифтом, подчеркивание, размер кегля, нумерованные/маркированные списки.
 2. Текстовые блоки – заголовки уровней H1-H6, абзацы, перенос на новую строку.
 3. Таблицы – любое количество строк, столбцов, фиксированная высота, ширина, заголовки.
 4. Вставка объектов – изображения, звуковые, текстовые, видеофайлы и т.д.
 5. Гиперссылки – на файл изображения, прайс-листа, страницу, на которую ссылается пункт
- Есть атрибуты открытия документа в текущем или новом окне.

Преимущества сайтов на HTML-коде:

- меньший "вес" по сравнению с CMS платформой;
- небольшой расход ресурсов сервера;
- высокий уровень безопасности данных, малая уязвимость к взлому;
- высокая стабильность;

Пройти бесплатно курсы можно у известных компаний

Рис.144. Страница с нумерованным и не нумерованным списком

ЧИТАЕМ

Ссылки – один из важных элементов веб-страницы. Ссылки могут быть на другой документ, сайт, на картинку, т.е. ссылки могут быть на любые объекты.

Для создания ссылки на веб-страницы используют двойной тег `<a>`. Текст, расположенный между тегами `<a>` и ``, по умолчанию становится синего цвета и подчёркивается. Цвет ссылки можно менять с помощью атрибутов.

Содержимое элемента `<a>` является текстом ссылки. Можно добавлять атрибуты. Атрибут `href` определяет адрес документа, на который следует перейти.

веб-адрес может быть:

- абсолютный, т.е. полный путь на ресурс;
- относительным, т.е. ссылка на документ внутри того же веб-сайта, при этом не пишется `https://www`

```
<a href="веб-адрес"> текст ссылки </a>
```

Например,

```
<a href="https://www.sites.ru/imag/ files.jpg "> фотография
</a> //полный путь к файлу files.jpg
<a href="imag/files.jpg"> фотография </a> //файл находится на
в папке imag этого сайта
<a href=" files.jpg"> фотография </a> // файл находится на в
папке этого сайта
```

Другие атрибуты

Accesskey – активация ссылки с помощью комбинации клавиш.

Coords – устанавливает координаты активной области.

Download – предлагает скачать указанный по ссылке файл.

hreflang – идентифицирует язык текста по ссылке.

name – устанавливает имя якоря внутри документа.

shape – задает форму активной области ссылки для изображений.

tabindex – определяет последовательность перехода между ссылками при нажатии на кнопку Tab.

target – имя окна или фрейма, куда браузер будет загружать документ.

title – добавляет всплывающую подсказку к тексту ссылки.

type – указывает MIME-тип документа, на который ведёт ссылка.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №47

Добавьте на свою страничку ссылки на бесплатные курсы известных организаций.

Например,

```
<body>
  <p><a href="practicum.yandex.ru"> HTML от Яндекс</a></p>
  <p><a href="stepik.org">Stepik</a></p>
</body>
```

Получите приблизительно следующий результат:

Пройти бесплатно курсы можно у известных компаний

[HTML от Яндекс](https://practicum.yandex.ru)

[Stepik](https://stepik.org)

Рис.145. Результат размещения ссылок на внешние ресурсы

ЧИТАЕМ

Рисунки и графика – очень важная составляющая для веб-страниц, они значительно улучшают и украшают, делают содержание информативнее. Это

могут быть графические файлы разных форматов растровой графики с расширением JPG (Joint Photographic Experts Group), PNG (Portable Network Graphics), GIF (Graphics Interchange Format) и векторной графики SVG (Scalable Vector Graphics).

Файлы формата JPG, PNG имеют качество сжатия выше, чем у GIF, они меньшего размера и хорошо загружаются на страницу. PNG и GIF позволяет вставлять изображения с прозрачным фоном, в GIF формате можно публиковать анимированные изображения.

Для публикации изображений используется тег `` с атрибутом `src=` [путь к нужному графическому файлу].

Для подписи картинки альтернативным текстом используется тег атрибут `alt`.

| формат | палитра | кол-во цветов | анимация | прозрачность | объем файла | качество изображения |
|--------|---------|---------------|----------|--------------|-------------|----------------------|
| PNG-8 | 8 бит | 256 | нет | да | | |
| PNG-24 | 24 бит | 16,7 млн. | нет | да | наибольший | лучшее |
| JPG | 24 бита | 16,7 млн. | нет | нет | | |
| GIF | | 2-256 | да | да | средний | |

Рис.146. Сравнительная таблица форматов графических файлов.

Изображения типа SVG (Scalable Vector Graphics — «масштабируемая векторная графика») предназначены для описания векторной и смешанной (векторно-растровой) двумерной графики, поддерживают анимацию и интерактивность. Это достаточно новый способ представления изображения на веб-страницах. Первая версия вышла в 2001 г.

Файлы типа SVG создают с помощью математического описания геометрических примитивов (линий, кругов, эллипсов, прямоугольников, кривых и так далее). В отличие от других изображений, SVG можно открывать средствами HTML или CSS и менять в любом редакторе кода. SVG — развивающийся формат с открытым кодом. Его использование может быть проблемным в некоторых браузерах, особенно старых версий.

Для размещения SVG файла на веб-страницу можно использовать те же теги, как для векторной графики, но в этом случае файл теряет свои преимущества. Для получения полных возможностей редактирования “на лету” вставляют svg-код прямо в html-файл.

Например,

```
<svg>
  <rect x="20" y="30" width="300" height="100" />
```


</svg>

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №48

Подготовьте файлы для размещения на своей страничке.

1. Обращение к файлу, который находится в одной папке с index.html.

В папку проекта скопируйте графический файл с расширением jpg. Дайте ему имя logo. Разместите файл logo.jpg на веб-странице, который находится в текущей папке проекта, ширина изображения устанавливается 200, высота 100 пикселей. Для получения изображения введите код HTML.

```

```

2. Обращение к файлу logo.jpg, который находится в папке img.

Создайте папку img внутри рабочей папки веб-страницы. В эту папку скопируйте файл logo.jpg. Для получения изображения введите код HTML.

```

```

3. Обращение к файлу с указанием полного локального пути.

Файл находится на диске D: в папке Информатика_11_класс/ Для получения изображения введите код HTML.

```

```

4. Обращение к файлу с указанием полного интернет-пути.

Найдите подходящую картинку в Интернете, скопируйте веб-адрес к этому файлу. Вставьте скопированный веб-адрес после атрибута src.

```

```

Выравнивание изображения посередине, слева, справа, по центру делается с помощью атрибута align=

```
<br>
<br>
```

Очень важным для оформления страницы является наличие фона. Для создания фона из одного сплошного цвета используется атрибут bgcolor со значением цвета.

Например,

```
<body bgcolor=«green»>
  <p align=«center»> на зеленом фоне </p>
</body>
```


Для использования изображения в качестве фона используется атрибут background с адресом фонового рисунка.

```
<body background=«../img/picture.gif»>  
  <p align=«center»> текст на фоне картинки</p>  
</body>
```

ЧИТАЕМ

Для создания таблицы используют тег <table>, он служит контейнером для элементов, формирующих содержимое таблицы. Формирование таблицы происходит последовательно, сначала строка <tr>, а внутри с помощью тега <td> формируется каждая ячейка. Для выделения заголовка таблицы используется <th>

Пример кода:

```
<p> Форматы изображений для WEB страниц </p>  
<table border="1">  
  <tr>  
    <th>Фотмат PNG -8</th>  
    <th>Фотмат PNG -24</th>  
  </tr>  
  <tr>  
    <td>Формат GPG</td>  
    <td>Формат SVG</td>  
  </tr>  
</table>
```



Рис.147. Пример кода создания таблицы

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №49

Добавьте в свой HTML документ таблицу следующего вида.

Форматы изображений для WEB страниц

| | |
|---------------|----------------|
| Фотмат PNG -8 | Фотмат PNG -24 |
| Формат GPG | Формат SVG |

Рис.148. Вид таблицы

Найдите ошибку в отображаемой таблице, исправьте её в коде.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. HTML-документ имеет определенную структуру. Какие три области документа можно выделить? Какие теги используются для определения каждой области?

2. Какой синтаксис имеют двойные теги и одиночные теги?
3. Что такое атрибуты тегов и для чего их используют?
4. Какие теги и каким образом используются для создания списков?

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. История развития HTML https://vertex-academy.com/tutorials/ru/html_history/
2. HTML5 Грамматика и ассоциированные API для HTML и XHTML <http://spec.piraruco.com/html5/>
3. <http://www.webremeslo.ru/spravka/spravka.html>
4. https://schoolsw3.com/tags/ref_attributes.php
5. Книга Джона Дакетта <https://monster-book.com/html-i-css-razrabotka-i-dizayn-veb-saytov>
6. Книги и учебники по веб-дизайну и веб-программированию. <https://html5beginner.github.io/books/index.html>

§5.4 Создание форм средствами HTML

ВСПОМИНАЕМ

Вспомним:

1. Что такое тэги? Чем отличаются парные тэги от одиночных?
2. Какие ещё элементы оформления гипертекстового документа могут нам понадобиться при оформлении?

ЧИТАЕМ

Важным элементом современных веб-сайтов являются формы. HTML формы используются для сбора пользовательских данных. Введенные данные в форму могут быть в дальнейшем отправлены на сервер для обработки.

Задаёт форму двойной тег `<form>`.

Элементами формы могут быть различные элементы ввода – текстовые поля, переключатели, кнопки отправки, флажки и т.п.

`<input type="text">` однострочное поле ввода текста

`<input type="submit">` кнопка для отправки формы

`<input type="radio">` переключатель (выбор одного из многих вариантов)

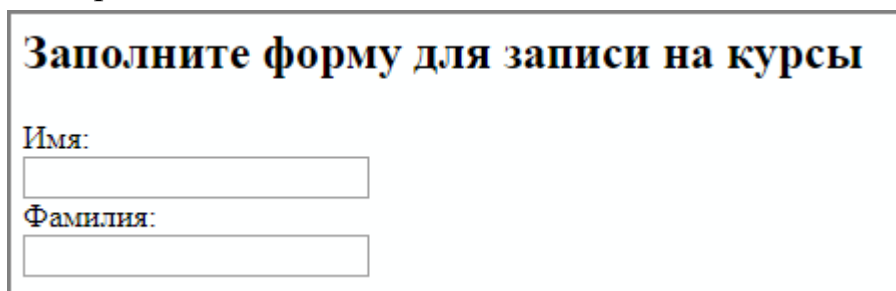
ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №50

Вот пример создания текстовой формы:

```
<p> <h2>Заполните форму для записи на курсы </h2></p>
<form>
  <label for="fname">Имя:</label><br>
  <input type="text" id="fname" name="fname"><br>
  <label for="lname">Фамилия:</label><br>
  <input type="text" id="lname" name="lname">
</form>
```

Создайте новый HTML-документ так, чтобы в браузере отражалось следующее содержание:



Заполните форму для записи на курсы

Имя:

Фамилия:

Рис.149. Результат просмотра формы

ЧИТАЕМ

Важным элементом любой формы является атрибут `action`. Он определяет действие, которое должно быть выполнено при отправке формы.

Обычно данные формы отправляются на страницу на сервере, когда пользователь нажимает на кнопку "Отправить".

В приведенном выше примере данные формы отправляются на страницу на сервере под названием `"/action_page.php"`. Эта страница содержит серверный скрипт, который обрабатывает данные формы:

```
<form action="/action_page.php">
```

Если атрибут `action` опущен, действие устанавливается на текущую страницу.

Кнопка `submit`

`<input type="submit">` определяет кнопку для представления в виде данных для обработчика форм.

Обработчик форм обычно представляет собой страницу на сервере со скриптом для обработки входных данных.

Обработчик форм задается в окне формы атрибутом `action`.

Именно так приведенный выше HTML-код будет отображаться в браузере:

Имя:

Андрей

Фамилия:

Щипунов

Отправить

Безусловно, процесс создания формы не обеспечивает фактическое взаимодействие с веб-страницей. Это так называемый frontend – процесс создания внешнего вида и пользовательского интерфейса. А для того, чтобы обрабатывать действия с формой, нужно создание соответствующей программы, иначе говоря, backend.

В рамках этого урока мы пока останемся в границах frontend.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №51

Создайте новый HTML-документ, а в нём анкету, скажем, регистрации на школьный фестиваль проектов. Используйте справочные материалы, чтобы реализовать не только текстовые блоки, но и множественный выбор, список и другие типы форм. Всего необходимо создать 7-8 полей и кнопки «Отправить» и «Сбросить», которые будут приводиться в действие файлами «feedback.php» и «clear.php».

§5.5 Создание форм и основы CSS

ВСПОМИНАЕМ

Вспомним:

1. Что такое frontend?
2. Для чего используются формы в HTML-документе?

ЧИТАЕМ

Для лучшего восприятия текста, особенно большого размера, необходимо его визуальное разделение. Для этого применяют выделение с помощью заголовков, абзацев, цветовое выделение, табличное, стили, цитаты, прямая речь и т.п. Иногда требуется выделить часть текста или целый блок текста, чтобы применить к нему форматирование. Во времена использования HTML4 тексты на блоки разделялись с помощью тега <div>...</div> и . Сейчас их называют несемантическими элементами, т.к. с точки зрения

поисковых систем и браузеров все элементы страницы состоят из однотипных блоков. С началом применения HTML5 начали применять семантическую верстку, т.е. верстка стала влиять на смысловое предназначение каждого блока и логическую структуру документа. Семантическая структура помогает лучше определять, какой сейчас блок, а пользователю понимать, о чём идёт речь. Если применяется инструмент зачитывания страниц, то семантическая структура помогает пользователям взаимодействовать с сайтом, например, адекватно понимать, где заголовок, где основной блок, боковая панель и т.д.

Существуют рекомендации разметки страницы на блоки с точки зрения семантики:

1. Использовать теги `<header>`, `<main>`, `<footer>` для размещения крупных смысловых блоков на каждой странице сайта.
2. Использовать теги `<nav>`, `<section>`, `<article>`, `<aside>` для размещения крупных смысловых разделов в блоках.
3. Использовать теги `<h1>`-`<h6>` для выделения заголовков всего документа и заголовки смысловых разделов.
4. Размещение мелких элементов в смысловых разделах, таких как списки, таблицы, демо-материалы, параграфы и переносы, формы, цитаты, контактная информация, ссылки, видео, изображения, кнопки, мелкие текстовые элементы и т.п.

Для примера, страница с выделением блоков различными семантическими элементами.

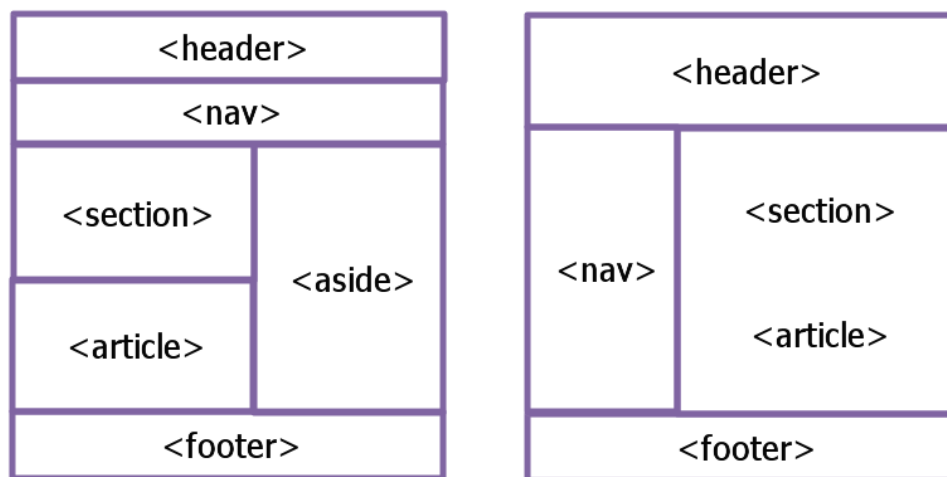


Рис.150. Семантические принципы разделения на блоки.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Список возможностей HTML велик. В спецификации свыше 100 тегов и более 130 атрибутов к ним. Также существуют команды обработчика событий, интерфейсов и т.д. Самостоятельно изучите справочники и разберитесь, как именно осуществляются следующие операции:

- возможности выравнивания текста по центру, влево, вправо, по ширине;
- возможности форматирования текста (изменение размера, начертания, кегля шрифта и т.п.);
- изменение цвета фона;
- варианты обтекания картинки текстом;
- рисование графики.

ЧИТАЕМ

Чуть раньше мы изучили некоторые возможности языка гипертекстовой разметки HTML. В предыдущих главах мы создали веб-страничку с помощью HTML, где объекты располагаются некрасиво, друг за другом. Использование CSS позволит в HTML-документе красиво и достаточно просто оформить расположение этих объектов.

На заре развития интернет-технологий использовали исключительно HTML, с помощью которого можно было выделить заголовок, выделить абзац, изменить начертание текста. Но со временем потребовалось страницы делать более привлекательными и интересными. Разработка CSS позволит улучшить внешний вид страницы.

CSS (Cascading Style Sheets — «каскадные таблицы стилей») не язык разметки — это язык таблицы стилей. Это означает, что он позволяет применять стили выборочно к элементам в документах HTML. CSS поддерживает таблицы стилей для конкретных носителей, поэтому веб-дизайнеры могут адаптировать информацию на веб-страницах к браузерам, брайлевским устройствам, носимым устройствам, слуховым устройствам, принтерам и т.д.

Каскадные таблицы стилей описывают правила форматирования элементов с помощью свойств и допустимых значений этих свойств. Познакомимся с этими свойствами:

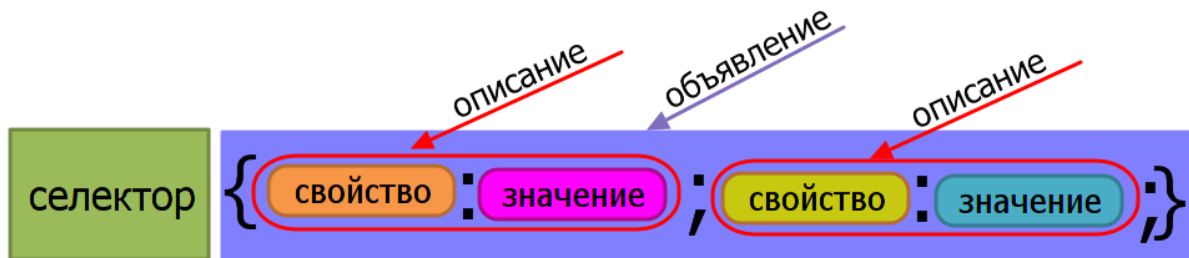


Рис.151. структура правил для CSS

1. **Селектор** — это элемент HTML к которому будет применен стиль. Например, мы хотим изменить шрифт у всех элементов, выделенных тегом `<p>`.
2. **Свойство** — это способ стилизации элемента HTML. Например, мы хотим изменить цвет, тогда в свойстве поставим `color`.

3. **Значение** — это то значение, которое вы выбрали для свойства. Например, мы хотим сделать цвет зеленым, то пишем `p { color : green ; }`. В этом случае применено правило к элементу `p` (абзац) и весь текст, который выделен с помощью этого элемента станет зеленым.

4. Все свойства и их значения образуют область, которую называют **объявление**. Объявление выделяется фигурными скобками. Свойство и значение разделяется двоеточием, после каждого описания обязательно ставится точка с запятой.

Таблицы стилей могут использоваться как внешние, так и внутренние, встроенные, импортированные.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №52

Внешняя таблица стилей создается в отдельном файле с расширением CSS. Внешнюю таблицу стилей подключают к веб-странице тегом `<link>`, расположенным внутри заголовка `<head> ... </head>`.

`rel="stylesheet"` указывает тип ссылки (ссылка на таблицу стилей).

Если добавить атрибут `media = "all"`, то стили применяются ко всему веб-сайту.

В редакторе кода создадим новый файл `style1.css` с содержанием

```
p { color: green; }
```

Далее перейдем в документ HTML и с помощью команды `link` вставим ссылку на внешний файл.

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <link href="style.css" rel="stylesheet" >
    <meta charset= "UTF-8">
    <title> Создание страницы на HTML5 </title>
```

Откроем в браузере файл `index.html` и убедимся, что все строки после тега `<p>` стали зеленого цвета.

Страница с использованием тегов HTML5

Понимание языков **HTML** и **CSS** полезно всем, чья работа связана со Всемирной паутиной:

дизайнеры смогут создавать более привлекательные и удобные в использовании сайты, редакторы — наполнять сайты более качественным контентом, специалисты по маркетингу — более эффективно общаться с целевой аудиторией, менеджеры разного уровня будут более четко формулировать требования при создании сайтов и получать лучшие результаты от своих сотрудников.

Основные элементы **HTML**:

1. Форматирование текста — выделение курсивом, жирным шрифтом, подчеркивание, размер кегля, нумерованные/маркированные списки.
 2. Текстовые блоки — заголовки уровней H1-H6, абзацы, перенос на новую строку.
 3. Таблицы — любое количество строк, столбцов, фиксированная высота, ширина, заголовок.
 4. Вставка объектов — изображения, звуковые, текстовые, видеофайлы и т.д.
 5. Гиперссылки — на файл изображения, прайс-листа, страницу, на которую ссылается документ.
- Есть атрибуты открытия документа в текущем или новом окне.

Преимущества сайтов на **HTML**-коде:

- меньший "вес" по сравнению с CMS;
- небольшой расход ресурсов сервера;
- высокий уровень безопасности данных, малая уязвимость к взлому;
- высокая стабильность;

Пройти бесплатно курсы можно у известных компаний

Рис. 152. Веб-страничка на которой цвет шрифта изменен на зеленый с помощью внешней таблицы стилей.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №53

Рассмотрим пример внутреннего стиля. Внутренние стили имеют приоритет над внешними.

Встроим стиль непосредственно в заголовке страницы, т.е. между тегами `<head> ... </head>`. Тогда стиль задается двойным тегом `<style>`.

Для примера изменим цвет и начертание заголовка выделенного тегом `<h1>`. Встроим стиль изменения между `<head> ... </head>`

```
<style>
  h1 {
    color: red;
    font-family: "Times New Roman", Georgia, Serif;
  }
</style>
```

Результат получается таким:

Страница с использованием тегов HTML5

Понимание языков HTML и CSS полезно всем, чья работа связана со Всемирной паутиной:

Рис. 153. Внешний вид после выполнения задания

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №54

Встроенные стили действуют на один элемент, для которого они заданы.

Применим стиль только к одной фразе. Сделаем жирное начертание и синий цвет букв.

```
<p style="font-weight: bold; color: blue;">
Преимущества сайтов на HTML-коде:</p>
```

Рис. 154. Фрагмент кода

Результат получится таким:

Преимущества сайтов на HTML-коде:

- меньший "вес" по сравнению с CMS;
- небольшой расход ресурсов сервера;
- высокий уровень безопасности данных, малая уязвимость к взлому;
- высокая стабильность;

Рис. 155. Внешний вид после выполнения задания

ЧИТАЕМ

Селектор – один из важных элементов CSS. С их помощью браузер определяет, к каким элементам будут применены стили в фигурных скобках. Для одного правила можно применять несколько селекторов, тогда селекторы отделяются запятой.

- *Универсальный селектор* – выбирает все элементы */*.
- *Селектор по типу* – все элементы указанного типа. Если используют несколько типов, то перечисление идет через запятую.
- *Селектор по классу* – все элементы с атрибутом class.
- *Селектор по идентификатору* – выбирает элементы с атрибутом id. Селектор начинается со знака #.
- *Селектор по атрибуту* – этот селектор хорошо использовать при поиске по шаблону. Сам селектор берется в квадратные скобки.
- *Селектор потомков* – выбирает элементы заданного типа, которые находятся внутри элементов-родителей.

- *Селектор дочерних элементов* – выбирает элементы заданного типа, а которые являются дочерними к указанному селектору.
- *Селектор соседних элементов* – выбирает элементы заданного типа, которые находятся после указанного в селекторе элемента.
- *Селектор родственных элементов* – выбирает элементы заданного типа, которые находятся после указанного в селекторе элемента и имеют общего родителя.
- *Селекторы псевдоклассов.*
- *Селекторы псевдоэлементов.*
- *Комбинирование селекторов CSS.*

Каждую категорию селекторов можно отдельно изучать с большим количеством примеров и особенностей применения.

Специфичность определяет, какие значения свойств CSS наиболее соответствуют элементу и будут применены браузером при просмотре страницы. Специфичность строится на правилах соответствия для селекторов CSS различных типов.

Пример селекторов, расположенных по возрастанию специфичности:

1. селекторы типов элементов (например, `h1`) и псевдоэлементов (например, `::first-letter`).
2. селекторы классов (например, `.proba`), селекторы атрибутов (например, `[href*="example"]`) и псевдоклассов (например, `:hover`).
3. селекторы идентификаторов (например, `#example`).

Универсальный селектор (`*`), комбинаторы (`+`, `>`, `~`, `' '`) и отрицающий псевдокласс (`:not()`) не влияют на специфичность.

Наивысшая специфичность будет у стилей внутренних, объявленных в элементе. Такие стили всегда приоритетнее правил из внешних файлов стилей.

Существует много селекторов для изучения, и вы можете найти более подробный список, например, самоучитель <https://puzzleweb.ru/> или Руководстве селекторов (en-US).

Теперь вспомним, что мы рассматривали с вами так называемое блочное оформление web-документа. У нас есть возможность задавать оформление целому блоку. Для этого мы воспользуемся тегом `<div>` для обозначения блока, атрибутом `class`. Внутри тега обозначим блок и установим этому блоку ширину 400 пикселей, а цвет фона установим желтый.

```
<div class = "blok1">    // означает, что выделенному фрагменту
дали имя blok1
    <h1> текст... </h1>
    <p> текст...</p>
</div>
<style type = "text/css"> /*означает, что стиль применяют к тегу
    с классом blok1*/
```

```
.blok1 {width: 400px;
        background-color: yellow;
    }
</style>
```

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №55

Разделение на блоки с помощью двойного тега <div>. Этот тег активно использовался во времена HTML4. С приходом HTML5 стали больше использовать другие семантические структуры, но тег <div> все еще остается актуальным.

В примере ниже <div> использует атрибут <class>, и с его помощью ширина этого блока становится равной 400, а цвет фона – желтым.

```
17 <body> |
18 <div class = "blok1">
19
20 <h1> Страница с использованием тегов HTML5 </h1>
21 <p> Понимание языков <strong> HTML и CSS </strong> полезно
22 дизайнеры смогут создавать более привлекательные и удобные
23 редакторы – наполнять сайты более качественным контентом,
24 специалисты по маркетингу – более эффективно общаться с це
25 менеджеры разного уровня будут более четко формулировать т
26 и получать лучшие результаты от своих сотрудников.<br>
27 </div>
28
29 <style type = "text/css">
30     .blok1 {
31         width: 400px;
32         background-color: yellow;
33     }
34 </style>
```

Рис. 156. Фрагмент кода

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №56

Рассмотрим еще один пример стилизации блоков с помощью двойного тега <section>, который появился в версии HTML5. Его стали применять для группировки логически связанного контента и создания разных разделов, например, новостей, контактной информации и т.п. Приведем пример, как с помощью <section> можно разделить текст на блоки.

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>Создание страницы на HTML5 </title>
5  </head>
6  <body>
7    <section class="blok1">
8      <h1>Страница с использованием тегов HTML5</h1>
9      <p>Понимание языков HTML и CSS полезно всем, чья работа связана со Всемирной паутиной:</p>
10   </section>
11   <section class="blok2">
12     <h3>Основные элементы HTML:</h3>
13     <ol>
14       <li>Форматирование текста – выделение курсивом, жирным шрифтом, подчеркивание, <br>
15         размер кегля, нумерованные/маркированные списки.</li>
16       <li>Текстовые блоки – заголовки уровней H1-H6, абзацы, перенос на новую строку. </li>
17       <li>Таблицы – любое количество строк, столбцов, фиксированная высота, ширина, заголовки.</li>
18       <li>Вставка объектов – изображения, звуковые, текстовые, видеофайлы и т.д.</li>
19       <li>Гиперссылки – на файл изображения, прайс-листа, страницу, на которую ссылается пункт меню и
20     </ol>
21   <style type = "text/css">
22     .blok1 {
23       width: 600px;
24       background-color: yellow;
25     }
26   </style>
27
28   <style type = "text/css">
29     .blok2 {
30       width: 300px;
31       background-color: #ADFF2F;
32     }
33   </style>
34 </body>
35 </html>

```

Рис. 157. Фрагмент кода

Результат получится таким:

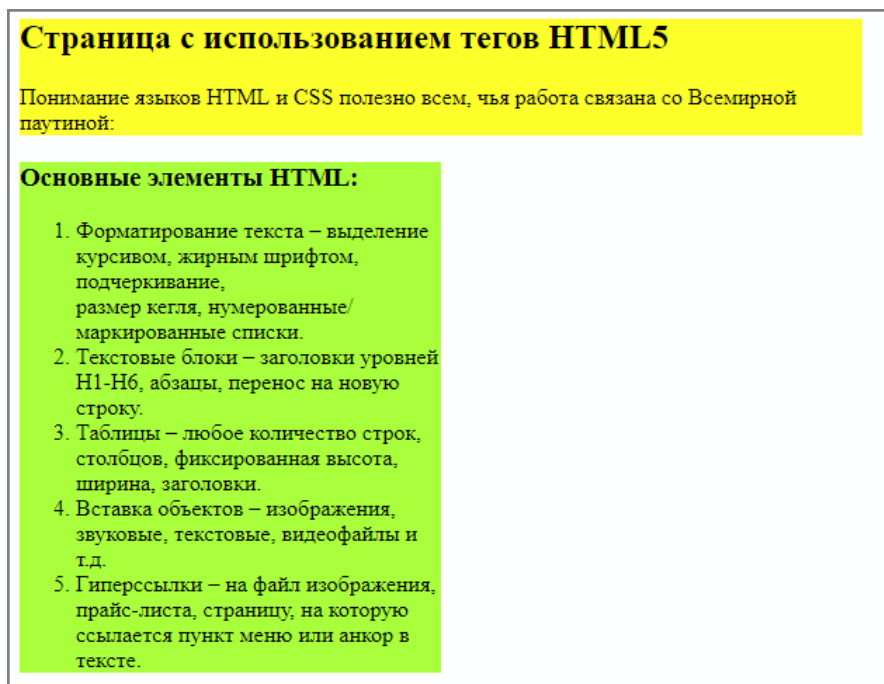


Рис. 158. Внешний вид после выполнения задания

На примере выше видно, что с помощью <section> разделили текст на блоки, а потом к каждому блоку добавили свой стиль оформления.

В HTML используются и другие секционные элементы, которые позволяют добавить смысловую или семантическую нагрузку страницам сайта, что обеспечивает компьютерным программам лучшее понимание их содержания.

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. Справочник CSS

<https://developer.mozilla.org/ru/docs/Web/CSS/Reference#selectors>

§5.6 Введение в JavaScript

ВСПОМИНАЕМ

Вспомним:

1. Что такое frontend и backend?
2. Как используются каскадные таблицы стилей? Для чего они нужны?

ЧИТАЕМ

Применение языка программирования JavaScript (JS) к документу HTML позволит добавить интерактивность на сайт, создавать игры 2D и 3D, приложения с базами данных, анимацию, всплывающие окна, спецэффекты, формы проверки, применить динамические стили, т.е. настроить взаимодействие с пользователем, настроить отклик при вводе данных, нажатии кнопок и т.п.

JavaScript — клиентский язык программирования, выполняется непосредственно в браузере пользователя на его локальном компьютере.

Для подключения JS-скриптов используется тег `<script>`. Вызов скрипта может быть из заголовка, после `<head>`, может быть в теле `<body>` или после него. Все зависит от того, в какой момент надо использовать скрипт.

Встроить команды JS в HTML документ можно также несколькими способами:

1. Подключением внешнего скрипта.
2. Встраиванием скрипта в код HTML.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №57

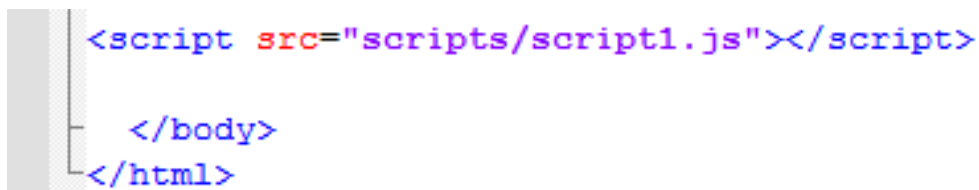
Подключение внешнего скрипта

В папке, где находится файл `index.html` от предыдущих примеров, создайте папку с именем `SCRIPTS`. В этой папке создайте файл с расширением `.js`, например, `script.js`.

В файл `script.js` добавьте следующий код:

```
var myHeading = document.querySelector('h1');  
myHeading.textContent = 'Подключаем JavaScript';
```

На основной странице HTML внесите команды для обращения к этому файлу перед тегом `</body>`.



```
<script src="scripts/script1.js"></script>  
  
</body>  
</html>
```

Рис. 159. Фрагмент кода

Открываем файл `index.html` в браузере и обнаруживаем, что заголовок “Страницы с использованием тегов HTML” заменен на текст, который был в скрипте. Браузер отработал таким образом: сначала выполнил все команды в HTML-документе, а в конце выполнил скрипт, в котором указывалось заменить текст для заголовка `<h1>`.

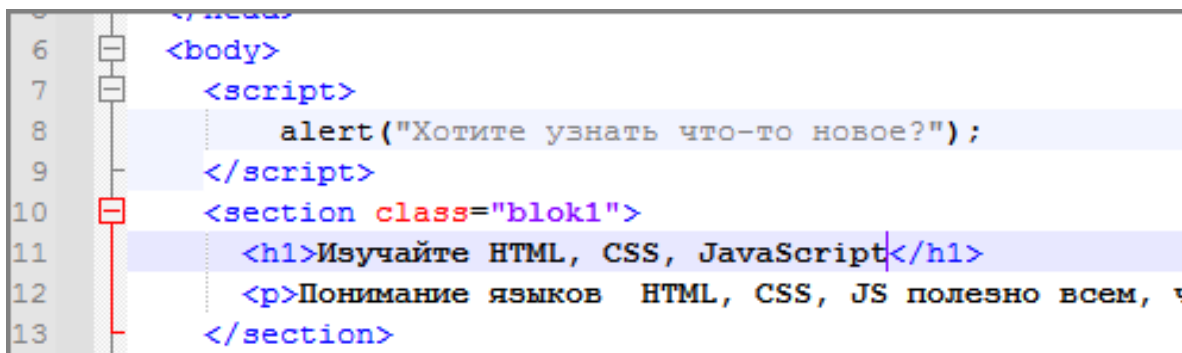
ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №58

HTML-элементы в браузере выполняются в том порядке, в котором они расположены в файле. Выполнение скрипта в документе HTML можно ставить в принципе в любом месте, главное, чтобы элементы HTML, с которыми работает скрипт к этому времени, уже успели загрузиться.

Рассмотрим пример загрузки скрипта сразу после `</body>`.

```
<script>  
alert("Хотите узнать что-то новое?");  
</script>
```



```
6 <body>  
7 <script>  
8     alert("Хотите узнать что-то новое?");  
9 </script>  
10 <section class="blok1">  
11 <h1>Изучайте HTML, CSS, JavaScript</h1>  
12 <p>Понимание языков HTML, CSS, JS полезно всем, ч  
13 </section>
```

Рис. 160. Фрагмент кода

Выполнение этого скрипта произойдет сразу, до загрузки основного текста страницы, и поэтому сначала на странице появится окно с вопросом и подтверждением действия, а потом уже появится вся страница.

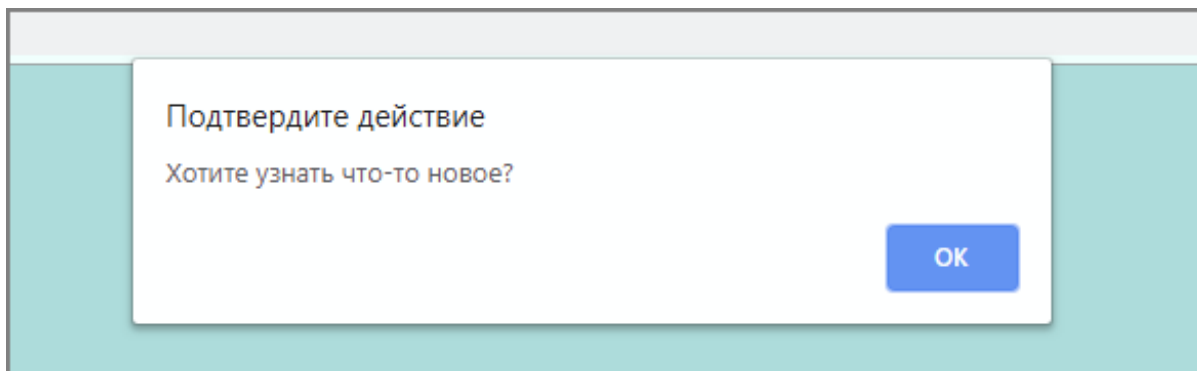


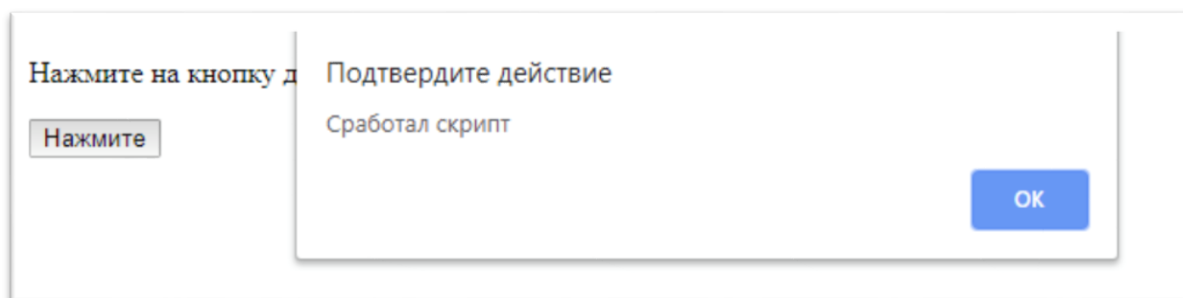
Рис. 161. Внешний вид после выполнения задания

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №59

Встроим скрипт непосредственно в созданный сайт. Предположим, нам надо сделать кнопку на сайте, чтобы после нажатия этой кнопки появлялось сообщение. Для этого действия можно использовать обработчик событий **onclick()**. Это событие будем вызывать внутри HTML-документа.

```
<p>Нажмите на кнопку для выполнения действия.</p>
<input type='button' value='Нажмите' onclick='showMessage()'>
<script> function showMessage()
    { alert('Сработал скрипт') }
</script>
```



ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №60

Рассмотрим некоторые примеры использования скриптов для HTML-документов.

1. С помощью скрипта можно поменять содержимое контента.

```
<script> function Hello()
{document.getElementById("Пример1").innerHTML = "Первый
пример";}
</script>
```

2. С помощью скрипта можно поменять значение HTML-атрибута.


```
<script> function dancer()  
{ document.getElementById("image").src = "picture.gif"; }  
</script>
```

3. С помощью скрипта можно поменять стили.

```
<script> function setColor()  
{ document.getElementById("demo").style.backgroundColor = "red";  
}  
</script>
```

4. С помощью скрипта можно поменять шрифты.

```
<script> function setFont() {  
document.getElementById("Пример1").style.fontSize ="14 px";}  
</script>
```

ЧИТАЕМ

JavaScript — это язык программирования, и поэтому все основные логические конструкции программирования остаются и работают, например, циклы, ветвления и т.п.

И конечно же, используются переменные. Напоминаем, что переменная представляет собой именованную область памяти, в которой хранятся данные (значения). Переменные имеют имя, тип и значение.

Имена переменных в JS чувствительны к регистру, и это следует учитывать. Остальные требования к имени стандартны.

Один из способов объявления переменной — с помощью команды `var` имя_переменной). Например,

```
var probal; /*объявили переменную с именем probal */  
var probal = 5; /*объявили переменную с именем probal и  
присвоили значение 5*/
```

Часто также применяют команду `let`.

Переменные в JS используются следующих типов:

1. *Строковый тип (string)* – хранение строки символов, которые заключаются в одинарные или двойные кавычки.

Пример:

```
var symbol1 = ""; // пустая строка, ноль символов  
var symbol2 = 'Привет';  
var symbol3 = "Привету";  
var symbol4 = "1000000"; // число в кавычках так же имеет  
строковый тип
```

Строки можно сравнивать, объединять (операция конкатенации +).

Можно создать новую строку путем объединения других строк.

2. *Числовой тип (number)* – хранение числовых значений.

Числа бывают целые и с плавающей точкой.

```
var a = 6; // целое десятичное числовое значение
var a = 6.78; // вещественное число
var b = 6.789E+2; // вещественное число, эквивалентно 6.789 X 102
var c = 6.7e-2; // вещественное число, эквивалентно 6.7 X 10-2
```

Допускается запись чисел десятичной системе счисления, а так же в восьмеричной и шестнадцатеричной.

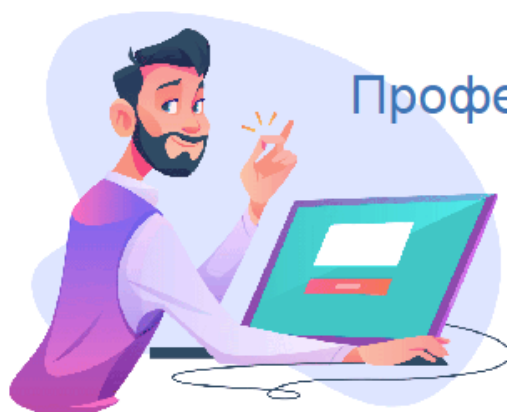
```
var b = 022; // восьмеричный формат
var c = 0xf2a; // шестнадцатеричный формат
```

3. *Логический тип (boolean)* – значение true (истина), false (ложь).

4. *Объект (object)*. Массивы тоже относятся к типу данных объекта.

§5.7 Диагностическая работа

Создайте страницу по образцу ниже. Цвета и изображения можно оставить на своё усмотрение.



Профессии будущего в ИТ

Мы живем в мире, где более 70% востребованных в будущем профессий еще не существуют, а 50% ныне существующих профессий вскоре превратятся во внештатные. Все меняется слишком быстро.

Сейчас огромный спрос во всех ИТ сферах:

- программировании;
- аналитике;
- робототехнике;
- тестировании;
- компьютерной безопасности.

Профессии будущего, представленные в [Атласе Новых Профессий](#).

| ИТ сектор | Робототехника и машиностроение | Биотехнологии |
|----------------------------------|--|-------------------------|
| Архитектор информационных систем | Оператор многофункциональных робототехнических устройств | Системный биотехнолог |
| Дизайнер интерфейсов | Проектировщик-эргономист | Архитектор живых систем |
| Сетевой юрист | Инженер-композитчик | Урбанист-эколог |
| ИТ-проповедник | Проектировщик детской робототехники | Биофармаколог |

Заполните анкету

Какие профессии вы считаете наиболее востребованными в будущем?

Введите профессии через запятую

Специалисты в какой области ИТ, по вашему мнению, сегодня востребованы больше всего?

Введите описание

Отправить

Рис. 162. Примерный внешний вид результата выполнения итогового задания

VI. Неравномерное кодирование, помехоустойчивые коды

§6.1 Помехоустойчивые коды

ВСПОМИНАЕМ

Вспомним:

1. Что такое «кодирование»?
2. Какие виды кодирования вам известны?

ЧИТАЕМ

Назначение помехоустойчивого кодирования – защита информации от помех и ошибок при передаче и хранении информации.

Без использования помехоустойчивого кодирования было бы невозможно передавать большие объемы информации (файлы), т.к. в любой системе передачи и хранении информации неизбежно возникают ошибки.

ОБСУЖДАЕМ

Вы наверняка знаете, что до появления карт памяти и «флешек» в качестве носителей цифровой информации использовались оптические диски. Иногда они встречаются и сейчас.

Например, возьмем CD или DVD диск. Из-за того, что все дорожки находятся на поверхности диска, информация хранится прямо на их поверхности в углублениях. Поверхности этих дисков бывают поцарапанными, на них остаются отпечатки пальцев. И, если бы не было помехоустойчивого кодирования, то информацию извлечь из них просто не получилось бы.

ЧИТАЕМ

В зависимости от используемых в системе настроек на обнаружение или исправление ошибок с помощью помехоустойчивого кода, можно:

- либо выполнить только обнаружение ошибок, при их наличии сделать запрос на повторную передачу пакета данных;
- либо произвести декодирование помехоустойчивого кода, т. е. исправить ошибки с его помощью;
- либо получить пакет информации, попробовать его исправить, и, если не получится, тогда отправить запрос на повторную передачу.

Любое помехоустойчивое кодирование добавляет контролируемую избыточность, за счет чего и появляется возможность восстановить информацию при частичной потере данных в канале связи.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Условие:

Рассмотрим мажоритарный метод (многократная передача): одну и ту же порцию информации передают многократно, а на приемной стороне принимается решение о том, была ли ошибка при передаче по каналу связи или нет.

У нас есть информация, которую мы хотим передать: 'TRUE'. Эту информацию повторяем несколько раз. В процессе передачи информации по каналу связи где-то возникла ошибка. Есть три пакета ('TRUE'|'TRUE'|'TRUE'), которые должны нести одну и ту же информацию.

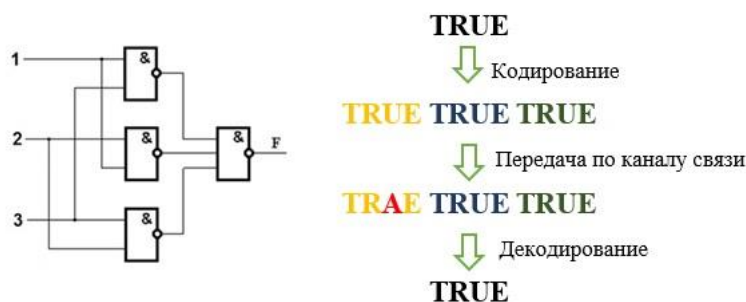


Рис. 163 Эквивалентная схема, реализующая мажоритарный метод

Рассуждение:

На схеме видно, что третий символ (U и A) отличается один от другого, хотя должны быть одинаковыми, значит есть ошибка.

Необходимо найти ошибку с помощью голосования: каких символов больше, символов U или символов A? Символов U больше, чем символов A, соответственно принимаем решение, что передавался символ U, а символ A ошибочный. Причем для исправления ошибок нужно как минимум 3 пакета информации, для обнаружения – как минимум 2 пакета информации.

ЧИТАЕМ

Параметры помехоустойчивого кодирования:

1) **скорость кода (R)** – характеризует долю информационных («полезных») данных в сообщении и определяется выражением: $R = k/n = k/(m+k)$, где n – количество символов закодированного сообщения (результата кодирования), m – количество проверочных символов, добавляемых при кодировании, k – количество информационных символов.

Параметры n и k часто приводят вместе с наименованием кода для его однозначной идентификации. Например, код Хэмминга (7,4), Рида-Соломона (15, 11) и т.д.

2) **кратность обнаруживаемых ошибок** – количество ошибочных символов, которые код может обнаружить.

3) **кратность исправляемых ошибок** – количество ошибочных символов, которые код может исправить (обозначается буквой t).

Под кратностью понимаются всевозможные ошибки, которые можно обнаружить.

Формирование проверочных бит производится по выражениям:

$$y_5 = x_1 \oplus x_2 \oplus x_3,$$

$$y_6 = x_2 \oplus x_3 \oplus x_4,$$

$$y_7 = x_1 \oplus x_2 \oplus x_4.$$

Таблица кодировки для (7,4)

| x_1 | x_2 | x_3 | x_4 | | | |
|-------|-------|-------|-------|-------|-------|-------|
| y_1 | y_2 | y_3 | y_4 | y_5 | y_6 | y_7 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Рис. 164 Код Хэмминга (7,4)

Анализируя рисунок выше, видим, что $k = 4$ (это входные данные x_1, x_2, x_3, x_4), $n = 7$ (это выходные данные от y_1 до y_7), $m = 3$ (это проверочные биты y_5, y_6, y_7).

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 2

Условие:

Рассмотрим самый простой метод помехоустойчивого кодирования – добавление одного бита четности (контроль чётности).

Есть некое информационное сообщение, состоящее из 8 бит, добавим девятый бит.

Если нечетное количество единиц, добавляем 1: 1 0 1 0 0 1 0 0 | 1.

Если четное количество единиц, добавляем 0: 1 1 0 1 0 1 0 0 | 0.

Рассуждение:

Если принятый бит чётности не совпадает с рассчитанным битом чётности, то считается, что произошла ошибка, например, 1 1 0 0 0 1 0 0 | 0.

В этом случае кратность исправляемых ошибок 0, так как мы не можем исправить ошибки ($t = 0$), а кратность обнаруживаемых 1.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 3

Условие:

Рассмотрим «Прямоугольный код» – код с контролем четности, позволяющий исправить одну ошибку.

У нас есть последовательность из 0 и 1, из которой можно составить прямоугольную матрицу. Возьмем, например, размером 4 x 4.

Рассуждение:

Затем для каждой строки и столбца посчитаем бит четности.

сообщение на входе

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |

сообщение на выходе

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |

Рис. 165 Демонстрация работы прямоугольного кода

И если в процессе передачи информации допустим ошибку (ошибка выделена желтым цветом), начинаем делать проверку. Нашли ошибку во втором столбце, третьей строке по координатам. Чтобы исправить ошибку, просто инвертируем 1 в 0, тем самым ошибка исправляется.

Этот прямоугольный код исправляет все однобитные ошибки, но не все двухбитные и трехбитные.

Сейчас на практике прямоугольный код не используется, но логика работы многих помехоустойчивых кодов основана именно на прямоугольном коде.

ЧИТАЕМ

Код Хэмминга

Код Хэмминга — наиболее известный из первых самоконтролирующихся и самокорректирующихся кодов, который позволяет устранить одну ошибку и находить двойную.

Код Хэмминга состоит из двух частей. Первая часть кодирует исходное сообщение, вставляя в него в определённых местах контрольные биты (вычисленные особым образом). Вторая часть получает входящее сообщение и заново вычисляет контрольные биты (по тому же алгоритму, что и первая часть). Если все вновь вычисленные контрольные биты совпадают с полученными, то сообщение получено без ошибок. В противном случае, выводится сообщение об ошибке и при возможности ошибка исправляется.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 4

Условие:

Рассмотрим принцип работы кода Хэмминга. Допустим, у нас есть сообщение «bars», которое необходимо передать без ошибок.

Решение:

Для этого сначала нужно наше сообщение закодировать при помощи кода Хэмминга.

Нам необходимо представить его в бинарном виде:

| Символ | ASCII код | Бинарное представление |
|--------|-----------|------------------------|
| b | 98 | 01100010 |
| a | 97 | 01100001 |
| r | 114 | 01110010 |
| s | 115 | 01110011 |

На этом этапе стоит определиться с длиной информационного слова, то есть длиной строки из нулей и единиц, которые мы будем кодировать. Допустим, у нас длина слова будет равна 16. Таким образом, нам необходимо разделить наше исходное сообщение («bars») на блоки по 16 бит, которые мы будем потом кодировать отдельно друг от друга. Так как один символ занимает в памяти 8 бит, то в одно кодируемое слово помещается ровно два ASCII символа. Итак, мы получили две бинарные строки по 16 бит:

| | | | | |
|----------|----------|---|----------|----------|
| b | a | и | r | s |
| 01100010 | 01100001 | | 01110010 | 01110011 |

После этого процесс кодирования распараллеливается, и две части сообщения («ba» и «rs») кодируются независимо друг от друга. Рассмотрим, как это делается на примере первой части.

Прежде всего, необходимо вставить контрольные биты. Они вставляются в строго определённых местах — это позиции с номерами, равными степеням двойки. В нашем случае (при длине информационного слова в 16 бит) это будут позиции 1, 2, 4, 8, 16. Соответственно, у нас должно получиться по 5 контрольных бит (выделены красным цветом):

Было:

| | | | | |
|----------|----------|---|----------|----------|
| b | a | и | r | s |
| 01100010 | 01100001 | | 01110010 | 01110011 |

Стало:

| | | | | |
|--------------|-----------|--|--------------|-----------|
| b | a | | r | s |
| **0*110*0010 | 011*00001 | | **0*111*0010 | 011*10011 |

Таким образом, длина каждого информационного слова увеличивается на 5 бит. До вычисления самих контрольных бит, мы обозначили их символом «*». Теперь необходимо вычислить значение каждого контрольного бита.

Значение каждого контрольного бита зависит от значений информационных бит, но не от всех, а только от тех, которые этот

контрольный бит контролирует. Для того, чтобы понять, за какие биты отвечает каждый контрольный бит, необходимо понять очень простую закономерность: контрольный бит с номером N контролирует все последующие N бит через каждые N бит, начиная с позиции N .

Например, первый контрольный бит ($N = 1$) контролирует все биты длиной один через один бит, начиная с первого, т.е. 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21. А, например, четвертый контрольный бит ($N = 4$) соответственно будет контролировать по четыре бита через каждые четыре, начиная с четвертого (см. рис. 1.4) и т.д.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | |
| x | | x | | x | | x | | x | | x | | x | | x | | x | | x | | x | 1 |
| | x | x | | | x | x | | | x | x | | | x | x | | | x | x | | | 2 |
| | | | x | x | x | x | | | | | x | x | x | x | | | | | x | x | 4 |
| | | | | | | | x | x | x | x | x | x | x | | | | | | | | 8 |
| | | | | | | | | | | | | | | | x | x | x | x | x | x | 16 |

Рис. 166. Контролируемые биты и контрольные биты

Здесь знаком «х» обозначены те биты, которые контролирует контрольный бит, номер которого указан справа.

То есть, к примеру, бит номер 12 контролируется битами с номерами 4 и 8. Ясно, что чтобы узнать, какими битами контролируется бит с номером N , надо просто разложить N по степеням двойки.

Для вычисления значения каждого контрольного бита нужно просто определить бит четности: берём каждый контрольный бит и смотрим, сколько среди контролируемых им битов единиц, получаем некоторое целое число и, если оно чётное, то ставим ноль, в противном случае ставим единицу.

Подсчитаем контрольные биты для нашего первого информационного слова, получим:

| b | a |
|------------------------|---|
| 000111010010 011100001 | |

Допустим, мы получили закодированное первой частью алгоритма сообщение, но оно пришло к нам с ошибкой. К примеру, мы получили такое (11-ый бит передан неправильно):

| b | a |
|------------------------|---|
| 000111010000 011100001 | |

Чтобы понять, есть ли ошибки в переданном коде или нет, нужно заново вычислить все контрольные биты (так же, как и в первой части) и сравнить их с контрольными битами, которые мы получили. Так, посчитав контрольные биты с неправильным 11-ым битом, мы получим такую картину:

| b | a |
|------------------------|---|
| 110111000010 011100001 | |

Как мы видим, контрольные биты под номерами: 1, 2, 8 не совпадают с такими же контрольными битами, которые мы получили ранее. Складываем номера позиций неправильных контрольных бит ($1 + 2 + 8 = 11$) и мы получаем позицию ошибочного бита. Теперь просто инвертировав его и отбросив контрольные биты, мы получим исходное сообщение. Аналогично поступаем со второй частью сообщения.

РЕШАЕМ В ТЕТРАДИ

Задание 36.

Вычислите скорость кода Хэмминга (7, 4).

РЕШАЕМ В ТЕТРАДИ

Задание 37.

Определите количество избыточных бит (m), если известна скорость кода Хэмминга $R = 11/15$.

РЕШАЕМ В ТЕТРАДИ

Задание 38.

Вычислите значения каждого контрольного бита для второй части сообщения из примера 4:

| r | s |
|------------------------|---|
| **0*111*0010 011*10011 | |

Это сообщение передалось с ошибкой:

| r | s |
|------------------------|---|
| **0*111*0010 011*10111 | |

Определите позиции ошибочных контрольных бит.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. В чем состоит основное назначение помехоустойчивого кода?
2. С какой целью при помехоустойчивом кодировании добавляется контролируемая избыточность?

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. Владимиров С. С. «Математические основы теории помехоустойчивого кодирования», курс лекций — 2014. URL: http://opds.spbsut.ru/data/_uploaded/mu/motpuk-lect-01.pdf
2. Вернер М. Основы кодирования (Мир программирования) — 2004

3. Код Хемминга. Демонстрирующая программа. URL: <https://orenstudent.ru/Hemming.htm>

§6.2 Помехоустойчивые коды. Решение типовых задач.

ВСПОМИНАЕМ

Вспомним:

1. В чем смысл мажоритарного метода?
2. Перечислите параметры помехоустойчивого кодирования.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Условие:

Виктор придумал алгоритм кодирования десятичного числа X :

1. Вначале строится двоичная запись числа X .
2. К этой записи справа дописываются ещё два разряда по следующему правилу:

а) складываются все цифры двоичной записи числа X , и осуществляется контроль четности. Например, 11110 преобразуется в запись 111100, а 10011 преобразуется в запись 100111;

б) над этой записью производятся те же действия дважды.

Полученная таким образом запись (в ней на два разряда больше, чем в записи исходного числа X) является двоичной кодовой записью (D) числа X . Укажите минимальное число D , которое превышает число 390 и может являться результатом работы данного алгоритма. В ответе это число запишите в десятичной системе счисления.

Решение:

Построим двоичную запись числа $390_{10} = 110000110_2$, тогда исходное число $X = 1100001_2$. После первого действия алгоритма получится запись 11000011_2 , а после второго такого же действия получится 110000110_2 . Значит, что число 390_{10} нам подходит, но оно не удовлетворяет нашему условию. Значит нам нужно взять число $X = 1100010_2$. Тогда $D = 110001010_2 = 394_{10}$. Это и будет ответ.

Ответ: 394.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 2

Условие:

Виктор придумал алгоритм кодирования десятичного числа $X > 1$, результатом работы которого является новое число D по следующему принципу:

1. Вначале строится двоичная запись числа X .
2. В конец записи дописывается последняя и первая цифра двоичной записи числа X .
3. Результат переводится в десятичную систему.

Например, число $X = 13_{10}$, двоичная запись которого 1101_2 , преобразуется в 110111_2 .

Результатом работы алгоритма будет число $D = 56_{10}$.

При каком наименьшем числе X в результате работы алгоритма получится $D > 170$? В ответе запишите это число в десятичной системе счисления.

Решение:

Число на выходе должно превышать $170_{10} = 10101010_2$, тогда исходное число X должно быть больше или равно $101010_2 = 42_{10}$.

Рассмотрим числа X , большие или равные 42, и определим наименьшее из них, удовлетворяющее нашему условию:

$42_{10} = 101010_2$, результатом работы алгоритма будет число $10101001_2 = 169_{10}$, не подходит;

$43_{10} = 101011_2$, результатом работы алгоритма будет число $10101111_2 = 175_{10}$, подходит под условие.

Таким образом, ответ — 43.

Ответ: 43

РЕШАЕМ В ТЕТРАДИ

Задание 39.

Виктор придумал алгоритм кодирования десятичного числа X :

1. Вначале строится двоичная запись числа X .
2. К этой записи дописывается справа два нуля, если число четное, или две единицы в противном случае

Укажите максимальное число X , после обработки которого с помощью этого алгоритма получается число менее 94. В ответе это число запишите в десятичной системе.

РЕШАЕМ В ТЕТРАДИ

Задание 40.

Виктор придумал алгоритм кодирования десятичного числа X :

1. Вначале строится двоичная запись числа X .

2. К этой записи дописываются справа ещё два разряда по следующему правилу: если X чётное, в конец числа (справа) дописываются два нуля, в противном случае справа дописываются две единицы. Например, двоичная запись 1010 числа 10 будет преобразована в 101000_2 – это результат работы данного алгоритма.

Укажите:

а) максимальное число X , для которого результат работы алгоритма будет меньше 134.

б) минимальное число X , для которого результат работы алгоритма будет больше 134.

В ответе эти числа запишите в десятичной системе счисления друг за другом без разделительных знаков.

РЕШАЕМ В ТЕТРАДИ

Задание 41.

Виктор придумал алгоритм кодирования десятичного числа X :

1. Вначале строится двоичная запись числа X .

2. К этой записи справа дописываются ещё два разряда по следующему правилу:

а) складываются все цифры двоичной записи числа X , и осуществляется контроль четности. Например, 11110 преобразуется в запись 111100, а 10011 преобразуется в запись 100111;

б) над этой записью производятся те же действия дважды.

Полученная таким образом запись (в ней на два разряда больше, чем в записи исходного числа X) является двоичной кодовой записью (D) числа X . Укажите минимальное число D , которое превышает число 97 и может являться результатом работы данного алгоритма. В ответе это число запишите в десятичной системе счисления.

РЕШАЕМ В ТЕТРАДИ

Задание 42.

Виктор придумал алгоритм кодирования десятичного числа X :

1. Вначале строится двоичная запись числа X .

2. Получаем новую запись D следующим образом: к этой записи дописываются справа ещё два разряда по следующему правилу:

а) в конец числа (справа) дописывается 1, если число единиц в двоичной записи числа чётно, и 0, если число единиц в двоичной записи числа нечётно;

б) к этой записи справа дописывается 1, если остаток от деления количества единиц на 2 равен 0, и 0, если остаток от деления количества единиц на 2 равен 1.

Укажите минимальное число D, которое превышает 54 и может являться результатом работы алгоритма. В ответе это число запишите в десятичной системе.

РЕШАЕМ В ТЕТРАДИ

Задание 43.

Виктор придумал алгоритм кодирования десятичного числа X:

1. Вначале строится двоичная запись числа X.

2. Получаем новую запись D следующим образом:

а) если число чётное, то к двоичной записи числа слева дописывается 1, а справа 0. Например, если для исходного числа 100 результатом будет являться число 11000;

б) если число нечётное, то к двоичной записи числа и слева и справа дописывается 11. Например, если для исходного числа 101 результатом будет являться число 1110111;

Укажите минимальное число X, после обработки которого с помощью этого алгоритма получается число, большее, чем 55. В ответе запишите это число в десятичной системе счисления.

ЧИТАЕМ

При построении двоичной записи числа X левые (незначащие) нули не записываются. Например, число 1 будет представлено как 1, а не как 01.

РЕШАЕМ В ТЕТРАДИ

Задание 44.

Виктор придумал алгоритм кодирования десятичного числа X:

1. Вначале строится двоичная запись числа X.

2. К этой записи справа дописываются ещё два разряда по следующему правилу:

а) складываются все цифры двоичной записи числа X, и осуществляется контроль четности. Например, 11110 преобразуется в запись 111100, а 10011 преобразуется в запись 100111;

б) над этой записью производятся те же действия дважды.

Полученная таким образом запись (в ней на два разряда больше, чем в записи исходного числа X) является двоичной кодовой записью (D) числа X. Укажите минимальное число D, которое превышает число 85 и может являться

результатом работы данного алгоритма. В ответе это число запишите в десятичной системе счисления.

РЕШАЕМ В ТЕТРАДИ

Задание 45.

Виктор придумал алгоритм кодирования десятичного числа X :

1. Вначале строится двоичная запись числа X .
2. Получаем новую запись D следующим образом:
 - а) если в полученной записи единиц больше, чем нулей, то справа приписывается единица.
 - б) если нулей больше или нулей и единиц поровну, справа приписывается ноль.
3. Полученное число переводится в десятичную запись и выводится на экран.

Пример. Дано число $X = 14$. Алгоритм работает следующим образом.

1. Двоичная запись числа X : 1110.
2. В записи больше единиц, справа приписывается единица: 11101.
3. На экран выводится десятичное значение полученного числа 29.

Какое наименьшее число, превышающее 100, может получиться в результате работы автомата?

РЕШАЕМ В ТЕТРАДИ

Задание 46.

Виктор придумал алгоритм кодирования десятичного числа X :

1. Вначале строится двоичная запись числа X .
2. К этой записи справа дописываются ещё два разряда по следующему правилу:
 - а) складываются все цифры двоичной записи числа X , и осуществляется контроль четности. Например, 11110 преобразуется в запись 111100, а 10011 преобразуется в запись 100111;
 - б) над этой записью производятся те же действия дважды.

Полученная таким образом запись (в ней на два разряда больше, чем в записи исходного числа X) является двоичной кодовой записью (D) числа X . Укажите минимальное число D , которое превышает число 55 и может являться результатом работы данного алгоритма. В ответе это число запишите в десятичной системе счисления.

§6.3 Равномерное кодирование

ВСПОМИНАЕМ

Вспомним:

1. Что такое «Контролируемая избыточность»?
2. Что мы понимаем под кратностью обнаруживаемых ошибок?

ЧИТАЕМ

Для определения количества информации различают содержательный и алфавитный подходы.

Если величина неопределенности, снимаемой некоторым сообщением, представляет собой содержащееся в сообщении количество информации, то такой подход к определению информации называют содержательным. При этом количество информации в сообщении зависит от его содержания. Содержательный подход делит сообщения на информативные и сообщения, не несущие новой, полезной информации. Если в сообщении отсутствует новая, полезная, снимающая неопределенность информация, то такое сообщение несет количество информации, равное нулю. Содержательный подход можно еще назвать субъективным, при нем трудно организовать формальную, общую для всех получателей систему определения количества информации в сообщениях.

Алфавитный подход к определению информации, в отличие от содержательного подхода, абстрагируется от содержания сообщений. Сторонник этого подхода, российский ученый А. Н. Колмогоров, рассматривает сообщение как слово, записанное символами некоторого алфавита. При этом количество информации в сообщении определяется не его содержанием, а количеством бит, необходимых для кодирования данного сообщения. Алфавитный подход хорошо формализуется, его также называют объективным.

Каждый алфавит имеет в составе конечное число символов, из которых он состоит – это мощность алфавита.

Например, мощность латинского алфавита равна 26. Мощность русского алфавита равна 32, если *е* и *ё* считать за один символ и т.д.

А как узнать длину двоичного слова, необходимого для кодирования одного символа любого алфавита?

Длина двоичного слова, необходимого для кодирования N сообщений, определяется *по формуле Хартли*, названной по имени предложившего ее американского инженера:

$$i = \log_2 N.$$

Эта формула определяет количество информации (i), которое имеем в случае получения одного из N равновероятных сообщений.

Напомним, что логарифмом называют показатель степени (i), в которую нужно возвести основание логарифма (в нашем случае – 2), чтобы получить заданное число (N). При $i = \log_2 N$ получаем $N = 2^i$.

Обозначим буквой p вероятность того, что произошло одно из N равновероятных событий. Так как $p = 1/N$, то $N = 1/p$, и формулу Хартли можно записать так: $i = \log_2 1/p$.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Условие:

Танцевальный партер Крокус Сити Холла вмещает в себя 1700 мест. Используется автоматизированная система продажи билетов. Каждый билет кодируется минимально возможным количеством бит, одинаковым для каждого места. Каков минимально возможный объем сообщения, содержащего в закодированном виде номера 20-ти забронированных мест?

Решение:

Так как каждый билет кодируется минимально возможным количеством бит, одинаковым для каждого места, то воспользуемся формулой Хартли $N = 2^i$. $N = 1700$, значит $i = 11$ бит. Тогда объем сообщения для 20-ти забронированных мест $I = 20 \cdot 11$ бит = 220 бит.

Ответ: 220 бит.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 2

Условие:

Каждый сотрудник предприятия «Луч» получает электронный пропуск, на котором записаны личный код сотрудника и срок действия пропуска. Личный код состоит из 15 символов, каждый из которых может быть одной из 26 заглавных латинских букв или 10 цифр. Для записи кода на пропуске отведено минимально возможное целое число байтов, при этом используют посимвольное кодирование, все символы кодируют одинаковым минимально возможным количеством битов. Срок действия записывается как номер года (число от 10 до 30, означающее год от 2010 до 2030) и номер месяца (число от 1 до 12).

Номер года и номер месяца записаны на пропуске как двоичные числа, каждое из них занимает минимально возможное число битов, а два числа вместе – минимально возможное число байтов. Сколько байтов занимает вся

информация на пропуске? В ответе запишите только целое число – количество байтов.

Решение:

Согласно условию, в личном коде могут быть использованы 36 символов. По формуле Хартли $N = 2^i$ имеем $2^5 < 36 < 2^6$, следовательно, для записи каждого из 15 символов необходимо 6 бит.

Для хранения всех 15 символов личного кода нужно $6 \cdot 15 = 90$ бит. Минимально возможное количество байт, вмещающее в себя 90 бит $= 12$ байт.

Для хранения номера года потребуется 5 бит, поскольку $2^5 = 32$, для хранения номера месяца – 4 бита, поскольку $2^4 = 16$. Минимально возможное количество байт, вмещающее в себя $5 + 4 = 9$ бит $= 2$ байта.

Значит, вся информация на пропуске занимает $12 + 2 = 14$ байт.

Ответ: 14.

РЕШАЕМ В ТЕТРАДИ

Задание 47.

Для регистрации на сайте пользователю требуется придумать надежный пароль. Длина пароля не менее 9 символов. В качестве символов могут быть использованы десятичные цифры, 26 различных букв латинского алфавита, причём все буквы используются в двух начертаниях: как строчные, так и прописные (регистр буквы имеет значение) и специальные символы (#, \$, %). Под хранение каждого такого пароля на компьютере отводится одинаковое и минимально возможное целое количество байтов. При этом используется посимвольное кодирование, и все символы кодируются одинаковым и минимально возможным количеством битов. Определите минимальный объём памяти, который используется для хранения 50 паролей. (Ответ дайте в байтах)

РЕШАЕМ В ТЕТРАДИ

Задание 48.

В некоторой стране автомобильный номер длиной 6 символов составляют из 12 заглавных латинских букв и десятичных цифр в любом порядке. Каждый такой номер в компьютерной программе записывается минимально возможным и одинаковым целым количеством байтов (при этом используют посимвольное кодирование и все символы кодируются одинаковым и минимально возможным количеством битов).

Определите объём памяти, отводимый этой программой для записи 170 номеров. (Ответ дайте в байтах)

РЕШАЕМ В ТЕТРАДИ

Задание 49.

Сколько спортсменов участвуют в велокроссе, если известно, что специальное устройство регистрирует прохождение каждым из участников промежуточного финиша, записывая его семизначный номер с использованием минимально возможного количества бит, одинакового для каждого из участников? После прохождения промежуточного финиша всеми спортсменами объем информации был равен 70 байтам.

РЕШАЕМ В ТЕТРАДИ

Задание 50.

При регистрации в сетевой компьютерной игре для каждого пользователя формируется индивидуальный идентификатор, состоящий из 16 символов латинского алфавита обоих регистров в произвольном порядке (26 заглавных и 26 строчных букв). В базе данных для хранения сведений о каждом пользователе отведено одинаковое минимально возможное целое число байт. При этом используют посимвольное кодирование идентификаторов, все символы кодируют одинаковым минимально возможным количеством бит. Кроме идентификатора для каждого пользователя в системе хранятся дополнительные сведения:

<Имя> – 12 символов из 26 латинских строчных букв;

<Год рождения> – числа от 1990 до 2005.

Каждое поле записывается с использованием минимально возможного количества бит, а дополнительные сведения – с использованием минимально возможного количества байт.

Сколько байт нужно для хранения сведений о 36 пользователях? В ответе запишите только целое число – количество байт.

РЕШАЕМ В ТЕТРАДИ

Задание 51.

Имеется два текста, причем во втором тексте количество символов в два раза больше, чем в первом. Первый текст составлен в алфавите мощностью 12 символов, а второй текст – в алфавите из 64 символов. Во сколько раз количество информации во втором тексте больше, чем в первом?

РЕШАЕМ В ТЕТРАДИ

Задание 52.

Объем сообщения – 22,5 Кбайт. Известно, что данное сообщение содержит 160 страниц текста, содержащего в среднем 192 символа на каждой странице. Какова мощность алфавита?

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Какой подход для определения количества информации называют содержательным?
2. Какой подход для определения количества информации называют алфавитным?
3. В чем отличие алфавитного подхода от содержательного?

§6.4 Создание алфавитно-частотного словаря

ВСПОМИНАЕМ

Вспомним:

1. Что такое мощность алфавита?
2. Как узнать длину двоичного слова, необходимого для кодирования одного символа любого алфавита?

ЧИТАЕМ

Алфавитно-частотный словарь – это набор слов данного языка вместе с информацией о частоте их встречаемости.

Для создания алфавитно-частотного словаря как правило берётся текст огромного объёма, проводится его обработка, и на выходе получают, сколько раз то или иное слово повторилось в тексте.

С такой задачей можно легко справиться, написав для нее программу на языке программирования Python.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Условие:

Напишем программу, которая построит алфавитно-частотный словарь для файла, в котором в столбик записаны слова, состоящие только из строчных букв латинского алфавита.

Входной файл *input.txt* содержит неизвестное количество строк, в каждой из которых записано слово, состоящее только из строчных букв латинского алфавита. Последняя строка файла – пустая. Программа должна вывести в алфавитном порядке все различные слова, которые встретились в файле, по одному слову в строке. После каждого слова через пробел записывается количество таких слов в файле.

Входные данные:

was
she
was
he
she
i
she

Выходные данные:

he 1
i 2
she 3
was 2

Решение:

Примерный код программы, реализующий данную задачу, может выглядеть так:

```
with open("input.txt", "r") as F:  
    s = [i.rstrip() for i in F.readlines()]  
for i in sorted(set(s)):  
    print(i, s.count(i))
```

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 2

Условие:

Входной файл input.txt состоит не более чем из 1000 символов A, S и D. Определите максимальное количество идущих подряд символов, среди которых каждые два соседних различны. Для выполнения этого задания следует написать программу.

Входные данные:

ASDAAAASSSSDDDDDDASDASASDDSDSAASD

Выходные данные:

9

Решение:

Примерный код программы, реализующий данную задачу, может выглядеть так:

```
with open("input.txt", "r") as F:  
    s = F.readline()  
m = 0  
k = 1  
for i in range(1, len(s)):  
    if s[i] != s[i-1]:  
        k = k + 1  
    if k > m:
```

```
        m = k
    else:
        k = 1
print(m)
```

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №60

Входной файл *input.txt* содержит неизвестное количество строчных букв латинского алфавита. Программа должна вывести в алфавитном порядке все различные символы, которые встретились в файле, по одному в строке. После каждого символа через пробел записывается количество таких символов в файле.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №61

Во входном файле *input.txt* находится цепочка из символов латинского алфавита A, B, C, D, E. Найдите количество цепочек длины 3, удовлетворяющих следующим условиям:

- 1-й символ – один из символов A, B, C или D;
- 2-й символ – один из символов B, D, E, который не совпадает с первым;
- 3-й символ – один из символов A, B, C, E, который не совпадает со вторым.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1

Во входном файле *input.txt* находится цепочка из символов латинского алфавита A, B, C, D, E. Найдите максимальную длину цепочки вида ABEABEABE... (составленной из фрагментов ABE, последний фрагмент может быть неполным).

Задание 2.

Цепочка из символов латинского алфавита V, W, X, Y, Z. Найдите длину самой длинной подцепочки, состоящей из символов X, Y или V (в произвольном порядке).

§6.5 Вероятностный подход к определению информации

ВСПОМИНАЕМ

Вспомним:

1. Что понимается под алфавитно-частотным словарем?
2. Что нужно сделать, чтобы создать такой словарь?

ЧИТАЕМ

Широко применяемым на практике является вероятностный подход к измерению информации. Главной отличительной особенностью вероятностного подхода от комбинаторного является тот факт, что он основан на вероятностных допущениях относительно пребывания какой-либо системы в различных состояниях. При этом общее число состояний системы не учитывается. За количество информации здесь принимается снятая неопределенность выбора из множества возможностей, имеющих, в общем случае, различную вероятность.

Вычислить количество информации для событий с различными вероятностями нам поможет формула, которую предложил в 1948 году «отец информационного века» американский инженер и математик Клод Шеннон.

Если I – количество информации, N – количество возможных событий, p_i – вероятности отдельных событий, где i принимает значения от 1 до N , то количество информации для событий с различными вероятностями можно определить по формуле:

$$I = - \sum_{i=1}^N p_i \cdot \log_2 p_i$$

можно расписать формулу в таком виде:

$$I = p_1 \cdot \log_2 \frac{1}{p_1} + p_2 \cdot \log_2 \frac{1}{p_2} + p_3 \cdot \log_2 \frac{1}{p_3} + \dots$$

Вероятность события выражается в долях единицы и вычисляется по формуле $p = K/N$, где K – величина, показывающая сколько раз произошло интересное нас событие, N – общее число возможных исходов какого-то процесса.

ОБСУЖДАЕМ

Рассматриваемые нами формулы классической теории информации первоначально были разработаны для технических систем связи, призванных служить обмену информацией между людьми. Задача оптимизации работы таких систем требовала, прежде всего, решить вопрос о количестве информации, передаваемой по каналам связи. Поэтому вполне естественно, что первые шаги в этом направлении сделали сотрудники Bell Telephone Company – Х. Найквист, Р. Хартли и К. Шеннон. Приведенные формулы

послужили К. Шеннону основанием для исчисления пропускной способности каналов связи и энтропии источников сообщений, для улучшения методов кодирования и декодирования сообщений, для выбора помехоустойчивых кодов, а также для решения ряда других задач, связанных с оптимизацией работы технических систем связи.

Возникает вопрос: можно ли применить формулу К. Шеннона для равновероятных событий?

Если:

$$p_1 = p_2 = \dots = p_N = \frac{1}{N}$$

тогда формула принимает вид:

$$I = - \sum_{i=1}^N \frac{1}{N} \cdot \log_2 \frac{1}{N} = \log_2 N$$

Мы видим, что формула Хартли является частным случаем формулы Шеннона.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Условие:

В составе поезда 20 вагонов. Среди них есть купейные и плацкартные вагоны. Сообщение о том, что ваш близкий родственник приезжает в купейном вагоне, несет $I = 2$ бита информации. Определите, сколько в поезде купейных вагонов?

Решение:

Обозначим K — искомое число купейных вагонов. Вероятность того, что знакомый приезжает в купейном вагоне, равна $p = K / N = K / 20$.

Для решения используем формулу: $I = \log_2 (1 / p)$. Для нашего примера имеем: $I = \log_2 (20 / K)$. Величина I по условию задачи равна 2. Подставим значения: $2^2 = 20 / K$, откуда получаем $K = 20 / 4 = 5$, т. е. в поезде 5 купейных вагонов.

Ответ: 5 вагонов.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 2

Условие:

В коробке находятся 24 фрукта: 12 яблок, 6 груш, 3 лимона и 3 апельсина. Какое количество информации содержится в сообщениях о том, что из мешка случайным образом были последовательно взяты с возвратом яблоко (I_1), груша (I_2), лимон (I_3) и апельсин (I_4).

Решение:

Определим вероятности утверждения "взять из коробки один фрукт":

яблоко: $p_1 = 12 / 24 = 1/2$,

грушу: $p_2 = 6 / 24 = 1/4$,

лимон: $p_3 = 3 / 24 = 1/8$,

апельсин: $p_4 = 3 / 24 = 1/8$.

После подстановки в формулу $I = \log_2 (1 / p)$ получаем: $I_1 = 1$ бит, $I_2 = 2$ бита, $I_3 = I_4 = 3$ бита.

Ответ: $I_1 = 1$ бит, $I_2 = 2$ бита, $I_3 = I_4 = 3$ бита.

При сравнении результатов вычислений получается следующая ситуация: вероятность выбора яблока больше, чем лимона или апельсина, а информации при этом получилось меньше. Это не случайность, а закономерность.

Качественную связь между вероятностью события и количеством информации в сообщении об этом событии можно выразить так: чем меньше вероятность некоторого события, тем больше информации содержит сообщение об этом событии.

Сравним теперь равновероятностные события с не равновероятностными и определим, какое сообщение содержит большее количество информации:

- В книжном стеллаже 8 полок. Книга нашлась на третьей полке. ($I = 3$ бит)
- Виктор получил за экзамен 4 балла (по 5-бальной системе единицы не ставят). ($I = 2$ бит)
- Бабушка испекла 9 пирожков с капустой, 9 пирожков с повидлом. Даша съела один пирожок. ($I = 0,5 + 0,5 = 1$ бит.)
- Бабушка испекла 8 пирожков с капустой, 10 пирожков с повидлом. Даша съела один пирожок. ($I = 0,52 + 0,47 = 0,99$ бит.)

Мы видим, что считая количество информации, мы получим большее значение, если события равновероятны.

РЕШАЕМ В ТЕТРАДИ

Задание 53.

Вы играете в шашки и сделали первый ход, причем у вас было 2 варианта хода каждой из 4 передних шашек. Сколько информации при этом получил ваш противник?

РЕШАЕМ В ТЕТРАДИ

Задание 54.

В классе 32 человека. За контрольную работу по информатике получено 8 пятерок, 17 четверок, 4 тройки и 3 двойки. Какое количество информации несет сообщение о том, что Иванов получил пятерку?

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. В каких случаях применяется формула Шеннона для измерения количества информации, а в каких формула Хартли?
2. В каком случае количество информации о событии достигает максимального значения? Приведите примеры.

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. Корогодина В.И., Корогодина В.Л. Информация как основа жизни. – Дубна: Издательский центр «Феникс», 2000. – 208 с.

§6.6 Неравномерные коды и условие Фано

ВСПОМИНАЕМ

Вспомним:

1. Какие события называются равновероятными, а какие неравновероятными? Приведите примеры.
2. Объясните качественную связь между вероятностью события и количеством информации в сообщении об этом событии?

ЧИТАЕМ

Наряду с равномерным кодированием, которое нельзя назвать оптимальным, существует более гибкий и менее объемный метод кодирования, который называется неравномерным. При таком кодировании число битов, отводящихся для кодировки разных символов, имеет разную длину в двоичном формате. Но при этом должно соблюдаться основное правило однозначного и правильного декодирования закодированной таким образом информации.

Для обеспечения однозначного декодирования текстовой информации, закодированной с помощью неравномерного кодирования, такие коды должны иметь значения согласно условиям Фано.

Условие Фано названо в честь его создателя, итальянско-американского ученого Роберта Фано. Условие является необходимым в теории кодирования при построении префиксного кода.

Данное условие формулируется следующим образом: «ни одно кодовое слово не может выступать в качестве начала любого другого кодового слова».

Существует также и обратное условие Фано: «ни одно кодовое слово не может выступать в качестве окончания любого другого кодового слова».

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Условие:

Дана последовательность, состоящая из букв «А», «Б», «В», «Г» и «Д». Для кодирования приведенной последовательности применяется неравномерный двоичный код, при помощи которого можно осуществить однозначное декодирование.

Известны двоичные эквиваленты букв: А-00, Б-010, В-011, Г-101, Д-111.

Можно ли для одного из символов Б, В или Г сократить длину кодового слова до 01, чтобы сохранить возможность однозначного декодирования? При этом коды остальных символов должны остаться неизменными.

Решение:

Для того, чтобы сохранилась возможность декодирования, достаточным является соблюдение прямого или обратного условия Фано.

Проведем последовательную проверку.

1. Пусть Б будет иметь код 01. Тогда имеем: А-00, Б-01, В-011, Г-101 и Д-111. Видим, что *прямое условие Фано* не выполняется: код символа «Б» совпадает с началом кода символа «В». *Обратное условие Фано* также не выполняется: код символа «Б» совпадает с окончанием кода символа «Г».

2. Пусть В будет иметь код 01. Тогда имеем: А-00, Б-010, В-01, Г-101 и Д-111. *Прямое условие Фано* не выполняется: код символа «В» совпадает с началом кода символа «Б». *Обратное условие* также не выполняется: код символа «В» совпадает с окончанием кода символа «Г».

3. Пусть Г будет иметь код 01. Тогда имеем: А-00, Б-010, В-011, Г-01 и Д-111. *Прямое условие Фано* не выполняется: код символа «Г» совпадает с началом кода символов «Б» и «В». Однако наблюдается выполнение обратного правила Фано: код символа «Г» не совпадает с окончанием кода всех остальных символов.

После проверки возможных вариантов решения задачи на соответствие прямому и обратному условию *Фано*, было установлено, что можно сократить длину буквы Г до 01.

Ответ: Г-01.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 2

Условие:

Для передачи по каналу связи сообщения, состоящего только из символов О, П, Р и Т, используется посимвольное кодирование: О-00, П-11, Р-010, Т-011. Через канал связи передаётся сообщение: ТОПОР. Закодируйте сообщение данным кодом. Полученное двоичное число переведите в восьмеричный код.

Решение:

Закодируем сообщение: ТОПОР – 011001100010. Теперь разобьём это представление на триады (тройки) справа налево и переведём полученный набор чисел в восьмеричный вид:

011 001 100 010 – 3 1 4 2.

Ответ: 3142

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

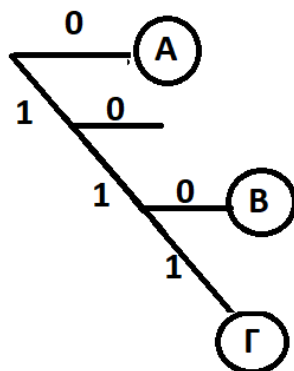
ПРИМЕР 3

Условие:

Для передачи по каналу связи сообщения, состоящего только из букв А, Б, В, Г, решили использовать неравномерный по длине код, причем известно, что А-0, В-110, Г-111. Как нужно закодировать букву Б, чтобы длина кода была минимальной и допускалось однозначное разбиение закодированного сообщения на буквы?

Решение:

Построим дерево:



Ответ: 10

ЧИТАЕМ

Код Хаффмана

Идея, положенная в основу кодирования Хаффмана, основана на частоте появления символа в последовательности: символ, который встречается в последовательности чаще всего, получает маленький код, а символ, который встречается реже всего, получает длинный код. Это нужно для уменьшения длины кодового слова, т.е. для оптимального кодирования.

Под оптимальным кодированием будем понимать такое кодирование символов в сообщении, которое приводит к минимально возможной длине кода этого сообщения. Оптимальному кодированию, как правило, соответствует минимальное время передачи сообщения и минимальный объем памяти для его хранения.

При таком кодировании возникает проблема определения при передаче конца кода одного символа и начала кода другого символа. Проблема решается кодированием символов алфавита префиксными кодами, у которых код каждого символа не является началом кода другого символа заданного алфавита. В этом случае последовательное побитовое считывание сообщения приводит к однозначному определению закодированных символов и полному восстановлению переданного сообщения. Среди множества алгоритмов, кодирующих каждый символ отдельно от других символов, наилучшее сжатие дает алгоритм Хаффмана.

Пусть задан алфавит из четырех символов (A, B, C и D) с частотой повторения в сообщениях соответственно 7, 6, 3 и 1. Для оптимального кодирования этих символов в двоичном алфавите применяют алгоритм Хаффмана, работу которого поясним на рисунке 6.1.

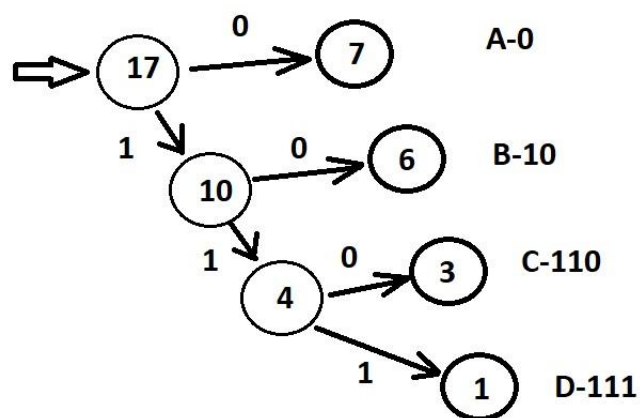


Рис. 6.1 Алгоритм Хаффмана

В выделенных кружках записаны заданные частоты появления в сообщениях четырех символов заданного алфавита. После записи этих четырех вершин начинаем строить граф (дерево), который определит искомые

коды. Сначала рисуется вершина графа, отмеченная суммой двух наименьших частот символов ($4 = 3 + 1$). Вершина 4 соединяется дугами с вершинами 1 и 3. Верхние дуги отмечаются нулем, нижние — единицей. Аналогично строится вершина 10, затем вершина 17. Проводятся и отмечаются дуги. Путь от вершины 17 к каждой из четырех заданных вершин отмечен кодом, которым кодируется соответствующий символ. Коды записаны справа от вершин.

При использовании метода Хаффмана процесс передачи сообщения делится на два этапа. Сначала сообщение просматривается с целью определения частотных характеристик символов. В соответствии с результатом анализа выполняется кодирование символов. После этого сообщение передается по каналу связи. На приемном конце канала связи с помощью дерева Хаффмана выполняется восстановление закодированных данных.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 4

Условие:

Требуется закодировать в заданном четырехбуквенном алфавите (А, В, С, D) слово DACAB.

Решение:

Для этого, подавая в вершину 17 дерева Хаффмана букву за буквой и записывая коды букв, получим 1110110010. При раскодировании действуем в обратном порядке. В вершину 17 подаем кодирующий слово код и записываем буквы слова: DACAB.

РЕШАЕМ В ТЕТРАДИ

Задание 55.

Для кодирования букв О, В, Д, П, А, Л решили использовать двоичное представление чисел 0, 1, 2, 3, 4 и 5 соответственно (с сохранением одного незначащего нуля в случае одноразрядного представления). Закодируйте последовательность букв ПОДВАЛ таким способом, результат запишите и восьмеричным, и шестнадцатеричным кодом.

РЕШАЕМ В ТЕТРАДИ

Задание 56.

Для кодирования сообщения, состоящего только из букв А, Б, В и Г, используется неравномерный по длине двоичный код: причем известно, что А-010, Б-011. Закодируйте буквы В и Г таким способом, чтобы закодированная последовательность символов ВГАГБВГ имела наименьшую двоичную длину.

Для решения постройте дерево Хаффмана.

В ответе через запятую напишите коды букв В и Г и минимальную длину закодированной последовательности.

РЕШАЕМ В ТЕТРАДИ

Задание 57.

Для кодирования некоторой последовательности, состоящей из букв А, S, D, F, G, решили использовать неравномерный двоичный код, удовлетворяющий прямому условию Фано. Это условие обеспечивает возможность однозначной расшифровки закодированных сообщений. Для букв А, S, D использовали соответственно кодовые слова 00, 01, 11. Для двух оставшихся букв – F, G – кодовые слова неизвестны. Укажите кратчайшее возможное кодовое слово для буквы F, при котором код будет удовлетворять указанному условию. Если таких кодов несколько, укажите код с наименьшим числовым значением.

РЕШАЕМ В ТЕТРАДИ

Задание 58.

По каналу связи с помощью равномерного двоичного кода передаются сообщения, содержащие только 4 буквы: А, В, С, D; для кодировки букв используются кодовые слова длины 5. При этом для набора кодовых слов выполнено такое условие: любые два слова из набора отличаются не менее чем в трёх позициях. Это свойство важно для расшифровки сообщений при наличии помех. Для кодирования букв А, В, С используются 5-битовые кодовые слова: А-01111, В-00001, С-11000.

Определите 5-битовое кодовое слово для буквы D, если известно, что оно начинается с 1 и заканчивается 0.

РЕШАЕМ В ТЕТРАДИ

Задание 59.

По каналу связи передаются сообщения, каждое из которых содержит 16 букв X, 8 букв Y, 4 буквы Z и 4 буквы V (других букв в сообщениях нет). Каждую букву кодируют двоичной последовательностью. При выборе кода учитывается прямое условие Фано. Кроме того, общая длина закодированного сообщения должна быть минимальной.

Какой код из приведённых ниже следует выбрать для кодирования букв X, Y, Z и V?

- 1) X-00, Y-01, Z-10, V-11
- 2) X-0, Y-10, Z-01, V-11
- 3) X-1, Y-01, Z-011, V-001
- 4) X-0, Y-10, Z-110, V-111

РЕШАЕМ В ТЕТРАДИ

Задание 60.

По каналу связи передаются сообщения, содержащие только 4 буквы Г, О, С, Т; для передачи используется двоичный код, допускающий однозначное декодирование. Для букв Т, О, Г используются такие кодовые слова: Т-111, О-0, Г-100.

Укажите кратчайшее кодовое слово для буквы С, при котором код будет допускать однозначное декодирование. Если таких кодов несколько, укажите код с наибольшим числовым значением.

РЕШАЕМ В ТЕТРАДИ

Задание 61.

Для кодирования некоторой последовательности, состоящей из букв А, В, С, D и Е, используется неравномерный двоичный код, позволяющий однозначно декодировать полученную двоичную последовательность. Вот этот код: А – 0; В – 100; С – 1010; D – 111; Е – 110. Требуется сократить для одной из букв длину кодового слова так, чтобы код по-прежнему можно было декодировать однозначно. Коды остальных букв меняться не должны.

Каким из указанных способов это можно сделать?

- 1) для буквы С – 010
- 2) это невозможно
- 3) для буквы С – 101
- 4) для буквы В – 10

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Почему неравномерное кодирование считают более оптимальным по сравнению с равномерным кодированием?
2. В чем заключается смысл прямого и обратного условия Фано?
3. Когда нужно использовать код Хаффмана, и в чем его суть?

§6.7 Сжатие. Алгоритм RLE

ВСПОМИНАЕМ

Вспомним:

1. Как можно обеспечить однозначное декодирование текстовой информации?
2. Сформулируйте прямое и обратное условие Фано.

ЧИТАЕМ

Информация, передаваемая по каналам связи, как мы уже говорили ранее, обладает некоторой долей избыточности. Клод Шеннон определил, что избыточность английского языка составляет около 50%. Близкие к этому оценки можно ожидать и для русского языка. Это означает, что если исключить из текста каждый второй символ, то текст с большой долей вероятности можно прочесть и восстановить.

У аудио- и видеоинформации степень избыточности во много раз больше, чем у текстовой информации. Чем больше степень избыточности информации, тем больше допускается степень ее сжатия. Из методов сжатия информации можно выделить JPEG — метод сжатия статических (например, фотографических) изображений, MPEG — методы сжатия видеоданных, MP3 — метод сжатия аудиоинформации (звуковых данных). Многие методы сжатия информации используют алгоритм Хаффмана.

Рассмотрим еще один метод сжатия данных – метод сжатия RLE (Run Length Encoding), или, в русском переводе, кодирование длин серий или кодирование повторов – это, наверное, самый простейший алгоритм сжатия данных, в котором повторяющиеся символы (серии, т.е. последовательности, состоящая из нескольких одинаковых символов) заменяются на один символ и число его повторов.

Алгоритм RLE – это алгоритм сжатия данных, который поддерживается большинством форматов файлов растровых изображений: TIFF, BMP и PCX. RLE подходит для сжатия любого типа данных независимо от его информационного содержания, но содержание данных влияет на коэффициент сжатия. Несмотря на то, что большинство алгоритмов RLE не могут обеспечить высокие коэффициенты сжатия более сложных методов, данный инструмент легко реализовать и быстро выполнить, что делает его хорошей альтернативой.

Сам алгоритм сжатия состоит в следующем:

1. начать с первого символа
2. добавить его в строку результата
3. посчитать число повторений символа и добавить это число в строку результата
4. взять следующий символ и повторять до конца исходной строки.

RLE работает, уменьшая физический размер повторяющейся строки символов. Эта строка, называемая run, обычно кодируется в два байта. Первый байт представляет количество символов в пробеге и называется счетчиком прогона. На практике кодированный прогон может включать от 1 до 128 и 256 символов. Счетчик обычно содержит число символов минус один (значение в

диапазоне значений от 0 до 127 и 255). Второй байт — это значение символа в прогоне, которое содержится в диапазоне значений от 0 до 255 и именуется значением запуска.

Без сжатия символьный пробег в 15 символов обычно требует 15 байтов для хранения: MMMMMMMMMMMMMMMM.

В той же строке после RLE-кодирования потребуется только два байта: 15M.

Кодировка 15M, сгенерированная для обозначения символьной строки, называется RLE-пакетом. В данном коде первый байт, 15, является счетчиком прогона и содержит необходимое количество повторений. Второй байт, M, является значением run и содержит фактическое повторяющееся значение в пробеге.

Новый пакет генерируется каждый раз, когда символ запуска изменяется, или каждый раз, когда количество символов в пробеге превышает максимальное количество. Предположим, что 15-символьная строка содержит 4 разных символа: AAAAAAbbbXXXXXt.

При использовании кодировки с длиной строки, она может быть сжата в четыре двухбайтовых пакета: 6A3b5X1t.

После кодирования по длине строки для 15-байтовой строки потребуется всего восемь байтов данных для представления строки в отличие от исходных 15 байтов. В этом случае кодирование во время выполнения давало коэффициент сжатия почти 2 к 1.

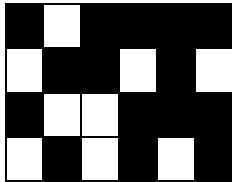
Длинные прогоны редки в некоторых типах данных. Например, открытый текст ASCII редко содержит длинные прогоны. В предыдущем примере последний пробег (содержащий символ t) был только одним символом в длину. 1-символьный прогон все еще работает. И счет запуска, и значение запуска должны быть записаны для каждого пробега в 2 символа. Для кодирования пробега с помощью алгоритма RLE требуется информация, состоящая не менее чем из двух символов. В связи с чем запуск одиночных символов на самом деле занимает больше места. По тем же причинам данные, состоящие полностью из 2-символьных прогонов, остаются неизменными после кодирования RLE.

Схемы алгоритма сжатия RLE отличаются простотой и высокой скоростью выполнения, но их эффективность зависит от типа кодируемых данных изображения. Черно-белое изображение, которое в основном белого цвета, например, страницы книги, будет очень хорошо кодироваться из-за большого количества смежных данных, имеющих одинаковый цвет. Однако изображение со многими цветами, например, фотография, не будет кодироваться так же хорошо. Это связано с тем, что сложность изображения выражается в виде большого количества разных цветов. И из-за этой сложности будет относительно мало прогонов одного цвета.

РЕШАЕМ В ТЕТРАДИ

Задание 62.

Черно-белое растровое изображение размером 4х6 кодируется построчно, начиная с левого верхнего угла и заканчивая в правом нижнем углу. При кодировании 1 обозначает черный цвет, а 0 – белый.



Используя метод RLE, запишите результат кодировки. Ответ дайте в одной строке без разделительных знаков.

РЕШАЕМ В ТЕТРАДИ

Задание 63.

В результате построчного кодирования черно-белого изображения размером 5х5 был получен следующий код:

511031101110111011103110201120.

Расшифруйте данное сообщение и воспроизведите исходное изображение.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Восстановите текст:

Б_л_е_п_р_с_о_и_о_и_

В_т_м_н_м_р_г_л_б_м!..

Кто автор этих строк?

2. Для чего используют сжатие информации и какие методы вы знаете?
3. В чем заключается суть метода сжатия RLE?

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. Алгоритм RLE: описание, особенности, правила и примеры. URL: <https://ruud.ru/it/18909-algorithm-rle-opisanie-osobennosti-pravila-i-primery/>

§6.8 Формат ZIP

ВСПОМИНАЕМ

Вспомним:

1. Для чего используется сжатие данных?
2. В чём заключаются особенности алгоритма RLE?

ЧИТАЕМ

Файл ZIP является распространенным форматом сжатия, применяющимся в различных архиваторах. Расширение ZIP разработано основателем корпорации PKWARE Филом Кацем в 1989 году специально для применения в рамках программного обеспечения PKZIP.

В таком архиве, могут содержаться данные с различной степенью сжатия. Преимуществом технологии сжатия файлов, которую использует формат ZIP, является отсутствие потери информации. Архив несет в себе как один, так и несколько файлов либо каталогов.

Примечательно, что в случае необходимости, ZIP-архивы можно соединять с модулями, отвечающими за распаковку контента, в таком случае пользователь получает единый исполняемый файл, несущий расширение EXE. Такие операционные системы, как Windows, Mac OS и Linux, а также свободные платформы, поддерживают формат в штатном режиме.

Технология доступна для внушительного ряда приложений, выполняющих функции резервного копирования, сжатия, обмена данными. Даже учитывая более инновационные технологии, в нынешнее время описываемый формат архивов всё еще актуален.

Обилие архиваторов от конкурентов, среди которых и такие программные комплексы, как B1 Free Archiver, Apple Archive Utility, RARLAB WinRAR и прочие, способны открыть файлы ZIP.

Возможность сжатия файлов очень важна для цифрового рабочего пространства. Мы можем отправлять больше данных на более высоких скоростях, чем когда-либо, поэтому ZIP-файлы являются таким популярным бизнес-инструментом по всему миру.

ZIP-файлы кодируют информацию в меньшее количество битов, тем самым уменьшая размер файла или файлов за счет удаления избыточных данных. Это так называемое «сжатие данных без потерь», которое гарантирует сохранность всех исходных данных. Давайте рассмотрим небольшой пример, чтобы понять, как это работает.

ОБСУЖДАЕМ

1. Так что на самом деле представляет собой zip-файл?
2. Как можно определить, что это архивный zip-файл?

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Условие:

Представим файл, который содержит следующие предложения:

- The best sharing and storage solution for your business (эффективное бизнес-решение для хранения файлов и предоставления общего доступа).
- Your business solution for the best sharing and storage (бизнес-решение для эффективного хранения файлов и предоставления общего доступа).

Необходимо зашифровать этот файл форматом ZIP.

Решение:

Каждое слово в этом файле появляется дважды. Если каждый символ и пробел в файле равны одной единице памяти, то размер всего файла будет составлять 110 единиц. Однако, если вы создадите нумерованный код для файла, данные можно выразить другим способом:

- 1 – The
- 2 – best
- 3 – sharing
- 4 – and
- 5 – storage
- 6 – solution
- 7 – for
- 8 – your
- 9 – business

Если записать его по-другому, предложение будет выглядеть так: 123456789896712345. Это означает, что первоначальный размер файла в 110 единиц можно уменьшить всего до 18 единиц, что является значительной экономией. Формат файла zip использует алгоритмы сжатия без потерь именно для этого, позволяя вам выразить ту же информацию более эффективным способом, просто удалив избыточные данные из файла.

ОБСУЖДАЕМ

Обсудите. Если необходимо, осуществите поиск в сети Интернет:

1. Каковы преимущества и недостатки формата файла zip?
2. Что такое 7z-файл?
3. Какие существуют альтернативы zip-файлам?

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Что такое ZIP файл?

2. Как вы понимаете фразу «сжатие данных без потерь»? Как реализует данный метод формат сжатия ZIP?
3. Какой архиватор используете вы в своей работе? Расскажите о его достоинствах и недостатках.

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. Файл формата ZIP — что это? – URL: <https://filesreview.com/ru/info/zip>
2. Формат файла ZIP – описание, как открыть? – URL: <https://www.azfiles.ru/extension/zip.html>
3. .ZIP Расширение файла – URL: <https://www.reviversoft.com/ru/file-extensions/zip>

§6.9 Контроль по теме "Неравномерное кодирование, помехоустойчивые коды"

ВАРИАНТ 1

- 1) Виктор придумал алгоритм кодирования десятичного числа X :
 1. Вначале строится двоичная запись числа X .
 2. Получаем новую запись D следующим образом: к этой записи дописываются справа ещё два разряда по следующему правилу:
 - а) в конец числа (справа) дописывается 1, если число оканчивается на ноль, и 0, если число оканчивается на 1;
 - б) к этой записи справа дописывается 1, если остаток от деления количества единиц на 2 равен 0, и 0, если остаток от деления количества единиц на 2 равен 1.Укажите минимальное число D , которое превышает 135 и может являться результатом работы алгоритма. В ответе это число запишите в десятичной системе.
- 2) Объем сообщения – 34,6 Кбайт. Известно, что данное сообщение содержит 110 страниц текста, содержащего в среднем 420 символов на каждой странице. Какова мощность алфавита?
- 3) При регистрации в сетевой компьютерной игре «Stardust» для каждого пользователя формируется индивидуальный идентификатор, состоящий из 23 символов латинского алфавита обоих регистров в произвольном порядке (26 заглавных и 26 строчных букв). При этом

используют посимвольное кодирование идентификаторов, все символы кодируют одинаковым минимально возможным количеством бит.

Кроме идентификатора для каждого пользователя в системе хранятся дополнительные сведения:

<Имя> – 8 символов из 26 латинских строчных букв;

<Год рождения> – числа от 1990 до 2020.

Каждое поле записывается с использованием минимально возможного количества бит.

В базе данных для хранения сведений о каждом пользователе отведено одинаковое минимально возможное целое число байт.

Сколько Кбайт нужно для хранения сведений о 175 пользователях? В ответе запишите только целое число – количество Кбайт.

4) Виктор за четверть по всем предметам получил 12 пятерок, несколько четверок, 3 тройки и 2 двойки. Сколько четверок получил Виктор, если известно, что вероятность получения “3” равна $0,125$.

5) Для кодирования слова КОЛПАК решили использовать неравномерный двоичный код, удовлетворяющий условию Фано. Для буквы Л использовали кодовое слово 0, для буквы П – кодовое слово 110. Какова наименьшая возможная суммарная длина этого кодового слова?

6) По каналу связи передаются сообщения, каждое из которых содержит 16 букв А, 8 букв Б, 4 буквы В и 2 буквы Г (других букв в сообщениях нет). Каждую букву кодируют двоичной последовательностью. При выборе кода учитывались два требования:

а) ни одно кодовое слово не является началом другого (это нужно, чтобы код допускал однозначное декодирование);

б) общая длина закодированного сообщения должна быть как можно меньше.

Какой код следует подобрать для кодирования букв А, Б, В и Г с целью получения наименьшей длины кодируемого сообщения?

ВАРИАНТ 2

1) Виктор придумал алгоритм кодирования десятичного числа X :

1. Вначале строится двоичная запись числа X .

2. Получаем новую запись D следующим образом: к этой записи дописываются справа ещё два разряда по следующему правилу:

а) к этой записи справа дописывается 1, если остаток от деления количества единиц на 2 равен 0, и 0, если остаток от деления количества единиц на 2 равен 1;

б) в конец числа (справа) дописывается 1, если число оканчивается на ноль, и 0, если число оканчивается на 1.

Укажите минимальное число D, которое превышает 116 и может являться результатом работы алгоритма. В ответе это число запишите в десятичной системе.

2) Алфавит некоторого языка состоит из 64 букв и символов. Объем сообщения, составленного из символов данного алфавита, содержит 22,5 Кбайт. Известно, что данное сообщение содержит 180 страниц текста. Сколько в среднем символов на каждой странице?

3) При регистрации в сетевой компьютерной игре «Team X» для каждого пользователя формируется индивидуальный идентификатор, состоящий из 14 символов латинского алфавита обоих регистров в произвольном порядке (26 заглавных и 26 строчных букв). При этом используют посимвольное кодирование идентификаторов, все символы кодируют одинаковым минимально возможным количеством бит.

Кроме идентификатора для каждого пользователя в системе хранятся дополнительные сведения:

<Имя> – 10 символов из 26 латинских строчных букв;

<Год рождения> – числа от 2000 до 2021.

Каждое поле записывается с использованием минимально возможного количества бит.

В базе данных для хранения сведений о каждом пользователе отведено одинаковое минимально возможное целое число байт.

Сколько байт нужно для хранения сведений о 150 пользователях? В ответе запишите только целое число – количество байт.

4) В коробке лежат кубики: 8 красных, несколько зеленых, 7 желтых, 12 синих. Определите количество зеленых кубиков, если количество информации, которое будет получено при вытаскивании зеленого кубика, равно 2 бит.

5) Для кодирования слова ПАРХОД решили использовать неравномерный двоичный код, удовлетворяющий условию Фано. Для буквы П использовали кодовое слово 00, для буквы Р – кодовое слово 11. Какова наименьшая возможная суммарная длина этого кодового слова?

б) По каналу связи передаются сообщения, каждое из которых содержит 6 букв А, 18 букв Б, 2 буквы В и 1 буква Г (других букв в сообщениях нет). Каждую букву кодируют двоичной последовательностью. При выборе кода учитывались два требования:

а) ни одно кодовое слово не является началом другого (это нужно, чтобы код допускал однозначное декодирование);

б) общая длина закодированного сообщения должна быть как можно меньше.

Какой код следует подобрать для кодирования букв А, Б, В и Г с целью получения наименьшей длины кодируемого сообщения?

VII. Сложность алгоритмов

§7.1 Тестирование и отладка, поиск ошибок в программе

ВСПОМИНАЕМ

Вспомним:

1. Что такое алгоритм?
2. Какие свойства алгоритмов вам известны?

ЧИТАЕМ

Тестирование и отладка. Понятия тестирования и отладки программ можно рассматривать как важную часть культуры программиста, так и как естественную часть процесса программирования в общем случае, ведь основная суть данных понятий заключается в непосредственном обнаружении проблем в работе написанных программ и их исправлении. Культурная составляющая этих понятий находится в сфере используемых инструментов и технологий для поиска и исправления ошибок, естественную (или наивную) составляющую можно описать простой необходимостью нахождения и исправления ошибок, с которой каждый из вас наверняка сталкивался, но выбор инструментов для решения подобных проблем может быть выбран не оптимально. Ниже мы с вами разберём основные принципы поиска и исправления ошибок, то есть займёмся культурным развитием программистских навыков.

В первую очередь необходимо определиться с понятиями тестирования и отладки.

Тестирование – это процесс выполнения программы с наблюдением с целью обнаружения некорректной работы.

Отладка – это средства установления конкретных причин некорректной работы программы с целью их исправления.

Этапы тестирования

Первый этап. При тестировании в первую очередь необходимо проверить работу на основе данных, наиболее характерных для обработки созданным вами программным обеспечением. То есть, если ваша программа должна работать с натуральными числами, то её необходимо в первую очередь проверить именно на них.

Второй этап. Важно рассмотреть так же и экстремальные (или крайние) наборы данных, которые предположительно могут не укладываться в общий алгоритм, если нет чёткого математического доказательства его верности. Так же в качестве экстремальных данных можно понимать и отсутствие данных вообще.

Третий этап. Ваша программа не должна работать с данными, на которые она не рассчитана, иначе может иметь место процесс нецелесообразного использования программного обеспечения, что приведёт пользователя в тупик. То есть если ваша программа работает с натуральными числами, она не должна работать с вещественными или целыми отрицательными числами, даже если результат обработки таких данных будет правдоподобным. Конечно, данный тезис не стоит доводить до абсурда и при решении, например, олимпиадных задач не стоит ставить ограничения – это лишь отнимет у вас время. Но при создании проектного продукта стоит провести тесты на данных, на которые программа не рассчитана, это может помочь вам найти ошибки, которые ранее не были обнаружены.

Важное правило. При этом при тестировании необходимо подбирать данные таким образом, чтобы каждый тестовый набор был специфичен по отношению к другим – это позволит рассматривать количественно и качественно разные ситуации, при этом тесты должны конструироваться так, чтобы их обработка программой задействовала как можно больше элементов написанного алгоритма.

Придерживайтесь этих правил при тестировании своих программ.

Отладчик

Отладка как правило заключается в рассмотрении работы конкретных участков кода во время обработки данных. Для этого чаще всего используются программные инструменты среды разработки или компилятора-интерпретатора-транслятора, которые чаще всего имеют название *отладчик* (*debugger*). Отладчик позволяет проводить исследование внутреннего поведения программы:

пошаговое выполнение программы или подпрограммы;

демонстрацию значений переменных или вычисляемых выражений;

остановку выполнения программ на «точках остановки», которые пользователь-отладчик может устанавливать произвольно.

Обычно к процессу отладки прибегают при выявлении некорректности работы программы в результате тестирования с целью установления конкретных затруднений обработки данных для восстановления функциональности всего программного обеспечения. Правила тестирования, обозначенные выше, стоит применять и при отладке.

ОБСУЖДАЕМ

Сравните понятия тестирования и отладки. В чём заключается их существенная разница, а в чём – сходство?

ЧИТАЕМ

Обработка исключений. Конструкция try-except. Проблемы в работе программ, которые допускают возможность продолжения работы программ в рамках общей структуры, называются исключениями. Примерами подобных проблем являются деление на ноль, обращение к области памяти с закрытым доступом, попытка выделить память при отсутствии памяти и т. д. С перечнем всех исключений можно ознакомиться в документации используемого языка программирования (в нашем случае Python).

Для того, чтобы при столкновении в своем выполнении с исключением программа могла продолжить функционировать, данное исключение должно быть предвидено программистом и обработано с помощью написанного кода.

В языке программирования Python существует сложная иерархия исключений, состоящая в общей сложности из около 50 системных исключений, открытая для создания своих собственных исключений пользователем-программистом.

ГОТОВИМСЯ К РАБОТЕ НА ПК

Рассмотрим несколько исключений с помощью интерактивного интерпретатора Python (примеры 1, 2).

Пример 1

```
>>> a = int(input())
Traceback (most recent call last):
  File "<pyshell>", line 1, in <module>
    a = int(input())
NameError: name 'input' is not defined
```

Пример 2

```
>>> 5 // 0
Traceback (most recent call last):
  File "<pyshell>", line 1, in <module>
    5 // 0
ZeroDivisionError: division by zero
```

В примере 1 представлена синтаксическая ошибка, в примере 2 – исключение (деление на 0).

Обработка исключений в языке программирования Python имеет определенную структуру (пример 3).

Пример 3

```
try:
    # блок кода, в котором планируется
    # отловить исключение
except SomeException:
```

```

    # блок кода, который необходимо выполнить,
    # если в блоке try было найдено
    # исключение SomeException
except OneElseException:
    # блок для исключения OneElseException
...
except LastOneexception:
    # последнее возможное исключение
else:
    # блок кода, который выполнится в том случае,
    # если исключение не было поймано
finally:
    # блок кода, который выполнится в любом случае

```

В примере 3 показано, что блоков эксепт может быть сколько угодно – синтаксис языка не накладывает на это ограничения. После эксепт пишется имя предполагаемого исключения, можно написать просто `Exception`, тогда будут пойманы любые исключения, которые возникнут в блоке `try`, но исключения не будут конкретизированы, что затруднит отладку.

После блоков эксепт идут два необязательных блока – `else` и `finally`. Код в блоке `else` выполнится только в том случае, если код в блоке `try` был выполнен без исключений и ошибок. Код в блоке `finally` будет выполнен в любом случае.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №62

Создайте в одной директории файл `'file.txt'` и файл со следующим кодом:

```

file = open('file.txt', 'r')
a = file.readline()
b = file.readline()
res = int(a) // int(b)

```

Попробуйте разместить в первых строках файла `'file.txt'` данные разной природы – целые числа, вещественные числа, строки. С каждым из наборов запустите вашу программу – посмотрите, что будет получаться.

ГОТОВИМСЯ К РАБОТЕ НА ПК

Если мы не уверены во внутренней природе файла `'file.txt'`, то можно предположить, что там могут быть записаны не числа в каждой строке, что, само собой, скажется на невозможности приведения данных в переменных `a` и `b` к целочисленному типу данных (`TypeError`). А если там всё же записаны числа, то никто не гарантирует того, что в переменную `b` будет записано не

число 0 (ZeroDivisionError). Так же, есть вероятность того, что такого числа и вовсе нет (FileNotFoundError).

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Данные ошибки можно обработать.

```
res: int # переменная создаётся заранее
        # с целью сохранения статуса видимости

try:
    file = open('file.txt', 'r')
    a = file.readline()
    b = file.readline()
    res = int(a) // int(b)
except TypeError:
    print('Из файла не было получено целое число')
except ZeroDivisionError:
    print('Было произведено деление на ноль')
except FileNotFoundError:
    print('Не удалось найти файл')
else:
    print(res)
```

На самом деле можно несколько упростить данный код. Как мы сказали, можно отлавливать вообще все возможные ошибки, и даже есть способ их идентификации (пример 6).

```
res: int # переменная создаётся заранее
        # с целью сохранения статуса видимости

try:
    file = open('file.txt', 'r')
    a = file.readline()
    b = file.readline()
    res = int(a) // int(b)
except Exception as exc:
    print(exc) # в данном случае мы получим имя
               # того исключения, которое было
               # отловлено
else:
    print(res)
```

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №63

С помощью инструмента отладки рассмотрите работу программы, код которой представлен ниже. Определите, каково назначение данной программы?

```
n = int(input())

res: int = 0
for i in range(n):
    if n % i == 0:
        res = i

print(res)
```

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1.

Определите, какое исключение было выдано системой при выполнении кода ниже. С помощью средства отладки исправьте код таким образом, чтобы данной ошибки не возникало, не меняя код содержательно – алгоритм должен остаться тот же.

```
n = int(input())

res: float = 0.0
for i in range(n - 1, -1, -1):
    res += 1 / i

print(res)
```

Сравните два варианта кода, решающих одну задачу, представленных ниже. Скажите, можно ли избежать использования конструкции try-except при реализации данного алгоритма? Если можно, то приведите свой исправленный алгоритм. Докажите его работоспособность с помощью автоматического переборного теста. Обратите внимание, что в переменной `res` хранится вещественное число – так что будьте аккуратны при сравнении данных.

Вариант 1

```
n = int(input())

res: int = 0
```

```

for i in range(n):
    if n % i == 0:
        res = i

print(res)

```

Вариант 2

```

n = int(input())

res: float = 0.0
for i in range(n - 1, -1, -1):
    try:
        res += 1 / i
    except Exception as ex:
        print(f'Exception: {ex}')

print(res)

```

3. Проиллюстрируйте на примере созданного оригинального вами кода обработку ошибки:

- a) IndexError;
- b) TypeError;
- c) ValueError;
- d) FileNotFoundError;
- e) ZeroDivisionError.

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. <https://docs.python.org/3/tutorial/errors.html>
2. Ознакомиться со стандартом тестирования и отладки ISO 9126 (<https://www.iso.org/standard/35733.html>)
3. Ознакомьтесь с информацией из документации языка программирования Python (<https://docs.python.org/3/tutorial/errors.html>), а так же ознакомьтесь с основными классами исключений (<https://docs.python.org/3/library/exceptions.html>).

§7.2 Вычислимые функции и универсальные исполнители

ВСПОМИНАЕМ

Давайте вспомним:

1. Что такое алгоритм?
2. Какие свойства есть у алгоритма?
3. Какие основные алгоритмические конструкции существуют?

ЧИТАЕМ

Вычисление и алгоритм. В данном уроке мы с вами начнём работать с такой математической дисциплиной, как *теория алгоритмов (теория вычислений – англ. computation theory)*. Несмотря на то, что ключевым в теории алгоритмов является понятие *алгоритма*, всё же объектом исследования данной дисциплины является *вычисление*. Так как именно оно является *необходимым условием* для возникновения алгоритма, реализующего конкретное вычисление. При этом конкретное вычисление может быть реализовано в рамках разных алгоритмов.

Теория вычислений задаёт несколько вопросов:

- может ли конкретная проблема быть решена?
- если проблема может быть решена, то:
 - с какой эффективностью?
 - с какой точностью?

Под проблемой понимается некоторая математическая функция, которую, в случае возможности получения её результата (выходных данных) при известном наборе аргументов (входных данных), мы будем называть *вычислимой функцией*.

Вопросы вычислимости, эффективности и точности вычисления (для вычислимой функции) функций проверяются с помощью *моделей вычислений*.

Модель вычислений – это математическая модель, определяющая множество операций и правила их использования для вычисления значения (выходных данных) математической функции при известном наборе аргументов (входных данных). То есть именно модель вычислений порождает правила формального описания вычисляемых функций и составления последовательности действий для их вычисления.

При определении модели вычислений должны быть определены:

- система команд (операций) и способ их выполнения;
- способ хранения данных;
- способ записи команд исполнителя (язык программирования);
- способ взаимодействия с исполнителем (система ввода-вывода данных).

Существует несколько классических моделей вычислений, которые используются для исследования функций и вычисляемых функций:

нормальные алгоритмы Маркова, лямбда-исчисление Чёрча, машины Тьюринга, машины Поста и др. Данные модели являются *универсальными исполнителями*.

Неформально, универсальный исполнитель – это исполнитель, который может получить выходные данные для любой вычислимой функции при известных входных данных.

Тезис Чёрча-Тьюринга: *«любая вычислимая функция может быть вычислена с помощью соответствующей машины Тьюринга или любой другой эквивалентной ей модели»*. Если доказано, что некоторая модель является эквивалентной модели машин Тьюринга, то такую модель называют Тьюринг-полной. Все модели, эквивалентные модели машин Тьюринга, образуют класс эквивалентности, представители которого и получили название *«универсальных исполнителей»*.

С помощью создания таких вычислительных моделей как универсальные исполнители, математики стремились к математической формализации понятия *алгоритм* с целью последующего решения ряда математических задач из разных областей математического знания.

До проведенных работ Тьюрингом, Чёрчем, Постом и Марковым, которые ввели по факту математическое описание алгоритма, бытовало его неформализованное понятие, сущностью которого была последовательность действий для достижения конкретной цели.

ОБСУЖДАЕМ

Обсудим, что такое алгоритм, вычисление, вычислимая функция, модель вычисления и универсальный исполнитель. Чем отличаются друг от друга формальное и неформальное определение понятия алгоритм?

ЧИТАЕМ

Нормальные алгоритмы Маркова. Подстановкой Маркова мы будем называть пару последовательностей символов (слов) P и Q из заранее заданного алфавита Σ и будем обозначать $P \rightarrow Q$.

$$P, Q \in \Sigma^*$$

– это все слова, составленные только из символов алфавита Σ .

Применить подстановку $P \rightarrow Q$ к слову W – означает заменить первое вхождение последовательности символов P в слове W на последовательность символов Q .

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1.

$$\Sigma = \{a_0, a_1\}$$

Пусть существует алфавит

и подстановка $a_0a_1 \rightarrow a_1$

При применении данной подстановки, к слову, $a_1a_0a_1$, мы получим слово a_1a_1 . Если же применить данную подстановку, к слову $a_1a_0a_1a_0a_1$ то мы получим слово $a_1a_1a_0a_1$, так как при единичной подстановке нас интересует лишь первое вхождение. Подстановки можно применять несколько раз подряд.

ЧИТАЕМ

Подстановка называется конечной в том случае, если после её применения слово изменять уже нельзя – нельзя использовать данную подстановку или любую другую. Такая подстановка обозначается как $P \rightarrow .Q$

Нормальным алгоритмом Маркова называют последовательность однократных применений системы подстановок (схемы) к заданному слову, завершаемую в одном из двух случаев:

- система подстановок оказывается неприменимой к слову;
- была применена конечная подстановка.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 2.

Пусть существует система подстановок

$$\begin{cases} a_0 \rightarrow a_1 & (1) \\ a_1 \rightarrow .a_0 & (2) \end{cases}$$

составленная из алфавита $\Sigma = \{a_0, a_1\}$. Тогда при применении данного нормального алгоритма Маркова к слову a_0a_0 произойдут следующие изменения:

$$a_0a_0 \xrightarrow{(1)} a_1a_0 \xrightarrow{(1)} a_1a_1 \xrightarrow{(2)} a_0a_1$$

ЧИТАЕМ

Обратите внимание, что при применении схемы подстановки используются согласно порядку их перечисления внутри схемы.

Давайте введём понятие пустой последовательности символов и обозначим её символом ε . Будем считать, что пустая последовательность символов не должна указываться в словаре при построении нормального алгорифма Маркова.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 3.

Пусть существует схема подстановок:

$$\left\{ \begin{array}{ll} 0a_0 \rightarrow .1 & (1) \\ 1a_0 \rightarrow a_00 & (2) \\ a_0 \rightarrow .1 & (3) \\ a_10 \rightarrow 0a_1 & (4) \\ a_11 \rightarrow 1a_1 & (5) \\ 0a_1 \rightarrow 0a_0 & (6) \\ 1a_1 \rightarrow 1a_0 & (7) \\ \varepsilon \rightarrow a_1 & (8) \end{array} \right.$$

составленная из алфавита $\Sigma = \{0, 1, a_0, a_1\}$ Тогда при применении данного нормального алгорифма Маркова к словам $\varepsilon, 0, 1, 11$ произойдут следующие изменения:

$$(1) \quad \varepsilon \xrightarrow{(8)} a_1 \xrightarrow{(8)} a_1a_1 \xrightarrow{(8)} a_1a_1a_1 \xrightarrow{(8)} \dots$$

$$\xrightarrow{(8)} \underbrace{a_1 \dots a_1}_n \xrightarrow{(8)} \dots$$

$$(2) \quad 0 \xrightarrow{(8)} a_10 \xrightarrow{(4)} 0a_1 \xrightarrow{(6)} 0a_0 \xrightarrow{(1)} 1$$

$$(3) \quad 1 \xrightarrow{(8)} a_11 \xrightarrow{(5)} 1a_1 \xrightarrow{(7)} 1a_0 \xrightarrow{(2)} a_00 \xrightarrow{(3)} 10$$

$$(4) \quad 11 \xrightarrow{(8)} a_111 \xrightarrow{(5)} 1a_11 \xrightarrow{(5)} 1a_11 \xrightarrow{(7)}$$

$$\xrightarrow{(7)} 11a_0 \xrightarrow{(2)} 1a_00 \xrightarrow{(2)} a_000 \xrightarrow{(3)} 100$$

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Как было сказано раньше, с помощью нормальных алгорифмов Маркова можно решить любую вычислимую функцию. В качестве примера дальше мы

приведём схему подстановок, которая способна найти сумму любого количества аргументов, представленных в унарной системе счисления.

Пример 4. Сумма чисел

Пусть у нас есть алфавит $\Sigma = \{1, +\}$ тогда схема подстановок

$$\begin{cases} 1+ \rightarrow +1 & (1) \\ +1 \rightarrow 1 & (2) \\ 1 \rightarrow .1 & (3) \end{cases}$$

задаёт операцию сложения n аргументов.

Представим, что у нас есть сумма $2 + 3 + 4$, в унарной системе счисления она будет выглядеть как $11 + 111 + 1111$, возьмём эту последовательность символов как входную последовательность для применения схемы, описанной выше. Рассмотрим изменение данной последовательности символов:

$$\begin{aligned} 11+111+1111 &\Rightarrow 1\boxed{1+}111+1111 \xRightarrow{(1)} \\ \xRightarrow{(1)} 1\boxed{+1}111+1111 &\Rightarrow \boxed{1+}1111+1111 \xRightarrow{(1)} \\ \xRightarrow{(1)} \boxed{+1}1111+1111 &\Rightarrow +1111\boxed{1+}1111 \xRightarrow{(1)} \\ \xRightarrow{(1)} +1111\boxed{+1}1111 &\Rightarrow +111\boxed{1+}11111 \xRightarrow{(1)} \dots \xRightarrow{(1)} \\ \xRightarrow{(1)} ++11111111 &\Rightarrow +\boxed{+1}11111111 \xRightarrow{(2)} \\ \xRightarrow{(2)} +\boxed{1}11111111 &\Rightarrow \boxed{+1}11111111 \xRightarrow{(2)} \\ \xRightarrow{(2)} \boxed{1}11111111 &\xRightarrow{(3)} 11111111 \end{aligned}$$

Описанная выше схема подстановок будет корректно работать для любой последовательности символов, составленных из символов алфавита $\Sigma = \{1, +\}$.

ЧИТАЕМ

Машины Поста. Машина Поста – это ещё одна модель вычислений и универсальный исполнитель. При её определении говорят, что она состоит из бесконечной ленты ячеек, которые могут быть либо помечены (1), либо нет (0). Вдоль данной ленты движется каретка, указывающая на конкретную

ячейку – такую ячейку называют текущей. Ниже представлен пример состояния ленты:

| | | | | | | | | | | | |
|---|---|----------|---|---|---|---|---|---|---|---|---|
| 0 | 1 | <u>0</u> | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
|---|---|----------|---|---|---|---|---|---|---|---|---|

Для записи программы для машины Поста используется набор команд:

- $V\ i$ – пометить текущую ячейку и перейти к команде i ;
- $X\ i$ – убрать пометку на текущей ячейке и перейти к команде i ;
- $L\ i$ – сдвинуться на одну ячейку влево и перейти к команде i ;
- $R\ i$ – сдвинуться на одну ячейку вправо и перейти к команде i ;
- $?\ i\ j$ – если текущая ячейка помечена, то перейти к команде i , иначе – перейти к команде j ;
- $!$ – остановить выполнение программы.

Так же считается, что если программа написана таким образом, что попытается пометить помеченную ячейку (или убрать пометку с непомеченной ячейки), то происходит ошибка и аварийная остановка выполнения программы. Программа считается корректно выполненной для конкретного набора данных в том случае, если не было получено ошибок при выполнении программы и была выполнена команда '!'.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 5

Пусть у нас есть программа для машины Поста:

- | | | | |
|----|----------|-----|-----------|
| 1. | $R\ 2$ | 7. | $X\ 8$ |
| 2. | $? 1\ 3$ | 8. | $L\ 9$ |
| 3. | $V\ 4$ | 9. | $? 8\ 10$ |
| 4. | $R\ 5$ | 10. | $!$ |
| 5. | $? 4\ 6$ | | |
| 6. | $L\ 7$ | | |

и начальное состояние ленты:

| | | | | | | | | |
|----------|---|---|---|---|---|---|---|---|
| <u>0</u> | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
|----------|---|---|---|---|---|---|---|---|

Тогда данная программа преобразует ленту к следующему состоянию:

Доходим до ближайшего нуля
и превращаем его в единицу

$$\boxed{\underline{0}} \boxed{1} \boxed{1} \boxed{0} \boxed{1} \boxed{1} \boxed{1} \boxed{0} \boxed{0} \xRightarrow{(1 \rightarrow 2)}$$

$$\xRightarrow{(1 \rightarrow 2)} \boxed{0} \boxed{\underline{1}} \boxed{1} \boxed{0} \boxed{1} \boxed{1} \boxed{1} \boxed{0} \boxed{0} \xRightarrow{(2 \rightarrow 1 \rightarrow 2)}$$

$$\xRightarrow{(2 \rightarrow 1 \rightarrow 2)} \boxed{0} \boxed{1} \boxed{1} \boxed{\underline{0}} \boxed{1} \boxed{1} \boxed{1} \boxed{0} \boxed{0} \xRightarrow{(3 \rightarrow 4)}$$

$$\xRightarrow{(3 \rightarrow 4)} \boxed{0} \boxed{1} \boxed{1} \boxed{\underline{1}} \boxed{1} \boxed{1} \boxed{1} \boxed{0} \boxed{0} \xRightarrow{(4 \rightarrow 5)}$$

Доходим до последней единицы
и превращаем ее в ноль

$$\xRightarrow{(4 \rightarrow 5)} \boxed{0} \boxed{1} \boxed{1} \boxed{1} \boxed{\underline{1}} \boxed{1} \boxed{1} \boxed{0} \boxed{0} \xRightarrow{(5 \rightarrow 4 \rightarrow 5)} \dots$$

$$\dots \xRightarrow{(5 \rightarrow 6)} \boxed{0} \boxed{1} \boxed{1} \boxed{1} \boxed{1} \boxed{1} \boxed{1} \boxed{\underline{0}} \boxed{0} \xRightarrow{(6 \rightarrow 7)}$$

$$\xRightarrow{(6 \rightarrow 7)} \boxed{0} \boxed{1} \boxed{1} \boxed{1} \boxed{1} \boxed{1} \boxed{\underline{1}} \boxed{0} \boxed{0} \xRightarrow{(7 \rightarrow 8)}$$

$$\xRightarrow{(7 \rightarrow 8)} \boxed{0} \boxed{1} \boxed{1} \boxed{1} \boxed{1} \boxed{1} \boxed{\underline{0}} \boxed{0} \boxed{0} \xRightarrow{(9 \rightarrow 8 \rightarrow 9)}$$

Возвращаемся в начальное положение

$$\xRightarrow{(9 \rightarrow 8 \rightarrow 9)} \dots \xRightarrow{(9 \rightarrow 10)} \boxed{\underline{0}} \boxed{1} \boxed{1} \boxed{1} \boxed{1} \boxed{1} \boxed{1} \boxed{0} \boxed{0}$$

Ясно видно, что программа, представленная выше, реализует функцию сложения двух целых чисел в унарной системе счисления.

Машины Поста очень сильно напоминают работу компьютера. За счёт простоты написания программ и своего устройства данная вычислительная модель очень легко ассоциируется с наивным понятием алгоритма.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1.

Создайте машину Поста (напишите программу для машины Поста), которая может находить сумму n аргументов.

Задание 2.

Предложите варианты нормального алгоритма Маркова для нахождения разности двух чисел.

Задание 3.

Напишите программы на языке программирования Python для моделирования моделей вычисления:

- а. нормальные алгоритмы Маркова;
- б. машины Поста.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Что такое универсальный исполнитель, вычислимая функция? Какова связь данных понятий?
2. Опишите основные особенности универсального исполнителя – алгоритмов Маркова.
3. Опишите основные особенности универсального исполнителя – машин Поста.

§7.3 Программирование машины Тьюринга

ВСПОМИНАЕМ

Давайте вспомним:

1. Что такое универсальный исполнитель, алгоритм, вычислимая функция?
2. Что такое машина Поста?
3. Что такое нормальные алгоритмы Маркова?

ЧИТАЕМ

Машина Тьюринга. Как мы уже говорили ранее, машина Тьюринга – это универсальный исполнитель, который состоит из бесконечной ленты ячеек, в каждой из которых записан символ некоторого алфавита. Так же машина Тьюринга подразумевает наличие устройства управления, состоящее из каретки, которая может изменять значение текущей ячейки.

На прошлом уроке мы рассматривали более простую формализацию алгоритма – машину Поста – которая по своему устройству очень похожа на машину Тьюринга. Но при этом программы для машины Тьюринга записываются несколько иначе.

Под конкретной машиной Тьюринга мы будем понимать систему семи множеств, описывающую обработку входных данных:

- $TTM = \{Q, \Sigma, \Gamma, \delta, q_0, B, F\}$, где TTM (англ. *the Turing machine*) – машина Тьюринга;
- $Q = \{q_0, q_1, \dots, q_n\}$ – множество состояний устройства управления машины Тьюринга;
- $\Sigma = \{a_0, a_1, \dots, a_m\}$ – входной алфавит, элементы которого используются для записи входного слова;
- $\Gamma = \{a_0, a_1, \dots, a_m, B, b_1, b_2, \dots, b_k\}$ – ленточный (технический) алфавит, включающий в себя входной алфавит $\Sigma \subset \Gamma$, пробельный символ $B \in \Gamma$ и некоторые другие символы, не являющиеся обязательными ($\{b_1, b_2, \dots, b_k\} \subset \Gamma$), но, возможно, требующиеся для работы конкретной машины Тьюринга;
- $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, N, R\}$ – функция переходов, которая в зависимости от текущего состояния $q_i (q_i \in Q)$ и настоящего символа в текущей ячейке $\gamma_j (\gamma_j \in \Gamma)$ определяет новое состояние, в которое переходит система, $q_{i0} (q_{i0} \in Q)$, записывает в текущую ячейку символ $\gamma_{j0} (\gamma_{j0} \in \Gamma)$ и передвигает каретку согласно команде *command* ($command \in \{L, N, R\}$);
- $q_0 (q_0 \in Q)$ – начальное состояние машины Тьюринга;
- $B (B \in \Gamma)$ – пробельный символ;
- $F (F \subset Q)$ – множество конечных состояний, при достижении которых машина Тьюринга останавливает свою работу.

Таким образом, для создания программы для машины Тьюринга нам достаточно конкретизировать систему, упомянутую выше.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Давайте рассмотрим пример создания машины Тьюринга.

Пусть машина Тьюринга задана следующей системой:

$TTM = \{Q, \Sigma, \Gamma, \delta, q_0, \lambda, \{q_5\}\}$, где

- $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$;
- $\Sigma = \{1, +\}$;
- $\Gamma = \{1, +, 0, \lambda\}$;
- Функция переходов представлена ниже в виде таблицы:

| δ | λ | 1 | + | = | 0 |
|----------|---------------------|---------------|---------------|---------------|---------------|
| q_0 | $(q_1, =, L)$ | $(q_0, 1, R)$ | $(q_0, +, R)$ | - | - |
| q_1 | (q_2, λ, R) | $(q_1, 1, L)$ | $(q_1, +, L)$ | $(q_1, =, L)$ | $(q_2, 0, R)$ |
| q_2 | - | $(q_3, 0, R)$ | $(q_2, 0, R)$ | - | $(q_4, =, L)$ |
| q_3 | $(q_1, 1, L)$ | $(q_3, 1, R)$ | $(q_3, +, R)$ | $(q_3, =, R)$ | - |
| q_4 | (q_5, λ, N) | - | $(q_4, +, L)$ | - | $(q, 1, L)$ |
| q_5 | - | - | - | - | - |

- q_0 – начальное состояние;
- λ – пробельный символ;
- q_5 – конечное состояние.

Если в качестве входной последовательности для обозначенной машины Тьюринга поступит 111+11, произойдут следующие её изменения:

Проходим по всей последовательности
символов туда-и-обратно

$$\begin{aligned}
 \boxed{1}11+11 &\xrightarrow{(q_0,1) \rightarrow (q_0,1,R)} 1\boxed{1}1+11 \xrightarrow{(q_0,1) \rightarrow (q_0,1,R)} \dots \\
 111\boxed{+}11 &\xrightarrow{(q_0,+) \rightarrow (q_0,+,R)} 111+\boxed{1}1 \xrightarrow{(q_0,1) \rightarrow (q_0,1,R)} \dots \\
 111+11\boxed{\lambda} &\xrightarrow{(q_0,\lambda) \rightarrow (q_1,=,L)} 111+1\boxed{1}= \xrightarrow{(q_1,1) \rightarrow (q_1,1,L)} \dots \\
 \boxed{\lambda}111+11= &\xrightarrow{(q_1,\lambda) \rightarrow (q_2,\lambda,R)}
 \end{aligned}$$

Переносим по одной единице из левой
части равенств правую

$$\begin{aligned}
 \lambda\boxed{1}11+11 &= \xrightarrow{(q_2,1) \rightarrow (q_3,0,R)} \lambda 0\boxed{1}1+11 = \xrightarrow{(q_3,1) \rightarrow (q_3,1,R)} \dots \\
 \lambda 011+11\boxed{=} &\xrightarrow{(q_3,=) \rightarrow (q_3,=,R)} \lambda 011+11=\boxed{\lambda} \xrightarrow{(q_3,\lambda) \rightarrow (q_1,1,L)} \\
 \lambda 011+11\boxed{=}1 &\Rightarrow \dots \Rightarrow \lambda 000+00=111\boxed{1}1 \Rightarrow \dots
 \end{aligned}$$

Восстанавливаем первоначальную запись

$$\begin{aligned}
 \lambda 000+00\boxed{=}11111 &\xrightarrow{(q_2,=) \rightarrow (q_4,=,L)} \lambda 000+0\boxed{0}=11111 \xrightarrow{(q_4,0) \rightarrow (q_4,1,L)} \\
 \lambda 000+\boxed{0}1=11111 &\Rightarrow \dots \Rightarrow \boxed{\lambda}111+11=11111 \xrightarrow{(q_4,\lambda) \rightarrow (q_5)} \\
 111+11=11111 &
 \end{aligned}$$

То есть мы снова написали программу для нахождения суммы двух чисел, представленных в унарной системе счисления. Причем, обратите внимание, в техническом алфавите Γ присутствует символ 0, он в данном случае выполняет роль пометки, отвечающей за те единицы, которые мы учли в сумме, после чего, когда все единицы оказались учтены, все нули заменяются обратно на единицы. Использование нулей оказалось достаточным для реализации операции суммы двух чисел.

Подумайте, является ли использование нулей в данной программе необходимостью? Можно ли реализовать операцию суммы без использования дополнительных символов?

Давайте попробуем построить более простую машину Тьюринга, которая просто увеличивала бы число, записанное в унарной системе счисления, на единицу. Условимся, что каретка в начальном состоянии находится на первой цифре числа.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 2

1. Давайте подумаем над тем, как должен выглядеть наивный алгоритм решения данной задачи. У нас есть последовательность единиц и наш фокус внимания находится на самой первой единице. Очевидно, что для решения данной задачи достаточно пройти до самой левой единицы, сделать ещё один шаг влево на пустую ячейку и записать туда единицу, после чего необходимо вернуться обратно – к самой первой единице.

2. При разбиении данной задачи на подзадачи, мы сперва можем утверждать, что у нас есть два состояния – движение вправо и движение влево. Двух состояний нам должно хватить, так как мы не будем выполнять никаких циклов, и ничего, кроме направления, нам запоминать не нужно, мы сменим направление только тогда, когда поставим ту самую единицу. Отталкиваясь от тех же рассуждений, мы можем прийти к выводу, что и во входном словаре нам нужен только один символ – единица.

3. Тогда определим основные множества:

$Q = \{q_0, q_1, q_2\}$, при этом q_2 будем считать конечным состоянием, а q_0 – начальным;

$$\Sigma = \{1\};$$

$$\Gamma = \{1, \lambda\}.$$

4. Теперь нам необходимо определить функцию перехода. Мы условились, что пока мы движемся вперёд, мы считаем, что находимся в одном состоянии, при этом смена состояния должна произойти тогда, когда мы дойдём до конца числа – до пустой ячейки. Это можно показать с помощью

двух функций перехода – $(q_0, 1) \rightarrow (q_0, 1, R)$ и, если мы встретим пустую ячейку, мы запишем в неё единицу, сменим состояние и начнём двигаться в другую сторону $(q_0, \lambda) \rightarrow (q_1, 1, L)$. Теперь мы будем двигаться до конца слова – $(q_1, 1) \rightarrow (q_1, 1, L)$, и остановимся на пустой ячейке – $(q_1, \lambda) \rightarrow (q_2, \lambda, R)$, после чего вернёмся на первую единицу. То есть, только что мы определили функцию перехода:

$$\delta = \{(q_0, 1) \rightarrow (q_0, 1, R)(q_0, \lambda) \rightarrow (q_1, 1, L)(q_1, 1) \rightarrow (q_1, 1, L)(q_1, \lambda) \rightarrow (q_2, \lambda, R),$$

или с помощью таблицы:

| δ | λ | 1 |
|----------|---------------------|---------------|
| q_0 | $(q_1, 1, L)$ | $(q_0, 1, R)$ |
| q_1 | (q_2, λ, R) | $(q_1, 1, L)$ |
| q_2 | - | - |

5. Таким образом, образованная нами машина Тьюринга $TTM = \{Q, \Sigma, \Gamma, \delta, q_0, \lambda, \{q_2\}\}$ полностью решает описанную выше задачу.

РЕШАЕМ В ТЕТРАДИ

Задание 64.

1. Приведите в качестве иллюстрации два примера того, что данная машина Тьюринга действительно решает поставленную перед ней задачу.
2. Пусть входное слово – это число, представленное в двоичной системе счисления. Создайте программу для машины Тьюринга, которая умножала бы это число на 2 (умножала бы это число на 4). Считайте, что первоначально каретка находится на первом символе входной последовательности.
3. Пусть входное слово – это число, представленное в двоичной системе счисления, создайте программу для машины Тьюринга, которая целочисленно делила бы это число на 2. Считайте, что первоначально каретка находится на первом символе входной последовательности.
4. Пусть в роли входного слова выступает четырёхбуквенное слово, составленное из букв алфавита $\Sigma = \{a, b\}$. Напишите программу для машины Тьюринга, которая в качестве выхода представляла перевёрнутое входное слово. Считайте, что первоначально каретка находится на первом символе входной последовательности.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1.

Напишите программу, которая имитировала бы работу машины Тьюринга.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Дайте определение машине Тьюринга, объясните устройство данной модели вычисления.
2. Объясните разницу между машинами Тьюринга и машинами Поста (нормальными алгорифмами Маркова).

§7.4 Понятие асимптотической сложности алгоритма

ВСПОМИНАЕМ

Давайте вспомним:

1. Что такое алгоритм?
2. Какие примеры формализации понятия алгоритм вы знаете?

ЧИТАЕМ

Асимптотическая сложность алгоритма. Такой раздел математики, как теория алгоритмов, занимается изучением свойств и закономерностей алгоритмов на основе их представления в виде формальных моделей. Одной из важных задач теории алгоритмов является асимптотический анализ сложности алгоритма.

Под асимптотическим анализом (функции) в математике понимают изучение поведения функции с помощью метода предельного перехода.

Сложность (или вычислительная сложность) алгоритма – это функция зависимости затрачиваемых ресурсов (память и время), используемых при работе некоторого алгоритма, от размера входных данных.

Таким образом, под асимптотическим анализом вычислительной сложности алгоритма мы будем понимать использование асимптотического анализа для оценки поведения алгоритма с точки зрения занимаемой им для обработки данных памяти $M(n)$ и затрачиваемого времени $T(n)$.

Несмотря на то, что можно определить точную функцию сложности алгоритма, подобный подход используется редко, так как существует вопрос конкретизации операций, которые будут считаться элементарными. А разрешение данного вопроса не всегда оптимально с точки зрения времени анализа конкретного алгоритма. Поэтому при оценке сложности конкретного алгоритма часто прибегают именно к асимптотическому анализу.

ОБСУЖДАЕМ

Что такое асимптотический анализ алгоритма? Какие задачи перед собой ставит человек, использующий метод асимптотического анализа в практической деятельности?

Как вы думаете, есть ли разница в подходах к использованию данного метода в сферах спортивного (олимпиадного) программирования и промышленного?

ЧИТАЕМ

Обозначения асимптотической сложности

Говорят, что функция $f_{(n)}$ ограничена сверху и снизу функцией $g_{(n)}$ ($f_{(n)} \in \Theta(g_{(n)})$), если $\exists (c_1 > 0, c_2 > 0, n_0) : \forall (n > n_0) c_1 * g_{(n)} \leq f_{(n)} \leq c_2 * g_{(n)}$,

то есть, если мы можем для конкретной функции $f_{(n)}$ найти такую функцию $g_{(n)}$, для которой существуют два коэффициента, при умножении на которые значение функции $g_{(n)}$ при увеличении аргумента будет всегда меньше при одном коэффициенте и больше при втором, начиная с некоторого фиксированного n_0 (рис. 167).

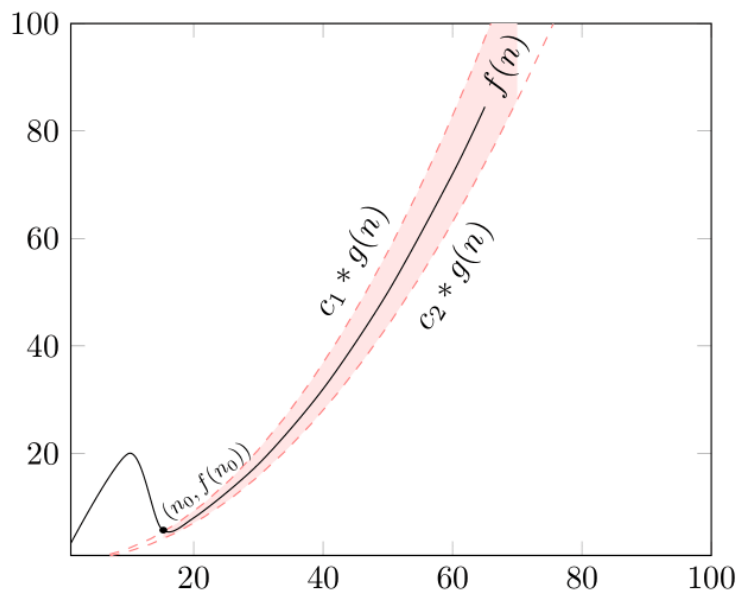


Рис. 167 Функция $F(n)$

Говорят, что функция $f_{(n)}$ ограничена снизу функцией $g_{(n)}$ ($f_{(n)} \in \Omega(g_{(n)})$), если $\exists (c > 0, n_0) : \forall (n > n_0) c * g_{(n)} \leq f_{(n)}$, то есть, если мы можем для конкретной функции $f_{(n)}$ найти такую функцию $g_{(n)}$, для которой существует коэффициент, при умножении на который значение функции $g_{(n)}$ при увеличении аргумента будет всегда меньше, начиная с некоторого фиксированного n_0 (рис. 168).

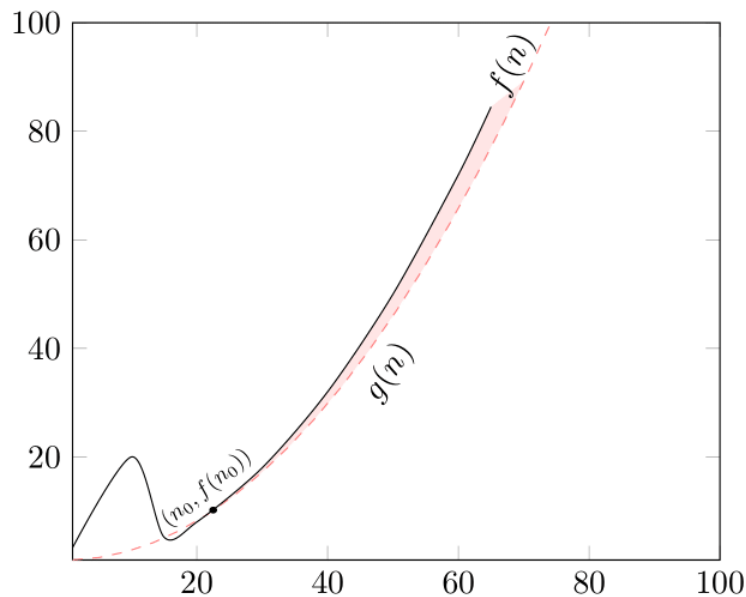


Рис. 168 Функция $F(n)$

Говорят, что функция $f(n)$ ограничена сверху функцией $g(n)$ ($f(n) \in O(g(n))$), если $\exists(c > 0, n_0) : \forall(n > n_0) c \cdot g(n) \geq f(n)$, то есть, если мы можем для конкретной функции $f(n)$ найти такую функцию $g(n)$, для которой существуют коэффициент, при умножении на который значение функции $g(n)$ при увеличении аргумента будет всегда больше, начиная с некоторого фиксированного n_0 (рис. 169).

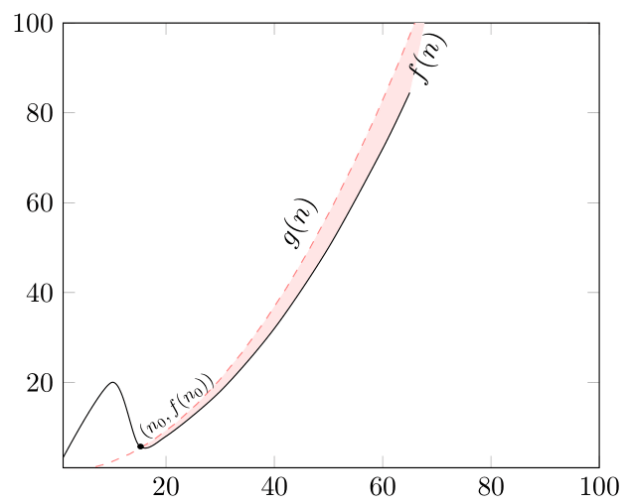


Рис. 169 Функция $F(n)$

Таким образом, при применении к понятию алгоритма асимптотической сложности, на данных графиках по оси абсцисс расположены значения, соответствующие количеству данных, подаваемых для обработки алгоритмом, а ось ординат можно интерпретировать как количество времени $T(n)$ или памяти $M(n)$, необходимое для обработки входных данных.

Основная задача использования асимптотического анализа в программировании – это оценка роста времени (или памяти) при выполнении алгоритма компьютером в зависимости от количества входных данных.

Согласно определениям асимптотической сложности O, Ω, Θ , что мы давали выше, асимптоты рассматриваются начиная с некоторого n_0 . Количество данных, меньшее n_0 , мы будем называть данными малого размера.

Мы будем считать, что алгоритмы с меньшей асимптотической сложностью являются более эффективными. При этом важно учитывать, что существуют ситуации, в которых алгоритм с большей асимптотической сложностью может быть эффективнее на данных малого объёма. Абстрактным примером подобного алгоритма может быть алгоритм, соотносящийся с функцией $f(n)$, график которой представлен на рисунках выше.

При оценке асимптотической сложности алгоритма, как правило происходит оценка сверху O , то есть оценка наихудшего времени выполнения (или занимаемой памяти) программой – подобный подход вполне оправдан, так как позволяет точно оценить окончание времени работы (или занимаемую память), то есть – верхнюю границу.

ОБСУЖДАЕМ

Сравните три основных способа оценки асимптотической сложности алгоритма? Какой из них наиболее часто применяется на практике? Почему?

ЧИТАЕМ

Распространённые сложности алгоритма

Давайте рассмотрим наиболее частые асимптотические сложности алгоритмов (рис. 170). Обратите внимание, что на представленном рисунке ось ординат имеет логарифмический масштаб – длина отрезка на такой шкале пропорциональна логарифму отношения величин на концах отрезка – это сделано с целью более иллюстративно продемонстрировать разницу между основными асимптотами.

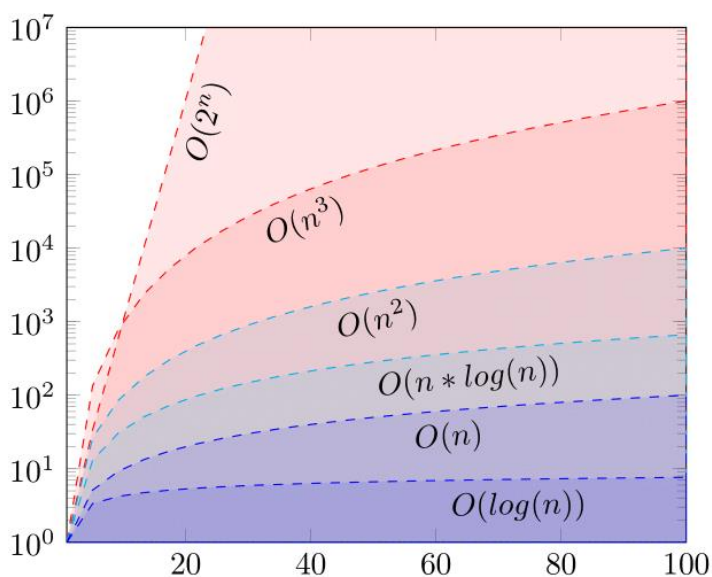


Рис. 170 Асимптотические сложности алгоритмов

Экспоненциальная сложность. Алгоритмы, которые оцениваются экспоненциальной асимптотой – это алгоритмы, которые не используются при решении задач, связанных с обработкой большого количества данных, так как экспоненциальный рост по своей природе очень быстрый. На рисунке мы указали $O(2^n)$ – это наиболее часто встречаемая сложность, но, очевидно, не единственная – в качестве основания могут быть и другие числа в зависимости от рассматриваемого алгоритма. В нашем конкретном примере данная асимптотическая сложность описывает алгоритмы, в которых увеличение входных данных на одну единицу увеличивает время выполнения (или затрачиваемую память) программы, реализующей данный алгоритм, в 2 раза. Наиболее ярким примером алгоритма с данной сложностью является рекурсивное решение задачи о Ханойской башне.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

По условию данной задачи имеется три стержня, на одном из которых расположено n колец с попарно отличными диаметрами, расположенными сверху вниз от меньшего к большему. Необходимо перенести всю башню с одного стержня на другой, при этом разрешается перемещать только по одному кольцу и класть меньшее исключительно на большее кольцо. Ниже представлена реализация решения данной задачи на языке программирования Python.

```
def hanoi(n, from_kernel=1, to_kernel=2, with_kernel=3):  
    '''  
    Рекурсивная функция hanoi решает классическую задачу  
    Ханойская башня. В качестве аргументов принимает может  
    Принимать четыре аргумента
```

АРГУМЕНТЫ

```
n : int  
    количество дисков,  
    которые должна перенести функция;  
from_kernel : int  
    стержень, с которого будут перенесены  
    n-дисков; имеет значение по умолчанию  
    для первого вызова;  
to_kernel : int  
    стержень, на который будут перенесены  
    n-дисков; имеет значение по умолчанию  
    для первого вызова;  
with_kernel : int  
    стержень, который будет использоваться
```



```

        как буфер; имеет значение по умолчанию
        для первого вызова.
'''

# перемещаем верхние n-1 диск на вспомогательный
# стержень для текущего шага
hanoi(n-1, from_kernel, with_kernel, to_kernel)

# перемещаем большой диск на стержень to_kernel
# -- описание вспомогательной функции опущено
help_hanoi(from_kernel, to_kernel)

# перемещаем верхние диски с буферного стержня
# на целевой стержень
hanoi(n-1, with_kernel, to_kernel, from_kernel)

```

То есть, подобное решение базируется на следующей идее:

1. Перенести башню из n-1 диска на буферный стержень;
2. Перенести большой диск на целевой стержень;
3. Перенести башню из n-1 диска с буферного стержня на целевой.

Таким образом, данная задача рекурсивно сводится до перенесения одного диска.

Давайте попробуем доказать, что алгоритмическая сложность данного алгоритма действительно $O(2^n)$.

Очевидно, что для решения задачи из одного диска – $T(1) = 1$, для двух – $T(2) = 3$. Для того, чтобы перенести башню из трёх дисков, нам необходимо перенести башню из двух дисков, потом переложить третий, потом перенести башню из двух дисков, то есть $T(3) = T(2) + T(1) + T(2) = 2 * T(2) + T(1) = 7$. Отсюда мы можем предположить, что для $T(n) = 2 * T(n-1) + T(1)$. Для того, чтобы переложить n+1 кольцо, нам необходимо $T(n+1) = 2 * T(n) + T(1)$ действий, что можно описать следующим образом $T(n+1) = 2 * T(n) + T(1) = 2 * (2 * T(n-1) + T(1)) + T(1)$, то есть количество действий в сравнении с задачей с n дисками увеличивается в два раза и прибавляется ещё одно действие.

Утверждаем, что $T(n) = 2^n - 1$. Получить данную функцию можно с помощью метода решения рекуррентных соотношений, а доказать её истинность с помощью метода математической индукции, что мы настоятельно рекомендуем вам самостоятельно проделать.

Согласно определению асимптотической сложности $O(n)$, очевидно, что $T(n) \in O(2^n)$, так как существуют такие $c = 1$, $n_0 = 0$, что $\forall (n > n_0) T(n) \leq 2^n$, так как $\forall (n > 0) (2^n - 1) \leq 2^n$.

ЧИТАЕМ

Квадратическая сложность. Асимптотическую сложность $O(n^2)$ называют квадратической. Примером алгоритма, соответствующего данной сложности являются некоторые алгоритмы сортировки, например – алгоритм сортировки пузырьком.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 2

Реализация данного алгоритма на языке программирования Python представлена ниже.

```
# n – количество элементов списка, которые необходимо
упорядочить
n: int
# считаем, что список arr уже заполнен n элементами
arr: list

# СОРТИРОВКА ПУЗЫРЬКОМ
# самый 'тяжёлый' элемент отправляем в конец списка
for i in range(n-1):
    # упорядочиваем 'тяжёлый' элемент в неупорядоченной
    # части списка
    for j in range(n-1-i):
        # меняем местами два соседних элемента,
        # если они расположены неупорядочено
        if arr[j] > arr[j+1]:
            arr[j], arr[j+1] = arr[j+1], arr[j]

# в этот момент элементы в списке arr упорядочены по неубыванию
```

При анализе данного алгоритма легко посчитать количество шагов цикла (внутреннего цикла). Мы $n-1$ раз уменьшаем количество итераций внутреннего цикла, при первом проходе по которому происходит $n-1$ итераций, то есть

$$T(n) = (n-1) + (n-2) + \dots + 2 + 1 = \frac{(n-1) * n}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$$

Функция $T(n)$ является квадратической, то есть мы можем подобрать такой коэффициент, что $T(n) \leq n^2$.

ЧИТАЕМ

Линеарифметическая сложность. Асимптотическую сложность $O(n \cdot \log(n))$ называют линеарифметической. Данная алгоритмическая сложность подразумевает более медленный рост, нежели квадратическая. Примерами алгоритмов с подобной сложностью являются некоторые виды сортировок и алгоритмы использующие аналогичные идеи. С ними мы обязательно будем разбираться в следующих уроках.

Линейная сложность. Асимптотическую сложность $O(n)$ называют линейной. Примером алгоритма с подобной сложностью является поиск элемента в неотсортированном списке, так как при решении подобной задачи нам необходимо проверить все n элементов.

Логарифмическая сложность. Асимптотическую сложность $O(\log(n))$ называют логарифмической. Наиболее ярким и важным примером алгоритма с подобной сложностью является алгоритм бинарного поиска элемента в отсортированном списке. Данный алгоритм обязательно будет рассмотрен в следующих уроках.

Константная сложность. Асимптотическую сложность $O(1)$ называют константной. Данной сложностью обладают алгоритмы, время выполнения которых не зависит от количества входных данных – обращение к элементу массива, нахождение суммы двух чисел и др.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1.

Реализуйте алгоритм сортировки пузырьком для упорядочивания строк в лексикографическом порядке (и в порядке увеличения длины строк).

Задание 2.

Напишите программу, которая бы подсчитала количество одиночных перестановок колец при решении задачи о ханойской башне для любого целого числа колец.

Задание 3.

Напишите программу поиска элемента в списке. Программа должна работать следующим образом: вводится целое число n – количество элементов в списке, после чего вводятся n элементов в одной строке. Программа должна в качестве результата своей работы предоставить отчёт о наличии элемента в списке («NO» или «YES»). Обоснуйте сложность данного алгоритма (в нотации O -большое).

Задание 4.

Напишите программу поиска максимума в двумерном массиве (в матрице). Программа должна работать следующим образом: вводится одно целое число n – размер матрицы по высоте и ширине (матрица квадратная), далее в n строках вводится по n целых чисел. Программа должна в качестве результата своей работы предоставить значение наибольшего числа. Обоснуйте сложность данного алгоритма (в нотации O -большое).

Подумайте, если бы эта задача была о трехмерной матрице, изменилась бы временная сложность?

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Что такое асимптотический анализ алгоритма?
2. Какие функции (способы) для оценки асимптотического поведения алгоритма вы знаете?
3. Какие сложности алгоритмов вы знаете (в нотации O -большое)?

§7.5 Сравнительный анализ сложности методов сортировки

ВСПОМИНАЕМ

Давайте вспомним:

1. Что такое асимптотическая сложность алгоритма?
2. Чем отличаются друг от друга O -большое, Ω -большая и Θ -большая? Какая из данных функций наиболее часто используется при оценке асимптотической сложности алгоритма?
3. Какие сложности чаще всего встречаются в практической деятельности?

ЧИТАЕМ

Алгоритмы сортировки. Ранее мы затрагивали понятие алгоритма сортировки. Неформально алгоритм сортировки – это алгоритм, используемый для упорядочивания элементов в структуре данных; чаще всего в качестве структуры данных используется массив; в случае с языком программирования Python – список.

Задача сортировки. Пусть у нас есть несколько объектов:

O_1, O_2, \dots, O_n ,

которые необходимо упорядочить согласно некоторой характеристике K_k (для объекта O_k), это значит, что сравниваются именно значения K этих объектов при определении их порядка в упорядоченной последовательности. Тогда, если для данной последовательности справедливы два свойства:

1) закон трихотомии: $\forall K_i, K_j$ может быть справедливо только одно из трёх соотношений – $a < b$, $b < a$, $a=b$;

2) закон транзитивности: $\forall K_i, K_j, K_k$ если $K_i < K_j$ и $K_j < K_k$, то $K_i < K_k$, то мы будем считать, что такую последовательность можно упорядочить, то есть найти такую перестановку объектов

$p(1), p(2), \dots, p(n)$, при которой

$K_{p(1)} \leq K_{p(2)} \leq \dots \leq K_{p(n)}$ в случае упорядочивания по невозрастанию.

Таким образом утверждается, что для упорядочивания (задача сортировки) необходимо и достаточно разместить объекты в такой последовательности, чтобы для любых двух из них при фиксации их позиции первый и второй объект находились в введённом отношении, например, для отношения “меньше или равно” (невозрастание) необходимо, чтобы каждый объект, размещённый в последовательность был меньше или равен, согласно рассматриваемой характеристике, чем любой объект, размещённый в последовательности после.

Также стоит заметить, что методы определения позиции объекта при сортировке делят на устойчивые и неустойчивые.

Устойчивой сортировкой называют алгоритм, при котором объекты с равным значением характеристики, по которой происходит упорядочивание, не изменяют своего положения относительно друг друга при перестановке, то есть

$$p(1) < p(2) \forall K_{p(i)} = K_{p(j)} \Rightarrow i < j$$

ОБСУЖДАЕМ

Обсудим:

1. Какую последовательность элементов можно упорядочить? Какими свойствами она должна обладать?
2. Если алгоритм сортировки называют устойчивым, что это значит?

ЧИТАЕМ

Классические методы сортировки

Наиболее часто при рассмотрении и знакомстве с понятием сортировки прибегают к *простым сортировкам*. Простыми их называют, так как они просты в понимании и освоении – в них заложены интуитивно понятные идеи. Одну из таких сортировок мы рассмотрели в предыдущем параграфе (*сортировка пузырьком*), когда говорили об асимптотической сложности $O(n^2)$. Почти все сортировки, именуемые простыми, осуществляются именно

с квадратичной сложностью, что делает их неэффективными при обработке большого количества данных, особенно в сравнении с линейно-арифметическими сортировками $O(n \cdot \log(n))$. Стоит заметить, что при оценке асимптотической сложности алгоритмов сортировки используют термины *худшее время алгоритма*, *среднее время алгоритма*, *лучшее время алгоритма*. Данные характеристики отвечают на вопрос поведения алгоритма в зависимости от входных данных.

Квадратичные сортировки

Как уже было сказано выше – *сортировка пузырьком* является квадратичной, её сложность была доказана в предыдущем параграфе. Ниже мы рассмотрим ещё несколько аналогичных сортировок и докажем, что они являются квадратичными.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Сортировку пузырьком можно модифицировать с помощью добавления проверки, проходили ли перестановки при очередном проходе по подпоследовательности объектов:

```
# n – количество эл-ов списка, которые необходимо упорядочить
n: int
# считаем, что список arr уже заполнен n элементами
arr: list
```

```
# СОРТИРОВКА ПУЗЫРЬКОМ
```

```
# самый 'тяжёлый' элемент отправляем в конец списка
for i in range(n-1):
```

```
    flag = False # переменная, отвечающая на вопрос,
                  # были ли перестановки на очередном
                  # проходе цикла
```

```
    # упорядочиваем 'тяжёлый' элемент в неупорядоченной
    # части списка
```

```
    for j in range(n-1-i):
```

```
        # меняем местами два соседних элемента,
        # если они расположены неупорядочено
```

```
        if arr[j] > arr[j+1]:
```

```
            arr[j], arr[j+1] = arr[j+1], arr[j]
```

```
            flag = True # отмечаем, что произошла хотя бы
                        # одна перестановка
```

```
    # если перестановок на очередном проходе внутреннего
```

```
# цикла не произошло, то список объектов упорядочен
if not flag:
    break # завершаем работу алгоритма
```

в этот момент элементы в списке arr упорядочены по неубыванию

При подобной модификации может случиться так, что алгоритм будет обрабатывать уже упорядоченный список объектов. При этом алгоритм выполнит только одну итерацию внешнего цикла – такая ситуация называется лучшим временем выполнения алгоритма и её корректная оценка – $O(n)$, при этом худшее время так и остаётся квадратическим.

ЧИТАЕМ

Сортировка выбором – простая сортировка, основная идея которой заключается в выборе объекта с минимальным значением характеристики из неотсортированной части списка (такой объект далее будем называть минимальным), после чего выбранный объект ставится в начало данной неотсортированной части. Чуть более формально данный алгоритм можно описать так:

- 1) считаем весь список объектов неотсортированным;
- 2) сортировка неотсортированной части списка:
 - a. находим минимальный объект в неотсортированной части списка;
 - b. производим обмен данного объекта с первым объектом неотсортированной части списка;
 - c. считаем следующей неотсортированной частью списка хвост текущей части (под хвостом понимается часть списка без первого элемента);
- 3) выполняем шаг 2, пока в неотсортированной части списка не останется 1 элемент.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 2

Ниже приведён алгоритм сортировки выбором, реализованный на языке программирования Python.

```
# n – количество эл-ов списка, которые необходимо упорядочить
n: int
# считаем, что список arr уже заполнен n числами
arr: list

# СОРТИРОВКА ВЫБОРОМ
for i in range(n-1):
    # выбираем минимальное значение в неотсортированной части
    списка
```

```

min_ind = i
for j in range(i, n):
    if arr[j] < arr[min_ind]:
        min_ind = j
# после того, как минимальный элемент найден, он помещается
# в начало неотсортированной части списка
arr[i], arr[min_ind] = arr[min_ind], arr[i]

# в этот момент элементы в списке arr упорядочены по неубыванию

```

Разберёмся, какое количество итераций внутреннего цикла в данном случае при выполнении описанного алгоритма. Так как алгоритм рассматривает как неотсортированные части списка – $n-1$ часть (весь список, весь список без первого элемента, весь список без первых двух элементов и т.д), и каждую неотсортированную часть проходит полностью, можно заключить, что

$$T(n) = (n - 1) + (n - 2) + \dots + 2 + 1 = \frac{(n - 1) * n}{2}$$

как и в случае с сортировкой пузырьком. То есть, $T(n) \in O(n^2)$.

ЧИТАЕМ

Сортировка вставками – простая сортировка, основная суть которой заключается в следующем:

- 1) считаем весь список объектов неотсортированным;
- 2) сортировка неотсортированной части списка:
 - a. выбираем первый объект из неотсортированной части списка;
 - b. сдвигаем часть отсортированной части списка, объекты в которой превышают выбранный, на один элемент вправо (сдвиг возможен за счёт выбранного элемента);
 - c. вставляем выбранный объект на освободившуюся позицию;
 - d. считаем следующей неотсортированной частью списка хвост текущей части (под хвостом понимается часть списка без первого элемента);
- 3) выполняем шаг 2, пока в неотсортированной части списка не останется ни одного элемента.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 3

Ниже приведён алгоритм сортировки вставками, реализованный на языке программирования Python.


```

# n – количество эл-ов списка, которые необходимо упорядочить
n: int
# считаем, что список arr уже заполнен n числами
arr: list

# СОРТИРОВКА ВСТАВКАМИ
for i in range(1, n):
    # выбираем первый элемент из неотсортированной части
    # считаем, что все элементы,
    # меньшие i – элементы отсортированной части
    this_elem = arr[i]

    # производим сдвиг всех элементов неотсортированной части,
    # которые превосходят this_elem, на одну позицию вправо
    j = i - 1
    while arr[j] > this_elem and j >= 0:
        arr[j + 1] = arr[j]
        j -= 1

    # в освободившееся место помещается (вставляется)
    # выбранный объект
    arr[j + 1] = this_elem

# в этот момент элементы в списке arr упорядочены по неубыванию

```

ОБСУЖДАЕМ

Обсудим:

1. В чем различия и в чем схожесть между сортировкой пузырьком, сортировкой выбором и сортировкой вставками?
2. Есть ли отличия в оценке худшего и лучшего их выполнения, можно ли их как-то модернизировать?
3. Какие идеи лежат в основе данных алгоритмов? Опишите их.

ЧИТАЕМ

Линеарифметические сортировки. Линеарифметическими называют сортировки, которые могут быть соотнесены с асимптотической сложностью $O(n \cdot \log(n))$. Несмотря на то, что при анализе алгоритмов нас в первую очередь интересует худшее время работы, в данном разделе представлен один из самых популярных методов сортировки на практике – quick sort или быстрая сортировка, худшее время работы которого – $O(n^2)$. Давайте рассмотрим идею, которая легла в основу создания данного алгоритма:

- 1) выберем объект из списка, назовём его опорным элементом;

2) перераспределим объекты в массиве таким образом, чтобы объекты, меньше опорного, помещались слева от него, а большие или равные – справа;

3) будем применять рекурсивно первые два шага по отношению к полученным левой и правой частям списка без включения опорного объекта.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 4

Используя язык программирования Python, можно написать следующую рекурсивную функцию:

```
def quick_sort(array: list):
    '''
    Функция, реализующая алгоритм быстрой сортировки Quicksort.
    Принимает в качестве аргумента одно значение
    -- список чисел:

    АРГУМЕНТЫ
    array : list
        список чисел.
    '''

    # если в качестве аргумента был подан пустой список,
    # не будем его обрабатывать
    if len(array) < 1:
        return list()

    # иначе обработаем поданный на вход список
    # для этого выберем опорный элемент
    sup_elem = array[0]

    # создадим списки для разделения
    l_list = list() # список объектов, меньших опорного
    e_list = list() # опорный объект и равные ему
    r_list = list() # список объектов, больших опорного

    # наполним эти списки
    for elem in array:
        if elem == sup_elem:
            # все элементы, равные значению опорного
            e_list.append(elem)
        elif elem < sup_elem:
            # все элементы левее опорного
```

```

        l_list.append(elem)
    else:
        # все элементы правее опорного
        r_list.append(elem)

# рекурсивно вызовем функцию
# для обработки списков l_list и r_list,
# после чего конкатенируем данные три списка
return quick_sort(l_list) + e_list + quick_sort(r_list)

```

ЧИТАЕМ

Лучшим временем выполнения данный алгоритм будет обладать в том случае, когда поступающий список объектов будет обладать действительно перемешанными значениями сортируемых объектов, а выбор опорного элемента всегда будет такой, что исходный список будет делиться на два равных по количеству элементов, это позволит нам утверждать, что глубина рекурсии будет равна $\log_2 n$, а с учётом того, что каждый вызов функции будет делать проход по максимум n элементов, мы можем утверждать, что в лучшем случае асимптотическая сложность алгоритма $T(n) \in O(n \cdot \log(n))$. Подобная асимптотическая сложность является очень хорошим результатом.

Но что будет, если в качестве входной будет подана уже отсортированная последовательность объектов? С учётом того, что в представленном алгоритме в качестве опорного объекта всегда берётся первый, такая ситуация будет порождать две подпоследовательности (не считая опорной), одна из которых всегда будет пустой, а вторая будет содержать все элементы первоначальной последовательности, за исключением опорного. Таким образом глубина рекурсии станет равна n (если точно, то $n-1$), а на каждом уровне будет осуществлен проход по всему списку. То есть мы приходим к выводу, что подобную временную сложность будет правильно оценить асимптотически как $O(n^2)$. Подобная ситуация является крайне маловероятной, поэтому на практике таким исходом чаще всего пренебрегают, так как данная сортировка за счёт использования малого количества элементарных операций чаще всего является быстрее других линейноарифметических сортировок.

Среднее же время выполнения данного алгоритма, рассчитываемое как математическое ожидание от всех возможных конфигураций последовательностей, даёт результат аналогичный лучшему времени выполнения – $T(n) \in O(n \cdot \log(n))$.

Сортировка слиянием – алгоритм сортировки, использующий принцип «разделяй и властвуй», как и предыдущий рассмотренный нами алгоритм сортировки – quick sort. Суть данного принципа заключается в разбиении

основной задачи на аналогичные задачи меньшего размера. В рамках данного метода сортировки этот принцип выглядит следующим образом:

- 1) разобьем изначальный список на две приблизительно равные части;
- 2) отсортируем каждую из частей по отдельности (рекурсивно вызвав данную функцию сортировки);
- 3) соединим данные отсортированные списки в один.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 5

Реализация данного алгоритма на языке программирования Python представлена ниже:

```
def merge_sort(array: list):  
    '''  
    Функция, реализующая алгоритм сортировки merge sort.  
    Принимает в качестве аргумента одно значение  
    -- список чисел:  
  
    АРГУМЕНТЫ  
    array : list  
        список чисел.  
    '''  
  
    # если нечего сортировать -- не сортируем  
    if len(array) < 2:  
        return array  
  
    # разбиваем изначальный список на две части  
    # и сортируем их независимо друг от друга  
    left_array = merge_sort(array[:len(array) // 2])  
    right_array = merge_sort(array[len(array) // 2:])  
  
    counter_l = 0    # счётчик для объектов левого списка  
    counter_r = 0    # счётчик для объектов правого списка  
    counter_all = 0  # счётчик упорядоченных элементов  
                    # в исходном списке  
  
    result_array = [0] * len(array)  # список для расстановки  
                                     # объектов в правильном  
порядке  
  
    # выставляем элементы из двух отсортированных списков
```

```

# left_array и right_array
while counter_l < len(left_array) and counter_r <
len(right_array):
    if left_array[counter_l] <= right_array[counter_r]:
        result_array[counter_all] = left_array[counter_l]
        counter_l += 1
    else:
        result_array[counter_all] = right_array[counter_r]
        counter_r += 1
    counter_all += 1

# если в каком-то из списков остались элементы,
# выставляем их
while counter_l < len(left_array):
    result_array[counter_all] = left_array[counter_l]
    counter_l += 1
    counter_all += 1
while counter_r < len(right_array):
    result_array[counter_all] = right_array[counter_r]
    counter_r += 1
    counter_all += 1

# записываем в изначальный список отсортированные элементы
for i in range(len(array)):
    array[i] = result_array[i]

return array

```

Давайте проведём анализ данного алгоритма. Очевидно, что разбиение пополам изначального списка из n элементов будет продолжаться $\log(n)$ раз, при этом на каждом уровне рекурсии мы будем выполнять суммарно n действий прохода по спискам. Таким образом, мы можем заключить, что временная сложность данного алгоритма оценивается как $T(n) \in O(n \cdot \log(n))$.

Стоит отметить, что для данного алгоритма ситуация наихудшего и наилучшего времени выполнения совпадают, так как разбиение списка на части не зависит от значений оцениваемых характеристик объектов.

ОБСУЖДАЕМ

Обсудим:

1. В чем различия и в чем схожесть между быстрой сортировкой и сортировкой слиянием?
2. Есть ли отличия в оценке худшего и лучшего их выполнения?

3. Какой принцип в качестве базовой идеи лёг в создание данных алгоритмов?

ГОТОВИМСЯ К РАБОТЕ НА ПК

Зачастую при создании программы нам важно не только получить результат, но и сделать это, затратив как можно меньше времени. Наиболее простой способ узнать время работы части кода выглядит следующим образом:

```
import time # импортируем модуль

start_time = time.time() # отмечаем текущее время до начала
                          # работы программы
# -----основной код программы-----
# вычитаем из времени start_time текущее время после выполнения
result_time = start_time - time.time()
```

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №63

Попробуйте экспериментально сравнить время работы представленных алгоритмов на одних и тех же данных. Для этого можно воспользоваться модулем **time**. С учётом того, что разница во времени может быть найдена на данных большого объема, рекомендуем случайным образом производить генерацию тестов – для этого воспользуйтесь модулем **random**.

ЧИТАЕМ

Непрактичные сортировки. Кроме действительно используемых в практической деятельности алгоритмов сортировки, существуют сортировки, которые на практике не применимы для решения действительных задач. Подобные сортировки обычно используются для иллюстраций и в качестве шутки. Примером одной из таких сортировок является болотная сортировка или Bogo Sort. Основная идея, которая легла в основания данного алгоритма, заключается в следующем – случайным образом перемешиваем элементы списка и проверяем, упорядочились ли они; если не упорядочились, снова мешаем и т.д. Реализация на Python рассмотрена ниже:

```
import random # так как алгоритм сортировки bogosort
              # основан на случайных перестановках,
              # то нам потребуются соответствующие инструменты

def bogosort(array: list):
    '''
```

Функция, реализующая неэффективный алгоритм сортировки Bogosort -- никогда им не пользуйтесь, так как его СРЕДНЯЯ временная сложность -- $O(n * n!)$, и есть вероятность, что алгоритм никогда не придёт к окончанию выполнения.

Принимает в качестве аргумента одно значение
-- список чисел:

АРГУМЕНТЫ

array : list

список чисел.

'''

пока элементы array не упорядочены,

случайно переставляем их

while not is_arranged(array):

array = permutation(array)

return array

def permutation(array: list):

'''

Вспомогательная функция случайного перемешивания,
используемая в реализации функции bogosort.

АРГУМЕНТЫ

array : list

список чисел, которые необходимо упорядочить

'''

определяем новое положение для каждого элемента

случайным образом

for i in range(len(array)):

j = random.randint(0, len(array) - 1)

array[i], array[j] = array[j], array[i]

return array

def is_arranged(array: list):

'''

Вспомогательная функция проверки порядка,
используемая в реализации функции bogosort.

```

АРГУМЕНТЫ
array : list
    список чисел, которые необходимо упорядочить
'''

# определяем новое положение для каждого элемента
# случайным образом
for i in range(len(array) - 1):
    if array[i] < array[i + 1]:
        return False
return True

```

Сложность данного алгоритма в некоторых исключительных ситуациях может проходить за $O(n)$, в случае если после первого перемешивания элементы действительно встали на свои места. Средняя временная сложность оценивается как $O(n \cdot n!)$, что должно занимать довольно большое время... А вот худшее время выполнения данного алгоритма неизвестно. При условии, если бы на компьютере действительно можно было бы генерировать случайные числа, даже маленький список объектов мог бы сортироваться бесконечное количество времени.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1.

Найдите в интернете описание алгоритма гномьей сортировки. Напишите программу, реализующую данный алгоритм, которая сортирует список списков по длине вложенных списков (и по сумме элементов вложенных списков).

Задание 2.

Преобразуйте сортировку слиянием таким образом, чтобы она была применима при решении задачи сортировки списка квадратных матриц по их сумме.

Задание 3.

Преобразуйте алгоритм быстрой сортировки таким образом, чтобы она была применима при решении задачи сортировки целых чисел по старшему разряду.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Назовите основные виды алгоритмов по сложности, которые используются в практической деятельности.
2. В чём разница между худшим, средним и лучшим временем выполнения? Какая из этих оценок наиболее применима при анализе конкретного алгоритма?
4. Для каких целей чаще используются непрактичные сортировки и квадратичные сортировки?
5. Назовите основные алгоритмы сортировок, которые вы знаете.

§7.6 Решение задач ЕГЭ (№27)

ВСПОМИНАЕМ

Давайте вспомним:

1. Что такое асимптотическая сложность алгоритма?
2. Как оценивается асимптотика выполнения алгоритма по времени и по памяти?

ЧИТАЕМ

В рамках данного параграфа будут разобраны классы задач, основной спецификой которых является наличие алгоритма решения, который можно описать временной сложностью $T(n) \in O(n)$ с потреблением памяти $M(n) \in O(1)$, то есть линейного алгоритма, память для работы которого не зависит от размера входных данных.

Примером одного из таких классов задач, может являться класс задач обработки групп чисел. Давайте разберём задачу и на её примере рассмотрим оптимизацию, которая позволит добиться оптимального времени выполнения $T(n)$ и оптимального количества занимаемой памяти $M(n)$.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 1

Условие:

Пусть у нас есть n пар чисел. Наша программа должна выбрать из каждой пары число таким образом, чтобы сумма выбранных чисел не делилась на 2 и была максимальной.

Рассуждение:

Самая первая и простая для воспроизведения идея, которая может возникнуть в голове у человека, решающего данную задачу – произвести перебор всех возможных вариантов комбинаций чисел. Реализация подобного

алгоритма с помощью языка программирования Python может выглядеть следующим образом:

```
n = int(input()) # введём количество двоек

pair_list = list() # список, в котором мы будем хранить пары чисел

# произведём ввод всех пар чисел
for _ in range(n):
    # вводим очередную пару и сохраняем её как кортеж двух чисел
    pair = tuple(map(int, input().split()))
    # сохраняем пару в список
    pair_list.append(pair)

# список, в котором для i-ой пары чисел хранится номер числа,
# который мы будем учитывать в сумме (сначала все -- 0)
# ПРИМЕЧАНИЕ: подобную информацию можно хранить в виде одного
# числа,
# рассматриваемого в двоичной системе счисления
num_number = [0] * n

res_sum = -1
while sum(num_number) != n:
    temp_sum = 0 # сумма очередного набора
    # считаем сумму
    for i in range(n):
        temp_sum += pair_list[i][num_number[i]]
    # если она удовлетворяет условию и больше того, что мы нашли
    # ранее
    if temp_sum > res_sum and temp_sum % 2 != 0:
        res_sum = temp_sum

    # берём следующее лексикографическое распределение
    # выбора элементов
    k = 0
    while True:
        temp = num_number[k]
        num_number[k] = (num_number[k] + 1) % 2
        if temp == 0 or k == len(num_number) - 1:
            break
        k += 1

temp_sum = 0 # сумма последнего набора
for i in range(n):
```

```

    temp_sum += pair_list[i][num_number[i]]
# если она удовлетворяет условию и больше того, что мы нашли
ранее
if temp_sum > res_sum and temp_sum % 2 != 0:
    res_sum = temp_sum

print(res_sum)

```

Давайте оценим данный алгоритм по времени выполнения. С учётом того, что мы перебираем все возможные комбинации длины n , составленные из нулей и единиц, то по знакомой формуле $N = 2^i$ мы можем заключить, что этих комбинаций 2^n штук. Таким образом временная сложность данного алгоритма может быть оценена как экспоненциальная $T(n) \in O(2^n)$, что является очень плохим результатом и совсем не соотносится с той временной сложностью, что мы обозначили выше.

Идея, с помощью которой данный алгоритм можно модифицировать и уйти от экспоненциальной сложности, заключается в совсем простом факте – нам надо из каждой пары взять максимальный элемент. Если полученная сумма не делится на два, то всё хорошо. Если полученная сумма все же делится на два, нам надо заменить какой-то элемент, учтённый в данной сумме и имеющий в качестве пары число другой чётности. При этом нам необходимо, чтобы подобная замена минимально сказалась на результирующей сумме. Для этого достаточно найти пару чисел разной чётности с минимальной разностью.

```

n = int(input()) # введём количество двоек

pair_list = list() # список, в котором мы будем хранить пары
чисел

# произведём ввод всех пар чисел
for _ in range(n):
    # вводим очередную пару и сохраняем её как кортеж двух чисел
    pair = tuple(map(int, input().split()))
    # сохраняем пару в список
    pair_list.append(pair)

res_sum = 0
min_dist = 1000000000000 # считаем, что разница между числами
любой

# пары не превосходит данного числа
for i in range(n):
    # учитываем в сумме максимальный элемент
    res_sum += max(pair_list[i])

```

```

        # находим минимальную разницу между элементами разной
чётности
        # ОЧЕВИДНО, что сумма четного и нечетного --
нечетное число
        if (pair_list[i][0] + pair_list[i][1]) % 2 == 1:
            min_dist = min(min_dist, abs(pair_list[i][0] -
pair_list[i][1]))

# проверяем полученный результат
if res_sum % 2 == 0:
    # и меняем его на минимальную разность чисел разной
чётности,
    # чем и меняем его четность
    res_sum -= min_dist

print(res_sum)

```

При подобном решении мы проходим всего один раз по списку, что позволяет нам утверждать, что временная сложность в данном случае линейно зависит от входных данных, то есть $T(n) \in O(n)$. Подводный результат временной сложности нас полностью устраивает. Но мы не оценивали затраты памяти. К слову, с учётом того, что и в первой и второй реализации мы храним только список пар (во первой реализации ещё список информации о том, какое число мы берем из каждой пары, что сказывается только в изменении константы), стоит отметить, что затраты по памяти мы можем оценить как $M(n) \in O(n)$, что не всегда является оптимальным. В рамках данной задачи нам не надо запоминать обрабатываемые числа, так как к ним мы обращаемся только один раз, так что использование списка в данном случае не является необходимым.

```

n = int(input()) # введём количество двоек

res_sum = 0
min_dist = 1000000000000 # считаем, что разница между числами
любой
# пары не превосходит данного числа

for i in range(n):
    # вводим очередную пару и сохраняем её
    # в виде кортежа на одну итерацию
    pair = tuple(map(int, input().split()))
    # учитываем в сумме максимальный элемент
    res_sum += max(pair)
    # находим минимальную разницу между элементами разной
чётности

```

```

#             ОЧЕВИДНО, что сумма четного и нечетного --
нечетное число
    if (pair[0] + pair[1]) % 2 == 1:
        min_dist = min(min_dist, abs(pair[0] - pair[1]))

# проверяем полученный результат
if res_sum % 2 == 0:
    # и меняем его на минимальную разность чисел разной
    четности,
    # чем и меняем его четность
    res_sum -= min_dist

print(res_sum)

```

При подобном решении размеры памяти, требуемой для выполнения программы, не изменятся в зависимости от размера входных данных, то есть, мы добились рекомендуемой оценки $M(n) \in O(1)$.

ЧИТАЕМ

Обратите внимание, что в разобранный выше примере решение, которое мы называли «оптимальным», было более компактно и проще написано, чем два альтернативных алгоритма. Вопрос прихода к такому решению может быть трудным за счёт «привычек» человеческого мышления к принципу алгоритма перебора, который чаще используется в бытовой деятельности. Важной задачей при обучении программированию является отказ от наработанных в быту алгоритмов и пересмотр существующих задач с точки зрения компьютерной метафоры – удобства представления и обработки данных в компьютере.

Так под оптимальным решением мы будем понимать алгоритм, который задействует наименьшее возможное количество памяти и времени при его асимптотической оценке. Очевидно, что алгоритм, при исполнении которого количество памяти не зависит от количества входных данных, является асимптотически оптимальным – ничего лучше этого просто быть не может. При действиях с целью улучшить алгоритм можно говорить про то, что называется микрооптимизацией – изменение реализации алгоритма с целью обеспечить быстроту его выполнения или уменьшение требований к памяти без изменения его асимптотической оценки, узкими примерами чего может быть использование двух переменных вместо трёх, что, очевидно, сказывается на занимаемой памяти при выполнении алгоритма (микрооптимизация по памяти), или уменьшение количества сравнений внутри цикла – что скажется на времени выполнения, так как уменьшится количество элементарных операций (микрооптимизация по времени). Подобные действия лишь меняют константу в описании функции требования к ресурсам (константу из

определений асимптотических оценок), но, формально, улучшают сам алгоритм.

При решении задач из описанного в начале параграфа класса задач нас в первую очередь интересует асимптотическая оптимизация, но не стоит забывать о том, чтобы стараться писать код оптимально с точки зрения микрооптимизации — подобный подход положительно сказывается на формировании культуры программирования.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №63

Напишите программу, которая принимает n пар чисел. Программа должна выбрать из каждой пары одно число таким образом, чтобы сумма выбранных чисел не делилась на 3 и была максимально возможной. Программа должна быть оптимальной по памяти и по времени.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №64

Напишите программу, которая принимает n троек чисел. Программа должна выбрать из каждой тройки одно число таким образом, чтобы сумма выбранных чисел не делилась на 5 и была максимально возможной. Программа должна быть оптимальной по памяти и по времени.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

ПРИМЕР 2

Условие:

Рассмотрим решение ещё одной задачи. Пусть нам необходимо написать программу, которая считывает n чисел. И среди считанных чисел находит количество пар чисел, которые в произведении друг с другом дают число, делящееся на 6.

Рассуждение:

Давайте подумаем, какие числа в произведении могут дать число, делящееся на 6. Очевидно, что

- 1) если одно число делится на 6, то его произведение с любым числом тоже будет делиться на 6;
- 2) если один множитель делится на 3, а второй делится на 2, то их произведение делится на 6.

В таком случае для решения поставленной задачи нам достаточно найти

- 1) количество таких чисел в последовательности, которые делятся на 6;

2) количество таких чисел в последовательности, которые делятся на 3, но не делятся на 6;

3) количество таких чисел в последовательности, которые делятся на 2, но не делятся на 6.

Условие неделимости на 6 во втором и третьем пункте введены для того, чтобы не учитывать пересечение, ведь очевидно, что любое число, делящееся на 6 одновременно с этим, делится на 2 и на 3.

После вычисления информации выше, мы можем утвердить, что результат решения задачи складывается из следующих слагаемых:

1) количество всех пар чисел, делящихся на 6;

2) количество всех пар чисел таких, что одно число делится на 6, а второе не делится на 6;

3) количество всех пар чисел таких, что одно число делится на 3, а второе делится на 2.

Основываясь на данных рассуждениях, можно написать следующую программу на языке программирования Python, представленную ниже.

```
n = int(input()) # введём количество чисел в последовательности
```

```
# будем хранить информацию о количестве чисел в последовательности
```

```
div_6 = 0 # делящихся на 6
```

```
div_3 = 0 # делящихся на 3, но не делящихся на 6
```

```
div_2 = 0 # делящихся на 2, но не делящихся на 6
```

```
# произведем подсчет количества чисел
```

```
# в соответствии с упомянутыми критериями
```

```
for _ in range(n):
```

```
    temp = int(input())
```

```
    if temp % 6 == 0:
```

```
        div_6 += 1
```

```
    elif temp % 3 == 0:
```

```
        div_3 += 1
```

```
    elif temp % 2 == 0:
```

```
        div_2 += 1
```

```
# количество всех пар чисел, делящихся на 6
```

```
both_div_6 = div_6 * (div_6 - 1) // 2
```

```
# количество всех пар чисел таких,
```

```
# что одно число делится на 6, а второе не делится на 6
```

```
not_both_div_6 = mod_6 * (n - mod_6)
```

```
# количество всех пар чисел таких,  
# что одно число делится на 3, а второе делится на 2  
div_2_3 = mod_2 * mod_3
```

```
# выводим сумму описанных пар чисел,  
# произведение которых делится на 6  
print(both_div_8 + not_both_div_6 + div_2_3)
```

Очевидно, что у нас нет необходимости рассчитывать слагаемые суммы, являющейся ответом на поставленную задачу в том виде, в котором это сделано в представленном коде:

```
both_div_8 = div_6 * (div_6 - 1) // 2  
not_both_div_6 = mod_6 * (n - mod_6)  
div_2_3 = mod_2 * mod_3
```

```
print(both_div_8 + not_both_div_6 + div_2_3)
```

И данный код можно заменить более компактной альтернативой, сразу рассчитывающей ответ на задачу:

```
print(div_6 * (div_6 - 1) // 2 + div_6 * (n - div_6) + div_2 *  
div_3)
```

Расчёт каждого слагаемого отдельно был произведен с целью более подробной иллюстрации.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №66

Напишите программу, которая принимает n чисел. Программа должна найти количество всех возможных пар из данной последовательности, элементы которых в произведении будут давать число, делящееся на 26. Программа должна быть оптимальной по памяти и по времени.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №67

Напишите программу, которая принимает n чисел. Программа должна вывести максимальное число, делящееся на 3. Программа должна быть оптимальной по памяти и по времени.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1.

Напишите программу, которая принимает n пар чисел. Программа должна выбрать из каждой пары одно число таким образом, чтобы сумма выбранных чисел не делилась на 3 и была минимально возможной. Программа должна быть оптимальной по памяти и по времени.

Задание 2.

Напишите программу, которая принимает n пар чисел. Программа должна выбрать пару, для которой верно, что разность элементов данной пары чётна, хотя бы один элемент пары не делится на 17 и сумма элементов пары является максимальной (из всех пар). Программа должна быть оптимальной по памяти и по времени.

Задание 3.

Напишите программу, которая принимает n чисел. Программа должна найти количество всех возможных пар из данной последовательности, элементы которых в произведении будут давать число, делящееся на 58. Программа должна быть оптимальной по памяти и по времени.

VIII. Алгоритмы

§8.1 Эффективное вычисление НОД и НОК, алгоритм Евклида

ВСПОМИНАЕМ

Вспомним:

1. Что такое алгоритм?
2. Что такое асимптотическая сложность алгоритма?
3. Что такое наибольший общий делитель и наименьшее общее кратное?

ЧИТАЕМ

Наверняка каждый из вас уже знает, что программирование, как в общем случае и математика, являются мощными инструментами познания действительности и оптимизации деятельности практически во всех сферах жизни человека. Если вы хотите грамотно использовать инструмент “программирование”, для вас является важной задачей усвоение понятий классических алгоритмов, чтобы “не изобретать велосипеды” и не подпираться ваши программы “костылями” (*Тут надо отметить, что это устойчивые выражения в сфере разработки программного обеспечения, где “изобретение велосипеда” означает решение уже решённой задачи своим методом, который очень часто на практике является неоптимальным; а “использование костылей” — это действия по исправлению ошибок написанной программы через добавление кусков кода, которые не подразумеваются в общей структуре программы или не работают в общем случае, что является плохой практикой при написании кода, и если возникает необходимость добавить “костыль”, то стоит пересмотреть сам подход к организации написанного алгоритма*) при решении очередной задачи.

Это ни в коем случае не означает, что вы не должны реализовывать свои новые идеи, но помните, что опора на огромный культурный багаж человеческих знаний сделает вашу деятельность эффективнее.

Само собой, в программировании вы будете встречаться с новыми задачами, и знание шаблонов (прим. алгоритмов) не всегда будет вам помогать в форме их буквального использования, но понимание самого процесса решения задачи в рамках алгоритма позволит вам по-новому рассмотреть стоящую перед вами задачу и через конкретизацию модернизировать существующий алгоритм.

ОБСУЖДАЕМ

Обсудите, что может являться “велосипедом” или “костылем” на примере тех программ, что вы писали ранее.

ЧИТАЕМ

Наибольший общий делитель (или НОД) для двух целых чисел a и b — это некоторое целое число c , для которого верны следующие утверждения:

- 1) числа a и b делятся на c без остатка;
- 2) из всех чисел, на которые a и b делятся без остатка, c является наибольшим.

Аналогичным образом можно определить и НОД для n -целых чисел. Наибольший общий делитель (или НОД) для n -целых чисел a_1, a_2, \dots, a_n — это некоторое целое число c , для которого верны следующие утверждения:

- 1) числа a_1, a_2, \dots, a_n делятся на c без остатка;
- 2) из всех чисел, на которые a_1, a_2, \dots, a_n делятся без остатка, c является наибольшим.

В различной литературе вы можете встретить разные обозначения для наибольшего общего делителя n -целых чисел a_1, a_2, \dots, a_n ; наиболее частые обозначения :

- $\text{GCD}(a_1, a_2, \dots, a_n)$;
- $\text{HCF}(a_1, a_2, \dots, a_n)$;
- $\text{НОД}(a_1, a_2, \dots, a_n)$;
- a_1, a_2, \dots, a_n .

Наименьшее общее кратное (или НОК) для двух целых чисел a и b — это наименьшее натуральное число, кратное одновременно и числу a , и числу b .

В различной литературе вы можете встретить разные обозначения для наименьшего общего кратного n -целых чисел a_1, a_2, \dots, a_n . В основном используются аббревиатуры: GCD — greatest common divisor; HCF — highest common factor; НОД — наибольший общий делитель. Но также может использоваться простой скобочный оператор (\dots) .

А для НОК: LCM — least common multiple; НОК — наименьшее общее кратное. Но также может использоваться скобочный оператор $[\dots]$.

- $\text{LCM}(a_1, a_2, \dots, a_n)$;
- $\text{НОК}(a_1, a_2, \dots, a_n)$;
- $[a_1, a_2, \dots, a_n]$.

ОБСУЖДАЕМ

Дайте определение наименьшего общего кратного для n -целых чисел самостоятельно по аналогии с тем, как мы получили ранее определение наибольшего общего делителя для n -целых чисел.

ЧИТАЕМ

С точки зрения математики существует простой и эффективный способ нахождения наибольшего общего делителя двух целых чисел, отличных от нуля. Данный алгоритм хорошо вам знаком из школьного курса математики – алгоритм Евклида. Напомним, что алгоритм выглядит следующим образом:

Пусть a и b – целые числа, не равные одновременно нулю, тогда построение системы следующих утверждений однозначно определяет наибольший общий делитель чисел a и b ,

$$\begin{cases} a = b * q_0 + r_1, & 0 < r_1 < b \\ b = r_1 * q_1 + r_2, & 0 < r_2 < r_1 \\ r_1 = r_2 * q_2 + r_3, & 0 < r_3 < r_2 \\ \dots \\ r_{n-2} = r_{n-1} * q_{n-1} + r_n, & 0 < r_n < r_{n-1} \\ r_{n-1} = r_n * q_n \end{cases}$$

который равен числу r_n .

Обратите внимание, что последнее уравнение получается тогда и только тогда, когда число r_{n+1} равно нулю, то есть, можно сказать, что наибольшим общим делителем для a и b является последний ненулевой остаток алгоритма Евклида.

Важно обратить внимание на то, что $(a, b) = (b, r_1) = (r_1, r_2) = \dots = (r_{n-2}, r_{n-1}) = (r_{n-1}, r_n) = r_n$, что мы в полном праве использовать при создании компьютерной реализации алгоритма. Из рассуждений выше видно, что нас не интересует значение q_i .

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Первый вариант реализации, который мы рассмотрим, является заведомо неоптимальным, но наиболее тривиальным (пример 1).

ПРИМЕР 1

```
a: int # считаем, что переменные a и b
b: int # имеют заданные значения

# будем продолжать выполнять действия,
# пока не получим нулевой остаток
while a * b:
    if a > b:
        a -= b
    else:
        b -= a
```

```

# в качестве результата принимаем сумму значений
# переменных a и b, так как в любом случае
# одно из них равно нулю, а второе является
# искомым значением наибольшего общего делителя
# изначальных значений переменных a и b
res: int = a + b

```

В примере 1 за q_k шагов цикла происходит вычитание из большего числа r_{k-1} меньшего числа r_k ровно q_k раз, после чего в качестве остатка в уменьшаемой переменной остается число r_{k+1} . То есть для получения следующего остатка подобная программа должна совершить q_k итераций.

Но мы помним факт

$$(a, b) = (b, r_1) = (r_1, r_2) = \dots = (r_{n-2}, r_{n-1}) = (r_{n-1}, r_n) = r_n$$

что позволяет нам перейти к операции получения остатка при делении, что должно ускорить работу вычисления наибольшего общего делителя (пример 2).

ПРИМЕР 2

```

a: int # считаем, что переменные a и b
b: int # имеют заданные значения

# будем продолжать выполнять действия,
# пока не получим нулевой остаток
while a * b:
    if a > b:
        a %= b
    else:
        b %= a

# в качестве результата принимаем сумму значений
# переменных a и b, так как в любом случае
# одно из них равно нулю, а второе является
# искомым значением наибольшего общего делителя
# изначальных значений переменных a и b
res: int = a + b

```

При этом данный код можно очень упростить с точки зрения написания (пример 3), не изменив его сложности $O(\log \min(a, b))$.

ПРИМЕР 3

```

a: int # считаем, что переменные a и b
b: int # имеют заданные значения

# будем продолжать выполнять действия,
# пока не получим нулевой остаток
while b:
    a, b = b, a % b

res: int = a

```

Также существует бинарный алгоритм Евклида, отличающийся реализацией от рассмотренных выше. Основным его преимуществом является скорость выполнения за счёт использования бинарного деления и бинарных сдвигов. Но данного преимущества нет при реализации алгоритма на языке программирования Python, так что при программировании на Python мы не рекомендуем его использовать, более подходящий язык программирования – C++. Ниже приведен код для ознакомления (пример 4).

ПРИМЕР 4

```

a: int # считаем, что переменные a и b
b: int # имеют заданные значения

shift: int = 0 # размер сдвига
res: int # НОД

if a == 0:
    res = a
elif b == 0:
    res = b
else:
    # пока хотя бы одно из чисел a и b
    # делятся на 2, делим их на два
    # (выполняем бинарный сдвиг вправо на 1 бит)
    while (a | b) & 1 == 0:
        shift += 1
        a >>= 1
        b >>= 1

    # пока a -- чётное, делим его на 2
    # (выполняем бинарный сдвиг вправо на 1 бит)
    while a & 1 == 0:
        a >>= 1

```

```

#
while b != 0:
    # пока a -- чётное, делим его на 2
    # (выполняем бинарный сдвиг вправо на 1 бит)
    while b & 1 == 0:
        b >>= 1

    if a > b:
        a, b = b, a

    b -= a

# результат будет записан в a со сдвигом
# на shift бит вправо, поэтому необходимо
# произвести обратный -- левый сдвиг
# на shift бит
res = a << shift

# по итогу выполнения данного алгоритма
# в переменной res будет записано значение НОД
# для первоначальных значений переменных a и b

```

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №68

Напишите рекурсивную функцию нахождения наибольшего общего делителя для произвольного количества аргументов.

Примечание. Для НОД верно, что

$$\text{gcd}(a_1, a_2, \dots, a_n) = \text{gcd}(\text{gcd}(a_1, a_2, \dots, a_{n-1}), a_n)$$

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Напишем код для нахождения наименьшего общего кратного произвольного количества целых чисел, для этого обернем уже разобранный алгоритм нахождения НОД для двух чисел в функцию (пример 5).

ПРИМЕР 5

```

def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

```

После чего напишем функцию для нахождения НОК для двух целых чисел (пример 6), которая будет обращаться к написанной нами ранее функции.

ПРИМЕР 6

```
def lcm(a, b):  
    return a * b // gcd(a, b)
```

И финальным шагом будет написание функции для нахождения НОК произвольного числа аргументов, которая будет рекурсивно действовать согласно упомянутому выше свойству (пример 7).

Пример 7

```
def lcm_multi(*args):  
    if len(args) == 2:  
        return lcm(args[0], args[1])  
    else:  
        return lcm(lcm_multi(*args[0:-1]), args[-1])
```

Таким образом, у нас получится следующий код:

```
def gcd(a, b):  
    while b:  
        a, b = b, a % b  
    return a
```

```
def lcm(a, b):  
    return a * b // gcd(a, b)
```

```
def lcm_multi(*args):  
    if len(args) == 2:  
        return lcm(args[0], args[1])  
    else:  
        return lcm(lcm_multi(*args[0:-1]), args[-1])
```

```
lcm_multi(3, 5, 9)    # 45  
lcm_multi(2, 4)       # 4  
lcm_multi(2, 4, 9, 3) # 36
```

Попробуйте самостоятельно реализовать данные программы и протестировать их.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №69

Напишите программу, которая принимала бы на вход два натуральных числа a и b , и проверяла бы, является ли дробь

$$\frac{a}{b}$$

несократимой.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1.

На циферблате необычных часов четыре стрелки. Известно, что первая стрелка делает полный оборот за k секунд, вторая – за l секунд, третья – за m секунд, четвёртая – за n секунд. Если в некоторый момент времени все стрелки находятся на одном делении, через сколько секунд своего хода от этого момента они снова встретятся на этом же делении?

Ввод. Вводятся четыре числа k, l, m, n ($1 \leq k, l, m, n \leq 10^4$).

Вывод. Выводится одно число – количество секунд.

Задание 2.

На плоскости существует отрезок, концы которого имеют целочисленные координаты. Необходимо посчитать количество точек на данном отрезке, имеющих целочисленные координаты.

Ввод. На вход программе подаётся две пары целых чисел x_1, y_1, x_2, y_2 , описывающих отрезок ($-10^9 \leq x_1, y_1, x_2, y_2 \leq 10^9$).

Вывод. В качестве ответа программа должна вывести одно число – количество целочисленных координат на отрезке (концы также необходимо учитывать).

Примечание. Утверждается, что количество целочисленных точек гипотенузы равно наибольшему общему делителю длин катетов плюс один, при условии, что концы отрезков катетов являются целочисленными точками.

Задание 3.

Прямоугольный треугольник на плоскости задан координатами своих вершин. Необходимо посчитать количество точек с целочисленными координатами, лежащими во внутренней области данного треугольника.

Ввод. На вход программе подаётся две пары целых чисел x_1, y_1, x_2, y_2 , описывающих гипотенузу, и пара чисел x_3, y_3 для формирования треугольника ($-1 \cdot 10^9 \leq x_1, y_1, x_2, y_2, x_3, y_3 \leq 10^9$).

Вывод. В качестве ответа программа должна вывести одно число – количество целочисленных координат во внутренней области треугольника.

Примечание. Для решения данной задачи удобно воспользоваться формулой Пика, которая связывает количество целочисленных точек на границе плоской фигуры и количество целочисленных точек во внутренней области фигуры с её площадью:

$$S = n + \frac{m}{2} - 1,$$

где S – площадь фигуры, n – количество точек во внутренней области, m – количество точек на границе фигуры.

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. Виноградов И. М. Основы теории чисел. — М.—Л.: ГИТТЛ, 1952. — 180 с.
2. Кнут Д.Э. Искусство программирования. Том 1. Основные алгоритмы. 3-е издание. Перевод с английского. Под общей редакцией Ю.В. Козаченко.

§8.2 Тесты на простоту, решето Эратосфена

ВСПОМИНАЕМ

Вспомним:

1. Что такое простое число?
2. Какие определения простоты вы знаете?

ЧИТАЕМ

Понятие простого числа в современной математике занимает важное место и используется в большом количестве математических областей в качестве инструмента исследования: арифметика, теория чисел, теория групп, теория шифрования и т. д. Существует несколько нерешенных математических задач, связанных с понятием простого числа. Простые числа сильно связаны и с программированием, особенно со сферами представления и шифрования информации.

Для того, чтобы использовать утилитарно простые числа в своей практической деятельности, в первую очередь необходимо определить,

является ли произвольное число простым. Решение данной задачи может быть получено большим количеством способов с различной нагрузкой на вычислительные и временные ресурсы. Несколько классических алгоритмов для решения данной задачи будет разобрано в рамках данного урока.

Поясним, что под простым числом классически понимается натуральное число n , имеющее ровно два различных делителя – единицу и само число n .

Проверка на простоту перебором делителей

Исходя из определения простого числа, можно сделать предположение, что нам достаточно подсчитать количество делителей при проверке числа на простоту – это действительно является достаточным условием для решения поставленной задачи.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

С точки зрения реализации алгоритма это может быть представлено в виде следующего кода (пример 1).

ПРИМЕР 1

```
# пусть в n записано предполагаемое простое число
n: int

# будем считать, что переменная count отвечает
# за количество делителей
count: int = 0

# произведём подсчёт всех возможных делителей
for i in range(1, n + 1):
    if n % i == 0:
        count += 1

# если делителей у числа n -- 2, то n -- простое
# иначе число n не является простым числом
is_prime: bool
if count == 2:
    is_prime = True
else:
    is_prime = False
```

Вроде всё просто, но для больших чисел данный алгоритм будет работать очень долго, его время работы, очевидно, есть $O(n)$. Мы можем его немного оптимизировать. Важно заметить, что нам не надо считать все делители числа – нам необходимо лишь найти делитель, отличный от единицы

и самого числа. При этом мы можем пройти до корня от числа, так как, если число n является составным, то один из его делителей q точно будет находиться на отрезке:

$$2 \leq q \leq \sqrt{n}$$

Данный факт следует из весьма простых рассуждений. Если число n является составным, то очевидно, что его можно представить в виде $n=q_1*q_2$, при этом $1 < q_1 \leq q_2$, откуда и следует соотношение

$$q_1 \leq \sqrt{n}$$

Именно по данному промежутку будет достаточно пройти при проверке числа на простоту, при этом асимптотически подобный алгоритм мы уже можем оценить как

$$O(\sqrt{n})$$

Его реализация описана ниже (пример 2).

ПРИМЕР 2

```
# пусть в n записано предполагаемое простое число
n: int

# предположим, что число n - простое,
# если оно не является единицей
is_prime: int = True if n != 1 else False

# попробуем опровергнуть данную гипотезу
# * мы будем пользоваться циклом while для того,
# чтобы не вычислять квадратный корень

div: int = 2 # перебираемый делитель
while div * div <= n:
    if n % div == 0:
        is_prime = False
        break
    div += 1
```

С помощью данного алгоритма можно легко определить простоту числа.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №71

Напишите программу, которая проверяет n чисел на простоту. Сначала вводится число n — количество чисел, которые необходимо проверить на простоту. Затем вводятся n чисел. Для каждого числа выведите “prime”, если число простое и “not prime”, если число не является простым.

ЧИТАЕМ

Но что если нам необходимо определить на отрезке все простые числа? Мы можем пройти по всему отрезку и проверять каждое число с помощью перебора его делителей, но будет ли это оптимально?

Давайте ознакомимся с культурным достоянием математической мысли, для того чтобы ответить на поставленный вопрос.

Решето Эратосфена

Решето Эратосфена — это алгоритм, позволяющий найти все простые числа из отрезка $[2, n]$ при заданном n .

Суть данного алгоритма заключается в следующем: сначала мы исходим из предположения, что все числа являются простыми и начинаем рассматривать их в порядке возрастания. Первое число — 2, мы говорим, что 2 простое и вычеркиваем из всего рассматриваемого ряда числа, кратные 2. Переходим к следующему числу, которое мы не вычеркнули (т.е. 3), и утверждаем, что оно простое, после чего вычёркиваем все следующие числа, кратные 3. Затем производится переход к следующему не зачёркнутому числу (т.е. 5), которое является простым, и вычеркиваем из следующих чисел ряда все, кратные пяти. Алгоритм таким образом продолжается до необходимой границы n , которая может быть заменена

$$\sqrt{n}$$

что можно легко доказать.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Запишем данный алгоритм с помощью языка программирования Python (пример 3).

ПРИМЕР 3

```
# пусть в n записана правая точка отрезка
n: int

# создаём список из (n + 1) элементов
# данный список по номеру элемента
# будет хранить вердикт о его простоте
bool_arr: list = [True] * (n + 1)
```

```

bool_arr[0] = False # ноль не является натуральным
bool_arr[1] = False # единица не является простым

num: int = 2 # num в данном случае является очередным
              # рассматриваемым числом
while num * num <= n:
    # если очередное число является простым
    if bool_arr[num] == True:
        # то мы вычёркиваем все кратные ему числа
        for i in range(num * num, n + 1, num):
            bool_arr[i] = False
    # переходим к следующему числу
    num += 1

# по итогу выполнения данного алгоритма
# в переменной bool_arr хранится список,
# элементы которого являются вердиктом о простоте
# своего номера

```

Временная сложность данного алгоритма составит $O(n \cdot \log(\log(n)))$. Для доказательства можно обратиться к формуле Мертенса (вторая теорема Мертенса). Что является значительно более быстрым вариантом на больших входных данных, нежели наша идея использовать перебор всех делителей для всех чисел.

Существуют более оптимизированные версии данных алгоритмов, с их реализацией и математической обоснованностью можно познакомиться в источниках, представленных в дополнительных материалах.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1.

Два простых числа, разница между которыми равна двум, называются близнецами. Напишите программу, которая бы проверяла, существует ли пара-близнец к данному введённому числу.

Ввод. Программа принимает на вход натуральное число n – число для проверки $1 \leq n \leq 10^5$.

Вывод. Программа должна вывести 0, если было введено не простое число, 1 – если введённое число простое и не имеет близнеца, m – наименьшее число близнец (если чисел близнецов более одного).

Задание 2.

Напишите программу, которая выводит все простые числа из заданного отрезка.

Ввод. Программа принимает на вход два числа a и b – границы отрезка $[a, b]$ $1 \leq a < b \leq 10^5$.

Вывод. Программа выводит через пробел все простые числа из отрезка $[a, b]$.

Задание 3.

Напишите программу, которая определяла бы, сколько пар чисел близнецов есть на отрезке $[a, b]$. Если среди двух чисел близнецов одно входит в отрезок, а второе – нет, то такая пара не засчитывается.

Ввод. Программа получает на вход два натуральных числа a и b $1 \leq a < b \leq 10^5$.

Вывод. Программа выводит одно число – количество пар чисел близнецов на заданном отрезке.

ДОПОЛНИТЕЛЬНЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

1. Вторая теорема Мертенса:
<https://mathworld.wolfram.com/MertensSecondTheorem.html>
2. Василенко О.Н. Теоретико-числовые алгоритмы в криптографии. — М.: МЦНМО, 2003. — 328 с. (Первая глава)

§8.3 Применение структур данных при решении задач

ЧИТАЕМ

Для развития навыков программирования вне зависимости от круга решения задач, который ставится целью для непосредственного изучения, очень важно формирование базовых знаний в сфере структуры данных и алгоритмов. Данные понятия являются фундаментом для творческого подхода к рассмотрению путей решения конкретной задачи.

В прошлых параграфах данной главы и главы, посвященной сложности алгоритмов, мы рассмотрели некоторые алгоритмы сортировки, нахождения наибольшего общего делителя и определения простоты. Описанные концепции являются очень важными для становления навыков решения различных классов задач с помощью компьютера, но, очевидно, не являются достаточными.

ВСПОМИНАЕМ

Для рассмотрения структур данных и алгоритмов знания об асимптотическом анализе являются необходимым условием. Поэтому, давайте вспомним, что такое асимптотическая сложность?

ЧИТАЕМ

Давайте формализуем понятие структуры данных. Под конкретной *структурой данных* чаще всего понимается некоторая система, позволяющая хранить набор однородных данных и обрабатывать их с помощью операций, подразумеваемых конкретной структурой данных (интерфейс). Однородность данных может быть определена как множество данных одного типа (например – целые числа) или более абстрактно – в виде всех данных схожей природы (например – любые числа из множеств \mathbb{N} , \mathbb{Z} , \mathbb{Q} , \mathbb{R} , \mathbb{C} и др.).

Сегодня мы познакомимся с простыми базовыми структурами данных – со стеком и с очередью.

Стек – это структура данных, основной идеей для которой является правило «*последним пришёл – первым ушёл*» (англ. last in – first out, LIFO). Если говорить более конкретно, то при использовании данной структуры данных, мы будем иметь доступ внутри программы только к последнему добавленному элементу. Основные операции, которые подразумевает стек для работы – это просмотр последнего добавленного элемента или его удаление, добавление нового элемента в стек.

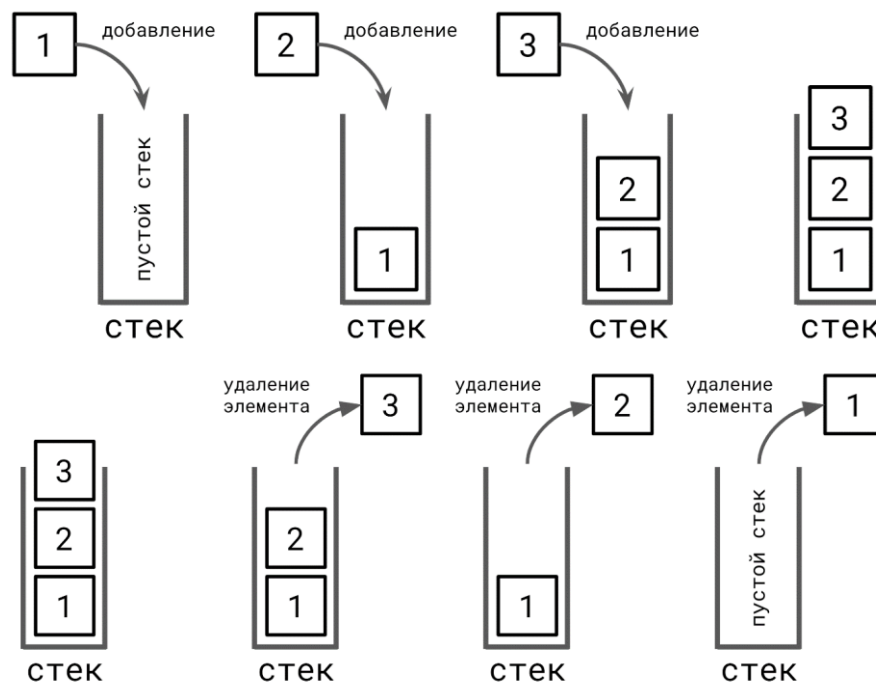


Рис. 171 Структура данных «Стек»

Подобная структура данных очень легко реализуется на языке программирования Python, для этого можно использовать в качестве основы список (list) и его методы.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Давайте приведём короткий бессодержательный пример на языке программирования Python, хорошо иллюстрирующий работу стека:

```
stack = list() # создаём стек на основе списка

# добавление элементов в стек можно реализовать
# с помощью метода списка append,
# который добавляет элемент в самый конец списка
stack.append(1) # добавляем в стек 1
stack.append(2) # добавляем в стек 2
stack.append(3) # добавляем в стек 3

# удаление элементов из стека можно реализовать
# с помощью метода pop, который при вызове без аргумента
# удаляет последний элемент из списка и возвращает его
print(stack.pop()) # забираем последний элемент и выводим его
print(stack.pop()) # забираем последний элемент и выводим его
print(stack.pop()) # забираем последний элемент и выводим его
# будет выведено в разных строках: 3, 2, 1
```

Рисунки, приведённые выше, хорошо иллюстрируют то, что выполняет данный алгоритм.

Со знанием асимптотической сложности $O(1)$ работы методов `append` и `pop`, мы можем предположить, что подобная реализация стека ничем не уступает более сложным реализациям в других языках (можете с ними ознакомиться при желании).

ЧИТАЕМ

При рассмотрении стека очень часто возникает вопрос – зачем нужна такая структура данных, в каких задачах её можно использовать?

Давайте рассмотрим задачу о правильной скобочной последовательности. В первую очередь необходимо пояснить, что скобочной последовательностью называют любую последовательность скобочных символов – это могут быть как круглые и квадратные скобки, так фигурные и треугольные (их наличие зависит от условий самой задачи).

Правильную скобочную последовательность чаще всего определяют индуктивно:

- любая пустая последовательность скобок (отсутствие скобок) есть правильная скобочная последовательность;
- если некоторая строка S – правильная скобочная последовательность, то и последовательность (S) так же является правильной;

• если две строки S_1 и S_2 – правильные скобочные последовательности, то их конкатенация S_1S_2 так же является правильной.

Приведём пример правильных скобочных последовательностей:

- $(())$;
- $(())(())$;
- $(((()))$.

В качестве скобочных последовательностей, не являющихся правильными, можно привести следующие примеры:

- $()$;
- $) ($;
- $(())(($.

Задача о скобочной последовательности формируется очень просто – пусть нам дана некоторая последовательность скобок, необходимо определить, является ли она правильной.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Существует несколько реализаций алгоритмов решения данной задачи. Они основаны на похожей идее, но при этом могут задействовать разные инструменты. Мы рассмотрим алгоритм с использованием стека как более удобный алгоритм для анализа последовательностей с разными видами скобок.

Идея алгоритма чрезвычайно проста: пройдем по всем элементам последовательности; если нам встретилась “открывающаяся” скобка – положим её в стек, если нам встретилась “закрывающаяся” скобка и в стеке в качестве последнего элемента лежит “открывающаяся” скобка, то мы удаляем её из стека, иначе скобочная последовательность не является правильной. Если после проверки последовательности стек оказался не пуст, значит, что мы не нашли для какого-то количества “открывающихся” скобок их пары – последовательность не является правильной.

Реализация на языке программирования Python может выглядеть следующим образом:

```
bracket_seq = input() # вводится скобочная последовательность

stack = list() # создаём стек для использования в алгоритме
is_correct = True # предполагаем, что последовательность
                    правильная

# проходим по всем скобкам последовательности
for bracket in bracket_seq:
    if bracket == '(':
```

```

        # если рассматриваемая скобка открывающаяся,
        # добавляем её в стек
        stack.append(bracket)
    elif bracket == ')' and len(stack) != 0:
        # если рассматриваемая скобка закрывающаяся,
        # и в стеке есть открывающиеся скобки,
        # то удаляем последнюю скобку из стека
        stack.pop()
    else:
        # если для закрывающейся скобки нет пары в стеке,
        # то последовательность не является правильной --
        # прекращаем цикл
        is_correct = False
        break

# если так вышло, что после прохождения алгоритма
# в стеке осталась хоть одна скобка --
# последовательность не является корректной
if len(stack) != 0:
    is_correct = False

if is_correct:
    print('ПРАВИЛЬНАЯ СКОБОЧНАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ')
else:
    print('НЕПРАВИЛЬНАЯ СКОБОЧНАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ')

```

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №72

Напишите программу, которая проверяла бы последовательность скобок разного вида на правильность. В качестве входных данных программа должна принимать строку, состоящую из скобок разного вида (это могут быть скобки '(', '[', '{', ')', ']', '}'). Ваша программа должна вывести «ПРАВИЛЬНАЯ СКОБОЧНАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ» в случае, если на вход была подана правильная скобочная последовательность, иначе – «НЕПРАВИЛЬНАЯ СКОБОЧНАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ».

Обратите внимание, что в данном случае пара “открывающаяся-закрывающаяся” скобки должны быть одного вида. То есть, например последовательность

[({})]

не является правильной в то время, как последовательность

(){}[]

является правильной скобочной последовательностью.

ЧИТАЕМ

Очередь – это структура данных, основной идеей для которой является правило «первым пришёл – первым ушёл» (англ. first in – first out, FIFO). Если говорить более конкретно, то при использовании данной структуры данных, мы будем иметь доступ внутри программы только к элементу, который находится впереди очереди. Основные операции, которые подразумевает стек для работы – это просмотр первого добавленного элемента или его удаление, добавление нового элемента в конец очереди.

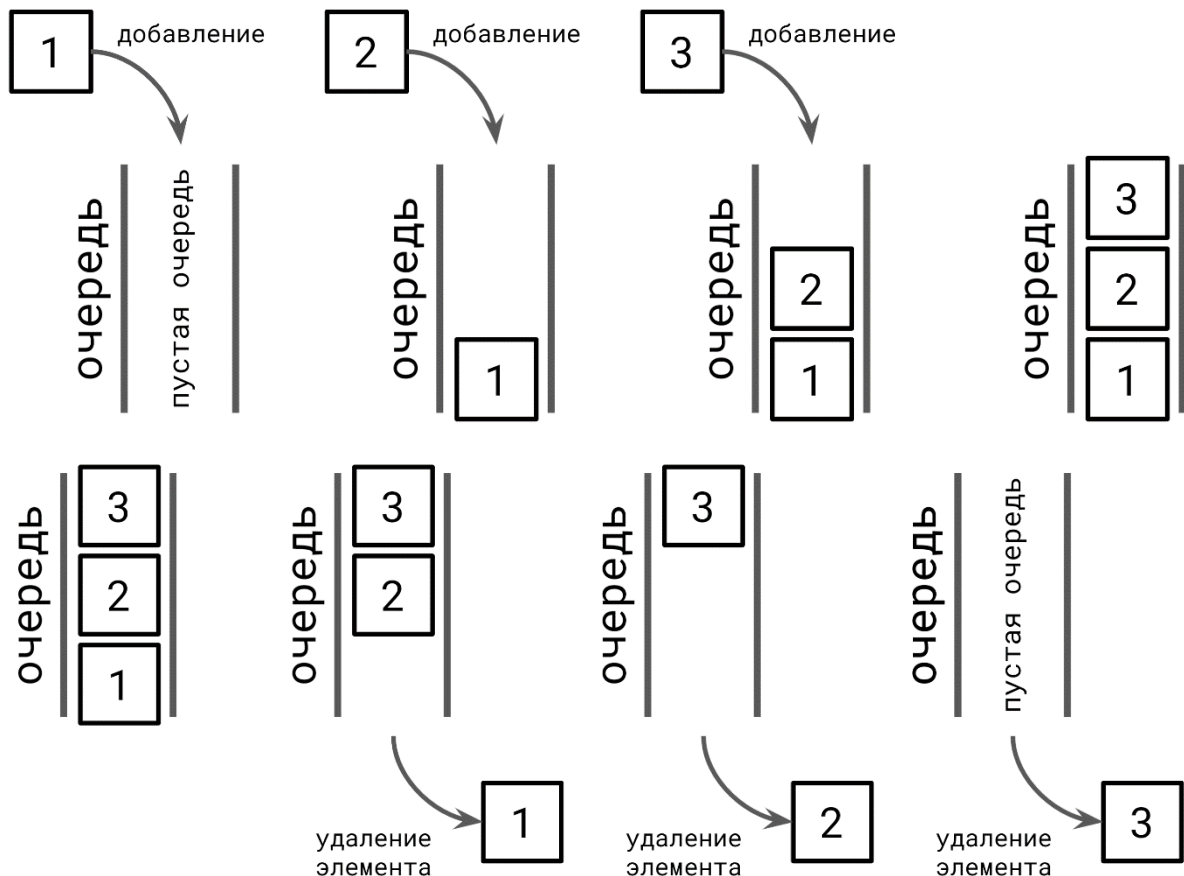


Рис. 172 Структура данных «Очередь»

Очередь, как и стек, очень легко реализуется на языке программирования Python с помощью списков (list) и его методов.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Давайте приведём короткий бессодержательный пример на языке программирования Python, хорошо иллюстрирующий работу стека:

```
queue = list() # создаём очередь на основе списка

# добавление элементов в очередь можно реализовать
# с помощью метода списка append,
# который добавляет элемент в самый конец очереди
queue.append(1) # добавляем в очередь 1
```

```

queue.append(2) # добавляем в очередь 2
queue.append(3) # добавляем в очередь 3

# удаление элементов из очереди можно реализовать
# с помощью метода pop, который при вызове с аргументом 0
# удаляет первый элемент из списка и возвращает его
print(queue.pop(0)) # забираем первый элемент и выводим его
print(queue.pop(0)) # забираем первый элемент и выводим его
print(queue.pop(0)) # забираем первый элемент и выводим его
# будет выведено в разных строках: 1, 2, 3

```

По своей сути данный код полностью предоставляет иллюстрацию работы очереди на рисунках выше.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Условие:

Необходимо написать программу, которая бы сначала считала бы все вводимые числа и поместила бы их в очередь, после чего вывела бы все числа из очереди до первого встреченного четного числа.

Рассуждение:

Будем считать, что в каждой вводимой последовательности всегда встречается чётное число.

Алгоритм решения данной задачи довольно прост:

```

queue = list() # создаём очередь на основе списка

n = int(input()) # вводим количество элементов
последовательности
# вводим все элементы и добавляем их в очередь
for _ in range(n):
    queue.append(int(input()))

# выводим и убираем из очереди первые нечетные элементы
while queue[0] % 2 != 0:
    print(queue.pop(0))

```

ЧИТАЕМ

Теперь у нас есть два очень важных инструмента, которые теперь можно использовать для решения большого круга задач. Мы с вами можем обсудить задачу вычисления значения арифметических выражений. Для этого сначала давайте рассмотрим некоторые базовые понятия.

Первое, что нам необходимо определить – это обратная польская нотация.

Обратной польской нотацией называется форма записи арифметических (и логических) выражений, в которой операнды (аргументы операции) расположены перед знаками операций. Подобная форма записи позволяет избавиться от скобок, объединяющих подвыражения и абстрагирующих приоритет выполнения операций.

Примером арифметического выражения, записанного в обратной польской нотации, может быть выражение:

$$a\ b\ +\ c\ * \iff (a\ +\ b)\ * \ c$$

В данном примере слева и справа показаны абсолютно идентичные арифметические выражения, где слева выражение записано в обратной польской нотации, а справа – в привычной нам (инфиксная нотация).

Обратная польская нотация удобна для представления в компьютере, так как для вычисления выражения в таком случае нам достаточно последовательно идти слева направо и применять арифметические операции к значениям, которые у нас уже условно учтены.

Давайте разберём алгоритм вычисления значения арифметического выражения, записанного в польской нотации. Для реализации данного алгоритма мы будем использовать стек:

- считываем очередной элемент последовательности:
 - если это операнд, записываем его в стек;
 - если это оператор, то достаём несколько операндов из стека (для унарных операций – один элемент, для бинарных – два, для тернарных – три и т.д.), вычисляем значение и кладем его в стек;
- возвращаемся к первому шагу, если прочитаны не все элементы последовательности.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

С точки зрения реализации данного алгоритма средствами языка программирования нет никаких сложностей:

```
# вводим последовательность, состоящую из операций и операндов,  
# описывающих арифметическое выражение в обратной польской  
нотации  
arithmetic_exp = input().split()  
  
stack = list()  
# проходим по элементам последовательности  
for elem in arithmetic_exp:  
    if elem.isnumeric():
```

```

        # если очередной элемент является операндом,
        # кладем его в стек
        stack.append(float(elem))
    else:
        # иначе кладем в стек результат вычисления значения
        операции,
        # с соответствующим ей количеством операндов из стека
        # ПРИМЕЧАНИЕ: в нашем случае рассматриваются четыре
        # базовые арифметические операции
        second_operand = stack.pop()
        first_operand = stack.pop()
        if elem == '-':
            stack.append(first_operand - second_operand)
        elif elem == '+':
            stack.append(first_operand + second_operand)
        elif elem == '/':
            stack.append(first_operand / second_operand)
        elif elem == '*':
            stack.append(first_operand * second_operand)

# результат вычисления всего выражения должен лежать в стеке,
# при условии корректности записи выражения,
print(stack.pop())

```

Можно более элегантно и подробно описать получение значения вычисления операции. Но при написании преследовалась цель наглядности. Аналогично можно описать любые операции, в том числе и логические и более абстрактные собственные, если решаемая задача того требует.

ЧИТАЕМ

Для вычисления значений с точки зрения компьютера обратная польская нотация безупречна, но человеку сложно воспринимать такую запись, поэтому в какой-то момент времени появилась необходимость придумать, как привычную нам запись арифметических выражений представить в виде обратной польской нотации.

Существует несколько алгоритмов для решения данной задачи, но наиболее простой и часто используемый сконструировал очень важный для всей отрасли информационных технологий учёный Эдсгер Вибе Дейкстра.

Данный алгоритм за счёт схожести с работой железнодорожной сортировочной станции получил название алгоритм сортировочной станции. Алгоритм использует в качестве инструментов для получения результата

(обратной польской нотации) как стек (стек операторов), так и очередь (очередь результатов).

Для арифметического выражения без использования скобок данный алгоритм можно записать в следующем виде:

- считываем очередной элемент последовательности:
 - если это операнд, записываем его в очередь;
 - если это оператор:
 - если стек операторов пуст, то оператор помещается в него;
 - если текущий оператор имеет более высокий приоритет, чем оператор на вершине стека, то текущий оператор помещается в стек;
 - если оператор имеет приоритет, равный приоритету оператора на вершине стека и оператор в стеке левоассоциативен, то происходит извлечение оператора из вершины стека (он отправляется в очередь результатов), а текущий оператор помещается в стек;
 - если оператор имеет приоритет ниже, чем у оператора на вершине стека, то верхний оператор извлекается из стека (отправляется в очередь результатов), после чего происходит повторное сравнение приоритета текущего оператора с вершиной стека;
- возвращаемся к первому шагу, если прочитаны не все элементы последовательности;
- когда все элементы входящей последовательности операторов и операндов проанализированы, все операторы из стека записываются в очередь.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

```
# вводим последовательность, состоящую из операций и операндов,
# описывающих арифметическое выражение в инфиксной нотации
arithmetic_exp = input().split()

# описываем приоритет операторов
operators = {
    '+': 1,
    '-': 2,    # считаем, что левоассоциативные операторы
               # приоритетнее
    '*': 3,
    '/': 4     # аналогично оператору "-"
}

stack = list() # стек для операторов
```



```

queue = list() # очередь для результатов

# проходим по элементам последовательности
for elem in arithmetic_exp:
    if elem.isnumeric():
        # если очередной элемент является операндом,
        # кладем его в очередь
        queue.append(int(elem))
    else:
        if len(stack) == 0:
            # если стек пуст, кладем в него операцию
            stack.append(elem)
        else:
            # иначе все операторы с большим или равным
приоритетом
            # (равным при левоассоциативности оператора в стеке)
            # переносим в очередь
            while True:
                if len(stack) == 0:
                    break
                elif operators[elem] <= operators[stack[-1]]:
                    queue.append(stack.pop())
                else:
                    break
            # и добавляем текущий оператор в стек
            stack.append(elem)

# переносим оставшиеся операторы из стека в очередь
# после анализа всех элементов последовательности
while len(stack) != 0:
    queue.append(stack.pop())

# последовательность элементов в очереди есть запись
# арифметического выражения в обратной польской нотации
print(queue)

# можно перевести в строку:
# str_exp = ' '.join([str(elem) for elem in queue])

```

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №71

Напишите программу, которая высчитывает значение выражения, записанного в привычной форме (в инфиксной нотации).

ЧИТАЕМ

В предыдущей главе мы рассматривали класс задач, основной спецификой которых является наличие алгоритма решения, который можно описать временной сложностью $T(n) \in O(n)$ с потреблением памяти $M(n) \in O(1)$, то есть линейного алгоритма, память для работы которого не зависит от размера входных данных.

В подобных задачах иногда можно использовать очередь. Давайте рассмотрим пример такой задачи.

Пусть каждую секунду программа принимает на вход новое число, не превышающее 1000, из списка, состоящего из n чисел. Необходимо найти минимальное четное произведение двух чисел, между моментами передачи которых прошло не менее 8 секунд.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Идея оптимального решения данной задачи довольно проста:

- 1) считываем первые 8 чисел;
- 2) убираем из очереди первое число и проверяем, является ли оно минимальным и минимальным четным;
- 3) считываем новое число;
- 4) если оно чётное, рассматриваем его произведение с минимальным значением вне рассматриваемого отрезка, иначе — произведение с минимальным чётным;
- 5) если данное произведение меньше минимального, то называем его минимальным;
- 6) пока программе на вход подаются новые значения, возвращаемся к пункту 2;
- 7) если получился нечетный результат, то выводим -1, иначе выводим результат.

Очевидно, что память, выделяемая для работы данного алгоритма, не изменяется от количества входных данных $M(n) \in O(1)$, а время зависит от входных данных линейно $T(n) \in O(n)$, так как мы только анализируем каждое новое поступающее число один раз.

Можно написать данный алгоритм на языке программирования Python следующим образом:

```
# считываем количество элементов последовательности
n = int(input())
```

```
queue = list() # очередь для хранения рассматриваемого отрезка
min_res = 10 ** 9 + 1 # ответ на задачу
min_even = 10 ** 9 + 1 # чётный минимум, не входящий в отрезок
```

```

min_other = 10 ** 9 + 1 # любой минимум, не входящий в отрезок

# считываем первые 8 чисел
for _ in range(8):
    queue.append(int(input()))

# анализируем оставшуюся последовательность
for _ in range(n - 8):

    # убираем самое "ранее" число
    temp = queue.pop(0)
    # смотрим, не является ли оно минимальным,
    # вышедшим из рассматриваемого отрезка
    if temp % 2 == 0:
        min_even = min(temp, min_even)
    min_other = min(temp, min_other)

    # считываем следующее
    next = int(input())
    pre_min: int
    # смотрим, не может ли оно дать нам
    # минимальное чётное в произведении с вышедшими
    if next % 2 == 0:
        pre_min = next * min_other
    else:
        pre_min = next * min_even

    min_res = min(pre_min, min_res)

    # кладем новое число в очередь
    queue.append(next)

# если чётного произведения так и не вышло, то выведем -1
if min_res % 2 == 0:
    print(min_res)
else:
    print(-1)

```

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1.

Написать программу с использованием стека, которая вводимые пользователем целые числа выводит в обратном порядке.

Задание 2.

Написать программу объединения двух очередей с равным количеством элементов; результирующая очередь должна содержать в себе чередующиеся элементы первой и второй очередей.

Задание 3.

Напишите программу определения правильности скобочной последовательности без использования стека, при условии, что в последовательности есть только круглые скобки.

Задание 4.

Попробуйте написать программу сортировочной станции для анализа арифметического выражения, в записи которого используются скобки и операция возведения в степень. Описание самого алгоритма можно найти в дополнительных источниках информации.

Задание 5.

Напишите программу, которая принимает на вход n чисел, не превышающих 1000. Необходимо найти минимальное нечетное произведение двух чисел, номера которых отличаются не менее, чем на 15

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Опишите особенности работы стека и очереди в сравнении.
2. Что такое правильная скобочная последовательность, обратная польская нотация?
3. Опишите алгоритм определения правильной скобочной последовательности.
4. В чём заключается идея, лежащая в основе алгоритма сортировочной станции? Что является результатом работы данного алгоритма?
5. Как посчитать результат арифметического выражения, записанного в обратной польской нотации?

§8.4 Рекурсивные алгоритмы, метод ветвей и границ

ВСПОМИНАЕМ

У вас наверняка уже есть представление о рекурсивных алгоритмах, так как мы затрагивали их ранее, давайте попробуем формализовать понятие рекурсивного алгоритма.

ЧИТАЕМ

При реализации тела функции мы можем вызывать другие функции для решения различных задач. Если внутри тела функции мы обращаемся к этой же функции, то подобный алгоритм называется рекурсивным.

В предыдущих уроках, посвящённых анализу сложности алгоритмов, мы уже обращались к рекурсии. Так, мы касались рекурсивных алгоритмов на примере задачи о Ханойской башне и задачи сортировки.

С помощью рекурсии можно решать и привычные для нас задачи. Существуют языки программирования (например, языки функционального программирования), в которых нет понятия цикла вообще. И в рамках таких ограничений циклический алгоритм заменяется рекурсивным.

Например, рекурсивный вывод первых n натуральных чисел может выглядеть следующим образом:

```
def func(n):  
    # пока передаваемый аргумент не равен 1  
    if n != 1:  
        # вызываем данную функцию от значения аргумента на 1  
        # меньше  
        func(n - 1)  
    # после чего выводим значение аргумента  
    print(n)
```

При рассмотрении рекурсии всегда стоит помнить о приоритете выполнения – если была вызвана какая-то функция, то она должна завершить свою работу, прежде чем интерпретатор будет выполнять следующие команды. Схематично описанный выше алгоритм можно изобразить следующим образом:

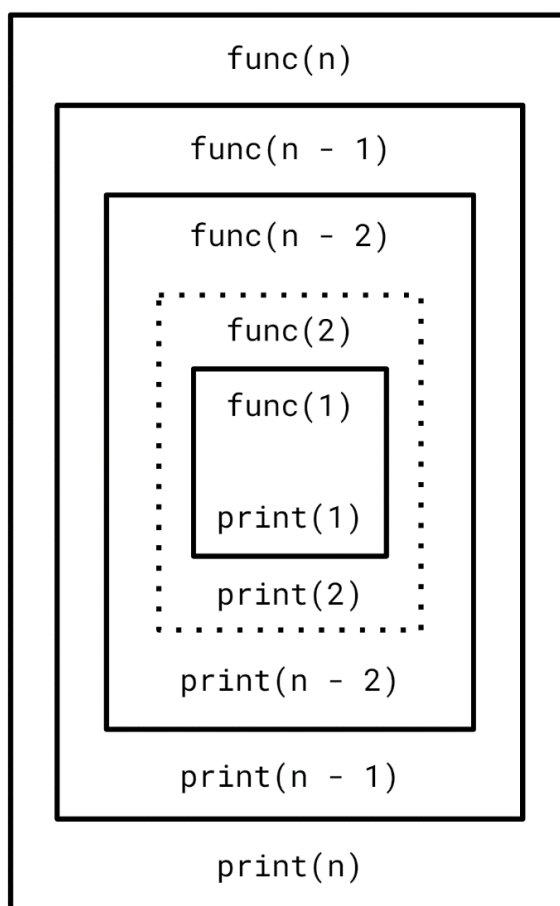


Рис. 173 Схема рекурсивной функции

То есть функция `func(n)` сначала вызовет функцию `func(n - 1)`, которая вызовет `func(n - 2)` и так будет продолжаться до тех пор, пока не будет вызвана функция `func(1)`, которая просто выведет 1, после чего произойдет возврат в функцию `func(2)`, у которой оставшееся действие, описанное в теле функции – вывести 2, после чего будет произведён возврат в тело функции `func(3)` и так будет продолжаться до тех пор, пока не произойдёт возвращение в функцию `func(n)` и не выведется число `n`. После чего функция завершит свою работу.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №72

Напишите программу, реализующую рекурсивный алгоритм, выводящий все числа от 1 до `n` в обратном порядке.

ЧИТАЕМ

Очень важно, чтобы рекурсивная функция при её конструировании была правильно описана – для этого очень важно продумать ограничение рекурсии. В приведённом выше примере за это отвечает условный оператор, который вызывает эту же функцию с уменьшенным значением аргумента только в том случае, если алгоритм ещё не дошёл до единицы. Изменение аргумента (аргументов) является единственно правильным вариантом при описании

вызова рекурсивной функции, если мы ожидаем, что функция когда-то завершит свою работу.

Так, например, при рассмотрении рекурсивного расчёта n-ого числа Фибоначчи, пользуются следующей формулой:

$$F(0) = 0, F(1) = 1, F(n) = F(n-1) + F(n-2)$$

что легко реализовать, например в следующем виде на языке программирования Python:

```
def fib(n):  
    if n == 0:  
        return 0  
    if n == 1:  
        return 1  
    return fib(n - 1) + fib(n - 2)
```

В данном случае мы с помощью условного оператора отдельно рассматриваем граничные значения аргументов для рекурсивной функции. Подобные значения называются базой рекурсии – аргументы функции, не требующие вложенного вызова рекурсивной функции.

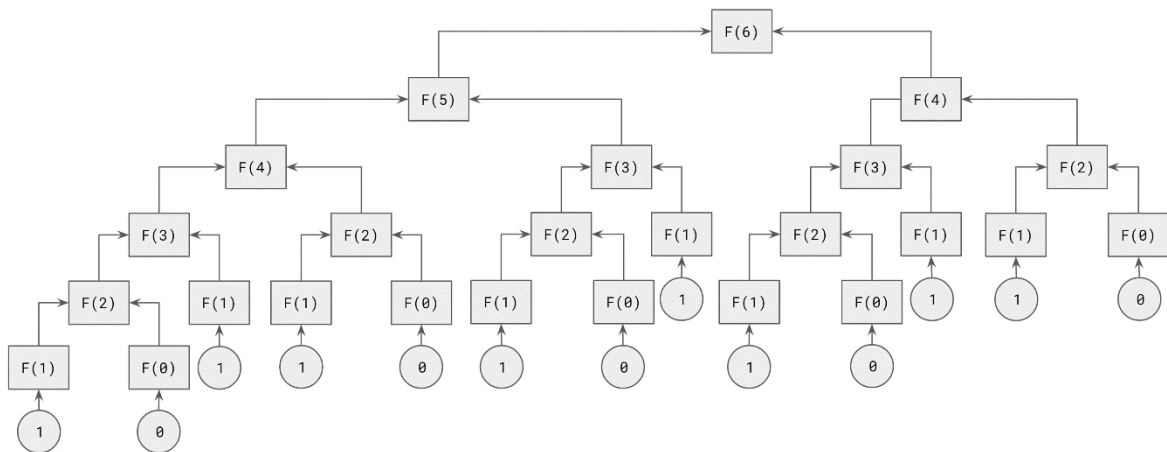


Рис. 174 База рекурсии

На рисунке выше представлены схема алгоритма вычисления шестого числа Фибоначчи. Кружочками на данной схеме показана база рекурсии. Как вы можете заметить – каждая ветвь дерева оканчивается каким-то элементом базы рекурсии.

Для проверки правильности написания рекурсивного алгоритма всегда можно представить дерево, на котором необходимо проследить, что каждая ветвь, порождаемая рекурсивным вызовом, имеет конец в базе рекурсии.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Существуют интуитивно понятные алгоритмы, которые вытекают из прямого определения термина. Наиболее ярким примером может быть степень числа:

$$a^n = 1 * \underbrace{a * \dots * a}_n$$

Мы вполне можем написать рекурсивный алгоритм для вычисления степени:

```
def power(number, p):
    if p == 0:
        return 1
    return number * power(number, p - 1)
```

Или циклический алгоритм:

```
number: int # предполагаем, что нам известны основание
p: int      # и показатель степени

res = 1
for _ in range(p):
    res *= number
# res является результатом вычисления степени числа
```

Но оба данных алгоритма зависят линейно от показателя степени. Алгоритм возведения в степень можно ускорить, сделав его логарифмическим.

Подобное упрощение по времени достигается всё так же из свойств степени, которые могут быть выражены в следующем виде:

$$a^n = \begin{cases} a^{n/2} * a^{n/2}, & n \div 2, \\ a^{n-1} * a, & n \not\div 2 \end{cases}$$

Данные измышления позволят сформировать бинарный алгоритм возведения в степень, который можно записать в следующем виде.

За базу рекурсии в данном алгоритме, как и в простом рекурсивном алгоритме возведения в степень, возьмём нуль, так как любое число в нулевой степени всегда равно 1. Вложенные вызовы функции будут осуществляться в зависимости от чётности n. На Python подобную рекурсивную функцию можно записать следующим образом:

```
def binary_pow(number, power):
    if power == 0:
        return 1
    if power % 2 == 1:
        return binary_pow(number, power - 1) * number
    temp = binary_pow(number, power // 2)
    return temp * temp
```


Зная такой подход и ещё одну важную теорему, которая будет сейчас введена, мы можем сильно упростить алгоритм вычисления n -ого числа Фибоначчи.

Для этого утвердим, что

$$(F_{n-2}, F_{n-1}) * \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} = (F_{n-1}, F_n)$$

после чего можно вывести следствие

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n = \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix}$$

Рекурсивный алгоритм вычисления чисел Фибоначчи, рассмотренный нами ранее имеет экспоненциальную сложность $T(n) = O(2^n)$. Можно привести линейный алгоритм вычисления n -ого числа Фибоначчи:

```
n: int # предполагаем, что n нам дано
fib_this = 0
fib_next = 1

for _ in range(n):
    fib_this, fib_next = fib_next, fib_this + fib_next

print(fib_this)
```

Но приведённая идея в совокупности с идеей бинарного возведения в степень позволяет нам находить n -ое число Фибоначчи за логарифмическое время $T(n) = O(\log(n))$.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №73

Напишите программу, которая вычисляет n -ое число Фибоначчи за логарифмическое время.

Напомним, что:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} * \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} a * e + b * g & a * f + b * h \\ c * e + d * g & c * f + d * h \end{pmatrix}$$

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1.

Напишите программу, реализующую рекурсивный алгоритм, выводящий все цифры натурального числа начиная с нулевого разряда.

Задание 2.

Напишите программу, реализующую вычисление значения функции Аккермана, если известно, что она задаётся:

$$A(m, n) = \begin{cases} n + 1, & m = 0 \\ A(m - 1, 1), & m > 0, n = 0 \\ A(m - 1, A(m, n - 1)), & m > 0, n > 0 \end{cases}$$

Задание 3.

Напишите программу, реализующую рекурсивный алгоритм вычисления факториала числа n .

Задание 4.

Напишите программу, вычисляющую сумму первых n элементов геометрической прогрессии, при известных b_1 , q – первом элементе последовательности и множителе. Напомним, что сумма элементов геометрической прогрессии определяется формулой:

$$S(n) = \begin{cases} \frac{b_1 * (1 - q_1^n)}{1 - q_1}, & q_1 \neq 1 \\ n * b_1, & q_1 = 1 \end{cases}$$

При решении данной задачи необходимо использовать бинарное возведение в степень.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Что такое рекурсивный алгоритм?
2. Приведите примеры рекурсивных алгоритмов, рассмотренных в рамках данного параграфа.
3. Сформулируйте основную идею быстрого возведения в степень.

§8.5 Алгоритмы на графах

ВСПОМИНАЕМ

Вспомним, что вы знаете о графах. Что такое граф? Какие виды графов Вам известны?

ЧИТАЕМ

Под графом в математике понимают множество объектов некоторого класса, называемые вершинами, и множество связей между данными объектами и обычно обозначают $\Gamma = (V, E)$, где V – это множество вершин, E – множество связей между вершинами (пары вершин).

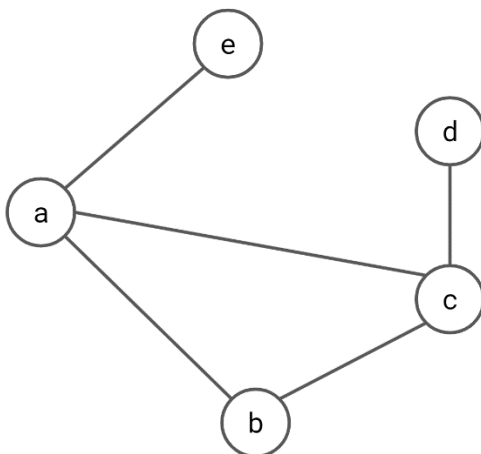


Рис. 175 Граф

На данном рисунке изображено графическое представление графа Γ , который аналитически записывается следующим образом:

$\Gamma = (V, E)$, где

- $V = \{a, b, c, d, e\}$;
- $E = \{(a, b), (a, c), (a, e), (b, c), (c, d)\}$.

Стоит заметить, что в данном случае представлен неориентированный граф. Неориентированный граф подразумевает под собой отсутствие направления (иерархии) между связанными объектами. Хорошим примером является граф, образованный городами (множество вершин) и дорогами, каждая из которых может соединять только два города (множество связей). Данный граф является неориентированным, в случае если каждая дорога позволяет двигаться в обе стороны – является двусторонней.

Ориентированный граф, как вы можете верно предположить, уже подразумевает наличие направления (иерархии) между двумя связанными объектами.

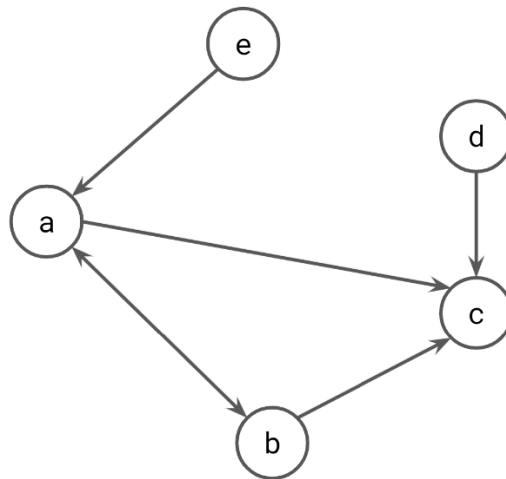


Рис. 176 Направленный граф

$\Gamma = (V, E)$ – ориентированный граф, где

- $V = \{a, b, c, d, e\}$;
- $E = \{(a, b), (b, a), (a, c), (e, a), (a, e), (b, c), (d, c)\}$.

Обратите внимание, что если в ориентированном графе задано направление из вершины a в вершину b и из вершины b в вершину a , то в описании связей необходимо указать обе данные связи.

Степень вершины – это количество связей, в которое конкретная вершина входит. Разделяют входящие и исходящие степени вершины для ориентированных графов, где под входящей степенью вершины называют количество связей, где данная вершина считается началом направления, под исходящей, соответственно, – количество связей, где вершина считается концом направления. В неориентированном графе данные значения совпадают, поэтому чаще всего обходятся только словом «степень» без уточнения, является ли она входящей или исходящей.

Взвешенный граф – это граф, ребрам которого присвоены некоторые «веса». Под весом обычно подразумевается число. В нашей метафоре, представляющей граф как множества городов и дорог, веса можно интерпретировать как длину дороги, связывающей два города.

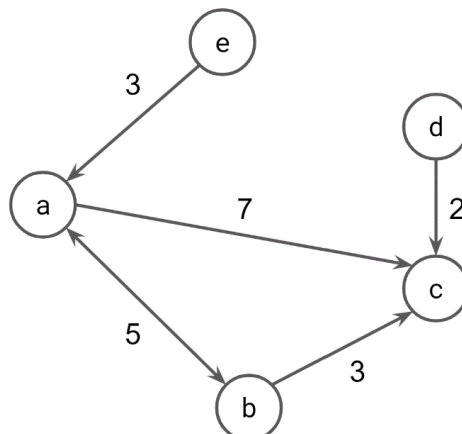


Рис. 177 Взвешанный граф

Путь в графе – это последовательность вершин, где каждые две соседние вершины имеют путь (связь) из первой вершины во вторую.

Пример пути в графе:

$a \rightarrow b \rightarrow c$

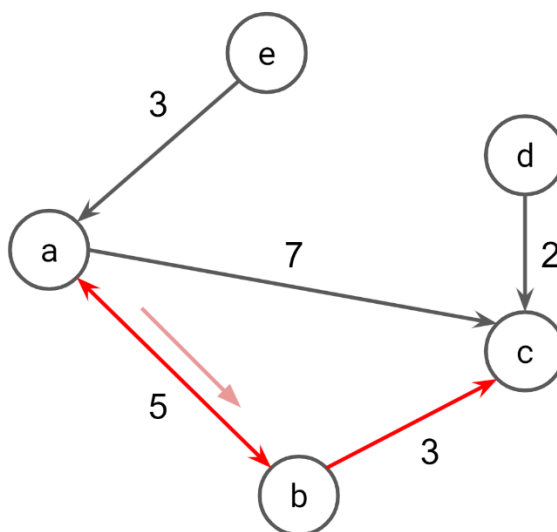


Рис. 178 Пример пути в графе

РЕШАЕМ В ТЕТРАДИ

Задание 65.

Опишите аналитически (опишите множества вершин и множество рёбер) графы Γ_1 и Γ_2 , представленные графически:

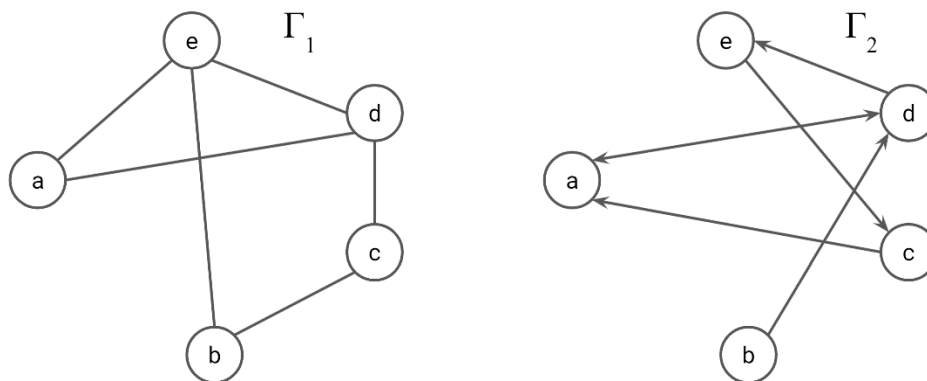


Рис. 179 Графы для задания

РЕШАЕМ В ТЕТРАДИ

Задание 66.

Изобразите графы, заданные аналитически:

а. $\Gamma = (V, E)$ – неориентированный граф, где

$V = \{a, b, c, d\}$ и

$E = \{(a, b), (b, c), (a, d), (c, d)\}$.

б. $\Gamma = (V, E)$ – ориентированный граф, где

$V = \{a, b, c, d, e, f\}$ и

$E = \{(a, d), (d, e), (e, d), (a, f), (f, c), (c, f), (c, b), (b, a)\}$.

ЧИТАЕМ

При решении задач с использованием компьютера понятие графа часто используется в качестве инструмента. И тут встает вопрос о том, как граф представить в памяти компьютера?

Существует два способа представления графа в памяти компьютера:

- 1) матрица смежности;
- 2) список смежности.

Матрица смежности для графа $\Gamma = (V, E)$ это матрица размером $|V| * |V|$, где наличие ненулевого значения в ячейке (i, j) подразумевает наличие ребра между вершинами i и j . Данное значение может описывать вес ребра у взвешенного графа. Таким образом можно сказать, что асимптотическая оценка занимаемой памяти в таком случае является квадратичной $M(|V|, |E|) \in O(|V|^2)$.

| | a | b | c | d | e |
|---|---|---|---|---|---|
| a | | 5 | 7 | | |
| b | 5 | | 3 | | |
| c | | | | | |
| d | | | 2 | | |
| e | 3 | | | | |

| | a | b | c | d | e |
|---|---|---|---|---|---|
| a | | 1 | 1 | | 1 |
| b | 1 | | 1 | | |
| c | 1 | 1 | | 1 | |
| d | | | 1 | | |
| e | 1 | | | | |

Данные таблицы изображают ориентированный взвешенный граф (а) и невзвешенный неориентированный граф (б). Оба были изображены в данном параграфе ранее. Если матрица описывает ориентированный граф, то считается, что номер строки является приоритетным – началом ребра.

Список смежности для графа $\Gamma = (V, E)$ по факту это список списков, где вложенный список вершины является описанием, с какими вершинами соединена данная.

| | | |
|---|-----|-----|
| a | b 5 | c 7 |
| b | a 5 | c 3 |
| c | | |
| d | c 2 | |
| e | a 3 | |

| | | | |
|---|---|---|---|
| a | b | c | e |
| b | a | c | |
| c | a | b | e |
| d | c | | |
| e | a | | |

Данные списки изображают ориентированный взвешенный граф (а) и невзвешенный неориентированный граф (б). Оба были изображены в данном параграфе ранее. При описании взвешенного графа необходимо записать вес ребра, при описании невзвешенного графа достаточно указать лишь вершину.

Матрицы смежности чаще всего используются при анализе графов, в которых количество рёбер $|E|$ близко по значению к количеству вершин в квадрате — $|V|*|V|$. Если количество рёбер значительно меньше количества вершин в квадрате, то лучше использовать список смежности.

РЕШАЕМ В ТЕТРАДИ

Задание 67.

Опишите с помощью матриц смежности и списков смежности графы, заданные аналитически:

- а. $\Gamma = (V, E)$ – неориентированный граф, где
 $V = \{a, b, c, d\}$ и
 $E = \{(a, b), (b, c), (a, d), (c, d)\}$.
- б. $\Gamma = (V, E)$ – ориентированный граф, где
 $V = \{a, b, c, d, e, f\}$ и
 $E = \{(a, d), (d, e), (e, d), (a, f), (f, c), (c, f), (c, b), (b, a)\}$.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Наиболее часто вам придется пользоваться списком смежности при решении различных задач. Давайте попробуем описать его создание с помощью языка программирования Python.

Пусть у нас есть невзвешенный неориентированный граф из n вершин и m рёбер. Программа принимает на вход m пар чисел, описывающих рёбра данного графа.

```
# вводится количество вершин и рёбер
n, m = tuple(map(int, input().split()))

# создаем список из n списков
graph = list()
for _ in range(n):
    graph.append(list())

# вводим m пар, описывающих рёбра
for _ in range(m):
    first, second = tuple(map(int, input().split()))
    # с учётом того, что граф неориентированный,
    # если существует ребро (a, b), то существует и (b, a)
    graph[first].append(second)
    graph[second].append(first)
```

Обратите внимание, что в данном алгоритме вершины нумеруются с нуля – вам могут попасться задачи, в которых вершины будут нумероваться с единицы.

Давайте рассмотрим ещё одну задачу. Пусть у нас есть взвешенный ориентированный граф из n вершин и m рёбер. Программа принимает на вход m троек чисел, где первые два числа (a , b) описывают ребра, а третье число – его вес.

```
# вводится количество вершин и рёбер
n, m = tuple(map(int, input().split()))

# создаем список из n списков
graph = list()
for _ in range(n):
    graph.append(list())

# вводим m троек, описывающих рёбра
for _ in range(m):
    first, second, w = tuple(map(int, input().split()))
    # с учётом того, что граф ориентированный,
    # мы добавляем связь только для первой вершины из двух,
    # при этом в качестве значения мы добавляем кортеж
    # из вершины и веса
    graph[first].append((second, w))
```

При программном описании графа можно пользоваться не только списками, но и другими структурами данных (коллекциями). Иногда для целей задачи может подойти словарь. Идея описания точно такая же.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №74

Напишите программу для описания неориентированного невзвешенного графа, состоящего из n вершин и m рёбер с использованием словаря.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №75

Напишите программу для описания ориентированного взвешенного графа, состоящего из n вершин и m рёбер с использованием словаря. При этом значением по ключу (номер вершины) должен быть новый словарь с ключом (второй вершины) и значением веса ребра.

ЧИТАЕМ

Мы научились представлять граф в памяти компьютера. Теперь нам нужны эффективные методы для работы с графом. Так, существует класс задач, в рамках решения которых необходимо пройти по всем вершинам графа – обойти граф. Обход графа – это поэтапное исследование каждой вершины графа.

Для обхода графа существует два базовых алгоритма – поиск в ширину (англ. breadth first search, BFS) и поиск в глубину (англ. depth first search, DFS) – это первые алгоритмы на графах, с которыми мы познакомимся.

Поиск в ширину отлично иллюстрирует свою идею своим названием – рассмотрение графа последовательно по уровням удалённости от стартовой вершины с целью поиска конкретной – перед тем как проанализировать вершины с удалённостью от стартовой, равной d , алгоритм проанализирует вершины с удалённостью, равной $(d - 1)$. В базовом варианте данный алгоритм проверяет, можно ли прийти из вершины a в вершину b . Данный алгоритм можно описать следующим образом:

1. выбирается стартовая вершина (её выбор зависит от целей задачи), она помещается в очередь;
2. из очереди убирается первая вершина и помечается как просмотренная:
 - a. если данная вершина является искомой, то алгоритм завершает свою работу;
 - b. иначе в очередь добавляются все непомяченные вершины, в которые можно прийти из той, что была удалена;
3. если очередь оказалась пуста, значит все возможные вершины были посещены, и искомая вершина не была найдена – окончание алгоритма;
4. перейти к пункту 2 данного алгоритма.

При реализации данного алгоритма можно не рассматривать момент определения искомой вершины с завершением работы алгоритма. Часто бывает достаточно проверить, является ли искомая вершина помеченной после выполнения алгоритма.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Поиск в ширину можно написать как в виде рекурсивного алгоритма, так и в классическом виде. Ниже будет представлен вариант алгоритма поиска в ширину в не рекурсивном виде на языке программирования Python:

```
n: int # количество вершин графа
m: int # количество рёбер графа
graph: list # пусть граф описан списком смежности

# предположим, что стартовой вершиной является 0
```

```

start_vertex = 0

# создаем список посещённых вершин
is_visited = [False] * n
queue = list() # очередь для работы BFS

# добавляем в очередь стартовую вершину
queue.append(start_vertex)
is_visited[start_vertex] = True # помечаем её

# пока очередь не пуста
while len(queue) != 0:
    # достаём из неё очередную вершину
    this_vertex = queue.pop(0)
    # и добавляем все её соседние непомяченные вершины в очередь
    for neighbour_vertex in graph[this_vertex]:
        if not is_visited[neighbour_vertex]:
            is_visited[neighbour_vertex] = True
            queue.append(neighbour_vertex)

# если i-ая вершина оказалась не посещённой после выполнения
# данного алгоритма -- значит в неё нельзя попасть
# из стартовой вершины

```

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №76

Напишите программу, которая анализирует неориентированный граф с n вершинами и m ребрами, и выносит вердикт – является ли он связным. При решении задачи необходимо использовать поиск в ширину.

Напомним, что граф называется связным, если для любой пары вершин существует путь, соединяющий их.

ЧИТАЕМ

При поиске в глубину, как следует из названия алгоритма, происходит движение по графу от вершины к вершине до тех пор, пока это возможно. Чуть более формально данный алгоритм можно описать следующим образом:

1. выбирается стартовая вершина (её выбор зависит от целей задачи), она помещается в стек;
2. из стека убирается “верхняя” вершина и помечается как просмотренная:

- а. если данная вершина является искомой, то алгоритм завершает свою работу;
 - б. иначе в стек добавляются все непомяченные вершины, в которые можно прийти из той, что была удалена;
3. если стек оказался пуст, значит все возможные вершины были посещены, и искомая вершина не была найдена – окончание алгоритма;
4. перейти к пункту 2 данного алгоритма.

Как и в случае с поиском в ширину, в данном алгоритме пункт 2.а, как и соответствующее заключение в пункте 3, могут не рассматриваться при реализации алгоритма в зависимости от поставленной задачи.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Поиск в глубину, как и поиск в ширину, можно написать как в виде рекурсивного алгоритма, так и в классическом виде. Ниже будет представлен вариант алгоритма поиска в глубину в не рекурсивном виде на языке программирования Python:

```
n: int # количество вершин графа
m: int # количество рёбер графа
graph: list # пусть граф описан списком смежности

# предположим, что стартовой вершиной является 0
start_vertex = 0

# создаем список посещённых вершин
is_visited = [False] * n
stack = list() # стек для работы DFS

# создаем список посещённых вершин
is_visited = [False] * n
stack = list() # очередь для работы DFS

# пока стек не пуст
while len(stack) != 0:
    # достаем из него очередную вершину
    this_vertex = stack.pop()
    # и добавляем все соседние непомяченные вершины в очередь
    for neighbour_vertex in graph[this_vertex]:
        if not is_visited[neighbour_vertex]:
            is_visited[neighbour_vertex] = True
            stack.append(neighbour_vertex)

# если i-ая вершина оказалась непосещённой после выполнения
```

```
# данного алгоритма -- значит в неё нельзя попасть
# из стартовой вершины
```

Как вы видите, алгоритм поиска в глубину отличается от поиска в ширину только заменой очереди на стек. Именно природа работы стека и очереди отличает данные алгоритмы.

ОБСУЖДАЕМ

Как влияет замена стека на очередь? Сказывается ли данная замена на функциональности данных алгоритмов? Какие задачи каким алгоритмом удобнее решать?

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1.

Напишите рекурсивные варианты алгоритмов поиска в глубину и поиска в ширину.

Задание 2.

Напишите программу, которая на основе анализа неориентированного графа с n вершинами и m ребрами отвечает на вопрос – какова наименьшая длина пути от вершины 0 до вершины k . При решении задачи необходимо использовать поиск в ширину.

Задание 3.

Напишите программу, которая на основе анализа неориентированного графа с n вершинами и m ребрами выводит путь от вершины 0 до вершины k . При решении задачи необходимо использовать поиск в глубину.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Какие множества определяются при аналитическом задании графа?
2. Что такое степень вершины (исходящая и входящая)? Как отличаются входящие и исходящие степени вершин в ориентированном и неориентированном графе?
3. Что такое вес ребра графа и путь в графе?
4. Какие способы представления графа в памяти компьютера вы знаете?
5. Что такое обход графа? Какие алгоритмы обхода графа вам известны?

§8.6 Жадные алгоритмы

ВСПОМИНАЕМ

- Давайте вспомним,
- что такое граф?

- в чём разница между ориентированным и неориентированным графом? между взвешенным и невзвешенным?
- что такое алгоритм?
- какие алгоритмы на графах вы уже знаете?

ЧИТАЕМ

Жадным алгоритмом называют алгоритм, который допускает, что при условии принятия оптимального решения при рассмотрении любой подзадачи конечное решение общей задачи так же будет оптимальным.

В общем случае большинство задач нельзя оптимально решить с помощью жадного алгоритма – более подробно мы поговорим об этом при рассмотрении динамического программирования. Важным для применимости жадного алгоритма является принцип жадного выбора.

Принцип жадного выбора применяется к решению задачи в том случае, если принятие оптимального решения подзадачи действительно ведёт к оптимальному решению самой задачи.

Для применимости принципа жадного выбора обычно приводится индуктивное доказательство:

- 1) применение жадного выбора при первом шаге решения (при решении первой подзадачи) не исключает оптимального решения всей задачи;
- 2) порождаемая подзадача решением предыдущей подзадачи полностью ей аналогична.

Давайте рассмотрим пример задачи, которую можно решить с помощью жадного алгоритма.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Пусть у нас есть неограниченное количество монет номиналом 25 у.е., 10 у.е., 5 у.е., и 1 у.е. Необходимо выдать сумму в K у.е. минимально возможным количеством монет.

Утверждается, что для решения данной задачи применим жадный алгоритм. То есть, для выдачи минимально возможного количества монет, мы будем давать монеты с наибольшим номиналом, какие можем.

```
k = int(input()) # вводим количество у.е. для выдачи
```

```
# создаем список с характеристикой монет
```

```
nominal_values = [25, 10, 5, 1]
```

```
# список результата --
```

```
# какая монета сколько раз входит в результат
```

```
res = list()
```

```
for nom in nominal_values:
    # берём монету максимального возможного размера в у.е.
    # максимально возможное количество раз
    res.append((nom, k // nom))
    k %= nom

print(res) # в res записан результат
```

ОБСУЖДАЕМ

1. Действительно ли к разобранной задаче можно применять жадный алгоритм? Обоснуйте возможность использования принципа жадного выбора при решении данной задачи.
2. Можно ли применить жадный алгоритм к аналогичной задаче, если у нас есть только монеты с номиналом 7 у.е., 5 у.е. и 1 у.е.

ЧИТАЕМ

С реализацией жадного алгоритма вы уже встречались ранее при рассмотрении неравномерного кодирования – построение оптимального кода с минимальной избыточностью по алгоритму Хаффмана является примером жадного алгоритма. Данный алгоритм сводится к построению бинарного дерева, что можно описать следующим образом:

- 1) выполним частотный анализ последовательности символом, результатом чего становится список символов с соответствующими им частотами, которые могут быть представлены в процентном или в количественном отношении;
 - 2) будем воспринимать каждый элемент списка, описанного в шаге (1), как дерево, состоящее из одной вершины;
 - 3) выберем из списка две вершины с минимальным весом – создадим для них вершину, соединенную с ним (родителя) с весом, равным их суммарному весу; левой
 - 4) полученная в пункте (3) вершина добавляется в список вместо двух вершин, связанных с ней;
 - 5) если в списке больше одной вершины, возвращаемся к пункту (3).
- Для более детального рассмотрения темы можно вернуться к параграфу 6.6.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №77

Напишите программу, которая при анализе текста, написанного русскими буквами и знаками пунктуации, строит неравномерный код согласно

алгоритму Хаффмана для всех встреченных символов. Программа не должна считать заглавные и строчные буквы разными символами.

ЧИТАЕМ

Идея жадного алгоритма применяется и некоторых важных алгоритмах на графах. Так идея жадности отражена в алгоритме Прима и в алгоритме Краскала.

Алгоритм Краскала нами будет рассмотрен в следующих параграфах, в рамках данного параграфа мы коснёмся алгоритма Прима. Для этого нам понадобится ввести некоторые понятия.

Остовным деревом (или *остовом*) графа мы будем называть дерево, являющееся связным подграфом данного графа, включающего в себя все вершины исходного графа.

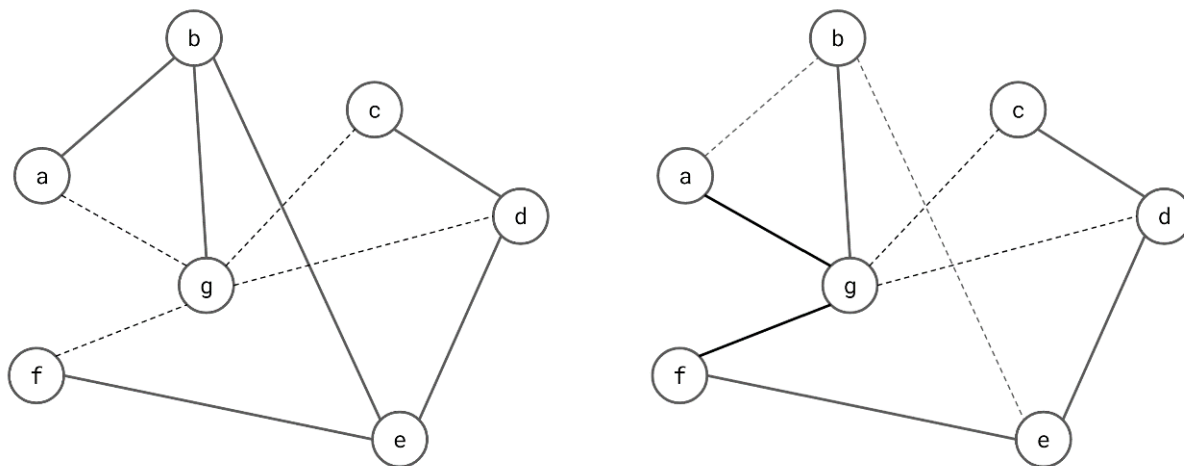


Рис. 180 Основное дерево графа

На рисунках выше изображены два односвязных неориентированных графа. Пунктирными линиями изображены рёбра, при извлечении которых будут получены остовные деревья – графы, состоящие из всех вершин и рёбер показанных в виде сплошных линий.

Как вы видите, различные остовные деревья могут соответствовать одному и тому же графу.

В случае рассмотрения взвешенных неориентированных графов часто возникает задача нахождения минимального остовного дерева.

Минимальным остовным деревом в взвешенном неориентированном графе называют остовное дерево с минимальным весом – с минимальной суммой весов рёбер, входящих в остов.

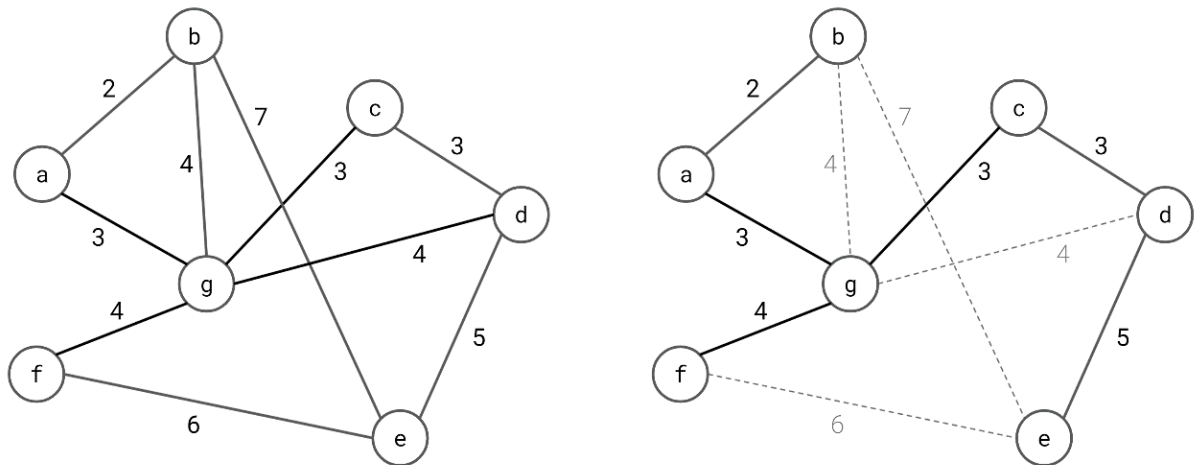


Рис. 181 Минимальное остовное древо

Так, на рисунке выше изображён граф и его минимальное остовное древо. Минимальность данного остова мы можем доказать или получить с помощью алгоритма перебора. Но перебор является почти всегда плохим решением задачи – так и в этом случае.

Алгоритм Прима – алгоритм построения минимального остовного дерева для взвешенного неориентированного графа. Данный алгоритм неформально можно описать следующим образом:

- 1) выбирается произвольная вершина;
- 2) для выбранной вершины выбирается ребро с минимальным весом, в которое входит данная вершина – вершины, включённые в выбранное ребро, образуют древо;
- 3) пока существуют невыбранные вершины, к текущему древу добавляется ребро с наименьшим весом, при условии, что такое добавление не создаёт цикл внутри конструируемого графа (конструируемый граф остаётся деревом).

Полученный таким образом граф является минимальным остовом исходного графа. Как видно из описания алгоритма – в нём действительно используется идея жадности – на каждом шагу добавляется минимально возможное ребро.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Давайте напишем реализацию алгоритма Прима на языке программирования Python.

Будем считать, что граф описывается следующим образом:

```
import operator # используется для нахождения минимального
                 # значения в словаре

# вводится количество вершин и рёбер
n, m = tuple(map(int, input().split()))
```



```

# создаем словарь из n словарей
# в качестве значения по ключам [i][j] будет лежать вес ребра
# между вершинами i и j
graph = dict()
for i in range(n):
    graph[i] = dict()

# вводим m троек, описывающих рёбра
for _ in range(m):
    first, second, w = tuple(map(int, input().split()))
    # с учётом того, что граф неориентированный,
    # если существует ребро (a, b), то существует и (b, a)
    graph[first][second] = w
    graph[second][first] = w

Тогда сам алгоритм Прима может выглядеть следующим образом:
# предположим, что стартовой вершиной является 0
start_vertex = 0

# пары, описывающие вершины, которые можно присоединить к дереву
# в виде
# номера вершины и веса присоединяемого ребра
for_visit = {start_vertex: 0}

# словарь вершин, из которых возможно провести ребро
leaves = {start_vertex: start_vertex}

# минимальное остовное дерево
min_span_tree = dict()
while len(for_visit) != 0:
    # выбираем вершину, которая при присоединении к дереву
    # добавляет минимальное по весу ребро
    vertex = min(for_visit.items(),
key=operator.itemgetter(1))[0]

    # добавляем данное ребро в конструируемое оставное дерево
    min_span_tree[vertex] = leaves[vertex]

    # отмечаем добавленное ребро
    del for_visit[vertex] # больше добавленную вершину не
                           # рассматриваем в качестве добавления
    del leaves[vertex]    # и не рассматриваем вершину

```

```

# из которой только что было
проведено ребро

# добавляем в качестве кандидатов на добавление возможных
соседей
# только что добавленной вершины
for neighbor_vertex in graph[vertex].keys():
    if neighbor_vertex not in min_span_tree:
        # возможной соседней вершиной не может являться
        # уже добавленная в остов вершина
        if neighbor_vertex not in for_visit:
            # если вершина ещё не рассматривалась для
добавления --
            # добавим её
            for_visit[neighbor_vertex] =
graph[vertex][neighbor_vertex]
            leaves[neighbor_vertex] = vertex
            elif graph[vertex][neighbor_vertex] <
for_visit[neighbor_vertex]:
            # если в соседнюю вершину
            # можно попасть из разных вершин минимального
остова,
            # выбираем ребро с минимальным весом
            for_visit[neighbor_vertex] =
graph[vertex][neighbor_vertex]
            leaves[neighbor_vertex] = vertex

# получаем словарь, в котором в качестве пары ключ-значение
# представлены вершины, образующие ребро минимального остова
print(min_span_tree)

```

Обратите внимание, что в словаре `min_span_tree` будет находиться пара ключ-значение “0: 0”. При анализе минимального остова это придётся учесть – например, можно удалить данную пару из словаря после завершения алгоритма. Временная оценка данного алгоритма $T(|V|, |E|) \in O(|V|^2)$.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Напишите программу для решения следующей задачи. Пусть у нас есть n городов и m дорог, связывающих пары городов. Каждая дорога описывается тремя целыми числами – `first_town`, `second_town` и `distance` – номера первого и второго городов (`first_town` и `second_town`), расстояние между которыми равно `distance`. Необходимо определить, какое минимальное количество километров

достаточно покрыть асфальтом, чтобы автомобилист из любого города смог доехать до любого другого города.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. В чём заключается основная сущность жадного алгоритма? Что такое жадный алгоритм?
2. Что такое остов графа? Что такое минимальный остов взвешенного графа?
3. Опишите работу алгоритма Прима.

§8.7 Динамическое программирование

ВСПОМИНАЕМ

Давайте вспомним,

- что такое рекурсивный алгоритм?
- какие обязательные условия необходимо соблюсти при реализации рекурсивного алгоритма, чтобы получить корректное решение исходной задачи?

ЧИТАЕМ

Под *динамическим программированием* обычно понимается подход к решению вычислительной задачи сведением её до аналогичных подзадач, аналогичных самой задаче. При этом важной особенностью динамического программирования является решение конкретной подзадачи единожды.

Так, классический рекурсивный алгоритм вычисления чисел Фибоначчи, приводимый нами ранее, не включает в себя условия решения подзадачи единожды.

```
def fib(n):  
    if n == 0:  
        return 0  
    if n == 1:  
        return 1  
    return fib(n - 1) + fib(n - 2)
```

То есть существуют шаги, которые мы будем повторять – вычисление промежуточных членов последовательности чисел Фибоначчи.

$$F_n = F_{n-1} + F_{n-2} = (\underline{F_{n-2}} + F_{n-3}) + \underline{F_{n-2}}$$

То есть в какой-то момент мы будем повторять расчёт элемента, который ранее уже получали.

Но мы можем избежать данного действия, если будем запоминать промежуточные шаги. Подобный подход называется мемоизацией (кешированием) – сохранение результатов выполнения функции для предотвращения повторных вычислений.

При решении большинства задач для мемоизации используют массивы (или списки в случае Python) или аналогичные структуры данных – иногда хорошо может подойти словарь.

Таким образом, мы можем сильно ускорить рекурсивное вычисление n -ого числа Фибоначчи.

```
fib_arr = list() # список для мемоизации

def fib(n):
    # обеспечиваем область видимости
    global fib_arr

    # крайние случаи
    if n == 0:
        return 0
    if n == 1:
        return 1
    # если F(n) не вычислялось -- вычислим
    if fib_arr[n] == -1:
        return fib(n - 1) + fib(n - 2)
    # иначе вернём уже подсчитанное значение
    return fib_arr[n]

number = int(input())
fib_arr = [-1] * (number + 1) # если значение F(k) не
                               # вычислялось,
                               # то считаем, что F(k) = -1
res = fib(number)
```

Подобное решение задачи поиска n -ого числа Фибоначчи называется динамическим программированием сверху-вниз. Это значит, что мы начинаем решать общую задачу, которую сводим до элементарных.

Есть ещё один подход, называемый динамическим программированием снизу-вверх. Суть которого заключается, что мы начинаем решать сначала элементарные подзадачи, требующиеся для решения общей задачи.

```
n = int(input())
```

```
fib_arr = [-1] * (n + 1)
fib_arr[0] = 0
fib_arr[1] = 1

for i in range(2, n + 1):
    fib_arr[i] = fib_arr[i - 1] + fib_arr[i - 2]

res = fib_arr[n]
```

Таким образом, можно упростить:

- *о динамическом программировании сверху-вниз* мы говорим, когда решаем задачу с помощью рекурсивного алгоритма и мемоизации;
- *о динамическом программировании снизу-вверх* мы говорим, когда при решении происходит индуктивное восхождение от крайнего случая к общему.

Со знанием того, что любой рекурсивный алгоритм можно привести к не рекурсивному, утвердим, что любую задачу, решаемую с помощью динамического программирования, можно решить, отталкиваясь как от принципа сверху-вниз, так и от принципа снизу-вверх.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Давайте разберём одну классическую задачу – она называется Кузнечик.

Перед клетчатой лентой сидит кузнечик. Длина ленты задаётся числом n . Известно, что в каждой клеточке есть некоторое количество зёрен, которое описывается отдельным числом для каждой клетки. Кузнечик может либо прыгать на следующую клетку, либо через одну. Оказываясь на конкретной клетке, кузнечик получает количество зёрен, равное числу, соответствующему данной клетке. Какое максимальное количество зёрен кузнечик может получить, добравшись до клетки под номером n ?

Очевидно, если бы в клетках были записаны только положительные числа, кузнечик бы прошёл по каждой клетке – задача была бы сводима к поиску суммы элементов массива. Но мы будем считать, что в клетке может быть записано отрицательное число.

С учётом того, что в клетку под номером k кузнечик может попасть из клеток под номерами $(k - 1)$ и $(k - 2)$, мы будем считать, что добравшись до k -ой клетки, кузнечик может получить количество зёрен равное максимальному от того, что он может получить в клетке с номером $(k - 1)$ и $(k - 2)$, и плюс значение в данной клетке.

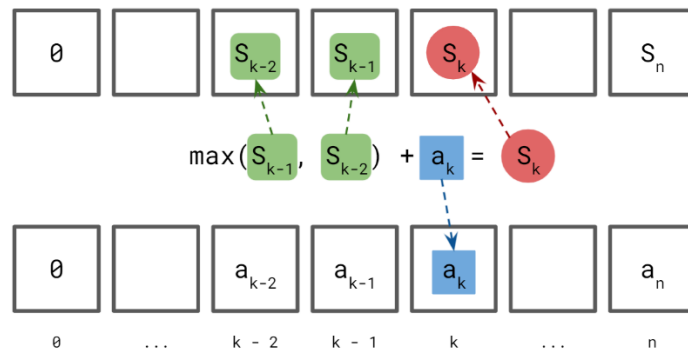


Рис. 182 Иллюстрация к задаче

То есть, мы можем рекурсивно описать формулу суммы для n -ой клетки:

$$\begin{aligned} \forall i \in [1, n] : a_i &\in A \\ S_0 &= 0 \\ S_1 &= a_1 \\ S_n &= \max(S_{n-1}, S_{n-2}) + a_n \end{aligned}$$

Давайте реализуем динамику сверху-вниз для решения данной задачи:

```
sum_arr: int # список максимальных сумм
arr: int # изначальное количество зёрен в клетках

def solve(n):
    global sum_arr
    global arr

    # рассматриваем крайние случаи -- нулевая и первая клетки
    if n < 2:
        sum_arr[n] = arr[n]
        return

    # если решения для предыдущих шагов не было, выполним
    if sum_arr[n - 2] == -1:
        solve(n - 2)
    if sum_arr[n - 1] == -1:
        solve(n - 1)

    # согласно формуле
    sum_arr[n] = max(sum_arr[n - 1], sum_arr[n - 2]) + arr[n]

n = int(input())
# скомпенсируем индексы за счёт добавления 0,
# создадим нужные списки
arr = [0] + list(map(int, input().split()))
sum_arr = [-1] * (n + 1)
```

```
# решим задачу
solve(n)
# выведем ответ
print(sum_arr[n])
```

Стоит обратить внимание, что в языке программирования Python нет оптимизации хвостовой рекурсии. Максимальное количество рекурсивных вызовов в Python составляет 999. Поэтому для большого количества клеток на ленте подобный алгоритм не подойдёт с точки зрения используемого нами языка, хотя формально алгоритм корректный.

Тогда давайте решим данную задачу динамикой снизу-вверх:

```
n = int(input())
# скомпенсируем индексы за счёт добавления 0,
# создадим нужные списки
arr = [0] + list(map(int, input().split()))
sum_arr = [-1] * (n + 1)

# описываем формулу
sum_arr[0] = arr[0]
sum_arr[1] = arr[1]
# производим расчёт для неопределённых случаев
for i in range(2, n + 1):
    sum_arr[i] = max(sum_arr[i - 1], sum_arr[i - 2]) + arr[i]

print(sum_arr[n])
```

Как вы можете заметить, подобное решение оказывается даже проще с точки зрения написания кода.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа №78

Перед клетчатой лентой сидит кузнечик. Длина ленты задаётся числом n . Известно, что в каждой клеточке есть некоторое количество зёрен, которое описывается отдельным числом для каждой клетки. Кузнечик может прыгать на следующую клетку, через одну клетку или на клетку, кратную двум. Оказываясь на конкретной клетке, кузнечик получает количество зёрен, равное числу, соответствующему данной клетке. Какое максимальное количество зёрен кузнечик может получить, добравшись до клетки под номером n ?

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1

Перед клетчатой лентой сидит кузнечик. Длина ленты задаётся числом n . Известно, что в каждой клеточке есть некоторое количество зёрен, которое описывается отдельным числом для каждой клетки. Кузнечик может прыгать на следующую клетку, через две клетки или через четыре клетки. Оказываясь на конкретной клетке, кузнечик получает количество зёрен, равное числу, соответствующему данной клетке. Какое максимальное количество зёрен кузнечик может получить, добравшись до клетки под номером n ?

Задание 2

Перед клетчатой лентой сидит кузнечик. Длина ленты задаётся числом n . Известно, что в каждой клеточке есть некоторое количество зёрен, которое описывается отдельным числом для каждой клетки. Кузнечик может прыгать на следующую клетку, через одну клетку, через две клетки, ..., через k клеток. Оказываясь на конкретной клетке, кузнечик получает количество зёрен, равное числу, соответствующему данной клетке. Какое максимальное количество зёрен кузнечик может получить, добравшись до клетки под номером n ?

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Что такое динамическое программирование?
2. В чем разница между динамическим программированием сверху-вниз и снизу вверх?
3. Насколько язык программирования Python подходит для реализации динамического программирования сверху-вниз при решении задачи с анализом большого количества данных?

§8.8 Раскраска графов

ВСПОМИНАЕМ

Давайте вспомним,

- что такое граф?
- какие классификации графов вы знаете?
- что такое минимальное остовное дерево?

ЧИТАЕМ

Под *раскраской графа* чаще всего понимают такое соответствие вершин и цветов (цвета обычно обозначают цифрами), при котором каждой вершине присвоен определённый цвет и для любых двух смежных вершин цвета не одинаковы.

В действительности раскраска графов часто применяется в прикладных задачах. Вы сами в неявном виде занимались раскраской графов, если решали задачи в рамках известной головоломки sudoku – если представить, что каждая ячейка головоломки является вершиной и связана со всеми вершинами (ячейками), в которых записаны числа, отличные от числа, соответствующего конкретной ячейке, то становится очевидно, что подобный граф можно раскрасить – и подобная раскраска есть суть расставления чисел в таблице. На основе алгоритмов раскраски графов чаще всего и создаются алгоритмы генерации задач sudoku.

Также широкое использование такой абстракции можно встретить при составлении различных расписаний: расписание транспорта, турнирные таблицы, расписание уроков в школе и т. д.

ОБСУЖДАЕМ

Обсудите, что может являться вершинами и связями (рёбрами) при задаче составления школьного расписания? Обдумайте все взаимосвязи школьных предметов, учителей, классов учеников, дней недели.

ЧИТАЕМ

На самом деле задача оптимальной раскраски произвольного графа является NP-полной, то есть на настоящий момент не существует алгоритма, время которого бы полиномиально зависело от количества входных данных (вершин и рёбер). То есть точный алгоритм, который в настоящий момент известен человечеству, является переборным и его время выполнения оценивается экспоненциальной сложностью.

Поэтому для раскраски графов с большим количеством вершин и рёбер обычно используют различные подходы, которые могут вести за собой неоптимальные результаты – вполне вероятно создание большего количества классов вершин (разбиение по краскам), чем минимально возможное. Но оптимальное решение порой можно получить.

Мы с вами в предыдущих параграфах рассматривали принцип жадного выбора. Для решения задачи раскраски существует жадный алгоритм, который называется *жадная раскраска*.

Основная суть жадной раскраски заключается в последовательном анализе вершин, находящихся в определённом порядке, и присваивания каждой вершине минимально возможного цвета.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Давайте попробуем написать реализацию жадной раскраски на языке программирования Python.

```

n: int # количество вершин графа
m: int # количество рёбер графа
graph: list # пусть граф описан списком смежности

# создаем список цветов вершин
# считаем, что изначально вершины не покрашены -- -1
vertex_colors = [-1] * n

# проходим по всем вершинам по порядку
for vertex in range(n):

    neighbour_colors = set()
    # добавляем все цвета соседей рассматриваемой вершины в
    множество
    for neighbour_vertex in graph[vertex]:
        if vertex_colors[neighbour_vertex] != -1:

neighbour_colors.add(vertex_colors[neighbour_vertex])

    # выбираем минимальный цвет, не совпадающий с цветами
    соседей
    temp_color = 0
    while temp_color in neighbour_colors:
        temp_color += 1

    vertex_colors[vertex] = temp_color

print(vertex_colors)

```

ОБСУЖДАЕМ

Попробуйте придумать пример, на котором данный алгоритм будет выдавать не оптимальный вариант раскраски вершин графа.

ЧИТАЕМ

В одном из предыдущих уроков мы рассматривали задачу поиска минимального остовного дерева взвешенного графа. Для решения данной задачи нами был рассмотрен алгоритм Прима.

В рамках данного урока мы рассмотрим ещё один алгоритм – алгоритм Краскала. Но перед этим нам стоит познакомиться с одной очень простой структурой данных – с системой непересекающихся множеств.

Система непересекающихся множеств (СНМ) – это структура данных, позволяющая содержать в себе информацию о непересекающихся

множествах (элементы множеств не должны совпадать). Сами множества имеют в данном случае древовидную структуру – каждое множество есть дерево, в котором элементы расположены в некоторой иерархии. СНП, чаще всего, позволяет выполнять следующие операции:

- `find(elem)` – поиск элемента, определение корня множества, в которое входит данный элемент;
- `merge(elem_1, elem_2)` – слияние множеств, операция, позволяющая объединить два дерева, в которые входят два указанных элемента.

Несмотря на сложное описание, подобная структура данных является довольно простой с точки зрения реализации. Так, чаще всего хранение данных выполняется с помощью простого массива или списка в случае Python.

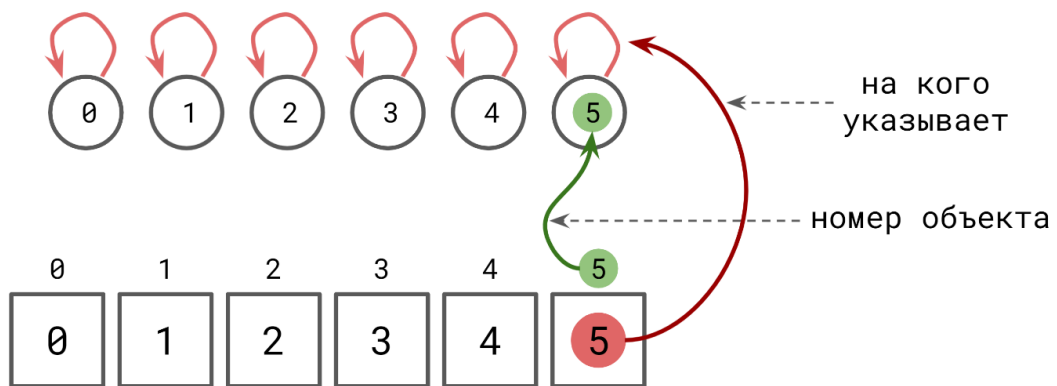


Рис. 183 Иллюстрация к задаче

Индексы массива (списка) в данном случае являются номерами всех объектов, а значение, которое хранится по индексу – это номер объекта, с которым связан данный. Пусть i — индекс элемента в списке, а j — значение элемента в списке по индексу i , что означает, что i -ый объект указывает на j -ый объект. Так, например, ниже представлена в виде графа и в виде массива конфигурация трёх независимых множеств.

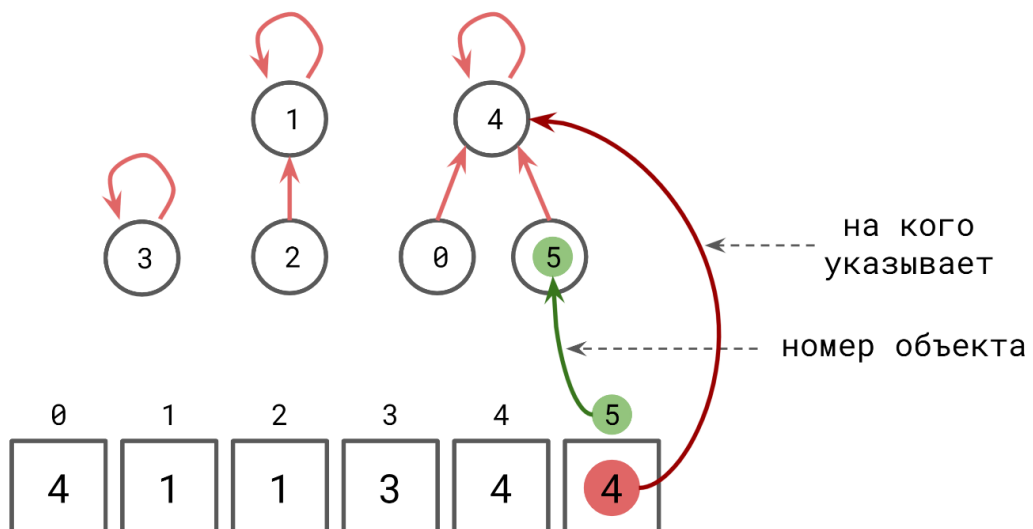


Рис. 184 Иллюстрация к задаче

При использовании подобной структуры данных вопрос реализации операций более сложен, чем вопрос представления данных в памяти компьютера.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Систему непересекающихся множеств можно описать на языке программирования Python следующим образом:

```
# система непересекающихся множеств
union_find = list()
# размеры множеств (деревьев), составляющих СНП
set_sizes = list()

def merge(elem_1, elem_2):
    # СНП и размер множеств объявлены глобально
    global union_find
    global set_sizes

    # находим корни указанных двух элементов
    elem_1_root = find(elem_1)
    elem_2_root = find(elem_2)
    # если их корни совпадают, то объединять нечего
    if elem_1_root == elem_2_root:
        return # завершаем работу
    # иначе, корень меньшего дерева
    # должен указывать на корень большего дерева
    if set_sizes[elem_1] < set_sizes[elem_2]:
        union_find[elem_1_root] = elem_2_root
        # изменяем размер дерева в корне
        set_sizes[elem_2_root] += set_sizes[elem_1_root]
    else:
        union_find[elem_2_root] = elem_1_root
        # изменяем размер дерева в корне
        set_sizes[elem_1_root] += set_sizes[elem_2_root]

def find(elem):
    # СНП и размер множеств объявлены глобально
    global union_find
    global set_sizes

    if union_find[elem] == elem:
        # если элемент указывает на себя,
        # возвращаем его индекс
        return elem
```

```

# иначе смотрим, на кого указывает элемент,
# на который указывает данный
new_root = find(union_find[elem])
# и переопределяем его указатель на корень всего дерева
union_find[elem] = new_root
return new_root

```

Обратите внимание, что функция `find` при переопределении корня использует идею динамического программирования, то есть для того, чтобы определить корень вершины `a`, нам необходимо определить корень вершины, на которую ссылается вершина `a`, что выполняется рекурсивно, при этом на каждом шаге рекурсии мы переопределяем корень для каждой очередной вершины.

Работу данной функции можно изобразить следующим образом.

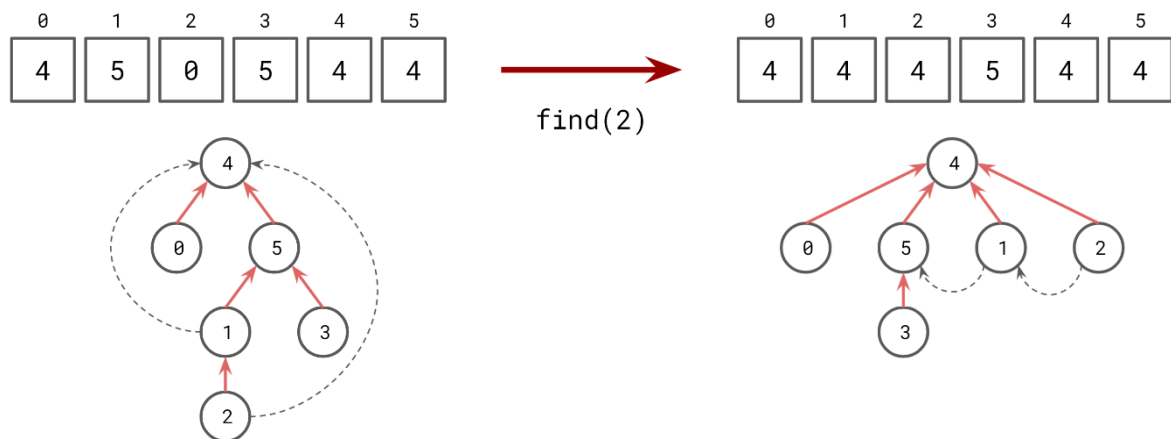


Рис. 185 Иллюстрация работы функции

То есть, при поиске и при объединении множеств (так как в объединении используется функция поиска) происходит перестройка дерева. Это выполняется для того, чтобы при каждом следующем поиске выполнять меньше действий (меньше рекурсивных вызовов).

Чтобы создать саму систему непересекающихся множеств, достаточно написать следующие строки:

```

n = int(input()) # количество объектов

union_find = list(range(n)) # изначально все объекты
                        # указывают на самих себя
set_sizes = [1] * n # изначально размер каждого дерева -- 1

```

На самом деле можно упростить данный код и написание последующей программы, если обернуть данный код функцией.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа № 79

Измените код работы с системой непересекающихся множеств с помощью создания функции инициализации `init_union_find (n)`, которая заполняет систему непересекающихся множеств `n` элементами.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа № 80

Проверьте, является ли неориентированный невзвешенный граф из `n` вершин и `m` ребер связным (можно ли из любой произвольной вершины попасть в любую другую вершину). При реализации программы, решающей данную задачу, воспользуйтесь системой непересекающихся множеств и полным перебором вершин.

ЧИТАЕМ

Алгоритм Краскала можно описать следующим образом:

- 1) сортируем все ребра по неубыванию их весов;
- 2) проверяем очередное ребро: если вершины ребра находятся не в одной компоненте связности, то добавляем данное ребро в конструируемое остовное дерево;
- 3) если остались свободные вершины, возвращаемся к шагу (2).

Как видно из алгоритма, нам необходимо проверять, находятся ли вершины, определяющие ребро, в одной компоненте связности. Данный факт легко описывать и проверять с помощью системы непересекающихся множеств.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Задача перед нами всё та же – найти остов взвешенного неориентированного графа, заданного `n` вершинами и `m` ребрами. Каждое ребро задаётся тройкой чисел – пара вершин и вес.

Решение данной задачи с помощью алгоритма Краскала можно написать на языке программирования Python следующим образом:

```
# вводится количество вершин и рёбер
n, m = tuple(map(int, input().split()))
init_union_find(n) # создадим СНМ по вершинам

# будем хранить граф в виде списка рёбер, где каждый элемент:
# (вес, первая вершина, вторая вершина)
edges = list()
for _ in range(m):
    first, second, w = tuple(map(int, input().split()))
```

```

edges.append((w, first, second))

# отсортируем ребра по весу (неубывание)
edges.sort()

min_span_tree = list()
# рассматриваем все рёбра, начиная с самого малого по весу
for edge in edges:
    # выделяем вершины ребра
    first = edge[1]
    first_root = find(first)
    second = edge[2]
    second_root = find(second)
    if first_root != second_root:
        # если вершины не находятся в одной компоненте
        # связности, то добавляем ребро
        min_span_tree.append(edge)
        # объединяем множества
        merge(first, second)

print(min_span_tree)

```

Обратите внимание, что в данной реализации анализируемый граф и минимальное остовное дерево мы просто храним в виде списка рёбер. При условии, что реализованная нами функция `find` для системы непересекающихся множеств работает за $T(n) \in O(\log(n))$, где n – количество объектов. А обращение внутри цикла в реализованном нами алгоритме к СНМ происходит порядка количества рёбер в графе, то можно утверждать, что асимптотическая сложность данного алгоритма без сортировки линейно-арифметически зависит от количества рёбер $T(|V|, |E|) = O(|E| * \log(|V|))$. Но при учёте сортировки и условия о том, что количество рёбер в графе не меньше количества вершин, можно утверждать $T(|V|, |E|) = O(|E| * \log(|E|))$. Асимптотическая сложность данного алгоритма меньше асимптотической сложности реализованного нами алгоритма Прима $O(|E|^2)$.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1

Напишите программу, которая сравнивала бы количество цветов раскраски взвешенного неориентированного графа и его минимального остовного дерева при использовании алгоритма жадной раскраски. Получение минимального остовного дерева должно происходить при помощи алгоритма Краскала.

Задание 2

Напишите переборный (точный) алгоритм для раскраски взвешенного неориентированного графа.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Что такое раскраска графа?
2. В чем заключается основная идея жадного алгоритма раскраски графа?
3. Что такое система непересекающихся множеств? Как данная структура данных может быть реализована в памяти компьютера?
4. Опишите основную идею алгоритма Краскала.

§8.9 Динамическое программирование

ВСПОМИНАЕМ

Давайте вспомним,

- что такое направленный невзвешенный граф?
- Какие способы представления графа в памяти компьютера мы знаем?

ЧИТАЕМ

Давайте представим следующую задачу – нам дано n городов, соединённых m однонаправленными дорогами, то есть по каждой дороге можно ехать только в одном направлении. Города и дороги между ними составляют ациклический граф. Вопрос задачи – сколько существует путей из города под номером a_0 в город под номером a_k ?

ОБСУЖДАЕМ

Какие способы решения данной задачи вы знаете?

РЕШАЕМ В ТЕТРАДИ

Задание 68.

Решите задачу. На рисунке — схема дорог, связывающих города 0-8. По каждой дороге можно двигаться только в одном направлении, указанном стрелкой. Сколько существует различных путей из города 0 в город 9?

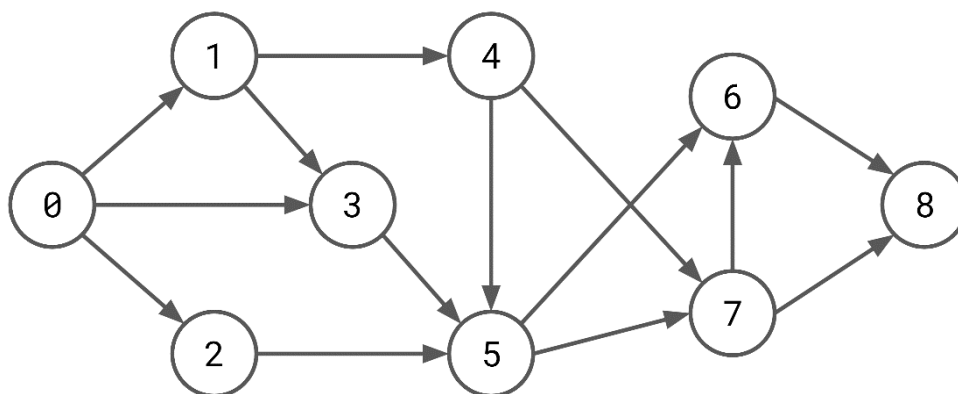


Рис. 186 Направленный граф к заданию 68

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Давайте разберём алгоритм решения данной задачи и попробуем написать его на языке программирования.

Пусть нам дан следующий граф, в качестве конкретного примера описанной задачи, состоящей из 8 вершин и 14 рёбер.

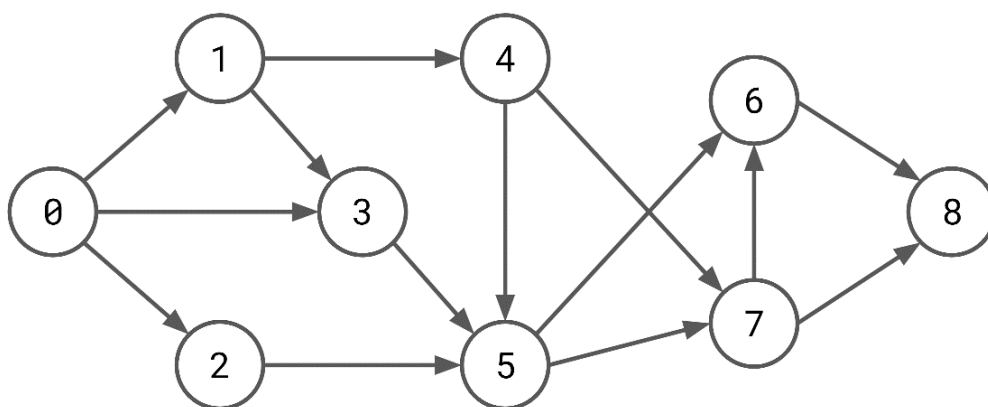


Рис. 187 Граф в 8 вершин и 14 рёбер

При рассмотрении данной задачи, можно прийти к простому выводу – количество путей в вершину 8 равно сумме количества возможных путей в вершину 7 и 6:

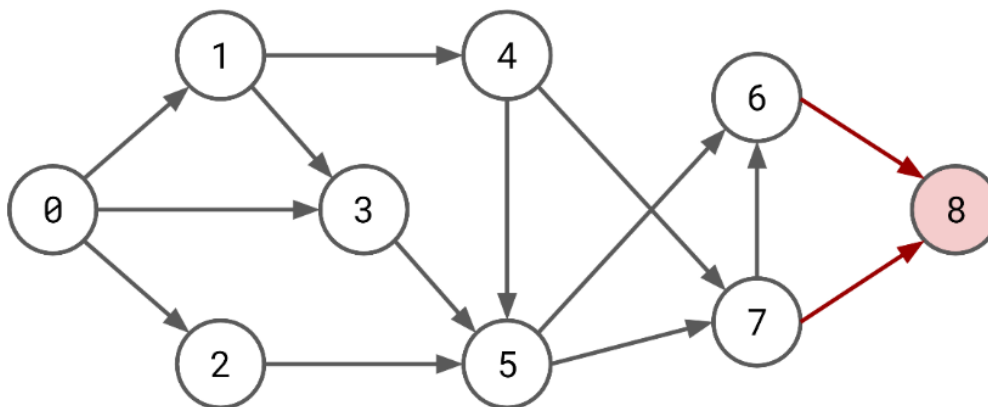


Рис. 188 Выделение итоговой вершины

При этом, для 6 и 7 вершины мы можем утвердить аналогично, что количество путей в вершину 7 равно сумме количества путей в вершину 4 и 5, а в вершину 6 – сумме количества путей в вершину 5 и 7:

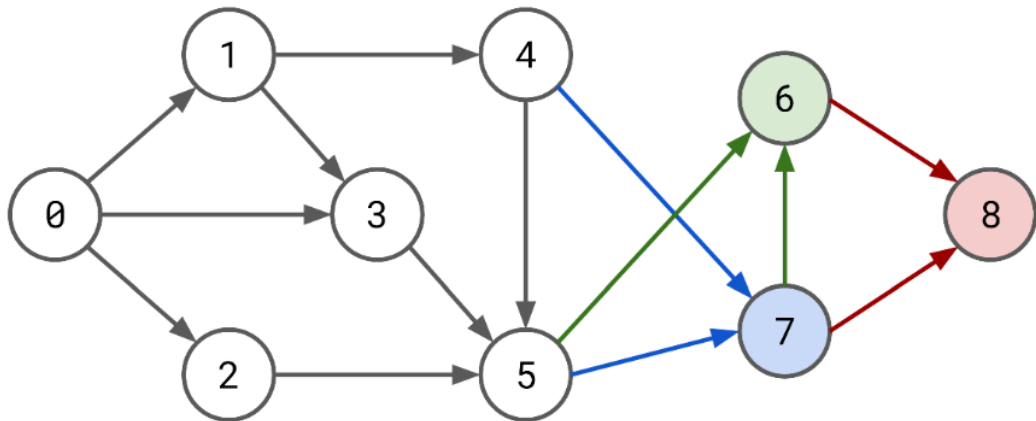


Рис. 189 Пути к вершинам 6 и 7

Данные рекурсивные рассуждения можно провести для всех вершин данного графа с подсчётом. При этом крайним случаем будем считать количество путей в вершину 0. Мы будем считать, что в вершину 0 из вершины 0 мы можем попасть только одним способом. Тогда можно написать следующий код, решающий данную задачу:

```
prices = list() # количество путей из 0-ой в i-ую вершину
graph = list() # описание графа
```

```
def solve(vertex):
    # обеспечиваем область видимости
    global prices
    global graph
    # если количество путей подсчитано, возвращаем его
    if prices[vertex] != 0:
        return prices[vertex]
    # иначе производим подсчёт
    res = 0
    for neighbour_vertex in graph[vertex]:
        # как сумму путей в соседние города, из которых
        # можем попасть в данный
        res += solve(neighbour_vertex)

    # запоминаем подсчитанное
    prices[vertex] = res
    # возвращаем количество путей в вершину vertex
    return prices[vertex]
```

```
# вводится количество вершин и рёбер
```

```

n, m = tuple(map(int, input().split()))

# создаём список для хранения количества путей
prices = [0] * n
prices[0] = 1
# список для описания графа
graph = [list() for _ in range(n)]

# ввод рёбер
for _ in range(m):
    first, second = tuple(map(int, input().split()))
    # для каждой вершины храним вершины, из которых
    # можно попасть в данную
    graph[second].append(first)

# находим количество путей в вершину с индексом n-1
print(solve(n - 1))

```

Как вы можете заметить, данный алгоритм есть воплощение классического динамического программирования сверху-вниз.

Обратите внимание, что мы представили рёбра графа в обратном порядке. Для каждой вершины мы записали, какие вершины на неё указывают.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа № 81

Напишите программу по принципу динамического программирования снизу-вверх для решения аналогичной задачи.

ЧИТАЕМ

Давайте рассмотрим ещё один класс задач, решаемых рекурсивно. Начнём с маленького примера.

Предположим, что у нас есть три команды:

- прибавить 1;
- умножить на 2;
- прибавить 3.

Какое количество способов получения числа n из единицы существует?

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Давайте построим математическую функцию, выражающую данную зависимость:

$$P(1) = 1$$

$$\forall (n > 1), \quad P(n) = \begin{cases} P(n-1), & n-3 < 1, n \neq 2 \\ P(n-1) + P(\frac{n}{2}), & n-3 < 1, n \div 2 \\ P(n-1) + P(n-3), & n-3 \geq 1, n \neq 2 \\ P(n-1) + P(n-3) + P(\frac{n}{2}), & n-3 \geq 1, n \div 2 \end{cases}$$

Записав функцию таким образом, мы можем буквально перевести её на язык программирования:

```
memory = list() # список для мемоизации

def solve(n):
    # рассматриваем крайний случай
    if n == 1:
        return 1

    # если количество известно, возвращаем его
    if memory[n] != -1:
        return memory[n]

    # иначе производим подсчёт
    first = 0
    second = 0
    third = 0

    # рассматриваем возможность выполнить операции
    first = solve(n - 1)
    if n % 2 == 0:
        second = solve(n // 2)
    if n - 3 >= 1:
        third = solve(n - 3)

    # складываем все возможные значения
    memory[n] = first + second + third
    return memory[n]

n = int(input())
memory = [-1] * (n + 1)
print(solve(n))
```

Обратите внимание, что данный алгоритм очень похож на алгоритм решения предыдущей рассмотренной задачи. В действительности эти две задачи эквивалентны. Если мы представим в качестве вершин числа от 1 до n, а используемые операции будем воспринимать как правила определения

ребра, то мы можем воспользоваться даже алгоритмом предыдущей задачи, где под количеством путей будем понимать количество способов получения из числа 1 числа n .

Данная задача снова решалась динамически сверху вниз.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа № 82

Напишите программу по принципу динамического программирования снизу-вверх для решения аналогичной задачи.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа № 83

2. Предположим, что у нас есть те же три команды:

- a. прибавить 1;
- b. умножить на 2;
- c. прибавить 3.

Напишите программу, которая отвечала бы, какое количество способов получения числа n из числа 3 существует, если при вычислениях мы не можем проходить через числа 6 и 12? Подсказка – числа 6 и 12 можно рассмотреть, как крайние случаи.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Предположим, что у нас есть три команды:

- a. прибавить 1;
- b. умножить на 2;
- c. умножить на 3.

Задание 1

Напишите программу, используя идею динамического программирования сверху-вниз, которая отвечала бы, какое количество способов получения числа n из числа 1 существует, если при вычислениях мы не можем проходить через числа 15 и 33?

Задание 2

Напишите программу, используя метафору графа (граф должен быть построен программой), которая отвечала бы, какое количество способов получения числа n из числа 1 существует, если при вычислениях мы не можем проходить через числа 15 и 33?

Задание 3

Напишите программу, которая отвечала бы, какое количество способов получения числа n из числа 1 существует, если при вычислениях мы обязательно должны пройти через число 12 и не пройти число 33?

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Опишите алгоритм, с помощью которого можно найти количество путей в ориентированном ациклическом графе.
2. Скажите, действительно ли n -ое число Фибоначчи – это количество способов получить из единицы натуральное число n с помощью операций добавления 1 и добавления 2?
3. Вспомните, чем отличается динамическое программирование сверху-вниз от рекурсивного алгоритма?
4. Действительно ли вопрос о количестве способов получения из числа a числа b с помощью ограниченного количества операций можно свести к вопросу о количестве путей в ориентированном ациклическом графе? Проведите соответствие понятий – чем в данном случае будут операции, промежуточные расчётные результаты операций?

§8.10 Применение законов алгебры логики

ВСПОМИНАЕМ

Давайте вспомним,

- что такое высказывания, алгебра логики, логические значения;
- основные логические операции;
- законы алгебры логики.

ЧИТАЕМ

Предположим, у нас есть система логических уравнений:

$$\{(x_1 \rightarrow x_2) \vee (x_1 \rightarrow x_3) = 1 (x_2 \rightarrow x_3) \vee (x_2 \rightarrow x_4) = 1 \dots (x_8 \rightarrow x_9) \vee (x_8 \rightarrow x_{10}) = 1$$

и нам необходимо определить количество решений данной системы.

Очень часто для решения систем логических уравнений, в которые включены однотипные логические выражения, используется метод отображений.

Он заключается в том, что изначально мы рассматриваем первое выражение как форму, которую потом будем применять к каждому последующему.

Так, изначально мы можем построить таблицу истинности для первого выражения из системы:

| x_1 | x_2 | x_3 | f |
|-------|-------|-------|-----|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Как видно из данной таблицы, здесь присутствует только один набор, на котором первое логическое выражение из системы является ложным.

Далее мы отталкиваемся от того, что при знании x_1, x_2 мы всегда можем подобрать такое x_3 , что выражение будет возвращать истинное значение.

Так, если $x_1, x_2 = (0, 0)$, то x_3 может быть в данном случае любым, $x_1, x_2 = (0, 1)$, то x_3 может быть в данном случае любым, если $x_1, x_2 = (1, 0)$, то x_3 может быть только 1, иначе логическое высказывание будет ложным, если $x_1, x_2 = (1, 1)$, то x_3 снова может быть любым. Данные рассуждения можно представить в виде графа:

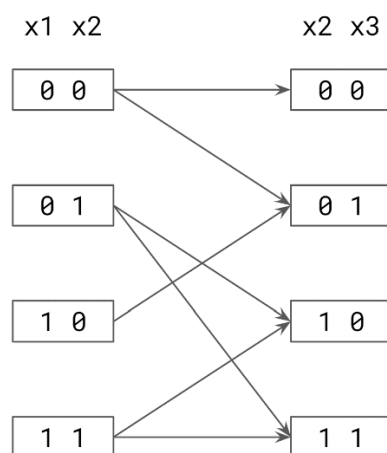


Рис. 190 Иллюстрирующий рассуждение граф

При этом посмотрите, что в качестве второй доли (вершины с наборами справа) мы взяли пару переменных x_2, x_3 . Это связано с тем, что мы на основе однотипности логических выражений в системе можем утверждать, что пара x_2, x_3 во втором выражении эквивалентна паре x_1, x_2 в первом. То есть, это и есть отображения. При этом мы считаем, что одна пара может отобразиться в другую, если их пересечение даёт истину в соответствующем логическом выражении. Мы можем продолжить данные рассуждения и получить следующий граф:

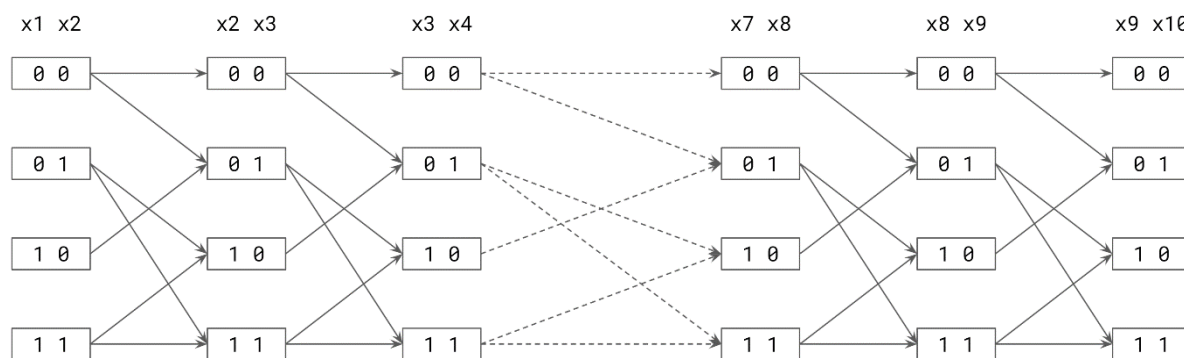


Рис. 191 Модель рассуждения

В общем случае мы свели данную задачу к задаче поиска количества путей, так как одно ребро в данном графе подразумевает “истинный” переход, то есть переход, который для конкретного выражения в виде пересечения значений переменных в подстановке делает его истинным. Таким образом, каждый путь в группу вершин x_9, x_{10} определяют набор значений, являющийся решением изначальной системы. То есть, сумма количества путей из группы x_1, x_2 в группу x_9, x_{10} есть ответ на поставленную изначально задачу.

В действительности данная задача может быть легко решена в тетради.

РЕШАЕМ В ТЕТРАДИ

Задание 69

Решите в тетради данную задачу, сведённую до поиска количества путей в графе.

ЧИТАЕМ

Но если количество логических высказываний будет сильно больше? Например, их будет n , где n может быть до 10000.

$$\{(x_1 \rightarrow x_2) \vee (x_1 \rightarrow x_3) = 1(x_2 \rightarrow x_3) \vee (x_2 \rightarrow x_4) = 1 \dots (x_{n-2} \rightarrow x_{n-1}) \vee (x_{n-2} \rightarrow x_n) = 1$$

Тогда данную задачу сложно решить привычным нам методом. Нужно будет идти к решению через поиск зависимостей. Но это может быть в некоторых случаях затруднительно с точки зрения времени. Тогда мы можем прийти к вычислениям.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа № 84

Подобную задачу удобно решить средствами языка программирования.

```
n = int(input()) # вводим количество x, входящих в выражения
системы
```

```
table = list() # матрица, характеризующая количество путей в
вершину,
```

```
# определённую парой переменных и парой значений
```



```

# формируем четыре строки, характеризующие наборы
# значений пары переменных
# table[0] --> (0, 0), table[1] --> (0, 1),
# table[2] --> (1, 0), table[3] --> (1, 1)
for _ in range(4):
    # для каждого набора перечисляем количество пар переменных
    # последовательных пар из n переменных -- (n - 1)
    table.append([0] * (n - 1))

# заполняем количество путей для вершин (x_1, x_2)
for i in range(4):
    table[i][0] = 1

for j in range(1, n - 1):
    # в вершину (x_j, x_(j + 1)) со значением (0, 0) можно
    # попасть
    # только из вершины (x_(j - 1), x_j) со значением (0, 0)
    table[0][j] += table[0][j - 1]
    # в вершину (x_j, x_(j + 1)) со значением (0, 1) можно
    # попасть
    # из вершины (x_(j - 1), x_j) со значением (0, 0) и
    # из вершины (x_(j - 1), x_j) со значением (1, 0)
    table[1][j] += table[0][j - 1] + table[2][j - 1]
    # в вершину (x_j, x_(j + 1)) со значением (1, 0) можно
    # попасть
    # из вершины (x_(j - 1), x_j) со значением (0, 1) и
    # из вершины (x_(j - 1), x_j) со значением (1, 1)
    table[2][j] += table[1][j - 1] + table[3][j - 1]
    # в вершину (x_j, x_(j + 1)) со значением (1, 1) можно
    # попасть
    # из вершины (x_(j - 1), x_j) со значением (0, 1) и
    # из вершины (x_(j - 1), x_j) со значением (1, 1)
    table[3][j] += table[1][j - 1] + table[3][j - 1]

# производим подсчёт решений (путей)
res = 0
for i in range(4):
    res += table[i][n - 2]

print(res)

```

Таким образом нами было приведено решение задачи поиска количества корней логической системы, представленной ниже для любого $\forall n: n > 2$.

$$\{(x_1 \rightarrow x_2) \vee (x_1 \rightarrow x_3) = 1 (x_2 \rightarrow x_3) \vee (x_2 \rightarrow x_4) = 1 \dots (x_{n-2} \rightarrow x_{n-1}) \vee (x_{n-2} \rightarrow x_n) = 1$$

Аналогичным образом можно поступать и с другими системами однотипных логических выражений.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Предположим, что у нас есть три команды:

- a. прибавить 1;
- b. умножить на 2;
- c. умножить на 3.

Задание 1

Напишите программу для нахождения количества решений системы логических уравнений для произвольно n :

$$\{(x_1 \rightarrow y_1) \rightarrow (x_2 \rightarrow y_2) = 1 (x_1 \rightarrow y_1) \rightarrow (x_2 \rightarrow y_2) = 1 \dots (x_{n-1} \rightarrow y_{n-1}) \rightarrow (x_n \rightarrow y_n) = 1$$

Задание 2

Напишите программу для нахождения количества решений системы логических уравнений для произвольно n :

$$\{(y_1 \rightarrow y_2 \wedge x_1) \wedge (x_1 \rightarrow x_2) = 1 (y_2 \rightarrow y_3 \wedge x_2) \wedge (x_2 \rightarrow x_3) = 1 \dots (y_{n-1} \rightarrow y_n \wedge x_{n-1}) \wedge (x_{n-1} \rightarrow x_n) = 1$$

Рекомендуем в качестве пар для отображения рассматривать

$$(x_i, y_i) \rightarrow (x_{i+1}, y_{i+1})$$

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Почему разобранный в рамках данного параграфа метода называется методом отображения?
2. Когда сведение задачи решения системы логических уравнений к своей подзадаче с помощью метода отображения допустимо?
3. Действительно ли метод отображения в решении системы однотипных логических уравнений переводит в задачу поиска количества путей в ориентированном графе?

§8.11 Поиск кратчайшего пути на графе

ВСПОМИНАЕМ

Давайте вспомним,

- что такое ненаправленный невзвешенный граф?
- какие способы представления графа в памяти компьютера мы знаем?
- что такое ненаправленный взвешенный граф?

ЧИТАЕМ

Ранее при знакомстве с графами мы с вами рассматривали базовые алгоритмы на графах, к числу которых отнесли обход графа в ширину и обход графа в глубину.

Данные алгоритмы могут помочь решить довольно разные задачи. Так, с помощью обхода в ширину, запущенного из вершины v_1 , можно найти кратчайшее расстояние от вершины v_1 до любой другой. По сути своей подобное уже включено в реализацию самого алгоритма. Так как алгоритм последовательно рассматривает вершины по уровню удалённости, то нам достаточно просто запоминать уровень удалённости для каждой вершины.

ОБСУЖДАЕМ

Какие средства языка позволяют удобно сохранять информацию о том, когда мы достигли конкретной вершины?

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Давайте попробуем написать алгоритм, решающий данную задачу. Для этого нам надо чуть-чуть изменить тот код, который мы уже писали ранее:

```
n: int # количество вершин графа
m: int # количество рёбер графа
graph: list # пусть граф описан списком смежности

# предположим, что стартовой вершиной является 0
start_vertex = 0

# is_visited = [False] * n
# список посещённых вершин можно заменить списком удалённости
dist = [-1] * n # -1 -- означает, что расстояние ещё не
подсчитано

queue = list() # очередь для работы BFS

# добавляем в очередь стартовую вершину
queue.append(start_vertex)
```

```

dist[start_vertex] = 0 # говорим, что от начальной вершины
                        # до неё самой -- расстояние равно 0

# пока очередь не пуста
while len(queue) != 0:
    # достаем из неё очередную вершину
    this_vertex = queue.pop(0)
    # и добавляем все соседние непомяченные вершины в очередь
    for neighbour_vertex in graph[this_vertex]:
        # если расстояние для соседней вершины не подсчитано
        if dist[neighbour_vertex] == -1:
            # то оно на единицу больше,
            # чем расстояние до текущей вершины
            dist[neighbour_vertex] = dist[this_vertex] + 1
            queue.append(neighbour_vertex)

```

Обратите внимание, что реализация ввода графа в данном алгоритме опущена.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа № 85

Напишите программу, которая для неориентированного невзвешенного графа с количеством вершин n и количеством рёбер m находит для каждой вершины расстояния до всех остальных вершин графа.

ЧИТАЕМ

Задача поиска кратчайшего расстояния в невзвешенном неориентированном графе является довольно простой. И с ней редко возникают трудности у программистов, знающих идеи BFS или DFS. Да, с помощью DFS данную задачу также можно решить, но алгоритм при этом будет больше усложнён, нежели BFS. Это связано с тем, что нам придётся посещать одни и те же вершины несколько раз и выбирать более короткий путь. Поэтому в практической деятельности подобную реализацию не рекомендуем, но для себя можно попробовать её написать.

Если перед нами встанёт похожая задача, но вместо невзвешенного графа даётся взвешенный, то переделать BFS для этих целей становится проблемой.

Существует несколько алгоритмов для определения кратчайшего пути во взвешенном графе. Мы рассмотрим один из них – алгоритм Дейкстры.

Алгоритм Дейкстры, как уже было сказано выше, является алгоритмом поиска кратчайших путей от заданной вершины до всех вершин рассматриваемого графа.

Давайте неформально опишем данный алгоритм:

- 1) полагаем, что расстояние до каждой вершины графа от начальной вершины до начала алгоритма – бесконечность;
- 2) утверждаем, что расстояние от начальной вершины до неё самой – нуль;
- 3) помещаем все вершины в очередь с приоритетом по расстоянию (чем меньше расстояние до вершины от начальной вершины, тем приоритетнее считается вершина);
- 4) достаём из очереди с приоритетом вершину (с минимальным расстоянием), отмечаем её как посещённую; проверяем все вершины, смежные с данной и ранее не посещённые – путь, проходящий через данную вершину в смежную, мог стать короче, в таком случае корректируем имеющиеся значения;
- 5) если посещены не все вершины, возвращаемся к пункту (4).

Как вы можете заметить, при реализации данного алгоритма основой является принцип жадного выбора. Мы фиксируем минимально возможное расстояние для каждой вершины, после чего утверждаем, что если в какую-то вершину (еще не рассмотренную) можно попасть из данной, то расстояние до этой вершины есть минимальное расстояние расстояний до вершин, из которых можно попасть в данную, плюс цена такого перехода.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Давайте посмотрим на примере, как работает алгоритм Дейкстры.

Пусть нам дан взвешенный неориентированный граф:

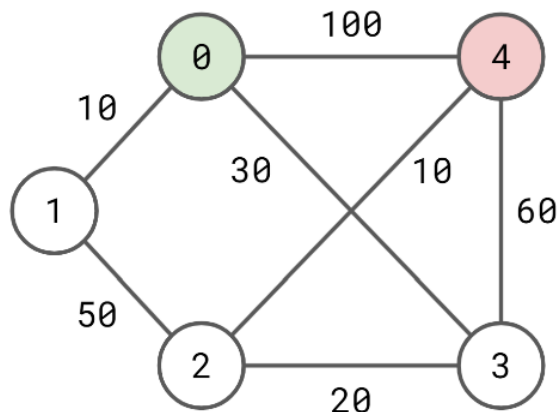


Рис. 192 Неориентированный граф

Нам необходимо найти наименьшее расстояние от вершины 0 до всех остальных вершин с помощью алгоритма Дейкстры. Тогда можно провести следующие рассуждения:

| 0 | 1 | 2 | 3 | 4 |
|---|-----|-----|-----|-----|
| 0 | inf | inf | inf | inf |

| 0 | 1 | 2 | 3 | 4 |
|---|----|-----|----|-----|
| 0 | 10 | inf | 30 | 100 |

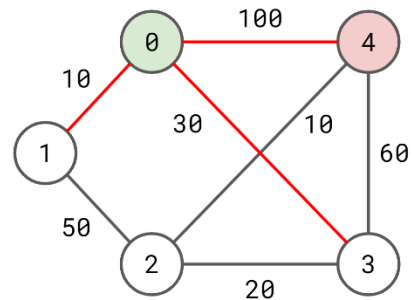
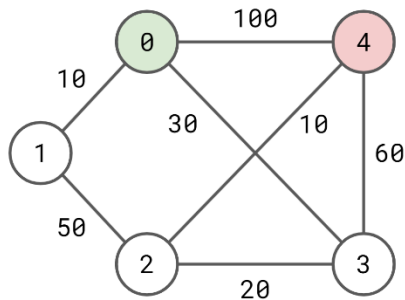


Рис. 193 Пример рассуждения

- 1) полагаем, что расстояние до каждой вершины графа от начальной вершины до начала алгоритма – бесконечность;
- 2) утверждаем, что расстояние от начальной вершины до неё самой – нуль;
- 3) рассматриваем вершину 0 как наиболее приоритетную – она соединена с вершинами 1, 3 и 4 – считаем, что расстояния до них равны весу рёбер с вершиной 0;

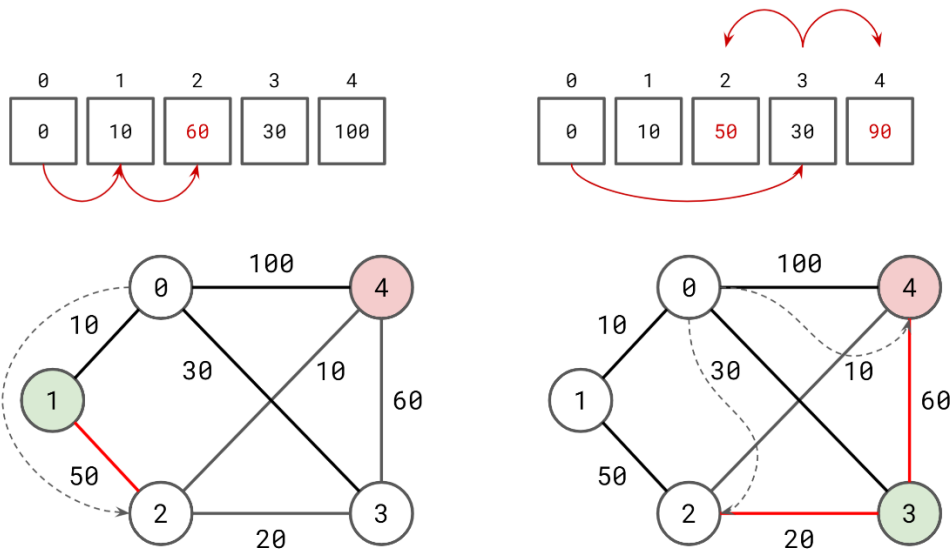


Рис. 194 Пример рассуждения

- 4) рассматриваем вершину 1 как наиболее приоритетную – она соединена с вершиной 2 (вершина 0 не рассматривается, так как из неё уже были проведены расчёты) – считаем, что расстояние до неё равно весу ребра (1, 2) плюс расстояние от вершины 0 до вершины 1;
- 5) рассматриваем вершину 3 как наиболее приоритетную – она соединена с вершиной 2 и 4 – считаем, что расстояние до них минимально возможное из тех, что уже приписаны им, и расстояний, проходящих через вершину 3;

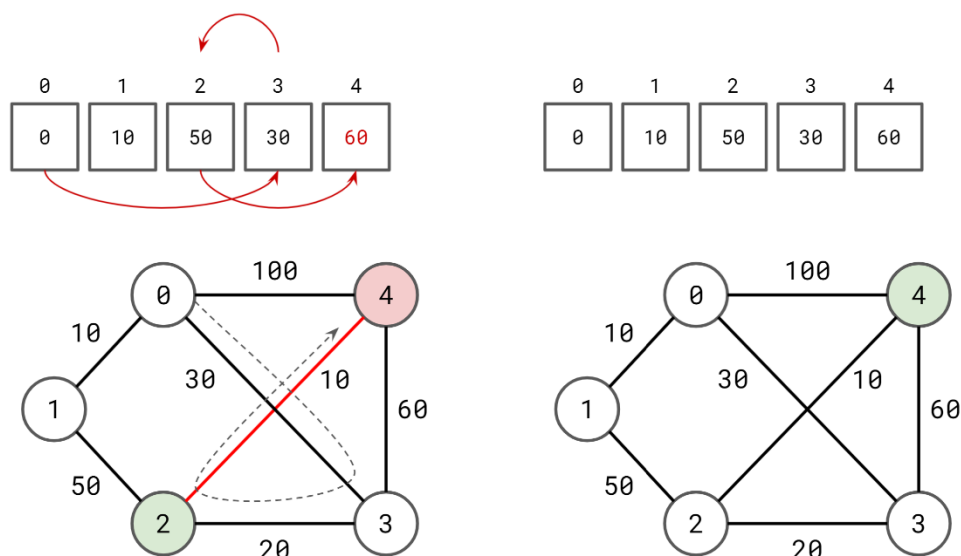


Рис. 195 Пример рассуждения

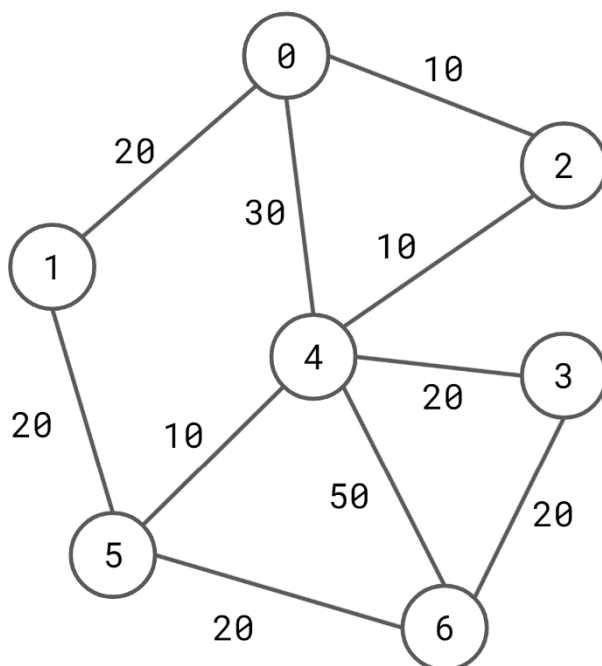
б) проводим аналогичные рассуждения для оставшихся вершин.

Таким образом, полученный список $[0, 10, 50, 30, 60]$ – есть список расстояний до каждой из вершин от вершины 0 изначального графа.

РЕШАЕМ В ТЕТРАДИ

Задание 70.

Найдите самый короткий путь из вершины 0 в вершину 6 в следующем графе. Докажите, что вы получили верный результат без прибегания к перебору всех возможных вариантов.



РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Разбираемый нами алгоритм можно без больших трудностей написать на языке программирования Python:

```
import math
from heapq import heappush # функция добавления пары (ключ,
                             значение)
from heapq import heappop  # функция возвращения пары (ключ,
                             значение)

                             # с минимальным ключом

# ввод количества вершин и рёбер в графе
n, m = tuple(map(int, input().split()))

# структура для хранения графа -- словарь словарей
graph = dict()
for i in range(n):
    graph[i] = dict()

# вводим m троек, описывающих рёбра
for _ in range(m):
    first, second, w = tuple(map(int, input().split()))
    # с учётом того, что граф неориентированный,
    # если существует ребро (a, b), то существует и (b, a)
    graph[first][second] = w
    graph[second][first] = w

dist = [math.inf] * n # считаем изначально, что путь
                       # до каждой вершины -- бесконечность
is_visited = [False] * n # считаем, что ни одна вершина не
                           посещена

# пусть вершиной, из которой мы будем измерять расстояние -- 0
start_vertex = 0
# считаем, что из начальной точки можно попасть в неё за
расстояние 0
dist[start_vertex] = 0

# создаём очередь с приоритетом
# за приоритет отвечает длина пути до вершины
priority_queue = list()

# заполняем очередь с приоритетом парами (расстояние до вершины,
вершина)
for vertex in range(n):
    heappush(priority_queue, (dist[vertex], vertex))
```



```

# проходим по всем вершинам
for _ in range(n):
    # достаём из очереди вершину с наименьшим приоритетом
    this_vertex = heappop(priority_queue)[1]
    # помечаем её как посещённую
    is_visited[this_vertex] = True
    # проходим по всем её неотмеченным соседям
    for vertex_neighbour in graph[this_vertex].keys():
        if not is_visited[vertex_neighbour]:
            prev_dist = dist[vertex_neighbour]
            new_dist = dist[this_vertex] +
graph[this_vertex][vertex_neighbour]
            # предполагаем, что из вершины this_vertex можно
быстрее
            # добраться, чем рассмотренные ранее случаи
            dist[vertex_neighbour] = min(prev_dist, new_dist)
            # добавляем в очередь вершину с новыми приоритетом
            # (с пересчитанным расстоянием)
            heappush(priority_queue, (dist[vertex_neighbour],
vertex_neighbour))

# в списке dist записаны расстояния от start_vertex до каждой i-
ой вершины
print(dist)

```

Обратите внимание, очередь с приоритетом в языке программирования Python можно организовать за счёт использования обычного списка list и модуля heapq, нужные функции из которого кратко описаны в коде программы.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа № 86

Напишите программу, которая для взвешенного неориентированного графа с n вершинами и m рёбрами находит наикратчайший путь от вершины 0 до вершины n , и выводит кроме расстояния так же последовательность вершин, которые входят в данный путь.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Напишите программу с использованием алгоритма Дейкстры для взвешенного графа с n вершинами и m рёбрами, которая проверяла бы,

существуют ли точки на равном расстоянии от вершины 0 и $(n - 1)$, и если существуют, называла бы их.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Как можно найти наикратчайшие расстояния от фиксированной точки в невзвешенном (взвешенном) графе до всех остальных точек?
2. Почему идея поиска в ширину удобнее, нежели идея поиска в глубину, при решении задачи о нахождении наикратчайших расстояний до всех остальных точек невзвешенного графа?
3. Опишите и охарактеризуйте алгоритм Дейкстры.

§8.12 Задача о рюкзаке

ВСПОМИНАЕМ

Давайте вспомним,

- что такое динамическое программирование?
- какие виды динамического программирования вы знаете?

ЧИТАЕМ

На этом занятии мы познакомимся с одной из классических задач динамического программирования, и с некоторыми её вариациями – с *задачей о рюкзаке*.

Условие оригинальной задачи довольно просто: пусть у нас есть N предметов, каждый n_i предмет имеет массу w_i и стоимость p_i , которые должны быть больше нуля. Необходимо ответить на вопрос – какие предметы необходимо взять, чтобы суммарная масса не превосходила W – массы, которую «способен» выдержать рюкзак, и при этом суммарная стоимость выбранных товаров была максимальной.

То есть нам надо выбрать товары таким образом, чтобы

$$\begin{cases} b_1w_1 + b_2w_2 + \dots + b_iw_i + \dots + b_Nw_N \leq W \\ b_1p_1 + b_2p_2 + \dots + b_ip_i + \dots + b_Np_N \rightarrow \max \end{cases}$$

то есть, нам необходимо подобрать такие $b_k \in \{b_1, b_2, \dots, b_N\}$, где b_k – является 1, если мы берём элемент k , иначе – 0.

ОБСУЖДАЕМ

Как вы думаете, какие сложности могут возникнуть в данной задаче, если попробовать решить её перебором? Какова сложность подобного алгоритма?

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа № 87

Напишите переборный алгоритм для решения данной задачи. Помните, что при переборе в данном случае нас интересует только включение или невключение элемента в множество, и нам необходимо проверить все возможные комбинации элементов, то есть перебрать все возможные b_k , для любого $\forall k \in [1, N]$.

Напоминаем, что подобный подход к решению задач затрагивался нами ранее, и вы можете обратиться к предыдущим параграфам.

ЧИТАЕМ

Мы подойдём к решению данной задачи через первичную формализацию. С учётом того, что мы анализируем сразу несколько параметров, стоит выделить основные зависимости: так как нас в первую очередь интересует суммарная стоимость предметов и мы её ищем. Назовём функцию $A(k, s)$ – функцией, определяющей и ставящей в соответствие паре количества предметов k в рюкзаке и допустимому весу s максимально возможную стоимость предметов, которые соответствуют данным величинам и анализируемому набору предметов.

Мы упоминали выше, что данная задача часто рассматривается в качестве примера динамического программирования. Поэтому давайте рассмотрим её решение именно с точки зрения применения идеи динамики – сведения общей задачи к решению аналогичных подзадач. Для этого нам необходимо определить, что будет для нас считаться известными случаями.

Для этого положим, что стоимость предметов в пустом рюкзаке равна нулю. Рюкзак может быть пуст в двух случаях: если в нём не лежит ни одного предмета – $A(0, s)$; или рюкзак не может выдержать никакого веса – $A(k, 0)$. Таким образом мы определили базу динамики.

Когда у нас поднимается вопрос о том, необходимо ли класть очередной предмет в рюкзак, ситуация разделяется на две:

- если мы кладём предмет k в рюкзак, то свободный вес данного рюкзака уменьшается на вес добавляемого предмета, но при этом мы увеличиваем суммарную стоимость предметов в рюкзаке на стоимость добавляемого предмета, то есть $A(k, s) = A(k-1, s-w_k) + p_k$;
- если мы не кладём предмет k в рюкзак, то свободный вес данного рюкзака не изменяется, а количество предметов остаётся равно количеству на один предполагаемый предмет меньше, то есть $A(k, s) = A(k-1, s)$;

Для решения задачи нам достаточно построить матрицу, где изменение одного индекса будет подразумевать изменение веса, а второго – изменение количества предметов в рюкзаке.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Давайте напишем код программы на языке программирования Python для решения данной задачи:

```
weight = int(input()) # вместимость рюкзака
number = int(input()) # количество анализируемых предметов

# списки для хранения веса и цены предметов
obj_weights = [0] * (number + 1)
obj_prices = [0] * (number + 1)

# вводим пары (вес, цена) для каждого предмета
for i in range(number):
    obj_weights[i + 1], obj_prices[i + 1] = tuple(map(int,
input().split()))

# создаём матрицу для хранения A(k, s)
a = list()
for _ in range(number + 1):
    a.append([0] * (weight + 1))

# перебираем все возможные предметы
for k in range(1, number + 1):
    # перебираем все возможные веса
    for s in range(1, weight + 1):
        # если для текущего веса можно добавить предмет
        if s >= obj_weights[k]:
            first = a[k - 1][s]
            second = a[k - 1][s - obj_weights[k]] +
obj_prices[k]
            # то проверяем, стоит ли добавлять очередной предмет
            a[k][s] = max(first, second)
        else:
            # иначе точно не добавляем
            a[k][s] = a[k - 1][s]

# протестируйте на придуманных примерах
print(*a, sep='\n')

# ответ на задачу лежит в a[number][weight]
```

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа № 88

Сконструируйте тесты для проверки верности данного алгоритма. Протестируйте программу.

ЧИТАЕМ

Часто можно встретить ситуацию, когда от программы требуется также указать, какие предметы были положены в рюкзак.

Для этого надо для каждого n_i элемента определить, входит ли он в наш рюкзак, для этого мы можем проанализировать ту таблицу, что уже построили.

Если предмет не входит, мы можем утверждать, что рюкзак с такой же вместимостью, но с меньшим набором допустимых предметов, будет содержать такую же стоимость, то есть $A(i, w) = A(i-1, w)$.

Если же предмет входит, мы можем проделать аналогичные рассуждения и прийти к выводу, что в таком случае $A(i, w) = A(i-1, w-w_i) + p_i$.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Напишем функцию, которая восстановит нам список предметов, которые входят в рюкзак:

```
objects = list() # список предметов в рюкзаке

def solve(k, s):
    # обеспечиваем область видимости
    global a
    global obj_weights
    global objects
    # проверяем крайний случай
    if a[k][s] == 0:
        return
    # проверяем, входит ли предмет в рюкзак
    if a[k][s] == a[k - 1][s]:
        solve(k - 1, s)
    else:
        solve(k - 1, s - obj_weights[k])
        objects.append(k)
```

Дополнив предыдущий пример данным кодом, мы получаем возможность назвать предметы, которые были положены в рюкзак – для этого после формирования матрицы a достаточно запустить данную функцию, после чего в списке `objects` будут лежать номера учтённых предметов.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1

Напишите программу, которая решает классическую задачу о рюкзаке и в качестве ответа выводит список предметов, которые были помещены в рюкзак, и их общую стоимость.

Задание 2

Решите задачу о мультипликативном рюкзаке. Пусть есть n предметов и m рюкзаков. Для каждого рюкзака определена его вместимость – w_i . Предметы описываются парой вес-цена. Задача – выбрать m непересекающихся множеств для помещения в рюкзаки, чтобы суммарная стоимость всех предметов во всех рюкзаках была максимальной.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Сформулируйте условие задачи о рюкзаке.
2. Опишите алгоритм получения максимальной возможной стоимости предметов, которые можно положить в рюкзак.
3. Опишите алгоритм получения номеров предметов, вошедших в рюкзак, если уже была решена задача о получении максимальной стоимости.

§8.13 Численные методы и их сходимость. Точность вычислений.

ЧИТАЕМ

Вам наверняка известен способ нахождения корней квадратного уравнения:

$$ax^2 + bx + c = 0 \quad D = \sqrt{b^2 - 4ac} \quad x_{1,2} = -b \pm \sqrt{D}$$

Но при решении подобной задачи часто можно столкнуться с подкоренными значениями, которые не являются полными квадратами – значит, являются иррациональными числами.

Проблема представления вещественных чисел (рациональных и иррациональных), имеющих бесконечную непериодическую или периодическую дробную часть, в памяти компьютера, связана в первую очередь с вопросом вычислимости и последующей применимости. Для этого используется процедура приближенного вычисления такого значения.

Изучение подобных процедур происходит в рамках сферы знаний численных методов. Под численными методами понимаются методы решения

вычислительных задач с помощью представления всех входных, промежуточных и выходных данных в виде чисел.

Для решения большого количества задач представления чисел и получения результатов различных операций существуют свои численные методы. Вычисление квадратного корня является задачей, для которой существует численный метод – решение данной задачи чаще всего происходит за счёт использования алгоритма, основанного на итерационной формуле Герона. Каждая итерация в рамках данного алгоритма подразумевает вычисление очередного элемента последовательности; каждый элемент данной последовательности есть некоторое приближение к точному результату:

$$x_{n+1} = \frac{1}{2} * \left(x_n + \frac{a}{x_n} \right)$$

В данной формуле

- a – положительное подкоренное значение;
- x_1 – любое положительное число;
- x_i – вычисляемый член последовательности, некоторое

приближение \sqrt{a}

Вне зависимости от выбора значения элемента x_1 ,

$$\lim_{n \rightarrow \infty} x_n = \sqrt{a}$$

формально x_1 влияет только на то, сколько шагов итерации нам придётся сделать.

Давайте попробуем понять, как работает сама формула, предложенная

Героном и почему она действительно стремится \sqrt{a} .

Выберем в качестве x_1 любое положительное число. Если оно больше значения вычисленного корня, то очевидно, что $\frac{a}{x_1}$ будет меньше его. Если x_1 меньше значения корня, то $\frac{a}{x_1}$ – больше. Таким образом, при выборе x_1 мы создаем отрезок, содержащий значение корня.

Рассчитываем

$$x_2 = \frac{1}{2} * \left(x_1 + \frac{a}{x_1} \right)$$

, что по своей сути есть середина обозначенного нами выше отрезка с

$$\frac{a}{x_1}$$

концами в точках x_1 и

Очевидно, что данное число будет ближе максимального расстояния до

$$\frac{a}{x_1}$$

результата от концов первоначального отрезка. Нетрудно доказать, что и будет расположен внутри этого отрезка.

Аналогичным образом можем провести рассуждения для x_2 . На каждом шаге мы будем получать вложенный отрезок. И чем больше мы будем производить итераций, тем ближе будем приближаться к результату.

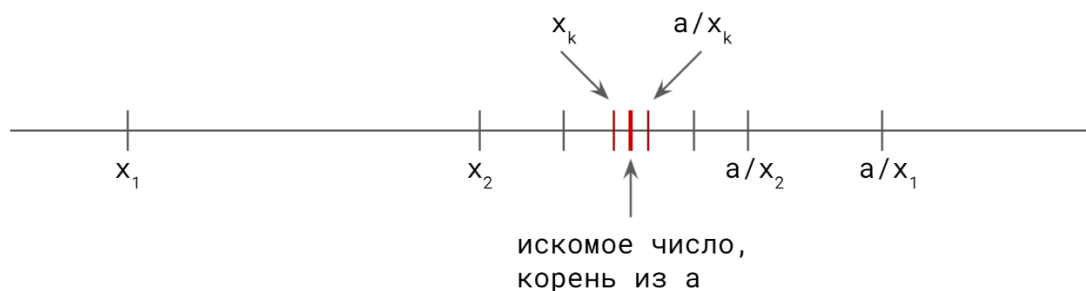


Рис. 196 Схема поиска результата

ОБСУЖДАЕМ

Но в какой момент необходимо остановить выполнения очередной итерации?

ЧИТАЕМ

На этот вопрос можно ответить только через введение понятия погрешности для приближенного числа.

Погрешность – это отклонение значения, измеренного или полученного в результате вычислений, от его истинного значения.

Чаще всего при оценке погрешности вычисления приближенного значения a точного числа x используют идею абсолютной погрешности Δ_a , под которой понимают расстояние между истинным и приближённым значениями, то есть $\Delta_a = |x - a|$.

Исходя из того, что данная формула не применима в нашем случае, так как точного значения числа x при вычислении мы не можем знать, то напрямую используется предельная абсолютная погрешность Δ_a^* – значение, ограничивающее абсолютную погрешность:

$$\Delta_a \leq \Delta_a^*, \text{ то есть } |x - a| \leq \Delta_a^*.$$

Применительно к алгоритму Герона для вычисления значения квадратного корня, мы можем в таком случае утвердить, что исходя из того, что точное значение квадратного корня \sqrt{a} находится в отрезке с концами в точках x_n и $\frac{a}{x_n}$, и мы принимаем x_n как некоторое приближение \sqrt{a} , то абсолютную погрешность данного приближения

$$\Delta_{x_n} = |\sqrt{a} - x_n|$$

мы можем ограничить предельной абсолютной погрешностью равной длине всего отрезка, содержащего точное значение:

$\Delta_a \leq \Delta_a^*$, где

$$\left| \frac{a}{x_n} - x_n \right| \leq \Delta_a^*$$

Таким образом, при знании требуемых ограничений на точность результата, мы можем получить момент остановки – когда x_n будет соответствовать ограничению предельной абсолютной погрешности.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Предположим, нам необходимо вычислить $\sqrt{5}$ с точностью в три знака после запятой.

Точность в три знака после запятой означает, что наше приближение не должно отличаться от истинного значения в первых трех знаках. А это значит, что:

$$|\sqrt{5} - x_n| < 0,001$$

С учётом того, что при получении такого x_n такого, что

$$\left| \frac{a}{x_n} - x_n \right| < 0,001$$

будет выполняться

$$|\sqrt{5} - x_n| \leq \left| \frac{a}{x_n} - x_n \right| < 0,001$$

то есть, указанная точность будет достигнута.

Проведём рассуждения и расчёты с помощью таблицы:

| n | x_n | a | x_{n+1} | $ \frac{a}{x_{n+1}} - x_{n+1} $ | Δ_a^* |
|---|--------------------------|----|-----------|---------------------------------|--------------|
| 1 | 7.0 (произвольное число) | 16 | 3.8571428 | 2.5608465 | 0.001 |
| 2 | 3.8571428 | 16 | 2.5767195 | 0.6362678 | 0.001 |
| 3 | 2.5767195 | 16 | 2.2585856 | 0.0448108 | 0.001 |
| 4 | 2.2585856 | 16 | 2.2361802 | 0.0002244 | 0.001 |

Таким образом мы получили приближение числа $\sqrt{5}$ с точностью до трёх знаков.

Численные методы часто используются в тех ситуациях, когда необходимо вещественное число, которым возможно пользоваться. Так в инженерном деле и в практической физике буквально для расчётов не используются числа из разряда $\pi, e, \sqrt{2}$ – берутся их округления.

Понятно, что компьютер как вычислитель позволяет сильно упростить вычисления для человека. Алгоритм Герона для приближения корня пишется очень просто:

```
import math

STANDART_DELTA = 10 ** -7

def my_sqrt(under_num, delta=STANDART_DELTA, x_i=7):
    # если достигнута заданная точность, возвращаем результат
    if abs(x_i - under_num / x_i) < delta:
        return x_i
    # иначе рассчитываем следующий элемент последовательности
    x_i_1 = (x_i + under_num / x_i) / 2
    # и анализируем его
    return my_sqrt(under_num, delta=delta, x_i=x_i_1)

# можно сравнить результаты
print(my_sqrt(5))
print(math.sqrt(5))
```

Таким образом мы можем вычислить значение любого квадратного корня с неотрицательным элементом с заданной точностью.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1

Протестируйте программу, представленную выше в сравнении с функцией `sqrt()` из модуля `math`. На основе тестирования подберите такую

погрешность для нашего метода, при которой значения работы данных двух функций будут совпадать. И помните про глубину рекурсии.

Задание 2

При знании, что

$$\sqrt[k]{a} = \lim_{n \rightarrow \infty} x_n$$

где

$$x_{n+1} = \frac{1}{k} * ((k - 1) * x_n + \frac{a}{x_n^{k-1}})$$

, напишите программу, рассчитывающую корень k-ой степени из числа a при предельной абсолютной погрешности 10^{-m} .

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Что такое численные методы? Где они применяются?
2. Опишите алгоритм Герона для нахождения значения квадратного корня.
3. Что такое погрешность? Абсолютная погрешность? Предельная абсолютная погрешность?

§8.14 Вычисление суммы степенного ряда

ЧИТАЕМ

Если у нас, благодаря арифметико-логическому устройству, есть базовые арифметические и логические операции, понятно, как получить возведение в степень, но как получить, например, функцию синуса?

В действительности, данную проблему позволяет решить математика. Так, утверждается, что любую функцию $f(x)$ можно разложить в бесконечную сумму степенных функций следующим образом:

$$f(x) = \sum_{n=0}^{+\infty} \frac{f^{(n)}(0)}{n!} x^n$$

В данной формуле под $f^{(n)}(0)$ понимается n -ая производная функции $f(x)$ в точке 0. Подобное разложение называется разложением в ряд Маклорена (частный случай ряда Тейлора).

Большое количество используемых на практике функций уже имеют известное разложение в ряд Маклорена (или Тейлора).

Так, несложно доказать, что функции синуса и косинуса имеют следующие разложения:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} + \dots$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} + \dots$$

Что в общем виде можно записать как:

$$\sin(x) = \sum_{n=0}^{+\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

$$\cos(x) = \sum_{n=0}^{+\infty} (-1)^n \frac{x^{2n}}{(2n)!}$$

Но тут поднимется вопрос, как мы можем вычислить сумму с бесконечным количеством слагаемых? Очевидно, что такого мы сделать не сможем. Но мы можем найти приближённое значение.

Обычно для остановки проверяют значение последнего рассчитанного слагаемого, и если оно меньше предельной абсолютной погрешности, то расчёт останавливают.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Давайте попробуем написать свою функцию расчёта значения функции синуса:

```
import math

DELTA = 10 ** -7

def my_sin(x, delta=DELTA):
    # результат выполнения функции
    res = 0.0
    # номер члена ряда
    n = 0
```

```

while True:
    # вычисляем очередной элемент последовательности
    temp = (x ** (2 * n + 1)) / float(math.factorial(2 * n +
1))

    # добавляем его в сумму с соответствующим знаком
    res += ((-1) ** n) * temp
    n += 1
    if temp < delta:
        return res

```

Подобное решение является работающим, но плохим – на каждом шаге используется возведение в степень и новый расчёт факториала – очень времязатратные операции. Это можно оптимизировать. Идея заключается в запоминании расчётов прошлой итерации.

```

def my_sin(x, delta=DELTA):
    # результат выполнения функции
    res = 0.0
    # номер члена ряда
    n = 1
    # первый член ряда
    x_temp = x
    while True:
        # добавляем очередной член ряда
        res += x_temp
        # рассчитываем следующий член дополняя до него текущий
        x_temp = -x_temp * x * x / ((2 * n) * (2 * n + 1))
        n += 1
        if abs(x_temp) < delta:
            return res

```

Таким образом, члены степенного ряда лучше рассчитывать на основе уже вычисленных слагаемых. Для этого просто необходимо найти их отношение, чтобы узнать природу изменения.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа № 89

Напишите аналогичную функцию для расчёта косинуса.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1

Напишите программу для получения значения функции $f(x) = e^x$, если известно, что

$$e^x = \sum_{n=0}^{+\infty} \frac{x^n}{n!}$$

Напишите программу для получения значения функции $f(x) = \arctg(x)$, если известно, что

$$\arcsin(x) = \sum_{n=0}^{+\infty} \frac{x^{2n+1}}{(2n+1)!} \frac{4^n n!}{(2n+1)}$$

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Что такое ряд Тейлора (Маклорена)?
2. В чём состоит опасность в задаче нахождения суммы степенного ряда?
3. При расчёте сходящегося степенного ряда в какой момент стоит перестать учитывать новые слагаемые?

§8.15 Решение нелинейных уравнений

ВСПОМИНАЕМ

Давайте вспомним,

- что такое нелинейное уравнение?
- что такое корень уравнения?
- что значит решить уравнение?

ЧИТАЕМ

Решение любого уравнения $f(x)$ заключается в нахождении корней, то есть таких значений $x_i \in X$, что $f(x_i) = 0$.

В практической и теоретической деятельности часто могут попасться сложные задачи, связанные с нахождением корней уравнений, которые невозможно решить точно. В таких ситуациях можно говорить только о приближённых значениях. Мы уже говорили о понятии погрешности. И погрешности в данном случае играют очень большую роль.

В рамках текущего урока мы рассмотрим несколько методов нахождения корней в нелинейных уравнениях. Напомним, что линейное уравнение имеет вид $f(x) = ax + c$. Соответственно, уравнения, которые нельзя привести к такому виду, называются нелинейными.

Сама по себе задача приближения корней чаще всего сводима к двум важным этапам:

1) нахождение интервалов, содержащих только одно решение уравнения;

2) уточнение корней на каждом интервале до заданной точности.

Для того, чтобы найти необходимые интервалы, нам необходимо актуализировать две важные теоремы.

Первая теорема довольно проста в восприятии – если некоторая непрерывная функция $f(x)$ на концах отрезка $[a, b]$ принимает значения разных знаков, значит на данном отрезке функция $f(x)$ имеет хотя бы одно решение.

Вторая теорема. Если некоторая непрерывная функция $f(x)$ на концах отрезка $[a, b]$ принимает значения разных знаков, и её производная $f'(x)$ на всём отрезке не меняет своего знака, значит на данном отрезке существует единственное решение.

Основываясь на данных теоремах, очень легко прийти к простой идее, которую чаще всего называют методом дихотомии.

Метод дихотомии (или метод деления пополам) заключается совсем в простой идее – положим, что мы нашли две точки a и b , такие что $f(a)f(b) \leq 0$, иными словами, некоторая функция на данном отрезке имеет хотя бы один корень (теорема 1). Тогда мы можем взять середину этого отрезка

$$c = \frac{a + b}{2}$$

и если $f(a)f(c) \leq 0$, то очевидно, что на отрезке $[a, c]$ имеется хотя бы один корень. Аналогичные рассуждения можно провести и в отношении отрезка $[c, b]$. По сути своей, процесс деления пополам (дихотомии) заключается в том, что каждый следующий шаг позволяет нам уточнить значение корня рассматриваемого уравнения $f(x)$ до нужной нам точности.

Сходимость данного метода линейна, очевидно, что с каждым шагом точность определения корня возрастает в два раза.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Давайте рассмотрим пример использования метода дихотомии.

Пусть у нас есть функция $f(x) = x^2 - 4$. Необходимо найти её корень на отрезке $[0, 10]$. Очевидно, что данная задача легко решается и без метода дихотомии. Она приведена в качестве иллюстрации.

```
# предельная абсолютная погрешность
```

```
DELTA = 10 ** -7
```

```
# создаём исследуемую функцию
```

```
# в данном случае f -- функция, которая при вызове f(x)
```

```
# вернёт x^2 - 4
```

```

f = lambda x: x * x - 4

# в качестве приближенного корня будем считать значение в center
center = -1
# границы отрезка
left = 0
right = 10

# метод дихотомии
while f(right) - f(left) >= DELTA:
    # определяем центр
    center = (right + left) / 2
    # если корень в правой половине
    if f(right) * f(center) <= 0:
        # отбрасываем левую половину
        left = center
    else:
        # иначе отбрасываем правую
        right = center

print(center)

```

ЧИТАЕМ

Аналогичным образом можно решать и другие задачи. Но, очевидно, когда перед вами есть произвольный многочлен, расположение корней которого вы не знаете, то интервал расположения корней необходимо узнавать самостоятельно.

На самом деле есть одна очень полезная теорема.

Третья теорема. Пусть существует функция

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x^1 + a_0$$

тогда, если $b = \max(|a_1|, |a_2|, \dots, |a_n|)$ и $c = \max(|a_0|, |a_1|, \dots, |a_{n-1}|)$, любой корень x $f(x)$

$$\frac{a_0}{|b + a_0|} \leq |x| \leq \frac{c + |a_n|}{|a_n|}$$

Таким образом, теперь мы можем определять интервал расположения корней для любого многочлена.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа № 90

Напишите программу, которая для заданного многочлена

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x^1 + a_0$$

находит интервал расположения корней (теорема 3) и пробует уточнить хотя бы один корень с предельной абсолютной погрешностью $\Delta = 10^{-7}$.

ОБСУЖДАЕМ

Для всех ли многочленов описанный подход нахождения корней будет работать? Можно ли его как-то доработать? Можно ли использовать теорему Безу?

ЧИТАЕМ

Метод простых итераций – ещё один метод численного решения уравнений.

Пусть нам необходимо найти решение уравнения $f(x) = 0$, тогда рассмотрим функцию нахождения приближения корня:

$$x_{i+1} = x_i - \lambda_0 f(x_i)$$

где в качестве λ_0 , при исследовании окрестности корня x , на котором $f(x)$ не меняет своего знака, можно взять константу того же знака, что и $f'(x)$.

При этом, чтобы метод простых итераций сходиллся, необходимо и достаточно, чтобы

$$\forall x \in [a, b] : 0 < \lambda_0 f'(x) < 2$$

Расчёт очередного члена x_{i+1} итерационной последовательности будем продолжать до тех пор, пока не $f(x_{i+1}) > \Delta$.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Давайте рассмотрим пример использования простых итераций.

Пусть у нас есть функция $f(x) = x^2 - 4$. Необходимо найти её корень на отрезке $[0, 10]$.

```
# предельная абсолютная погрешность
DELTA = 10 ** -7
```

```
# создаём исследуемую функцию
# в данном случае f -- функция, которая при вызове f(x)
# вернёт x^2 - 4
f = lambda x: x * x - 4
```

```

# производная f
df = lambda x: 2 * x

# границы отрезка
left = 0
right = 10

# так как df монотонно возрастает,
# возьмем в качестве lambda 2 / df(x_max)
la_0 = 2 / df(right)

# в качестве приближённого корня
# возьмем середину рассматриваемого отрезка
x_this = (left + right) / 2
# рассчитаем следующий элемент последовательности
x_next = x_this - la_0 * f(x_this)

# выполним итерационное приближение
while f(x_next) > DELTA:
    x_this = x_next
    x_next = x_this - la_0 * f(x_this)

print(x_next)

```

ОБСУЖДАЕМ

Подумайте, есть ли какие-то универсальные способы нахождения λ_0 и определения отрезка, содержащего один корень?

ЧИТАЕМ

На самом деле, метод простых итераций, при котором

$$\lambda_0 = \frac{1}{f'(x)}$$

называется методом Ньютона.

Стоит сделать важное замечание, бывают ситуации, что при расчёте x_{n+1} , согласно формуле метода Ньютона

$$x_{n+1} = x_n - \frac{f(x)}{f'(x)}$$

может произойти выход значения x_{n+1} за пределы рассматриваемого отрезка $[a, b]$. В таком случае можно сделать корректировку итерационного члена согласно методу дихотомии (половинного деления).

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1

- 1) Напишите программу поиска производной для многочлена

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x^1 + a_0$$

который задаётся коэффициентами при x .

Задание 2

- 2) Напишите программу для вычисления корней полинома

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x^1 + a_0$$

методом Ньютона, без явного задания интервала расположения корней.

Задание 3

Ознакомьтесь с теоремой Штурма в дополнительных материалах. Напишите программу отделения корней многочлена с помощью метода Штурма. Уточните данные корни с помощью метода Ньютона.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

- 1) Что такое метод дихотомии (метод деления пополам), метод простых итераций, метод Ньютона? Каковы их отличия?
- 2) Как вы считаете, можно ли получить производную алгоритмически, без явного её указания в программе? Для каких функций это сделать легко?

§8.16 Интерполяция

ЧИТАЕМ

При анализе данных в науке (результаты экспериментов, характеристик выборки и т.д.) часто поднимается вопрос анализа и обобщения. Примером такого подхода является метод аппроксимации или приближения.

Так, при анализе дискретных данных появляется задача получения функции, описывающей их. Значимость получения подобной функции заключается в возможности предсказания промежуточных значений.

Такой уточнённый метод аппроксимации называется интерполяцией – нахождение промежуточных значений функции по имеющемуся дискретному набору пар аргумент-значение.

При формальном описании метода интерполяции утверждается, что задачей интерполяции является нахождение интерполяционной функции $F(x)$, для которой верно, что все базовые точки $(x_i, f(x_i))$ некоторой функции $f(x)$ являются решением $F(x_i) = f(x_i)$, при неизвестности $f(x)$. Таким образом можно считать функцию $F(x)$ приближением функции $f(x)$, которая существует для ограниченного количества точек:

$$(x_i, f(x_i)), i \in [1, n] \Rightarrow F(x_i) = f(x_i)$$

Чаще всего в качестве $F(x)$ строят функцию, которая является многочленом – такой подход называется интерполяцией многочленами. Интерполяция многочленами используется при решении из-за своей простоты – существует несколько методов построения интерполяционных многочленов, которые легко реализуются средствами языков программирования.

Давайте рассмотрим один из таких методов – построение интерполяционного многочлена Лагранжа.

Пусть у нас есть набор точек $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$, где для любого

$$\forall i, j \in [0, n] : x_i \neq x_j$$

И нам необходимо построить такой многочлен $L(x)$, что

$$\forall i \in [0, n] : L(x_i) = y_i$$

Тогда утверждается, что данный многочлен можно получить следующим образом:

$$L(x) = \sum_{i=0}^n y_i l_i(x)$$

где:

$$l_i(x) = \prod_{j=0, j \neq i}^n \frac{x - x_j}{x_i - x_j} = \frac{x - x_0}{x_i - x_0} * \frac{x - x_1}{x_i - x_1} * \dots * \frac{x - x_{i-1}}{x_i - x_{i-1}} * \frac{x - x_{i+1}}{x_i - x_{i+1}} * \dots * \frac{x - x_n}{x_i - x_n}$$

Таким образом мы можем найти значение $L(x)$ в любой точке при знании набора $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Давайте напишем функцию для вычисления в конкретной точке значения многочлена Лагранжа, определяемого набором точек (x_0, y_0) , (x_1, y_1) , ..., (x_n, y_n) .

```
def lagrange(x: float, points_list: list) -> float:
    """
    Функция для расчёта значения многочлена Лагранжа  $L(x)$ ,
    построенного по точкам points_list  $\leftarrow (x, y)$ 
    в конкретной точке x

    АРГУМЕНТЫ:
        x : float
            точка, значение функции Лагранжа в которой
            вычисляется
        points_list : list
            список точек, на основе которых строится многочлен
            Лагранжа

    РЕЗУЛЬТАТ
        L(x) : float
            значение многочлена Лагранжа  $L$  в точке x
    """

    # значение многочлена Лагранжа  $L$  в точке x
    lagrange_x = 0.0

    # внешний цикл для перебора всех y
    for i in range(len(points_list)):
        numerator = 1.0 # числитель  $l_i(x)$ 
        denominator = 1.0 # знаменатель  $l_i(x)$ 
        # внутренний цикл для перебора всех x для построения
        l_i(x)
        for j in range(len(points_list)):
            if i != j:
                numerator *= x - points_list[j][0]
                denominator *= points_list[i][0] -
points_list[j][0]

        # добавление рассчитанного слагаемого
        # в записи многочлена Лагранжа
        lagrange_x += points_list[i][1] * numerator /
denominator
```

Таким образом, при вызове данной функции с аргументами x для поиска значения в этой точке $L(x)$ и списка точек $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$, данная функция вернёт нам $L(x)$.

ЧИТАЕМ

Не всегда рассмотрение дискретного набора данных удобно с помощью интерполяции – иногда более общие методы аппроксимации могут быть удобнее.

Так, очень часто пользуются методом наименьших квадратов, который основан на минимизации суммы квадратов отклонений некоторых функций от заданных значений.

Пусть у нас есть набор точек $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ и нам необходимо построить такую линейную функцию

$$y_i^* = ax_i + b$$

отклонение которой от всех y_i будет минимальным, более формально, нам необходимо минимизировать

$$\sum_{i=0}^n (y_i - y_i^*)^2 \rightarrow \min$$

то есть, подобрать такие a и b , что

$$\sum_{i=0}^n (y_i - ax_i - b)^2 \rightarrow \min$$

Утверждается, что подобная сумма будет минимальна в том случае, если

$$\left\{ a = \frac{\sum_{i=0}^n ((x_i - \bar{x})(y_i - \bar{y}))}{\sum_{i=0}^n (x_i - \bar{x})^2} b = \bar{y} - a\bar{x} \right.$$

при этом, под \bar{x} понимается среднее значение x из известных пар, а \bar{y} – среднее значение y .

Таким образом, отталкиваясь от имеющихся пар $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$, мы можем получить коэффициенты линейного уравнения $f^*(x) = ax + b$, что и будет являться приближением функции, описывающей точки из набора $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа № 91

Напишите программу, которая с помощью метода наименьших квадратов находила бы линейное приближение некоторой функции $f(x)$, определяющей набор точек $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

В уроке был разобран линейный метод наименьших квадратов. В общем случае с помощью методов наименьших квадратов можно приближать функции и других классов многочленов, например – квадратичной функции

$$y_i^*(x) = ax^2 + bx + c$$

То есть, задача

$$\sum_{i=0}^n (y_i - y_i^*)^2 \rightarrow \min$$

превращается в задачу поиска значений a, b, c , при которых

$$\sum_{i=0}^n (y_i - ax_i^2 - bx_i - c)^2 \rightarrow \min$$

Для нахождения коэффициентов a, b , и c необходимо решить следующие уравнения:

$$\begin{cases} \frac{\partial}{\partial a} [\sum_{i=0}^n (y_i - ax_i^2 - bx_i - c)^2] = 0 \\ \frac{\partial}{\partial b} [\sum_{i=0}^n (y_i - ax_i^2 - bx_i - c)^2] = 0 \\ \frac{\partial}{\partial c} [\sum_{i=0}^n (y_i - ax_i^2 - bx_i - c)^2] = 0 \end{cases}$$

1. Решите данные уравнения и выразите a, b и c для общего случая.
2. Напишите программу квадратичной регрессии, которая находила бы квадратичное приближение некоторой функции $f(x)$, определяющей набор точек $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Что такое аппроксимация и интерполяция? Опишите их соотношения.
2. Какой метод аппроксимации вы знаете?
3. Какой метод интерполяции вы знаете?
4. В чём существенная разница между методом многочленов Лагранжа и методом наименьших квадратов?

§8.17 Численное интегрирование

ЧИТАЕМ

Численное интегрирование – это множество численных методов, решающих задачу приближённого вычисления определённого интеграла.

Для нас важно вспомнить, что геометрический смысл определённого интеграла

$$\int_a^b f(x) \cdot dx$$

заключается в том, что если

$$\forall x \in [a, b] : f(x) \geq 0$$

то численное значение интеграла есть суть площадь фигуры, ограниченной линиями:

$$y=f(x) \quad y=0 \quad x=a \quad x=b.$$

Так, значение интеграла

$$\int_{-2}^4 (x^2 - x + 4) \cdot dx$$

$\int_{-2}^4 (x^2 - x + 4) \cdot dx$ равно закрашенной области на графике:

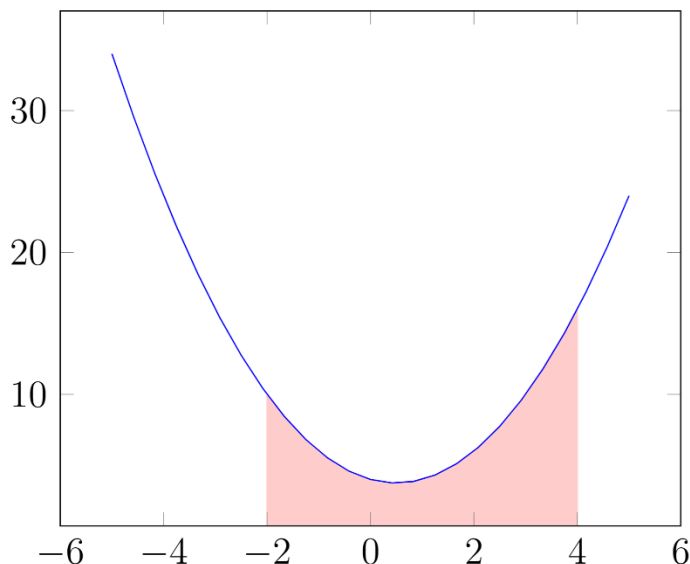


Рис.197 Значение интеграла на графике

При численном интегрировании задача нахождения значения определённого интеграла сводится до нахождения площади замощения более простыми фигурами области, определяемой интегралом:

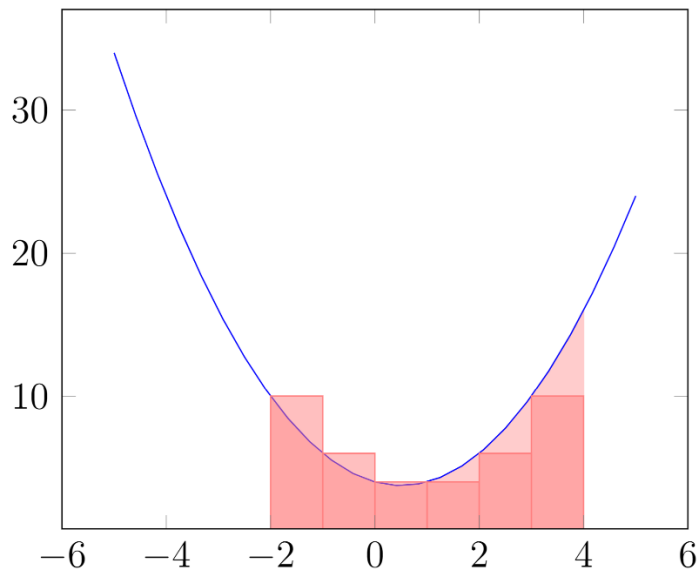


Рис.198 Значение при численном интегрировании

При этом приближение к истинному значению зависит от разбиения описываемой области:

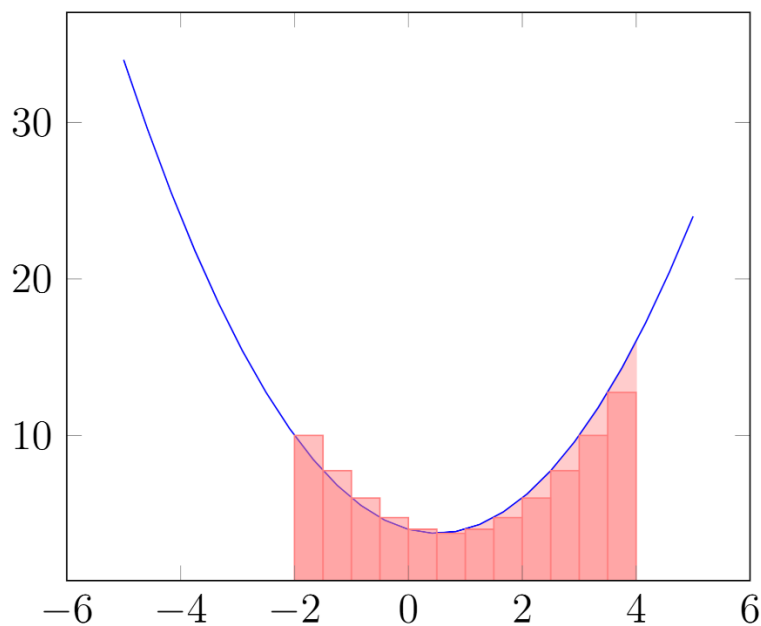


Рис.199 Значение при численном интегрировании

Очевидно, что суммарная площадь прямоугольников во втором случае более приближена к истинному значению площади рассматриваемой криволинейной трапеции.

Иллюстрируемый нами метод разбиения криволинейной трапеции прямоугольниками называется численным интегрированием – методом прямоугольников. Формально данный метод можно описать следующим образом:

$$\int_a^b f(x) \cdot dx \approx \frac{b-a}{n} (y_0 + y_1 + \dots + y_{n-1})$$

где значения $y_0, y_1, \dots, y_{\{n-1\}}, y_n$ – это значения функции $f(x)$ в точках $y_0, y_1, \dots, y_{\{n-1\}}, y_n$, которые в свою очередь делят отрезок $[a, b]$ на n равных отрезков, которые и порождают рассматриваемое нами разбиение.

Описанная нами формула называется формулой левых прямоугольников, так как мы фиксируем левый край прямоугольника. Можно составить аналогичным образом формулу правых прямоугольников:

$$\int_a^b f(x) \cdot dx \approx \frac{b-a}{n} (y_1 + y_2 + \dots + y_n)$$

Но данные подходы к множеству различных задач будут давать в среднем похожую точность.

Чтобы увеличить точность, можно воспользоваться формулой средних прямоугольников, где выбирается серединная точка прямоугольника:

$$\int_a^b f(x) \cdot dx \approx h(f(x_0 + \frac{h}{2}) + f(x_1 + \frac{h}{2}) + \dots + f(x_{n-1} + \frac{h}{2}))$$

где под h понимается

$$h = \frac{b-a}{n}$$

Очевидно, прямоугольники будут давать очень грубое приближение при большом разбиении отрезка. Поэтому можно перейти к замещению площади трапециями, которые чаще всего лучше приближают значение интеграла:

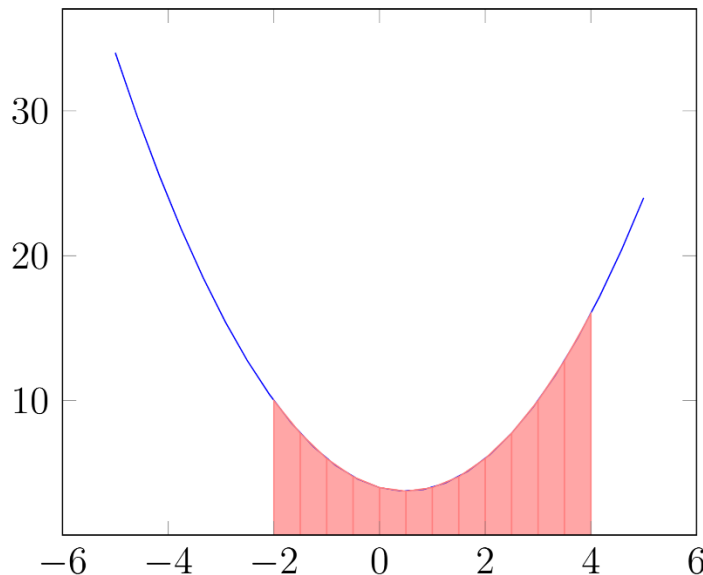


Рис.200 Значение при численном интегрировании

Очевидно, что тогда формулу численного интегрирования методом трапеций можно записать как:

$$\int_a^b f(x) \cdot dx \approx \sum_{i=1}^n \left[\frac{f(x_{i-1}) + f(x_i)}{2} (x_i - x_{i-1}) \right]$$

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Пусть задана

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x^1 + a_0$$

Напишите программу, которая сравнивает результаты нахождения определённого интеграла

$$\int_{x_{left}}^{x_{right}} f(x) \cdot dx$$

с помощью методов прямоугольников (правых, левых, средних) и метода трапеций.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Что такое численное интегрирование?
2. Какие методы численного интегрирования вы знаете?
3. Чем отличаются методы правых прямоугольников, левых прямоугольников, средних прямоугольников?
4. Что такое численное интегрирование методом трапеций?

IX. Парадигмы программирования

§9.1 Процедурно и модульно-ориентированное программирование

ВСПОМИНАЕМ

Давайте вспомним:

1. Что такое программирование?
2. Какие языки программирования вы знаете? Про какие слышали?

ЧИТАЕМ

Порой у новичка в программировании могут возникнуть трудности при переходе с одного языка, который он освоил на определённом уровне, к другому языку программирования – часто это происходит за счёт применения в рамках нового языка качественно других подходов в решении какой-либо классической задачи, чем те, к которым начинающий программист привык в рамках изучения первично рассматриваемого языка программирования.

Так, вы наверняка сможете понять, что написано ниже, но мы полагаем, что это осуществимо только из-за правильного использования названий:

```
my_max :: [Int] -> Int
my_max [x] = x
my_max (x:xs) | (my_max xs) > x = my_max xs
               | otherwise      = x
```

На самом деле это простая функция, написанная на языке программирования Haskell – полностью функциональном языке программирования – которая находит максимальное значение среди элементов некоторого списка чисел.

Внимательно проанализировав данную функцию, даже без знания языка можно догадаться, что она является рекурсивной. В функциональном программировании отсутствует понятие цикла в привычном нам смысле – там цикл заменяется рекурсией. Вот тут подобный подход к решению задач становится в противоречие с тем, как мы привыкли решать задачу поиска максимума.

Подобные ограничения и аналогичные правила составления кода и являются отличительной чертой языков программирования друг от друга.

Наиболее фундаментальные идеи формируют философии языков и неявно конструируют их. Такие идеи являются частью парадигмы программирования.

В науке и философии под парадигмой понимают совокупность концепций и идей, формирующих базу для формирования научной деятельности. Примером изменения парадигмы в науке можно называть

переход научного общества от механики Ньютона к теориям относительности Эйнштейна. Данные концепции кардинально отличаются друг от друга на уровне постулатов.

Понятие парадигмы программирования, по аналогии с философско-научным понятием, можно охарактеризовать как совокупность идей и понятий, определяющих подход к программированию. То есть, как научная парадигма определяет фундамент методологии последующих исследований, так и парадигма программирования определяет инструментарий программиста.

Очень часто можно встретить классификацию парадигм программирования, заключающуюся в разделении на:

- императивную парадигму;
- декларативную парадигму.

Императивный стиль программирования заключается в написании программ как последовательностей действий для достижения желаемого программистом результата.

Декларативный стиль программирования заключается в написании программ через указания, что программист хочет получить.

ПРИМЕР ИМПЕРАТИВНОГО СТИЛЯ

```
num_list: list # дан некоторый список чисел
sqr_list = list() # хотим получить список их квадратов
# описываем, как это сделать
for elem in num_list:
    sqr_list.append(elem * elem)
```

ПРИМЕР ДЕКЛАРАТИВНОГО СТИЛЯ

```
num_list: list # дан некоторый список чисел
# говорим, что хотим получить список их квадратов
sqr_list = list(map(lambda x: x * x, num_list))
```

Таким образом, пример выше хорошо иллюстрирует, что данные подходы могут быть реализованы средствами одного языка программирования, которым может выступать какой-либо современный язык программирования высокого уровня. Но всё же, большинство современных языков программирования оцениваются сообществом как императивные в силу истории их развития и приоритета с точки зрения языковых конструкций в построении самого языка.

ОБСУЖДАЕМ

На самом деле чисто декларативные языки программирования, к которым можно отнести языки функционального (Haskell) и логического

(Prolog) программирования, не получили большого развития и популярности в коммерческой разработке. Как вы думаете, почему?

ЧИТАЕМ

Несмотря на то, что чистые функциональные языки используются редко, они произвели большое влияние на популярные декларативные языки, что мы обсудим в одном из следующих параграфов.

Если обратиться к императивному стилю, то его пространство может представить множество широко используемых парадигм, примерами которых могут стать процедурная парадигма и объектно-ориентированная парадигма, о которых преимущественно пойдёт речь в данной главе.

Под процедурной парадигмой программирования подразумевается подход к программированию, заключающийся в комбинации последовательных команд в единые процедуры (подпрограммы) средствами используемого языка.

Процедура – это комбинация последовательных императивных команд, которую можно использовать как одну инструкцию (команду). Под понятие процедуры часто относят в качестве включения понятие функции.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Давайте попробуем решить какую-нибудь классическую задачу в процедурном стиле. С учётом того, что функция считается процедурой, разделение исходной задачи на подзадачи и создание процедур для решения соответствующих подзадач можно считать процедурным подходом к решению задачи.

Пусть перед нами стоит задача поиска максимума среди списка целых чисел. Мы приведём несколько гипертрофированный вариант процедурного стиля написания кода:

```
# ПОДЗАДАЧИ:
# 1. Организовать ввод данных
# 2. Найти максимальное значение среди чисел
# 3. Вывести найденный максимум

# БЛОК ОПИСАНИЯ ДАННЫХ
n: int # количество чисел в последовательности
arr_list = list() # последовательность чисел
res_max: int # максимальное значение

# БЛОК ОПИСАНИЯ ПРОЦЕДУР
# процедура ввода
def _input():
```

```

global n
global arr_list
n = int(input())
for _ in range(n):
    arr_list.append(int(input()))

# процедура поиска максимума
def _max():
    global n
    global arr_list
    global res_max
    res_max = arr_list[0]
    for i in range(1, n):
        res_max = max(res_max, arr_list[i])

# процедура вывода максимума
def _print():
    print(res_max)

# БЛОК МАНИПУЛЯЦИИ НАД ДАННЫМИ
_input()
_max()
_print()

```

Очевидно, что подобный стиль написания не всегда имеет право на существование. В данном случае он приведён в иллюстративных целях. Также стоит заметить, что не все из описанных процедур написаны логично. Так, процедура `_max` будет более удачно сконструирована, если будет анализировать не глобально определённый список, а список, передаваемый в качестве аргумента, и возвращать высчитанное значение, а не записывать его в глобальную переменную. Также очевидно, что при решении данной задачи создание процедуры вывода значения одной переменной не целесообразно. Так что, вполне логично будет переписать данный код так:

```

def _input():
    n = int(input())
    arr_list = list()
    for _ in range(n):
        arr_list.append(int(input()))
    return arr_list

def _max(arr_list):
    res_max = arr_list[0]

```

```
for i in range(1, len(arr_list)):
    res_max = max(res_max, arr_list[i])
return res_max

# решение задачи с помощью процедур
my_list = _input()
max_my_list = _max(my_list)
print(max_my_list)

# или в одну строчку
print(_max(_input()))
```

Собственно, и первый, и второй случай есть проявление процедурного стиля программирования.

ЧИТАЕМ

Выделение часто используемых последовательностей команд (операций) в отдельные процедуры в какой-то момент стало важным инструментом – когда количество программного кода стало сильно расти. То есть процедурное программирование стало преодолением проблемы увеличения сложности программного кода. Большинство императивных парадигм программирования появились по тому же принципу – с целью решить очередной кризис возрастающей сложности кода.

Так, за процедурным подходом к написанию программ пришла идея *модульного программирования* – программирования, в котором широко используется идея разделения программного кода на модули. Внутри модулей могут быть как отдельные программы, функции, данные или классы (о классах поговорим позже).

Под модулем обычно понимают отдельную (отделённую часть программы), которая объединяет в себе схожие по природе или назначению программные элементы. Понятие инкапсуляции как метода объединения данных и процедур и отделения их от основного кода программы, можно сказать, появилось в широкой практике программирования благодаря модульному подходу к программированию.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Давайте приведём пример модульного подхода при написании программ.

Пусть у нас есть задача, требующая широких математических вычислений. Тогда нам нужны конкретные математические функции и константы.

```
# my_math.py
# описание содержания файла my_math.py

# константы модуля
MY_PI = 3.14
MY_E = 2.71

# функции (процедуры) модуля
def my_abs(num):
    if num < 0:
        return -num
    return num

def my_sqr(num):
    return num * num

def my_factorial(num):
    res = 1
    for i in range(2, num + 1):
        res *= i
    return res
```

Выше описано содержимое файла my_math.py, который мы можем использовать как подключаемый модуль в реализации алгоритмов решения разнообразных (в данном случае – очень узких – задач).

```
# main.py
# описание содержания файла main.py
import my_math # подключение модуля
                # (расположен в одной директории с my_math.py)

# использование содержимого модуля my_math
print(my_math.my_abs(my_math.MY_E - my_math.MY_PI))
```

Таким образом мы можем создавать для своих целей различные модули, код которых не будет загромождать наш основной код.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1

Ранее мы с вами рассматривали численные методы. Численные методы явно и неявно очень часто используются в практической деятельности. Напишите модуль, объединяющий разобранные нами численные методы. Добавьте необходимые константы (например, стандартная абсолютная погрешность) и переменные. Комментируйте свой код.

Задание 2

Составьте модуль, в котором бы были функции, связанные с алгоритмами на графах. В данном случае необходимы функция создания графа (подаются количества вершин и рёбер, после чего возвращается список смежности, описывающий граф), функция проверки связности (через DFS), функция нахождения расстояний от фиксированной точки до всех остальных (BFS).

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Что такое парадигма программирования?
2. Какие парадигмы программирования вам известны?
3. В чём отличие между процедурной парадигмой и модульной? Взаимоисключающие ли они?

§9.2 Основные понятия ООП

ВСПОМИНАЕМ

Давайте вспомним:

1. что такое парадигма программирования?
2. какие парадигмы программирования вам известны?

ЧИТАЕМ

На прошлом уроке мы рассмотрели две очень важные императивные парадигмы программирования: процедурное и модульное программирование. В рамках этого занятия мы познакомимся с основными понятиями объектно-ориентированной парадигмы (объектной парадигмой, стилем).

Под объектно-ориентированной парадигмой (ООП) чаще всего понимается методология программирования, в которой программа есть система взаимодействия объектов, каждый из которых является представителем (экземпляром) некоторого класса.

Чаще всего при формализации понятия ООП выделяют три важных принципа:

- инкапсуляция;
- наследование;
- полиморфизм.

Более детально данные принципы будут рассмотрены несколько позже.

Средством изучения объектно-ориентированного программирования в нашем случае является язык программирования Python. Стоит сказать, что сам по себе язык Python является с точки зрения пользователя мультипарадигмальным, то есть пользователь-программист может писать программы в рамках большинства распространённых парадигм, как это частично было показано в предыдущем уроке. Но при этом по своей внутренней природе (с точки зрения реализации языка) он является чисто объектно-ориентированным, то есть все языковые формы, с которыми взаимодействует программист, есть суть объекты.

Объект является одним из базовых понятий ООП, наравне с классом, основными принципами (наследование, инкапсуляция, полиморфизм), полями, методами и т.д. Давайте чуть лучше попробуем познакомиться с данными понятиями.

Класс, если говорить неформально, это некоторая схема, согласно которой создаются и функционируют объекты. Так, класс определяет наиважнейшие характеристики, которыми должны обладать каждый его представитель, какие действия можно совершать над объектами данного класса (поведение), каким образом создаются объекты (конструкторы). Выделение всех описанных свойств есть процесс абстрагирования, то есть выделения наиважнейших свойств для компьютерной реализации (моделирования) реального объекта.

Давайте попробуем на примере создания класса окружностей в плоскости проиллюстрировать и понять основные понятия объектно-ориентированного программирования.

Давайте выделим главные характеристики окружности при её математическом описании. Чаще всего окружность задаётся тремя величинами – двумя числами, характеризующими расположение центра окружности по двум осям, и третьим числом – радиусом окружности. Данных характеристик достаточно для того, чтобы описать любую окружность в декартовой плоскости.

Описанные нами характеристики называются полями. Каждое поле при создании конкретного экземпляра класса (объекта) должно получить значение – быть проинициализировано.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Так, чтобы создать класс достаточно написать следующие строки:

```
# создание класса Circle
class Circle:
    pass # пустого класса
```

В данном случае мы создали класс Circle, у которого пока нет выделенных характеристик и описанного поведения для его представителей. Давайте обозначим поля данного класса.

```
# создание класса Circle
class Circle:

    # описание полей
    center_x: float # x-координата центра окружности
    center_y: float # y-координата центра окружности
    radius: float # радиус окружности
```

В рамках кода, представленного выше, мы обозначили поля (переменные класса), которые будут описывать конкретные экземпляры класса. На самом деле для программирования на языке Python такая запись не является обязательной – переменные класса можно создать в конструкторе объектов – но подобная запись, в тот же момент времени, является хорошим тоном, который мы рекомендуем вам соблюдать.

Класс без своих экземпляров не имеет причин для существования, поэтому нам необходимо описать правило создания очередного представителя класса (объекта).

```
# создание класса Circle
class Circle:

    # описание полей
    center_x: float # x-координата центра окружности
    center_y: float # y-координата центра окружности
    radius: float # радиус окружности

    # описываем метод создания объекта класса Circle
    def __init__(self, _center_x, _center_y, _radius):
        # говорим, что у создаваемой окружности (self)
        # x-координата центра будет _center_x
        self.center_x = _center_x
        # y-координата центра будет _center_y
        self.center_y = _center_y
        # радиус -- _radius
        self.radius = _radius
```

Обратите внимание, что все функции, реализующие поведение объекта класса (в том числе и конструктор-инициализатор), должны содержать в себе аргумент `self` – данный аргумент говорит, что действие объекта направлено на себя. Когда мы хотим изменить или получить значение поля объекта – для обращения принято писать `self.имя_поля`. При этом все функции, обращающиеся к объекту или изменяющие его внутреннее состояние (значения полей), называются методами.

Для того, чтобы создать объект класса, необходимо вызвать конструктор класса – это делается с помощью указания имени класса и передачи в круглых скобках необходимых аргументов. На самом деле вы это уже много раз делали:

```
# создание экземпляра класса list с аргументом в виде множества
my_list = list({1, 2, 4})
```

Для создания экземпляра класса `Circle` нам необходимо передать три параметра – два числа координат для задания центра, и число определяющее радиус создаваемой окружности (аргумент `self` при вызове конструктора или других методов не передаётся – он используется только в описании методов).

```
# конец описания класса Circle ...
```

```
# создание экземпляра класса Circle
circle = Circle(1, 2, 3) # создание круга
                        # с центром в точке (1, 2)
                        # и радиусом величины 3
```

ЧИТАЕМ

Мы описали основные характеристики экземпляров класса и даже способ их создания. Но это всё ещё не полноценный класс с точки зрения функциональности. Классы обычно создаются для упрощения деятельности программиста с точки зрения написания кода. Так, при создании класса нам необходимо руководствоваться какой-то конкретной задачей. Например, это может быть задача, связанная с большим количеством работы по манипуляции с представителями класса. Какие манипуляции чаще всего проводят с окружностью? Это может быть расчёт длины окружности, проверка на пересечение двух окружностей и т.д.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Давайте реализуем метод нахождения длины окружности – зная формулу длины окружности, это очень просто сделать:

```
def circum_len(self):
    return 2.0 * math.pi * self.radius
```

Аналогично можно создать и другие методы, например – метод нахождения количества точек пересечения двух окружностей:

```
def circles_intersection(self, other_circle):
    # рассчитываем расстояние между центрами окружностей
    center_dist = math.sqrt(
        (self.center_x - other_circle.center_x) **
2 +
        (self.center_y - other_circle.center_y) **
2
    )

    # рассматриваем все возможные случаи
    if center_dist > self.radius + other_circle.radius:
        return 0
    elif center_dist == self.radius + other_circle.radius:
        return 1
    elif ... # функция не дописана
```

Данные методы можно применять для уже существующих окружностей:

создание класса Circle

```
class Circle:

    # описание полей
    center_x: float # x-координата центра окружности
    center_y: float # y-координата центра окружности
    radius: float # радиус окружности

    # описываем функцию создания объекта класса Circle
    def __init__(self, _center_x, _center_y, _radius):
        # говорим, что у создаваемой окружности (self)
        # x-координата центра будет _center_x
        self.center_x = _center_x
        # y-координата центра будет _center_y
        self.center_y = _center_y
        # радиус -- _radius
        self.radius = _radius

    # метод для нахождения длины окружности
    def circum_len(self):
        return 2.0 * math.pi * self.radius

    # метод для нахождения количества точек пересечения
```

```

def circles_intersection(self, other_circle):
    # рассчитываем расстояние между центрами окружностей
    center_dist = math.sqrt(
        (self.center_x -
other_circle.center_x) ** 2 +
        (self.center_y -
other_circle.center_y) ** 2
    )

    # рассматриваем все возможные случаи
    if center_dist > self.radius + other_circle.radius:
        return 0
    elif center_dist == self.radius + other_circle.radius:
        return 1
    elif ...

# создание двух экземпляров класса Circle
circle_1 = Circle(0, 3, 2)
circle_2 = Circle(1, 1, 4)

len_circle_1 = circle_1.circum_len() # находим длину первой
окр.
len_circle_2 = circle_2.circum_len() # находим длину второй
окр.

# находим количество точек пересечения первой окр. со второй
inter_points_1 = circle_1.circles_intersection(circle_2)
# находим количество точек пересечения второй окр. с первой
inter_points_2 = circle_2.circles_intersection(circle_1)

# количества точек пересечения должны быть равны
print(inter_points_1 == inter_points_2)

```

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1

Напишите и протестируйте класс, описывающий круги. Данный класс должен содержать три поля (координаты и величину радиуса), конструктор-инициализатор для создания объектов, метод нахождения площади круга и метод нахождения количества точек пересечения (inf, 1, 0).

Задание 2

Напишите и протестируйте класс, описывающий прямоугольники на плоскости. Данный класс должен содержать четыре поля (координаты верхнего левого угла и нижнего правого), конструктор-инициализатор для создания объектов, метод нахождения площади прямоугольника и метод нахождения пересечения двух прямоугольников, возвращающий новый прямоугольник, являющийся пересечением.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Что такое объектно-ориентированная парадигма программирования?
2. Что называется классом, объектом, методом, полем?
3. Какое ключевое слово используется для создания класса в языке программирования Python?
4. Чем отличаются функции и методы?

§9.3 Инкапсуляция, наследование, полиморфизм

ВСПОМИНАЕМ

Давайте вспомним:

1. объектно-ориентированное программирование?
2. что называется классом, объектом?
3. что такое поля и методы?
4. что такое конструктор?

ЧИТАЕМ

На прошлом уроке мы познакомились с базовыми понятиями – класс, объект, поле, метод. В рамках данного параграфа нами будет произведено уточнение основных принципов объектно-ориентированного программирования – инкапсуляция, наследование и полиморфизм.

Инкапсуляция – это принцип в программировании, согласно которому данные и инструменты для манипуляции над ними хранятся вместе. По сути своей понятие класса есть воплощение инкапсуляции, как и понятия модуля, о чём мы говорили ранее. Действительно, согласно правилам организации класса, доступ к значениям переменных класса (полям) может быть произведён только внутри класса. Причём изменение и анализ данных значений происходит за счёт использования методов (инструмент манипуляции над данными). То есть, с самой концепцией инкапсуляции мы с вами уже познакомились на наивном уровне, только затронув понятия класса.

В рамках изучения объектного подхода к программированию мы будем возвращаться к уточнению понятия инкапсуляции.

Полиморфизм – это свойство операции (функции) обрабатывать данные разных типов. Так, например, операция ‘+’ определена в языке программирования как для целых и вещественных чисел, так и для строк – говорят, что операция ‘+’ полиморфна. В ООП, когда говорят о полиморфизме, подразумевается, что объекты разных классов могут быть обрабатываемы схожим образом. Конечно, тут подразумеваются объекты схожей природы, например, плоские геометрические фигуры: круг, квадрат, прямоугольник, треугольник и т.п. Принцип полиморфизма утверждает, что имена методов схожих классов, выполняющих схожий функционал, должны одинаково называться и иметь равное количество аргументов. Так, если абстракция задачи подразумевает под собой нахождение площади фигур-экземпляров описанных выше классов, то методы, находящие данные значения, должны иметь одинаковые имена.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Представленный код ниже есть суть полиморфизм операции (метода) вычисления площади area для фигур разных классов.

```
class Triangle:
```

```
    a: float
```

```
    b: float
```

```
    c: float
```

```
    def __init__(self, _a, _b, _c):
```

```
        self.a = _a
```

```
        self.b = _b
```

```
        self.c = _c
```

```
    def area(self):
```

```
        p = (self.a + self.b + self.c) / 2
```

```
        res = math.sqrt(p * (p - self.a) * (p - self.b) * (p - self.c))
```

```
class Circle:
```

```
    radius: float
```

```
    def __init__(self, _radius):
```

```
        self.radius = _radius
```

```

    def area(self):
        return math.pi * radius * radius

class Rectangle:

    a: float
    b: float

    def __init__(self, _a, _b):
        self.a = _a
        self.b = _b

    def area(self):
        return a * b

# пусть у нас есть список, в который входят
# треугольник, круг и прямоугольник
shape_list = [Triangle(2, 2, 3), Circle(5), Rectangle(3, 7)]
# посчитаем их суммарную площадь
sum_area = 0.0
for shape in shape_list:
    # применяем для каждой фигуры свой метод area
    sum_area += shape.area()

```

ЧИТАЕМ

Схожесть описанных выше классов позволяет нам описать ещё один основной принцип объектно-ориентированного программирования — *наследование*. Под наследованием понимается механизм, согласно которому конкретный класс (дочерний класс) может наследовать свойства (поля и методы) другого класса (класса предка). В таком случае иногда говорят, что дочерний класс является частным случаем класса предка.

Очевидно, что описанные классы геометрических фигур имеют что-то общее — хотя бы то, что они все являются геометрическими фигурами. Давайте попробуем вычленить общие свойства у квадрата и круга.

Очевидно, что и квадрат, и круг на плоскости имеют какое-то расположение, при чём для обоих видов данных фигур верно, что их расположение на плоскости удобно рассматривать, отталкиваясь от их центра. Возьмём данное свойство за их общее.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Тогда, давайте рассмотрим реализацию класса общей плоской фигуры Shape:

```
# реализация абстрактного класса Shape
class Shape:

    # ПОЛЯ. Точки центра абстрактной фигуры Shape
    center_x: float
    center_y: float

    # конструктор класса Shape
    def __init__(self, _center_x, _center_y):
        self.center_x = _center_x
        self.center_y = _center_y

    # возвращает точку центра
    def get_center(self):
        return (self.center_x, self.center_y)
```

Тогда для того, чтобы сказать, что классы Circle и Square являются наследниками класса Shape, при их определении необходимо это указать следующим образом:

```
# класс круга
class Circle(Shape):
    pass

# класс квадрата
class Square(Shape):
    pass
```

При этом мы можем теперь создавать экземпляры данных классов и взаимодействовать с ними как с абстрактными фигурами:

```
square = Square(1, 2)  # создаем абстрактный квадрат
                        # с центром в (1, 2)
circle = Circle(7, 7)  # создаем абстрактный круг
                        # с центром в (7, 7)

print(square.get_center())  # --> (1, 2)
print(circle.get_center())  # --> (7, 7)
```

ЧИТАЕМ

При этом мы можем уточнить особенности этих фигур за счёт добавления оригинальных (по отношению к классу предку) переменных класса и методов. Но в таком случае добавленные свойства будут относиться только к классу-наследнику, и производить манипуляции над ними в отношении класса-предка будет являться ошибкой.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Давайте построим класс абстрактной фигуры заново, добавив туда два метода – метод нахождения площади и метод нахождения периметра, но будем считать, что для абстрактной фигуры они не определены:

```
# реализация абстрактного класса Shape
class Shape:

    # ПОЛЯ. Точки центра абстрактной фигуры Shape
    center_x: float
    center_y: float

    # конструктор класса Shape
    def __init__(self, _center_x, _center_y):
        self.center_x = _center_x
        self.center_y = _center_y

    # возвращает точку центра
    def get_center(self):
        return (self.center_x, self.center_y)

    # метод для получения площади фигуры
    def area(self):
        return None # значение не определено

    # метод для получения периметра фигуры
    def perimeter(self):
        return None # значение не определено

И реализуем класс квадрата:
# класс квадрата
class Square(Shape):

    # ПОЛЯ. Кроме полей абстрактной фигуры,
    # у квадрата должна быть описана сторона
    side: float

    # конструктор квадрата
```

```

def __init__(self, _center_x, _center_y, _side):
    # сначала производим инициализацию квадрата
    # как абстрактной фигуры -- вызываем конструктор предка
    super().__init__(_center_x, _center_y)
    # после чего инициализируем собственные поля квадрата
    self.side = _side

# переопределяем метод получения площади
def area(self):
    return self.side * self.side

# переопределяем метод получения периметра
def perimeter(self):
    return 4 * self.side

```

Обратите внимание, что во время инициализации объекта класса Квадрат происходит обращение к конструктору класса предка `super().__init__(_center_x, _center_y)` – данное действие является обязательным, так как по своей сути, создание конструктора при определении класса Квадрат – есть переопределение. Обращение к конструктору класса предка происходит с целью, чтобы квадрат при создании был инициализирован как абстрактная фигура, то есть имел центр.

Также при определении класса Квадрат происходит переопределение методов получения площади и периметра. Это значит, что при попытке узнать площадь квадрата, мы действительно получим её, а не как в случае с абстрактной фигурой – None.

ВЫПОЛНЯЕМ ЗАДАНИЕ НА КОМПЬЮТЕРЕ

Практическая работа № 92

Аналогичным образом определите класс Круга. Протестируйте работу методов для представителей всех классов.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание 1

Создайте класс полиномиальных функций

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

с полями `n` – степень многочлена, `a_list` – список коэффициентов a_i , с методом численного нахождения корней и методом нахождения значения многочлена в точке x_0 .

Задание 2

Создайте класс квадратичных функций $f(x) = ax^2 + bx + c$, который является классом наследником класса полиномиальных функций. Переопределите конструктор и метод нахождения корней (через дискриминант).

Задание 3

Создайте класс линейных функций $f(x) = ax + b$, который является классом наследником класса полиномиальных функций. Переопределите конструктор и метод нахождения корней.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Что такое инкапсуляция?
2. Что такое полиморфизм?
3. Что такое наследование?
4. Опишите процесс наследования – что такое класс-предок и класс-наследник? Какие свойства при наследовании получает класс-наследник?

§9.4 Элементы функционального программирования в Python

ВСПОМИНАЕМ

Давайте вспомним:

1. что такое парадигмы программирования?
2. какие парадигмы программирования вам известны?

ЧИТАЕМ

Ранее мы упоминали функциональную парадигму программирования. В рамках этого параграфа мы обсудим её чуть подробнее и рассмотрим её влияние на конструкции языка программирования Python.

Под функциональным программированием понимается подход к решению классических задач с помощью вычисления значений функций в математическом их понимании, из чего следует отсутствие понятия переменной как способа хранения данных.

Основные объекты функционального программирования:

- функции высших порядков;
- чистые функции;
- рекурсия.

Чистые функции – это элементарные единицы любой программы, написанной в функциональном стиле. Основной их особенностью является то, что они лишь принимают аргумент и возвращают свой результат.

Функции высших порядков – это функции, способные в качестве аргумента получать не только данные, но и другие функции, а также они могут возвращать функции в качестве результата своей работы.

Рекурсия в функциональном программировании – это основной способ повторения некоторых действий. Формально в языках функционального программирования нет циклов – их роль выполняют рекурсивные функции.

ОБСУЖДАЕМ

Как вы считаете, использовали ли вы какие-либо элементы, свойственные функциональному программированию?

ЧИТАЕМ

Функциональная парадигма сильно повлияла на формирование различных языков программирования, в том числе и Python.

Python, формально говоря, не позволяет писать в чистом функциональном стиле, но обладает некоторыми функциональными инструментами, с которыми вы уже встречались.

Примером может быть функция `map` – она является функцией высшего порядка, так как в качестве аргументов принимает одну функцию и составной тип данных (коллекцию). В прошлых уроках чаще всего вы могли встретить её использования для ввода последовательности элементов в следующем виде:

```
new_list = list(map(int, input().split()))
```

Функция `map` всегда возвращает итерируемый объект, который мы чаще всего воспринимаем как список. Элементы данного объекта получаются путем применения переданной в качестве аргумента в функции относительно элементов второго аргумента – некоторой коллекции объектов. То есть в том смысле, в котором мы использовали данную функцию – это приведение списка переданных данных (вводимых с клавиатуры) к целым или вещественным числам. В примере выше – к целым числам.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

В качестве первого аргумента мы можем передавать написанные нами функции:

```
# функция возведения в квадрат
def sqr(x):
    return x * x
```

```
m_list = [1, 2, 3, 4] # изначальный список чисел
res = list(map(sqr, m_list)) # --> [1, 4, 9, 16]
```

Также часто в качестве аргумента используют lambda-функции. Если говорить упрощённо, то lambda-функции в Python – это анонимные функции (не имеющие имени). Их использование позволяет часто сохранить компактность кода. Обычно lambda-функция выглядит следующим образом

lambda аргументы: значения

Так, с помощью lambda можно существенно проще решить задачу возведения элементов списка в квадрат:

```
m_list = [1, 2, 3, 4] # изначальный список чисел
res = list(map(lambda x: x * x, m_list)) # --> [1, 4, 9, 16]
```

ЧИТАЕМ

В Python есть и другие функции высших порядков, примером чего могут служить:

- filter(function, collection) – функция высшего порядка, которая согласно функции выбора function выбирает элементы из набора collection;
- reduce(function, collection) – функция высшего порядка, которая согласно функции function производит операцию последовательно над всеми элементами последовательности collection.

РАЗБИРАЕМ ПРИМЕР РЕШЕНИЯ ЗАДАЧИ

Очень хорошо работу данных функций можно проиллюстрировать на следующих примерах:

```
from functools import reduce
```

```
# изначальный список
```

```
m_list = [1, 2, 3, 4]
```

```
print(list(filter(lambda x: x % 2 == 0, m_list))) # --> [2, 4]
```

```
print(list(filter(lambda x: x % 3 == 0, m_list))) # --> [3]
```

```
print(reduce((lambda x, y: x + y), m_list)) # --> 10 -- сумма
```

```
print(reduce((lambda x, y: x * y), m_list)) # --> 24 --  
произведение
```


В действительности очень большое количество языков программирования переняли идеи функционального программирования – это произошло в первую очередь из-за удобства.

ВОПРОСЫ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Что такое функциональное программирование?
2. Какие инструменты функционального программирования вы знаете?
3. Какие из этих инструментов представлены в языке программирования Python?

Ответы на задания

Задания в тетради

Задание 1.

1772 x 1181

Задание 2.

65 536

Задание 3.

360; 165

Задание 4.

44,1 кГц; 480 кбит/с; 24 бит

Задание 5.

20 Мбайт

Задание 6.

40 кГц

Задание 7.

C:\Сданные\Млекопитающие\Виды слонов.txt

Задание 8.

C:\Музыка\Хиты\Лучшие\Про_цветы\

Задание 9.

C:\Class\9b\Homework\Program\task.pas

Задание 10.

C:\Olimp\Ivanov\ex.pas

Задание 11

2

Задание 12.

4

Задание 13.

2

Задание 14.

4

Задание 15.

13

Задание 16.

4) = $D^2 + D$

Задание 17.

1) = $A + C$

Задание 18.

2) 1

Задание 19.

1) – 0,7

Задание 20.

40

Задание 21.

Значение: 1156

Ссылка: $\text{=СЧЁТЕСЛИ(FA30:QB375;">=1100")-СЧЁТЕСЛИ(FA30:QB375;">1700")}$

Задание 22.

Пусть А, В, С – первый, второй, третий парень соответственно, тогда возможны варианты А2В1С3 или С1В2А3

Задание 23.

Пусть Ис, Ин, Ф – история, информатика, физика. Тогда возможны варианты Ис1Ин3Ф2 или Ис2Ин1Ф3

Задание 24.

73

Задание 25.

4 часа 30 минут

Задание 27.

134

Задание 28.

у второго игрока, так как 456 кратно 4.

Задание 29.

1а. $S > 17$ 1б. 17 2. 14 и 16 3. 13

Задание 30.

Вероятность того, что игрок угадал дверь с призом, будет $1/100$. Вероятность того, что вы не угадали, и приз окажется за одной из оставшихся девяноста девяти дверей, будет $99/100$. Ведущий открывает 98 из этих невыбранных 99 дверей, причем за всеми 98 — козы. Теперь 98 из 99 дверей справа открыты, а вероятность того, что машина окажется за оставшейся девятой дверью не изменилась и равна $99/100$. Выбор нужно менять.

Если ведущий открывает лишь одну дверь с правой стороны, то вероятность того, что машина окажется за одной из 99 дверей справа, равна $99/100$. Разделим её на количество оставшихся неоткрытыми дверей (98), получим приблизительно 0,0101 и она чуть больше вероятности 0,01 ($1/100$), что первоначально выбранная вами дверь слева окажется с призом. Поэтому всё же предпочтительнее поменять свой выбор, хотя и шансы изменились незначительно.

Задание 31.

Реляционные – Г, Распределённые – В, Сетевые – Б, Локальные – Д,
Иерархические – А.

Задание 31.

1. в; 2. б; 3. г.

Задание 33.

2)

Задание 34.

3)

Задание 35.

Опишите словами действия, выполняемые в запросах:

1. из таблицы sys.databases выбрать все значения трех полей: name, database_id, create_data
2. выбрать все записи из таблицы tAuthors
3. выбрать записи из таблицы tAuthors в порядке уменьшения значения поля AuthorId
4. определить максимальный возраст автора из таблицы tAuthors
5. вставить в таблицу tAuthors новую запись (“Уильям”, “Шекспир”, 51), проверить вставку выводом всех строк таблицы
6. Ответ: обновить запись со значением поля AuthorId равным 6 новыми значениями полей AuthorFirstName, AuthorLastName, AuthorAge, проверить обновление выводом всех строк таблицы
7. Ответ: удалить из таблицы tAuthors запись со значением 5 в ключевом поле AuthorId
8. Ответ: из таблицы tBooks выбрать значения двух полей bookId и BookTitle такие, что по id автора им соответствует значение поля AuthorFirstName “Александр” из таблицы tAuthors
9. Ответ: выбрать значения полей BookId и BookTitle из таблицы tBooks, которые соответствуют одним и тем же автором из таблицы tAuthors, объединяя данные двух таблиц
10. Ответ: выбрать все записи из таблицы tBooks, в которых id автора не соответствует автору с именем Александр или Сергей

Задание 36.

4/7

Задание 37.

4

Задание 38.

Построим новую таблицу контроля бит для данного информационного слова и подсчитаем каждый контрольный бит:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | |
| x | | x | | x | | x | | x | | x | | x | | x | | x | | x | | x | 1 |
| | x | x | | | x | x | | | x | x | | | x | x | | | x | x | | | 2 |
| | | | x | x | x | x | | | | | x | x | x | x | | | | | x | x | 4 |
| | | | | | | | x | x | x | x | x | x | x | x | | | | | | | 8 |
| | | | | | | | | | | | | | | | x | x | x | x | x | x | 16 |

Получим:

| r | s |
|------------------------|---|
| 010111110010 011110011 | |

Видим, что ошибочный бит стоит под номером 19. Разложим число 19 по степеням двойки: $19 = 16 + 2 + 1$. Значит, ошибочные контрольные биты будут стоять на 1, 2 и 16 позиции.

Задание 39.

22.

Задание 40.

3233

Задание 41.

102.

Задание 42.

56.

Задание 43.

3.

Задание 44.

21.

Задание 45.

103.

Задание 46.

58.

Задание 47.

400.

Задание 48.

680 байт.

Задание 49.

80.

Задание 50.

720 байт

Задание 51.

3

Задание 52.

64

Задание 53.

3 бит

Задание 54.

2 бит

Задание 55.

восьмеричный код – 31145

шестнадцатеричный код – 3265.

Задание 56.

00, 1, 13

Задание 57.

100

Задание 58.

10110

Задание 59.

4

Задание 60.

110

Задание 61.

3

Практические работы

Практическая работа №9

54

Практическая работа №9

20 156725

Практическая работа №28

eeddsc

Практическая работа №35

200 первого и 400 второго вида.

Самостоятельные работы

Задание к уроку §1.8

1d, 2f, 3g, 4h, 5b, 6a, 7i, 8j, 9k, 10e, 11c

Задание к уроку §2.2

2; 4; 2;

Задание к уроку §3.2

56

Задание к уроку §3.6

Задание 1.

1. 37 день
2. 1987, 3
3. 281
4. при значениях, больших 1.

Задание 2.

Ответьте на вопросы, выполнив вычислительный эксперимент:

1. около 20
2. около 24 с четвертью лет

Задание 3.

1. В результате этих экспериментов можно увидеть, что к концу месяца масса бактерий стремится к 7236 г.

2. При начальной массе в 18000 г. уже через три дня бактерии погибнут. Вычислительный эксперимент показывает, что существует такой интервал значений начальной массы (от 2764 г. до 17236 г.), при котором в течение некоторого времени масса бактерий стабилизируется на уровне 7236 г. Если же взять начальную массу за пределами этого интервала, то бактерии погибнут.

Задание к уроку §3.7

Задание 1.

70 женских, 80 мужских костюмов.

Задание 2.

подбор параметра - Excel

Файл Главная Вставка Разметка страницы Формулы Данные Рецензирование Вид

Получение внешних данных Обновить все Подключения Свойства Изменить связи Подключения

Сортировка Фильтр Дополнительно Сортировка и фильтр

Текст по столбцам Удалить дубликаты Проверка данных Работа с данными

Минимальное заполнение Консолидация Анализ "что если" Отношения

G2 X ✓ fx =E2-F2

| | A | B | C | D | E | F | G | H |
|----|-------------------|---------------------|---------------|------------------------|-------|-----------------------|---------|---|
| 1 | Количество товара | Цена закупки за шт. | Сумма закупки | Цена реализации за шт. | Доход | Налог на прибыль, 60% | Прибыль | |
| 2 | 125 | 300 | 37500 | 500 | 25000 | 15000 | 10000 | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |
| 6 | | | | | | | | |
| 7 | | | | | | | | |
| 8 | | | | | | | | |
| 9 | | | | | | | | |
| 10 | | | | | | | | |
| 11 | | | | | | | | |
| 12 | | | | | | | | |
| 13 | | | | | | | | |
| 14 | | | | | | | | |
| 15 | | | | | | | | |
| 16 | | | | | | | | |

Количество товара * (цена закупки - цена реализации)

Доход - налог на прибыль

Результат подбора параметра

Подбор параметра для ячейки G2.
Решение найдено.

Подбираемое значение: 10000
Текущее значение: 10000

OK Отмена

Задание 3.

Получение внешних данных Обновить все Подключения Свойства Изменить связи Подключения

Сортировка Фильтр

E2 X ✓ fx =D2*2*C2

| | A | B | C | D | E |
|----|--------------------|-------------------------|------------------|-------------|-----------------|
| 1 | Площадь территории | Высота снежного покрова | Оплата за 0,5 м³ | Объем снега | Оплата дворнику |
| 2 | 100 | 0,14285714 | 35 | 14,28571 | 1000 |
| 3 | | | | | |
| 4 | | | | | |
| 5 | | | | | |
| 6 | | | | | |
| 7 | | | | | |
| 8 | | | | | |
| 9 | | | | | |
| 10 | | | | | |
| 11 | | | | | |
| 12 | | | | | |
| 13 | | | | | |
| 14 | | | | | |

Результат подбора параметра

Подбор параметра для ячейки E2.
Решение найдено.

Подбираемое значение: 1000
Текущее значение: 1000

OK Отмена

Задание к уроку §3.9

| Задание 1 | Задание 2 |
|---|--|
| а) 0,17 б) 0,50 в) 0,0030 г) 0,17 д) 0,0010 е) 60,61% ж) 0,0025 | а) 0,083 б) 0,50 в) 0,0 г) 0,17 д) 0,0 е) 60,65% ж) 0,0017 |

Задание к уроку §3.10

Задание 2.

| Игрок 1 \ Игрок 2 | Орел | Решка |
|-------------------|----------|----------|
| Орел | (-10;10) | (10;-10) |
| Решка | (10;-10) | (-10;10) |

Рассматриваемая игра является антагонистической (выигрыш одного игрока равен проигрышу другого) и может быть сведена к матричной игре, которая полностью задается матрицей выигрышей одного из игроков, например, игрока 1.

Задание 3

| Игрок 1 \ Игрок 2 | Камень | Ножницы | Бумага |
|-------------------|---------|---------|---------|
| Камень | (0; 0) | (1; -1) | (-1; 1) |
| Ножницы | (-1; 1) | (0; 0) | (1; -1) |
| Бумага | (1; -1) | (-1; 1) | (0; 0) |

Задание к уроку §3.11

Задание 1.

При замене знаков в квадрате со стороной 2 на противоположные количество минусов может измениться только на 2 или 4. Следовательно, во всей таблице число, выражающее количество минусов, остаётся всегда нечётным (инвариантное свойство), а потому не может стать равным 0, что и требовалось доказать.

Задание к уроку §3.13

Задание 1.

14

Задание 2

13

Задание 3

20; 22

Диагностические работы

по теме "Графика и мультимедиа"

Вариант 1

1.
12
2.
64
3.
20
4.
8

Практическое задание:

Ответ: 33 1231875

Решение с помощью Python:

```
def F( n ):  
    global s  
    s += 2*n+4  
    if n > 1:  
        s += 3*n - 2  
    return s+F(n-1)+F(n-4)  
    return s  
  
n = 0  
while True:  
    n += 1          # первое значение n = 1  
    s = 0           # нужно обнулять сумму перед каждым вызовом  
    F(n)            # подсчитали сумму  
    if s > 1000000: break; # если нашли, выход из цикла  
print( n, s )
```

Вариант 2

1.
32
2.
16
- 3.

15

4.

32

Практическое задание:

Ответ: 24 2339137

Решение с помощью Python:

```
def F( n ) :  
    global s  
    s+=(4 * n + 1)  
    if n > 1:  
        s+=(2 * n +5)  
        F(n - 1)  
        F(n - 2)  
n = 24  
s = 0  
F(n)  
print( n, s )
```

по теме "Графика и мультимедиа"

Вариант 1

1.

C:\DATA\GR\VIDEO\CIX – WAVE.mp4

2.

4

3.

1.

4.

20.

Практическое задание:

1.

```
print(input().replace("ma", "mama"))
```

2.

```
with open('text.txt','r') as file:
```

```
    n = int(file.readline().strip("\n"))
```

```
    h = list(map(int, file.readline().strip("\n").split(" ")))
```

```

d = []
for i in range(n):
    if h[i]**0.5 == round(h[i]**0.5):
        d.append(h[i])
with open("text2.txt", "w") as file:
    answ = str(d)[1: -1].replace(", ", " ")
    file.write(answ)

```

Вариант 2

1.

Ответ: C:\WIN\MEDIA\AUDIO

2.

2

3.

4

4.

4

Практическое задание:

1.

```

print(' '.join([list(i)[0].upper() + ('').join(list(i)[1:]) for
i in input().split()]])

```

2.

```

with open('text.txt', encoding='utf-8') as file:
    a = file.read()
a = [i for i in a.split('\n') if 'n' in i or 'N' in i]
with open('text1.txt', 'w') as file:
    file.write('\n'.join(a))

```

по теме " Неравномерное кодирование, помехоустойчивые коды "

Вариант 1

1) 138

2) 64

3) 4

4) 7

5) 16

6) 0; 10; 110; 111

Вариант 2

1) 122

2) 170

3) 2400

4) 9

5) Ответ: 19

6) Ответ: 10; 0; 110; 111

