Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Долгих Т. Ф., Мелехов А. П., Полякова Н. М., Романов М. Н., Ширяева Е. В.

Электронное учебное пособие

«Основы программирования. Python 3»

(для направлений подготовки 01.03.02 «Прикладная математика и информатика», 01.03.03 «Механика и математическое моделирование»)

Ростов–на–Дону

2017

Институт математики, механики и компьютерных наук им. И. И. Воровича
ФГАОУВО «Южный федеральный университет»

Пособие подготовлено сотрудниками кафедры вычислительной математики и математической физики Института математики, механики и компьютерных наук им. И. И. Воровича ФГАОУВО «Южный федеральный университет»

Еленой Владимировной Ширяевой,
Максимом Николаевичем Романовым,
Татьяной Фёдоровной Долгих,
Андреем Петровичем Мелеховым,
Натальей Михайловной Поляковой

Работа поддержана Базовой частью государственного задания
Министерства образования и науки РФ № 1.5169.2017/8.9

Современный и быстро развивающийся язык Python является одним из самых простых в изучении и приятным в использовании языков программирования. Может использоваться для программирования в императивном, в объектно-ориентированном и функциональном стиле.

Учебное пособие содержит необходимый теоретический материал, большое количество примеров и заданий. В данной части пособия рассмотрены простые алгоритмы и способы их реализации с помощью языка Python.

Рекомендуется студентам естественных факультетов университета.

© Е. В. Ширяева, М. Н. Романов, Т. Ф. Долгих, А. П. Мелехов, Н. М. Полякова

Электронное пособие подготовлено в системе XeTeX

Макет, компьютерная вёрстка Е. В. Ширяевой

Содержание

1 Python. Начало работы **6**

1.1 Об именах файлов и каталогов 6

1.2 Об именах в программах… 6

2 Первые программы на языке Python **7**

2.1 Справочная информация 7

2.2 Ввод и вывод данных 10

2.3 Реализация линейных алгоритмов 14

2.4 Упражнения 17

2.5 Задачи 21

3 Оператор условного перехода **24**

3.1 Условный оператор с одной ветвью 24

3.2 Условный оператор с двумя ветвями 25

3.3 Тернарная условная операция 27

3.4 Вложенные условные операторы 28

3.5 Каскадные условные инструкции 31

3.6 Задачи 35

4 Операторы циклов **39**

4.1 Оператор цикла с условием 39

4.2 Трассировочная таблица 40

4.3 Работа с цифрами целого числа 41

4.4 Задачи–I 43

4.5 Вычисление бесконечных сумм с помощью циклов с условиями . . . 46

4.6 Задачи-II 52

4.7 Оператор цикла с параметром 54

4.8 Задачи–III 59

5 Функции (часть I) **64**

5.1 Описание функции 64

5.2 Вызов функции 65

5.3 Значения параметров по умолчанию 68

5.4 Задачи 70

6 Строковый тип данных **74**

6.1 Функции и методы строк 75

6.1.1 Метод поиска в строке 75

6.1.2 Метод replace 76
6.1.3 Методы split и join 78
6.2 Задачи 81

7 Одномерные списки 84
7.1 Простейшие операции со списками 84
7.2 Использование списков в качестве параметров подпрограмм 86
7.3 Замена, удаление и вставка элементов 90
7.4 Задачи 94

8 Поиск элемента в списке 99
8.1 Линейный поиск элемента в списке 99
8.2 Задачи-I 100
8.3 Двоичный поиск элемента в массиве 102
8.4 Задачи-II 103

9 Сортировка списков 104
9.1 Сортировка выбором 104
9.2 Сортировка обменом 106
9.3 Сортировка включением 108
9.4 Методы сортировки в Python 109
9.5 Задачи 111

10 Двумерные массивы 112
10.1 Примеры работы с двумерными массивами 112
10.2 Задачи 115

11 Множества 121
11.1 Задание множеств и функция set() 121
11.2 Основные приемы работы с данными множественного типа 122
11.3 Задачи 127

12 Приложение 1. Математический пакет Maple 128
12.1 Начало работы с пакетом Maple 128
12.2 Дифференцирование и интегрирование 137
12.3 Графика в Maple. Элементарное введение 139

13 Приложения 142
13.1 Немного математики 142
13.2 Таблица ASCII-кодов некоторых символов 143
13.3 Системы счисления 145

Список литературы 158

Об авторах 161

Предметный указатель 161

Интерфейс пользователя 162

1 Python. Начало работы

1.1 Об именах файлов и каталогов

Имя файла отвечает всем требованиям, предъявляемым к идентификаторам языка Python (это последовательность латинских букв, цифр и знаков подчеркивания, начинающаяся либо с буквы, либо со знака подчеркивания).

Договоримся называть файлы с программным кодом именами:

Name_S_T.py,

где Name — фамилия студента (латиницей!), S — № раздела, T — № задачи, py — стандартное расширение для файлов Python.

Например, Ivanov_2_11.py — это имя файла студента Иванова, содержащего программный код для решения задачи 2.11.

Договоримся создавать отдельные каталоги для хранения решений задач каждой лабораторной работы. Например, каталог Ivanov_Lab3 должен хранить все программы из лабораторной работы 2.

1.2 Об именах в программах...

Договоримся не использовать два подряд идущих символа нижнего подчеркивания в начале и в конце идентификатора (такие имена следует использовать только так, как написано в документации).

Чтобы случайно не использовать для именования своего идентификатора ключевое слово или стандартный идентификатор рекомендуется в сомнительных случаях посмотреть список ключевых слов и список стандартных идентификаторов. используемого языка программирования.

Печать списка ключевых слов языка Python в интерактивном режиме

```
>>> import keyword
>>> keyword.kwlist              # печать списка
```

Проверка существования ключевого слова языка Python

```
>>> keyword.iskeyword('kotik')
```

Печать списка стандартных идентификаторов языка Python

```
>>> dir(__builtins__)
```

2 Первые программы на языке Python

2.1 Справочная информация

Пример 2.1. При записи арифметических выражений следует учитывать приоритеты операций (см. таблицу ниже).

Некоторые арифметические операции

```
>> 2**10  # возведение в степень
1024
>> -2**6
-64
>> (-2)**6
64
```

Таблица приоритетов (в порядке уменьшения приоритета)

Арифметические операции	Описание
`**`	возведение в степень
`~`	побитовое «НЕ»
`+, -`	унарные плюс и минус (смена знака)
`*, /, %, //`	умножение, деление, остаток, целочисленное деление
`+, -`	сложение и вычитание
`>>, <<`	побитовые сдвиги (вправо и влево)
`&`	побитовое «И»
`^, |`	побитовые «Исключающее ИЛИ» и «ИЛИ»

Пример 2.2. В Python числа можно записывать в системах счисления по основаниям 2, 8, 16 и 10, при этом перед числами указываются префиксы, отвечающие за разные системы счисления (см. таблицу ниже). Результат вычислений записывается в десятичной системе счисления. Десятичный результат можно представить в виде строки, изображающей число в соответствующей системе счисления. Для этого используются функции преобразования `bin()`, `oct()` и `hex()`.

Работа с двоичными числами

```
>> 0b10 + 0b10       # сложение двоичных чисел
4
>> bin(0b10 + 0b10) # использование функции преобразования
'0b100'
```

Целочисленный литерал	Префикс	Пример
десятичные числа	не используется	50
двоичные числа	«0b» или «0B»	0b101
	здесь «0b»: символы «ноль» и «b»	
восьмеричные числа	«0o» или «0O»	0o10
16-ричные числа	«0x» или «0X»	0xdeadbeef

Некоторые встроенные математические функции

Имя функции	Возвращает
`abs(x)`	$\|x\|$, $x \in \mathbb{Z}, \mathbb{R}, \mathbb{C}$
`min(arg1, arg2, ...)`	наименьший из двух или более аргументов.
`max(arg1, arg2, ...)`	наибольший из двух или более аргументов.
`pow(x, n)`	x^n; эквивалент оператора степени: `x**n`.

В языках программирования, в которых нет операции возведения в степень, для вычисления x^n можно использовать экспоненциально-логарифмическую запись $x^n = e^{n \ln x} \quad (n > 0)$.

Пример 2.3. Для вычисления более сложных выражений используются методы, которые часто бывают определены в дополнительных библиотеках. Наиболее востребованной для нас будет библиотека `math`. Перед обращением к возможностям этой библиотеки её следует подключить с помощью команды `import`.

Подключение библиотеки

```
>>> import math
>>> a = math.sqrt(abs(-9))
>>> a
3.0
```

Здесь использованы две функции sqrt() и abs(), но только для sqrt() понадобилось подключение библиотеки math. Обратите внимание на разницу вызовов стандартной функции и библиотечной функции.

Чтобы не писать всё время math можно использовать следующее подключение библиотеки `from math import *`.

Чтобы не писать math

```
>>> from math import *
>>> a = sqrt(abs(-9))
```

Список методов из любой библиотеки можно напечатать командой

`dir(имя библиотеки)`,

а вызвать документацию по отдельному методу позволяет команда

`имя библиотеки.имя метода.__doc__`

Например,

Вызов списка методов и констант библиотеки math

```
import math
print(dir(math))
```

Вызов документации по отдельному методу

```
import math
print(math.exp.__doc__)
```

Описание функции exp()

```
exp(x)

Return e raised to the power of x.
```

Некоторые константы из модуля math

Имя	Возвращает
`pi`	значение числа $\pi \approx 3{,}1415$ (с имеющейся точностью)
`e`	основание натуральных логарифмов $e \approx 2{,}718281$ (с имеющейся точностью)
`inf`	$+\infty$ (значение с плавающей точкой)
`nan`	значение в плавающей точкой «not a number» (NaN) value

Некоторые степенные и логарифмические функции из модуля math

Имя функции	Возвращает
`sqrt(x)`	$\sqrt{x}$, где $x \geqslant 0$
`log(x)`	$\ln x$, где $x > 0$
`log10(x)`	$\lg x$, где $x > 0$
`log(x, b)`	$\log_b x$, где $x, b > 0$, $b \neq 1$
`exp(x)`	e^x. В частности, `exp(1)` позволит получить основание натурального логарифма $e \approx 2{,}71828$

Некоторые тригонометрические функции из модуля math

Имя функции	Возвращает
sin(x)	$\sin x$ (аргумент — радианная мера угла)
cos(x)	$\cos x$ (аргумент — радианная мера угла)
tan(x)	$\mathrm{tg}\, x$ (аргумент — радианная мера угла)
asin(x)	$\arcsin x$ (результат в радианах)
acos(x)	$\arccos x$ (результат в радианах)
atan(x)	$\mathrm{arctg}\, x$ — угол в радианах, удовлетворяющий условию $x = \mathrm{tg}\, y$, где $-\frac{\pi}{2} < y < \frac{\pi}{2}$
degrees(x)	преобразует угол, заданный в радианах, в градусы
radians(x)	преобразует угол, заданный в градусах, в радианы

2.2 Ввод и вывод данных

Пример 2.4 (ввод и вывод строк)**.** Продемонстрируем ввод строк с клавиатуры и вывод их на печать.

Ввод и вывод строк

```
1 print('Ваше имя?')
2 # в переменную name помещается строка, введенная с клавиатуры
3 name = input()
4 print('Здравствуйте, ' + name + '!')
```

Результат работы программы

```
Ваше имя?
Не скажу
Здравствуйте, Не скажу!
```

Комментарии в языке Python начинаются со знака «решётка» «#» (однострочные) или заключаются между тремя одинарными или двойными кавычками.

Пример 2.5 (ввод и вывод чисел)**.** Программу из примера 2.3 запишем не в виде отдельных команд, а в виде отдельного файла. А также слегка модифицируем код.

Отдельный файл с программой

```
1 import math
2 print('Введите целое число')
3 x = input()  # ввод данных с клавиатуры; x - строка
4 x = int(x)   # преобразование строки к целому типу
5 a = math.sqrt(abs(x))
6 print('a =', a)  # вывод результата на печать
```

Результат работы программы

```
Введите целое число
-9
3.0
```

Можно уменьшить число строк программного кода, объединив строки 2–4.

Ввод и преобразование строки в одной инструкции

```
1 import math
2 x = int(input('Введите целое число '))
3 a = math.sqrt(abs(x))
4 print('a =', a)
```

Пример 2.6 (управление выводом)**.** Продемонстрируем использование параметров sep и end оператора печати print(), а также управляющей последовательности для табуляции (см. также таблицу ниже).

Параметры sep, end и табуляция

```
1 a, b = 10, -4   # в Python'е можно делать и так
2 print(a, b)
3 print(a, b, sep = '\t')     # разделитель -- гор. таб-ция
4 print('a =', a, end = '\t') # в конце строки -- гор. таб-ция
5 print('b =', b)
```

Результат работы программы

```
10 -4
10      -4
a = 10  b = -4
```

Управляющие последовательности

Последовательность	Назначение
\\	символ обратного слеша (остается один символ \)
\’	апостроф (остается один символ ‘)
\”	кавычка (остается один символ “)
\n	новая строка (перевод строки)
\a	звонок
\b	забой
\f	перевод страницы
\r	возврат каретки
\t	горизонтальная табуляция
\v	вертикальная табуляция
\другое	не является экранированной последовательностью (символ обратного слеша сохраняется)

Использование форматированного вывода

Общий вид форматированного вывода

```
print('[текст] %[-][кол-во позиций]тип данных' %значение)
```

Параметры, указанные в квадратных скобках, могут отсутствовать.

тип данных — указывается один из возможных типов данных: s для строк, d для целых чисел и g, f, e, E для вещественных чисел;

значение — выводимое значение. Если выводимое значение — выражение, то оно заключается в круглые скобки. Если выводимых значений несколько, то они указываются в круглых скобках через запятую.

Тип general – по возможности 5 знаков после запятой

```
print('%g' %(pow(2, 1/2)))             # 1.41421
print('%g' %(0.0001 * pow(2, 1/2)))    # 0.000141421
print('%g' %(0.00001 * pow(2, 1/2)))   # 1.41421e-05
```

Вывод вещественного числа

```
k = pow(2, 1/2)
print('%d' %k)  # 1
print('%f' %k)  # 1.414214
print('%e' %k)  # 1.414214e+00
print('%E' %k)  # 1.414214E+00
```

Напомним общий вид форматированного вывода

```
print('[текст] %[-][кол-во позиций]тип данных' %значение)
```

[текст] — поясняющий текст;

[-] — число выравнивается по левому краю поля вывода, отведённого под запись числа (иначе, число выравнивается по правому краю);

[кол-во позиций] — количество позиций, выделяемых под вывод числа. Если параметр отсутствует, то поле вывода будет минимально необходимой ширины.

«-» – выравнивание по левому краю

```
k = pow(2, 1/2)
print('%9.2e'  %k)  # _1.41e+00
print('%-9.2e' %k)  # 1.41e+00_
```

Здесь _ — пробел.

Примеры форматированного вывода

Вывод вещественного числа с фиксированной точкой

```
k = pow(2, 1/2)
print('%9.7f' %k)  # 1.4142136
print('%.2f'  %k)  # 1.41
```

Вывод вещественного числа с плавающей точкой

```
k = pow(2, 1/2)
print('%14.7e' %k)     #  1.4142136e+00
print('%14.7e' %(-k))  # -1.4142136e+00
print('%.2e'  %k)      # 1.41e+00
```

Вывод нескольких значений

```
k = pow(2, 1/2)
print('%9.2e; %.4e; %d' %(k, k, k))
```

Результат

```
 1.41e+00; 1.4142e+00; 1
```

2.3 Реализация линейных алгоритмов

Пример 2.7 (вычисление числа π)**.** Вычислить число π по формуле:

$$\pi = 16 \operatorname{arctg} k_1 - 4 \operatorname{arctg} k_2, \tag{2.1}$$

где $k_1 = \frac{1}{5}$, $k_2 = \frac{1}{239}$. В ответе выдавать четыре знака после запятой.

Получение числа π

```
1 import math
2 k1 = 1/5
3 k2 = 1/239
4 Pi2 = 16 * math.atan(k1) - 4 * math.atan(k2)
5 print('Pi2 = %6.4f' %Pi2)
6 print('Pi2 = %6.4f' %math.pi)
```

Тестирование программы: никакого ввода данных с клавиатуры в данной программе не предусмотрено. Поэтому проверим совпадение полученного числа со значением библиотечной константы языка Python `math.pi()` (для этого в код добавлена последняя строка).

Результат работы программы

```
Pi2 = 3.1416
Pi2 = 3.1416_
```

2.1. На самом деле четвертая цифра в числе π это 5 (подчеркнута):

$$\pi \approx 3{,}141\underline{5}9,$$

но в программе произошло округление, так как цифра, следующая за четвертой, — 9.

Пример 2.8 (вычисление по формулам). Даны $x, y \in \mathbb{R}$. Вычислить

$$u = \sqrt{e^{x+y}}, \quad v = -\frac{\ln(e^{x+y} + e - 1)}{2\sqrt{e^{x+y}}} + x. \tag{2.2}$$

Решение. Обозначим $t = e^{x+y}$ — выражение, трижды повторяющееся в (2.2).

Вычисление по формулам

```
import math
x = float(input('x = '))
y = float(input('y = '))
print('x =', '%5.2f' %x)
print('y =', '%5.2f' %y)
t = math.exp(x + y)
u = math.sqrt(t)
v = -math.log(t + math.e - 1) / 2 / u + x
print('u = %5.2f' %u)
print('v = %5.2f' %v)
```

Тестовый пример. Данные для тестового примера нужно подбирать таким образом, чтобы можно было легко посчитать ответ.

Например, легко вычислить значения u и v при $x = -2$, $y = 2$.

Покажем это

$$u = \sqrt{e^{-2+2}} = \sqrt{e^0} = \sqrt{1} = 1 \;\Rightarrow\; \underline{u = 1},$$

$$v = -\frac{\ln(e^{-2+2} + e - 1)}{2\sqrt{e^{-2+2}}} - 2 = -\frac{\ln(e^0 + e - 1)}{2\sqrt{e^0}} - 2 = -\frac{\ln e}{2} - 2 = -\frac{1}{2} - 2 \;\Rightarrow$$

$$\Rightarrow\; \underline{v = -2{,}5}.$$

Результат работы программы

```
x = -2
y = 2
x = -2.00
y =  2.00
u =  1.00
v = -2.50
```

Пример 2.9 (работа с цифрами числа)**.** Записать двузначное число в обратном порядке ($34 \to 43$).

Решение. Известно, что любое десятичное n-значное целое число A можно представить в виде суммы:

$$A = a_{n-1} \cdot 10^{n-1} + a_{n-2} \cdot 10^{n-2} + \ldots + a_2 \cdot 10^2 + a_1 \cdot 10 + a_0, \qquad (2.3)$$

где a_i ($i = 0, \ldots, n-1$) — коэффициенты (целые числа от 0 до 9) при соответствующих степенях числа десять. Например, $273 = 2 \cdot 10^2 + 7 \cdot 10 + 3$.

Для двузначного числа необходимо определить каждую цифру числа и использовать (2.3) в виде

$$\widetilde{A} = a_0 \cdot 10 + a_1,$$

где a_1 — это первая цифра исходного числа (теперь она переносится в конец числа), a_0 — последняя цифра исходного числа.

Для отделения цифр числа используются операции % и //, см. таблицу на стр. 7.

Работа с цифрами числа

```
1 x = int(input())
2 y = (x % 10) * 10 + x // 10
3 print('x =', x)
4 print('y =', y)
```

2.2. Операция // обладает более высоким приоритетом, чем операция сложения, поэтому при определении y выражение x // 10 не взято в скобки. Операции * и % обладают одинаковым приоритетом, но для удобства чтения кода выражение x % 10 заключено в скобки.

2.4 Упражнения

◊ **3.1.** Запишите по правилам языка Python числа (в дробной части периодической дроби указывать 4 цифры):

1) LXVII; 2) $-1,(23)$; 3) $-0,1(6)$; 4) $-2,8 \cdot 10^{-7}$; 5) $\frac{1}{100\,000}$; 6) 10^6.

◊ **3.2.** Запишите выражения по правилам языка Python

$$1)\ \frac{\sqrt{b^2-3ac}}{2a}; \qquad 2)\ \cos^2(x+\pi); \qquad 3)\ \cos(x+\pi)^2; \qquad 4)\ e^{\sqrt{x}}.$$

◊ **3.3.** Перепишите выражения в традиционной математической форме:

1) `math.sqrt(a+2) - pow(a-2, 2) * math.pi;`

2) `a+b / (c+2*a) - (a+b) / c+2*y;`

3) `a*b / (c+d) - (c-d) / b*(a+b);`

4) `math.pi + pow(math.cos((a+b)/2), 2).`

◊ **3.4.** Запишите двумя способами выражение для получения числа e (основание натурального логарифма) на языке Python.

◊ **3.5.** Не используя функцию pow() и операцию **, запишите по правилам языка Python выражения ($x > 0$):

1) x^{-1}; 2) x^{-2}; 3) x^3; 4) $1/x^{-2}$.

◊ **3.6.** Запишите по правилам языка Python выражения ($x > 0$):

1) x^{100}; 2) 2^{1+x}; 3) $x^{\sqrt{2}}$; 4) $\sqrt[3]{1+x}$.

Указание. Задания выполните в двух вариантах:

1) с использованием функции pow();

2) без использованием функции pow() и операции **.

◊ **3.7.** Запишите по правилам языка Python выражения ($x > 0$):

1) $\log_3 x$; 2) $\log_3 e^x$.

Указание. Задания выполните в двух вариантах:

1) с использованием библиотечной функции для $\log_a b$;

2) с использованием только функции $\ln x$.

◊ **3.8.** Запишите по правилам языка Python выражения ($x > 0$):

1) $\lg x^2$; 2) $\ln|x| + \log_5 x + \lg 8$.

Указание. Задания выполните с использованием библиотечных функций для $\log_a b$ и $\lg x$.

◊ **3.9.** Запишите по правилам языка Python выражения:

1) $1{,}5 + |x| + |1{,}5 + x|$; 2) $|x| - 1 + |x - 1|$; 3) $\sqrt{|x| + x^2}$;

4) $\cos^2 x^2$; 5) $x^{-1}\operatorname{tg} x$; 6) $\sin^{-1} x$; 7) $\cos x^{-1}$;

8) $\operatorname{ch} x$; 9) $\operatorname{sh} x$;

10) $\operatorname{arcctg} 10^2$; 11) $\operatorname{arcctg} x^2$; 12) $\arcsin x$; 13) $\arccos x$.

◊ **3.10.** Переменной z присвойте

1) среднее арифметическое значений переменных a, b и c;

2) среднее геометрическое значений переменных a, b и c.

◊ **3.11.** Запишите операторы присваивания для нахождения площади треугольника по двум сторонам a, b и углу g между ними. Рассмотреть случаи, когда g — 1) радианная мера угла; 2) градусная мера угла.

Указание. Операторы запишите в двух вариантах: с использованием соответствующих функций преобразования из модуля math и без их использования.

Напоминание. $S = \frac{1}{2}ab\sin g$; $x^\circ = \frac{\pi x}{180}$ рад.

◊ **3.12.** Дано a. Используя оператор присваивания, неограниченное количество дополнительных переменных и только указанную операцию, получите:

1) $5a$ за 3 операции "+"; 2) $7a$ за 4 операции "+";
3) a^6 за 3 операции "∗"; 4) a^7 за 4 операции "∗";
5) a^8 за 3 операции "∗"; 6) a^{10} за 4 операции "∗".

Например, для вычисления a^5 за три операции умножения можно использовать последовательность операторов присваивания:

```
b = a * a   (a^2);      c = a * b * b   (a^5).
```

◊ **3.13.** Запишите оператор присваивания для вычисления расстояния между двумя точками $A_1(x_1, y_1)$ и $A_2(x_2, y_2)$ по формуле

$$|A_1A_2| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

◊ **3.14.** Определите тип переменных, участвующих в следующих операторах присваивания:

```
1)  m0 = 'Python';        2)  m1 = '';
3)  m2 = True;            4)  m4 = 3.0;
5)  m4 = 3.0;             6)  m5 = -30;
7)  m6 = -3.5e-2;         8)  m7 = 'False';
9)  m8 = 2 * '3';        10)  m9 = '1' + '2';
```

◊ **3.15.** Указажите правильные операторы вывода:

```
1)  print(x, x + 1);      2)  print(7);
3)  print('Cat');        4)  print(math.exp(x));
5)  print(x + 3.14);      6)  print(x + 3,14);
7)  print(math.Pi);       8)  print(x).
```

◊ **3.16.** Какие двоичные числа будут выведены на экран в результате выполнения последовательности операторов

```
print('1101')
print('11', end='')
print('10')
```

В ответе укажите двоичные числа и результат их суммирования.
Результат запишите в системах счисления по основаниям 2, 8, 16 и 10.

◊ **3.17.** Запишите оператор для вывода на экран значений переменных K (K — целое, $0 < K < 1000$) и R (R — вещественное, $0 < R < 100$) в виде:

␣␣ddd␣␣␣␣␣␣dd.ddd_

Здесь d — некоторая десятичная цифра.

Указание. Используйте форматированный вывод чисел.

◊ **3.18.** Что будет выведено в результате выполнения последовательности операторов (считать, что на порядок выделяется 2 разряда):

```
print('Pi =%6.3f' % math.pi)
print('Pi =%10f' % math.pi)
print('e =%9f' % math.exp(1))
print('e =%7.4f' % math.exp(1))
print('5+5 =%10d' % (5+5))
```

◊ **3.19.** Что будет напечатано в результате выполнения последовательности операторов:

```
x, y = 14, 4
print(divmod(x,y))
print(divmod(-x,-y))
print(divmod(-x,y))
print(divmod(x,-y))
```

◊ **3.20.** Что будет напечатано в результате выполнения последовательности операторов:

```
a = 13 // (16 % 7)
b = 32 % a * 3 - 19 % 3 * 2
print(a)
print(b)
```

Расстановкой одной пары скобок во втором операторе присваивания добейтесь, чтобы переменная b приняла значение 1) 2; 2) 12.

◊ **3.21.** Что будет выведено на экран в результате выполнения оператора:

1) `print(math.trunc(5.25) + round(6.75))`

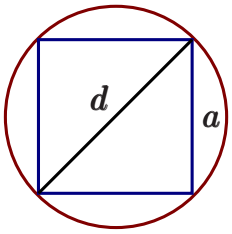
2) `print(round(-17.1) + math.trunc(17.1))`

2.5 Задачи

☎ Все задачи должны быть протестированы на простых примерах.

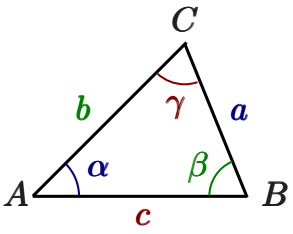
☎ Образец оформления программного кода см. в примере 2.8. Комментарии можно не использовать.

2.1. Известна площадь квадрата S. Найти длину стороны квадрата, диаметр d и площадь S_1 круга, описанного вокруг этого квадрата.



Указание. Для вычисления значений a, d и S_1 (см. рисунок и условие) запишите три оператора присваивания, в правой части которых используйте только площадь квадрата S.

2.2. Даны величины сторон треугольника a, b, c. Найти градусную меру всех углов треугольника.



Указания. 1) Воспользуйтесь теоремой косинусов и найдите косинусы углов.

2) С помощью функции арккосинус найдите углы.

2.3. Определите доход S_n клиента банка, если он разместил в банке сумму S на n лет, а процентная ставка составляет p %.

Указания. а) Для расчёта используйте формулу простых процентов

$$S_n = S\left(1 + \frac{pn}{100}\right).$$

б) Для расчёта используйте формулу сложных процентов

$$S_n = S\left(1 + \frac{p}{100}\right)^n.$$

☎ Формат вывода ответов для заданий 2.4–2.7, связанных с вычислением выражений в разных системах счисления

<Выражение> = <Ответ в указанной системе счисления>

Пример результата работы программы

```
0b111 + 0x49 = 80
0b111 + 0x49 = 0b1010000
0b111 + 0x49 = 0o120
0b111 + 0x49 = 0x50
```

2.4. Найти значение выражения $10101101_2 - 255_8 + \mathrm{D}_{16}$ в десятичной, двоичной, восьмеричной и шестнадцатиричной системах счисления.

Образец оформления ответа см. выше.

2.5. Найти значение выражения $101001_2 + 255_8 - 1\mathrm{E}_{16}$ в десятичной, двоичной, восьмеричной и шестнадцатиричной системах счисления.

Образец оформления ответа см. выше.

2.6. Найти значение выражения $100_8 \cdot 8 + 100_8/8$ в десятичной, двоичной, восьмеричной и шестнадцатиричной системах счисления.

Образец оформления ответа см. выше.

2.7. Найти значение выражения $10_{16} \cdot 16 + 10_{16}/16$ в десятичной, двоичной, восьмеричной и шестнадцатиричной системах счисления.

Образец оформления ответа см. выше.

Указания. Для повторяющихся выражений ввести переменные. Подобрать входные данные так, чтобы можно было легко протестировать программу[1].

2.8. Даны x, y. Вычислить u, v, если:

$$u = \sqrt{e^{x+y}}, \quad v = -\frac{\ln(e^{x+y} + e - 1)}{2\sqrt{e^{x+y}}}.$$

[1] Задачи, выделенные голубым цветом, рассматриваются в теоретической части курса.

2.9 ([14]). Даны x, y, z. Вычислить u, v, если:

$$u = \sin\left|(y - \sqrt{|x|})\left(x - \frac{y}{z^2 + \frac{x^2}{4}}\right)\right|, \quad v = \cos\left(z^2 + \frac{x^2}{4}\right).$$

2.10 ([14]). Даны x, y. Вычислить u, v, если:

$$u = \frac{1 + \sin^2(x+y)}{2 + \left|x - \frac{2x}{1+|\sin(x+y)|}\right|}, \quad v = x - \frac{x^2}{1 + \sin^2(x+y)}.$$

2.11. Найти первую и последнюю цифры заданного трехзначного числа.

2.12. Найти каждую цифру трехзначного числа.

2.13. Найти каждую цифру четырехзначного числа.

2.14. Поменять местами первую и последнюю цифры трехзначного числа (например, $345 \to 543$). Предполагается, что в числе отсутствуют нулевые цифры.

2.15. В трехзначном числе поменять местами цифры, отвечающие за десятки и сотни (например, $345 \to 435$). Предполагается, что в числе отсутствуют нулевые цифры.

2.16. Осуществить циклический сдвиг всех цифр трехзначного числа влево (например, $345 \to 453$). Предполагается, что в числе отсутствуют нулевые цифры.

2.17. Осуществить циклический сдвиг всех цифр трехзначного числа вправо (например, $345 \to 534$). Предполагается, что в числе отсутствуют нулевые цифры.

---

3 Оператор условного перехода

Оператор условного перехода (условный оператор) предназначен для выбора одной из двух альтернативных ветвей алгоритма в зависимости от значения некоторого проверяемого условия. Существуют условные операторы с одной и двумя ветвями.

3.1 Условный оператор с одной ветвью

Синтаксис условного оператора с одной ветвью

```
if условие:
    блок инструкций
```

Здесь условие — некоторое логическое выражение.

Если условие истинно, то выполняется блок инструкций; если же условие ложно, то блок инструкций не будет выполнен и управление передается следующему оператору программы.

В языке Python для выделения блоков инструкций используются отступы. Все инструкции, которые относятся к одному блоку, должны иметь равную величину отступа слева (4 пробела; символ табуляции не рекомендуется).

Простейший пример использования условной инструкции

```
a == -9
if a < 0:
    print('Nice!')  # результат Nice!
```

Пример 3.1 ($y = |x|$)**.** Определить модуль введенного числа, не используя функции abs().

$y = |x|$

```
1 x = int(input('Введите целое число '))
2 y = x
3 if x < 0: y = -x
4 print('|', x, '| = ', y, sep='')
```

Пример 3.2 (поиск наибольшего и наименьшего из двух чисел; if)**.** Ввести два различных целых числа x и y. Присвоить переменной Max большее из этих чисел, а Min — меньшее.

Поиск max(x,y) и min(x,y). I

```
print('Введите 2 числа')
x = int(input())
y = int(input())
Max = x
Min = y
if x < y:
    Max = y
    Min = x
print('max =', Max)
print('min =', Min)
```

Поиск max(x,y) и min(x,y). II

```
print('Введите 2 числа')
x = int(input())
y = int(input())
Max, Min = x, y
if x < y:
    Max, Min = y, x
print('max =', Max)
print('min =', Min)
```

3.2 Условный оператор с двумя ветвями

Синтаксис условного оператора с двумя ветвями

```
if условие:
    блок инструкций 1
else:
    блок инструкций 2
```

Если условие истинно, то выполняется блок инструкций 1, иначе выполняется блок инструкций 2.

Пример 3.3 (поиск наибольшего и наименьшего из двух чисел; if-else)**.** Постановка задачи: см. пример 3.2.

Поиск max(x,y) и min(x,y). III

```
print('Введите 2 числа')
x = int(input())
y = int(input())
if x > y:
    Max, Min = x, y
else:
    Max, Min = y, x
```

Пример 3.4 (проверка числа на чётность)**.** Определить чётность введённого числа.

Проверка числа на чётность. I

```
k = int(input())
s = 'чётное'
if (k % 2 != 0):
    s = 'нечётное'
print(s)
```

Проверка числа на чётность. II

```
k = int(input())
if (k % 2 == 0):
    print('even')  # чётное
else:
    print('odd')   # нечётное
```

Пример 3.5 (проверка принадлежности числа отрезку)**.** Даны целые a, b. Проверить принадлежность введенного числа отрезку $[a, b]$.

Принадлежность числа отрезку

```
a = int(input('a = '))
b = int(input('b = '))
x = int(input('x = '))
if (a <= x <= b):
    print('Да')
else:
    print('Нет')
```

Результат работы программы

```
a = 1
b = 4
x = 3
Да
```

3.3 Тернарная условная операция

Простейшую условную инструкцию

```
if X:
    A = Y
else:
    A = Z
```

можно сократить до одной строки, используя следующую конструкцию:

```
A = Y if X else Z
```

В результате работы инструкции, выполнится выражение Y, если значение X истинно, иначе выполнится выражение Z.

Пример 3.6 (поиск наибольшего из двух чисел; if-else)**.** Найти $\max(x, y)$.

Поиск max(x,y)

```
if x > y:
    Max = x
else:
    Max = y
```

Поиск max(x,y); условная операция

```
Max = x if x > y else y
```

Пример 3.7 (проверка числа на чётность; условная операция)**.** Перепишем программу для проверки чётности числа, приведенную в примере 3.4, с использованием тернарной условной операции.

Проверка числа на чётность; условный оператор

```
if (k % 2 == 0): print('Even')  # чётное
else: print('Odd')   # нечётное
```

Проверка числа на чётность; условная операция. Способ 1

```
print('Even') if (k % 2 == 0) else print('Odd')
```

Проверка числа на чётность; условная операция. Способ 2

```
print('Even' if (k % 2 == 0) else 'Odd')
```

3.1. Тернарная условная операция может использоваться внутри других выражений. Например,

```
R = x + (100 if (k % 2 == 0) else -100)
```

В этом случае условную операцию желательно заключать в скобки.

3.4 Вложенные условные операторы

Алгоритм, реализованный в примерах 3.2 и 3.5, не обрабатывает случая равенства введенных чисел. Рассмотрим более общую задачу.

Пример 3.8 (поиск наибольшего и наименьшего из двух чисел; if-else в if)**.** Ввести два целых числа x и y. Если числа x и y равны, то вывести сообщение об этом, иначе присвоить переменной Max большее из этих чисел, а Min — меньшее.

if-else в if

```
print('Введите 2 числа')
x = int(input())
y = int(input())
if x == y: print('Числа равны')
else:
    if x > y:
        Max = x
        Min = y
    else:
        Max = y
        Min = x
    print('max = ', Max)
    print('min = ', Min)
```

Стиль оформления программного кода. При оформлении кода, содержащего вложенные условные операторы, ключевое слово else выравнивается по тому слову if, к которому оно относится.

Пример 3.9 (if-else в if-else)**.** В зависимости от введенного значения $x \in \mathbb{Z}$ присвоить переменной y следующие значения

$$y = \begin{cases} -50, & \text{если } x < 0; \\ 5, & \text{если } x = 0; \\ 100, & \text{если } x > 0. \end{cases}$$

Решение. Для решения будем использовать вложенный условный оператор.

```
1 x = int(input())
2 if x < 0:
3     y = -50
4 else:  # т.е. x >=0
5     if x == 0:
6         y = 5
7     else:  # т.е. x > 0
8         y = 100
9 print('y = ', y)
```

Опишем процесс выполнения оператора `if` (строки 2–8): сначала проверяется условие `x < 0`. Если оно истинно, переменная y принимает значение -50 (строка 3) и остальная часть оператора `if` (строки 4–8) не выполняется (следующий исполняемый оператор стоит в строке 8). Если первое условие ложно (т. е. $x \geqslant 0$), то управление передается вложенному условному оператору — проверяется второе условие (`x = 0`) и, если оно истинно, то y принимает значение 5 (строка 6), а если ложно, то 100 (строка 8).

Пример 3.10 (if-else в if, if в if-else, if-else в if-else)**.** В следующей паре условных операторов слово `else` ставится в соответствие второму `if`, т. е. оператор `if-else` вложен в оператор `if`.

if-else в if

```
if x == 1:
    if y > x:
        print('x = 1, y > x')
    else:
        print('x = 1, y <= x')
```

Для того чтобы парой слову `else` было первое `if`, необходимо их отступ слева сделать одинаковым:

if в if-else

```
if x == 1:
    if y > x:
        print('x = 1, y > x')
else:
     print('x != 1')
```

В этом фрагменте оператор `if` вложен в оператор `if-else`.

Для наглядности приведем указанные выше операторы в виде блок схем.

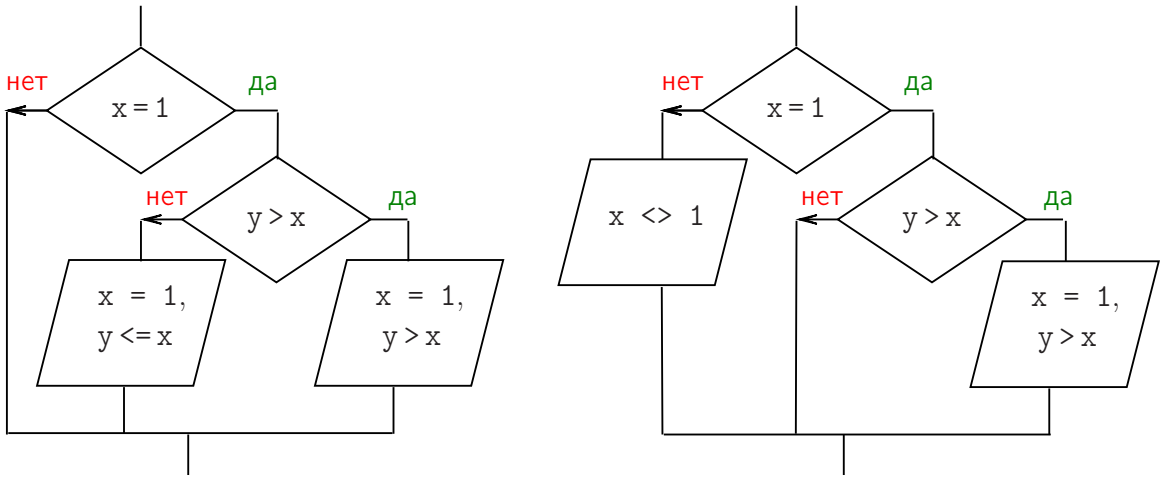


Рис. 1. Блок-схемы (слева if-else в if; справа if в if-else)

if-else in if-else

```
if x == 1:
    if y > x: print('x=1, y>x')
    else: print('x=1, y<=x')
else: print('x<>1')
```

3.5 Каскадные условные инструкции

В каскадной условной инструкции условия `if`, …, `elif` проверяются по очереди. При истинности какого-то из условий выполняется соответствующий блок. Если все проверяемые условия ложны, то выполняется блок `else`, если он присутствует.

Пример 3.11. Приведем примеры использования каскадных условных инструкций.

Какой четверти принадлежит точка (x,y)?

```
x = float(input('x = '))
y = float(input('y = '))
if x > 0 and y > 0: print('I четверть')
elif x < 0 and y > 0: print('II четверть')
elif x < 0 and y < 0: print('III четверть')
elif x > 0 and y < 0: print('IV четверть')
else:
    print('Точка лежит на координатной оси')
    print('или является началом координат.')
```

Где лежит точка?

```
a = int(input())
print('Точка лежит ', end='')
if a < -5:
    print('слева от [-5;5]')
elif a <= 5: # условие a >= -5 выполнилось автоматически
    print('в [-5;5]')
else:
    print('справа от [-5;5]')
```

Решение задачи из примера 3.9

```
x = int(input())
if x < 0:
    y = -50
elif x == 0:
    y = 5
else:
    y = 100
print('y = ', y)
```

Пример 3.12 (вычисление значения по формуле)**.** Даны целые числа a, b. Найти значение выражения

$$y = \frac{\max(a,b)}{a-b} + \sqrt{a+b}. \tag{3.1}$$

Указание. Функцию max() не использовать.

Решение. При вычислениях по формулам необходимо определить области определения всех функций и написать соответствующие условные операторы, чтобы избежать аварийной остановки программы.

В формуле (3.1) следует ожидать возникновения исключительных ситуаций в знаменателе дроби и при вычислении квадратного корня. Знаменатель дроби должен удовлетворять условию $a - b \neq 0$ или $a \neq b$; подкоренное выражение должно быть неотрицательным, т. е. $a + b \geqslant 0$ или $a \geqslant -b$.

Напишем программу двумя способами.

1 способ. Будем вычислять выражение только в случае выполнения обоих условий: $(a \neq b)$ & $(a \geqslant -b)$. Если это невозможно, то сообщать об ошибке (тип ошибки не определять).

I способ (алгоритм)

```
Если (a <> b) и (a >= -b), то
  1. Вычислить максимум.
  2. Посчитать значение выражения по формуле (3).
  3. Напечатать ответ.
иначе (т.е. если либо a = b, либо a < -b)
  Выдать сообщение "Ошибка"
```

Программа в этом случае имеет вид:

I способ

```
1  import math
2  print('Введите целые a, b ')
3  a = int(input('a = '))
4  b = int(input('b = '))
5  if (a != b) and (a >= -b):
6      if a > b: Max = a
7      else:  Max = b
8      y = Max / (a - b) + math.sqrt(a + b)
9      print('y = %6.3f' %y)
10 else: print('Ошибка')
```

2 способ. Будем определять тип исключительной ситуации и сообщать об этом пользователю. Условия теперь проверяются по порядку, но все равно требуемое выражение вычисляется только в случае, если оба условия выполняются.

II способ (алгоритм)

```
Если a = b, то
  если a = 0, то выдать сообщение ”0/0”;
  иначе выдать сообщение ”x=0 в y/x”;
иначе (т.е. если a <> b)
  если a < -b, то выдать сообщение ”x<0 в sqrt(x)”;
  иначе (т.е. если a <> b и a >= -b)
    1. Вычислить максимум;
    2. Вычислить значение выражения по формуле.
    3. Напечатать ответ.
```

II способ

```
1  if a == b:
2      if a == 0: print('0/0')
3      else: print('x=0 в y/x')
4  else:
5      if a < -b: print('x<0 в sqrt(x)')
6      else:
7          if a > b: Max = a
8          else: Max = b
9          y = Max / (a - b) + math.sqrt(a + b)
10         print('y = %6.3f' %y)
```

 **3.2.** Обратим внимание, что максимум здесь вычисляется только в случае необходимости (не перед всеми проверками). Проверки условий всегда следует начинать с простейших.

Тестирование программ проведем для одних и тех же наборов данных. Запишем ожидаемые результаты:

Набор данных	a	b	Ожидаемый результат
№ 1	3	3	Ошибка вычисления, так как знаменатель равен нулю.
№ 2	0	0	Неопределенность: $0/0$.
№ 3	3	-4	Ошибка вычисления, так как подкоренное выражение меньше нуля.
№ 4	3	-3	Ответ: $y = 0{,}5$.
№ 5	4	5	Ответ: $y = -2$.

Приведем результаты работы программ, реализующих алгоритм 1 и алгоритм 2

Набор данных	**Результат работы**	
	программа 1	программа 2
№ 1	Ошибка	x=0 в y/x
№ 2	Ошибка	0/0
№ 3	Ошибка	x<0 в sqrt(x)
№ 4	y = 0.500	y = 0.500
№ 5	y = -2.000	y = -2.000

3.6 Задачи

☎ Образцы оформления программного кода см. в примерах раздела «Оператор условного перехода».

3.1. Найти среди заданных чисел a, b, c, d количество чисел, равных нулю.

3.2. Переменные a, b, c содержат некоторые целые значения. Если значение переменной b неотрицательно, то поменять местами значения переменных a и c. Ответ выдавать в виде:

```
a = ...;  b = ...;  c = ...
```

3.3. Найти среди заданных чисел a, b, c, d количество положительных и количество отрицательных чисел.

3.4. Найти корни квадратного уравнения

$$ax^2 + bx + c = 0,$$

заданного коэффициентами a, b и c ($a \neq 0$). Использовать следующий алгоритм:

если $D < 0$, то выдать сообщение «Уравнение имеет только мнимые корни»;

если $D = 0$, то вычислить корень и выдать результат в виде «$x = \ldots$»;

если $D > 0$, то вычислить корни и выдать результат в виде «$x_1 = \ldots$, $x_2 = \ldots$»;

Указания. Для повторяющихся выражений ввести переменные. Подобрать коэффициенты квадратного уравнения так, чтобы можно было полностью протестировать программу.

3.5. Найти корни биквадратного уравнения

$$ax^4 + bx^2 + c = 0,$$

заданного коэффициентами a, b и c ($a \neq 0$).

Указания. Для повторяющихся выражений ввести переменные. Подобрать коэффициенты биквадратного уравнения так, чтобы можно было полностью протестировать программу.

Указания. Провести тестирование с помощью программы Maple:

1) задать функцию $f(x) = ax^4 + bx^2 + c$;

2) исследовать поведение функции $f(x)$ при различных значениях a, b, c.

3.6 (семинар)**.** Дано $x \in \mathbb{R}$. Присвоить переменной b

1, если ближайшее к значению x целое число — четное и отличное от нуля;
No, в противном случае.

Указания. Задачу решить в трёх вариантах: 1) с использованием оператора if-else; 2) с использованием оператора if; 3) с использованием тернарного оператора.

3.7. Определить является ли введенное целое число двузначным. Если является, то определить последнюю цифру числа.

Указания. Задачу решить в трёх вариантах: 1) с использованием оператора if-else; 2) с использованием оператора if; 3) с использованием тернарного оператора.

3.8. Определить является ли введенное целое число трехзначным. Если является, то выдать две последние цифры числа.

Указания. Задачу решить в трёх вариантах: 1) с использованием оператора if-else; 2) с использованием оператора if; 3) с использованием тернарного оператора.

3.9. Дано натуральное число N. Если число N двузначное, то найти сумму цифр числа. Если число N трехзначное, то найти произведение ненулевых цифр числа.

3.10 (★)**.** Дано целое трехзначное число K. Если цифра числа K, отвечающая за десятки, принадлежит отрезку $[1;5]$, то упорядочить все цифры числа K по убыванию. Например, $326 \to 632$, $476 \to 476$.

3.11. Даны целые числа a, b, c. Найти

1) $\max(a+b+c,\ abc)$;

2) $\min(a^2+b,\ b^2+c)$;

3) $\max(\min(a+b,\ ab),\ \min(b+c,\ bc))$.

Указания. 1) Задания выполнить с использованием тернарного оператора. 2) Для проверки использовать функции `min()` и `max()`. 3) Подобрать тесты для полного тестирования задачи.

3.12. Дано x. Вычислить значение функции

$$1)\ y = \begin{cases} x^3 & \text{при } x \in [-3;3); \\ x^2 & \text{при } x \notin [-3;3); \end{cases} \qquad 2)\ y = \begin{cases} x^2 & \text{при } x \in [0;1); \\ x^{-2} & \text{при } x \notin [0;1). \end{cases}$$

3.13. Дано x. Вычислить значение функции

$$1)\ y=\begin{cases}0 & \text{при } x\leqslant 0;\\ x & \text{при } 0<x\leqslant 1;\\ x^2 & \text{в противном случае.}\end{cases} \qquad 2)\ y=\begin{cases}\lg x & \text{при } x>0;\\ 0 & \text{при } x=0;\\ x^2 & \text{при } x<0.\end{cases}$$

3.14. Даны целые числа a, b, c. Найти значение выражения

$$1)\ \frac{\min(a,b,c)}{\ln a};\quad 2)\ \frac{\max(a,b,c)}{\min(a,5)};\quad 3)\ \frac{\max(a,b,c)}{\min(a,b,c)};\quad 4)\ \frac{\max(a,b,c)}{\operatorname{sign} a}.$$

При выполнении следующих заданий учтите, что

Вещественные числа a и b равны, если выполнено условие

$$|a-b|\leqslant \varepsilon,$$

где ε — некоторое малое число, например, $\varepsilon = 0{,}0001$.

3.15. Дано: $x, y, r \in \mathbb{R}$, $r \geqslant 1$. Выяснить, принадлежит ли точка с координатами (x, y):

1) кругу радиуса r с центром в начале координат;

2) кольцу с центром в начале координат с внешним радиусом $2r$ и внутренним радиусом r.

3.16. Определить, является ли треугольник, заданный координатами вершин $P_1(x_1, y_1)$, $P_2(x_2, y_2)$ и $P_3(x_3, y_3)$, равносторонним.

3.17. Проверить, лежит ли точка $P(x_1, y_1)$ на прямой $y = ax + b$. При положительном ответе найти расстояние от точки P до начала координат; при отрицательном — найти на прямой точку, имеющую такую же ординату, как у точки P.

3.18. Проверить, пересекаются ли прямые

$$a_1x + b_1y + c_1 = 0,\quad a_2x + b_2y + c_2 = 0 \quad \text{и} \quad a_3x + b_3y + c_3 = 0$$

в одной точке, т. е. выполнено ли условие (определитель матрицы равен 0)

$$\Delta_3 = \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix} = 0?$$

3.19. Определить, пересекаются ли парабола $y = cx^2 + dx + f$ и прямая $y = ax + b$. При положительном ответе найти точки пересечения.

3.20. Определить, пересекаются ли параболы $y = ax^2 + bx + c$ и $y = dx^2 + ex + f$. При положительном ответе найти точки пересечения.

3.21. Получится ли забить круглый водосток диаметра d кирпичом, размеры которого a, b, c?

4 Операторы циклов

4.1 Оператор цикла с условием

Синтаксис оператора цикла с условием (while; с предусловием):

```
while условие: оператор
```

При выполнении данного оператора проверяется условие (условие повторения цикла), и если оно соблюдается, то выполняется оператор (оператор цикла). Как только на очередном шаге окажется, что условие не соблюдается, то выполнение оператора цикла прекращается.

Оператор цикла с предусловием может ни разу не выполниться. Тело цикла не выполняется, если условие повторения цикла при первой проверке оказалось ложным.

Пример 4.1 ($a^i < b$, $i = 1, 2, 3, \ldots$; while-do)**.** Даны вещественные числа a ($a > 1$) и b. Получить и вывести на экран все члены бесконечной последовательности a^i, $i = 1, 2, 3, \ldots$ (т. е. a, a^2, a^3,...), меньшие числа b.

Вычисление a^i, $i = 1, 2, 3, \ldots$

```
a = float(input('a (>1) = '))
b = float(input('b = '))
if a < b:
    d = a  # переменная для степеней числа a
    while d < b:
        print('%5.2f' %d)
        d = d * a
else:
    print('Решения нет');
```

При выполнении тела цикла while переменная d последовательно принимает значения a, a^2, a^3,... Значение d будет изменяться до тех пор, пока оно не станет больше или равно значению b. Если $a \geqslant b$ с самого начала, то не будет выведено ни одного члена последовательности a^i, $i = 1, 2, \ldots$

Пусть, например, $a = 2$, $b = 11$. Пошаговую работу программы представим в виде таблицы

d	проверка условия d < 11	печать результата
2	+	2
$2 \cdot 2 = 4$	+	4
$4 \cdot 2 = 8$	+	8
$8 \cdot 2 = 16$	−	

4.2 Трассировочная таблица

Пример 4.2 (a_i, $i = 1, 2, 3, \ldots$; while-do; трассировочная таблица)**.** Дано действительное $b > 0$. Последовательность a_1, a_2,... образована по закону:

$$a_1 = 1, \quad a_i = \frac{a_{i-1}^2}{2} + i - 1 \quad (i = 2, 3, \ldots).$$

Получить первый член последовательности, больший $2b$.

Решение. При написании кода следует четко следовать заданным формулам.

В программах допишем операторы печати для построения трассировочной таблицы. Такие таблицы позволяют осуществлять контроль за исполнением цикла.

Важно. Оператор для вывода ответа должен присутствовать всегда.

Цикл while

```
1  b = float(input('b = '))
2  i = 1
3  a = 1  # инициализация a_1 = 1
4  print('%3d %7.2f %8s' %(i, a, a <= 2*b))   # I строка трассы
5  while a <= 2 * b:
6      i = i + 1                               # изменение счетчика
7      a = a * a / 2 + i - 1                   # вычисление a_i
8      print('%3d %7.2f %8s' %(i, a, a <= 2*b))  # строка трассы
9
10 print('%6.2f' %a) # вывод ответа
```

Результат работы программ

```
b = 8
  1    1.00     True
  2    1.50     True
  3    3.12     True
  4    7.88     True
  5   35.07    False
 35.07
```

4.3 Работа с цифрами целого числа

Пример 4.3 (цифры целого числа; while)**.** Подсчитать количество цифр в записи натурального числа.

Подсчет количества цифр в числе

```
m = int(input())  # исходное число
Kol = 0           # количество цифр
while m > 0:
    Kol = Kol + 1
    m = m // 10
print('Кол. цифр числа', m, '=', Kol)
```

Пример 4.4 (максимальная цифра целого числа; while)**.** Найти максимальную цифру в записи натурального числа.

Поиск максимальной цифры в числе

```
m = int(input())  # исходное число
Maxm = 0          # максимальная цифра
while m > 0:
    d = m % 10    # цифра числа
    if d > Maxm: Maxm = d
    m = m // 10
print(Maxm)
```

Пример 4.5 (сумма чётных цифр целого числа; while)**.** Найти сумму чётных цифр в записи натурального числа.

Сумма чётных цифр в числе

```
m = int(input())    # исходное число
S = 0               # инициализация суммы
while m > 0:
    d = m % 10      # цифра числа
    if d % 2 == 0: # если цифра чётная
        S += d      # суммирование
    m = m // 10
print(S)
```

Пример 4.6 (все цифры целого числа нечётные?; while)**.** Дано натуральное число n. Проверить, верно ли, что в записи числа n все цифры нечётные.

Решение. Приведем два варианта реализации. В I варианте используется вспомогательная переменная логического типа, которая изменяется в цикле, во варианте II в теле цикла происходит только модификация числа.

Все цифры целого числа нечетные? Вариант I

```
n = int(input())
b = (n % 2 != 0)
while (n > 0) and b:
    b = (n % 2 != 0)
    n = n // 10
print(b)
```

Все цифры целого числа нечетные? Вариант II

```
n = int(input())
while (n > 0) and (n % 2 != 0):
    n = n // 10
print(n == 0)
```

4.4 Задачи–I

☎ Образцы оформления программного кода см. в примерах 4.1–4.6.

4.1. Дано действительное $b > 0$. Найти первый отрицательный член последовательности a_1, a_2,..., образованной по закону:

$$a_1 = b, \quad a_i = a_{i-1} - \frac{i}{\sqrt{i-1}} \quad (i = 2, 3, \dots).$$

Указание. В программе должны присутствовать операторы для построения трассировочной таблицы.

4.2. Дано действительное $b < 0$. Найти первый неотрицательный член последовательности a_1, a_2,..., образованной по закону:

$$a_1 = b, \quad a_i = \frac{a_{i-1} + |\sin i|}{i - \sin^2 i} \quad (i = 2, 3, \dots).$$

Указание. В программе должны присутствовать операторы для построения трассировочной таблицы.

4.3. Даны действительные a, h, $x_{нач}$, $x_{кон}$. Вычислить значения функции

$$f(x) = \frac{\sin(x - a^3)}{a^2 + a + 7{,}3}$$

для всех действительных x: $x_{нач} \leqslant x \leqslant x_{кон}$; шаг изменения x равен h.

Найти количество значений $f(x)$, сумму положительных $f(x)$ и произведение отрицательных $f(x)$.

4.4. Дано действительное $x_{нач} \leqslant 0$. Вычислить значения функции

$$f(x) = e^{0.2x} + \sqrt[3]{e^{0.2x}} + \sqrt[5]{e^{0.2x}},$$

если $h \in (0; 1)$ — шаг изменения значения x ($x = x_{нач}$; $x_{нач} + h$; ...). Вычисления проводить до тех пор, пока выполняется неравенство $e^{0.2x} < 5|x_{нач}|$.

4.5 ([14]). Дано положительное a ($a < 1$). Найти наибольшее число вида 2^{-n}, $n \geqslant 0$, меньшее a.

Указание. В программе должны присутствовать операторы для построения трассировочной таблицы.

4.6 ([14]). Дано положительное a ($a < 1$). Найти наименьшее число вида 3^{-n}, $n \geqslant 0$, большее a.

Указание. В программе должны присутствовать операторы для построения трассировочной таблицы.

4.7 ([14]). Дано $a \in \mathbb{R}$. Среди чисел

$$1, \quad 1+\frac{1}{2}, \quad 1+\frac{1}{2}+\frac{1}{3}, \quad \ldots$$

найти первое, большее a.

Указание. В программе должны присутствовать операторы для построения трассировочной таблицы.

4.8. Найти наибольший общий делитель натуральных чисел m и n (обозначение: НОД(m,n)), используя алгоритм Евклида.

4.9. Найти наименьшее общее кратное натуральных чисел m и n, если известно равенство

$$\text{НОК}(m,n) = \frac{mn}{\text{НОД}(m,n)}.$$

Указание. Для нахождения НОК(m,n) используйте алгоритм Евклида.

4.10. Найти наибольший общий делитель трех натуральных чисел m, n и k, используя алгоритм Евклида.

Указание. НОД$(m,n,k) =$ НОД(НОД$(m,n),k)$.

4.11. Дано натуральное число n (количество цифр в числе заранее неизвестно). Проверить, совпадают ли первая и последняя цифры в записи числа n.

4.12. По заданному натуральному числу n найти число m, в записи которого цифры следуют в обратном порядке, например:

$$n = 1234, \quad m = 4321 \quad \text{или} \quad n = 14532, \quad m = 23541$$

4.13. Проверить, является ли натуральное число n простым.

4.14. Даны два целых положительных числа $n > 1$ и $m > n$. Напечатать все простые числа из диапазона $[n,m]$.

4.15. Дано натуральное число n. Проверить, есть ли в записи числа n цифра 5.

4.16. Дано натуральное число n. Проверить, есть ли в записи числа n цифры 1 или 3.

4.17. Дано натуральное число n. Проверить, является ли это число правильной записью 8-ричного числа.

4.18. Дано натуральное число n. Проверить, упорядочены ли цифры числа в порядке убывания (слева направо). Например, 873 (упорядочены по убыванию), 2713 или 379 (не упорядочены по убыванию).

4.19 (*)**.** Дано натуральное число n (количество цифр в числе заранее неизвестно). Проверить, что в записи числа n все цифры разные.

4.20. Разложить натуральное число n на простые множители. (Ответ выдавать в виде «произведения», например, 60 = 2 * 2 * 3 * 5).

4.21. Найти количество простых множителей натурального числа n.

4.22. Найти и вывести на экран все простые делители натурального числа n.

4.23. Вывести на экран все простые делители натурального числа n. Найти количество таких делителей.

4.24. Дано натуральное число (количество цифр в числе заранее неизвестно). Возвести каждую цифру числа в степень, равную разряду цифры. Например, при входном числе 234 ответ: 1 3 4.

Результат работы программы:

```
Дано число 234
4^0=1
3^1=3
2^2=4
```

4.25. Дано целое число (количество цифр в числе заранее неизвестно). Найти сумму чисел, получающихся следующим образом: каждое число равно цифре исходного числа, возведенной в степень, равную разряду цифры. Например, для числа 1374 ответ: 18, т. е.

$$\overset{3\ 2\ 1\ 0}{1\,3\,7\,4} = 1\ (=4^0) + 7\ (=7^1) + 9\ (=3^2) + 1\ (=1^3) = 18.$$

4.5 Вычисление бесконечных сумм с помощью циклов с условиями

Рассматриваем сумму вида

$$\sum_{i=i_{\text{нач}}}^{\infty} y_i = y_{i_{\text{нач}}} + y_{i_{\text{нач}}+1} + \dots$$

Общий вид рекуррентного соотношения:

$$y_i = C \cdot y_{i-1}, \tag{4.1}$$

где C — некоторое выражение, на которое надо домножить слагаемое y_{i-1}, чтобы получить y_i.

Рекуррентное соотношение необходимо дополнить информацией о начальном значении

$$y_{i_{\text{нач}}} = C_{i_{\text{нач}}} \tag{4.2}$$

и об интервале изменения индексов

$$i = i_{\text{нач}} + 1, i_{\text{нач}} + 2, \dots \tag{4.3}$$

Значение индекса «$i_{\text{нач}}$» зависит от вида исходного выражения: как правило, это 0 или 1.

Коэффициент C в соотношении (4.1) для конкретной суммы можно получить с помощью простых математических вычислений (см. рис. 2). Другой способ использован при решении примера 4.7.

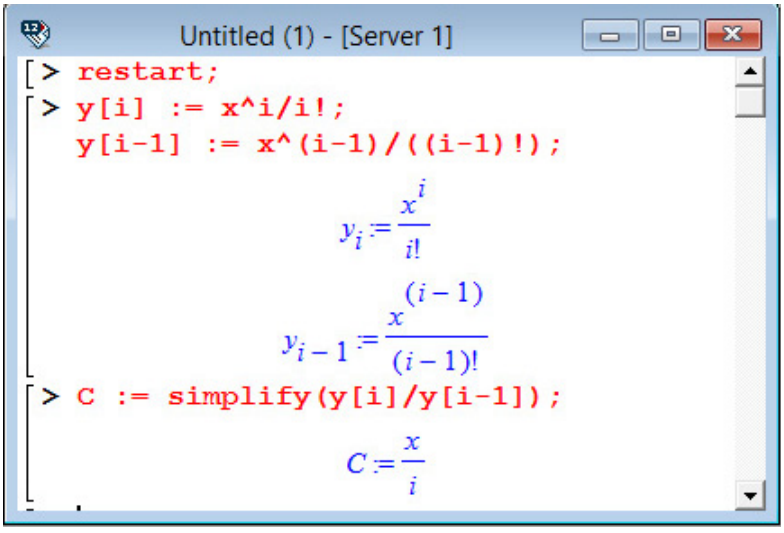


Рис. 2. Получение коэффициента C для (4.4) с помощью программы Maple

Пример 4.7 (вычисление $\sum_{i=0}^{\infty} \frac{x^i}{i!}$). Вычислить приближенное значение бесконечной суммы

$$\sum_{i=0}^{\infty} \frac{x^i}{i!} = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots = e^x. \tag{4.4}$$

Вычисления проводить до тех пор пока очередное слагаемое не станет меньше числа $\varepsilon = 10^{-5}$ (это слагаемое не учитывать). На печать вывести ответ и количество членов суммирования, потребовавшихся для достижения заданной точности.

Условие окончания процесса суммирования

$$|y_i| < \varepsilon$$

означает, что некоторое слагаемое y_i будет давать вклад, несущественный по сравнению с искомой суммой, и его учитывать уже не надо.

На рис. 3 приведены результаты вычисления y_i на примере суммы

$$\sum_{i=1}^{\infty} y_i = \sum_{i=1}^{\infty} \frac{1}{i^2}$$

для $\varepsilon = 0{,}01$.

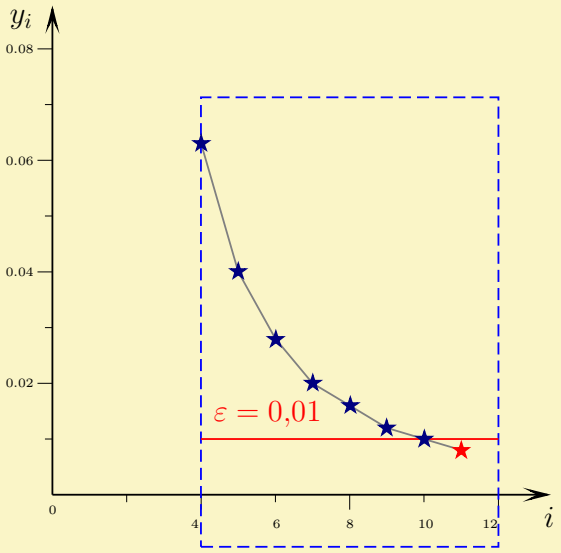


Рис. 3. Значения $y_i = 1/i^2$

Решение. При любых вычислениях надо стараться так применять оператор цикла, чтобы каждый следующий шаг максимально использовал сделанное на предыдущих шагах. Заметим, что вычисляемую сумму можно представить в виде

$$1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots = y_0 + y_1 + y_2 + \ldots,$$

где

$$y_0 = 1, \quad y_1 = \frac{x}{1!}, \quad y_2 = \frac{x^2}{2!}, \quad y_3 = \frac{x^3}{3!}, \quad \ldots$$

Видно, что каждый следующий член y_i получается из предыдущего y_{i-1} умножением на x/i, т. е.

$$y_i = \frac{x^i}{i!} = \frac{x^{i-1}}{(i-1)!} \cdot \frac{x}{i} = y_{i-1} \cdot \frac{x}{i}.$$

Итак, можем записать соотношения

$$y_0 = 1, \quad y_i = \frac{x}{i} y_{i-1}, \quad i = 1, 2, \ldots \tag{4.5}$$

В таблице приведены связи общих формул (4.1)–(4.3) с (4.5)

Формула	Общий вид	Аналог в (4.5)	Примечание
(4.1)	$y_{i_{нач}} = C_{i_{нач}}$	$y_0 = 1$	$i_{нач} = 0, \quad C_{i_{нач}} = 1;$
(4.2)	$y_i = C \cdot y_{i-1}$	$y_i = \frac{x}{i} y_{i-1}$	$C = \frac{x}{i};$
(4.3)	$i = i_{нач} + 1, i_{нач} + 2, \ldots$	$i = 1, 2, \ldots$	

Программа пишется (строго) по соотношениям (4.5). Например, сначала кодируется условие

$$y_0 = 1 \quad (\texttt{i = 0 и y = 1}).$$

В теле цикла записывается рекуррентная формула

$$y_i = y_{i-1}\frac{x}{i} \quad (\texttt{y = y * x / i})$$

и наращивается счетчик цикла i. Переменная Sum нужна для нахождения суммы элементов y_i и она инициализируется нулевым значением (Sum = 0) до начала цикла.

$\sum_{i=0}^{\infty} \frac{x^i}{i!}$

```
1  import math
2  EPS = 1E-5
3  x = float(input('x = '))
4  Sum = 0
5  i = 0
6  y = 1 #  нач. усл.
7  while abs(y) >= EPS:
8      Sum = Sum + y  # суммирование
9      i = i + 1      # i=1,2,...
10     y = y * x / i  # рек. формула
11 print('Число шагов цикла = ', i)
12 print('Сумма:', '%9.6f' %Sum)
13 print('Проверка: exp(', x, ') =%9.6f' %math.exp(x))  # !
```

Последняя строка добавлена для проверки алгоритма, так как известно, что сумма первых слагаемых ряда $\sum_{i=0}^{\infty} \frac{x^i}{i!} = e^x$.

Результат работы программы

```
x = 0.1
Число шагов цикла =  4
Сумма:  1.105167
Проверка: exp( 0.1 ) = 1.105171
```

При выполнении программы переменные Sum, i, y последовательно принимают значения

$$\begin{array}{ll} \texttt{Sum} & 0,\quad 1,\quad 1+\frac{x}{1!},\quad 1+\frac{x}{1!}+\frac{x^2}{2!},\quad \ldots; \\ \texttt{i} & 0,\quad 1,\quad 2,\quad \ldots\ ; \\ \texttt{y} & 1,\quad \frac{x}{1!},\quad \frac{x^2}{2!},\quad \ldots \end{array}$$

Можно для контроля вычислений в тело цикла добавить оператор печати вида

```
print(i, '%10.6f' %Sum, '%9.6f' %y,'\t', (abs(y) >= EPS))
```

При $x = 0{,}1$ на экран будет выведен текст

Трассировочная таблица

```
1    1.000000  0.100000   True
2    1.100000  0.005000   True
3    1.105000  0.000167   True
4    1.105167  0.000004   False
```

Как видно, на четвертом шаге условие

$$\texttt{abs(y) >= Eps} \quad (4\cdot 10^{-6} < 1\cdot 10^{-5})$$

перестало выполняться и программа вышла из цикла.

$\sum_{i=0}^{\infty} \frac{x^i}{i!}$ (использование встроенных функций Python)

```
1  import math
2  EPS = 1E-5
3  x = float(input('x = '))
4  Sum = 0
5  i = 0
6  y = 1 #  нач. усл.
7  while abs(y) >= EPS:
8      Sum = Sum + y
9      i = i + 1
10     y = x**i/math.factorial(i)   # не рекуррентная формула
11 print('Число шагов цикла = ', i)
12 print('Сумма:', '%9.6f' %Sum)
13 print('Проверка: exp(', x, ')=%9.6f' %math.exp(x))
```

Пример 4.8 (вычисление $\sum_{i=1}^{\infty} \frac{1}{i^2}$)**.** Вычислить приближенное значение бесконечной суммы

$$\sum_{i=1}^{\infty} \frac{1}{i^2} = \sum_{i=1}^{\infty} y_i. \tag{4.6}$$

Вычисления проводить до тех пор пока очередное слагаемое не станет меньше числа $\varepsilon = 10^{-5}$ (это слагаемое не учитывать).

Решение. Общий член суммирования

$$y_i = \frac{1}{i^2}, \quad i = 1, 2, \ldots$$

Формулу перепишем в виде

$$\underbrace{y_1 = 1}_{\text{инициализация}} \qquad \underbrace{y_i = \frac{1}{i^2}, \quad i = 2, 3, \ldots}_{\text{в теле цикла}}$$

Рекуррентной формулы здесь не требуется, т. е. её использование скорее затруднит написание программы, чем поможет этому.

Вычисление $\sum_{i=1}^{\infty} \frac{1}{i^2}$

```
1 Sum = 0  # для вычисления суммы
2 i = 1
3 y = 1    # I шаг
4 while y >= 1E-5:
5     Sum = Sum + y  # вычисление суммы
6     i = i + 1      # наращивание счетчика
7     y = 1 / i / i  # общая формула
8 print('Ответ:', '%8.5f' %Sum)
```

Для проверки результата можно использовать или пакет Maple (см. раздел 12.1) или библиотеку Python'а symp.

$\sum_{i=1}^{\infty} \frac{1}{i^2}$. Библиотека sympy

```
from sympy import *
i = Symbol('i')
S = Sum(1/i/i, (i, 1, oo)).doit().evalf()
print(S) # 1.64493406684823
```

Обозначение знака бесконечности ∞ в библиотеке sympy: oo

Пример 4.9. Вычислить сумму с точностью $\varepsilon = 10^{-5}$

$$1 - \frac{1}{2^2} + \frac{1}{3^2} - \frac{1}{4^2} + \ldots = 1 + \left(-\frac{1}{2^2}\right) + \frac{1}{3^2} + \left(-\frac{1}{4^2}\right) + \ldots =$$

$$= 1 + \frac{(-1)}{2^2} + \frac{1}{3^2} + \frac{(-1)}{4^2} \ldots = \sum_{i=1}^{\infty} \frac{(-1)^{i+1}}{i^2} = \sum_{i=1}^{\infty} y_i. \qquad (4.7)$$

Решение. Для нахождения степени $(-1)^{i+1}$ не будем использовать ни операцию, ни функцию.

Проследим за изменением знака

i	1	2	3	4	...
$(-1)^{i+1}$	1	-1	1	-1	...

Переменная для смены знака:

z = 1 до цикла

z = -z в теле цикла

Вычисление $\sum_{i=1}^{\infty} \frac{(-1)^{i+1}}{i^2}$

```
Sum = 0  # для вычисления суммы
i = 1
y = 1    #  I шаг
z = 1    #  переменная для смены знака
while abs(y) >= 1E-5:
    Sum = Sum + y
    i = i + 1
    z = -z
    y = z / i / i
print('Ответ:', '%8.5f' %Sum)
```

Результат: 0.82246

Вычисление $\sum_{i=1}^{\infty} \frac{(-1)^{i+1}}{i^2}$. Библиотека sympy

```
from sympy import *
i = Symbol('i')
S = Sum((-1)**(i+1)/i/i, (i, 1, oo)).evalf()
print(S) #
```

Результат: 0.822467033424113 (подчеркнут результат реализации нашего алгоритма)

4.6 Задачи-II

4.26 ([14]). Вычислить приближенное значение бесконечной суммы

$$1)\ \frac{1}{1\cdot 2}+\frac{1}{2\cdot 3}+\frac{1}{3\cdot 4}+\ldots;$$

$$2)\ \frac{1}{1\cdot 3}+\frac{1}{2\cdot 4}+\frac{1}{3\cdot 5}+\ldots;$$

$$3)\ \frac{1}{1\cdot 2\cdot 3}+\frac{1}{2\cdot 3\cdot 4}+\frac{1}{3\cdot 4\cdot 5}+\ldots$$

Вычисления продолжать до тех пор пока очередное слагаемое не окажется меньше числа $\varepsilon = 10^{-5}$.

Указание. Осуществите проверку результата при помощи пакета Maple или библиотеки sympy.

4.27 ([14]). Вычислить приближенное значение бесконечной суммы

$$1)\ 1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\ldots; \qquad 2)\ 1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\ldots$$

Вычисления продолжать до тех пор пока очередное слагаемое не окажется по модулю меньше числа $\varepsilon = 10^{-5}$.

Указание. Осуществите проверку результата при помощи пакета Maple или библиотеки sympy.

4.28. Дано действительное x. Вычислить значение функции

$$y = \operatorname{ch} x = \sum_{i=0}^{\infty} \frac{x^{2i}}{(2i)!} = \sum_{i=0}^{\infty} y_i,$$

используя соотношения

$$y_0 = 1, \quad y_i = \frac{x^2}{(2i-1)2i}\cdot y_{i-1}, \quad i = 1, 2, 3, \ldots$$

Вычисления прекратить при $|y_i| < \varepsilon$, $\varepsilon = 10^{-5}$. Определить количество итераций.

4.29. Вычислить отрицательный корень уравнения

$$x^3 - x + 0{,}5 = 0,$$

используя рекуррентную формулу $x_i = \sqrt[3]{x_{i-1} - 0{,}5}$ $(i = 1, 2, \ldots)$. Вычисления прекратить при $|x_{i+1} - x_i| < \varepsilon$, $\varepsilon = 10^{-4}$. Определить количество итераций.

Указание. Начальное значение x_0 определите при помощи пакета Maple: постройте график функции $y = x^3 - x + 0{,}5$ и возьмите значение, близкое к искомому отрицательному корню.

4.30. Дано действительное x. Вычислить приближенное значение бесконечной суммы:

1) $-\ln(1-x) = x + \frac{x^2}{2} + \frac{x^3}{3} + \ldots \quad (|x| < 1);$

2) $\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots \quad (|x| < 1);$

3) $-\ln\left(1 - \frac{x-1}{x}\right) = \frac{x-1}{x} + \frac{(x-1)^2}{2x^2} + \frac{(x-1)^3}{3x^3} + \ldots \quad (x > 0{,}5).$

На печать выдавать значение суммы, посчитанное с заданной точностью, и значение функции.

Указание. Используйте рекуррентные соотношения.

4.31. Дано действительное x ($|x| \leqslant \pi/4$). Вычислить приближенное значение бесконечной суммы:

1) $\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots;$

2) $\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots.$

На печать выдавать значение суммы, посчитанное с заданной точностью, и значение функции.

Указание. Используйте рекуррентные соотношения.

4.7 Оператор цикла с параметром

Синтаксис оператора цикла с параметром (for-to)

```
for ПЦ in итерируемый объект: блок инструкций
```

Здесь

- ПЦ — параметр цикла;
- итерируемый объект — строка, список, словарь, файл,...
- блок инструкций представляет собой тело цикла (любой оператор, в том числе составной).

Синтаксис заголовка оператора for i in range(n)

```
for i in range(количество шагов цикла):
```

Синтаксис заголовка оператора for i in range(n, m, [step])

```
for i in range(нач. знач., кон. знач.+1, [шаг]):
```

Здесь в [...] отмечена необязательная часть.

Синтаксис заголовка оператора for i in [элементы списка]):

```
for i in [элементы списка]:
```

Пример 4.10 (нахождение $\sum_{i=1}^{10} i$). Вычислить сумму первых десяти натуральных чисел $\sum_{i=1}^{10} i$.

for i in range(10):

```
S = 0
for i in range(10):
    S = S + (i+1)
print('S =', S)
```

(for i in range(1,11):

```
S = 0
for i in range(1,11):
    S = S + i
print('S =', S)
```

for i in range(10,0,-1):

```
S = 0
for i in range(10,0,-1):
    S = S + i
print('S =', S)
```

while i <= 10:

```
S = 0
i = 1
while i <= 10:
    S = S + i
    i += 1
print('S =', S)
```

Пример 4.11 (цикл с параметром)**.** Демонстрация разных заголовков цикла с параметром.

Функция range(n)

```
for i in range(4):
    print(i)
```

Функция range(n, m)

```
for i in range(0, 4):
    print(i)
```

Функция range(n, m, step)

```
for i in range(0, 4, 1):
    print(i)
```

Функция range(n, m, -step)

```
for i in range(3, -1, -1):
    print(3-i)
```

Элементы списка

```
for i in [0, 1, 2, 3]:
    print(i)
```

Элементы кортежа

```
for i in (0, 1, 2, 3):
    print(i)
```

Вывод букв слова

```
for i in 'котик':
    print(i)
```

Счёт

```
for i in 'один', 'два', 'три':
    print(i)
```

Список значений разного типа для параметра цикла

```
for i in 1, 2, 3, 'раз', 'два', True:
    print(i)
```

Пример 4.12 (табулирование функций)**.** Цикл for удобно использовать для получения таблиц функций. Например, пусть дано $x \in \mathbb{R}$, необходимо вывести таблицу значений функции $\sin x$ в виде

x	$\sin x$
-0.2	-0.1987
-0.1	-0.0998
0.0	0.0000
$\ldots$	$\ldots$

где x принимает значения $-0{,}2$, $-0{,}1$,..., $0{,}2$.

Табулирование функции

```
1 import math
2 print('   x  | sin(x) ') # шапка таблицы
3 print('----------------')
4
5 for i in range(-2, 3):
6     x = 0.1 * i
7     y = math.sin(x);
8     print('%4.1f' %x, ' | %7.4f' %y)
```

Результат работы программы

```
   x  | sin(x)
----------------
 -0.2 | -0.1987
 -0.1 | -0.0998
  0.0 |  0.0000
  0.1 |  0.0998
  0.2 |  0.1987
```

Пример 4.13 (нахождение $\min\limits_k y_k$). Определить наименьшее из чисел

$$y_k = k \sin\left(n + \frac{k}{n}\right), \quad k = 1, 2, \ldots, n.$$

Поиск наименьшего из чисел

```
1  import math
2  n = int(input('n = '))
3  Min = math.sin(n + 1/n)
4  kMin = 1
5  for k in range(2, n+1):
6      y = k * math.sin(n + k/n)
7      if y < Min:
8          Min = y
9          # запоминается новый номер эл-нта с мин. знач.
10         kMin = k
11
12 print('kMin =', kMin, '  Min =', '%6.3f' %Min)
```

Наименьшее из чисел y_k ($k = 1, 2, \ldots, n$) определяется за n шагов. Первый шаг состоит в инициализации переменной Min — ей присваивается значение первого члена последовательности $y_1 = \sin(n + 1/n)$. В цикле проводится вычисление всех остальных членов последовательности и сравнение их с минимальным.

Результаты работы программы:

I тест

```
n = 5
kMin = 3  Min = -1.894
```

II тест

```
n = 6
kMin = 1  Min = -0.116
```

Если добавить в программный код соответствующие операторы печати, то можно получить таблицы, позволяющие оценить правильность полученного решения:

```
 k |   y
-----------
 1 | -0.883
 2 | -1.546
 3 | -1.894
 4 | -1.858
 5 | -1.397
```

```
 k |   y
-----------
 1 | -0.116
 2 |  0.100
 3 |  0.645
 4 |  1.497
 5 |  2.614
 6 |  3.942
```

Пример 4.14 (использование break и else в циклах)**.** Ищем в списке слово raccoon.

Использование break и else

```
1 s = 'raccoon'
2 for i in ['cat', 'dog', 'mouse']:
3     if i == 'mise':
4         print('Слово', s, 'в списке есть')
5         break
6 else: # значит вышли не по break
7     print('Слова', s, 'в списке нет')
```

Результат работы для списка ('cat', 'dog', 'mouse')

```
Слова raccoon в списке нет
```

Результат работы для списка ('cat', 'dog', 'raccoon')

```
Слово raccoon в списке есть
```

4.8 Задачи–III

☎ В заданиях 4.32–4.34 запись «d.dddd», где d — это любая десятичная цифра, означает, что вещественное число должно быть записано в виде с фиксированной точкой и иметь после запятой 4 цифры.

4.32. Дано: $m, n \in \mathbb{Z}$ $(m < n)$. Вывести таблицу значений функции

$$f(x) = \sin x$$

в виде

x	$y = f(x)$
d.dd	d.dddd
...	...

где $x_i = 0{,}1i$, $y_i = f(x_i)$ $(i = m, \ldots, n)$; $\mathrm{d} = 0, \ldots, 9$.

4.33. Дано: $a, b \in \mathbb{R}$ $(a < b)$, $n \in \mathbb{N}$. Вывести таблицу значений функции

$$f(x) = \cos x$$

в виде

x	$y = f(x)$
d.ddd	d.dddd
...	...

где $x_i = a + hi$, $h = \dfrac{b-a}{n}$, $y_i = f(x_i)$ $(i = 0, 1, \ldots, n)$; $\mathrm{d} = 0, \ldots, 9$.

4.34. Дано: $h \in \mathbb{R}$, $m, n \in \mathbb{Z}$ $(m < n)$. Вывести таблицу значений функции

$$f(x) = \frac{x^2}{\sqrt{2x-1}}$$

в виде

x	$y = f(x)$
d.dd	d.dddd
...	...

где $x_i = 1 + ih$, $y_i = f(x_i)$ $(i = m, \ldots, n)$; $\mathrm{d} = 0, \ldots, 9$. В случае, когда значение функции не может быть получено (например, при $x = 0{,}5$ знаменатель дроби равен нулю) строка таблицы должна иметь вид:

0.50	–

4.35. Напечатать фрагмент таблицы истинности логической функции

$$F(x, y, z) = \neg y \wedge (x \vee \neg z),$$

для которой $F(x, y, z) =$ True.

Указание. Ответ выдавать в виде

```
x y z f
0 0 0 1
1 0 0 1
...
```

4.36. Напечатать фрагмент таблицы истинности логической функции

$$F(x, y, z) = (x \to y) \wedge z,$$

для которой $F(x, y, z) =$ False.

Указание. Ответ выдавать в виде

```
x y z f
0 0 0 0
0 1 0 0
...
```

4.37. Вывести на экран все делители натурального числа n и подсчитать их количество.

4.38. Определить, является ли заданное число n совершенным.

Совершенным называется натуральное число, равное сумме всех своих делителей, исключая само число. Например: $28 = 1 + 2 + 4 + 7 + 14$.

4.39. Найти первые три совершенных числа.

4.40. Дано натуральное n. Вычислить:

$$1)\ \sum_{i=1}^{n} \frac{(-1)^i}{2i+1}; \qquad 2)\ \sum_{i=1}^{n} \frac{(-1)^{i+1}}{i(i+1)}.$$

Указание. Осуществите проверку результата при помощи пакета Maple или библиотеки sympy.

4.41. Дано натуральное n. Вычислить:

$$1)\ \sum_{i=1}^{n}\frac{(-1)^i(i+1)}{i!};\qquad 2)\ \sum_{i=1}^{n}\frac{i!}{\sum\limits_{j=1}^{i}\frac{1}{j}}.$$

Указание. Осуществите проверку результата при помощи пакета Maple или библиотеки sympy.

4.42. Дано: $n\in\mathbb{N}$, $x\in\mathbb{R}$. Вычислить:

$$1)\ \prod_{i=1}^{n}\left(1+\frac{\sin ix}{i!}\right);\qquad 2)\ \prod_{i=1}^{n}\left(\frac{i}{i+1}-\cos^i|x|\right).$$

Указание. Осуществите проверку результата при помощи пакета Maple или библиотеки sympy.

4.43 ([14])**.** Дано: $n\in\mathbb{N}$, $x\in\mathbb{R}$. Вычислить:

$$1)\ \ \sin x+\sin^2 x+\ldots+\sin^n x;\qquad 2)\ \ \cos x+\cos x^2+\ldots+\cos x^n.$$

Указание. Осуществите проверку результата при помощи пакета Maple или библиотеки sympy.

4.44. Дано: $m\in\mathbb{N}$, $x\in\mathbb{R}$. Вычислить:

$$(1+x)^m=\sum_{k=0}^{m}C_m^k x^k;$$

где $C_m^k=\dfrac{m!}{k!(m-k)!}=\dfrac{m(m-1)(m-2)\ldots(m-k+1)}{k!}$ — биномиальный коэффициент; заметим, что

$$\frac{C_m^i}{C_m^{i-1}}=\frac{m-i+1}{i},\quad i=1,\ldots,m.$$

4.45. Вычислить:

$$1)\ 1+\frac{1}{2^3}+\ldots+\frac{1}{50^3};\qquad 2)\ \frac{1}{2^2}+\frac{1}{4^2}+\frac{1}{8^2}+\ldots+\frac{1}{128^2}.$$

Результаты для сравнения: 1) 1,20186; 2) 0,33331.

4.46. Вычислить:

$$1)\ 1-\frac{1}{2!}+\ldots-\frac{1}{10!};$$

$$2)\ \left(2+\frac{1}{1!}\right)\left(2-\frac{1}{2!}\right)\left(2+\frac{1}{3!}\right)\ldots\left(2-\frac{1}{8!}\right).$$

Результаты для сравнения: 1) 0,63212; 2) 306,58649.

4.47. Дана последовательность из n целых чисел. Найти произведение и сумму положительных элементов последовательности, следующих после первого нулевого элемента.

4.48. Дана последовательность из n целых чисел. Найти значение и номер последнего из минимальных элементов последовательности (предполагается, что таких элементов может быть несколько).

4.49. Дана последовательность из n целых чисел. Найти отношение суммы положительных элементов последовательности к количеству отрицательных элементов. (Учесть случай, когда отрицательных элементов нет.)

4.50. Дана последовательность из n целых чисел. Найти наибольший элемент и его номер среди элементов, следующих после первого нулевого элемента (если таких элементов несколько, то первый из них).

4.51. Дана последовательность из n целых чисел. Найти произведение первых подряд идущих положительных элементов последовательности.

4.52. Последовательность $\{a_i\}_{i=1}^{n}$ образована из n целых чисел. Определить сколько раз в данной последовательности меняется знак при переходе к следующему элементу. Будем говорить, что имеет место перемена знака, если для некоторого значения i: $a_i \cdot a_{i+1} < 0$.

4.53 ([14]). Дано: натуральное n, действительные y_1,..., y_n. Найти:

$$1)\quad \max(|z_1|,\ldots,|z_n|),\quad \text{где}\quad z_i=\begin{cases} y_i & \text{при } |y_i|\leqslant 2,\\ 0{,}5 & \text{в противном случае;}\end{cases}$$

$$2)\quad \min(|z_1|,\ldots,|z_n|),\quad \text{где}\quad z_i=\begin{cases} y_i & \text{при } |y_i|>1,\\ 2 & \text{в противном случае;}\end{cases}$$

$$3)\quad z_1^2+\ldots+z_n^2,\quad \text{где}\quad z_i=\begin{cases} y_i & \text{при } 0<y_i<10,\\ 1 & \text{в противном случае.}\end{cases}$$

4.54 (★). Дано: $n \in \mathbb{N}$, $x \in \mathbb{R}$. Среди чисел $y_k = k \cos x^{2k}$, $k = 1, 2, \ldots, n$, найти ближайшее к какому-нибудь целому числу.

Пояснение к задаче. Обозначим: z_k — ближайшее целое к y_k ($k = 1, 2, \ldots, n$); $\varepsilon_k = |y_k - z_k|$.

При $x = 5{,}5$ и $n = 8$ получим следующую последовательность значений

k	y_k	z_k	ε_k
1	0.39389908	0	0.39389908
2	−1.30633272	−1	0.30633272
3	−2.99311668	−3	0.00688332
4	−1.20738400	−1	0.20738400
5	−1.20524969	−1	0.20524969
6	0.58352289	1	0.41647711
7	3.88899163	4	0.11100837
8	−5.65685425	−6	0.34314575

Как видно из таблицы минимальное ε_k будет при $k = 3$. Значит ближайшим к некоторому целому числу будет третий элемент последовательности: $y_3 = -2.99311668$.

5 Функции (часть I)

С помощью функции можно оформить отдельный алгоритм, а затем его использовать один или несколько раз в основной программе. Это удобно, например, если какой-то код повторяется несколько раз. Также с помощью функций большую задачу разбивают на маленькие подзадачи и их оформляют в виде функций.

Ранее уже использовались стандартные функции Python, такие как `input()`, `len()`, `print()` и т.д. Научимся писать свои функции.

5.1 Описание функции

Описание функции

```
def имя_функции(список параметров):
    тело функции
    [return значение_функции]  # в […] отмечена необязательная часть
```

После ключевого слова def идёт имя функции, с помощью которого затем она будет вызываться. Список параметров может быть пустым, но круглые скобки надо ставить обязательно. В конце строки заголовка ставится двоеточие. Начиная со следующей строки, идет код функции. Выйти из функции можно в любой момент, используя инструкцию `return`. Если инструкция `return` используется без аргументов или она вообще не используется, то функция будет возвращать значение None.

Для того чтобы функция вернула не одно значение, а более, можно после инструкции return через запятую записать несколько переменных:

```
return a, b
```

Функция вернет кортеж из этих переменных. Результат вызова такой функции можно будет использовать во множественном присваивании:

```
n, m = имя_функции(список параметров)
```

Также можно собрать результат в любую структуру и вернуть ее.

Функция должна быть записана выше своего использования в любом месте программы (обычно выше текста основной программы). Также описания функций можно выносить в отдельные файлы, которые называются модулями.

5.2 Вызов функции

Вызов функции осуществляется по её имени, за ним стоят круглые скобки, внутри которых ей передаются фактические параметры, если они есть. Функция всегда возвращает значение (в частности может вернуть значение None), но его можно игнорировать.

Синтаксис вызова функции:

```
[переменная =] имя_функции(список параметров)
```

Пример 5.1 (функция без параметров)**.** Функции без параметров и выходных значений обычно используются для отображения каких-либо статичных текстов, например, меню:

Функция без параметров и значения

```
1 def print_menu():
2     print('   Меню')
3     print('1: Метод 1')
4     print('2: Метод 2')
5     print('3: Выход')
6
7 print_menu()  # вызов функции без параметров и значения
```

При наличии параметров в простейшем случае параметры функции записываются как последовательность идентификаторов, разделенных запятыми.

Пример 5.2 (функции с параметрами)**.** Описание и вызов функции для нахождения $x + y$.

Функция для нахождения $x + y$

```
1 def sum2(x, y):
2     return x + y
3
4 S = sum2(5, 3)
```

Пример 5.3 (функция поиска максимума)**.** Функция для поиска максимума из двух целых чисел может быть записана следующим образом

Описание и примеры вызова функции для поиска максимума

```
def Max(x, y):
    if x > y:
        return x
    else:
        return y

c = Max(3,4)
print(Max(abs(-5), c))
```

Пример 5.4 (более одного результата)**.** Для возвращения функций более одного значения используется список

Два результата у функции

```
def MinMax(a, b):
    if a > b:
        return [b, a]
    else:
        return [a, b]
```

Тогда результат вызова функции можно будет использовать либо так

Примеры вызовов функций

```
minab = MinMax(56, 17)[0]
maxab = MinMax(56, 17)[1]
print('min =', minab)
print('max =', maxab)
```

либо во множественном присваивании:

Примеры вызовов функций

```
minab, maxab = MinMax(56, 17)
print('min =', minab)
print('max =', maxab)
```

Пример 5.5 (функция для вычисления площади треугольника)**.** Вычислить площадь треугольника по формуле Герона (параметры a, b, c — здесь стороны треугольника)

Использование функции с параметрами

```
import math
def geron(a, b, c):
    p = (a + b + c) / 2
    return math.sqrt(p * (p - a) * (p - b) * (p - c))

print('Площадь =', geron(3, 4, 5))
```

При вызове функции первому аргументу a предается первое значение 3, второму аргументу b – 4, и третьему c — 5. Такой способ передачи аргументов, когда данные передаются в порядке их перечисления, называется позиционным.

Другой способ передачи параметров — по ключу (или по имени). Используя его можно было бы вызвать предыдущую функцию так:

```
print('Площадь =', geron(a = 3, b = 4, c = 5))
```

Причем перечислять параметры можно в любом порядке:

```
print('Площадь =', geron(c = 5, b = 4, a = 3))
```

Можно комбинировать два способа вызова (позиционный и по имени) в одном вызове:

```
print('Площадь =', geron(3, 4, c = 5))
```

Тогда при вызове позиционные параметры обязательно должны идти перед параметрами передаваемыми по имени:

Сначала позиционные!

```
print('Площадь =', geron(c = 5, 3, 4) ) # ошибка,
# сначала должны идти позиционные,
# а потом передаваемые по имени параметры
```

5.3 Значения параметров по умолчанию

При определении функции для ее параметров можно задавать значения по умолчанию, для этого они задаются в виде последовательности пар:

идентификатор = значение.

Пример 5.6 (значения по умолчанию; задание)**.** Приведем заголовки функций со значениями по умолчанию.

Заголовки функций

```
def geron2(a=3, b=4, c=5):
def point_xy(x=0, y=0):
def set_pixel(color, x=0, y=0):
def set_pixel_no(x=0, y=0, color): # ошибка (см. ниже)
```

Причем, если в функции используются как параметры без начальных значений, так и с начальными значениями (как в функции `set_pixel`), то обычные параметры всегда должны идти перед параметрами с начальными значениями.

Пример 5.7 (значения по умолчанию; вызов)**.** Можно при вызове функции вообще не передавать параметры со значениями по умолчанию.

Значения по умолчанию; вызов

```
geron2()          # вызывается с параметрами по умолчанию
point_xy(10)      # параметры x=10 и y=0
set_pixel('red')  # параметры color='red', x=0, y=0
geron2(7, b = 8)  # параметры:
# a = 7 - передается по позиции,
# b = 8 - передается по имени,
# c = 5 - используется значение по умолчанию.
```

 **5.1.** Значения по умолчанию задаются один раз при определении функции, а не при каждом вызове!

Пример 5.8 (сравнение двух выражений с заданной точностью)**.** Вещественные числа по умолчанию сравниваются с машинной точностью. На практике такая точность бывает нужна редко. Напишем функцию для сравнения двух выражений с заданной точностью. Точность по умолчанию возьмем $\varepsilon = 0.000001$.

Сравнение с заданной точностью

```
def Compare(x, y, eps = 1e-6):
    return abs(x - y) <= eps

print(Compare(1.55, 1.6));      # вернет False
print(Compare(1.55, 1.6, 0.1));  # вернет True
```

5.4 Задачи

5.1. Вычислить площадь выпуклого четырехугольника, заданного длинами всех сторон и диагонали (см. рис. 4).

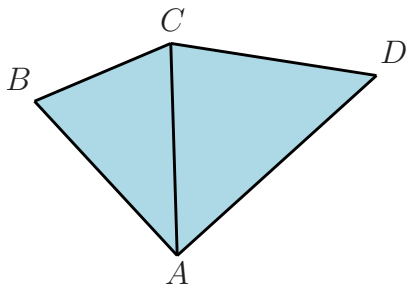


Рис. 4. Заданный четырехугольник

Указание. В программу добавьте проверку существования треугольника с заданными сторонами.

5.2. Даны целые a, b и c. Получить значение выражения

$$\frac{\max(a, a+b) + \max(a, b+c)}{1 + \max(a+bc, 4)}.$$

Указание. Для нахождения максимума используйте функцию Max из примера 5.3.

5.3. Даны целые a, b. Получить значения выражений

$$u = \min(a, b), \quad v = \min(ab, a+b), \quad w = \min(u + v^2, 14).$$

Указание. Для нахождения минимума создайте функцию на основе функции Max из примера 5.3.

5.4. Описать функции для вычисления а) $\operatorname{tg} x$; $\operatorname{sh} x$; б) $\operatorname{ctg} x$; $\operatorname{ch} x$, используя только функции exp(), sin(), cos(), atan() из библиотеки math.

Сравнить результаты функций со значениями, посчитанными с помощью библиотечных функций для вычисления $\operatorname{tg} x$, $\operatorname{ctg} x$, $\operatorname{sh} x$ и $\operatorname{ch} x$.

5.5. Описать функцию, которая для любого целого аргумента возвращает количество цифр в его записи.

5.6. Описать функцию, которая для любого целого аргумента возвращает целое значение, полученное изменением порядка следования цифр на обратный.

5.7. Описать функцию для печати таблицы Пифагора.

Указания. Определите параметры со значениями по умолчанию $n = 10$ и $m = 10$ — количество строк и столбцов в таблице; используйте разные вызовы: позиционный, по имени, с использованием значений по умолчанию (см. примеры выше).

5.8. Описать функцию для печати таблицы «сумма квадратов» в виде матрицы:

0	1	2	3
1	2	5	10
2	5	8	13
3	10	13	18

Указания. Число строк и столбцов в матрице передавать как параметры со значениями по умолчанию $n = 3$ и $m = 3$; используйте разные вызовы: позиционный, по имени, с использованием значений по умолчанию (см. примеры выше).

5.9. Описать функцию для вычисления значения $\cos^n x$ с помощью формул понижения степени:

$$\cos^n x = \begin{cases} \dfrac{1 + \cos 2x}{2}, & \text{если } n = 2; \\ \dfrac{3\cos x + \cos 3x}{4}, & \text{если } n = 3; \\ \dfrac{3 + 4\cos 2x + \cos 4x}{8}, & \text{если } n = 4; \\ \dfrac{10\cos x + 5\cos 3x + \cos 5x}{16}, & \text{если } n = 5. \end{cases}$$

Для сравнения вычислить $\cos^n x$ с помощью цикла (оформить в виде функции).

5.10. Описать функцию для вычисления значения $\sin^n x$ с помощью формул понижения степени:

$$\sin^n x = \begin{cases} \dfrac{1 - \cos 2x}{2}, & \text{если } n = 2; \\ \dfrac{3\sin x - \sin 3x}{4}, & \text{если } n = 3; \\ \dfrac{3 - 4\cos 2x + \cos 4x}{8}, & \text{если } n = 4; \\ \dfrac{10\sin x - 5\sin 3x + \sin 5x}{16}, & \text{если } n = 5. \end{cases}$$

Для сравнения вычислить $\sin^n x$ с помощью цикла (оформить в виде функции).

5.11. Описать функцию для нахождения суммы положительных четных чисел, меньших заданного числа m.

5.12. Описать функцию для нахождения суммы положительных нечетных чисел, меньших заданного числа m.

5.13. Описать функцию для нахождения суммы целых положительных чисел, принадлежащих отрезку $[a, b]$ и кратных числу m.

5.14. Дано действительное x. Вычислить приближенное значение бесконечной суммы:

$$\frac{1}{2}\ln\left(\frac{1+x}{1-x}\right) = x + \frac{x^3}{3} + \frac{x^5}{5} + \ldots \quad (|x| < 1).$$

На печать выдавать значение суммы, посчитанное с заданной точностью, и значение функции.

Указание 1. Используйте рекуррентные соотношения.

Указание 2. Для приближенного вычисления суммы используйте функцию с параметрами x и eps.

5.15. Дано действительное x. Вычислить приближенное значение бесконечной суммы:

$$\frac{1}{2}\ln\left(\frac{1+\frac{1}{x}}{1-\frac{1}{x}}\right) = \frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \ldots \quad (|x| > 1).$$

На печать выдавать значение суммы, посчитанное с заданной точностью, и значение функции.

Указание 1. Используйте рекуррентные соотношения.

Указание 2. Для приближенного вычисления суммы используйте функцию с параметрами x и eps.

5.16. Дано: $n \in \mathbb{N}$, $x \in \mathbb{R}$. Вычислить:

$$\sum_{i=1}^{n}\left(\frac{1}{i!} + \sqrt{|x|}\right).$$

Указание 1. Для вычисления факториала используйте рекуррентные соотношения.

Указание 2. Для вычисления суммы используйте функцию с параметрами n и x.

Указание 3. Осуществите проверку результата при помощи пакета Maple или библиотеки sympy.

5.17. Дано: $n \in \mathbb{N}$, $x \in \mathbb{R}$. Вычислить:

$$\sum_{i=1}^{n} \frac{x^i + i}{i!}.$$

Указание 1. Для вычисления факториала и степенной функции используйте рекуррентные соотношения.

Указание 2. Для вычисления суммы используйте функцию с параметрами n и x.

Указание 3. Осуществите проверку результата при помощи пакета Maple или библиотеки sympy.

5.18. Даны число $n \in \mathbb{N}$, первый член b_1 и знаменатель $q \neq 0$ геометрической прогрессии. Опишите функцию для определения суммы S_n первых n членов прогрессии по формуле

$$S_n = \frac{b_1 \cdot (q^n - 1)}{q - 1}.$$

Указание. Для определения степени числа q^n ($n \in \mathbb{N}$) используйте рекуррентное соотношение.

5.19. Даны число $n \in \mathbb{N}$, первый член b_1 и ненулевой знаменатель $q \neq 1$ геометрической прогрессии. Опишите функцию для определения модуля произведения P_n первых n членов прогрессии по формуле

$$|P_n| = \sqrt{|b_1 \cdot b_n|^n}.$$

Указание 1. Для определения n-го элемента геометрической прогрессии используйте формулу

$$b_n = b_1 \cdot q^{n-1}.$$

Указание 2. Для определения степени числа q^n ($n \in \mathbb{N}$) используйте рекуррентное соотношение.

---

6 Строковый тип данных

Тип данных str предназначен для хранения последовательности символов с произвольным доступом.

Символы в строке нумеруются с нуля. Функция `len()` служит для определения длины строки — количества элементов в строке. Длина пустой строки `len('')` равна нулю.

При выполнении программы переменная типа `str` вводится с клавиатуры без апострофов.

В языке Python строковый тип — неизменяемый тип, т. е. изменить символ в строке следующей инструкцией нельзя: `S[1] = 'I'`

Пример 6.1. Дана строка S. Получить новую строку NS, переставив символы исходной строки в обратном порядке.

Перестановка символов в строке (3 варианта)

```
S = 'kotik'
NS = ''
for c in S: NS = c + NS

S = 'kotik'
NS = ''
for i in range(len(S)): NS = S[i] + NS

S = 'kotik'
NS = ''
for i in range(len(S)-1,-1, -1): NS = NS + S[i]
```

Пример 6.2. Пусть определена переменная Date, в которой содержится строка с датой формата:

ДД МММ, ГГГГ;

где ДД — число месяца (позиции 1–2), МММ — месяц (4–6) и ГГГГ — год (9–12).

Извлечение названия месяца из исходной строки:

```
Month = Date[3:6]
```

Поиск месяца

```
Date = '22 ОСТ, 2017'
Month = Date[3:6]
print(Month)
```

Пример 6.3. Если длина строки нечетное число, то удалить среднюю букву.

Удаление буквы

```
S = 'Коала'
D = len(S)
print(D)
if D % 2 != 0:
    S = S[:D//2] + S[D//2+1:]
print(S)
```

6.1 Функции и методы строк

6.1.1 Метод поиска в строке

Метод count() подсчитывает количество вхождений строки Образец в Строку

```
Строка.count(Образец[, Старт][, Финиш])
```

Указание необязательных параметров Старт, Финиш позволяет выполнять подсчет числа вхождений строки Образец в срезе строки [Старт:Финиш].

Подсчитываются только непересекающиеся вхождения, например:

Поиск подстроки в строке

```
print(('7' * 10).count('77'))   # вернёт 5
```

Поиск подстроки в строке. Использование срезов

```
# Поиск в строке с третьего по 10 символ
print(('enotik begemotik').count('e',3,10)) # вернёт 1

# Поиск в строке с третьего символа по конец строки
print(('enotik begemotik').count('e',3))    # вернёт 2
```

Методы find() ищет в Строке подстроку Образец

```
Строка.find(Образец[, countS])
```

Поиск подстроки в строке происходит слева направо. Для поиска справа налево существует метод rfind().

Если подстрока найдена, то функция возвращает индекс первого вхождения искомой подстроки, иначе — возвращает значение -1.

Пример 6.4. Пусть определена переменная Date, в которой содержится строка с датой формата:

```
Число Месяц, Год
```

Извлечь название месяца

Поиск месяца

```
Date = '2 October, 2017'
Month = Date[Date.find(' ')+1:Date.find(',')]
print(Month)
```

6.1.2 Метод replace

Метод replace() заменяет в строке все вхождения подстроки oldS на подстроку newS

```
Строка.replace(oldS, newS[, countS])
```

Необязательный параметр countS указывает, что заменены будут только первые countS из найденных строк.

Замена подстроки в строке

```
print('mmcs. 1 course'.replace('s', 'S'))
# вернет 'mmcS. 1 courSe'
```

Замена подстроки в строке. Третий параметр

```
print('mmcs. 1 course'.replace('s', 'S', 1))
# вернет 'mmcS. 1 course'
```

Пример 6.5. Заменить все вхождения подстроки del на Ins. Метод replace() не использовать.

Замена

```
# k  { позиция вхождения искомой подстроки }
S = 'Del и del. Del или del'
print(S)
while S.find('del') != -1:
    k = S.find('del')
    D = len('del')
    S = S[:k] + 'Ins' + S[k+D:]
print(S)
```

Пример 6.6 (перестановка слов)**.** Дана строка, состоящая из нескольких слов (два и более), разделенных пробелом. Переставьте первое и последнее слова местами. Результат запишите в строку и выведите получившуюся строку.

Указание. При решении этой задачи не использовать инструкцию if.

Перестановка слов

```
S = input()
S1 = S[:S.find(' ')]
S2 = S[S.rfind(' ')+1:]
S0 = S[S.find(' '):S.rfind(' ')+1]
SNew = S2 + S0 + S1
print(SNew)
```

Программа будет работать правильно только, если в начале и в конце строки нет пробелов. Поэтому перед использованием среза требуется привести строку в соответствие нашим правилам — «в начале и в конце строки пробелы отсутствуют». Для удаления пробелов в начале и в конце строки используется метод strip().

Пример 6.7 (выбор слов из предложения)**.** Программа отображает в отдельной строке каждое слово исходной строки s. При этом предполагается, что слова в s отделяются одним-единственным пробелом. В начале и в конце строки пробелов нет.

Указание. Использовать только метод find и срезы.

Выбор слов из предложения

```
S = 'Delete и del. Insert или del'
while ' ' in S:
    k = S.find(' ')
    SN = S[:k]
    S = S[k+1:]
    print(SN)
SN = S    # последнее слово
print(SN) # печать последнего слова
```

 **6.1.** Если предложение всегда будет заканчиваться пробелом, то последние две строки не нужны.

6.1.3 Методы split и join

Метод split()

```
строка.split([вид разделителя])
```

преобразует строку в список подстрок, разделенных по умолчанию пробелами.

Метод join()

```
вид разделителя.join(строка)
```

позволяет создавать строки из списка строк.

Разбиение строки

```
s = 'red yellow green'
print(s.split())      # разделитель по умолчанию пробел
                      # ['red', 'yellow', 'green']

s = 'red, yellow, green'
print(s.split(','))   # разделитель <<,>>
                      # ['red', ' yellow', ' green']

s = 'red, yellow, green'
print(s.split(', '))  # разделитель <<,>> + пробел
                      # ['red', 'yellow', 'green']
```

Создание списка строк из строки

```
s = input()              # ввод 1 2 3 (Enter = конец ввода)
a = s.split()            # результат ['1', '2', '3']
```

Создание списка чисел из строки

```
a = input().split()      # строка 12 30 8 678 23
for i in range(len(a)):
    a[i] = int(a[i])     # преобразование элементов строки
print(a)                 # [12, 30, 8, 678, 23]
```

Создание списка чисел из строки. Использование генератора

```
a = [int(s) for s in input().split()]
print(a)
```

Пример 6.8 (пустые строки!)**.** Иногда в списке могут появляться артефакты в виде пустых строк. Часто происходит это из-за некорректного ввода (пользователь, например, ставит то один, то два пробела). Что в этом случае выбрать в качестве разделителя? И вновь простой выход — предварительная корректировка строки.

Появление пустых строк в списке

```
a = input().split('.')
print(a)
```

Результат работы программы

```
Котик. Енотик.. Обормотик котик.
['Котик', ' Енотик', '', ' Обормотик котик', '']
```

Использование метода join()

```
1 a = ['red', 'green', 'blue'] # список строк
2 print(' '.join(a))           # red green blue
3 print(''.join(a))            # redgreenblue
4 print('***'.join(a))         # red***green***blue
5 s = '-'.join(a)
6 print(s + ' new string')     # red-green-blue new string
```

Основные строковые методы

Метод	Назначение
ord(c)	перевод символа (односимвольной строки) в его код ASCII
chr(i)	перевод код ASCII в символ (односимвольную строку)
S.find(str,[start],[end])	поиск подстроки в строке, возвращает номер первого вхождения или -1
S.rfind(str,[start],[end])	поиск подстроки в строке, возвращает номер последнего вхождения или -1
S.replace(old,new)	замена всех вхождений подстроки old на подстроку new
S.split(slist)	разбиение строки по разделителю и создание списка строк slist
S.isdigit()	состоит ли строка из цифр
S.isalpha()	состоит ли строка из букв
S.isalnum()	состоит ли строка из цифр или букв

Основные строковые методы (окончание)

Метод	Назначение
`S.islower()`	состоит ли строка из символов в нижнем регистре
`S.isupper()`	состоит ли строка из символов в верхнем регистре
`S.isspace()`	состоит ли строка из неотображаемых символов
`S.istitle()`	начинаются ли слова в строке с заглавной буквы
`S.upper()`	преобразование строки к верхнему регистру
`S.lower()`	преобразование строки к нижнему регистру
`S.startswith(str)`	начинается ли строка `S` с шаблона `str`
`S.endswith(str)`	заканчивается ли строка `S` шаблоном `str`
`S.capitalize()`	преобразование первого символа строки в верхний регистр, а всех остальных в нижний
`S.count(str,[start],[end])`	возвращает количество непересекающихся вхождений подстроки в диапазоне `[start,end]` (0 и длина строки по умолчанию)
`S.lstrip([chars])`	удаление пробельных символов в начале строки
`S.rstrip([chars])`	удаление пробельных символов в конце строки
`S.strip([chars])`	удаление пробельных символов в начале и в конце строки
`S.swapcase()`	преобразование символов нижнего регистра в верхний, а верхнего — в нижний
`S.title()`	преобразование первой буквы каждого слова в верхний регистр, а всех остальных в нижний
`S.format(*args, **kwargs)`	форматирование строки

6.2 Задачи

☎ Задачи из этого раздела рекомендуется решать в двух вариантах: 1) используя только функцию `len()` и срезы; 2) используя методы строк.

Палиндром — текст, одинаково читающийся от начала к концу и от конца к началу («А роза упала на лапу Азора», А. Фет).

6.1. Описать функцию, позволяющую определить является ли введенная строка палиндромом, результат работы функции — булев.

6.2. Вводится строка, состоящая из слов, разделенных пробелами. Подсчитать количество слов в тексте.

6.3. Выяснить, является ли данный текст десятичной записью целого числа.

6.4. Написать программу для перевода целых чисел из десятичной системы счисления в систему счисления по основанию $N = 2, \ldots, 9$. Результат вычисления — строка.

6.5. Написать программу, которая принимает на входе текст, содержащий последовательность заглавных и строчных букв, затем печатает этот текст только заглавными буквами.

6.6. В заданном тексте

а) найти наибольшее количество цифр, идущих подряд;

б) определить наличие символов, отличных от букв и пробелов;

в) заменить буквы N, D и Y соответственно на M, T и U.

6.7. Напечатать названия всех чисел из диапазона $[0, 30]$.

Фрагменты печати результата

```
ноль; один; два; три; ... девять;
десять; одиннадцать; двенадцать; ... девятнадцать;
двадцать; двадцать один; ... двадцать девять;
тридцать;
```

Указания. 1) Опишите функцию `NumToStr`, возвращающую название цифры; 2) Опишите функцию `DecToStr`, возвращающую название десятичного числа из диапазона $[0, 30]$ (используйте в функцию `DecToStr` функцию `NumToStr`).

6.8. Оформить в виде функции алгоритм сложения двух целых двузначных чисел m и n, записанных в системе счисления с основанием b ($2 \leqslant b < 10$). Параметры функции: a) входные параметры — строки символов, содержащие запись целых чисел m и n в системе счисления с основанием b; b) выходной параметр — строка символов, содержащая запись результата в той же системе счисления.

6.9. Оформить в виде функции алгоритм сложения двух произвольных целых чисел m и n, записанных в системе счисления с основанием b ($2 \leqslant b < 10$). Параметры функции: a) входные параметры — строки символов, содержащие запись целых чисел m и n в системе счисления с основанием b; b) выходной параметр — строка символов, содержащая запись результата в той же системе счисления.

6.10. Напечатать в алфавитном порядке все различные латинские буквы, входящие в некоторый заданный текст.

6.11. Создать и напечатать новую строку, содержащую в алфавитном порядке все различные русские буквы, входящие в некоторый заданный текст.

6.12. Для каждого символа строки указать сколько раз он встречается в тексте.

6.13. Дана строка, состоящая из нулей и единиц. Написать и вызвать функцию для подсчета в тексте количества подстрок '101'. Например, в строке

```
1101 1010101 1011001
```

число подстрок '101' равно 5.

6.14. Дана строка, состоящая из различных цифр. Для каждой цифры подсчитать, сколько слов начинаются на эту цифру. Например, для строки

```
103 25 9   1178   23  45 99
```

ответ должен иметь вид

Результат работы программы

```
 103 25 9   1178   23  45 99
1: 2
2: 2
4: 1
9: 2
Not: 0 3 5 6 7 8
```

6.15. Написать программу для решения следующей задачи.

Дана программа для исполнителя

```
НАЧАЛО
    ПОКА нашлось(17) ИЛИ нашлось(277) ИЛИ нашлось(3777)
        заменить(17, 2)
        заменить(277, 3)
        заменить(3777, 1)
    КОНЕЦ ПОКА
КОНЕЦ
```

На вход программе подаётся строка длины 101, состоящая из цифры 2, за которой следуют 100 идущих подряд цифр 7. Какая строка получится в результате применения программы к этой строке?

Указания. Использовать методы работы со строками `replace()` и `find()`.

6.16. Написать программу для решения следующей задачи.

Дана программа для исполнителя

```
НАЧАЛО
    ПОКА нашлось(222) ИЛИ нашлось(555)
        ЕСЛИ нашлось(555)
        ТО заменить(555, 2)
        ИНАЧЕ заменить(222, 5)
    КОНЕЦ ЕСЛИ
    КОНЕЦ ПОКА
КОНЕЦ
```

На вход программе подаётся строка, состоящая из 146 идущих подряд цифр 5. Какая строка получится в результате применения программы к этой строке?

Указания. Использовать методы работы со строками `replace()` и `find()`.

6.17. Найти самое короткое слово в тексте. Если таких слов несколько, то найти последнее. Решение оформить в виде функции.

6.18. Найти самое длинное слово в тексте. Если таких слов несколько, то найти первое. Решение оформить в виде функции.

7 Одномерные списки

7.1 Простейшие операции со списками

Пример 7.1 (сложение полиномов)**.** Написать программу для сложения двух полиномов одинаковой степени. Коэффициенты полинома целые числа.

Любой полином вида

$$a_k x^k + a_{k-1} x^{k-1} + \ldots + a_1 x^1 + a_0$$

может быть описан вектором его коэффициентов $a_0, a_1, \ldots, a_k$ (роль индексов играет степень переменной x). Для реализации векторов[2] в языке Python используются одномерные списки.

Рассмотрим сложение двух полиномов одинаковой степени

$$F(x) = f_k x^k + f_{k-1} x^{k-1} + \ldots + f_1 x^1 + f_0;$$

$$G(x) = g_k x^k + g_{k-1} x^{k-1} + \ldots + g_1 x^1 + g_0.$$

Операция сложения двух полиномов сводится к сложению коэффициентов при соответствующих степенях x.

Алгоритм решения: для простоты будем считать, что полином имеет степень 5 (эту степень определим как переменную n). Полиномы будем задавать их коэффициентами. Полиномам-слагаемым и полиному-сумме соответствуют переменные f, g и h, имеющие тип `list`.

Результат вычислений будем выдавать в виде таблицы коэффициентов при соответствующих степенях x. Если оба коэффициента одновременно равны нулю, то эти строчки печатать не будем.

Для теста возьмем полиномы:

$$F(x) = 3x^2 + 2x + 3 \quad \text{и} \quad G(x) = 2x^3 + 7x^2 - 2x$$

Результат: $H(x) = F(x) + G(x) = 2x^3 + 10x^2 + 3$.

[2] В других языках программирования используется понятие «одномерный массив».

Сложение полиномов

```
n = 5
f = [3, 2, 3, 0, 0, 0]  # инициализация списка f
g = []
h = []

# печать уже заданного списка f
print('Заданы коэффициенты полинома F')
for i in range(n+1):
    print('F[', i,'] = ', f[i], sep='', end=' ')

# ввод элементов списка g и процесс сложения полиномов
print('\nВв. целые коэф. полинома g степени не выше', n)
for i in range(n+1):
    g.append(int(input('G['+ str(i) +'] = '))) # ввод эл-тов списка
    h.append(f[i] + g[i])                      # сложение полиномов

print('--------------------')

for i in range(n, -1, -1):
    if (f[i] != 0 or g[i] != 0) and h[i] != 0:
        res = str(h[i]) +'x^' + str(i)
        print('%+6s' % res)
```

Результат работы программы

```
Заданы коэффициенты полинома F
F[0] = 3 F[1] = 2 F[2] = 3 F[3] = 0 F[4] = 0 F[5] = 0
Вв. целые коэф. полинома G степени не выше 5
G[0] = 0
G[1] = -2
G[2] = 7
G[3] = 2
G[4] = 0
G[5] = 0
--------------------
  2x^3
 10x^2
  3x^0
```

7.2 Использование списков в качестве параметров подпрограмм

Пример 7.2 (использование параметра-списка)**.** Написать и использовать четыре подпрограммы:

input_vector	получение списка случайным образом
print_vector	печать списка
equal_objects	сравнение двух элементов списка
create_vector	формирование нового списка из чётных элементов исходного списка

Описания функций

```
import random

def print_vector(vector):
    for element in vector: print(element, end=' ')
    print()

def input_vector(count_elements, vector):
    for i in range(count_elements):
        x = random.randint(1, 5)
        vector.append(x)

def create_vector(vector_source):
    vector_result = []
    for element in vector_source:
        if element % 2 == 0: vector_result.append(element)
    return vector_result

def equal_objects(object_1, object_2):
    return object_1 == object_2
```

Основная часть программы

```
random.seed() # Настройка генератора случайных чисел
a = []
input_vector(5, a)
print_vector(a)
b = create_vector(a)
print_vector(b)
print(equal_objects(a[0], a[1]))
print(equal_objects(a, b))
```

Функция create_vector в качестве результата своей работы возвращает список. Эту функцию можно было записать короче, если использовать генератор списка.

Использование генератора списка

```
def create_vector(vector_source):
  return [element for element in vector_source if element % 2 == 0]
```

Функция equal_objects проверяет на равенство два любых объекта. Это могут быть не только списки или их элементы, но и просто символы, строки или другие объекты.

Примеры вызова функции equal_objects

```
print(equal_objects(a[0], a[1]))
print(equal_objects(a, b))
print(equal_objects('w', 'w'))
print(equal_objects(print_vector(a), print_vector(b)))
```

Приведенная в примере функция input_vector возвращает список через параметр функции. Ниже представлена функция, возвращающая список с помощью return.

Функция заполнения списка-2

```
import random

def input_vector(count_elements):
    vector = []
    for i in range(count_elements):
        x = random.randint(1, 5)
        vector.append(x)
    return vector

random.seed() # Настройка генератора случайных чисел
a = input_vector(5)
...
```

Пример 7.3 (поиск максимального элемента списка и его номера)**.** Найти максимальный элемент в списке и значение индекса этого элемента.

Решение. Достаточно найти индекс максимального элемента (по индексу можно восстановить значение любого элемента).

Поиск максимального элемента и его индекса

```
import random

count_elements = 5

def idx_max_elem(vector):
    if not vector: return -1
    imax = 0
    for i in range(1, len(vector)):
        if vector[i] > vector[imax]: imax = i
    return imax

random.seed()
a = [random.randint(1, 5) for i in range(count_elements)]
print(a)
print('Max = a[%d] = %d' % (idx_max_elem(a), a[idx_max_elem(a)]))

"""
Tests
"""
a = [5, 7, 12, 4, -1] # Test 1
assert idx_max_elem(a) == 2, "Error in Test 1"
print("Test 1: OK")

a = []                # Test 2
assert idx_max_elem(a) == -1, "Error in Test 2"
print("Test 2: OK")
```

Если в массиве максимальный элемент не один (например, $a_0 = 1$, $a_1 = 13$, $a_2 = 5$, $a_3 = 13$, $a_4 = 7$), то будет найден элемент с наименьшим номером (первый максимальный). С помощью условия

```
vector[i] >= vector[imax]
```

ищется последний максимальный элемент.

Функция assert (утверждать) позволяет прервать выполнение программы, если логическое условие ложно. При этом выводится сообщение, которое указано после запятой. Сообщение может и отсутствовать.

Функции, которые возвращают логическое значение (True/False) называются **предикатами**.

Пример 7.4 (использование в качестве параметра элемента списка)**.** Пусть имеются все приведенные выше описания. Напишем функцию для сравнения двух элементов списка. Результат вычисления — булев. Операция сравнения двух элементов списка ничем не отличается от операции сравнения двух чисел, т. е. вид функции может быть, например, такой

Сравнение двух элементов списка-1

```
def sravn1(x, y):
    if x == y:
        return True
    else:
        return False
```

Наиболее классический вид функции использует условный оператор.

Сравнение двух элементов списка-2

```
def sravn2(x, y):
    return x == y
```

Более компактный вид записи функции. Он является наиболее предпочтительным.

Вывод результата работы функции может быть помещен в тело программы, например:

Пример-1 вызова подпрограммы

```
print(sravn3(a[0], a[1]))
```

Сравниваются два первых элемента

Пример-2 вызова подпрограммы

```
print(sravn3(a[0], a[len(a)-1]))
```

Сравниваются первый и последний элементы

Заметим, что ошибочным при описании функции sravn мог быть следующий заголовок подпрограммы

Ошибочный заголовок функции

```
def sravn(a[i], a[j]):
```

7.3 Замена, удаление и вставка элементов

Пример 7.5 (обмен значениями двух списков)**.** Произвести обмен значениями между двумя списками.

Решение. Копирование одного списка в другой можно осуществить простым присваиванием `a = b`.

Обмен между двумя списками-1

```
a = [1, 2, 3]
b = [4, 5]
c = a
a = b
b = c
```

Обмен значениями с использованием вспомогательной переменной.

Обмен между двумя списками-2

```
a = [1, 2, 3]
b = [4, 5]
a, b = b, a
```

Обмен значениями с использованием кортежного присваивания.

Пример 7.6 (замена элементов списка по значению)**.** Заменить все отрицательные элементы списка нулями.

Замена элементов списка по значению-1

```
a = [5, -2, 16, 8, -3]
for i in range(len(a)):
    if a[i] < 0: a[i] = 0
print(a)
```

Пример классической замены.
Результат: `[5, 0, 16, 8, 0]`

Замена элементов списка по значению-2

```
def func(x):
    if x < 0: return 0
    return x

a = [5, -2, 16, 8, -3]
a = list(map(func, a))
print(a)
```

функция `map()` позволяет применить функцию `func()` ко всем элементам списка `a`.

Замена элементов списка по значению-3. Использование lambda-функции

```
a = [5, -2, 16, 8, -3]
a = list(map(lambda x: 0 if x < 0 else x, a))
print(a)
```

Пример 7.7 (замена элементов списка по номеру)**.** Поменять местами два элемента списка с номерами k1 и k2.

Замена элементов списка по номеру-1

```
a = [5, 9, -2, 4, 1]
k1 = 2
k2 = 4
Time = a[k1]
a[k1] = a[k2]
a[k2] = Time
print(a)
```

Классический алгоритм обмена с использованием вспомогательной переменной.

Результат: [5, 9, 1, 4, -2]

Замена элементов списка по номеру-2

```
a = [5, 9, -2, 4, 1]
k1, k2 = 2, 4
a[k1], a[k2] = a[k2], a[k1]
print(a)
```

Использование кортежного присваивания позволяет сократить код и избавиться от дополнительной переменной.

Пример 7.8 (удаление элемента списка по номеру)**.** Удалить элемент списка с заданным номером.

Способ 1. Удаление элемента путем сдвига с последующим отсечением последнего элемента. Похожий способ обычно используется при работе со статическими массивами в других языках программирования.

Удаление элемента списка по номеру-1

```
a = [8, 0, 3, 2, 5, 1, 4]
print(a)
nom = int(input('Введите номер элемента для удаления: '))
for i in range(nom, len(a)-1): a[i] = a[i+1]
a[len(a)-1] = 0
a.pop() # удаление последнего элемента
print(a)
```

Способ 2. Удаление элемента с использованием метода списка pop. В качестве параметра указывается индекс удаляемого элемента. Метод pop() возвращает значение удаляемого элемента списка.

Удаление элемента списка по номеру-2

```
b = [8, 0, 3, 2, 5, 1, 4]
nom = 3
b.pop(nom) # удаление элемента с номером nom
print(b)
```

Способ 3. Аналог первого способа, но отличие в использовании среза списка вместо метода pop().

Удаление элемента списка по номеру-3

```
c = [1, 2, 3, 4, 5, 6]
print(c)
nom = 4
for i in range(nom, len(c)-1): c[i] = c[i+1]
c = c[:-1] # срез списка без последнего элемента
print(c)
```

Пример 7.9 (вставка числа в список)**.** Вставить заданное число в список на место с заданным номером.

В примерах ниже попутно продемонстрированы разные способы инициализации списка случайным образом.

Способ 1. Классический сдвиг элементов вправо.

Вставка числа в список-1

```
import random
# Настраиваем генератор случайных чисел на новую последовательность.
# По умолчанию используется системное время.
random.seed()
a = [] # пустой список
n = 10 # количество элементов списка
a_min = 0 # минимальное значение элемента списка
a_max = 9 # минимальное значение элемента списка
for i in range(n):
    # генерация чисел из [a_min; a_max]
    a.append(random.randint(a_min, a_max))
print(a)

# ------------------ вставка числа в список ----------------------
x = int(input("Введите число для вставки: "))
k = int(input("Введите позицию для вставки: "))
a.append(0) # расширяем список на один элемента, добавив 0 в конец
n += 1      # увеличиваем количество элементов на один
for i in range(n-1, k, -1): # сдвигаем вправо
    a[i] = a[i-1]
a[k] = x
print(a)
```

Способ 2. Использование метода списка `insert()`.

Вставка числа в список-2

```
import random

MIN_EL = 0
MAX_EL = 9
COUNT_ELEMENTS = 10

random.seed()
a = [random.randint(MIN_EL, MAX_EL) for i in range(COUNT_ELEMENTS)]
print(a)

# ------------------ вставка числа в список ----------------------
x = int(input("Введите число для вставки: "))
k = int(input("На какую позицию будем вставлять? "))
a.insert(k, x)
print(a)
```

Способ 3. Использование среза списка.

Вставка числа в список-3

```
import random

COUNT_ELEMENTS = 10

random.seed()
a = [random.choice(range(10)) for i in range(COUNT_ELEMENTS)]
print(a)

# ------------------ вставка числа в список ----------------------
x = int(input("Введите число для вставки: "))
k = int(input("На какую позицию будем вставлять? "))
a[k:k] = [x]
print(a
```

7.4 Задачи

☎ При решении задач из этого раздела используйте списки.

7.1. Написать программу для сложения двух полиномов разной степени. Коэффициенты полиномов-слагаемых получать случайным образом.

7.2. Дано: $n \in \mathbb{N}$, $t \in \mathbb{R}$, $a_i \in \mathbb{R}$ $(i = 0, \ldots, n)$. Вычислить значение полинома

$$a_k x^k + a_{k-1} x^{k-1} + \ldots + a_1 x^1 + a_0$$

и его производной в точке t.

7.3. Дана последовательность натуральных чисел. Определить количество вхождений в неё чётных чисел.

Указание. В программе описать подпрограммы: 1) для ввода элементов списка с клавиатуры; 2) для определения количества вхождений в список четных чисел.

7.4. Дана последовательность натуральных чисел. Определить, количество элементов последовательности, отличных от её последнего элемента.

Указание. В программе описать подпрограммы: 1) для ввода элементов списка с клавиатуры; 2) для определения количества элементов списка, отличных от его последнего элемента.

7.5. Дана последовательность натуральных чисел. Определить, что больше: сумма четных элементов последовательнсти или сумма нечетных элементов.

Указания. 1) Инициализировать список в программе. 2) Описать и вызвать функцию для печати элементов списка.

7.6. Дана последовательность символов s. Создать новую последовательность p, элементы которой получить по закону $p_0 = s_9$, $p_1 = s_8$, ..., $p_8 = s_1$, $p_9 = s_0$.

Указания. 1) Инициализировать список в программе. 2) Печать элементов списка оформить в виде подпрограммы.

7.7. Дана последовательность s (все элементы — малые буквы латинского алфавита). Создать новую последовательность p, в которой будут содержаться элементы последовательности s, упорядоченные по алфавиту.

Указания. 1) Инициализировать список в программе. 2) Печать элементов списка оформить в виде подпрограммы.

Ограничение. Сортировку списка не использовать.

7.8. Дана последовательность целых чисел $a_0, \ldots, a_9$. Найти сумму и количество элементов последовательности, кратных заданному числу.

7.9. Дана последовательность целых чисел $a_0, \ldots, a_9$. Найти произведение и количество четных элементов последовательности.

7.10. Дана последовательность целых чисел $a_0, \ldots, a_9$. Найти и вывести на экран все элементы последовательности, сумма цифр в записи которых чётна.

7.11. Дана последовательность действительных чисел $a_0, \ldots, a_{10}$. Найти сумму элементов последовательности с k_1-го по k_2-ой номер (k_1, k_2 вводятся с клавиатуры).

7.12. Дана последовательность действительных чисел $a_0, \ldots, a_{10}$. Найти сумму элементов последовательности, принадлежащих промежутку от x до y (x и y вводятся с клавиатуры).

7.13. Дана последовательность натуральных чисел $a_0, \ldots, a_{10}$. Найти и вывести на экран все пары элементов последовательности, один из которых — квадрат другого.

7.14. Даны два вектора A и B с одинаковым числом элементов. Написать программу, позволяющую выполнить одну из операций, выбранных пользователем:

1 : Получение элементов векторов случайным образом;
2 : Ввод элементов векторов с клавиатуры;
3 : Вывод векторов на печать;
4 : $C = A - B$ (C — новый вектор);
5 : $B = r(A - B)$ (r — любое целое число);
6 : $A = r!A$ $(1 \leqslant r \leqslant 5)$.
7 : Скалярное произведение векторов:$d = A \cdot B$.

Указание. Все операции со списками должны быть оформлены в виде подпрограмм с параметрами. Назначение подпрограмм:

1) Получение элементов одного вектора случайным образом.
2) Ввод элементов одного вектора с клавиатуры.
3) Вывод элементов одного вектора.
4) Разность двух векторов.
5) Умножение вектора на число.
6) Скалярное произведение векторов.

Скалярное произведение векторов: $\boldsymbol{a} \cdot \boldsymbol{b} = \sum_{i=1}^{n} a_i b_i$.

7.15. Даны два вектора A и B с одинаковым числом вещественных элементов. Написать программу, позволяющую выполнить одну из операций, выбранных пользователем:

1 : Получение элементов вектора случайным образом;
2 : Ввод элементов векторов с клавиатуры;
3 : Вывод векторов на печать;
4 : $C = A + B$;
5 : $B = (A + B)p$ (p — любое действительное число);
6 : $A = -r!A$ $(1 \leqslant r \leqslant 5)$;
7 : Норма вектора:$k = \|A\|$.

Указание. Все операции со списками должны быть оформлены в виде подпрограмм с параметрами. Назначение подпрограмм:

1) Получение элементов одного вектора случайным образом.
2) Ввод элементов одного вектора с клавиатуры.
3) Вывод элементов одного вектора.
4) Сумма двух векторов.
5) Умножение вектора на число.
6) Норма вектора.

Для вычисления нормы вектора $\|\boldsymbol{a}\|$ используйте формулу:

$$\|\boldsymbol{a}\| = \sqrt{\sum_{i=1}^{n} a_i^2}.$$

7.16. Удалить из целочисленного списка все четные элементы.

7.17. Удалить из целочисленного списка все элементы, принадлежащие промежутку от x до y (x и y вводятся с клавиатуры).

7.18. Удалить из целочисленного списка все элементы с максимальным значением (предполагается, что имеется несколько таких элементов).

7.19. Удалить из целочисленного списка все элементы с минимальным значением (предполагается, что имеется несколько таких элементов).

7.20. Вставить некоторое заданное число в целочисленный список до и после элемента с заданным номером.

7.21. Вставить в целочисленный список перед каждым четным элементом цифру ноль.

До	17	2	23	2	4

После	17	0	2	23	0	2	0	4

7.22. Вставить в одномерный целочисленный список после каждого нечетного элемента цифру ноль.

До	17	2	23	2	4

После	17	0	2	23	0	2	4

7.23. В списке поменять местами первый элемент и элемент с максимальным значением (предполагается, что такой элемент один). Например,

исходный список: $(2, -3, 4, 15, -2, -6, 7)$; результат: $(15, -3, 4, 2, -2, -6, 7)$.

7.24. В списке поменять местами второй элемент и элемент с минимальным значением (предполагается, что такой элемент один). Например,

исходный список: $(2, -3, 4, 15, -2, -6, 7)$; результат: $(2, -6, 4, 15, -2, -3, 7)$.

7.25. В списке поменять местами элементы: первый и последний отрицательный. Например,

исходный список: $(2, -3, 4, 5, -2, -6, 7)$; результат: $(2, -6, 4, 5, -2, -3, 7)$.

7.26. Дан список, состоящий из $2n$ элементов. Поменять местами первую и вторую его половины.

До	a_0	a_1	a_2	a_3	a_4	a_5

После	a_3	a_4	a_5	a_0	a_1	a_2

7.27. Дан список A, состоящий из $2n$ элементов. Изменить порядок следования элементов на обратный.

До	a_0	a_1	a_2	a_3	a_4	a_5

После	a_5	a_4	a_3	a_2	a_1	a_0

7.28. Дан список A, состоящий из $2n$ элементов. Поменять местами его элементы, стоящие на четных и нечетных местах.

До	a_0	a_1	a_2	a_3	a_4	a_5

После	a_1	a_0	a_3	a_2	a_5	a_4

7.29. Изменить порядок следования чисел в списке A: сначала положительные числа, затем все остальные.

7.30. Элементы списка — целые числа (положительные и отрицательные). Расположенные рядом числа одного знака образуют серию. Найти

1) максимальную длину серии положительных чисел;

2) самую длинную серию положительных чисел.

7.31. Элементы списка — целые числа (положительные и отрицательные). Расположенные рядом числа одного знака образуют серию. Найти

1) максимальную длину серии отрицательных чисел;

2) самую длинную серию чисел одного знака.

7.32. Найти два элемента списка, модуль разности которых максимален, и поменять их местами.

7.33. Считая элементы списка a_i ($i = 0, \ldots, n-1$) координатами точек числовой прямой, найти и вывести на экран длины всех отрезков $[a_i, a_j]$.

Пример вывода результата работы программы:

```
Дан список:  -2  10  -9 -12  -3 -16
(0,1) = 12  (0,2) =  7  (0,3) = 10  (0,4) =  1  (0,5) = 14
(1,2) = 19  (1,3) = 22  (1,4) = 13  (1,5) = 26
(2,3) =  3  (2,4) =  6  (2,5) =  7
(3,4) =  9  (3,5) =  4
(4,5) = 13
```

7.34. Дан список n целых чисел A (границы изменения индексов от 0 до $n-1$). Сформировать новый список B, содержащий номера элементов списка A с заданным значением.

Например, если список A имеет следующие элементы

$$A = (13,\ 4,\ 7,\ 13,\ 5,\ -4,\ 17,\ 13),$$

то при заданном значении 13 список B будет иметь вид $B = (0,\ 3,\ 7)$.

8 Поиск элемента в списке

8.1 Линейный поиск элемента в списке

Пусть a_i ($i = 0, \ldots, n-1$) — элементы целочисленного списка; T — некоторое целое число. Выяснить, входит ли данное число T в список $a_0, \ldots, a_{n-1}$ и, если входит, то каково значение p, для которого $a_p = T$.

Линейный поиск-I заключается в переборе всех элементов списка и поочередном их сравнении с искомым элементом (в условии цикла).

Условие окончания поиска:

<элемент найден> или <достигнут конец списка>

Результат поиска: индекс искомого элемента (если он есть) или информация об отсутствии искомого элемента.

Поиск элемента в списке

```
T = int(input('Введите искомый элемент: '))
#  --------------- поиск ---------------
i = 0
while (i < len(a) and a[i] != T):
  i += 1
#  ---------- анализ результата поиска ----------
if i == len(a):
  print('Числа в списке нет')
else:
  print('Номер первого вхождения', i)
```

Поиск элемента в списке с использование метода index()

```
T = int(input('Введите искомый элемент: '))

if T not in a:
    print('Числа в списке нет')
else:
    print('Номер первого вхождения', a.index(T))
```

Линейный поиск-II заключается в добавлении в конец массива фиктивного элемента, хранящего искомое значение T. Элемент будет найден всегда. В конце поиска следует проверить, что найденный элемент не фиктивный.

Условие окончания поиска: <элемент найден>

Поиск элемента в массиве (метод фиктивного элемента)

```
T = int(input('Введите искомый элемент: '))
a.append(T)     # фиктивный элемент
#  ------------------ поиск --------------------
i = 0
while a[i] != T:
    i += 1
#  ---------- анализ результата поиска ----------
if i == len(a) - 1:
    print('Числа в списке нет')
else:
    print('Номер первого вхождения', i)
```

8.2 Задачи-I

☎ При решении задач из этого раздела используйте списки.

8.1. Элементы последовательности — буквы и цифры. Определить, верно ли, что последовательность содержит хотя бы одну цифру 5.

8.2. Элементы последовательности — натуральные числа. Определить, верно ли, что последовательность содержит хотя бы одно чётное число.

8.3. Элементы последовательности — символы латинского алфавита. Определить, верно ли, что последовательность содержит хотя бы одну гласную букву.

8.4. Элементы последовательности натуральные числа. Определить, все ли они различны.

8.5. Элементы последовательностей A и B — натуральные числа. Определить, верно ли, что каждый элемент последовательности A содержится в последовательности B.

8.6. Элементы последовательности натуральные числа. Определить, есть ли среди элементов последовательностей числа с одинаковой последней цифрой.

8.7. Элементы последовательностей натуральные числа. Определить, верно ли, что каждое число встречается не более двух раз.

8.8. Определить, есть ли в последовательности простые числа.

8.9 (★)**.** Дана последовательность натуральных чисел. Определить количество вхождений в неё каждого числа.

Результат работы программы

```
Дано:  8  21  23  10  2  8  23  23
  2 - 1 шт.
  8 - 2 шт.
 21 - 1 шт.
 23 - 3 шт.
```

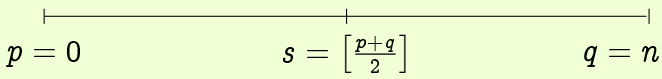
Указания. 1) Инициализировать список в программе. 2) Описать и вызвать функцию для печати элементов списка.

Ограничение. В программе использовать только один список.

8.3 Двоичный поиск элемента в массиве

Пусть a_i $(i = 0, \ldots, n-1)$ — элементы упорядоченного списка (пусть элементы упорядочены по возрастанию $a_0 < \ldots < a_{n-1}$); T — некоторое число. Выяснить, входит ли данное число T в список $a_0, \ldots, a_{n-1}$, и если входит, то каково значение p, для которого $a_p = T$.

Алгоритм деления пополам. Возьмем в качестве границ поиска индексы элементов $p = 0$ и $q = n - 1$.



До тех пор пока границы не совпадут, будем делить отрезок индексов пополам ($s = (p+q)\ //\ 2$) и шаг за шагом сдвигать границы индексов следующим образом:

$a_s < T$	p	q
да	s+1	прежнее
нет	прежнее	s

Поиск закончится, когда $p = q$. Искомый элемент будет найден, если выполняется условие

```
a[p] = T
```

Пример 8.1 (двоичный поиск элемента в упорядоченном списке)**.** Продемонстрируем метод двоичного поиска для последовательности целых чисел A, упорядоченной по возрастанию

$$a_0 = 3 \quad a_1 = 7 \quad a_2 = 8 \quad a_3 = 10 \quad a_4 = 13 \quad a_5 = 15 \quad a_6 = 16 \quad a_7 = 18$$

$$a_9 = 21 \quad a_{10} = 23 \quad a_{11} = 37 \quad a_{12} = 39 \quad a_{13} = 40 \quad a_{14} = 44 \quad a_{15} = 53.$$

Разыскиваемый элемент — 9

s	a[s]	a[s] < T	p	q	p ≠ q
			0	15	True
7	18	False	0	7	True
3	10	False	0	3	True
1	7	True	2	3	True
2	8	True	2	2	False

a[2] = 9?: нет ⇒

элемента нет

Разыскиваемый элемент — 44

s	a[s]	a[s] < T	p	q	p ≠ q
			0	15	True
7	18	True	8	15	True
11	37	True	12	15	True
13	40	True	14	15	True
14	44	False	14	14	False

a[14] = 44?: да ⇒

элемент найден, его номер 14

8.4 Задачи-II

8.10. Дан список целых чисел A, упорядоченный по возрастанию. Инициализировать список A в программе следующими значениями:

$$a_0 = 3 \quad a_1 = 7 \quad a_2 = 8 \quad a_3 = 10 \quad a_4 = 13 \quad a_5 = 15 \quad a_6 = 16 \quad a_7 = 18$$

$$a_8 = 21 \quad a_9 = 23 \quad a_{10} = 37 \quad a_{11} = 39 \quad a_{12} = 40 \quad a_{13} = 44 \quad a_{14} = 53.$$

Написать программу, реализующую алгоритм двоичного поиска элемента в списке.
Указание 1. В программе должны присутствовать операторы для построения таблицы, демонстрирующей ход поиска (здесь T — разыскиваемое значение):

Разыскиваемый элемент — ____

s	a[s]	a[s] < T	p	q	p ≠ q
…	…	…	…	…	…

a[____] = ____?: _____ (да/нет) ⇒

Результат поиска: …

Указание 2. Программа должна быть написана в двух вариантах: простом (делим отрезок до совпадения границ) и оптимальном (делим отрезок до совпадения границ или до нахождения элемента).

8.11. Дан список целых чисел A, упорядоченный по убыванию. Инициализировать список A в программе следующими значениями:

$$a_0 = 53 \quad a_1 = 44 \quad a_2 = 40 \quad a_3 = 39 \quad a_4 = 37 \quad a_5 = 23 \quad a_6 = 21 \quad a_7 = 18$$

$$a_8 = 16 \quad a_9 = 15 \quad a_{10} = 13 \quad a_{11} = 10 \quad a_{12} = 8 \quad a_{13} = 7 \quad a_{14} = 3.$$

Написать программу, реализующую алгоритм двоичного поиска элемента в массиве.
Указание 1. В программе должны присутствовать операторы для построения таблицы, демонстрирующей ход поиска (здесь T — разыскиваемое значение):

Разыскиваемый элемент — ____

s	a[s]	a[s] < T	p	q	p ≠ q
…	…	…	…	…	…

a[____] = ____?: _____ (да/нет) ⇒

Результат поиска: …

Указание 2. Программа должна быть написана в двух вариантах: простом (делим отрезок до совпадения границ) и оптимальном (делим отрезок до совпадения границ или до нахождения элемента).

9 Сортировка списков

Постановка задачи сортировки списка по возрастанию: дан числовой список $x_1, x_2, \ldots, x_n$, элементы которого попарно различны; требуется переставить элементы списка так, чтобы после перестановки они были упорядочены в порядке возрастания: $x_1 < \ldots < x_n$.

9.1 Сортировка выбором

Алгоритм сортировки выбором (I вариант):

1) найти элемент с наибольшим значением;

2) поменять значениями найденный элемент и последний в рассматриваемом подмассиве;

3) уменьшить на единицу количество просматриваемых элементов;

4) если ⟨количество элементов для следующего просмотра больше единицы⟩, то ⟨повторить пункты, начиная с 1-го⟩.

3	7	5	2	6	1
3	1	5	2	6	7
3	1	5	2	6	7
3	1	2	5	6	7
2	1	3	5	6	7
1	2	3	5	6	7

Сортировка выбором по возрастанию, I вариант

```
for i in range(len(a)-1, 0, -1):
    # пусть макс. значение у I эл-нта в подмассиве
    imax = 0
    for j in range(1, i+1):
        if a[j] > a[imax]:
            imax = j
    # обмен значений
    a[imax], a[i] = a[i], a[imax]
```

Алгоритм сортировки выбором (II вариант).

1) найти элемент с наименьшим значением;

2) поменять значениями найденный элемент и первый в рассматриваемом подмассиве;

3) сдвинуть левую границу просматриваемых элементов вправо (будет на один элемент меньше);

4) если ⟨количество элементов для следующего просмотра больше единицы⟩, то ⟨повторить пункты, начиная с 1-го⟩.

Сортировка выбором по возрастанию, II вариант

```
for i in range(0, len(a)):
    # пусть мин. значение у I эл-нта в подмассиве
    imin = i
    for j in range(i+1, len(a)):
        if a[j] < a[imin]:
            imin = j
    # перестановка элементов
    a[imin], a[i] = a[i], a[imin]
```

9.2 Сортировка обменом

Алгоритм сортировки обменом (метод пузырька). Последовательно сравниваются соседние элементы, и если выполняется неравенство

$$x_k > x_{k+1}$$

(сортировка по возрастанию), то поменять элементы x_k и x_{k+1} местами; в результате элемент с наибольшим значением окажется в конце массива.

Следующие просмотры списка применяются ко всем элементам, кроме последнего, и т. д.

Исходный список	3	7	5	2	6	1
Список после I шага	3	5	2	6	1	7
Список после II шага	3	2	5	1	6	7
Список после III шага	2	3	1	5	6	7
Список после IV шага	2	1	3	5	6	7
Список после V шага	1	2	3	5	6	7

Обозначения:
сортируемый интервал списка
отсортированный фрагмент списка.

Сортировка обменом, I вариант

```
for i in range(0, len(a)-1):
    for j in range(0, len(a)-i-1):
        # если соседи не упорядочены
        if a[j] > a[j+1]:
            # перестановка соседей
            a[j], a[j+1] = a[j+1], a[j]
```

II вариант реализации метода сортировки обменом — экономичный. Заведем переменную `sort` типа `Boolean`. Если `sort = True`, значит перестановка на данном шаге была произведена, в противном случае `sort` остается равным `False`. Если на некотором шаге не было произведено ни одной перестановки, то сортировка прекращается.

Обычный вариант

1	3	2	5	6	7
1	2	3	5	6	7
1	2	3	5	6	7
1	2	3	5	6	7
1	2	3	5	6	7
1	2	3	5	6	7

Экономичный вариант

1	3	2	5	6	7
1	2	3	5	6	7
1	2	3	5	6	7
1	2	3	5	6	7

Экономичный вариант метода сортировки обменом выгоден для частично упорядоченных списков. Иначе, он будет выполняться даже дольше обычного варианта сортировки обменом.

Сортировка обменом, II вариант

```
sort = True   #  список надо сортировать
i = 0
while sort:
    sort = False
    for j in range(0, len(a)-i-1):
        # если соседи не упорядочены
        if a[j] > a[j+1]:
            sort = True  # есть перестановка
            # перестановка соседних элементов
            a[j], a[j+1] = a[j+1], a[j]
    i += 1
```

9.3 Сортировка включением

Алгоритм сортировки включением (простыми вставками): просматривать последовательно $a_1, \ldots, a_{n-1}$ и каждый новый элемент a_i вставлять на подходящее место в уже упорядоченную последовательность $a_0, \ldots, a_{i-1}$. Это место определяется последовательным сравнением a_i с упорядоченными элементами $a_0, \ldots, a_{i-1}$.

14	**7**	3	1	11
7	14	**3**	1	11
3	7	14	**1**	11
1	3	7	14	**11**
1	3	7	11	14

Схематично каждая строка таблицы может быть представлена в виде

упорядоченные $a_0, \ldots, a_{i-1}$ элемент a_i неупорядоченные $a_{i+1}, \ldots, a_{n-1}$

Сортировка включением, I вариант

```
1 for i in range(1, len(a)):
2     y = a[i]
3     j = i - 1
4     while (j >= 0) and (y < a[j]):
5         a[j+1] = a[j]
6         j = j - 1
7     a[j+1] = y
```

9.4 Методы сортировки в Python

A.sort([key=функция]	Сортирует список на основе функции

Пример 9.1 (сортировка средствами языка Pythom)**.** Приведем несколько примеров сортировок списков встроенными средствами языка Python.

Сортировка средствами Python

```
A = [13, 7, 8, 17, 13, 15, 16, 8, 21, 3, 37, 39, 40, 4, 13]
A.sort()
print(A)
A.sort(reverse=True)
print(A)
```

Результат работы программы

```
[3, 4, 7, 8, 8, 13, 13, 13, 15, 16, 17, 21, 37, 39, 40]
[40, 39, 37, 21, 17, 16, 15, 13, 13, 13, 8, 8, 7, 4, 3]
```

Python (сортировка по длине строки)

```
A = ['a', 'kotik' , 'cccc', 'bbb']
A.sort(key=len)
print(A)
```

Результат работы программы

```
['a', 'bbb', 'cccc', 'kotik']
```

Средства Python (сортировка по алфавиту I символа)

```
A = ['ar', 'car', 'as', 'bar', 'abt', 'abg']
print('Исх. список: ', A)

# Функция для сортировки списка в алф. порядке по I символу:
def sortByAlphabet1(inputStr):
   return inputStr[0] # Ключ - I символ в каждой строке

A.sort(key = sortByAlphabet1) #
print('По I букве:  ', A)
```

Результат работы программы

```
Исх. список:  ['ar', 'car', 'as', 'bar', 'abt', 'abg']
По I букве:   ['ar', 'as', 'abt', 'abg', 'bar', 'car']
```

Средства Python (сортировки по алфавиту)

```
A = ['ar', 'car', 'as', 'bar', 'abt', 'abg']
print('Исх. список:  ', A)

# Функция для сортировки списка в алф. порядке:
def sortByAlphabet(inputStr):
    return inputStr # Ключ - вся строка

A.sort(key = sortByAlphabet)
print('Алф. порядок: ', A)
```

Результат работы программы

```
Исх. список:   ['ar', 'car', 'as', 'bar', 'abt', 'abg']
Алф. порядок:  ['abg', 'abt', 'ar', 'as', 'bar', 'car']
```

Пример 9.2 (случайная сортировка)**.** Часто требуется элементы исходного списка наоборот перемешать, т. е. установить случайный порядок элементов.

В этом случае требуется использовать в качестве ключа функцию random(), которая выдает числа в случайном порядке от 0 до 1.

Средства Python (случайная сортировка)

```
from random import random

def randomOrder_key(elem):
    return random()
```

Примеры сортировки в случайном порядке

```
a = [1, 2, 3, 4, 5, 6]
print('Исх. список:  ', a)

b = sorted(a, key = randomOrder_key)
print(b)  # [5, 3, 2, 4, 6, 1]

c = sorted(a, key = randomOrder_key)
print(c)  # [2, 1, 5, 4, 3, 6]
```

9.5 Задачи

9.1. Сортировать по возрастанию элементы списка с индексами от i_1 до i_2 включительно. Метод сортировки: 1) выбором; 2) обменом; 3) включением.

Указание. В программе должны присутствовать операторы для построения таблицы, демонстрирующей ход сортировки.

9.2. Используя любой алгоритм сортировки, расположить числа в списке в порядке убывания модулей.

Указание. В программе должны присутствовать операторы для построения таблицы, демонстрирующей ход сортировки.

10 Двумерные массивы (списки списков)

10.1 Примеры работы с двумерными массивами

Ниже приведены фрагменты кода заполнения и печати массивов.

Ввод элементов массива с клавиатуры

```
N = int(input())
a = [0] * N
for i in range(N):
    a[i] = [0] * N
    for j in range(N):
        s = 'a[' + str(i) + '][' + str(j) + '] = '
        a[i][j] = int(input(s))
```

Печать двумерного массива в виде таблицы

```
a = [[1, 2, 3], [4, 5, 6]]
for i in range(len(a)):
    for j in range(len(a[i])):
        print(a[i][j], end=' ') # печать по столбцам
    print()                     # переход на новую строчку
```

Описание и вызов функции печати 2d-массива. Вариант 1

```
def printArr(a):
    for i in range(len(a[0])):
        for j in range(len(a)): print(a[i][j], end='\t')
        print()

A = [[1, 2, 3], [4, 5, 6]]
printArr(A)
```

Напомним, что переменная цикла for в Python может перебирать любые элементы любой последовательности (в частности, списки). Используем это свойство для печати списка списков.

Описание и вызов функции печати 2d-массива. Вариант 2

```
def printArr2(a):
    for row in a:
        for elem in row:
            print(elem, end='\t')
        print()

A = [[1, 20, 3, 4], [55, 6], [7, 8, 9]]
printArr2(A)
```

Пример 10.1 (инициализация элементов заданному закону)**.** Выберем следующий способ заполнения матрицы: каждый элемент матрицы a представляет собой строку, отражающую номер строки и номер столбца матрицы, т. е.

$$a_{4\times 3} = \begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \\ a_{30} & a_{31} & a_{32} \end{pmatrix} = \begin{pmatrix} 00 & 01 & 02 \\ 10 & 11 & 12 \\ 20 & 21 & 22 \\ 30 & 31 & 32 \end{pmatrix}.$$

Инициализация элементов массива по заданному закону

```
N = 3
a = [0] * (N+1)
for i in range(len(a)):
    a[i] = [0] * N
    for j in range(len(a[i])):
        a[i][j] = str(i) + str(j)
```

Предыдущий список можно получить, используя вложенный генератор списков.

Генерация списков

```
N = 3
c = [[str(i)+str(j) for j in range(N)] for i in range(N+1)]
```

Пример 10.2 (генераторы списков)**.** Более простой пример инициализации списка списков с помощью генератора: массив 3×4 заполняется нулями.

Генерация списков

```
N = 3
M = 4
a = [[0] * M for i in range(N)]
```

Пример 10.3 (перестановка элементов матрицы)**.** Дана матрица A, состоящая из $2n \times 2n$ элементов. Поменять первую строку матрицы с последней строкой, вторую — с предпоследней и т. д.

Исходная матпица				Преобразованная матрица			
1	0	1	5	4	7	4	8
2	9	2	7	3	5	0	3
3	5	0	3	2	9	2	7
4	7	4	8	1	0	1	5

Перестановка элементов массива

```
for i in range(len(a)//2):
    M = len(a[i])
    for j in range(M):
        a[i][j], a[M-i-1][j] = a[M-i-1][j], a[i][j]
```

В данном примере можно переставлять просто строки

Перестановка строк массива

```
for i in range(len(a)//2):
    M = len(a[i])
    a[i], a[M-i-1] = a[M-i-1], a[i]
```

10.2 Задачи

10.1. Сформировать квадратную матрицу, элементы которой являются натуральными числами, расположенными по схеме:

а)
$$\begin{matrix}1&2&3\\4&5&6\\7&8&9\end{matrix}$$
б)
$$\begin{matrix}9&8&7\\6&5&4\\3&2&1\end{matrix}$$
в)
$$\begin{matrix}1&4&7\\2&5&8\\3&6&9\end{matrix}$$
г)
$$\begin{matrix}9&6&3\\8&5&2\\7&4&1\end{matrix}$$
д)
$$\begin{matrix}1&2&3\\6&5&4\\7&8&9\end{matrix}$$

(программа должна правильно работать для матриц $A_{n\times n}$, где n — любое.)

10.2. Сформировать квадратную матрицу, элементы которой являются натуральными числами, расположенными по схеме:

$$\begin{matrix}1&2&3&4&5&6\\2&1&2&3&4&5\\3&2&1&2&3&4\\4&3&2&1&2&3\\5&4&3&2&1&2\\6&5&4&3&2&1\end{matrix}$$

10.3 (★). Построить «нормальный магический квадрат».

Нормальный магический квадрат (НМК) — это матрица порядка n ($n \geqslant 1$, $n \neq 2$), заполненная натуральными числами от 1 до n^2 таким образом, что сумма чисел в каждой строке, в каждом столбце и на обеих диагоналях оказывается одинаковой.

Магический квадрат порядка 3

2	7	6	→ 15
9	5	1	→ 15
4	3	8	→ 15
↙	↓	↓	↓ ↘
15	15	15	15 15

Сумма чисел в каждой строке, столбце и на диагоналях, называется магической константой. Магическая константа НМК зависит только от n и определяется формулой

$$M(n) = \frac{n(n^2+1)}{2}.$$

Первые значения магических констант											
Порядок n	3	4	5	6	7	8	9	10	11	12	13
$M(n)$	15	34	65	111	175	260	369	505	671	870	1105

10.4. Заполнить массив размеров $(2n+1) \times (2n+1)$ по правилу:

а)

```
* * * * * * *
*       *   *
*       *   *
* * * * * * *
*       *   *
*       *   *
* * * * * * *
```

б)

```
*       *
  *   *
    * *
      *
    * *
  *   *
*       *
```

в)

```
*       *       *
  *     *     *
    *   *
* * * * * * *
    *   *
  *     *     *
*       *       *
```

10.5. Написать программу, которая формирует единичную матрицу порядка, задаваемого пользователем.

10.6. Дана квадратная матрица A, состоящая из $2n \times 2n$ элементов. Сформировать новую матрицу перестановкой блоков по схеме

1) $\begin{array}{|c|c|}\hline I & \text{II} \\ \hline \text{IV} & III \\ \hline \end{array} \longleftrightarrow \begin{array}{|c|c|}\hline III & \text{II} \\ \hline \text{IV} & I \\ \hline \end{array}$. 2) $\begin{array}{|c|c|}\hline I & \text{II} \\ \hline \text{IV} & III \\ \hline \end{array} \longleftrightarrow \begin{array}{|c|c|}\hline III & \text{IV} \\ \hline \text{II} & I \\ \hline \end{array}$

Например, для первой схемы:

Исходный массив				Преобразованный массив			
1	1	2	2	3	3	2	2
1	1	2	2	3	3	2	2
4	4	3	3	4	4	1	1
4	4	3	3	4	4	1	1

10.7. Сформировать матрицу A (3×3), элементы которой получить случайным образом $(0, \ldots, 9)$. Если среди элементов матрицы A есть нулевые, то заменить их значения на -1. На печать выдавать исходную и преобразованную матрицы.

10.8. Написать программу для сложения двух матриц (2×2), элементы которых получены случайным образом $(0, \ldots, 9)$. На печать выдавать исходные матрицы и результат сложения.

10.9. Дана целочисленная матрица A (3×3). Элементы каждой строки матрицы поделить на максимальный элемент данной строки. На печать выдавать 1) исходную матрицу; 2) вектор-столбец максимальных элементов каждой строки; 3) преобразованную матрицу.

10.10. Дана целочисленная матрица A (3×3). Элементы каждой строки матрицы A умножить на минимальный элемент данной строки. На печать выдавать 1) исходную матрицу; 2) вектор-столбец минимальных элементов каждой строки; 3) преобразованную матрицу.

10.11. Дана целочисленная матрица A ($m \times n$). Найти номер строки с максимальной суммой элементов.

10.12. Дана целочисленная матрица A ($m \times n$). Найти элемент матрицы, расположенный на пересечении строки с максимальной суммой элементов и столбца с минимальной суммой элементов.

10.13. Дана матрица A ($m \times n$), состоящая из элементов целого типа. Вычислить среднее арифметическое элементов для каждого столбца матрицы.

10.14. Дана матрица A ($m \times n$), состоящая из элементов целого типа. Вычислить среднее арифметическое элементов для каждой строки матрицы.

10.15 (★)**.** Дана целочисленная матрица A ($m \times n$). Определить, есть ли в матрице A строки, элементы которых образуют убывающую последовательность.

10.16. Поменять местами в матрице A две строки с заданными номерами (номера строк вводятся с клавиатуры).

10.17. Поменять местами в матрице A два столбца с заданными номерами (номера столбцов вводятся с клавиатуры).

10.18. В матрице вставить строку из нулей после строки с номером k.

10.19. В матрице вставить строку из нулей до строки с номером k.

10.20. В матрице вставить перед всеми строками, в которых есть нулевой элемент, первую строку.

10.21 (★)**.** В матрице вставить перед и после всех строк, в которых есть нулевой элемент, строку, состоящую из единиц.

10.22. В матрице удалить строку с номером k.

10.23. В матрице удалить столбец с минимальным элементом массива.

10.24. В матрице удалить все строки, в которых есть нулевой элемент.

10.25. В матрице удалить все столбцы, в которых есть нулевой элемент.

10.26 (★)**.** В матрице удалить все строки и столбцы, на пересечении которых стоят нулевые элементы.

10.27 (линейный поиск). Элементы квадратной матрицы A — целые числа. Проверить, является ли матрица единичной.

10.28 (линейный поиск). Элементы матртицы A — натуральные числа. Определить, все ли они имеют одинаковое значение.

10.29 (линейный поиск). Элементы матртицы A — натуральные числа. Определить, все ли они различны.

10.30. Транспонировать исходную матрицу размеров $m \times n$.

10.31. Найти норму квадратной матрицы:

$$1)\ \|A\| = \sum_{i=1}^{n} \max_{1 \leqslant j \leqslant n} |a_{ij}|; \qquad 2)\ \|A\| = \sum_{j=1}^{n} \max_{1 \leqslant i \leqslant n} |a_{ij}|;$$

$$3)\ \|A\| = \sqrt{\sum_{i,j=1}^{n} a_{ij} \cdot a_{ji}}.$$

10.32. Найти определитель матрицы третьего порядка.

10.33. Найти произведение вектора $\boldsymbol{b} = (x_1, \ldots, x_n)$ и матрицы $A_{n \times m}$.

Произведение вектора-строки $\boldsymbol{b}_n$ и матрицы $A_{n \times m}$ — вектор-строка $\boldsymbol{c}_m$, компоненты которого вычисляются по формуле

$$c_j = \sum_{i=1}^{n} b_i a_{ij}, \quad j = 1, \ldots, m.$$

Например,

$$\begin{pmatrix} 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} 1 & 5 \\ 0 & 6 \\ 4 & 7 \end{pmatrix} =$$

$$= \begin{pmatrix} 1 \cdot 1 + 2 \cdot 0 + 3 \cdot 4 & 1 \cdot 5 + 2 \cdot 6 + 3 \cdot 7 \end{pmatrix} =$$

$$= \begin{pmatrix} 13 & 38 \end{pmatrix}.$$

10.34. Найти произведение матрицы $A_{n\times m}$ и вектора $\boldsymbol{b} = (x_1, \ldots, x_m)$.

Произведение матрицы $A_{n\times m}$ и вектора-столбца $\boldsymbol{b}_m$ — вектор-столбец $\boldsymbol{c}_n$, компоненты которого вычисляются по формуле

$$c_i = \sum_{j=1}^{m} a_{ij} b_j, \quad i = 1, \ldots, n.$$

Например,

$$\begin{pmatrix} 5 & 6 & 7 \\ 1 & 0 & 4 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 5\cdot 1 + 6\cdot 2 + 7\cdot 3 \\ 1\cdot 1 + 0\cdot 2 + 4\cdot 3 \end{pmatrix} = \begin{pmatrix} 38 \\ 13 \end{pmatrix}.$$

10.35. Найти произведение вектора $\boldsymbol{x} = (x_1, \ldots, x_n)$ и квадратной матрицы $A_{n\times n}$. В полученном векторе найти номера максимального и минимального элементов.

10.36. Найти произведение квадратной матрицы A порядка n и вектора $\boldsymbol{x} = (x_1, \ldots, x_n)$. В полученном векторе найти максимальную разность элементов.

10.37. Найти произведение двух матриц $A_{n\times m}$ и $B_{m\times p}$.

Произведением прямоугольной матрицы A размеров $n \times m$ на матрицу B размеров $m \times p$ называется прямоугольная матрица $C_{n\times p}$:

$$A_{n\times m} \times B_{m\times p} = C_{n\times p}.$$

каждый элемент которой вычисляется по формуле:

$$c_{ik} = \sum_{s=1}^{m} a_{is} \cdot b_{sk}, \quad i = 1, \ldots, n, \quad k = 1, \ldots, p,$$

т. е. элемент произведения, расположенный в i-й строке, в k-м столбце равен сумме произведений элементов i-й строки первого сомножителя на соответственные элементы k-го столбца второго сомножителя.

Произведение матриц определено, если число столбцов первого сомножителя равно числу строк второго.

Будем говорить, что матрица A имеет седловую точку a_{ij}, если a_{ij} является минимальным элементом в i-й строке и максимальным в j-м столбце.

Например, матрица

$$A = \begin{pmatrix} 5 & 3 & 4 \\ 6 & 2 & 3 \\ 11 & 1 & 3 \end{pmatrix}$$

имеет седловую точку $a_{12} = 3$ (в первой строке, втором столбце).

10.38 (★)**.** Дана целочисленная прямоугольная матрица. Определить номер строки и номер столбца любой седловой точки матрицы.

10.39 (★)**.** Дана целочисленная прямоугольная матрица. Определить номера строк и столбцов всех седловых точек матрицы.

11 Множества

Множество состоит из различных элементов, порядок которых в множестве не определен.

Элементы множества: данные неизменяемых типов (числа, строки, кортежи).

11.1 Задание множеств и функция set()

Пример 11.1 (задание непустого множества)**.** Множество задается перечислением всех его элементов в фигурных скобках:

```
A = {1, 2, 3}.
```

Каждый элемент может входить в множество только один раз. Если задать множество следующим образом

```
A = {3, 1, 2, 3, 3, 2, 2},
```

то будет создано множество `{1, 2, 3}`.

Пример 11.2 (задание пустого множества и функция `set()`)**.** Функция `set()` используется для создания множества.

Пустое множество задается следующим образом

```
A = set().
```

Создание множества	Пример	Результат
из строки	`A = set('racoon')`	множество из 5 элементов
из списка	`A = set([1, 6, 7, 2, 1])`	`{1, 2, 6, 7}`

Пример 11.3 (использование генератора для создания множества)**.** Создать множество целых чисел от 0 до $n-1$ можно с помощью генератора

```
d = {i for i in range(n)}
```

11.2 Основные приемы работы с данными множественного типа

Количество элементов в множестве определяется с помощью функции `len()`.

Для проверки принадлежности элемента множеству используется зарезервированное слово `in`:

Демонстрация len() и in

```
b = {1, 3, 3, 5, 9}
print('кол-во элементов: ', len(b))               # результат 4
print('наличие элемента 4 в b: ', 4 in b)         # результат False
print('отсутствие элемента 4 в b: ', not 4 in b)  # результат True
print('отсутствие элемента 3 в b: ', 3 not in b)  # результат False
```

Множества легко вывести на экран (порядок вывода элементов в множестве может быть непредсказуем)

Печать множеств

```
a = set()
b = {1, 3, 3, 5, 9}
c = set('hello world!')
print(a, b, c, sep='\n')
```

Результат-1

```
set()
{1, 3, 5, 9}
{'!', 'h', 'o', 'r', 'l', 'w', ' ', 'd', 'e'}
```

Результат-2

```
set()
{1, 3, 5, 9}
{'d', ' ', 'l', 'h', 'o', 'e', 'r', '!', 'w'}
```

Пример 11.4 (поэлементная печать множества)**.** Для печати элементов множества можно использовать цикл for:

Печать множеств

```
word = set('programming')
for c in word:
    print(c, end='')
```

Примеры результатов

```
armpgnio  # первый результат
prgonmai  # второй результат
```

Пример 11.5 (преобразование множеств)**.** Множества можно преобразовывать в строку, список или кортеж, используя соответствующие методы.

Преобразование множества к другому типу

```
Set = set('programming')  # множество
List = list(Set)          # список
Tuple = tuple(Set)        # кортеж
Str = str(Set)            # строка
```

Методы добавления, удаления элементов из множества

Метод	Описание
add(x)	добавление элемента x в множество
discard(x)	удаление элемента x из множества
remove(x)	удаление элемента x из множества. Если удаляемый элемент отсутствует в множестве, то генерируется исключение KeyError
pop()	удаляет из множества первый элемент и возвращает его значение. Если множество пусто, то генерируется исключение KeyError. Примечание: так как множество не упорядочено, какой элемент будет первым, неизвестно
clear()	очистка множества

Пример 11.6 (добавление элемента в список и очистка списка)**.** Метод add() добавляет элемент в множество.

Добавление элемента в список и очистка списка

```
s = {1, 2, 10}
print(s)  # {1, 2, 10}
s.add(5)
print(s)  # {1, 2, 10, 5}
s.add(7)
print(s)  # {1, 2, 5, 7, 10}
s.clear()
print(s)  # set()
```

Пример 11.7 (удаление случайного элемента)**.** Использование метода pop() позволяет не только удалить элемент из множества, но и запомнить значение удаленного элемента. Если применить метод pop() к пустому списку, то генерируется исключение KeyError — несуществующий ключ (во множестве, в данном случае). Для обработки исключений используется конструкция try-except.

Удаление случайного элемента

```
s1 = {-1, 2, 10}
print(s1)    # {2, 10, -1}
s1.pop()
print(s1)    # {10, -1}
d = s1.pop()
print(s1, 'удалено число', d)    # {-1} удалено число 10
# обработка исключения, если происходит удаление из пустого списка
s1 = {}
try:
    s1.pop()        # инструкция, которая может породить исключение
except Exception:
    print('Error')  # перехват исключения
print(s1)    # {}
```

Пример 11.8 (сравнение методов удаления элемента из списка)**.** Методы discard(x) и remove(x) удаляют элемент x из множества. Метод remove(x) генерирует исключение KeyError, при попытке удалить отсутствующий элемент.

Удаление элемента из списка

```
s = {1, 2, 10}
s.discard(2)
print(s)     # {1, 10}
s.discard(5) # попытка удалить несущ-щий элемент ошибку не вызывает
print(s)     # {1, 10}
#----------------------------------------
s = {1, 2, 10}
s.remove(2)
print(s)     # {1, 10}
'''
s.remove(5)  # попытка удалить несущ-щий элемент вызывает ошибку
print(s)
'''
```

Функции с логическим результатом

`b.isdisjoint(c)`	проверяет нет ли пересечения множеств (True, если нет)
`b.issubset(c)` `b <= c`	$b \subset c$ (True, если все элементы b принадлежат c)
`b.issuperset(c)` `b >= c`	$b \supset c$ (True, если все элементы c принадлежат b)

Пример 11.9 (демонстрация методов isdisjoint(), issubset(), issuperset()). Продемонстрируем работу методов `isdisjoint()`, `issubset()`, `issuperset()`.

Демонстрация методов isdisjoint(), issubset(), issuperset()

```
A = {1, 2, 10}
B = {1, 12, 100}
C = {11,14}
print(A.isdisjoint(B))  # False. Нет ли пересечения?
print(A.isdisjoint(C))  # True
#----------------------------------------
B = {1, 12, 100}
C = {1, 100}
print(B.issubset(C))    # False. Все элементы b принадлежат c
print(B.issuperset(C))  # True. Все элементы c принадлежат b
```

Объединение, пересечение, разность множеств

`a.update(b)`	добавление элементов множества b в a
`F = b.union(c, d)` `F = b | c | d)`	объединение множеств ($F = b \cup c \cup d$).
`F = b.intersection(c, d)` `F = b & c & d`	пересечение ($F = b \cap c \cap d$).
`F = A.difference(B)` `F = A - B`	разность множеств A и B (элементы, входящие в A, но не входящие в B)

Пример 11.10 (метод update()). Метод `update()` служит для добавления элементов одного множества в другое.

Демонстрация метода update()

```
A = {'a', 'c'}
B = {1, 12, 100}
A.update(B)  # добавление элементов множества B в A
print(A)     # {1, 'c', 100, 'a', 12}
```

Пример 11.11 (объединение, пересечение и разность множеств)**.** Продемонстрируем работу методов для объединения, пересечения нескольких множеств, а также разности множеств.

Объединение множеств

```
A = {'a', 'c'}
B = {1, 12, 100}
C = {1, 100}
D = {-5, -7}
S1 = B.union(C, D)
print(S1)           # {1, 100, -7, -5, 12}
S11 = set.union(B, C, D)
print(S11)          # {1, 100, -7, -5, 12}
```

Пересечение множеств и разность множеств

```
A = {12, 5, 100, -3}
B = {1, 12, 100}
C = {1, 100}
D = {-5, -7, 1}
P = B.intersection(C, D) # пересечение (или P = B & C & D)
print(P) # {1}
#----------------------------------------
R = A.difference(B) # разность (или R = A - B)
print(R) # {5, -3}
```

11.3 Задачи

11.1. Написать программу, выделяющую из некоторого заданного множества подмножество четных чисел.

11.2. Создать множество согласных букв, входящих в строку.

11.3. Дано множество символов, состоящее из различных строчных латинских букв. Напечатать элементы данного множества в алфавитном порядке.

11.4. Даны два списка с одинаковым количеством элементов. Выяснить, верно ли, что оба массива отличаются не более, чем порядком следования чисел.

Указание. Постройте множества, состоящие из элементов списков.

11.5. Из двух исходных списков составить новый список, содержащий только общие элементы (без повторений).

11.6. Используя алгоритм «решето Эратосфена», найти все простые числа из промежутка $[2, n]$.

Алгоритм:

1) создать исходное множество натуральных чисел;
2) создать пустое множество простых чисел;
3) в переменную `Next` поместить первое простое число;
4) пока ⟨исходное множество не пусто⟩ делать
   4.1) добавить в множество простых чисел текущее простое число `Next`;
   4.2) удалить из множества натуральных чисел все кратные `Next`;
   4.3) получить следующее простое число — новое значение `Next` (первое, оставшееся в исходном множестве после выполнения 4.2).

11.7. Используя алгоритм «решето Эратосфена», найти и напечатать в порядке убывания все простые числа из промежутка $[m, n]$.

---

12 Приложение 1. Математический пакет Maple

Пакет Maple — интерактивная программа, позволяющая в диалоговом режиме проводить аналитические выкладки и вычисления. Несмотря на то, что пакет имеет свой язык программирования, в данном курсе этот аспект рассматриваться не будет.

Представленный материал позволит научиться проводить простейшие вычисления и строить графики функций, используя программу Maple.

12.1 Начало работы с пакетом Maple

12.1. Используйте классический вариант программы Maple. Значок приложения Maple 11 (Classic worksheet) имеет вид:

Внешне Maple очень похож на текстовый редактор. Вся работа происходит в окне с документом, который может содержать формулы, рисунки, текст, комментарии и проч. Набор и редактирование Maple-команд происходят обычным образом с помощью клавиатуры и мыши.

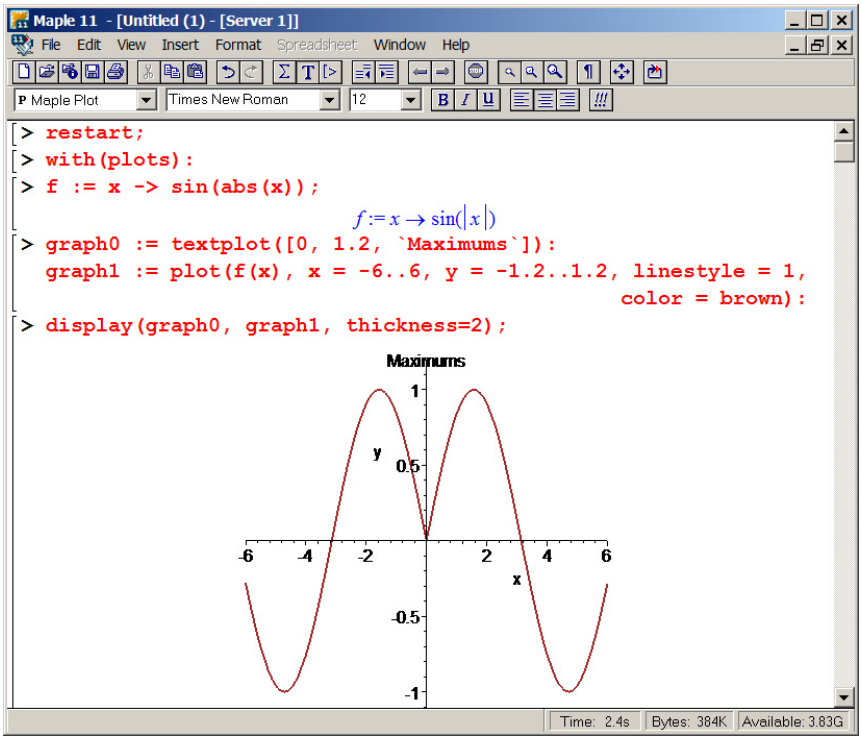


Рис. 5. Окно Maple 11 (Classic Worksheet)

Работа в Maple проходит в режиме сессии — пользователь вводит команды, которые обрабатываются Maple. Команды набираются после приглашения ([>). Каждая команда должна завершаться разделителем: «;» или «:». В первом случае после нажатия клавиши Enter будет выведен результат исполнения команды или сообщение об ошибке, во втором случае команда выполняется, но результат не выводится.

Основные объекты

Объектами Maple могут быть числа, строки, имена переменных, команды, например,

```
345;  2.718281828;  3.8e4;  1.8e-3;  `Maple`;  Pi;  restart:
```

Запись числа в виде `271.8` называется записью числа с фиксированной точкой, а запись `2.718e2` называется записью числа с плавающей точкой и означает $2{,}718 \cdot 10^2$. Числа с плавающей точкой имеют мантиссу (число с фиксированной точкой) и масштабный множитель (порядок).

Экспоненциальную форму записи используют, как правило, при записи слишком малых или слишком больших чисел.

Оператор

```
Digits := n
```

задает длину мантиссы для операций с плавающей запятой.

В Maple есть все основные математические константы, приведем лишь некоторые из них

Имя константы	Описание
`Pi`	Число π
`I`	Мнимая единица
`Infinity`	Бесконечность

Имена этих констант являются зарезервированными, а их значения не могут быть переопределены в отличие от ряда управляющих констант (например, `Digits`, см. с. 129).

Строкой (`string`) является любой набор символов, заключенный в обратные кавычки (левая верхняя клавиша на клавиатуре): `` `Maple` ``.

 **12.2.** В некоторых версиях Maple для отображения строк используются двойные кавычки.

В Maple различаются малые и заглавные буквы

`sin(Pi);` `sin(pi);` (результат 0, $\sin(\pi)$).

Каждая переменная Maple имеет имя — набор символов (букв, цифр, знака подчеркивания), начинающийся с буквы, например,

`a; A; n1; newvalue; new_value; new_Value;`

В качестве имен переменных запрещено использовать слова Maple-языка, например, `and`, `do`, `then`, `to` …

Проверка совпадения с существующим зарезервированным словом:

`?name`

(здесь `name` — проверяемое имя переменной).

Если имя переменной совпадает с каким-либо зарезервированным словом, то Maple выдаст соответствующую страницу справочной системы (см. рис. 6).

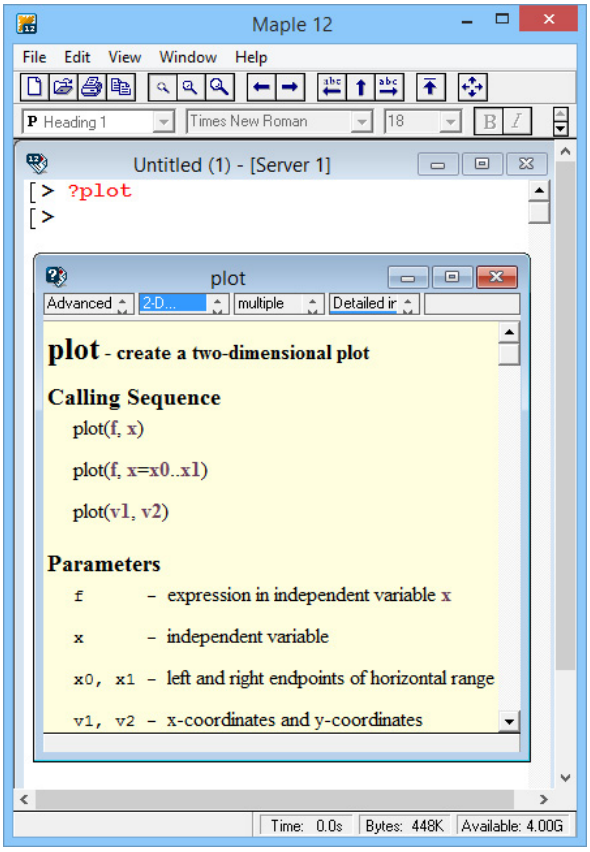


Рис. 6. Страница справки

Скобки, применяемые в Maple		
	Область применения	Пример использования

( )	задают порядок выполнения операций в выражениях; обрамляют аргументы функций и параметры в записи команд;	`(a+b)*c+d;`
		`cos(0);`
		`with(plots):`

[ ]	для работы с индексными величинами;	`a[i]+b[i]; A[i,j];`
{ }	для формирования множеств, систем уравнений;	`solve({x+y=2, 3*x=y},{x,y});`

Знаки операций					
+	сложение	134+765	/	деление	13.4/7
-	вычитание	134-765	^	возведение в степень	3^7
*	умножение	13.4*7	!	факториал	4!

Порядок выполнения арифметических операций соответствует стандартным математическим правилам.

Знаки операций отношения					
>	больше	<=	меньше либо равно	=	равно
<	меньше	>=	больше либо равно	<>	не равно

Некоторые стандартные функции	
e^x	`exp(x)`
$\log_a x$	`log[a](x)`
$\ln x$	`ln(x)` или `log(x)`
$\lg x$	`log10(x)`
$\sqrt{x}$	`sqrt(x)`
$\lvert x \rvert$	`abs(x)`
$\operatorname{sgn} x$	`signum(x)`
$\sin(x)$; $\cos(x)$; $\operatorname{tg}(x)$; $\operatorname{ctg}(x)$	`sin(x); cos(x); tan(x); cot(x)`
$\arcsin(x)$; $\arccos(x)$	`arcsin(x); arccos(x)`
$\operatorname{arctg}(x)$; $\operatorname{arcctg}(x)$	`arctan(x); arccot(x)`

Каждая функция имеет аргумент, который заключается в круглые скобки. В качестве аргументов функции могут указываться числа, константы, имена переменных, арифметические выражения и проч.

Из констант, переменных, знаков арифметических операций, имен функций и скобок составляются выражения.

Пример 12.1 (запись выражений)**.** Выражение

$$\frac{(1.4+7.65)(13.4-5.45)}{1.4}\,3!$$

в Maple запишется в виде

Строка в окне Maple

```
[> (1.4 + 7.65) * (13.4 - 5.45) / 1.4 * 3!;
```

Знак # служит для вставки комментария. Текст строки после этого символа игнорируется.

Например, дробь $\frac{a+b}{ab}$ может быть представлена в виде

Строка в окне Maple

```
[> (a + b) / a / b; # лучше делить последовательно
```

Пример 12.2 (непривычное представление выражений)**.** Выражения

$\sin^2(x+10)$ — квадрат функции; $\sin(x+10)^2$ — квадрат аргумента функции

в Maple можно записать в виде

Строка в окне Maple

```
[> (sin(x+10))^2;
[> sin((x+10)^2);
```

На печати результат выглядит непривычно

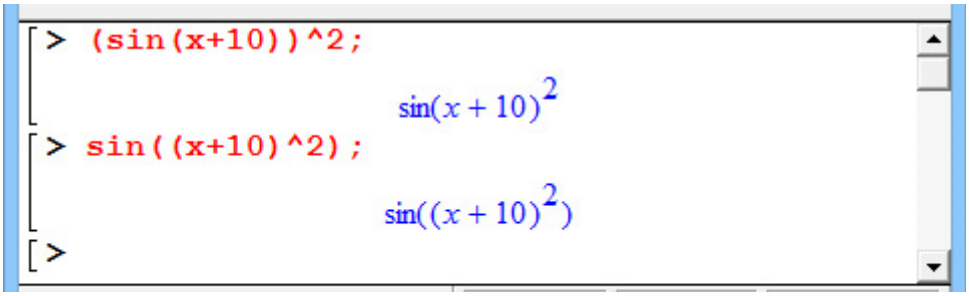


Рис. 7. Непривычное представление выражений

Оператор присваивания

В результате выполнения оператора присваивания (:=) переменной, стоящей слева от этого знака, присваивается значение некоторого выражения (стоящего справа от знака :=). Например,

Операторы присваивания

```
a := 0;    b := c;
x1 := b * b - 4 * a * c;    x := x1;
```

12.3. Обратите внимание на способ расстановки пробелов в приведенном примере. Желательно каждый оператор (выражение, команду) изображать в редакторе Maple в отдельной строке. Если в одной строке стоит более одного оператора, то между ними следует вставлять не менее двух пробелов.

Пример 12.3 (использование знаков = и :=)**.** Присвоим переменной d сумму чисел 3 и 8. Приравняем Sum0 значение переменной d:

Знаки := и =

```
[> d := 3 + 8;    Sum0 = d;
```

Выдадим на печать значения d и Sum0:

Получение значений переменных

```
[> d;
[> Sum0;
```

Результат

$$11$$

$$Sum0$$

Переменная Sum0 оказалась без значения, так как вместо знака присваивания был использован знак равенства.

Для отмены всех сделанных присваиваний и начала нового сеанса без выхода из программы Maple используется команда restart.

Как видно из примера 12.3, для выдачи значения любой скалярной переменной достаточно в строке ввода указать имя самой переменной.

Для просмотра содержимого индексных переменных (например, вектора с именем Expr) используется команда eval(Expr).

Команда evalm(Expr) вычисляет матричное выражение. В выражении используются матрицы в качестве операндов и некоторые операции (см. пример 12.4).

Пример 12.4 (использование команд eval и evalm)**.** Для вывода на экран матрицы или вектора используется команда eval. Команда evalm здесь вычисляет произведение вектора-строки и матрицы (количество элементов вектора должно совпадать с числом строк матрицы) и сумму двух матриц.

```
> a := array(1..3, 1..2, [[1,5], [0,6], [4,7]]):
> b := array(1..3, [1,2,3]):
> c := array(1..3, 1..2, [[1,1], [2,2], [3,4]]):
> eval(a);
                        [1  5]
                        [0  6]
                        [4  7]
> eval(b);
                      [1, 2, 3]
> eval(c);
                        [1  1]
                        [2  2]
                        [3  4]
> evalm(b &* a);   # умножение вектора на матрицу
                      [13, 38]
> evalm(a + c);    # сумма двух матриц
                        [2   6]
                        [2   8]
                        [7  11]
```

Рис. 8. Вывод векторов и матриц

Две идущие подряд точки (..) в опциях команд применяются для определения интервала изменения переменных.

Вычисление пределов

Команды для вычисления пределов

```
Limit(Expr, x = Val, Dir);  и  limit(Expr, x = Val, Dir);
```

Здесь `Expr` — выражение, для которого вычисляется предел (функция или n-й член последовательности); `x = Val` — значение точки, в которой вычисляется предел; `Dir` — необязательный параметр, который может принимать значения: `left` (предел слева), `right` — предел справа, `real` (действительный) или `complex` (комплексный).

Сравните результат работы операторов

```
[> Limit(exp(x), x = infinity);
[> limit(exp(x), x = infinity);
[> Limit((x-1)/x, x = infinity);   value(%);
[> Limit((x-1)/x, x = infinity):   % = value(%);
```

Команда `value(Expr)` вычисляет алгебраическое выражение с именем `Expr`. В упражнении в качестве выражения команды `value` используется знак `%`, который означает, что требуется найти значение предыдущего выражения (предела в данном случае).

Служебное слово `infinity` служит для обозначения бесконечности (символ ∞).

Пример 12.5 (вычисление предела)**.** Предел $\lim\limits_{x\to\infty}\left(1+\frac{1}{x}\right)^x$ равен числу e. Соответствующая команда для его вычисления:

Вычисление предела

```
[> Limit((1 + 1 / x)^x, x = infinity): % = evalf(%);
```

Для получения значения величины e здесь использована команда

`evalf(Expr)`,

которая приводит выражение `Expr` к форме записи с фиксированной запятой или с плавающей запятой в зависимости от величины результата.

Результат будет иметь вид

$$\lim_{x\to\infty}\left(1+\frac{1}{x}\right)^x = 2.718281828 \qquad (12.1)$$

Суммирование

Сумму некоторых элементов можно получить с помощью команды

```
sum(s, k = m..n);
```

здесь s — суммируемое выражение (число или имя переменной в простейшем случае); k — диапазон изменения индекса суммирования (от m до n, где m и n, вообще говоря, целые).

Например, $x + x^2 + x^3 + x^4 + x^5$, можно получить командой

$x + x^2 + x^3 + x^4 + x^5$

```
[> sum(x^k, k = 1..5);
```

Для вывода суммы в виде $\sum_{k=1}^{5} x^k$ используется команда Sum.

Пример 12.6 (вычисление суммы)**.** Равенство

Информативный вывод

```
[> Sum(x^k, k = 1..5) = sum(x^k, k = 1..5);
```

позволит получить на экране сумму в виде

$$\sum_{k=1}^{5} x^k = x + x^2 + x^3 + x^4 + x^5. \qquad (12.2)$$

12.4. Подобно суммированию можно записывать и произведение некоторых выражений, например,

$$\prod_{k=1}^{3} (x + k) = (x + 1)(x + 2)(x + 3).$$

Соответствующие Maple-команды: `Product()` (`product()`).

12.2 Дифференцирование и интегрирование

Дифференцирование

Команды для нахождения производных

```
Diff(Expr, x);  и  diff(Expr, x);
```

Здесь Expr — выражение, от которого вычисляется производная; x — переменная, по которой происходит дифференцирование.

Сравните результаты команд Diff и diff

```
[> Diff(x^3/3, x);   value(%);
[> diff(x^3/3, x);
[> diff(x^3/3, x$2);
```

В последнем выражении после знака $ указан порядок дифференцирования. В данном случае будет вычислена производная второго порядка.

Производную второго порядка $f''(x)$ можно найти и просто продифференцировав $f'(x)$. Например,

```
[> df := diff(x^3/3, x);  diff(df, x);
```

Здесь сначала нашли первую производную и присвоили ее значение переменной df, а затем продифференцировали df один раз по x.

Много операторов в одной командной строке

```
[> f := x -> x^(3/5);
   Diff(f(x), x);
   value(%);
   subs(x = 0.5, %);
```

Переход на новую строку без выполнения команды:

<Shift> + <Enter>.

Команда subs позволяет получить результат алгебраического выражения при конкретном значении x.

Интегрирование

Команды для нахождения неопределенных интегралов

`Int(Expr, x)` и `int(Expr, x)`

Здесь `Expr` — интегрируемое выражение; `x` — переменная интегрирования.

Команды для нахождения определенных интегралов

`Int(Expr, x = a..b)` и `int(Expr, x = a..b)`

Здесь `a`, `b` — отрезок интегрирования.

Пример 12.7 (нахождение определенного интеграла)**.** Вычислить интеграл

$$\int_0^{\pi} \frac{dx}{3 + 2\cos x}.$$

Подынтегральную функцию

$$f(x) = \frac{1}{3 + 2\cos x}$$

зададим следующим образом:

Задание функции

```
[> f := x -> 1/(3 + 2*cos(x));
```

Проинтегрировав функцию $f(x)$ на отрезке $[0, \pi]$

Интегрирование функции на отрезке

```
[> Int(f(x), x = 0..Pi):   % = evalf(%);
```

получим

$$\int_0^{\pi} \frac{dx}{3 + 2\cos x} = 1.404962946 \qquad (12.3)$$

Численное интегрирование

Численный метод	Maple-оператор
средних прямоугольников	`middlesum()`, `middlebox()`
левых прямоугольников	`leftsum()`, `leftbox()`
правых прямоугольников	`rightsum()`, `rightbox()`
трапеций	`trapezoid()`

Maple: интегрирование (численный метод)

```
[> with(student);
[> f := x -> ...;                          # подынтегральная функция f(x)
[> a := ...;   b := ...;                   # пределы интегрирования
[> n := 2;                                 # число прямоугольников
[> evalf(middlesum(f(x), x=a..b, n)); # результат: число
[> middlebox(f(x), x=a..b, n);         # результат: рисунок
```

12.3 Графика в Maple. Элементарное введение

Команда plots

Простейшая команда для построения графика функции одной переменной:

```
plot(func1, func2, ..., x = a..b, y = c..d, options)
```

здесь `func1`, `func2`,... — выражения, зависящие от переменной x (если строится график одной функции, то фигурные скобки можно не указывать); `a..b` — интервал изменения переменной x; `c..d` — выводимый интервал по оси ординат; `options` — опции, меняющие свойства графика (необязательный параметр, см. таблицу на с. 139).

Некоторые опции двумерной графики

`title = 'Name'`	заголовок рисунка
`coords = polar`	полярные координаты (по умолчанию — декартовы)
`style = line/point`	стиль вывода графика — линиями или точками
`color = colorvalue`	цвет выводимых линий (`black`, `blue`, `red`, ...)
`thickness = n`	толщина линии ($n = 1, 2, \ldots$)
`linestyle = n`	($n = 1, 2, \ldots$) тип выводимой линии (непрерывная $n = 1$; пунктирная).

Пример 12.8 (построение графика функции). Команда

График функции $\cos x$, $x \in [-2\pi, 2\pi]$

```
[> plot(cos(x), x = -2*Pi..2*Pi);
```

позволяет построить график функции $\cos x$ на интервале $[-2\pi, 2\pi]$. Область вывода графика по оси ординат выбирается автоматически, также как и цвет линии.

Удобно до вызова команды `plot` определить функцию и пределы для вывода графика по оси абсцисс (команда plot в этом случае будет иметь универсальную запись):

```
[> f := x -> cos(x):   t := 2 * Pi:
[> plot(f(x), x = -t..t);
```

Пример 12.9 (построение двух графиков в одной области вывода). Команда для построения графиков функций $x/2$ и $\sin|x|$ на интервале $[-6, 6]$ (область вывода по оси ординат $[-2, 2]$) имеет вид

Графики функций $x/2$, $\sin|x|$, $x \in [-6, 6]$, $y \in [-2, 2]$

```
[> f := x -> x / 2:
   g := x -> sin(abs(x)):
[> plot({f(x), g(x)}, x = -6..6, y = -2..2, color = [blue, red]);
```

Здесь дополнительно установлен цвет графиков (синий и красный).

Команда display

Более удобной для вывода графических образов является команда

```
display([pic1, pic2, ...], options)
```

Здесь образы `pic1, pic2, ...` выводятся на одном рисунке в общих осях координат.

Перед использованием команды вывода `display` необходимо подключить пакет `plots` (см. пример 12.10) и определить переменные с именами `pic1`, `pic2`,... через соответствующие команды для построения графиков.

Команда для подключения пакета имеет вид

```
with(package):
```

где `package` — имя подключаемого пакета. Например, пакет `plots` содержит команды для расширенной работы с графикой.

Пример 12.10 (использование команды display)**.** Построение графиков функций $y = \sin x$ и $y = \cos x$ с использованием команды display.

Использование команды display

```
[> f := x -> sin(x):
[> g := x -> cos(x):
[> graph01 := textplot([3.1, 0.8, 'y = sin(x)', color = brown]):
[> graph02 := textplot([-1.5, 0.8, 'y = cos(x)', color = blue]):
[> graph1 := plot(f(x), x = -2*Pi..2*Pi, y = -1.2..1.2,
                  linestyle = 3, color = brown):
[> graph2 := plot(g(x), x = -2*Pi..2*Pi, y = -1.2..1.2,
                  linestyle = 1, color = blue):
[> display(graph01, graph02, graph1, graph2, thickness=2);
```

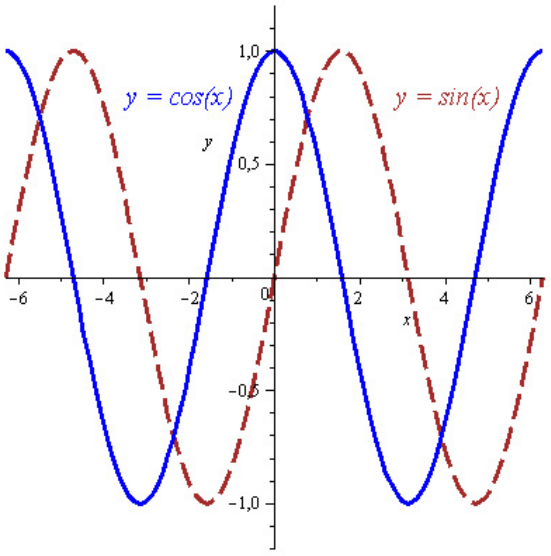


Рис. 9. Результат построения

Команда `textplot([X, Y, 'Текст'])` позволяет поместить в точке с координатами X, Y любую текстовую строку.

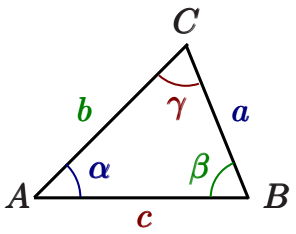
13 Приложения

13.1 Немного математики

Теорема косинусов:

$$a^2 = b^2 + c^2 - 2 \cdot b \cdot c \cdot \cos\alpha,$$

α — угол, противолежащий стороне a (см. рис.).



Теорема синусов:

$$\frac{a}{\sin\alpha} = \frac{b}{\sin\beta} = \frac{c}{\sin\gamma}.$$

Гиперболические функции (синус и косинус):

$$\operatorname{sh} x = \frac{e^x - e^{-x}}{2}, \qquad \operatorname{ch} x = \frac{e^x + e^{-x}}{2}.$$

Замена основания логарифма

$$\log_a x = \frac{\ln x}{\ln a}, \quad \log_a b = \frac{1}{\log_b a}.$$

Обратные тригонометрические функции

$$\arcsin(\sin y) = y, \quad -\frac{\pi}{2} \leqslant y \leqslant \frac{\pi}{2}; \qquad \arccos(\cos y) = y, \quad 0 \leqslant y \leqslant \pi,$$

$$\operatorname{arcctg} x + \operatorname{arctg} x = \frac{\pi}{2}, \quad \arcsin x = \operatorname{arctg} \frac{x}{\sqrt{1-x^2}}, \quad \operatorname{arctg} 1 = \frac{\pi}{4},$$

$$\arccos x = \begin{cases} \operatorname{arctg} \dfrac{\sqrt{1-x^2}}{x}, & 0 < x \leqslant 1, \\ \pi + \operatorname{arctg} \dfrac{\sqrt{1-x^2}}{x}, & -1 \leqslant x < 0, \end{cases}$$

$$\arccos x = 2 \operatorname{arctg} \sqrt{\frac{1-x}{1+x}}.$$

Единичная матрица третьего порядка:

$$\mathbf{I} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Определитель матрицы второго порядка:

$$\Delta_2 = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{21}a_{12}$$

Определитель матрицы третьего порядка:

$$\Delta_3 = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13} \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}.$$

13.2 Таблица ASCII-кодов некоторых символов

Разные значки–I

32 —	36 — $	40 — (	44 — ,
33 — !	37 — %	41 —)	45 — -
34 — ”	38 — &	42 — *	46 — .
35 — #	39 — ’	43 — +	47 — /

Цифры

48 — 0	50 — 2	52 — 4	54 — 6	56 — 8
49 — 1	51 — 3	53 — 5	55 — 7	57 — 9

Разные значки–II

58 — : 59 — ; 60 — < 61 — = 62 — > 63 — ? 64 — @

Прописные буквы латинского алфавита

65 — A	69 — E	73 — I	77 — M	81 — Q	85 — U	89 — Y
66 — B	70 — F	74 — J	78 — N	82 — R	86 — V	90 — Z
67 — C	71 — G	75 — K	79 — O	83 — S	87 — W	
68 — D	72 — H	76 — L	80 — P	84 — T	88 — X	

Разные значки–III

91 — [92 — \ 93 —] 94 — ^ 95 — _ 96 — ‘

Разные значки–IV

123 — { 124 — | 125 — } 126 — ~

Строчные буквы латинского алфавита

97 — a	101 — e	105 — i	109 — m	113 — q	117 — u	121 — y
98 — b	102 — f	106 — j	110 — n	114 — r	118 — v	122 — z
99 — c	103 — g	107 — k	111 — o	115 — s	119 — w	
100 — d	104 — h	108 — l	112 — p	116 — t	120 — x	

Прописные буквы русского алфавита

128 — А	132 — Д	136 — И	140 — М	144 — Р	148 — Ф	152 — Ш	156 — Ь
129 — Б	133 — Е	137 — Й	141 — Н	145 — С	149 — Х	153 — Щ	157 — Э
130 — В	134 — Ж	138 — К	142 — О	146 — Т	150 — Ц	154 — Ъ	158 — Ю
131 — Г	135 — З	139 — Л	143 — П	147 — У	151 — Ч	155 — Ы	159 — Я

Строчные буквы русского алфавита–I

160 — а	162 — в	164 — д	166 — ж	168 — и	170 — к	172 — м	174 — о
161 — б	163 — г	165 — е	167 — з	169 — й	171 — л	173 — н	175 — п

Строчные буквы русского алфавита–II

224 — р	226 — т	228 — ф	230 — ц	232 — ш	234 — ъ	236 — ь	238 — ю
225 — с	227 — у	229 — х	231 — ч	233 — щ	235 — ы	237 — э	239 — я

13.3 Системы счисления

Перевод чисел в десятичную систему счисления

Развернутая запись числа. В произвольной позиционной системе счисления любое число представляется суммой

$$N_p = \pm(a_{i-1}p^{i-1} + \ldots + a_1p^1 + a_0p^0 + a_{-1}p^{-1} + a_{-2}p^{-2} + \ldots + a_{-m}p^{-m}),$$

где p — **основание системы счисления** ($p > 1$), т. е. количество цифр, доступных для записи чисел; i — количество целых разрядов числа; a_k — цифры данной системы счисления.

Правило перевода. Число, представленное в любой системе счисления, необходимо записать в развернутой форме и вычислить его значение.

Пример 13.1. Подробный процесс перевода p-ичных чисел:

$142_5 = 1 \cdot 5^2 + 4 \cdot 5^1 + 2 \cdot 5^0 = 5^2 + 4 \cdot 5 + 2 = 25 + 20 + 2 = 47_{10}$;

$142_8 = 1 \cdot 8^2 + 4 \cdot 8^1 + 2 \cdot 8^0 = 8^2 + 4 \cdot 8 + 2 = 64 + 32 + 2 = 98_{10}$;

$12\mathrm{C}_{16} = 1 \cdot 16^2 + 2 \cdot 16^1 + \mathrm{C} \cdot 16^0 = 16^2 + 2 \cdot 16 + \mathrm{C} = 256 + 32 + 12 = 300_{10}$.

Для сокращения записи сумму, выделенную красным цветом, рекомендуется пропускать. Особенно это касается перевода чисел из двоичной системы счисления

$$101_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 2^2 + 1 = 5_{10}.$$

Пример 13.2. Примеры перевода нецелых чисел:

$$54{,}5_8 = 5 \cdot 8^1 + 4 \cdot 8^0 + 5 \cdot 8^{-1} = 5 \cdot 8 + 4 + \frac{5}{8} = 44{,}625_{10},$$

$$1001{,}01_2 = 2^3 + 2^0 + 2^{-2} = 8 + 1 + \frac{1}{4} = 9{,}25_{10}.$$

Перевод чисел из десятичной системы счисления

Правило перевода целой части числа. Целая часть числа делится на основание p новой системы счисления, остаток от деления запоминается. Полученное частное вновь делится на p, остаток запоминается. Процесс продолжается до тех пор, пока частное не станет меньше делителя. Остатки от деления на p выписываются в порядке, обратном их получению.

Пример 13.3. Переведем десятичное числа 35 в систему счисления с основанием p ($p = 2, 8, 16$). В таблицах, демонстрирующих процесс перевода, в левом столбце записаны частные, в правом — остатки от деления на p. Стрелки обозначают направление записи остатков.

35	1
17	1
8	0
4	0
2	0
1	1 ↑

$35_{10} = 100011_2$.

35	3
4	4 ↑

$35_{10} = 43_8$.

35	3
2	2 ↑

$35_{10} = 23_{16}$.

Компактный способ записи процесса перевода целого десятичного числа $X_{10} \to Y_p$ на примере $35 = X_2$:

$$35_1 \quad 17_1 \quad 8_0 \quad 4_0 \quad 2_0 \quad 1_1 = 100011_2.$$

Здесь индексы — остатки от деления указанного числа на $p = 2$, которые в ответе выписываются в обратном порядке (справа налево).

Пример 13.4. Перевод числа 72 в системы счисления с основаниями $p = 2$, 3, 4, 5:

$$\begin{aligned}
p = 2: &\quad 72_0 \quad 36_0 \quad 18_0 \quad 9_1 \quad 4_0 \quad 2_0 \quad 1_1 = 1001000_2,\\
p = 3: &\quad 72_0 \quad 24_0 \quad 8_2 \quad 2_2 = 2200_3,\\
p = 4: &\quad 72_0 \quad 18_2 \quad 4_0 \quad 1_1 = 1020_4,\\
p = 5: &\quad 72_2 \quad 14_4 \quad 2_2 = 242_5.
\end{aligned}$$

Правило перевода дробной части числа. Дробная часть числа последовательно умножается на основание новой системы счисления, после чего целая часть запоминается и отбрасывается. Процесс продолжается до тех пор, пока дробная часть не станет равной нулю. Целые части выписываются после запятой в порядке их получения.

Пример 13.5. Перевод числа $0{,}1875_{10}$ в восьмеричную систему счисления.

В левом столбце таблицы записаны целые части, в правом — дробные. Стрелка обозначает направление записи остатков.

0	1875
↓ 1	5
4	0

$0{,}1875_{10} = 0{,}14_8$.

Компактный способ записи процесса перевода дробного десятичного числа $0{,}X_{10} \to Y_p$ продемонстрируем на примере перевода числа $0{,}1875$ в 8-ричную систему счисления:

$$_0|1875 \quad _1|5 \quad _4|0 = 0{,}14$$

Здесь индексы — целые части, которые запоминаются и отбрасываются. В ответе целые части выписываются в порядке их получения.

Пример 13.6. Перевод числа $137{,}8755$ в 8-ричную систему счисления.

Целая и дробная части переводятся отдельно, а затем складываются. Результат перевода дробной части получим с четырьмя знаками после запятой.

$$137 = X_8: \quad 137_1 \quad 17_1 \quad 2_2 = 211_8.$$

$$0{,}8755 = Y_8: \quad _0|8755 \quad _7|004 \quad _0|032 \quad _0|256 \quad _2|048 = 0{,}7002.$$

Ответ: 211,7002.

Быстрый перевод чисел: $X_{10} \leftrightarrow Y_2$

Пример 13.7. Быстрый перевод продемонстрируем на примере перевода числа 53 в двоичную систему счисления.

Решение. Представляем заданное число в виде сумм степеней двойки, начиная с самой большой степени:

$$53 = 32 + 16 + 4 + 1 = 2^5 + 2^4 + 2^2 + 2^0 =$$

Наличие некоторой степени двойки в сумме означает, что при переводе числа в двоичную систему счисления на позициях, совпадающих с показателем степени двойки находится цифра 1, в остальных случаях 0:

$$= 2^5 + 2^4 + 2^2 + 2^0 = {}^{5\ 4\ 3\ 2\ 1\ 0}1\,1\,0\,1\,0\,1_2 .$$

Двоичные триады

Восьмеричная система счисления ($8 = 2^3$)

Каждая 8-ричная цифра может быть представлена тройкой двоичных цифр — **триадой**.

Цифра$_{10}$	Цифра$_8$	Триада$_2$
0	0	000
1	1	001
2	2	010
3	3	011
4	4	100
5	5	101
6	6	110
7	7	111

Двоичные тетрады

Шестнадцатиричная система счисления ($16 = 2^4$)

Каждая 16-ричная цифра может быть представлена четверкой двоичных цифр — **тетрадой**.

$Число_{10}$	$Цифра_{16}$	$Тетрада_2$
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Перевод чисел: $X_2 \to Y_{2^k}$ и $X_{2^k} \to Y_2$

Системы счисления, основания которых являются степенями одного числа, называют **родственными**.

Системы счисления, родственные двоичной: 4, 8, 16.

Запись $p = 2^k$ означает, что число p представляет собой степень числа 2 с показателем k:

- $p = 16 = 2^4$, здесь $k = 4$.

Запись $X_2 \to Y_{2^k}$ означает, что число переводится из двоичной системы счисления в родственную систему счисления.

Правило перевода числа из двоичной системы счисления в родственную систему счисления ($X_2 \to Y_{2^k}$)

1) Разбить цифры двоичного числа на группы по k цифр (где k — значение показателя степени двойки у основания новой системы счисления).

2) Каждую группу заменить соответствующей цифрой новой системы.

Для целого числа разбиение по группам производится **справа налево**, при необходимости дополняя нулями крайнюю **левую** группу.

Например, при переводе целого двоичного числа в 8-ричную систему ($k = 3$) число разбиваем на триады справа налево

$$1111010101_2 = \underset{\leftarrow\!-\!-}{001}\ \underset{\leftarrow\!-\!-}{111}\ \underset{\leftarrow\!-\!-}{010}\ \underset{\leftarrow\!-\!-}{101}$$

Для дробного числа разбиение на группы производится

- ← **справа налево** для цифр, стоящих слева от запятой, при необходимости дополняя нулями крайнюю **левую** группу;
- → **слева направо** для цифр, стоящих справа от запятой, при необходимости дополняя нулями крайнюю **правую** группу

При переводе в 8-ричную систему счисления ($k = 3$) число разбиваем на двоичные триады; при переводе в 16-ричную систему счисления ($k = 4$) — двоичные тетрады

$$1111010101{,}11_2 = \overleftarrow{001}\ \overleftarrow{111}\ \overleftarrow{010}\ \overleftarrow{101}\,,\,\overrightarrow{110}_2 = \overleftarrow{0011}\ \overleftarrow{1101}\ \overleftarrow{0101}\,,\,\overrightarrow{1100}_2.$$

Пример 13.8. Перевести в 8- и 16-ричную системы счисления число $110{,}001_2$.

Решение. Разбиваем число $110{,}001_2$ на триады и тетрады, которые затем заменяем соответствующими 8- и 16-ричными цифрами.

Двоичные триады	110	,	001
8-ричные цифры	6	,	1

Двоичные тетрады	0110	,	0010
16-ричные цифры	6	,	2

Ответ: $110{,}001_2 = 6{,}1_8 = 6{,}2_{16}$.

Правило перевода числа из системы счисления с основанием, равным степени двойки, в двоичную систему счисления ($X_{2^k} \to Y_2$)

каждую цифру числа преобразовать в группу двоичных цифр; количество двоичных цифр в группах равно показателю степени двойки в старой системе счисления.

Пример 13.9. Выполнить перевод числа $10{,}47_8$ в двоичную систему счисления.

Решение. Каждую цифру числа $10{,}47_8$ заменяем двоичной триадой:

8-ричные цифры	1	0	,	4	7
Двоичные триады	001	000	,	100	111

Ответ: $10{,}47_8 = 1000{,}100111_2$.

Перевод чисел: $X_p \to Y_s$

Правило перевода числа из произвольной системы счисления в произвольную систему счисления ($X_p \to Y_s$):

1) перевести число из исходной системы счисления в **десятичную**;
2) перевести десятичное число в требуемую систему счисления.

Кратко правило запишется в виде: $X_p \to Z_{10} \to Y_s$.

Правило перевода $X_{2^k} \to Y_{2^m}$:

1) перевести число из исходной системы счисления в **двоичную**;
2) перевести двоичное число в требуемую систему счисления.

Кратко правило запишется в виде: $X_{2^k} \to Z_2 \to Y_{2^m}$.

Арифметические операции

Арифметические действия в любой позиционной системе выполняются подобно аналогичным операциям с десятичными числами. Выполнение арифметических операций в системе счисления с основанием p удобно проводить по таблицам сложения и умножения.

+	0	1
0	0	1
1	1	10

×	0	1
0	0	0
1	0	1

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

Пример 13.10. Сложение «больших» чисел:

а) $100000100_{(2)} + 111000010_{(2)} = 1011000110_{(2)}$.

$$\begin{array}{r} 100000100 \\ +\ 111000010 \\ \hline 1011000110 \end{array}$$

б) $147{,}14_{(8)} + 350{,}34_{(8)} = 517{,}5_{(8)}$.

$$\begin{array}{r} 147{,}14 \\ +\ 350{,}34 \\ \hline 517{,}50 \end{array}$$

в) $12\mathrm{C}{,}3_{(16)} + 3\mathrm{B}3{,}5_{(16)} = 4\mathrm{DF}{,}8_{(16)}$.

$$\begin{array}{r} 12\mathrm{C}{,}3 \\ +\ 3\mathrm{B}3{,}5 \\ \hline 4\mathrm{DF}{,}8 \end{array}$$

Пример 13.11. Вычитание чисел:

а) $1001{,}01_{(2)} - 110{,}1_{(2)} = 10{,}11_{(2)}$.

$$\begin{array}{r} 1001{,}01 \\ -\ 110{,}1 \\ \hline 10{,}11 \end{array}$$

б) $411{,}2_{(8)} - 231{,}54_{(8)} = 157{,}44_{(8)}$.

$$\begin{array}{r} 411{,}2 \\ -\ 231{,}54 \\ \hline 157{,}44 \end{array}$$

в) $4\mathrm{D}2{,}8_{(16)} - 3\mathrm{B}3{,}5_{(16)} = 11\mathrm{F}{,}3_{(16)}$.

$$\begin{array}{r} 4\mathrm{D}2{,}8 \\ -\ 3\mathrm{B}3{,}5 \\ \hline 11\mathrm{F}{,}3 \end{array}$$

Пример 13.12. Умножение «больших» чисел:

а) $117_8 \cdot 53_8 = 6505_8$.

$$\begin{array}{r} 117 \\ \times\ 53 \\ \hline 355 \\ +\ 613 \\ \hline 6505 \end{array}$$

б) $117_{16} \cdot 53_{16} = 5\mathrm{A}75_{16}$.

$$\begin{array}{r} 117 \\ \times\ 53 \\ \hline 345 \\ +\ 573 \\ \hline 5\mathrm{A}75 \end{array}$$

Некоторые факты, которые следует запомнить

Умножение числа на величину основания приводит к перемещению запятой, отделяющей целую часть от дробной, на один разряд **вправо**:

$$777{,}77_{10} \cdot 10 = 7777{,}7_{10}; \qquad 101{,}01_2 \cdot 2 = 1010{,}1_2;$$

Деление числа на величину основания приводит к перемещению запятой, отделяющей целую часть от дробной, на один разряд **влево**:

$$777{,}77_{10} : 10 = 77{,}777_{10}; \qquad 101{,}01_2 : 2 = 10{,}101_2.$$

Представление целых чисел

В k-разрядной ячейке может храниться 2^k различных значений целых чисел.

Диапазон значений целых неотрицательных чисел: от 0 до $2^k - 1$.

Алгоритм получения внутреннего представления положительного числа N, хранящегося в k разрядном машинном слове:

1) перевести число N в двоичную систему счисления;
2) результат перевода дополнить слева незначащими нулями до k разрядов.

Например, внутреннее представление числа 37 в двухбайтовой ячейке:

$$0000\,0000\,0010\,0101_2.$$

Для представления отрицательных чисел используется дополнительный код, который позволяет заменить арифметическую операцию вычитания операцией сложения $A - B = A + (-B)$. Это существенно упрощает работу процессора и увеличивает его быстродействие.

Классический алгоритм получения внутреннего представления отрицательного числа N, хранящегося в k разрядном машинном слове:

1) получить внутреннее представление положительного числа N;
2) получить **обратный код** этого числа заменой 0 на 1 и 1 на 0 (инвертировать значения всех бит);
3) к полученному числу прибавить 1.

Данная форма называется **дополнительным кодом**.

Пример 13.13 (классика)**.** Получение внутреннего представления числа -17 в однобайтной ячейке.

Решение.

1) Получение внутреннего представления числа 17:

$$17 = 10001_2 = (\text{добавляем незначащие нули до 8 разрядов}) = 0001\,0001_2.$$

2) Получение обратного кода (инвертирование бит): $1110\,1110_2$
3) Получение дополнительного кода (добавление 1): $11101110_2 + 1 = 11101111_2$.

Внутреннее представление числа -17: 11101111.

Альтернативный алгоритм получения внутреннего представления отрицательного числа N, хранящегося в k разрядном машинном слове:

1) вычесть из положительного числа N единицу;
2) полученное число перевести в двоичную систему счисления;
3) результат дополнить слева незначащими нулями до k разрядов;
4) инвертировать значения всех бит.

Пример 13.14 (альтернатива)**.** Получение внутреннего представления числа -17 в однобайтной ячейке с использованием альтернативного алгоритма.

Решение.

1) $17 - 1 = 16$;
2) $16 = 2^4 = 10000_2$;
3) $0001\,0000_2$;
4) $1110\,1111_2$.

Внутреннее представление числа -17: 11101111.

Достоинство альтернативного способа заключается в том, что не надо в двоичной системе счисления проводить операцию добавления единицы.

Классический алгоритм перевода дополнительного кода в десятичное число:

1) инвертировать дополнительный код;
2) к полученному коду прибавить единицу (результат: модуль отрицательного числа);
3) перевести результат в десятичное число;
4) приписать знак отрицательного числа.

Пример 13.15 (классика)**.** По дополнительному коду 11101111_2 восстановить десятичное отрицательное число.

Решение.

1) Инвертирование дополнительного кода: $0001\,0000_2$.
2) Добавление единицы: $1\,0000_2 + 1 = 1\,0001_2$.
3) Перевод в десятичное число: $10001_2 = 17$.
4) Приписывание знака «минус»: -17.

Альтернативный алгоритм перевода дополнительного кода в десятичное число:

1) инвертировать дополнительный код;
2) перевести результат в десятичное число;
3) к полученному коду прибавить единицу (результат: модуль отрицательного числа);
4) приписать знак отрицательного числа.

Пример 13.16 (альтернатива)**.** По дополнительному коду 11101111_2 восстановить десятичное отрицательное число.

Решение.

1) Инвертирование дополнительного кода: $0001\,0000_2$.
2) Перевод в десятичное число: $10000_2 = 16$
3) Добавление единицы: $16 + 1 = 17$.
4) Приписывание знака «минус»: -17.

Список литературы

[1] Лутц М. Изучаем Python. 4-е изд. Пер.с англ. СПб.: Символ-Плюс, 2011.

[2] Саммерфелд М. Программирование на Python 3. Подробное руководство. Пер. с англ. СПб.: Символ-Плюс, 2009.

[3] Бизли Д. Python. Подробный справочник. Символ-Плюс, 2010.

[4] Васильев А. Н. Python на примерах. Практический курс по программированию. СПб.: Наука и Техника, 2016.

[5] Россум Г., Дрейк Ф. Л. Дж., Откидач Д. С. и др. Язык программирования Python. 2001. (версии от 1.5.2 до 2.0)

[6] Документация по языку Python. URL: https://www.python.org/doc/ (дата обращения: 1.09.2017).

[7] Python 3.6.2 documentation. URL: https://docs.python.org/3/ (дата обращения: 1.09.2016).

[8] Ссылки на русскоязычные ресурсы. URL: https://wiki.python.org/moin/RussianLanguage (дата обращения: 1.09.2017).

[9] Онлайн изучение языка Python для начинающих. URL: http://pythontutor.ru/ (дата обращения: 1.09.2017).

[10] Онлайн изучение языка Python. URL: http://learnpython.org/ (дата обращения: 1.09.2017).

[11] Редактор Pyzo. URL: http://www.pyzo.org/start.html

[12] Вирт Н. Алгоритмы + структуры данных = программы. М.: Мир, 1985.

[13] Ахо А. В., Хопкрофт Дж. Э., Ульман Д. Д. Структуры данных и алгоритмы. М.: Изд. Дом «Вильямс», 2000.

[14] Абрамов С. А., Зима Е. В. Начала информатики. М.: Наука, 1989.

[15] Амелина Н. И., Демяненко Я. М., Лебединская Е. Н. и др. Задачи по программированию. М.: Вуз. книга, 2000.

[16] Говорухин В. Н., Цибулин В. Г. Компьютер в математическом исследовании. Учебный курс. СПб.: Питер, 2001.

[17] Говорухин В. Н., Цибулин В. Г. Maple — система аналитических вычислений для математического моделирования. [Электронный ресурс]. URL: http://www.math.rsu.ru/mexmat/kvm/mme/courses/maple_c/ (дата обращения: 2.12.2015).

[18] Могилев А. В., Пак Н. И., Хеннер Е. К. Практикум по информатике. М.: Издательский центр «Академия», 2001.

[19] Ширяева Е. В., Романов М. Н., Долгих Т. Ф. Практикум по курсу «Основы информатики» (электронное учебное пособие). Ростов-на-Дону, ЮФУ. Компьютерная разработка фонда компьютерных изданий ЮФУ, регистр. № 844 от 7.12.2015. [Электронный ресурс]. URL: http://www.open-edu.sfedu.ru/node/2853 (дата обращения: 1.09.2016).

При создании электронного пособия использовались системы LaTeX 2ε, XeTeX и материалы книги:

Жуков М. Ю., Ширяева Е. В. LaTeX 2ε: искусство набора и вёрстки текстов с формулами. Ростов н/Д: Изд-во ЮФУ, 2009. 192 с.

Список иллюстраций

1 Блок-схемы (слева if-else в if; справа if в if-else) 30
2 Получение коэффициента C для (4.4) с помощью программы Maple . . 46
3 Значения $y_i = 1/i^2$ 47
4 Заданный четырехугольник 70
5 Окно Maple 11 (Classic Worksheet) 128
6 Страница справки 130
7 Непривычное представление выражений 132
8 Вывод векторов и матриц 134
9 Результат построения 141

Об авторах

Ширяева Елена Владимировна — кандидат физико-математических наук, доцент кафедры «Вычислительная математика и математическая физика» института математики, механики и компьютерных наук им. И. И. Воровича ФГАОУВО «Южный федеральный университет».

Область научных интересов — массоперенос в многокомпонентных смесях, конечно-разностные методы, метод конечных элементов, течения вязкой жидкости, вычислительная математика, программирование.

Романов Максим Николаевич — кандидат физико-математических наук, старший преподаватель кафедры «Вычислительная математика и математическая физика» института математики, механики и компьютерных наук им. И. И. Воровича ФГАОУВО «Южный федеральный университет».

Область научных интересов — изотермические и неизотермические течения жидкости между вращающимися цилиндрами с проницаемыми и непроницаемыми стенками (проблема Куэтта–Тейлора), численный анализ нелинейных задач теории гидродинамической устойчивости, изучение пересечения бифуркаций и возникновения хаотических режимов, программирование, проектирование баз данных.

Мелехов Андрей Петрович — кандидат физико-математических наук, старший преподаватель кафедры «Вычислительная математика и математическая физика» института математики, механики и компьютерных наук им. И. И. Воровича ФГАОУВО «Южный федеральный университет».

Долгих Татьяна Федоровна — ассистент кафедры «Вычислительная математика и математическая физика» института математики, механики и компьютерных наук им. И. И. Воровича ФГАОУВО «Южный федеральный университет».

Область научных интересов — течения вязкой жидкости, массоперенос в многокомпонентных смесях, метод конечных элементов, вычислительная математика, программирование.

Полякова Наталья Михайловна — ассистент кафедры «Вычислительная математика и математическая физика» института математики, механики и компьютерных наук им. И. И. Воровича ФГАОУВО «Южный федеральный университет».

Интерфейс пользователя

Навигация по электронному документу

Навигация по электронному документу может осуществляться с помощью клавиш и колесика мыши, клавиатуры — используются стандартные горячие клавиши программы Adobe Acrobat; элементов управления программы Adobe Acrobat (панель навигации и список закладок в окне Bookmarks), а также панели навигации учебного пособия  (см. правый верхний угол страницы):

Пиктограмма	Действие
◄	предыдущая страница
►	следующая страница
↶	предыдущий просмотр
↷	следующий просмотр
■	переход на страницу N
🔍	поиск слов в документе

Список горячих клавиш для работы с электронным документом в программе Adobe Acrobat

Действие	Комбинация клавиш
Полноэкранный режим	Ctrl + L
Выход из полноэкранного режима	Esc
Растянуть по ширине экрана	Ctrl + 2
Переход к началу документа	Home или Shift + Ctrl + Page Up или Shift + Ctrl + Up Arrow
Переход к концу документа	End или Shift+Ctrl+Page Down или Shift+Ctrl+Down Arrow
Переход на страницу N	Shift + Ctrl + N
Предыдущий экран	Page Up или Return
Следующий экран	Page Down или Shift + Return
Предыдущая страница	Left Arrow или Ctrl + Page Up
Следующая страница	Right Arrow или Ctrl + Page Down
Предыдущий просмотр	Alt + Left Arrow
Следующий просмотр	Alt + Right Arrow