

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Братский государственный университет»

*А.Н. Ефремова*

# **Программирование**

(I часть)

*Методические указания  
к выполнению лабораторных работ*

Братск  
Издательство Братского государственного университета  
2020

Ефремова А.Н. Программирование (1 часть): методические указания. – Братск: Изд-во БрГУ, 2020. – 104 с.

В методических указаниях даются рекомендации к выполнению лабораторных работ по дисциплине «Программирование» для обучающихся направления 09.03.02 «Информационные системы и технологии» всех форм обучения.

Рецензент В.А. Мельникова, канд. техн. наук, доцент кафедры ИиПМ (ФГБОУ ВО «БрГУ», г. Братск)

Отпечатано в авторской редакции  
в издательстве ФГБОУ ВО «БрГУ»  
665709, Братск, ул. Макаренко, 40  
Заказ 47

*Электронная версия издания размещена  
в локальной сети ФГБОУ ВО «БрГУ»  
в разделе «Библиотека»*

© Ефремова А.Н., 2020  
© ФГБОУ ВО «БрГУ», 2020

## Введение

Методические указания предназначены для использования обучающимися в процессе выполнения лабораторных работ по дисциплине «Программирование».

Процесс изучения дисциплины направлен на формирование компетенций:

ОПК-6: способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий.

В результате освоения дисциплины обучающийся должен:

**знать** синтаксис выбранного языка программирования, особенности программирования на этом языке, стандартные библиотеки языка программирования;

**уметь** составлять алгоритмы, использовать базовые алгоритмические структуры для решения типовых задач; применять выбранные языки программирования для написания программного кода; писать и отлаживать коды на языке программирования, тестировать работоспособность программы;

**владеть** навыками разработки алгоритмов и программ для решения практических задач в области информационных систем и технологий; навыками выбора, применения методов и алгоритмов и технологии программирования для решения стандартных профессиональных задач; языком программирования высокого уровня, методами отладки и тестирования работоспособности программы.

## Лабораторная работа № 1

### ЗНАКОМСТВО С ИНТЕГРИРОВАННОЙ СРЕДОЙ РАЗРАБОТКИ IDLE PYTHON. РАБОТА С БИБЛИОТЕКОЙ MATH

*Цель:* изучить интерфейс интегрированной среды разработки IDLE Python; возможности используемых функции и констант библиотеки Math.

#### Краткие теоретические сведения

##### 1. Режимы работы Python

Интегрированная среда разработки **IDLE** (Integrated Development Environment) Python поддерживает два режима работы:

- *интерактивный режим.* В этом режиме команды выполняются сразу после их вызова. Результат выполнения команд сразу выводится на экран (рис. 1). Этот режим работы хорошо подходит для написания простых программ и тестирования работы фрагментов кода;

- *программный режим.* В этом режиме сначала записывается вся программа, а потом эта программа выполняется полностью. Предварительно текст программы нужно сохранить. Программа на языке Python носит имя «сценарий» или «скрипт» (рис. 2).

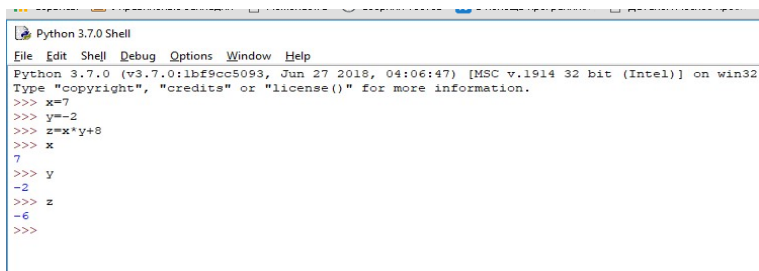


Рис. 1. Окно Python в интерактивном режиме

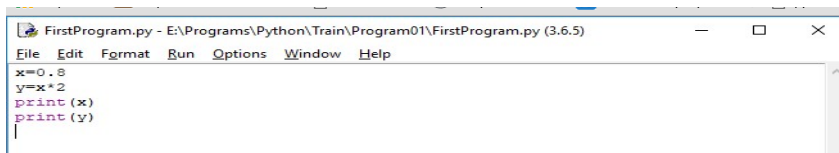


Рис. 2. Окно Python в программном режиме

## ***2. Особенности интерактивного режима***

В интерактивном режиме интерпретатор выполняет инструкции и сразу выводит результат. Сохранить эти инструкции в файле нельзя.

Интерактивный режим работы интегрированной среды Python используется для получения мгновенных расчетов, проведения экспериментов, тестирования программ.

В интерактивном режиме удобно выполнять эксперименты над небольшими фрагментами кода, которые затем могут быть перенесены в скрипты, которые реализуются в программном режиме.

## ***3. Ввод в интерактивном режиме многострочных инструкций***

Инструкции, которые имеют две и более строки в интерактивном режиме должны завершаться дополнительной пустой строкой. Это означает, чтобы завершить многостроковую инструкцию нужно два раза нажать клавишу Enter.

### **Пример.**

```
>>> x=3
>>> y=8
>>> if x>0: # многострочная инструкция
    y=5 # здесь нужно 2 раза нажать Enter

>>>
```

В приведенном выше коде выполняется многострочная инструкция **if**. Чтобы выйти из инструкции **if** нужно два раза нажать клавишу Enter.

## ***4. Особенности программного режима***

Программный режим позволяет сохранять программы длительное время. В программном режиме текст программы сначала записывается в файл, затем этот файл выполняется в интегрированной среде. Файлы на языке Python имеют расширение **\*.py** и называются **модулями**.

Модули – это простые текстовые файлы. Код модулей может выполняться сколько угодно. Интерпретатор Python выполняет весь программный код в модуле.

Файлы модулей, которые запускаются на выполнение непосредственно, еще называются сценариями.

### **5. Недостатки интерактивного режима**

Интерактивный режим имеет следующие взаимосвязанные недостатки:

- программный код, который вводится, нигде не сохраняется. Чтобы повторно запустить один и тот же программный код, его нужно снова ввести;
- программы, которые введены в интерактивном режиме, после выполнения интерпретатором Python исчезают. Чтобы выполнить их повторно, нужно их снова набрать или использовать операции копирования, что неэффективно.

### **6. Вызов и выполнение программы в программном режиме**

Если система находится в интерактивном режиме, то программный режим можно вызвать с помощью команды:

#### **File->New File**

как показано на рис. 3.

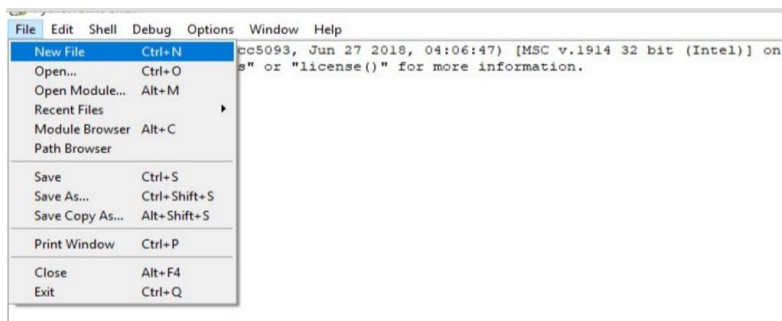


Рис. 3. Команда **File->New File**

В результате откроется новое окно редактора, в котором нужно ввести текст на языке Python. На рис. 4 изображено окно файла **myprog1.py** с текстом программы.

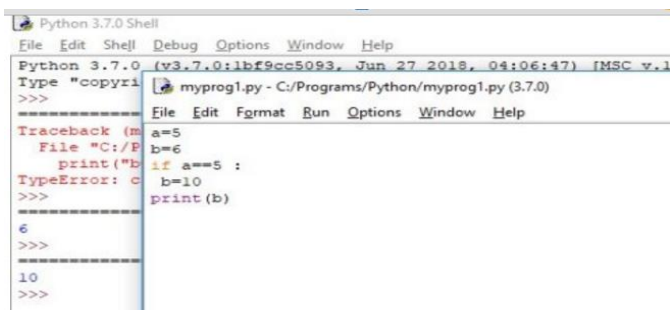


Рис. 4. Окно файла myprog1.py с текстом программы на языке Python

Чтобы запустить файл **myprog1.py** на выполнение нужно вызвать команду **Run** из меню **Run** или нажать клавишу **F5** (рис. 5).

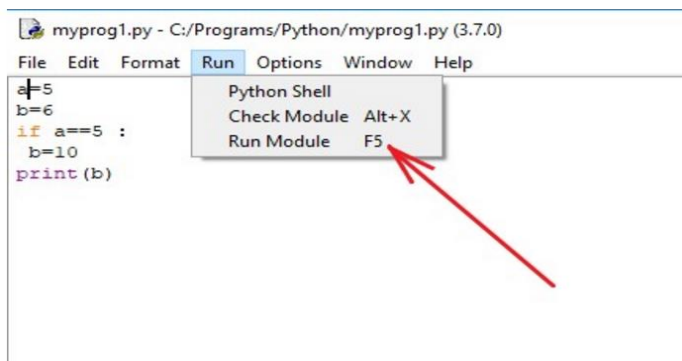


Рис. 5. Команда запуска программы на выполнение

В результате выполнения файла, произойдет переход в интерактивный режим со следующим результатом выполнения программы

```
===== RESTART: C:/Programs/Python/myprog1.py
=====
10
>>>
```

## 7. Библиотека Math

### Специальные константы библиотеки math

В библиотеке Math в Python есть две важные математические константы.

## Число Пи из библиотеки math

Первой важной математической константой является число **Пи** ( $\pi$ ). Оно обозначает отношение длины окружности к диаметру, его значение 3,141592653589793. Чтобы получить к нему доступ, сначала импортируем библиотеку **math**, затем получить доступ к константе, вызывая **pi** – **math.pi**:

```
1 # -*- coding: utf-8 -*-  
2 import math  
3 math.pi
```

Вывод

```
1 3.141592653589793
```

Данную константу можно использовать для вычисления площади или длины окружности ( $\pi r^2$ ). Далее представлен пример простого кода, с помощью которого это можно сделать:

```
1 # -*- coding: utf-8 -*- |  
2 import math  
3 radius=2  
4 print('Площадь окружности с радиусом 2 равна',math.pi*(radius**2))
```

Вывод

Площадь окружности с радиусом 2 равна 12.566370614359172

## Число Эйлера из библиотеки math

Число Эйлера ( $e$ ) является основанием натурального логарифма. Оно также является частью **библиотеки Math** в Python. Получить доступ к числу можно следующим образом:

```
2 import math  
3 math.e
```

Вывод

```
1 2.718281828459045
```

В следующем примере представлено, как можно использовать вышеуказанную константу:



```
1 import math
2
3 print((math.e + 6 / 2) * 4.32)
```

Вывод

```
1 24.702977498943074
```

## Экспонента и логарифм библиотеки math

В данном разделе рассмотрим функции библиотеки Math в Python, которые используются для нахождения экспоненты и логарифмов.

### Функция экспоненты exp() в Python

Библиотека Math в Python поставляется с функцией **exp()**, которую можно использовать для вычисления значения **e**.

К примеру,  $E^x$  – экспонента от  $x$ . Значение  $E$  равно 2,718281828459045.

Метод может быть использован со следующим синтаксисом:

```
1 math.exp(x)
```

Параметр **x** может быть положительным или отрицательным числом. Если **x** не число, метод возвращает ошибку. Рассмотрим пример использования данного метода:

```
1 import math
2
3 # Инициализация значений
4 an_int = 6
5 a_neg_int = -8
6 a_float = 2.00
7
8 # Передача значений методу exp() и вывод
9 print(math.exp(an_int))
10 print(math.exp(a_neg_int))
11 print(math.exp(a_float))
```

## Вывод

```
1 403.4287934927351
2 0.00033546262790251185
3 7.38905609893065
```

Можно также применить данный метод для встроенных констант, что продемонстрировано ниже:

```
1 # -*- coding: utf-8 -*-
2 import math
3 print(math.exp(math.e))
4 print(math.exp(math.pi))
```

## Вывод

```
1 15.154262241479262
2 23.140692632779267
```

При передаче не числового значения методу будет сгенерирована ошибка **TypeError**, как показано далее:

```
2 import math
3 print(math.exp("20"))
```

## Вывод

```
print(math.exp("20"))
```

```
TypeError: must be real number, not str
```

Как видно из примера выше, генерируется ошибка **TypeError**.

## Функция логарифма `log()` в Python

Функция `log()` возвращает логарифм определенного числа.

Натуральный логарифм вычисляется относительно основания **E**. В следующем примере показано использование функции логарифма:

```
1 import math
2
3 print("math.log(10.43):", math.log(10.43))
4 print("math.log(20):", math.log(20))
5 print("math.log(math.pi):", math.log(math.pi))
```

В скрипте выше методу передаются числовые значения с различными типами данных. Также рассчитывается натуральный логарифм константы **pi**. Вывод следующий:

```
1 math.log(10.43): 2.344686269012681
2 math.log(20): 2.995732273553991
3 math.log(math.pi): 1.1447298858494002
```

### Функция **log10()** в Python

Метод **log10()** возвращает логарифм по основанию **10** определенного числа. К примеру:

```
1 import math
2
3 # Возвращает log10 числа 50
4 print("log10 числа 50 равен:", math.log10(50))
```

Вывод

```
1 log10 числа 50 равен: 1.6989700043360187
```

### Функция **log2()** в Python

Функция **log2()** возвращает логарифм определенного числа по основанию **2**. К примеру:

```
1 import math
2
3 # Возвращает log2 числа 16
4 print("log2 числа 16 равен:", math.log2(16))
```

Вывод

```
1 log2 числа 16 равен: 4.0
```

### Функция **log(x, y)** в Python

Функция **log(x, y)** возвращает логарифм числа **x** по основанию **y**. К примеру:

```

1 import math
2
3 # Возвращает логарифм 3,4
4 print("Логарифм 3 по основанию 4 равен:", math.log(3, 4))

```

Вывод

```

1 Логарифм 3 по основанию 4 равен: 0.6309297535714574

```

## Функция $\log_1 p(x)$ в Python

Функция  $\log_1 p(x)$  рассчитывает логарифм(1+x), как представлено ниже:

```

1 import math
2
3 print("Значение логарифма(1+x) от 10 равно:", math.log1p(10))

```

Вывод

```

1 Значение логарифма(1+x) от 10 равно: 2.3978952727983707

```

## Арифметические функции в Python

Арифметические функции используются для представления чисел в различных формах и осуществления над ними математических операций. Далее представлен перечень самых популярных арифметических функций:

**ceil()**: округление определенного числа вверх;

**fabs()**: возвращает модуль (абсолютное значение) указанного числа;

**floor()**: округление определенного числа вниз;

**gcd(a, b)**: получение наибольшего общего делителя чисел a и b;

**fsum(iterable)**: возвращает сумму всех элементов итерируемого объекта;

**expm1()**: возвращает  $(e^x)-1$ ;

**exp(x)-1**: когда значение  $x$  слишком мало, вычисление  $\exp(x)$ -1 может привести к значительной потере в точности.  $\expm1(x)$  может вернуть вывод с полной точностью.

В следующем примере показано использование перечисленных выше функций:

```
1 # -*- coding: utf-8 -*-|
2 import math
3 num=-4.28
4 a=14
5 b=8
6 num_list=[10,8.25,75,7.04,-86.23,-6.43,8.4]
7 x=1e-4 #Малое значение x
8 print('Число:',num)
9 print('Округление числа вниз',math.floor(num))
10 print('Округление числа вверх',math.ceil(num))
11 print('Модуль числа',math.fabs(num))
12 print('Наибольший общий делитель a и b',str(math.gcd(a,b)))
13 print('Сумма элементов списка',str(math.fsum(num_list)))
14 print('Е^x (при использовании функции exp()) равно',math.exp(x)-1)
15 print('Е^x (при использовании функции expm1()) равно',math.expm1(x))
```

## Вывод

```
Число: -4.28
Округление числа вниз -5
Округление числа вверх -4
Модуль числа 4.28
Наибольший общий делитель a и b 2
Сумма элементов списка 16.029999999999998
Е^x (при использовании функции exp()) равно 0.0001000050001667141
Е^x (при использовании функции expm1()) равно 0.00010000500016667084
```

К числу других математических функций относятся:

— **pow()**: принимает два вещественных аргумента, возводит первый аргумент в степень, значением которой является второй аргумент, после чего возвращает результат. К примеру, **pow(2, 2)** эквивалентно выражению **2 \*\* 2**;

– **sqrt()**: возвращает **квадратный корень** определенного числа.

Примеры данных методов представлены ниже:

Возведение в степень

```
1 math.pow(3, 4)
```

Вывод

```
1 81.0
```

Квадратный корень

```
1 math.sqrt(81)
```

Вывод

```
1 9.0
```

## Тригонометрические функции в Python

Модуль **math** в Python поддерживает все тригонометрические функции. Самые популярные представлены ниже:

**sin(a)**: Возвращает синус "a" в радианах;

**cos(a)**: Возвращает косинус "a" в радианах;

**tan(a)**: Возвращает тангенс "a" в радианах;

**asin(a)**: Возвращает инвертированный синус. Аналогичным образом работают "atan" и "acos";

**degrees(a)**: Конвертирует угол "a" из радиан в градусы;

**radians(a)**: Конвертирует угол "a" из градусов в радианы.

Рассмотрим следующий пример:

```

1 import math
2
3 angle_In_Degrees = 62
4 angle_In_Radians = math.radians(angle_In_Degrees)
5
6 print('Значение угла:', angle_In_Radians)
7 print('sin(x) равен:', math.sin(angle_In_Radians))
8 print('tan(x) равен:', math.tan(angle_In_Radians))
9 print('cos(x) равен:', math.cos(angle_In_Radians))

```

## Вывод

```

1 Значение угла: 1.0821041362364843
2 sin(x) равен: 0.8829475928589269
3 tan(x) равен: 1.8807264653463318
4 cos(x) равен: 0.46947156278589086

```

Обратите внимание, что вначале мы конвертировали значение угла из градусов в радианы для осуществления дальнейших операций.

## Конвертация типов числа в Python

Python может конвертировать начальный тип числа в другой указанный тип. Данный процесс называется «преобразованием». Python может внутренне конвертировать число одного типа в другой, когда в выражении присутствуют смешанные значения. Такой случай продемонстрирован в следующем примере:

```

1 3 + 5.1

```

## Вывод

```

1 8.1

```

В вышеприведенном примере целое число **3** было преобразовано в вещественное число **3.0** с плавающей точкой. Результатом сложения также является число с плавающей точкой (или запятой).

Однако иногда вам необходимо явно привести число из одного типа в другой, чтобы удовлетворить требования параметра функции или оператора. Это можно сделать с помощью различных встроенных функций Python.

Например, чтобы преобразовать целое число в число с плавающей точкой, мы должны вызвать функцию **float()**, как показано ниже:

```
1 a = 12
2 b = float(a)
3 print(b)
```

Вывод

```
1 12.0
```

Целое число типа **integer** было преобразовано в вещественное число типа **float**. **float** также можно конвертировать в **integer** следующим образом:

```
1 a = 12.65
2 b = int(a)
3 print(b)
```

Вывод

```
1 12
```

Вещественное число было преобразовано в целое через удаление дробной части и сохранение базового числа. Обратите внимание, что при конвертации значения в **int** подобным образом число будет усекаться, а не округляться вверх.

### Ход выполнения

1. Изучить интерфейс интегрированной среды разработки IDLE Python.
2. Изучить возможности библиотеки Math.



## Лабораторная работа № 2

### ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМОВ ЛИНЕЙНОЙ СТРУКТУРЫ

**Цель работы:** ознакомиться с понятием линейной алгоритмической структуры; приемами программной реализации на языке программирования Python; произвести отладку и тестирование полученных программ.

#### **Задание:**

1. Изучить краткие теоретические сведения.
2. Разработать алгоритмы для решения задач (блок-схемы).
3. Записать алгоритмы на языке программирования Python.
4. Произвести отладку и тестирование программы.
5. Оформить отчет по лабораторной работе.

**Отчет о лабораторной работе** должен содержать следующие сведения:

1. Название и цель работы.
2. Формулировку задачи, схему алгоритма, программный код и результаты решения задачи.
3. Вывод по работе в целом.

#### **Краткие теоретические сведения**

##### ***Алфавит языка Python***

Основой языка программирования Python, как и любого другого языка, является алфавит – набор допустимых символов, которые можно использовать для записи программы. Это:

1. латинские прописные и строчные буквы (A, B, C, ..., X, Y, Z, a, b, c, ..., x, y, z);
2. русские прописные и строчные буквы (можно использовать, но это является очень плохим стилем);
3. арабские цифры (0, 1, 2, ..., 7, 8, 9);
4. специальные символы (знак подчёркивания; знаки препинания; круглые, квадратные скобки; знаки арифметических операций, # – знак комментариев и др.).

5. В языке существует также некоторое количество различных цепочек символов, рассматриваемых как единые смысловые эле-

менты с фиксированным значением. Такие цепочки символов называются служебными (зарезервированными ключевыми) словами.

### ***Идентификаторы и общие правила их написания***

Для того чтобы программа решения задачи обладала свойством массовости, следует употреблять не конкретные значения величин, а использовать их обозначения для возможности изменения по ходу выполнения программы их значений. Для обозначения в программе **переменных** и **постоянных** величин используются имена – идентификаторы (*identification* – установление соответствия объекта некоторому набору символов).

Программа на Python представляет собой последовательность **инструкций**, которые называются **операторами**. Необходимо учитывать, следующее:

- в идентификатор не могут входить пробелы, специальные символы алфавита;
- идентификатор начинается только с буквы или со знака подчеркивания;
- идентификатор может состоять из букв, цифр и знака подчеркивания;
- при написании идентификаторов можно использовать как прописные, так и строчные буквы латинского алфавита;
- идентификатор не должен являться зарезервированным словом.

**Прописные и строчные буквы в именах различаются, например, *f* и *F* – две разные переменные.**

### ***Оператор присваивания***

Действия, выполняемые компьютером в процессе решения задачи, записываются в виде операторов алгоритмического языка. Изменение значения переменной осуществляется **оператором присваивания**.

Присваивание в Python означает связывание значения с некоторым именем переменной.

Действие, выполняемое этим оператором, обозначается знаком =.

Слева от этого знака записывается имя той переменной, которой нужно присвоить новое значение (например, *x*=). Справа может быть:

1. Число, например  $x=5$ .

При выполнении оператора  $x=5$  в оперативной памяти по некоторому адресу будет создан объект, получивший значение 5. Затем будет создана переменная  $x$ , которой будет присвоен адрес этого объекта. Такой механизм занесения значений в ячейки памяти отличает Python от других языков программирования.

2. Выражение, например, арифметическое  $x=(3*a)/2$  или логическое  $x=a>1$ .

3. Другая переменная, например  $x=a$ . Операторы выполняются в той последовательности, в которой они записаны в программе.

Возможна запись оператора присваивания в следующем виде:

$*$ = оператор – умножение с присваиванием, например,  $x*=5$  идентичен оператору  $x=x*5$ .

$/$ = оператор – деление с присваиванием, например,  $x/=5$  идентичен оператору  $x=x/5$ .

$\%$ = оператор – остаток от деления с присваиванием, например,  $x\%=5$  идентичен оператору  $x=x\%5$ .

$+$ = оператор – сложение с присваиванием, например,  $x+=5$  идентичен оператору  $x=x+5$ .

$-$ = оператор – вычитание с присваиванием, например,  $x-=5$  идентичен оператору  $x=x-5$ .

### ***Типы данных***

Типы данных (табл.1) относятся к самым фундаментальным понятиям любого языка программирования. Для определения (объявления) переменных, интерпретатору или компилятору нужна следующая информация:

- имя переменной – по имени осуществляется связь переменной в программе с оперативной памятью компьютера;
- тип переменной – позволяет компилятору определить, какого вида информация хранится в переменной;
- значение переменной – определяет содержание информации, которая помещается в переменную.

В программах, написанных на Python, нет как такого раздела описания переменных, как, например, в Pascal.

В Python используется так называемая динамическая типизация, когда типы данных выясняются во время выполнения программы. Переменная сохраняет ссылку на объект определенного типа, а не сам объект. В момент переприсваивания значения друго-

го типа переменная будет ссылаться на другой объект, при этом изменится и тип переменной.

Таблица 1

| Название      | Обозначение    | Допустимые значения                         | Пример            |
|---------------|----------------|---|-------------------|
| Целочисленный | int (integer)  | Любое, размер ограничен оперативной памятью | 325               |
| Вещественный  | float          | Любые числа с дробной частью                | 9.23 0.0 - 1.7e-6 |
| Строковый     | str (string)   | Любые символы из таблицы Unicode            | "x=" "Строка"     |
| Логический    | bool (boolean) | False и True                                |                   |

В вещественном числе целая часть от дробной отделяется точкой, при этом перед точкой и после неё должно быть, по крайней мере, по одной цифре. Пробелы внутри числа недопустимы. Тип переменной определяется в тот момент, когда ей присваивается новое значение.

Выяснить, на какой тип данных ссылается переменная, поможет функция **type()**, имеющая синтаксис:

**type(имя переменной)** или **type(значение)**

Например,

```
>>> b = 3.85
>>> tb = type(b)
>>> tb
<class 'float'>
```

Для преобразования чисел из вещественных в целые и наоборот в Python определены функции **int()** и **float()**. Например, **int(12.6)** даст в результате 12, а **float(12)** даёт в результате 12.0 (десятичный разделитель – точка). Основные операции с числами приведены в табл. 2.

Таблица 2

| Операция | Описание  |
|----------|---|
| $x + y$  | Сложение (сумма $x$ и $y$ )   |
| $x - y$  | Вычитание (разность $x$ и $y$ )   |
| $x * y$  | Умножение (произведение $x$ и $y$ )   |
| $x/y$    | Деление $x$ на $y$ (частное). <b>Внимание!</b> Если $x$ и $y$ целые, то результат всегда будет целым числом! Для получения вещественного результата хотя бы одно из чисел должно быть вещественным.<br>Пример:<br>$100/8 \rightarrow 12$ ,<br>а вот<br>$100/8.0 \rightarrow 12.5$               |
| $x//y$   | Целочисленное деление (результат — целое число). Если оба числа в операции вещественные, получается вещественное число с дробной частью, равной нулю.<br>Пример:<br>$100//8 \rightarrow 12$<br>$101.8//12.5 \rightarrow 8.0$<br>(для сравнения<br>$101.8/12.5 \rightarrow 8.1440000000000001$ ) |
| $x\%y$   | Остаток от целочисленного деления $x$ на $y$<br>Пример:<br>$10\%4 \rightarrow 2$  |
| $x**y$   | Возведение в степень ( $x$ в степени $y$ ). Работает и для вещественных чисел.<br>Примеры:<br>$2**3 \rightarrow 8$<br>$2.3**(-3.5) \rightarrow 0.05419417057580235$   |
| $-x$     | Смена знака числа   |

Кроме того, в Python для операций с числами используются функции **abs()** (вычисление абсолютного значения — модуля, **abs(-3) → 3**), **pow()** (возведение в степень, **pow(2,3) → 8**), **divmod()** (вычисление результата целочисленного деления и остатка, **divmod(17,5) → (3,2)**) и **round()** (округление, **round(100.0/6) → 17.0**). Эти функции являются "встроенными", что означает, что для их использования нет необходимости подключать дополнительные модули. Все прочие функции для работы с числами (математиче-

ские), такие как вычисление квадратного корня, синуса и пр. требуют подключения модуля **Math** (см. лабораторную работу № 1).

### ***Кортеж***

**Кортеж** в Python – это упорядоченный набор объектов, в который могут одновременно входить объекты разных типов (числа, строки и другие структуры, в том числе и кортежи). В дальнейшем эти объекты будем называть элементами кортежа.

Кортеж задаётся перечислением его элементов в круглых скобках через запятую, например

```
t=(12, 'b', 34.6, 'derevo ')
```

С использованием допустимой в Python цепочки присваиваний можно элементам кортежа сразу сопоставить какие-нибудь переменные:

```
t=(x, s1, y, s2)=(12, 'b', 34.6, 'derevo ')
```

В этом случае элемент кортежа и соответствующая переменная будут указывать на одни и те же значения, т.е. значение `t[0]` будет равно значению `x`, а `t[3]`, соответственно, `s2`.

Однако эти переменные могут изменяться независимо от элементов кортежа. Присвоение нового значения переменной `s1` никак не влияет на элемент `t[1]`. А вот для элементов кортежа значения изменить уже нельзя, поскольку для Python кортеж относится к неизменяемым последовательностям.

Кортеж может быть пустым (для его определения нужно написать `t=()`), а может содержать только один элемент (например, `t=('domik',)`). Для кортежа из одного элемента **обязательно добавлять запятую** после имени или значения этого элемента.

Кортежи могут получаться в результате работы функций Python, например, уже упоминавшаяся функция **divmod()** возвращает кортеж из двух элементов.

Кортежи могут использоваться для хранения характеристик каких-нибудь предметов, существ или явлений, если эти предметы, существа или явления характеризуются фиксированным набором свойств. Например, в виде кортежа можно записать фамилию ученика и его оценки за полугодие.

Поскольку кортежи являются неизменяемыми последовательностями, операции с кортежами не меняют исходные кортежи (табл. 3).

Таблица 3

| Функция<br>или<br>операция                     | Описание и результат  |
|--|---|
| <code>len(t)</code>                            | Определяется количество элементов кортежа (результатом является число) <b>t</b>   |
| <code>t1 + t2</code>                           | Объединение кортежей. Получается новый кортеж, в котором после элементов кортежа <b>t1</b> находятся элементы кортежа <b>t2</b> . Пример:<br><code>t1=(1,2,3)</code><br><code>t2=('raz', 'dva')</code><br><code>t3=t1+t2</code><br><code>t3 → (1, 2, 3, 'raz', 'dva')</code>  |
| <code>t * n</code> или<br><code>(n * t)</code> | <b>n</b> -кратное повторение кортежа <b>t</b> Пример:<br><code>t2=('raz', 'dva')</code><br><code>t2*3 → ('raz','dva', 'raz', 'dva', 'raz', 'dva')</code>  |
| <code>t[i]</code>                              | Выбор из <b>t</b> элемента с номером <b>i</b> , нумерация начинается с 0 (первый элемент имеет номер 0) Если <b>i</b> <0, отсчёт идёт с конца (первый элемент кортежа имеет номер 0, последний имеет номер -1). Пример:<br><code>t3=(1, 2, 3, 'raz', 'dva')</code><br><code>t3[2] → 3</code><br><code>t3[-2] → 'raz'</code>   |
| <code>t[i : j : k]</code>                      | Срез – кортеж, содержащий элементы кортежа <b>t</b> с номерами от <b>i</b> до <b>j</b> с шагом <b>k</b> (элемент с номером <b>i</b> входит в итоговый кортеж, а элемент с номером <b>j</b> уже не входит). Если <b>k</b> не указан (использован вариант <code>t[i : j]</code> ), то элементы идут подряд (равносильно <code>t[i : j : 1]</code> ). Пример:<br><code>t3=(1, 2, 3, 'raz', 'dva')</code><br><code>t3[1:4] → (2, 3, 'raz')</code> |
| <code>min(t)</code>                            | Определяется элемент с наименьшим значением в соответствии с алфавитным («словарным») порядком. Пример:<br><code>t3=(1, 2, 3, 'raz', 'dva')</code><br><code>min(t3) → 1</code>  |
| <code>max(t)</code>                            | Определяется элемент с наибольшим значением в соответствии с алфавитным («словарным») порядком. Пример:<br><code>t3=(1, 2, 3, 'raz', 'dva')</code><br><code>max(t3) → 'raz'</code>  |

Важно понимать, что «словарный» порядок – сначала числа по возрастанию, затем строки, начинающиеся на цифры в порядке их возрастания, затем строки, начинающиеся на прописные буквы в алфавитном порядке, а затем строки, начинающиеся на строчные буквы также в алфавитном порядке – всегда используется в вычислительной технике при сортировке имён объектов. Строку можно преобразовать в кортеж с помощью функции **tuple()**, например:

```
s='amamam '  
t=tuple(s)  
t → ('a', 'm', 'a', 'm', 'a', 'm')
```

При работе с кортежами заменить значение элемента кортежа нельзя. Если при написании программы возникает такая необходимость, это может свидетельствовать о том, что кортеж является неподходящей структурой данных для решения задачи.

### ***Структура программы***

Программа на Python представляет из себя последовательность команд для ввода данных, вычислений и других операций с данными и вывода результатов. Простые команды (операторы) принято записывать по одной строке на оператор. В составных операторах (с которыми ещё предстоит познакомиться) большую роль играют пробелы в начале строки (отступы). Программа создаётся в виде текстового файла в любом текстовом редакторе. Использование интегрированных сред разработки (IDE) обеспечивает подсветку синтаксиса и выделение особенностей структуры программы, а также упрощает поиск ошибок в написании команд. Файл с программой должен иметь «расширение» **.py** (например, **my\_program.py**). Первую строку программы рекомендуется оформить как комментарий, в котором указывается кодировочная таблица («кодировка») данных файла, в противном случае Python не сможет правильно интерпретировать любые символы, встретившиеся в тексте программы и выходящие за пределы основной латиницы (в том числе кириллицу). Указанная в этом комментарии кодировка должна соответствовать действительной кодировке файла.

Основные особенности написания программ на Python:

1. Регистр учитывается.
2. Типы не объявляются и могут меняться в ходе программы (динамические).
3. Важны отступы для вложенных операторов (обычно **4 пробела** или **TAB**).



4. Один оператор – одна строка (если не помещается, для переноса «\» или взять в скобки).

5. В конце строки нет знака «;», можно использовать при написании нескольких инструкций в одной строке (через «;»).

### *Ввод данных: функция input()*

**a=input()** # Ввод строки с клавиатуры и запись в переменную a  
**a=int(a)** # преобразование строки в целое число

Можно объединить считывание строк и преобразование типов, если вызывать функцию **int** для того значения, которое вернет функция **input**:

**a = int(input())**

Ввод трех чисел через пробел

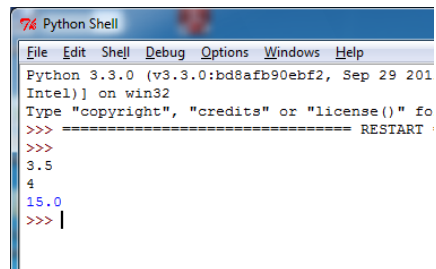
**a, b, c = map(int, input().split())**

Если число **a** вещественного типа, то вместо функции **int**, нужно использовать **float**, например:

**a = float(input())**

Например,

```
a = float(input())  
b = float(input())  
c = (a + b)*2  
print(c)
```



```
Python Shell  
File Edit Shell Debug Options Windows Help  
Python 3.3.0 (v3.3.0:bd8afb90ebf2, Sep 29 2011, Intel) on win32  
Type "copyright", "credits" or "license()" for more  
>>> ===== RESTART  
>>>  
3.5  
4  
15.0  
>>> |
```

### *Вывод данных: функция print()*

Функция **print** может выводить не только значения переменных, но и значения любых выражений.

Например, допустима запись **print(2 \*\* 3 + 2)**.

Также при помощи функции **print** можно выводить значение не одного, а нескольких выражений, для этого нужно перечислить их через запятую:

```
a = 1  
b = 2  
print(a, '+', b, '=', a + b)
```

В данном случае будет напечатан текст **1 + 2 = 3**

Сначала выводится значение переменной *a*, затем строка из знака “+”, затем значение переменной *b*, затем строка из знака “=”, наконец, значение суммы *a + b*.

### ***Особенности функции print()***

Выводимые данные разделяются одним пробелом;

Если понадобится изменить это правило, то применяют специальный параметр **sep** (*separator – разделитель*);

Например:

```
a,b,c = 2,3,1  
print(a, b, c, sep='-')
```

Результат:

2-3-1

Для того, чтобы совсем убрать разделитель при выводе нужно передать параметр **sep**, равный пустой строке:

```
print(a, '+', b, '=', a + b, sep = '')
```

Для того, чтобы значения выводились в новой строке, нужно в качестве параметра **sep** передать строку, состоящую из специального символа новой строки, которая задается так:

```
print(a,b,sep='\n')
```

Символ '\n' в текстовых строках является указанием на обозначение специального символа, в зависимости от того, какой символ записан после него. Наиболее часто употребляется символ новой строки '\n'.

Для того, чтобы вставить в строку сам символ, нужно повторить его два раза: '\\n'.

### ***Форматирование вещественных чисел при выводе***

```
x=2.71828  
print("%.2f"%x)  
x,y=2.71828,3.1415  
print("%.2f"%x,"%.2f"%y)
```

*Результат*

2.72

2.72 3.14

%4 – всего символов

2f – в дробной части

### *Линейные алгоритмы*

**Линейный алгоритм** – алгоритм, в котором вычисления выполняются строго последовательно. Типичная блок-схема линейного алгоритма показана на рис. 6.



Рис. 6. Типичная схема линейного алгоритма

Далее рассмотрим задачи с линейной структурой алгоритма.

**Задача 1.** Дано два числа  $a$  и  $b$ . Сделать так, чтобы их значения поменялись местами.

**Постановка задачи:** Имеются две переменные с какими-то определёнными значениями. Пусть значение  $a$  равно  $x$ , а значение  $b$  равно  $y$ . Требуется, чтобы значение  $a$  стало равно  $y$ , а значение  $b$  стало равно  $x$ .

**Метод решения (общий):** Использовать дополнительную переменную  $c$ , в которую временно записать начальное значение переменной  $a$ , присвоить переменной  $a$  значение переменной  $b$ , а потом переменной  $b$  присвоить значение переменной  $c$ , блок схема алгоритма (рис. 7).

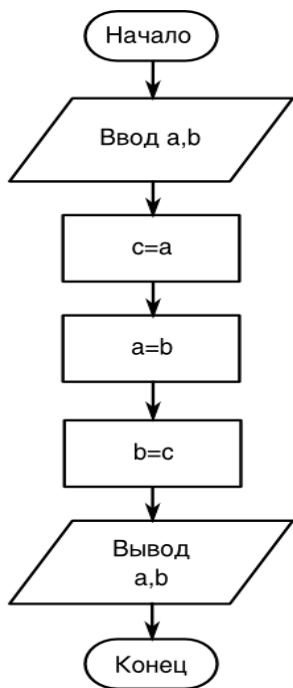


Рис. 7. Блок схема алгоритма обмена значениями

**Метод решения с использованием особенностей Python:** использовать два кортежа. В первом будут определены переменные  $a$  и  $b$  и их значения, а второй сформируем из этих же переменных, но в обратном порядке.

## Текст программы на Python

```
# -*- coding: utf-8 -*-  
# Перестановка местами двух чисел с использованием кортежа  
#  
z=(a,b)=eval1(input('Введите исходные значения (a, b) через за-  
пятую ='))  
(a,b)=(b,a)  
print ('Новое значение a: ', a, '\n', 'Новое значение b: ', b)
```

### Результат

Введите исходные значения (a, b) через запятую: =6,7

Новое значение a: 7

Новое значение b: 6

**Задача 2.** Вычислить площадь круга и длину окружности по заданному радиусу.

**Постановка задачи:** Исходными данными является значение радиуса. Требуется вычислить площадь круга и длину окружности, блок-схема алгоритма (рис. 8).

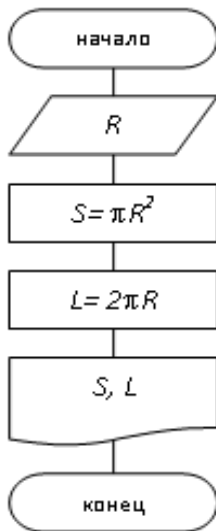


Рис. 8. Блок-схема алгоритма

---

<sup>1</sup> Функция **eval()** выполняет строку с кодом и возвращает результат выполнения.

### Текст программы на Python

```
# -*- coding: utf-8 -*-  
import math  
r=input('Введите исходные значения радиуса r=')  
r=float(r)  
S=math.pi*pow(r,2)  
L=2*math.pi*r  
print (' Площадь круга S=', "%4.2f"%S, '\n', 'Длина окружности  
L=', "%4.2f"%L)
```

### Результат

Введите исходные значения радиуса r=6

Площадь круга S= 113.10

Длина окружности L= 37.70

**Задача 3.** Вычислить значение функции

$$a = \frac{2 \cos^2 x^3}{\sqrt[5]{\operatorname{arctg} x^2}} + e^{2x^2}, \text{ при } x=2$$

**Постановка задачи:** Исходными данными является значение  $x=2$ . Требуется вычислить значение функции  $a$ , блок-схема алгоритма (рис. 9).

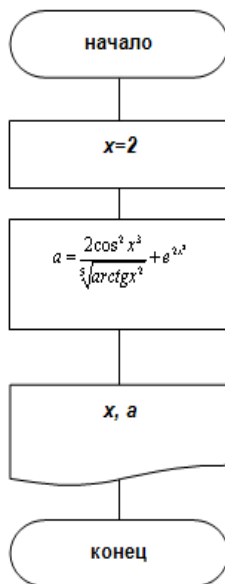


Рис. 9. Блок схема алгоритма

### Текст программы на Python

```
import math
x=2
a=2*pow(math.cos(pow(x,3)),2)/pow(math.atan(pow(x,2)),1/5)+m
ath.exp(2*pow(x,2))
print(' при x=',x, 'a=', "%6.2f"%a)
```

### Результат

при x= 2 a= 2981.00

### Задачи для самостоятельной работы

1. Вычислить значения переменных по заданным расчетным формулам. Вывести на экран значения исходных данных и результатов, сопровождая вывод именами переменных.

| № варианта | Расчетные формулы   | Исходные данные |       |        |
|------------|---|-----------------|-------|--------|
|            |   | x               | y     | z      |
| 1          | $a = y \sqrt[3]{ x } + \cos(y - 3)$ $b = \frac{1 + \cos(y - 2)}{x^3 + \sin^2 z}$                                    | 6,251           | 0,827 | 25,001 |
| 2          | $a = e^{ x - y } +  \sqrt{x} + y ^{x + y}$ $b = \frac{1 + \cos(y - 2)}{x^3 + \sin^2 z}$                             | 2,735           | 3,823 | 0,666  |
| 3          | $a = \frac{\sqrt{ x - 1 } - \sqrt[3]{y}}{1 + \frac{x^2}{2} + \frac{y^2}{4}}$ $b = x (\operatorname{Tg} z + e^{-x})$ | 0,622           | 3,325 | 5,541  |
| 4          | $a = \frac{y^{x+1} + x + \frac{y}{2}}{\sqrt[3]{x} + 3 - 2(x + y)}$ $b = (x - 1)^{\frac{1}{\sin z}}$                 | 3,741           | 15,4  | 0,252  |

| № варианта | Расчетные формулы   | Исходные данные |        |       |
|------------|---|-----------------|--------|-------|
|            |   | x               | y      | z     |
| 5          | $a = \ln \left  1 - \frac{2y}{1 + x^2 + y^3} \right $ $b = \cos^2 \operatorname{Tg} \left( \frac{1}{z} \right)$     | 1,625           | 0,825  | 0,16  |
| 6          | $a = 5 \operatorname{Tg} x - 2 \operatorname{Tg} y$ $b = \frac{x + 3(x - y) + x^2}{ x - z ^2 + \sqrt[3]{y}}$        | 2,444           | 9,869  | 0,166 |
| 7          | $a = \left  \frac{y}{x^x} - \sqrt[3]{y} \right $ $b = \frac{\operatorname{Tg}(y - z)}{1 + \frac{y - x}{(y - x)^2}}$ | 1,827           | 18,221 | 3,298 |
| 8          | $a = \frac{x + \frac{y}{5 + \sqrt{x}}}{ y + x  + \sqrt[3]{x}}$ $b = e^{z-1} + \operatorname{Tg}^2 x$                | 47,8            | 5,5    | 2,3   |
| 9          | $a = \frac{2 \cos(x - \frac{\pi}{6})}{\frac{1}{z} + \sin^2 y}$ $b = 1 + \frac{z^2}{3 -  y }$                        | 1,426           | 1,22   | 3,5   |
| 10         | $a = \cos \ln x + \ln  y $ $b = \frac{1}{z + \frac{x^2}{y + z}} + e^z$  | 1,743           | 0,457  | 7,833 |



| № варианта | Расчетные формулы   | Исходные данные |        |        |
|------------|---|-----------------|--------|--------|
|            |   | x               | y      | z      |
| 11         | $a = \left  x^{\frac{y}{x}} - \sqrt[3]{\frac{y}{x}} \right $ $b = (y-x) \left  \frac{y - \frac{z}{y-x}}{1 + (y-x)^2} \right $ | 1,825           | 18,225 | -3,298 |
| 12         | $a = \frac{y^2 x + e^{-x} \cos zx}{zx - e^{-x} \sin zx + 1}$ $b = e^{2x} \ln(y + x - z^{3x} \ln(z - x))$                      | 0,3             | 0,5    | 2,9    |
| 13         | $a = \sqrt{x^2 + z} - b^2 \sin^3 \frac{(x+y)}{x}$ $b = \cos^2 x^3 - \frac{x}{\sqrt{y^2 + z^2}}$                               | -2,9            | 1,5    | 15,5   |

2. Вычислить значения искомых величин, формулы для вычислений и данные для тестирования определить самостоятельно.

| № варианта | Формулировка задачи   |
|------------|---|
| 1          | Вычислить периметр и площадь прямоугольного треугольника по заданным длинам двух катетов $a$ и $b$ .                  |
| 2          | Заданы координаты трех вершин треугольника $(x_1, y_1)$ , $(x_2, y_2)$ , $(x_3, y_3)$ . Найти его периметр и площадь. |
| 3          | Даны два числа. Найти среднее арифметическое кубов этих чисел.  |
| 4          | Вычислить расстояние между двумя точками с данными координатами $(x_1, y_1)$ и $(x_2, y_2)$ .                         |
| 5          | Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.                            |
| 6          | Дана сторона равностороннего треугольника. Найти площадь этого треугольника, его высоты.                              |
| 7          | Найти площадь кольца, внутренний радиус которого равен $r$ , а внешний – заданному числу $R$ ( $R > r$ ).             |

| № варианта | Формулировка задачи  |
|------------|--|
| 8          | Треугольник задан величинами своих углов и радиусом описанной окружности. Найти стороны треугольника.                            |
| 9          | Найти площадь равнобедренной трапеции с основаниями $a$ и $b$ и углом $\alpha$ при большем основании $a$ .                       |
| 10         | Найти все углы треугольника со сторонами $a$ , $b$ , $c$ . Предусмотреть в программе перевод радианной меры угла в градусы.      |
| 11         | Известна длина окружности. Найти площадь круга, ограниченного этой окружностью.  |
| 12         | Составить программу вычисления объема цилиндра и конуса, которые имеют одинаковую высоту $H$ и одинаковый радиус основания $R$ . |
| 13         | Три сопротивления $R_1$ , $R_2$ , $R_3$ соединены параллельно. Найти сопротивление соединения.                                   |

### Лабораторная работа № 3

## ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМОВ РАЗВЕТВЛЕННОЙ СТРУКТУРЫ

**Цель работы:** ознакомиться с понятием разветвленной алгоритмической структуры; приемами программной реализации на языке программирования Python; произвести отладку и тестирование полученных программ.

#### Задание:

1. Изучить краткие теоретические сведения.
2. Разработать алгоритмы для решения задач (блок-схемы).
3. Записать алгоритмы на языке программирования Python.
4. Произвести отладку и тестирование программы.
5. Оформить отчет по лабораторной работе.

**Отчет о лабораторной работе** должен содержать следующие сведения:

1. Название и цель работы.
2. Формулировку задачи, схему алгоритма, программный код и результаты решения задачи.
3. Вывод по работе в целом.

## Краткие теоретические сведения

**Алгоритм разветвляющегося вычислительного процесса** – алгоритм, в котором в зависимости от значений некоторого признака производится выбор одного из нескольких направлений, называемых ветвями. В основе организации разветвления лежит проверка логического условия, которое может быть истинно или ложно.

Алгоритм разветвляющейся структуры принято называть развилкой (альтернативой, выбором). Частный случай разветвления, когда одна ветвь не содержит никаких действий, называют **обходом** или **неполной развилки** (неполный выбор) рис. 10.

Схема базовой структуры ветвления (**полный выбор**) показана на рис. 11. На рис. 12 показана схема **множественного выбора** (для задач в которых более трех условий).

**Частный вид логического условия** – это операции типа  $=$ ,  $>$ ,  $<$ ,  $\neq$ ,  $\leq$ ,  $\geq$ . Допускается также объединение нескольких условий в одно при помощи логических операций и, или, не и др.

Логические значения в Python представлены двумя величинами – логическими константами True (Истина) и False (Ложь).

Логические значения получаются в результате логических операций и вычисления логических выражений (табл. 4).

Таблица 4

| Операция<br>или выра-<br>жение | Описание   |
|--------------------------------|--|
| $>$                            | Условие «больше» (например, проверяем, что $a > b$ )   |
| $<$                            | Условие «меньше» (например, проверяем, что $a < b$ )   |
| $==$                           | Условие равенства (проверяем, что $a$ равно $b$ )  |
| $!=$                           | Условие неравенства (проверяем, что $a$ не равно $b$ )   |
| not x                          | Отрицание (условие $x$ не выполняется)   |
| x and y                        | Логическое «И» (умножение). Чтобы выполнилось условие $x$ and $y$ , необходимо, чтобы одновременно выполнялись условия $x$ и $y$ |
| x or y                         | Логическое «ИЛИ» (сложение). Чтобы выполнилось условие $x$ or $y$ , необходимо, чтобы выполнилось одно из условий                |
| x in A                         | Проверка принадлежности элемента $x$ множеству (структуре) $A$   |
| $a < x < b$                    | Эквивалентно $(x > a)$ and $(x < b)$   |



Рис. 10. Неполный выбор

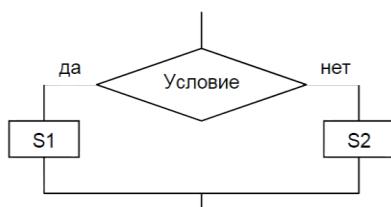


Рис. 11 Схема базовой структуры развилки

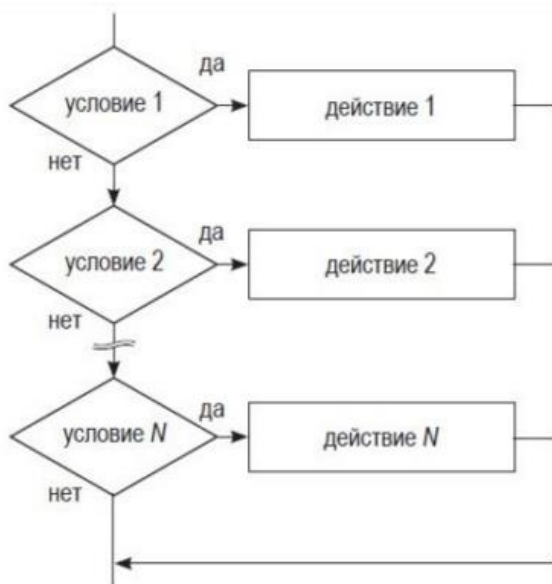


Рис. 12. Множественный выбор

## Программная реализация алгоритма разветвленной структуры

Часто, при реализации алгоритмов, нужно выполнять выбор хода выполнения программы в зависимости от заданного условия. Для обеспечения этого современные языки программирования (Java, Java Script, C#, C++ и прочие) содержат соответствующие управляющие операторы (control statements), одним из которых есть оператор ветвления **if**.

Язык программирования Python также имеет в своем арсенале инструкцию **if**, работа которой ничем не отличается от такой же инструкции других языков программирования.

В языке Python инструкция **if** имеет следующие четыре формы:

- полная форма **if-elif-else**;
- сокращенная форма **if**;
- сокращенная форма **if-else**;
- сокращенная форма **if-elif**.

### Полная форма инструкции **if (if-elif-else)**. Принцип выполнения

В языке программирования Python общая форма инструкции **if** имеет вид:

```
if выражение1 : набор1
elif выражение2 : набор2
...
elif выражениеN : наборN
else:
    набор N+1
```

Здесь **выражение1**, **выражение2**, **выражениеN** – соответствующее условное выражение, которое проверяется на истинность. Если условное выражение истинно, то его значение равно **True** (истина).

В этом случае выполняется соответствующий набор (множество) операторов **набор1**, **набор2**, **наборN**, **набор N+1**.

Инструкция **if** работает следующим образом:

Если **выражение1** равно **True**, то выполняется **набор1** операторов, в противном случае происходит переход к проверке условия

**выражение2**. По такому же принципу обрабатываются ветви **выражение2**, ..., **выражениеN**.

Если ни одно из выражений **выражение1**, **выражение2**, ..., **выражениеN** не выполняется (равно **False**), то выполняется **наборN+1**, который следует после ключевого слова **else**.

### **Сокращенная форма if. Принцип выполнения**

Сокращенная форма инструкции **if** не содержит блоков **elif** и **else** и имеет следующий вид:

**if выражение: набор**

В этом случае **набор** операторов выполняется если выражение истинно, то есть равно **True**. В противном случае, выполняется следующий после **if** оператор.

### **Сокращенная форма if-else. Принцип выполнения**

Сокращенная форма может иметь общий вид if-else:

**if выражение: набор1**  
**else: набор2**

Приведенная выше форма оператора **if** работает по следующему принципу. Сначала проверяется значение *выражение*. Если *выражение* истинно (равно **True**), то выполняется **набор1** операторов. В противном случае (выражение равно **False**) выполняется **набор2** операторов.

### **Сокращенная форма if-elif. Принцип выполнения**

Сокращенная форма **if-elif** может иметь следующий вид:

**if выражение1 : набор1**  
**elif выражение2 : набор2**  
...  
**elif выражениеN : наборN**

В этой форме оператор **if** подобен оператору **switch** в других языках программирования. Оператор **if-elif** работает по следующему принципу:

Если значение **выражение1** истинно (равное **True**), то выполняется **набор1** и происходит выход из оператора **if-elif**. В противном случае (значение **выражение1** равно **False**) происходит переход к следующей ветви оператора на проверку **выражение2**. Таким образом, выполняется та ветвь оператора **if-elif**, значение выражения в которой равно **True**. Если ни одно из выражений **выражение1, выражение2, ..., выражениеN** не выполняется, тогда происходит переход к следующему после **if-elif** оператору.

### Пример программной реализации конструкции **if-elif-else**(полная форма)

**Задача 1:** Задано целочисленное значение  $n=1..3$  и радиус  $r$ . По заданному значению  $n$  вычислить: 1 – длина окружности; 2 – площадь круга; 3 – объем шара.

#### Текст программы:

```
# -*- coding: utf-8 -*-
import math
print("Значение n =1..3")
n=int(input('Введите значение N='))
r=float(input('Введите значение радиуса r='))
# вычисление, оператор if
if n==1:
    s=2*math.pi*r
    print("Длина окружности = ", "%4.2f"%s)
elif n==2:
    s=math.pi*pow(r,2)
    print("Площадь круга = ", "%4.2f"%s)
elif n==3:
    s=4.0/3*math.pi*pow(r,3)
    print("Объем шара = ", "%4.2f"%s)
else:
    print("Неверно введены данные")
print("-----")
```

В результате выполнения вышеприведенного кода будет получен следующий результат:

```
Введите значение N=1
Введите значение радиуса r=6.9
Длина окружности = 43.35
-----
```

## Пример использования сокращенной формы инструкции **if**

**Задача 2.** Решить квадратное уравнение для заданных коэффициентов  $a$ ,  $b$ ,  $c$ .

В примере демонстрируется использование оператора **if** для нахождения решения квадратного уравнения. Код приведен для программного режима. С помощью функции `print()` выводится результат вычислений и результат проверки. Если значение дискриминанта  $d < 0$ , то выводится сообщение «Уравнение не имеет корней».

### Текст программы:

```
# -*- coding: utf-8 -*-
a=2 # задать значение a,b,c
b=5
c=-4
d=b*b-4*a*c # вычислить d
if d>=0: # оператор if
    x1=(-b+(d**0.5))/(2*a)
    x2=(-b-(d**0.5))/(2*a)
    print("d = ",d) # вывод результата
    print("x1 = ", x1)
    print("x2 = ", x2)
    print
    print("Проверка:")
    print("a*x1*x1+b*x1+c = ", a*x1*x1+b*x1+c)
    print("a*x2*x2+b*x2+c = ", a*x2*x2+b*x2+c)
else:
    print("уравнение не имеет решения")
```

Как видно из примера, если в инструкции **if** нужно выполнить несколько операторов подряд (сгруппировать), то они выделяются с помощью отступов.

После запуска на выполнение, в среде **Python** будет получен следующий результат:

```
d = 57
x1 = -3.1374586088176875
x2 = 0.6374586088176875
Проверка:
a*x1*x1+b*x1+c = 0.0
a*x2*x2+b*x2+c = 0.0
```



### Пример использования вложенной инструкции if-elif.

**Задача 3.** Определить количество дней в месяце и наличие високосного года.

В данном примере демонстрируется использование вложенной инструкции **if-elif** для проведения расчета количества дней в месяце на основе следующих входных данных:

- год;
- номер месяца в году.

#### Текст программы:

```
year=2108 # установить некоторый год
month=2 # установить некоторый месяц
#
# определение високосного года
f_leap_year=True # в начале принять, что год високосный
if year%400==0:
    print(year, "- год високосный")
elif year%100==0:
    print(year, "- год невисокосный")
    f_leap_year=False
elif year%4==0:
    print(year, "- год високосный")
else:
    print(year, "- год невисокосный")
    f_leap_year=False
print("-----")
#
# определение количества дней в месяце
if month==2:
    if f_leap_year==True: # вложенный оператор if-else
        nDays=29
    else:
        nDays=28
elif (month==4)or(month==6)or(month==9)or(month==11):
    nDays=30
else:
    nDays=31
print("Количество дней в месяце:", month)
print("nDays=", nDays)
```

При определении количества дней для месяца 2 (февраль), используется вложенная инструкция **if-else**. Вложенность определяется с помощью отступов:

```
...
if month==2:
    if f_leap_year==True: # вложенный оператор if-else
        nDays=29
    else:
        nDays=28
elif ...
```

После запуска на выполнение программа выдаст следующий результат:

```
2108 - год високосный
-----
Количество дней в месяце: 2
nDays= 29
```

### Задачи для самостоятельной работы

1. Ввести с клавиатуры вещественные числа  $a$ ,  $b$ ,  $c$ . Определить максимальное (минимальное) значение из трех выражений.

| № варианта | Критерий поиска | Выражение 1 | Выражение 2 | Выражение 3 |
|------------|-----------------|-------------|-------------|-------------|
| 1          | максимум        | $a+b+c$     | $abc$       | $a-b$       |
| 2          | минимум         | $a+2b$      | $b+3c$      | $c$         |
| 3          | минимум         | $2a-b$      | $c^2$       | $b+c$       |
| 4          | максимум        | $a/c$       | $b+c$       | $5c$        |
| 5          | минимум         | $3a$        | $a+b+7c$    | $8c$        |
| 6          | максимум        | $5a+7b$     | $b-c$       | $3c$        |
| 7          | минимум         | $a^2$       | $a+2b-c$    | $c$         |
| 8          | максимум        | $a+b+c$     | $3a+8$      | $a+7c$      |
| 9          | минимум         | $ab$        | $ac$        | $bc$        |
| 10         | максимум        | $a$         | $a^2-b$     | $ac$        |

## 2. Вычислить значение функции для данного $x$ :

| № варианта | Функция   | № варианта | Функция   |
|------------|---|------------|---|
| 1          | $y = \begin{cases} 2x+5, x < 0 \\ x^2+6\cos(x), x = 0 \\ x^3+x, x > 0 \end{cases}$                  | 6          | $y = \begin{cases} \sqrt{2x+5}, x < 3 \\ (x^2+6)^3, x = 3 \\ 2x+1, x > 3 \end{cases}$       |
| 2          | $y = \begin{cases} \sqrt{2x+5x^2}, x < 2 \\ x^2+6x+1, x = 2 \\ \frac{x^3+1}{4x}, x > 2 \end{cases}$ | 7          | $y = \begin{cases} x+5x, x < 0 \\ (x^2+6)^2, x = 0 \\ \frac{x^2+x}{x+2}, x > 0 \end{cases}$ |
| 3          | $y = \begin{cases} (2x+5)^2, x < 4 \\ x^2+6, x = 4 \\ \sqrt{x^3+x}, x > 4 \end{cases}$              | 8          | $y = \begin{cases} \frac{2x+5}{3x^3}, x < 5 \\ 6x^2+6, x = 5 \\ x^2+2, x > 5 \end{cases}$   |
| 4          | $y = \begin{cases} \frac{4x+2}{2x}, x < 3 \\ \sqrt{5x^2+6x}, x = 3 \\ x^3+x, x > 3 \end{cases}$     | 9          | $y = \begin{cases} \frac{2x+5}{x}, x < 0 \\ x^2+6\cos(x), x = 0 \\ x^3, x > 0 \end{cases}$  |
| 5          | $y = \begin{cases} 5x+9, x < 0 \\ x^3+6x^2, x = 0 \\ x^3+x^2+x, x > 0 \end{cases}$                  | 10         | $y = \begin{cases} 2x, x < 2 \\ x^2+1, x = 2 \\ x^3+4, x > 2 \end{cases}$                   |

## 3. Разработать алгоритм и программу для решения задачи

| № варианта | Формулировка задачи  |
|------------|--|
| 1          | По номеру дня недели (натуральному числу) выдать в качестве результата количество и перечень занятий в Вашей группе  |
| 2          | По заданному году и номеру месяца определить количество дней в этом месяце   |
| 3          | Для каждой введенной цифры вывести соответствующее ей название на английском языке (0 – zero, 1 – one, 2 – two, ...) |
| 4          | По данному числу (1–12) вывести название соответствующего месяца и число дней в нем                                  |
| 5          | Получить словесное описание школьных отметок (1 – плохо, 2 – неудовлетворительно, ...)                               |
| 6          | По номеру месяца выдать название следующего за ним месяца (при $m = 1$ получаем февраль)                             |

| № варианта | Формулировка задачи  |
|------------|--|
| 7          | По введенному номеру времени года (1 – зима, 2 – весна, 3 – лето, 4 – осень) выдать соответствующие этому времени года месяцы, количество дней в каждом из месяцев |
| 8          | По вводимому числу от 1 до 5 (номеру курса) выдать соответствующее сообщение «Привет, $k$ -курсник». Например, при $k = 1$ : «Привет, однокурсник»                 |
| 9          | По данному натуральному числу от 1 до 12 (номеру месяца) выдать все приходящиеся на этот месяц праздничные дни   |
| 10         | Имеется пронумерованный список деталей: 1) шуруп, 2) гайка, 3) винт, 4) гвоздь, 5) болт. По номеру детали вывести на экран ее название                             |

## Лабораторная работа № 4

### ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМОВ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ

**Цель работы:** ознакомиться с понятием циклической алгоритмической структуры; приемами программной реализации на языке программирования Python; произвести отладку и тестирование полученных программ.

#### Задание:

1. Изучить краткие теоретические сведения.
2. Разработать алгоритмы для решения задач (блок-схемы).
3. Записать алгоритмы на языке программирования Python.
4. Произвести отладку и тестирование программы.
5. Оформить отчет по лабораторной работе.

**Отчет о лабораторной работе** должен содержать следующие сведения:

1. Название и цель работы.
2. Формулировку задачи, схему алгоритма, программный код и результаты решения задачи.
3. Вывод по работе в целом.

## Краткие теоретические сведения

**Циклом** называется фрагмент алгоритма или программы, который может **повторяться** несколько раз (в том числе и нуль раз). Каждая циклическая конструкция начинается заголовком цикла и заканчивается конечным оператором. Между ними располагаются операторы, называемые «**телом цикла**». Количество повторений выполнения команд (операторов), составляющих тело цикла, определяется **условием окончания цикла**. Условием окончания может быть достижение некоторого значения специальной переменной, называемой **параметром цикла** (переменной цикла), или выполнение (прекращение выполнения) некоторого условия.

В Python существуют два типа циклических выражений:

- Цикл **while**
- Цикл **for**

### Цикл **while** в Python

Инструкция **while** в Python повторяет указанный блок кода **пока** указанное в цикле логическое выражение будет оставаться **истинным** (рис. 13).

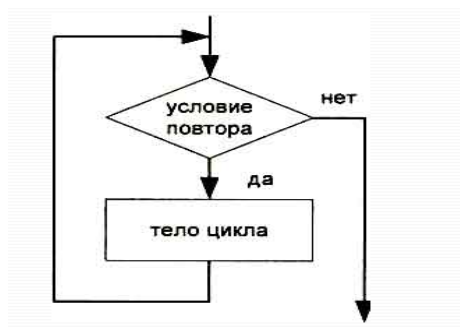


Рис. 13. Условное обозначение на схемах алгоритмов оператора **while**

Синтаксис цикла **while**:

**while** логическое выражение:

**команда 1**

**команда 2**

...

**команда n**

После ключевого слова **while** указывается условное выражение, и пока это выражение возвращает значение **True**, будет выполняться блок инструкций, который идет далее.

Все инструкции, которые относятся к циклу **while**, располагаются на последующих строках и должны иметь отступ от начала строки (4 пробела).

### Цикл for в Python:

Цикл **for** в Python обладает способностью перебирать элементы любого комплексного типа данных (например, строки или списка).

Синтаксис цикла **for**:

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

```
    команда 1
```

```
    команда 2
```

```
    ...
```

```
    команда n
```

Переменной **i** присваивается значение первого элемента функции **range()**, после чего выполняются команды. Затем переменной **i** присваивается следующее по порядку значение и так далее до тех пор, пока не будут перебраны все элементы функции **range()** (рис. 14).

Функция **range()** является универсальной функцией Python для создания списков (list) содержащих арифметическую прогрессию. Чаще всего она используется в циклах **for**.

**range**(старт, стоп, шаг) – так выглядит стандартный вызов функции **range()** в Python. По умолчанию старт равняется нулю, шаг единице.

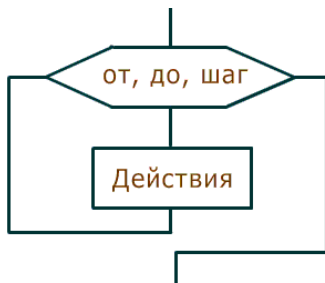


Рис. 14. Условное обозначение на схемах алгоритмов оператора **For**

**Задача 1.** Вычислить  $y = \frac{a^3}{a^2 + x^2}$ ,  $x$  с промежутка  $[-3, 10]$

с шагом 0,5.

**Постановка задачи:** Имеется переменная  $x$ , изменяющаяся на диапазон от  $-3$  д  $10$  с шагом 0,5. Значение переменной  $a$  не задано и вводится с клавиатуры пользователем. Требуется протабулировать значение функции  $y = \frac{a^3}{a^2 + x^2}$ .

**Метод решения:** ввести значение переменной  $a$  (вещественного типа), задать начальное значение параметра цикла  $x = -3$ , установить условие выполнения цикла (пока  $x \leq 10$  выполняется «тело цикла»), изменить начальное значение цикла на единицу шага 0,5, блок-схема алгоритма (рис. 15).

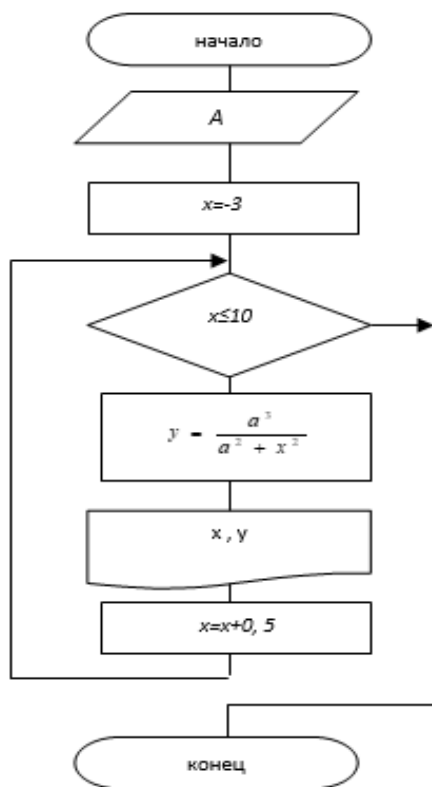


Рис. 15. Блок-схема решения задачи

## Текст программы на Python

```
# -*- coding: utf-8 -*-
x=-3
a=float(input('введите значение a= '))
while (x<=10):
    y=pow(a,3)/(pow(a,2)+pow(x,2))
    print('при x=',x,' y=', "%4.2f"%y)
    x+=0.5
```

### *Результат*

введите значение a= 6

при x= -3 y= 4.80  
при x= -2.5 y= 5.11  
при x= -2.0 y= 5.40  
при x= -1.5 y= 5.65  
при x= -1.0 y= 5.84  
при x= -0.5 y= 5.96  
при x= 0.0 y= 6.00  
при x= 0.5 y= 5.96  
при x= 1.0 y= 5.84  
при x= 1.5 y= 5.65  
при x= 2.0 y= 5.40  
при x= 2.5 y= 5.11  
при x= 3.0 y= 4.80  
при x= 3.5 y= 4.48  
при x= 4.0 y= 4.15  
при x= 4.5 y= 3.84  
при x= 5.0 y= 3.54  
при x= 5.5 y= 3.26  
при x= 6.0 y= 3.00  
при x= 6.5 y= 2.76  
при x= 7.0 y= 2.54  
при x= 7.5 y= 2.34  
при x= 8.0 y= 2.16  
при x= 8.5 y= 2.00  
при x= 9.0 y= 1.85  
при x= 9.5 y= 1.71  
при x= 10.0 y= 1.59



**Задача 2.** Вычислить  $y = \frac{a^3}{a^2 + x^2}$ ,  $x$  с промежутка  $[-3, 10]$  с шагом 1.

**Постановка задачи:** Имеется переменная  $x$ , изменяющаяся на диапазон от  $-3$  до  $10$  с шагом 1. Значение переменной  $a$  не задано и вводится с клавиатуры пользователем. Требуется протабулировать значение функции  $y = \frac{a^3}{a^2 + x^2}$ .

**Метод решения:** ввести значение переменной  $a$  (вещественного типа), задать начальное значение параметра цикла  $x = -3$ , конечное значение цикла  $x = 10$ , шаг равен 1, сформировать «тело цикла», блок-схема алгоритма (рис. 16).

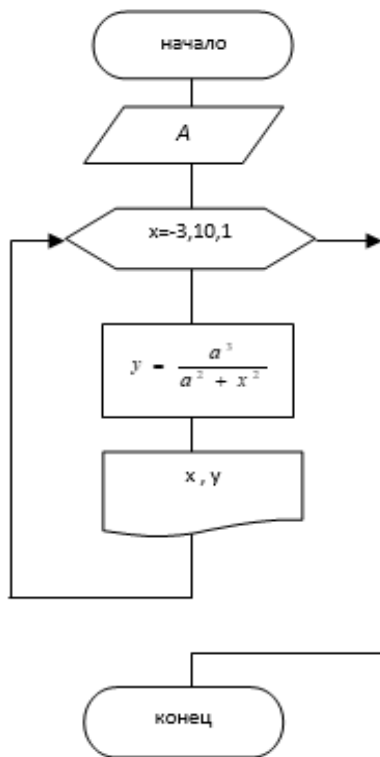


Рис. 16. Блок-схема решения задачи

## Текст программы на Python

```
# -*- coding: utf-8 -*-
a=float(input('введите значение a= '))
for x in range(-3, 10+1,1):
    y=pow(a,3)/(pow(a,2)+pow(x,2))
    print('при x=',x,' y=', "%4.5f"%y)
```

### *Результат*

```
введите значение a= 2.2
при x= -3 y= 0.76936
при x= -2 y= 1.20452
при x= -1 y= 1.82329
при x= 0 y= 2.20000
при x= 1 y= 1.82329
при x= 2 y= 1.20452
при x= 3 y= 0.76936
при x= 4 y= 0.51094
при x= 5 y= 0.35684
при x= 6 y= 0.26072
при x= 7 y= 0.19777
при x= 8 y= 0.15468
при x= 9 y= 0.12404
при x= 10 y= 0.10156
```

**Задача 3.** В переменную  $a$  последовательно вводятся числа -9, 0, 2, 6, 7, 11, -6. Найти среднее арифметическое всех чисел до тех пор, пока не встретится число  $N$ .

**Постановка задачи:** В переменную  $a$  последовательно вводятся числа -9, 0, 2, 6, 7, 11, -6. Найти среднее арифметическое всех чисел до тех пор, пока не встретится число  $N$ .

**Метод решения:** организовать цикл **for** для последовательного ввода заданных чисел в переменную  $a$  (не список, не кортеж). Для элементов не равных  $N$ , найти среднее арифметическое, результат показать. Для нахождения среднего арифметического вводятся 3 переменных;  $S$  – сумма элементов по условию,  $k$  – количество элементов по условию,  $sr$  – среднее значение по условию. Для этих переменных устанавливаются начальные значения, равные 0. Блок-схема алгоритма (рис. 17).

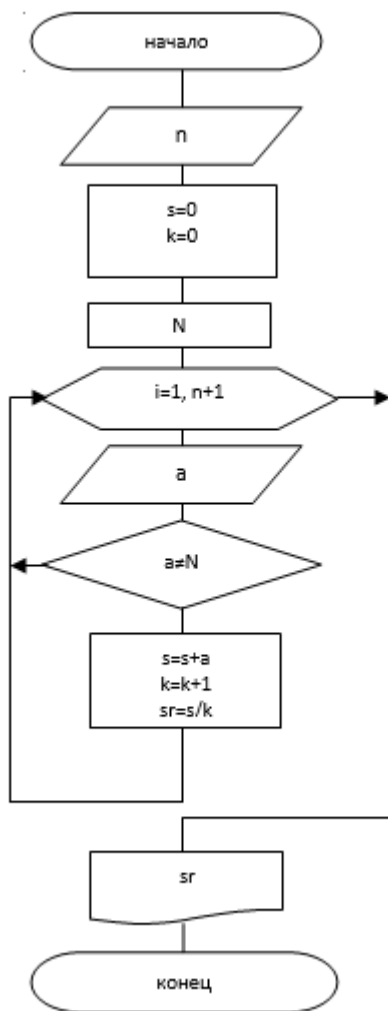


Рис. 17. Блок-схема решения задачи

### Текст программы на Python

```

# -*- coding: utf-8 -*-
n=int(input('Введите количество элементов: '))
s=0
k=0

```

```

N=int(input('введите критическое значение= '))
for i in range(1,n+1):
    a=int(input('введите элемент '))
    if a!=N:
        s=s+a
        k=k+1
        sr=s/k
    else:
        print('среднее арифметическое элементов до',N,'-',sr)
        break

```

### *Результат*

```

Введите количество элементов: 7
введите критическое значение= 11
введите элемент -9
введите элемент 0
введите элемент 2
введите элемент 6
введите элемент 7
введите элемент 11
среднее арифметическое элементов до 11 - 1.2

```

При работе с циклами используются операторы ***break*** и ***continue***. Оператор ***break*** предназначен для досрочного прерывания работы цикла ***while***.

### **Пример.**

```

# -*- coding: utf-8 -*-
a = 0
while a >= 0 :
    if a == 7 :
        break
    a += 1
    print ("A")

```

### *Результат*

```

A
A
A
A

```

A  
A  
A

Оператор *continue* запускает цикл заново, при этом код, расположенный после данного оператора, не выполняется.

### Пример.

```
# -*- coding: utf-8 -*-
```

```
a = - 1
```

```
while a < 10 :
```

```
    a += 1
```

```
    if a >= 7 :
```

```
        continue
```

```
    print ("A")
```

### Результат

A  
A  
A  
A  
A  
A  
A

При запуске данного кода символ "A" будет напечатан 7 раз, несмотря на то, что всего будет выполнено 11 проходов цикла.

В приведенном выше коде, выход из цикла произойдет при достижении переменной *a* значения 7. Если бы не было этого условия, то цикл выполнялся бы бесконечно.

## Задачи для самостоятельной работы

### Вариант 1

1. Найти значение выражения  $y = \frac{2x+3}{x^2}$  при  $x \in [2, 15]$  с шагом 1.

2. Найти значение выражения  $y = \frac{2x+3}{x^2}$  при  $x \in [2, 5]$  с шагом 0,2.

3. В переменную  $a$  последовательно вводятся числа -5, 3, 2, 9, 6, 11, -6. Найти среднее арифметическое всех чисел до тех пор, пока не встретится число 6.

4. С клавиатуры последовательно вводятся восемь целых чисел (2, 3, -4, 5, 7, 10, 16, 12). Найти и вывести на печать минимальное из чисел.

### *Вариант 2*

1. Найти значение выражения  $y = \frac{x^3 + 3}{x^2}$  при  $x \in [4, 15]$  с шагом 1.

2. Найти значение выражения  $y = \frac{x^3 + 3}{x^2}$  при  $x \in [4, 10]$  с шагом 0,5.

3. В переменную  $a$  последовательно вводятся числа -9, 18, 2, 6, 7, 0, 11, -6. Найти произведение всех чисел до тех пор, пока не встретится 0.

4. С клавиатуры последовательно вводятся восемь целых чисел (2, 3, -4, 5, 7, 10, 16, 12). Найти и вывести на печать максимальное из чисел.

### *Вариант 3*

1. Найти значение выражения  $y = 2x^3 + 16x$  при  $x \in [1, 9]$  с шагом 1.

2. Найти значение выражения  $y = 2x^3 + 16x$  при  $x \in [1, 9]$  с шагом 0,25.

3. В переменную  $a$  последовательно вводятся числа -9, 18, 2, 6, 7, 0, 11, -6. Найти сумму всех чисел до тех пор, пока не встретится 0.

4. С клавиатуры последовательно вводятся восемь целых чисел (10, 15, 23, -45, 88, -100, 1, 2). Найти и вывести на печать минимальное из чисел.

### *Вариант 4*

1. Найти значение выражения  $y = \frac{x^4 + 3}{3x}$  при  $x \in [4, 15]$  с шагом 1.

2. Найти значение выражения  $y = \frac{x^4 + 3}{3x}$  при  $x \in [4, 10]$

с шагом 0,1.

3. В переменную  $a$  последовательно вводятся числа 7, 18, 2, 6, 7, -9, 0, 11, -6. Найти сумму всех чисел до тех пор, пока не встретится первое отрицательное число.

4. С клавиатуры последовательно вводятся восемь целых чисел (25, 14, 10, 5, 13, 16, 17, 98). Найти и вывести на печать максимальное из чисел.

### *Вариант 5*

1. Найти значение выражения  $y = x^5 + 5\sqrt{x}$  при  $x \in [1, 14]$

с шагом 1.

2. Найти значение выражения  $y = x^5 + 5\sqrt{x}$  при  $x \in [1, 4]$

с шагом 0,25.

3. В переменную  $a$  последовательно вводятся числа 7, 18, 2, 6, 7, -9, 0, 11, -6. Найти среднее арифметическое всех чисел до тех пор, пока не встретится первое отрицательное число.

4. С клавиатуры последовательно вводятся восемь целых чисел (-1, 2, 5, 6, 7, 9, -10, 19). Найти и вывести на печать минимальное из чисел.

### *Вариант 6*

1. Найти значение выражения  $y = x\sqrt{x+5}$  при  $x \in [-4, 10]$

с шагом 1.

2. Найти значение выражения  $y = x\sqrt{x+5}$  при  $x \in [-4, 4]$

с шагом 0,25.

3. В переменную  $a$  последовательно вводятся числа 6, 18, 2, 6, -4, -9, 0, 11, -6. Найти среднее арифметическое всех чисел до тех пор, пока не встретится первое число, не кратное 2.

4. С клавиатуры последовательно вводятся восемь целых чисел (15, 12, 16, 79, 87, 87, 0, -2). Найти и вывести на печать максимальное из чисел.

### *Вариант 7*

1. Найти значение выражения  $y = x + x^2 + 2$  при  $x \in [-6, 6]$

с шагом 1.

2. Найти значение выражения  $y = x + x^2 + 2$  при  $x \in [-4, 6]$  с шагом 0,2.

3. В переменную  $a$  последовательно вводятся числа -9, 0, 11, -7, 7, 18, 6, -4. Найти среднее арифметическое всех чисел до тех пор, пока не встретится первое число, кратное 2.

4. С клавиатуры последовательно вводятся восемь целых чисел (15, 12, 16, 79, 87, 88, 0, -2). Найти и вывести на печать минимальное из чисел.

### *Вариант 8*

1. Вычислить значение выражения  $y = \frac{3a + 4}{a^2 - 5a - 90\sqrt{a}}$  для  $a = 1 \dots 10$ .

2. Вычислить значение выражения  $y = \frac{3a + 4}{a^2 - 5a - 90\sqrt{a}}$ , где  $a$  изменяется от 1 до 5 с шагом 0,5.

3. В переменную  $a$  последовательно вводятся числа 7, 18, 2, 6, 7, -9, 16, 11, -6, 12. Найти произведение всех чисел до тех пор, пока не встретится первое число, кратное 4.

4. С клавиатуры последовательно вводятся восемь целых чисел (2, 3, -4, 5, 7, 10, 16, 12). Найти и вывести на печать максимальное из чисел.

### *Вариант 9*

1. Вычислить значение выражения  $y = \frac{\sqrt{a^2 + 4}}{a - 9}$  для  $a = -5 \dots 8$ .

2. Вычислить значение выражения  $y = \frac{\sqrt{a^2 + 4}}{a - 9}$ , где  $a$  изменяется от -1 до 5 с шагом 0,5.

3. В переменную  $a$  последовательно вводятся числа -9, 18, 2, 6, 7, 20, 11, -6. Найти произведение всех чисел до тех пор, пока не встретится 20.

4. С клавиатуры последовательно вводятся восемь целых чисел (1, 12, 16, 79, 77, -87, 0, -2). Найти и вывести на печать минимальное из чисел.



### Вариант 10

1. Найти значение выражения  $y = x^2\sqrt{2x+5}$  при  $x \in [1, 10]$  с шагом 1.
2. Найти значение выражения  $y = x^2\sqrt{2x+5}$  при  $x \in [1, 20]$  с шагом 2.
3. В переменную  $a$  последовательно вводятся числа -9, 18, 2, 6, 7, 20, 11, -6. Найти сумму всех чисел до тех пор, пока не встретится 11.
4. С клавиатуры последовательно вводятся восемь целых чисел (12, 3, -4, 5, 7, 10, 16, 12). Найти и вывести на печать максимальное из чисел.

### Вариант 11

1. Найти значение выражения  $y = \frac{\sqrt{x+4}}{x^3}$  при  $x \in [4, 15]$  с шагом 1.
2. Найти значение выражения  $y = \frac{\sqrt{x+4}}{x^3}$  при  $x \in [4, 6]$  с шагом 0,1.
3. В переменную  $a$  последовательно вводятся числа -9, 18, 2, 6, 7, 20, 11, -6. Найти произведение всех чисел до тех пор, пока не встретится 11.
4. С клавиатуры последовательно вводятся восемь целых чисел (15, 12, 16, 79, -87, 87, 0, -2). Найти и вывести на печать минимальное из чисел.

### Вариант 12

1. Найти значение выражения  $y = x^5 + 5\sqrt{x}$  при  $x \in [1, 14]$  с шагом 1.
2. Найти значение выражения  $y = x^5 + 5\sqrt{x}$  при  $x \in [1, 4]$  с шагом 0,25.
3. В переменную  $a$  последовательно вводятся числа -9, 0, 2, 6, 7, 11, -6. Найти среднее арифметическое всех чисел до тех пор, пока не встретится число 7.
4. С клавиатуры последовательно вводятся восемь целых чисел (12, 3, -45, 5, 79, 10, 16, 12). Найти и вывести на печать максимальное из чисел.

### Вариант 13

1. Вычислить  $y = \frac{a+4}{a^2+xa}$ ,  $x$  с промежутка  $[2, 14]$  с шагом 1.

2. Вычислить  $y = \frac{a+4}{a^2+xa}$ ,  $x$  с промежутка  $[2, 4]$  с шагом 0,5.

3. В переменную  $a$  последовательно вводятся числа 12, 6, 7, 11, -6, 10, 15, -4. Найти среднее арифметическое всех чисел до тех пор, пока не встретится отрицательное число.

4. С клавиатуры последовательно вводятся восемь целых чисел (15, -1, 16, -7, 77, 87, 0, -2). Найти и вывести на печать минимальное из чисел.

### Вариант 14

1. Вычислить  $y = \frac{a^3}{a^2+x^2}$ ,  $x$  с промежутка  $[-3, 10]$  с шагом 1.

2. Вычислить  $y = \frac{a^3}{a^2+x^2}$ ,  $x$  с промежутка  $[3, 10]$  с шагом 0,5.

3. В переменную  $a$  последовательно вводятся числа 12, 6, 7, 11, -6, 0, 10, 15. Найти произведение всех чисел до тех пор, пока не встретится 0.

4. С клавиатуры последовательно вводятся восемь целых чисел (12, 3, -4, 18, 79, 100, 16, 12). Найти и вывести на печать максимальное из чисел.

## Лабораторная работа № 5

### ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ОБРАБОТКИ ОДНОМЕРНЫХ МАССИВОВ

**Цель работы:** ознакомиться с понятием массива; приемами программной реализации на языке программирования Python; произвести отладку и тестирование полученных программ.

#### Задание:

1. Изучить краткие теоретические сведения.
2. Разработать алгоритмы для решения задач (блок-схемы).
3. Записать алгоритмы на языке программирования Python.
4. Произвести отладку и тестирование программы.
5. Оформить отчет по лабораторной работе.

**Отчет о лабораторной работе** должен содержать следующие сведения:

1. Название и цель работы.
2. Формулировку задачи, схему алгоритма, программный код и результаты решения задачи.
3. Вывод по работе в целом.

### **Краткие теоретические сведения**

Для того чтобы было удобно работать с большим количеством данных, обычно дают общее имя группе переменных, которая называется **массивом**.

**Массив** – это группа переменных одного типа, расположенных в памяти друг за другом и имеющих общее имя.

Имена (идентификаторы) массивов строятся по тем же правилам, что и имена переменных.

Перед тем как использовать массив, надо присвоить ему имя, определить тип входящих в массив переменных (элементов массива) и их количество. По этим сведениям компьютер вычислит, сколько места требуется для хранения массива, и выделит в памяти нужное число ячеек. Для работы с массивами необходимо научиться: выделять память нужного размера под массив; записывать данные в нужную ячейку массива; читать данные из ячейки массива.

### ***Массивы в языке Python***

В языке Python нет такой структуры данных как «массив».

Для организации массива в Python можно использовать такие структуры данных, как списки, кортежи, множества и диапазоны, которые представляют нумерованные наборы объектов.

Каждый элемент набора содержит ссылку на объект, который, в свою очередь, может представлять объект произвольного типа данных и иметь неограниченную степень вложенности.

В решении этого задания для хранения однотипных данных (массивов данных) предлагается использовать структуру данных, которая называется список. Список представляет собой последовательность элементов, пронумерованных от 0 (нуля). Элементы списка могут иметь различные типы. Список можно задать перечислением элементов списка в квадратных скобках, например, список можно задать так:

```
A = [1, 3, 5, 7, 11, 13]
B = ['Red', 'Orange', 'Yellow', 'Green']
```

или так:

```
C = [1, 3, 'Old', 7, 'Or', 13]
```

В списке A – 6 элементов, а именно: A[0] == 1, A[1] == 3, A[2] == 5, A[3] == 7, A[4] == 11, A[5] == 13.

Список B состоит из 4 элементов, каждый из которых является строкой, а вот список C состоит из шести элементов, часть которых – число, а часть – строка.

При обращении к элементам списка можно использовать как положительные индексы, так и отрицательные. Если индекс положительный, то счет ведется от нуля до максимального элемента, слева направо. Если индекс отрицательный, то счет ведется справа налево: A[-1] == 13, A[-6] == 1.

Количество элементов в списке (длину списка), можно получить при помощи функции **len**, например, len(A) == 6.

Существует несколько способов работы со списком. Разработчики предлагают различные варианты от специальных модулей до библиотек.

```
# -*- coding: utf-8
# Генераторы списков
N = 10
A = [i for i in range(N)]
print(A)
A = list(range(N))
print(A)

A = [i*i for i in range(N)]
print(A)

A = [i for i in range(100)
     if i % 7 == 0]
print(A)
A = [i for i in range(100)
     if i % 7 == 0 and i % 10 == 1]
print(A)
```

*Результат*

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

[0, 7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98]

[21, 91]

# -\*- coding: utf-8

# **Вывод массива**

N = 10

A = [i for i in range(N)]

print(A)

for i in range(N):

print(A[i])

for x in A:

print(x, end=" ")

print()

print( \*A )

*Результат*

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

0

1

2

3

4

5

6

7

8

9

0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7 8 9

# -\*- coding: utf-8

**# Ввод массива**

N = 5

A = [0]\*N

print("Введите 5 элементов массива")

for i in range(N):

    A[i] = int(input())

print(A)

print("Введите 5 элементов массива")

A = [ int(input()) for i in range(N) ]

print(A)

print("Введите 5 элементов массива")

for i in range(N):

    print( "A[{ }]=".format(i), end="" )

    A[i] = int(input())

print(A)

*Результат*

Введите 5 элементов массива

3

4

5

1

6

[3, 4, 5, 1, 6]

Введите 5 элементов массива

-3

4

6

8

6

[-3, 4, 6, 8, 6]

Введите 5 элементов массива

A[0]=

1

A[1]=

4

A[2]=

5

A[3]=

7

A[4]=

8

[1, 4, 5, 7, 8]

# -\*- coding: utf-8

**#Заполнение массива случайными числами**

from random import randint

N = 25

A = [0]\*N

for i in range(N):

    A[i] = randint(20, 100)

print(A)

A = [randint(20, 100) for i in range(N)]

print(A)

*Результат*

[25, 78, 96, 41, 68, 78, 31, 70, 55, 43, 99, 62, 85, 88, 95, 63, 29,  
39, 74, 98, 53, 100, 97, 57, 71]

[29, 53, 68, 53, 39, 70, 38, 49, 76, 65, 89, 89, 20, 32, 84, 64, 27,  
35, 29, 71, 95, 38, 31, 72, 76]

# -\*- coding: utf-8

**# Поиск максимума из отрицательных элементов**

from random import randint

N = 10

A = [randint(-20,200) for i in range(N)]

print(A)

```

M = -21
for x in A:
    if x < 0 and x > M:
        M = x
if M == -21:
    print( "Нет отрицательных!" )
else:
    print( M )

```

```

M = A[0]
for x in A:
    if x < 0:
        if M >= 0 or x > M:
            M = x
if M >= 0:
    print( "Нет отрицательных!" )
else:
    print( M )
count = 0
for x in A:
    if x < 0:
        if count == 0 or x > M:
            M = x
        count += 1
if count == 0:
    print( "Нет отрицательных!" )
else:
    print( M )

```

```

B = [x for x in A if x < 0]
if B:
    print( max(B) )
else:
    print( "Нет отрицательных!" )

```

*Результат*

```
[-5, 36, 105, 185, 120, 98, 24, 189, 191, 153]
```

```
-5
```

```
-5
```

```
-5
```

```
-5
```



## Типовые задачи обработки одномерного массива

### *Сумма элементов массива по условию*

#### Пример 1

```
# -*- coding: utf-8 -*-
from random import randint
N=int(input('Количество элементов массива: '))
s=0
for i in range(1,N+1): #с указанием первого элемента массива
    a=randint(-35, 30)
    print('A['+i,']элемент массива A=',a)
    if a<0:
        s=s+a
print('Сумма отрицательных элементов массива a=',s)
```

#### *Результат*

```
Количество элементов массива: 5
A[ 1 ]элемент массива A= 15
A[ 2 ]элемент массива A= -7
A[ 3 ]элемент массива A= 8
A[ 4 ]элемент массива A= 16
A[ 5 ]элемент массива A= 10
Сумма отрицательных элементов массива a= -7
```

#### Пример 2

```
# -*- coding: utf-8 -*-
from random import randint
N=int(input('Количество элементов массива: '))
s=0
for i in range(N): #без указания первого элемента массива,
тогда 0-ой элемент

    a=randint(-35, 30)
    print('A['+i,']элемент массива A=',a)
    if a<0:
        s=s+a
print('Сумма отрицательных элементов массива a=',s)
```

### *Результат*

Количество элементов массива: 5

A[ 0 ]элемент массива A= 14

A[ 1 ]элемент массива A= 9

A[ 2 ]элемент массива A= 10

A[ 3 ]элемент массива A= 25

A[ 4 ]элемент массива A= 11

Сумма отрицательных элементов массива a= 0

### **Пример 3**

```
# -*- coding: utf-8 -*-
```

```
from random import randint
```

```
N=int(input('Количество элементов массива: '))
```

```
a=[0]*N #с использованием списка
```

```
print('Исходный массив')
```

```
for i in range(N):
```

```
    a[i]=randint(-35, 30)
```

```
print(a)
```

```
s = 0
```

```
for i in range(N):
```

```
    if a[i] < 0 :
```

```
        s += a[i]
```

```
print('Сумма отрицательных элементов массива a=',s)
```

### *Результат*

Количество элементов массива: 7

Исходный массив

[-10, -15, 28, -24, 16, 20, 13]

Сумма отрицательных элементов массива a= -49

### ***Количество элементов массива по условию***

```
# -*- coding: utf-8 -*-
```

```
from random import randint
```

```
N=int(input('Количество элементов массива: '))
```

```
k=0
```

```
for i in range(1,N+1):
```

```
    a=randint(-35, 30)
```

```
    print('A['+i,']элемент массива A=',a)
```

```

    if a<0:
        k=k+1
print('Количество отрицательных элементов массива A=',k)

```

*Результат*

Количество элементов массива: 7

```

A[ 1 ]элемент массива A= 22
A[ 2 ]элемент массива A= 13
A[ 3 ]элемент массива A= -29
A[ 4 ]элемент массива A= 9
A[ 5 ]элемент массива A= 11
A[ 6 ]элемент массива A= 16
A[ 7 ]элемент массива A= 2

```

Количество отрицательных элементов массива A= 1

*Произведение элементов массива по условию*

```

# -*- coding: utf-8 -*-
from random import randint
N=int(input('Количество элементов массива: '))
pr=1
for i in range(1,N+1):
    a=randint(-35, 30)
    print('A['+i,']элемент массива A=',a)
    if a<0:
        pr=pr*a
print('Произведение отрицательных элементов массива A=',pr)

```

*Результат*

Количество элементов массива: 10

```

A[ 1 ]элемент массива A= 3
A[ 2 ]элемент массива A= -34
A[ 3 ]элемент массива A= -15
A[ 4 ]элемент массива A= -11
A[ 5 ]элемент массива A= 15
A[ 6 ]элемент массива A= -13
A[ 7 ]элемент массива A= 12
A[ 8 ]элемент массива A= 19
A[ 9 ]элемент массива A= -10
A[ 10 ]элемент массива A= -27

```

Произведение отрицательных элементов массива A= 19691100

### ***Среднее арифметическое элементов массива по условию***

```
# -*- coding: utf-8 -*-
from random import randint
N=int(input('Количество элементов массива: '))
s=0
k=0
for i in range(1,N+1):
    a=randint(-35, 30)
    print('A['+i,']элемент массива A=',a)
    if a<0:
        s=s+a
        k=k+1
    sr=s/k
print('Среднее арифметическое отрицательных элементов мас-
сива A=',sr)
```

#### ***Результат***

Количество элементов массива: 10

A[ 1 ]элемент массива A= -4

A[ 2 ]элемент массива A= 22

A[ 3 ]элемент массива A= -33

A[ 4 ]элемент массива A= 3

A[ 5 ]элемент массива A= -17

A[ 6 ]элемент массива A= 10

A[ 7 ]элемент массива A= 22

A[ 8 ]элемент массива A= 15

A[ 9 ]элемент массива A= -17

A[ 10 ]элемент массива A= 17

Среднее арифметическое отрицательных элементов массива  
A= -17.75

#### ***Максимальный элемент массива и его порядковый номер***

```
coding: utf-8 -*-
```

```
from random import randint
```

```
N=int(input('Количество элементов массива: '))
```

```
max=-35
```

```
for i in range(1,N+1):
```

```
    a=randint(-35, 30)
```

```
    print('A['+i,']элемент массива A=',a)
```

```
    if a>max:
```

```

        max=a;k=i
    print('Максимальный элемент массива A=',max,'\n', 'порядко-
    вый номер=',k)

```

### *Результат*

Количество элементов массива: 10  
 A[ 1 ]элемент массива A= 13  
 A[ 2 ]элемент массива A= -22  
 A[ 3 ]элемент массива A= 10  
 A[ 4 ]элемент массива A= -29  
 A[ 5 ]элемент массива A= -27  
 A[ 6 ]элемент массива A= -3  
 A[ 7 ]элемент массива A= 17  
 A[ 8 ]элемент массива A= -30  
 A[ 9 ]элемент массива A= 1  
 A[ 10 ]элемент массива A= -22  
 Максимальный элемент массива A= 17  
 порядковый номер= 7

### *Сортировка массива методом пузырька*

```

from random import randint
N=int(input('Количество элементов массива: '))
a=[0]*N
print('Исходный массив')
for i in range(N):
    a[i]=randint(-35, 30)
print(a)
print('Массив после сортировки')
for i in range(N-1):
    for j in range(N-1-i):
        if a[j] > a[j+1]:
            a[j], a[j+1] = a[j+1], a[j]
print(a)

```

### *Результат*

Количество элементов массива: 10  
 Исходный массив  
 [19, -7, -31, 14, 27, -24, -29, 26, 21, 27]  
 Массив после сортировки  
 [-31, -29, -24, -7, 14, 19, 21, 26, 27, 27]

Стандартные решения языка для нашего примера достаточны.

**Задача:**

1. Образовать два массива  $X$  и  $Y$ , состоящие из 12 вещественных чисел.

2. Образовать массив  $Z$  по правилу:

$$Z_j = Y_j \times s - 2 \times X_j$$
$$s = \begin{cases} j & \text{при } \cos j \leq 0,3 \\ -j & \text{при } \cos j > 0,3 \end{cases}$$

3. Найти среднее арифметическое значение элементов массива  $X$  и массива  $Y$ .

4. Найти сумму элементов массива  $Y$ , удовлетворяющих условию  $-3,5 < x_i < 3,5$ .

5. Найти значение и порядковый номер наибольшего элемента массива  $Z$ .

6. Упорядочить массив  $Y$  по возрастанию значений его элементов.

7. Вывести на печать массив  $X$ , массив  $Z$ , массив  $Y$ , среднее арифметическое элементов массива  $X$  и массива  $Y$ , сумму элементов массива  $Y$ , удовлетворяющих заданному условию, значение и порядковый номер наибольшего элемента массива  $Z$ , упорядоченный массив  $Y$ .

**Постановка задачи:** необходимо образовать три массива, один из которых образуется по формуле из двух других массивов. Выполнить действия с этими массивами.

**Метод решения:** Переменные  $X$  и  $Y$  используем для создания массива. Массив  $Z$  формируем при помощи цикла, обрабатывая последовательно каждый элемент. Для нахождения среднего арифметического значения элементов массивов  $X$  и  $Y$  складываем все элементы каждого массива в переменной  $sum$  и делим на  $n$  – количество элементов в массиве. Для нахождения суммы элементов массива  $Y$  в цикле проверяем каждый элемент по условию и умножаем на переменную  $s$ , которая изначально равна 0. Если по окончании цикла переменная  $s$  равна 0, то на экран выводится сообщение об отсутствии удовлетворяющих условию элементов. Для нахождения наибольшего элемента массива  $Z$  и его порядкового номера вводим переменные  $max$  и  $pos$ . В цикле сравнивается каждый элемент, после чего тот, кто больше, присваивается нужной переменной. Для сортировки массива  $Y$  по возрастанию используем инструменты смены позиций элементов в списке (рис. 18).

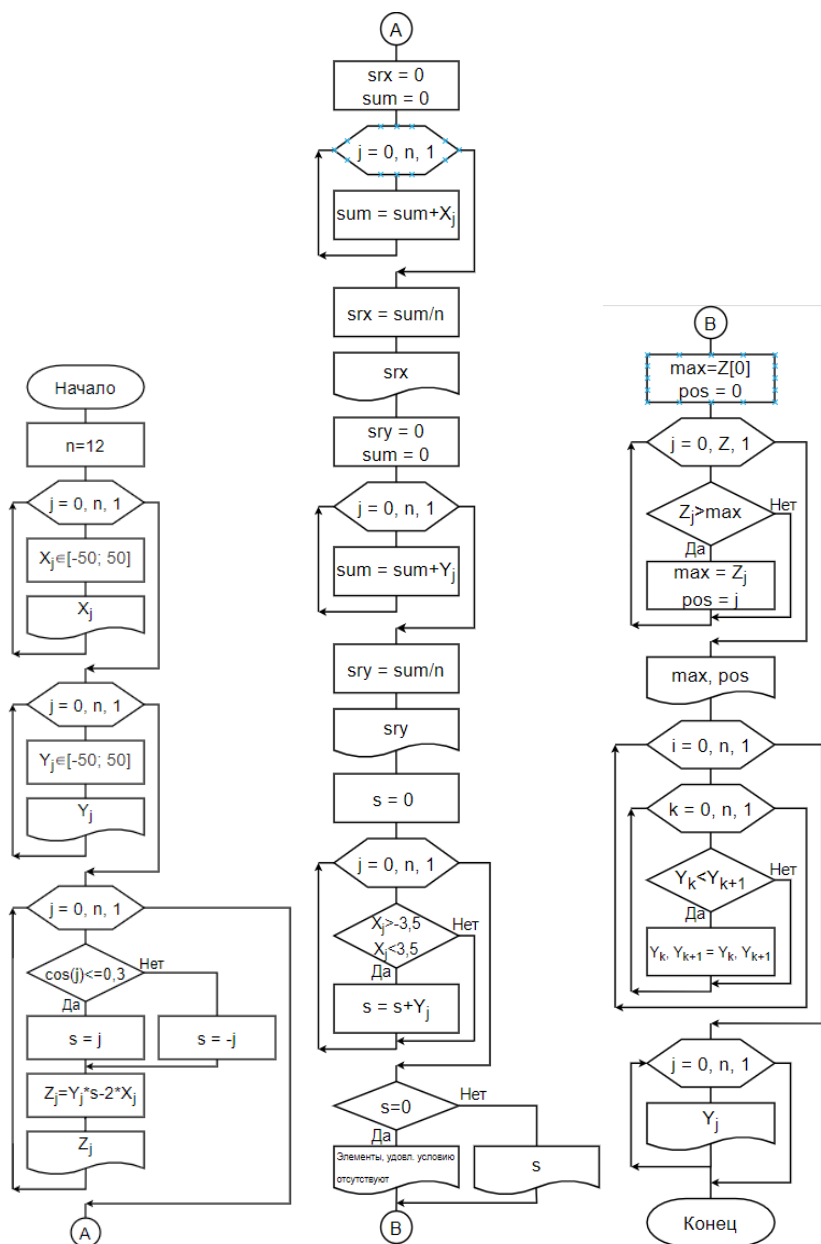


Рис. 18. Блок-схема решения задачи

## Текст программы на Python

```
# -*- coding: utf-8 -*-
import math
import random
n=12
print('Массив X:')
X=[]
for j in range(n):
    X.append(random.uniform(-50,50))
    print("{0: 7.2f}".format(X[j]), end=" ")
    print()
print('Массив Y:')
Y=[]
for j in range(n):
    Y.append(random.uniform(-50,50))
    print("{0: 7.2f}".format(Y[j]), end=" ")
    print()
Z=[0]*n
print('Массив Z:')
for j in range(n):
    if math.cos(j) <= 0.3: s=j
    else: s=-j
    Z[j]=Y[j]*s-2*X[j]
    print("{0: 7.2f}".format(Z[j]), end=" ")
    print()
srx = 0
sum = 0
for j in range(n):
    sum = sum + X[j]
srx = sum/n
print('Среднее арифметическое массива X: ', "%4.2f"%srx)
sry = 0
sum = 0
for j in range(n):
    sum = sum + Y[j]
sry = sum/n
print('Среднее арифметическое массива Y: ', "%4.2f"%sry)
s = 0
for j in range(n):
    if (X[j]> -3.5) and (X[j] < 3.5):
```



```

        s = s+Y[j]
    if s == 0 : print('Элементы, удовлетворяющие условию отсут-
ствуют')
    else: print('Сумма элементов массива Y, удовлетворяющих
условию:', "%4.2f"%s)
    max=Z[0]
    pos = 0
    for j in range(len(Z)):
        if Z[j]>max: max=Z[j]; pos=j
    print ('Наибольший элемент массива Z: ', "%4.2f"%max, ' его
номер: ', pos+1)
    for i in range(n):
        for k in range(n-1):
            if Y[k] > Y[k+1]:
                Y[k], Y[k+1] = Y[k+1], Y[k]
    print('Массив Y, отсортированный по возрастанию:')
    for j in range(n):
        print("{0: 7.2f}".format(Y[j]), end=" ")
    print ()

```

### *Результат*

Массив X:

```

16.94
 4.43
11.60
41.16
-21.76
-47.61
-42.41
21.17
-42.78
-40.10
-36.11
43.56

```

Массив Y:

```

-22.62
-10.77
16.39
13.64
25.49

```

48.51  
25.91  
39.42  
14.44  
-41.71  
24.76  
2.02  
Массив Z:

-33.88  
1.90  
9.58  
-41.40  
145.47  
337.76  
-70.63  
-318.25  
201.09  
-295.22  
319.81  
-64.93

Среднее арифметическое массива X: -7.66

Среднее арифметическое массива Y: 11.29

Элементы, удовлетворяющие условию отсутствуют

Наибольший элемент массива Z: 337.76 , его номер: 6

Массив Y, отсортированный по возрастанию:

-41.71  
-22.62  
-10.77  
2.02  
13.64  
14.44  
16.39  
24.76  
25.49  
25.91  
39.42  
48.51

## Задачи для самостоятельной работы

### Вариант 1

1. Образовать массив по правилу:

$$X_j = \begin{cases} \arctg \frac{\sqrt{j+2}}{n+2} & \text{при } tgj \leq 3 \\ e^{j+\cos n} & \text{при } tgj > 3, \end{cases}$$

2. Образовать массив  $Z$ , состоящий из 10 вещественных чисел.

3. Найти сумму элементов массивов  $X$  и  $Z$ .

4. Вычислить количество положительных элементов массива  $X$ .

5. Образовать массив  $Y$ , заменив в массиве  $Z$  положительные элементы 1, отрицательные  $-1$ .

6. Упорядочить массив  $Y$  по возрастанию значений его элементов.

7. Вывести на печать массив  $X$ , массив  $Z$ , массив  $Y$ , сумму элементов массива  $X$  и массива  $Z$ , количество положительных элементов массива  $X$ , упорядоченный массив  $Y$ .

### Исходные данные

$$n = 5$$

$$j = 1, 10$$

### Вариант 2

1. Образовать массив по правилу:

$$X_j = \begin{cases} \sin j & \text{при } \cos j < 0,5 \\ \cos j & \text{при } \cos j \geq 0,5. \end{cases}$$

2. Образовать массив  $Z$ , состоящий из 10 вещественных чисел.

3. Найти произведение элементов массива  $X$ ;  $j = 1, 10$ .

4. Вычислить количество положительных элементов массивов  $X$  и  $Z$  и вывести на печать.

5. Образовать массив  $Y$ , заменив в массиве  $X$  положительные элементы 1, а отрицательные нулем.

6. Упорядочить массив  $Y$  по убыванию значений его элементов.

7. Вывести на печать массив  $X$ , массив  $Z$ , массив  $Y$ , произведение элементов массива  $X$ , упорядоченный массив  $Y$ .

### Исходные данные

$$j = 1, 10$$

### Вариант 3

1. Образовать массив по правилу:

$$X_k = \begin{cases} k & \text{при } \cos k < 0,35 \\ 1/k & \text{при } \cos k \geq 0,35. \end{cases}$$

2. Образовать массив  $Z$ , состоящий из 10 вещественных чисел.

3. Найти произведение элементов каждого из массивов  $X$  и  $Z$ ;  $j = 1, 10$ .

4. Вычислить количество положительных элементов массива  $X$ , меньших 3,2 и вывести на печать.

5. Образовать массив  $Y$ , каждый элемент которого равен  $Y_k = X_k \times k$ .

6. Упорядочить массив  $Y$  по убыванию значений его элементов.

7. Вывести на печать массив  $X$ , массив  $Z$ , массив  $Y$ , произведение элементов массива  $X$  и массива  $Z$ , упорядоченный массив  $Y$ .

### Исходные данные

$$k = 1, 10$$

$$j = 1, 10$$

### Вариант 4

1. Образовать два массива  $A$  и  $B$ , состоящие из 12 вещественных чисел.

2. Образовать массив  $Z$  по правилу:

$$Z_i = a_i - b_i \times \sin i.$$

3. Найти произведение элементов массива  $A$  и сумму элементов массива  $Z$ .

4. Вычислить количество элементов массива  $B$ , меньших 3,2 и вывести на печать.

5. Образовать массив  $Y$ , каждый элемент которого равен  $Y_i = Z_i \cdot i$ .

6. Упорядочить массив  $Y$  по убыванию значений его элементов.

7. Вывести на печать массив  $A$ , массив  $B$ , массив  $Z$ , произведение элементов массива  $A$  и сумму элементов массива  $Z$ , массив  $Y$ , упорядоченный массив  $Y$ .

### Исходные данные

$$i = 1, 12$$

### Вариант 5

1. Образовать два массива  $X$  и  $Y$ , состоящие из 12 вещественных чисел.

2. Образовать массив  $Z$  по правилу:

$$Z_k = X_k + m \times Y_k,$$

$$m = \begin{cases} k & \text{при } \sin k \leq 0,3 \\ -k & \text{при } \sin k > 0,3 \end{cases}$$

3. Найти среднее арифметическое значение элементов массива  $Z$ ;  $k = 1, 12$ .

4. Найти произведение элементов массива  $X$ , удовлетворяющих условию  $-3,5 < x_i < 3,5$ .

5. Вычислить суммы отдельно отрицательных и положительных элементов массива  $Z$ .

6. Упорядочить массив  $Y$  по убыванию значений его элементов.

7. Вывести на печать массив  $X$ , массив  $Y$ , массив  $Z$ , произведение элементов массива  $X$ , среднее арифметическое значение элементов массива  $Z$ , сумму отрицательных и сумму положительных элементов массива  $Z$ , упорядоченный массив  $Y$ .

### Исходные данные

$i = 1, 12$

### Вариант 6

1. Образовать два массива  $X$  и  $Y$ , состоящие из 12 вещественных чисел.

2. Образовать массив  $Z$  по правилу:

$$Z_k = X_k^2 + m \times Y_k,$$

$$m = \begin{cases} k & \text{при } \cos k \leq 0,3 \\ -k & \text{при } \cos k > 0,3 \end{cases}$$

3. Найти среднее арифметическое значение элементов массива  $Z$ ;  $k = 1, 12$ .

4. Найти произведение элементов массива  $Y$ , удовлетворяющих условию  $-3,5 < x_i < 3,5$ .

5. Найти значения и порядковые номера наименьших элементов массивов  $X$  и  $Y$ .

6. Упорядочить массив  $X$  по возрастанию значений его элементов.

7. Вывести на печать массив  $X$ , массив  $Z$ , массив  $Y$ , среднее арифметическое элементов массива  $X$ , значения и порядковые номера наименьших элементов массивов  $X$  и  $Y$ , упорядоченный массив  $X$ .

*Исходные данные*

$i = 1, 12$

*Вариант 7*

1. Образовать два массива  $X$  и  $Y$ , состоящие из 15 целых чисел.

2. Образовать массив  $Z$  по правилу:

$$Z_j = Y_j \times s - 2 \times X_j,$$

$$s = \begin{cases} j & \text{при } \cos j \leq 0,3 \\ -j & \text{при } \cos j > 0,3 \end{cases}$$

3. Найти количество отрицательных элементов массива  $X$  и массива  $Y$ .

4. Найти сумму элементов массива  $Y$ , удовлетворяющих условию  $Y_i > 5$

5. Найти значение и порядковый номер наименьшего элемента массива  $Z$ .

6. Упорядочить массив  $Y$  по убыванию значений его элементов.

7. Вывести на печать массив  $X$ , массив  $Z$ , массив  $Y$ , среднее арифметическое элементов массива  $X$  и массива  $Y$ , сумму элементов массива  $Y$ , удовлетворяющих заданному условию, значение и порядковый номер наибольшего элемента массива  $Z$ , упорядоченный массив  $Y$ .

*Исходные данные*

$j = 1, 12$

*Вариант 8*

1. Образовать два массива  $X$  и  $Y$ , состоящие из 12 вещественных чисел.

2. Образовать массив  $Z$  по правилу:

$$Z_j = Y_j \times s - 2 \times X_j^2,$$

$$s = \begin{cases} j & \text{при } \cos j \leq 0,3 \\ -j & \text{при } \cos j > 0,3 \end{cases}$$

3. Найти среднее арифметическое значение положительных элементов массива  $X$  и массива  $Y$ .
4. Найти сумму элементов массива  $Y$ .
5. Найти значение и порядковый номер наименьшего элемента массива  $X$ .
6. Упорядочить массив  $Z$  по возрастанию значений его элементов.
7. Вывести на печать массив  $X$ , массив  $Z$ , массив  $Y$ , среднее арифметическое положительных элементов массива  $X$  и массива  $Y$ , сумму элементов массива  $Y$ , значение и порядковый номер наименьшего элемента массива  $X$ , упорядоченный массив  $Z$ .

*Исходные данные*

$$j = 1, 12$$

*Вариант 9*

1. Образовать два массива  $A$  и  $B$ , состоящие из 12 вещественных чисел.
2. Образовать массив  $C$  по правилу:  

$$C_i = A_i + B_i,$$
3. Определить количество отрицательных элементов массива  $A$  и массива  $B$ .
4. Найти произведение элементов массива  $C$ .
5. Для векторов  $A_i$  и  $B_i$  найти скалярное произведение  

$$S = \sum_{i=1}^n A_i \times B_i$$
6. Упорядочить массив  $C$  по убыванию значений его элементов.
7. Вывести на печать массив  $A$ , массив  $B$ , массив  $C$ , количество отрицательных элементов массива  $A$  и массива  $B$ , произведение элементов массива  $C$ , скалярное произведение векторов  $A_i$  и  $B_i$ , упорядоченный массив  $C$ .

*Исходные данные*

$$i = 1, 12$$

### *Вариант 10*

1. Образовать два массива  $A$ , состоящий из 12 вещественных чисел.
2. Образовать массив  $B$  по правилу:  
 $B_i = iA_i$ .
3. Определить количество элементов массива  $B$  по правилу  $0 \leq B_i \leq 10$ .
4. Найти произведение ненулевых элементов массива  $A$  и массива  $B$ .
5. Образовать массив  $C$ , заменив в массиве  $B$  положительные элементы 1, отрицательные  $-1$ .
6. Упорядочить массив  $A$  по убыванию значений его элементов.
7. Вывести на печать массив  $A$ , массив  $B$ , массив  $C$ , количество элементов массива  $B$  по правилу  $0 \leq B_i \leq 10$ ., произведение элементов массива  $A$  и массива  $B$ , упорядоченный массив  $A$ .

### *Исходные данные*

$i = 1, 12$

## **Лабораторная работа № 6**

### **ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ОБРАБОТКИ ДВУМЕРНЫХ МАССИВОВ**

**Цель работы:** ознакомиться с понятием двумерного массива; приемами программной реализации на языке программирования Python; произвести отладку и тестирование полученных программ.

#### **Задание:**

1. Изучить краткие теоретические сведения.
2. Разработать алгоритмы для решения задач (блок-схемы).
3. Записать алгоритмы на языке программирования Python.
4. Произвести отладку и тестирование программы.
5. Оформить отчет по лабораторной работе.

**Отчет о лабораторной работе** должен содержать следующие сведения:



1. Название и цель работы.
2. Формулировку задачи, схему алгоритма, программный код и результаты решения задачи.
3. Вывод по работе в целом.

## **Краткие теоретические сведения**

### ***Двумерные массивы***

#### ***1. Обработка и вывод вложенных списков***

Часто в задачах приходится хранить прямоугольные таблицы с данными. Такие таблицы называются матрицами или двумерными массивами. В языке программирования Питон таблицу можно представить в виде списка строк, каждый элемент которого является в свою очередь списком, например, чисел. Например, приведём программу, в которой создаётся числовая таблица из двух строк и трех столбцов, с которой производятся различные действия:

```
a = [[1, 2, 3], [4, 5, 6]]
print(a[0])
print(a[1])
b = a[0]
print(b)
print(a[0][2])
a[0][1] = 7
print(a)
print(b)
b[2] = 9
print(a[0])
print(b)
```

#### ***Результат***

```
[1, 2, 3]
[4, 5, 6]
[1, 2, 3]
3
[[1, 7, 3], [4, 5, 6]]
[1, 7, 3]
[1, 7, 9]
[1, 7, 9]
```

Здесь первая строка списка `a[0]` является списком из чисел [1, 2, 3]. То есть `a[0][0] == 1`, значение `a[0][1] == 2`, `a[0][2] == 3`, `a[1][0] == 4`, `a[1][1] == 5`, `a[1][2] == 6`.

Для обработки и вывода списка, как правило, используют два вложенных цикла. Первый цикл перебирает номер строки, второй цикл бежит по элементам внутри строки. Например, вывести двумерный числовой список на экран построчно, разделяя числа пробелами внутри одной строки, можно так:

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
for i in range(len(a)):
    for j in range(len(a[i])):
        print(a[i][j], end=' ')
    print()
```

*Результат*

```
1 2 3 4
5 6
7 8 9
```

Переменная цикла **for** в Python может перебирать не только диапазон, создаваемый с помощью функции **range()**, но и вообще перебирать любые элементы любой последовательности.

Последовательностями в Питоне являются списки, строки, а также некоторые другие объекты. Ввод двумерного массив, используя свойство цикла **for**:

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
for row in a:
    for elem in row:
        print(elem, end=' ')
    print()
```

*Результат*

```
1 2 3 4
5 6
7 8 9
```

Для вывода одной строки можно воспользоваться методом **join()**:

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
for row in a:
    print(' '.join([str(elem) for elem in row]))
```

*Результат*

1 2 3 4

5 6

7 8 9

Для подсчета суммы всех чисел в списке, используем два вложенных цикла

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
s = 0
for i in range(len(a)):
    for j in range(len(a[i])):
        s += a[i][j]
print(s)
```

*Результат*

45

То же самое с циклом не по индексу, а по значениям строк:

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
s = 0
for row in a:
    for elem in row:
        s += elem
print(s)
```

*Результат*

45

## 2. Создание вложенных списков

Пусть даны два числа: количество строк **n** и количество столбцов **m**. Необходимо создать список размером **n×m**, заполненный нулями.

**Первый способ:** сначала создадим список из **n** элементов (для начала просто из **n** нулей). Затем сделаем каждый элемент списка ссылкой на другой одномерный список из **m** элементов:

```
n = 3
m = 4
a = [0] * n
for i in range(n):
    a[i] = [0] * m
print(a)
```

*Результат*

```
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

Другой (но похожий) способ: создать пустой список, потом **n** раз добавить в него новый элемент, являющийся списком-строкой:

```
n = 3
m = 4
a = []
for i in range(n):
    a.append([0] * m)
print(a)
```

*Результат*

```
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

Но еще проще воспользоваться генератором: создать список из **n** элементов, каждый из которых будет списком, состоящих из **m** нулей:

```
n = 3
m = 4
```

```
a = [[0] * m for i in range(n)]
print(a)
```

*Результат*

```
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

В этом случае каждый элемент создается независимо от остальных (заново конструируется список `[0] * m` для заполнения очередного элемента списка), а не копируются ссылки на один и тот же список.

### *3. Ввод двумерного массива*

Пусть программа получает на вход двумерный массив в виде  $n$  строк, каждая из которых содержит  $m$  чисел, разделенных пробелами.

```
# в первой строке ввода идёт количество строк массива
n = int(input())
a = []
for i in range(n):
    a.append([int(j) for j in input().split()])
print(a)
```

*Результат*

```
5
-4
-8
4
3
6
[[-4], [-8], [4], [3], [6]]
```

Или, без использования сложных вложенных вызовов функций:

```
# в первой строке ввода идёт количество строк массива
n = int(input())
a = []
for i in range(n):
    row = input().split()
```

```

for i in range(len(row)):
    row[i] = int(row[i])
a.append(row)
print(a)

```

*Результат*

```

5
1
2
3
4
5
[[1], [2], [3], [4], [5]]

```

Можно сделать то же самое и при помощи генератора:

```

# в первой строке ввода идёт количество строк массива
n = int(input())
a = [[int(j) for j in input().split()] for i in range(n)]
print(a)

```

*Результат*

```

5
6
8
4
-8
-1
[[6], [8], [4], [-8], [-1]]

```

#### *4. Пример обработки двумерного массива*

Пусть дан квадратный массив из **n** строк и **n** столбцов. Необходимо элементам, находящимся на главной диагонали, проходящей из левого верхнего угла в правый нижний (то есть тем элементам  $a[i][j]$ , для которых  $i==j$ ) присвоить значение 1, элементам, находящимся выше главной диагонали – значение 0, элементам, находящимся ниже главной диагонали – значение 2. То есть необходимо получить такой массив (пример для  $n==4$ ):

```
1 0 0 0
2 1 0 0
2 2 1 0
2 2 2 1
```

Рассмотрим несколько способов решения этой задачи. Элементы, которые лежат выше главной диагонали – это элементы  $a[i][j]$ , для которых  $i < j$ , а для элементов ниже главной диагонали  $i > j$ . Таким образом, мы можем сравнивать значения  $i$  и  $j$  и по ним определять значение  $A[i][j]$ . Получаем следующий алгоритм:

```
n = 4
a = [[0] * n for i in range(n)]
for i in range(n):
    for j in range(n):
        if i < j:
            a[i][j] = 0
        elif i > j:
            a[i][j] = 2
        else:
            a[i][j] = 1
for row in a:
    print(' '.join([str(elem) for elem in row]))
```

*Результат*

```
1 0 0 0
2 1 0 0
2 2 1 0
2 2 2 1
```

Если усложнить алгоритм, то можно обойтись вообще без условных инструкций.

Сначала заполним главную диагональ, для чего нам понадобится один цикл:

```
for i in range(n):
    a[i][i] = 1
```

Затем заполним значением 0 все элементы выше главной диагонали, для чего нам понадобится в каждой из строк с номером  $i$

присвоить значение элементам  $a[i][j]$  для  $j=i+1, \dots, n-1$ . Здесь нам понадобятся вложенные циклы:

```
for i in range(n):
    for j in range(i + 1, n):
        a[i][j] = 0
```

Аналогично присваиваем значение 2 элементам  $a[i][j]$  для  $j=0, \dots, i-1$ :

```
for i in range(n):
    for j in range(0, i):
        a[i][j] = 2
```

Получаем

```
n = 4
a = [[0] * n for i in range(n)]
for i in range(n):
    for j in range(n):
        if i < j:
            a[i][j] = 0
        elif i > j:
            a[i][j] = 2
        else:
            a[i][j] = 1
for row in a:
    print(' '.join([str(elem) for elem in row]))
```

*Результат*

```
1 0 0 0
2 1 0 0
2 2 1 0
2 2 2 1
```

Можно также внешние циклы объединить в один и получить еще одно, более компактное решение:

```
n = 4
a = [[0] * n for i in range(n)]
for i in range(n):
```



```

    for j in range(0, i):
        a[i][j] = 2
    a[i][i] = 1
    for j in range(i + 1, n):
        a[i][j] = 0
for row in a:
    print(' '.join([str(elem) for elem in row]))

```

*Результат*

```

1 0 0 0
2 1 0 0
2 2 1 0
2 2 2 1

```

А вот такое решение использует операцию повторения списков для построения очередной строки списка.  $i$ -я строка списка состоит из  $i$  чисел 2, затем идет одно число 1, затем идет  $n-i-1$  число 0:

```

n = 4
a = [0] * n
for i in range(n):
    a[i] = [2] * i + [1] + [0] * (n - i - 1)
for row in a:
    print(' '.join([str(elem) for elem in row]))

```

*Результат*

```

1 0 0 0
2 1 0 0
2 2 1 0
2 2 2 1

```

А можно заменить цикл на генератор:

```

n = 4
a = [0] * n
a = [[2] * i + [1] + [0] * (n - i - 1) for i in range(n)]
for row in a:
    print(' '.join([str(elem) for elem in row]))

```

### *Результат*

```
1 0 0 0
2 1 0 0
2 2 1 0
2 2 2 1
```

### *5. Вложенные генераторы двумерных массивов*

Для создания двумерных массивов можно использовать вложенные генераторы, разместив генератор списка, являющегося строкой, внутри генератора всех строк. Напомним, что сделать список из  $n$  строк и  $m$  столбцов можно при помощи генератора, создающего список из  $n$  элементов, каждый элемент которого является списком из  $m$  нулей:

```
[[0] * m for i in range(n)]
```

Но при этом внутренний список также можно создать при помощи, например, такого генератора: `[0 for j in range(m)]`. Вложив один генератор в другой, получим вложенные генераторы:

```
[[0 for j in range(m)] for i in range(n)]
```

Но если число 0 заменить на некоторое выражение, зависящее от  $i$  (номер строки) и  $j$  (номер столбца), то можно получить список, заполненный по некоторой формуле.

Например, пусть нужно задать следующий массив (для удобства добавлены дополнительные пробелы между элементами):

```
0 0 0 0 0 0
0 1 2 3 4 5
0 2 4 6 8 10
0 3 6 9 12 15
0 4 8 12 16 20
```

В этом массиве  $n = 5$  строк,  $m = 6$  столбцов, и элемент в строке  $i$  и столбце  $j$  вычисляется по формуле:  $a[i][j] = i * j$ .

Для создания такого массива можно использовать генератор:

```
[[i * j for j in range(m)] for i in range(n)]
```

### Задача

1. Сформировать матрицу  $A(12,12)$  случайными целыми числами в диапазоне  $[-100; 100]$ .
2. Вычислить произведение элементов каждого столбца матрицы  $A$ , образовав массив  $B$ .
3. Переписать в массив  $C$  все положительные элементы четных строк матрицы  $A$ .
4. Вывести на печать элементы строки, в которой находится максимальный элемент матрицы  $A$ .
5. Упорядочить по возрастанию элементы каждой нечетной строки матрицы  $A$ .
6. Вывести на печать матрицу  $A$ , массивы  $B$ ,  $C$ , матрицу  $A$  после сортировки.

**Постановка задачи:** Необходимо сформировать матрицу, на основе неё создать два массива. Выполнить действия с матрицей и массивами.

**Метод решения:** Используем переменную  $A$  для формирования матрицы. Переменные  $B$  и  $C$  используются для массивов.

Для формирования массива  $B$  вводим переменную  $pr$ , которая будет использована для хранения значения произведения, и переменную  $k$ , которая будет обозначать индекс элементов массива  $B$ . В цикле находим произведение 1 столбца матрицы  $A$ , вносим это значение в первый элемент массива  $B$ , переменной  $pr$  присваивается значение 1. Цикл повторяется для всех столбцов.

Для формирования массива  $C$ , вводим переменную  $count$ , которая будет отсчитывать количество элементов в массиве и переменную  $q$ , обозначающую индекс элемента. В цикле проверяем значение номера строки на чётность, если это чётная строка, то проверяется знак каждого элемента в строке. Если элемент положительный, то в массив  $C$  заносится это значение, а переменная  $count$  увеличивается на 1.

Введём переменную  $max$ , которая будет хранить значение наибольшего элемента, и переменную  $imax$ , которая будет хранить номер строки, в которой находится максимальный элемент. Путём сравнения элементов с переменной  $max$  находим максимальный элемент. В цикле, проходящем только по столбцам матрицы, выводим на печать элементы строки  $imax$ .

Для сортировки элементов строк матрицы  $A$ , используем цикл. Сначала проверяется нечетность строки, после этого методом пузырька при помощи переменной  $m$  элементы в строке сортируются по возрастанию (рис. 19).



## Текст программы на Python

```
# -*- coding: utf-8 -*-
from random import randint
n = 12
A = [[0] * n for i in range(n)]
print("Исходный массив A")
for i in range(n):
    for j in range(n):
        A[i][j]=randint(-100, 100)
for i in range(n):
    for j in range(n):
        print("%4d" % A[i][j], end = ' ')
    print()
print()

pr=1
k=-1
B=[0]*100
for j in range(n):
    for i in range(n):
        pr=pr*A[i][j]
        k=k+1
        B[k] = pr
    pr=1
print("Массив B")
for k in range(n):
    print('B['+k+1,']=','%4d' % B[k], end = ' ')
    print()
print()

count=0
q=-1
C=[0]*100
for i in range(n):
    if i%2 != 0:
        for j in range(n):
            if A[i][j] > 0:
                q=q+1
                C[q] = A[i][j]
```

```

        count=count+1
print('Массив C')
for q in range(count):
    print('C['+q+1,']=','%4d'%C[q])
print()
max = -101
imax=-1
for i in range(n):
    for j in range(n):
        if A[i][j] > max:
            max = A[i][j]
            imax=i

print('Строка «', imax+1, '», содержащая максимальный эле-
мент', max)
for j in range(n):
    print("%4d" % A[imax][j], end = ' ')
print()

print('Отсортированный массив A')
for i in range(n):
    if i%2!=1:
        for h in range(n-1):
            for j in range(n-1):
                if A[i][j]>A[i][j+1]:
                    m=A[i][j]
                    A[i][j]=A[i][j+1]
                    A[i][j+1]=m

for i in range(n):
    for j in range(n):
        print("%4d" % A[i][j], end = ' ')
print()

```

## Результат

Исходный массив А

|     |     |     |     |     |     |     |     |     |     |      |     |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|-----|
| 58  | -3  | -48 | -4  | -85 | 96  | -41 | 28  | 77  | 3   | 41   | -56 |
| 7   | -3  | -20 | 65  | 81  | -95 | -28 | 1   | 88  | -48 | -90  | 25  |
| 37  | 13  | -95 | 57  | -72 | -43 | 99  | -58 | -25 | 26  | 99   | 74  |
| 57  | 11  | 80  | 35  | -80 | 42  | 21  | 89  | -90 | 94  | -45  | -49 |
| 76  | -15 | -18 | 69  | -87 | -3  | 18  | 92  | -63 | 71  | -84  | -34 |
| 63  | 63  | -85 | 67  | -37 | 37  | 40  | -10 | 90  | 21  | -53  | -22 |
| 29  | -61 | 88  | -18 | 32  | 33  | 71  | -97 | -65 | -9  | -100 | 3   |
| 35  | -15 | 54  | -19 | -14 | -72 | -30 | -36 | -90 | 53  | 16   | -43 |
| 19  | 91  | 56  | 70  | 98  | 20  | -98 | 80  | -13 | -45 | 21   | -24 |
| 64  | -24 | -10 | -41 | -1  | 99  | 78  | 51  | 27  | 49  | 30   | 66  |
| 61  | -58 | 75  | -5  | -20 | -20 | 75  | 57  | 51  | -11 | 14   | 12  |
| -41 | -96 | -8  | 61  | 18  | -32 | -23 | 22  | -38 | -91 | -40  | -54 |

Массив В

```
B[ 1 ] = -12655230623633364480
B[ 2 ] = 13532664348403200
B[ 3 ] = -17823457935360000000
B[ 4 ] = -717873602051970000
B[ 5 ] = 2017687926030336000
B[ 6 ] = 5504617717530624000
B[ 7 ] = -48263437980838080000
B[ 8 ] = 2375722952347852800
B[ 9 ] = -343997851082886000000
B[ 10 ] = -552461969647748160
B[ 11 ] = 41312149224192000000
B[ 12 ] = -502779227927961600
```

Массив С

```
C[ 1 ] = 51
C[ 2 ] = 69
C[ 3 ] = 92
C[ 4 ] = 42
C[ 5 ] = 78
C[ 6 ] = 38
C[ 7 ] = 53
C[ 8 ] = 8
C[ 9 ] = 71
C[ 10 ] = 63
C[ 11 ] = 68
C[ 12 ] = 43
C[ 13 ] = 35
C[ 14 ] = 1
C[ 15 ] = 11
C[ 16 ] = 81
C[ 17 ] = 75
C[ 18 ] = 90
C[ 19 ] = 2
C[ 20 ] = 21
C[ 21 ] = 63
C[ 22 ] = 3
C[ 23 ] = 1
C[ 24 ] = 37
C[ 25 ] = 12
C[ 26 ] = 58
C[ 27 ] = 31
C[ 28 ] = 23
C[ 29 ] = 3
C[ 30 ] = 21
C[ 31 ] = 65
C[ 32 ] = 74
```

Строка « 3 », содержащая максимальный элемент 99

|    |    |    |     |    |     |    |    |     |     |     |    |
|----|----|----|-----|----|-----|----|----|-----|-----|-----|----|
| 84 | 37 | 41 | -75 | 84 | -43 | 37 | 99 | -17 | -52 | -87 | 41 |
|----|----|----|-----|----|-----|----|----|-----|-----|-----|----|

Отсортированный массив A

|      |     |     |     |     |     |     |     |     |     |     |     |
|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| -31  | -9  | -5  | 25  | 27  | 36  | 44  | 49  | 60  | 73  | 88  | 90  |
| -25  | 51  | -47 | 69  | 92  | 42  | 78  | 38  | -52 | 53  | -40 | -25 |
| -87  | -75 | -52 | -43 | -17 | 37  | 37  | 41  | 41  | 84  | 84  | 99  |
| 8    | -92 | -7  | -10 | 71  | 63  | 68  | 43  | -35 | -91 | -36 | -73 |
| -93  | -90 | -90 | -70 | -30 | 1   | 7   | 30  | 41  | 58  | 72  | 95  |
| 35   | 1   | -75 | -75 | 11  | 81  | 75  | 90  | -73 | 2   | 21  | 63  |
| -86  | -69 | -54 | -52 | -40 | -12 | -3  | 4   | 22  | 31  | 65  | 97  |
| -14  | 3   | -77 | -88 | -26 | -42 | 1   | -46 | -28 | 37  | -25 | -70 |
| -100 | -98 | -76 | -65 | -64 | -62 | -42 | -17 | 17  | 39  | 68  | 71  |
| 12   | -20 | -36 | -50 | 58  | -76 | -10 | 31  | -85 | -26 | -33 | 23  |
| -87  | -83 | -44 | -34 | -23 | -21 | -11 | -5  | 19  | 31  | 71  | 92  |
| 3    | 21  | -54 | 65  | 74  | -43 | -79 | -69 | -8  | -87 | -25 | -54 |

## Задачи для самостоятельной работы

### Вариант № 1

1. Сформировать матрицу  $A(10,10)$  целыми случайными числами в диапазоне  $[-48, 63]$ .
2. Переписать в массив  $B$  все положительные элементы матрицы  $A$ , расположенные выше главной диагонали.
3. Найти в каждой строке матрицы  $A$  наибольший элемент, образовав массив  $C$ .
4. Найти сумму  $Sum$  и количество  $Kol$  элементов кратных пяти среди элементов матрицы  $A$ .
5. Упорядочить по убыванию элементы каждой четной строки матрицы  $A$ .
6. Вывести на печать: исходную матрицу  $A$ , массивы  $B$  и  $C$ ,  $Sum$  и  $Kol$ , матрицу  $A$  после сортировки.

### Вариант № 2

1. Сформировать матрицу  $A(10,10)$  вещественными случайными числами в диапазоне  $[-80,100]$ .
2. Переписать в массив  $B$  все положительные элементы матрицы  $A$ , расположенные на главной диагонали.
3. Найти в каждой строке матрицы  $A$  наименьший элемент, образовав массив  $C$ .
4. Найти произведение  $P$  количество  $K$  четных элементов матрицы  $A$ .



5. Упорядочить по убыванию элементы каждой нечетной строки матрицы  $A$ .

6. Вывести на печать: исходную матрицу  $A$ , массивы  $B$  и  $C$ ,  $P$  и  $K$ , матрицу  $A$  после сортировки.

### *Вариант № 3*

1. Сформировать матрицу  $A(14,14)$  случайными вещественными числами в диапазоне  $[-100, 100]$ .

2. Переписать в массив  $B$  все положительные элементы матрицы  $A$ , расположенные на главной диагонали.

3. Найти в каждой строке матрицы  $A$  наименьший элемент, образовав массив  $C$ .

4. Найти произведение  $P$ , количество  $K$  четных элементов матрицы  $A$ .

5. Упорядочить по убыванию элементы каждой нечетной строки матрицы  $A$ .

6. Вывести на печать: исходную матрицу  $A$ , массивы  $B$  и  $C$ ,  $P$  и  $K$ , матрицу  $A$  после сортировки.

### *Вариант № 4*

1. Сформировать матрицу  $B(12,12)$  случайными целыми числами в диапазоне  $[-50,50]$ .

2. Найти в каждом столбце наименьшее значение среди положительных элементов матрицы  $B$ , образовав массив  $A$ .

3. Образовать матрицу  $C$ , переписав элементы матрицы  $B$  каждой строки в обратном порядке.

4. Найти сумму  $C1$  отрицательных элементов нечетных строк матрицы  $B$ .

5. Упорядочить по убыванию элементы каждого нечетного столбца матрицы  $C$ .

6. Вывести на печать: исходную матрицу  $B$ , массивы  $A$  и  $C$ ,  $C1$ , матрицу  $C$  после сортировки.

### *Вариант № 5*

1. Сформировать матрицу  $A(10,10)$  случайными вещественными числами в диапазоне  $[-100,85]$ .

2. Вычислить сумму положительных элементов побочной диагонали матрицы  $A$ .

3. Переписать в массив  $B$  все элементы матрицы  $A$ , расположенные ниже побочной диагонали.

4. Найти кол-во положительных элементов каждой строки матрицы  $A$ , образовав массив  $C$ .
5. Упорядочить массив  $B$  по убыванию, образовав массив  $D$ .
6. Вывести массивы  $A, B, C, D$ , сумму положительных элементов побочной диагонали матрицы  $A$ .

*Вариант № 6*

1. Сформировать матрицу  $A(16,16)$  случайными вещественными числами в диапазоне  $[-120,70]$ .
2. Найти для каждого столбца матрицы  $B$  число элементов кратных пяти и наименьший из полученных результатов, вывести полученные данные.
3. Переписать в массив  $B$  все элементы матрицы  $A$ , расположенные выше главной диагонали.
4. Найти в каждой строке матрицы  $A$  наибольший элемент и поменять его местами с элементом главной диагонали, образовав массив  $C$ .
5. Упорядочить по убыванию элементы каждой строки матрицы  $A$ , образовав массив  $M$ .
6. Вывести на печать массивы  $A, B, C, M$ .

*Вариант № 7*

1. Сформировать матрицу  $A(13,13)$  случайными целыми числами в диапазоне  $[-10,120]$ .
2. Вычислить произведение элементов главной диагонали матрицы  $A$ .
3. Переписать в массив  $B$  все элементы матрицы  $A$ , расположенные выше главной диагонали.
4. Вывести на печать строку, в которой расположен минимальный элемент матрицы  $A$ .
5. Найти максимальные элементы каждого столбца матрицы  $A$ , образовав массив  $C$ .
6. Вывести на печать массивы  $A, B, C$ , произведение элементов матрицы  $A$ .

*Вариант № 8*

1. Сформировать матрицу  $A(11,11)$  случайными целыми числами в диапазоне  $[-100,100]$ .

2. Вычислить и запомнить суммы всех положительных элементов каждого столбца матрицы  $A$ , образовав массив  $B$ .

3. Найти максимальный и минимальный элементы каждой строки матрицы  $A$  и поменять их местами, образовав матрицу  $C$ .

4. Вычислить среднее арифметическое элементов матрицы  $A$ .

5. Переписать в массив  $M$  все элементы матрицы  $C$ , не удовлетворяющие условию  $-1 \leq C(i,j) \leq 1$ .

6. Вывести на печать массивы  $A$ ,  $B$ ,  $C$ ,  $M$ , среднее арифметическое элементов матрицы  $A$ .

### *Вариант № 9*

1. Сформировать матрицу  $A(15,15)$  случайными вещественными числами в диапазоне  $[-70,45]$ .

2. Вычислить и запомнить произведение и число отрицательных элементов каждого столбца матрицы  $A$ .

3. Найти в каждой строке матрицы  $A$  минимальный элемент и поместить его на место первого, образовав матрицу  $B$ .

4. Вычислить среднее арифметическое каждой строки матрицы  $A$ , образовав массив  $C$ .

5. Найти в массиве  $C$  элементы, удовлетворяющие условию  $C(i)=0$  и вывести на печать их порядковые номера.

6. Вывести на печать массивы  $A$ ,  $B$ ,  $C$ .

### *Вариант № 10*

1. Сформировать матрицу  $A(12,12)$  случайными целыми числами в диапазоне  $[-100,100]$ .

2. Найти среднее арифметическое всех положительных элементов каждой строки матрицы  $A$ , образовав массив  $B$ .

3. Переписать в массив  $C$  все элементы матрицы  $A$ , удовлетворяющие условию  $A(I,J) > 1$ .

4. Найти минимальный и максимальный элементы матрицы  $A$  и поменять их местами, образовав матрицу  $D$ .

5. Найти произведение всех элементов матрицы  $A$ , исключая нулевые.

6. Вывести на печать массивы  $A$ ,  $B$ ,  $C$ ,  $D$ , номера строк и столбцов минимального и максимального элементов матрицы  $A$ , значение произведения элементов матрицы  $A$ .

### *Вариант № 11*

1. Сформировать матрицу  $A(13,13)$  случайными целыми числами в диапазоне  $[-50,85]$ .
2. Переписать в массив  $B$  все элементы матрицы  $A$ , расположенные ниже побочной диагонали.
3. Упорядочить элементы каждой строки матрицы  $A$  по возрастанию.
4. Найти сумму элементов матрицы  $A$  кратных числу  $K$ , их количество и результаты вывести на печать.
5. Переписать в массив  $C$  все элементы матрицы  $A$ , удовлетворяющие условию  $A(I,J)>9$ .
6. Вывести на печать массивы  $A, B, C$ .

### *Вариант № 12*

1. Сформировать матрицу  $A(12,12)$  случайными вещественными числами в диапазоне  $[-45,100]$ .
2. Найти среднее арифметическое отрицательных элементов каждого столбца матрицы  $A$ , образовав массив  $B$ .
3. Вычислить сумму положительных элементов главной диагонали матрицы  $A$ .
4. Переписать в массив  $C$  все элементы матрицы  $A$ , расположенные выше главной диагонали.
5. Найти наименьший элемент каждой строки матрицы  $A$ , образовав массив  $D$ .
6. Вывести на печать массивы  $A, B, C, D$ .

### *Вариант № 13*

1. Сформировать матрицу  $A(14,14)$  случайными целыми числами в диапазоне  $[-100,100]$ .
2. Вычислить произведение элементов главной диагонали матрицы  $A$ .
3. Вывести на печать элементы столбца матрицы, в котором расположено максимальное значение.
4. Найти количество элементов матрицы  $A$ , удовлетворяющих условию:  $-100 \leq A(I,J) \leq 100$  и образовать из этих элементов массив  $B$ .
5. Отсортировать элементы каждой строки матрицы  $A$  по возрастанию.
6. Вывести на печать массивы  $A, B$  и матрицу  $A$  после сортировки.

### *Вариант № 14*

1. Сформировать матрицу  $A(12,12)$  случайными вещественными числами в диапазоне  $[-68,90]$ .
2. Образовать матрицу  $B$  из нечетных строк матрицы  $A$ .
3. Найти количество и сумму тех элементов матрицы  $A$ , которые делятся на число 5 и на 7. Результаты вывести на печать.
4. Переписать в массив  $C$  все элементы матрицы  $A$ , значение которых больше числа  $K$ .
5. Упорядочить элементы каждого столбца матрицы  $A$  в порядке возрастания.
6. Вывести на печать массивы  $A$ ,  $B$ ,  $C$  и матрицу  $A$  после сортировки.

### *Вариант № 15*

1. Сформировать матрицу  $A(18,18)$  случайными целыми числами в диапазоне  $[-100,100]$ .
2. Найти минимальные элементы каждой строки матрицы  $A$ , образовав массив  $C$ .
3. Переписать в массив  $B$  все элементы матрицы  $A$ , расположенные ниже побочной диагонали.
4. Найти сумму  $C1$  из элементов главной диагонали матрицы  $A$ .
5. Упорядочить массив  $B$  по убыванию, образовав массив  $M$ .
6. Вывести на печать: исходную матрицу  $A$ , массивы  $B$ ,  $C$ ,  $M$  и значение  $C1$ .

### *Вариант № 16*

1. Сформировать матрицу  $A(13,13)$  случайными вещественными числами в диапазоне  $[-75,80]$ .
2. Переписать в массив  $B$  все элементы матрицы  $A$ , расположенные выше побочной диагонали.
3. Вычислить сумму  $C1$  отрицательных элементов главной диагонали матрицы  $A$ .
4. Найти в каждой строке матрицы  $A$  наибольший элемент и поменять его местами с элементом главной диагонали, образовав матрицу  $C$ .
5. Упорядочить по убыванию элементы каждого четного столбца матрицы  $A$ .
6. Вывести на печать: исходную матрицу  $A$ , массив  $B$ , матрицу  $C$ , матрицу  $A$  после сортировки и значение  $C1$ .

*Вариант № 17*

1. Сформировать матрицу  $A(13,13)$  случайными целыми числами в диапазоне  $[-10,150]$ .
2. Вычислить произведение элементов каждой строки матрицы  $A$ , образовав массив  $B$ .
3. Переписать в массив  $C$  все отрицательные элементы нечетных строк матрицы  $A$ .
4. Вывести на печать элементы строки, в которой находится минимальный элемент матрицы  $A$ .
5. Упорядочить по возрастанию элементы каждой четной строки матрицы  $A$ .
6. Вывести на печать: исходную матрицу  $A$ , массивы  $B$ ,  $C$ , матрицу  $A$  после сортировки.

*Вариант № 18*

1. Сформировать матрицу  $A(10,10)$  случайными вещественными числами в диапазоне  $[-50,50]$ .
2. Образовать матрицу  $B$ , исключая первую и последнюю строки матрицы  $A$ .
3. Найти среди сумм строк матрицы  $A$  наибольшее значение  $C1$ .
4. Переписать в массив  $C$  все элементы нечетных строк матрицы  $A$ .
5. Упорядочить по убыванию элементы каждого столбца матрицы  $A$ .
6. Вывести на печать: исходную матрицу  $A$ , матрицу  $B$ , массив  $C$ ,  $C1$ , матрицу  $A$  после сортировки.

*Вариант № 19*

1. Сформировать матрицу  $A(17,17)$  случайными целыми числами в диапазоне  $[-50,50]$ .
2. Вычислить сумму положительных элементов побочной диагонали матрицы  $A$ .
3. Переписать в массив  $B$  все элементы матрицы  $A$ , расположенные ниже побочной диагонали.
4. Найти кол-во положительных элементов каждой строки матрицы  $A$ , образовав массив  $C$ .
5. Упорядочить массив  $B$  по убыванию, образовав массив  $D$ .
6. Вывести массивы  $A$ ,  $B$ ,  $C$ ,  $D$ , сумму положительных элементов побочной диагонали матрицы  $A$ .

### *Вариант № 20*

1. Сформировать матрицу  $A(19,19)$  случайными вещественными числами в диапазоне  $[-100,100]$ .
2. Образовать одномерный массив  $B$  из отрицательных элементов четных строк матрицы  $A$ .
3. Вычислить сумму  $C1$  элементов главной диагонали матрицы  $A$ .
4. Записать в массив  $C$  первые 10 положительных элементов матрицы  $A$ .
5. Упорядочить элементы нечетных столбцов матрицы  $A$  в порядке убывания.
6. Вывести на печать: исходную матрицу  $A$ , массив  $B$ , массив  $C$ ,  $C1$ , матрицу  $A$  после сортировки.

### **Список использованных источников**

1. Прохоренок, Н.А. Python 3. Самое необходимое / Н.А. Прохоренок, В.А. Дронов. – 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2019. – 608 с.
2. Федоров, Д.Ю. Программирование на языке высокого уровня Python : учебное пособие для прикладного бакалавриата / Д.Ю. Федоров. – М. : Изд-во Юрайт, 2017. – 126 с.

## Оглавление

|  |     |
|--|-----|
| Введение .....   | 3   |
| Лабораторная работа № 1. Знакомство с интегрированной средой разработки IDLE Python. Работа с библиотекой Math ..... | 4   |
| Лабораторная работа № 2. Программная реализация алгоритмов линейной структуры .....                                  | 17  |
| Лабораторная работа № 3. Программная реализация алгоритмов разветвленной структуры .....                             | 34  |
| Лабораторная работа № 4. Программная реализация алгоритмов циклической структуры .....                               | 44  |
| Лабораторная работа № 5. Программная реализация обработки одномерных массивов .....                                  | 58  |
| Лабораторная работа № 6. Программная реализация обработки двумерных массивов .....                                   | 80  |
| Список использованных источников .....   | 103 |