

Министерство образования науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
Тульский государственный университет

Институт прикладной математики и компьютерных наук
Кафедра
Информационная безопасность

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ
К ПРАКТИЧЕСКИМ РАБОТАМ
№ 1-14**

по дисциплине

Современные IT-технологии

Форма обучения: очная, заочная, очно-заочная

Разработчик:
к.т.н., доц. Баранова Е.М.

Тула, 2022 г.

Практическая работа №1

Установка среды программирования Python

1.1. Цель лабораторной работы

Изучение среды программирования языка Python

1.2 Теоретические положения

Установка среды программирования

Можно сказать, что язык программирования Python практически ничем не ограничен, поэтому может быть использован для создания крупнейших проектов, в основу которых положены принципы обработки информации, хранящейся в файлах, созданных, например, в электронных таблицах или системах управления базами данных (СУБД).

Сегодня Python используется, например, разработчиками Google и Yandex. Основные характеристики языка Python – это простота и универсальность.

Среда программирования на языке Python называется IDLE (Integrated Development and Learning Environment). IDLE представляет собой интегрированную среду разработки и обучения на языке Python (программу для написания проектов на языке Python), созданную с помощью библиотеки Tkinter. Программа является бесплатной и ее можно скачать по ссылке: <https://python.org/downloads/windows/>.

Для установки программы под операционную систему Windows необходимо выбрать Latest Python 3 Release – Python 3.4.1 (версия может быть и другая). Затем необходимо выбрать download page. Откроется список файлов, которые можно загрузить.

Если операционная система Windows 32-х битная, то потребуется файл Windows x86 MSI installer. Если операционная система Windows 64-х битная, то потребуется файл Windows x86-64 MSI installer. После этого открывается загрузившийся файл. Файл подписан Python Software Foundation. Далее следует нажать на кнопку Выполнить и следовать инструкции по установке. Компоненты программы можно оставить те, которые предложены по умолчанию.

Для установки программы под операционную систему Linux системы (Ubuntu, Linux Mint и другие) необходимо проверить, установлен ли Python по умолчанию. Для этого в консоли (вызов консоли часто ctrl+alt+t) необходимо набрать команду: python3. Если Python установлен, то консоль выдаст сообщение о том, какая версия программы установлена.

Если Python не установлен, то в консоли необходимо набрать команду: `sudo apt-get install python3`, после чего будет установлен пакет Python 3.

Также установить пакет Python 3 можно через Mintinstaller/Synaptic/центр приложений Ubuntu.

В Python для Linux нет предустановленной среды IDLE. Для установки этой среды потребуется установить пакет Idle3 или Python3-idle.

Установка среды IDLE под Linux необязательна, так как программы можно писать в текстовом редакторе и запускать их через консоль.

Запуск среды

После запуска среды IDLE, откроется окно для написания текстов программ на языке Python (рис.1).

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> |
```

Рисунок 1 – Окно для написания текстов программ на языке Python (интерактивный режим)

Первый запуск среды IDLE позволяет писать программы в интерактивном режиме, без сохранения файла.

Для перехода в основной режим необходимо выбрать меню File/New File или использовать для этого сочетание клавиш Ctrl + N.

Синтаксис языка Python

При написании программ на языке Python следует пользоваться правилами:

1. Конец строки является концом инструкции (точка с запятой не требуется).

2. Вложенные инструкции объединяются в блоки по величине отступов. Отступ может быть любым, главное, чтобы в пределах одного вложенного блока отступ был одинаков. Целесообразно использовать несколько пробелов или знак табуляции.

3. Вложенные инструкции в Python записываются в соответствии с одним и тем же шаблоном, когда основная инструкция завершается двоеточием, вслед за которым располагается вложенный блок кода, обычно с отступом под строкой основной инструкции, например,

```
if x > y:
    print(x).
```

В примере *if x > y:* – основная инструкция, *print(x)* – вложенный блок инструкций.

4. Можно записать несколько инструкций в одной строке, разделяя их точкой с запятой, например,

```
a = 1; b = 2; print(a, b),
```

однако этот случай подходит для небольших программ, если структура программы велика, это может привести не к читаемости кода.

5. Допустимо записывать одну инструкцию в нескольких строках. Достаточно ее заключить в пару круглых, квадратных или фигурных скобок, например,

```
if (a = 1 and b = 2 and
    c = 3 and d = 4):
    print('d').
```

6. Тело составной инструкции может располагаться в той же строке, что и тело основной, если тело составной инструкции не содержит составных инструкций, например,

if x > y: print(x).

В примере `if x > y` – тело составной инструкции, `print(x)` – тело основной инструкции, `:` - разделение инструкций, записанных в одной строке.

Простейшие операторы и программы языка Python

- `input` – оператор ввода;
- `print` – оператор вывода;
- инструкция `if-elif-else` – оператор условия, причем сначала записывается часть `if` с условным выражением, далее могут следовать одна или более необязательных частей `elif`, и, наконец, необязательная часть `else` (часть `elif` может заменять конструкции `switch – case of` в других языках программирования).
- цикл `for` проходится по любому объекту (например, массиву, строке или списку), и во время каждого прохода выполняет тело цикла;
- оператор `break` досрочно прерывает цикл;
- оператор `continue` начинает следующий проход цикла, минуя оставшееся тело цикла.

1.3. Оборудование и программное обеспечение

1. Персональная ЭВМ.
2. Операционная система Windows/Linux.
3. Среда программирования IDLE или другая среда, содержащая язык программирования Python.

1.4. Задание на работу

1. Изучить теоретические положения лабораторной работы.
2. Скачать и установить среду IDLE или другую среду, содержащую язык программирования Python.
3. Изучить команды меню главного окна программы, синтаксис и простейшие операторы языка Python.
4. Научиться сохранять набранные тексты программ в файлы.
5. Научиться загружать сохраненные файлы с текстами программ в окно редактирования среды IDLE (другой среды).
9. Ответить на контрольные вопросы.
10. Оформить отчет.

1.5. Контрольные вопросы

1. Опишите синтаксис языка Python.
2. Опишите простейшие операторы языка Python.
3. Опишите структуру ветвления (выбора) языка Python.
4. Что позволяет осуществлять функция `len` в языке Python?
5. Что такое объект-контейнер?
6. Что означает запись `end= ''` на языке Python?

1.6. Правила оформления отчета

Отчет должен содержать:

1. Номер, название и цель лабораторной работы.
2. Описание процесса выполнения примеров 1-3 (коды и результаты работы программ).

3. Описание процесса изменения примеров 1-3 в соответствии с заданием (коды и результаты работы программ).
4. Вывод по работе.

Практическая работа №2

Простейшие программы на языке Python

2.1. Цель лабораторной работы

Получение навыков по составлению и отладке простейших схем и программ на языке Python

2.2. Теоретические положения

Схема программы – ориентированный граф, предписывающий однозначное выполнение действий при работе компьютерной программы. Составление схем программ осуществляется в соответствии с ГОСТ 19.701-90 «Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения».

Часто схему программы называют алгоритмом, однако алгоритм – более широкое понятие, так как способы задания алгоритмов – различны, например, словесный, табличный, графический и другие способы. Схема программы – это графическое изображение алгоритма, на базе которого работает программа.

Указанный ГОСТ регламентирует блоки и правила их оформления при графическом изображении схем программ. Выбор блока зависит от действий, заложенных в программу. Каждое конечное действие программы в схеме описывается одним блоком.

Программа – запись действий, которые должны выполняться компьютером, составленные на «понятном» для него языке.

Простейшая программа выполняет последовательность действий, не имея ветвлений и циклов. Такая базовая структура программы называется следование или последовательность (рисунок 2).

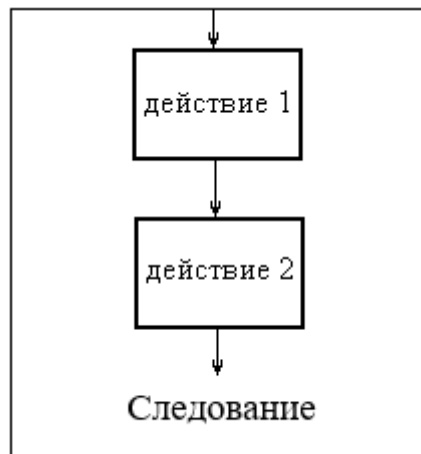


Рисунок 2 – Базовая структура,
которая называется следование или последовательность

2.3 Примеры для составления схем и программ

Пример 1: Требуется вычислить значение функции $y(x) = x^2$, при $x=3$.

Схема работы программы для примера 1 представлена на рисунке 3.

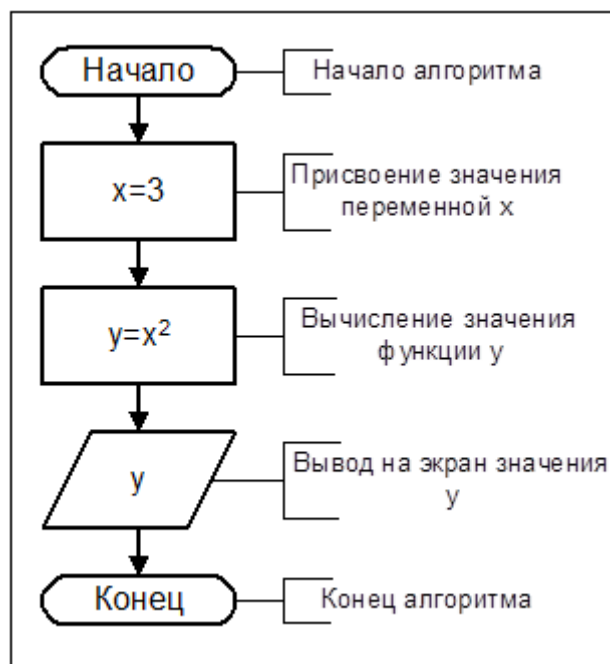


Рисунок 3 – Схема работы программы для примера 1

Текст программы, составленный на языке Python для примера 1 представлен на рисунке 4.

```

x = 3
y = x**2
print('Значение функции равно', y)

```

Рисунок 4 – Текст программы, составленный на языке Python для примера 1

Схема работы программы для примера 2 представлена на рисунке 5. Причем, новые операторы в тексте программы выделены крупным шрифтом.

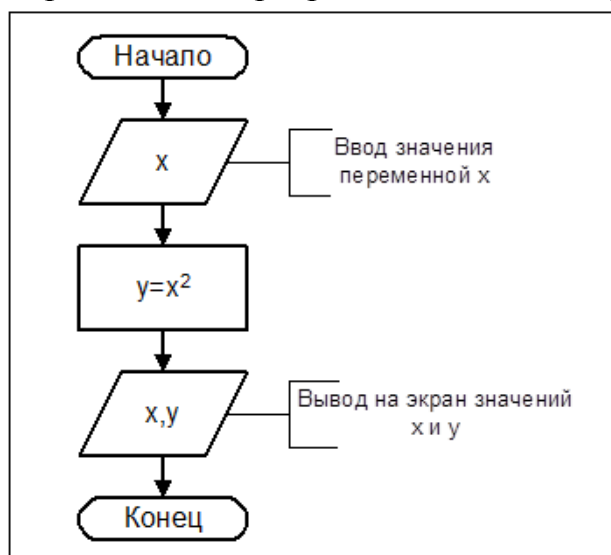


Рисунок 5 – Схема работы программы для примера 2

Текст программы для примера 2 представлен на рисунке 6.

```

x = int(input())
y = x**2
print(f'x = {x}, y = {y}')

```

```

x = int(input('Введите значение x: '))
y = x**2
print(f'x = {x}, y = {y}')

```

а)

б)

Рисунок 6 – Текст программы для примера 2 (варианты записи программы)

2.5. Оборудование и программное обеспечение

1. Персональная ЭВМ.
2. Операционная система Windows/Linux.
3. Среда программирования IDLE или другая среда, содержащая язык программирования Python.

2.6. Задание на работу

1. Изучить теоретические положения лабораторной работы.
2. Изучить команды меню главного окна программы, синтаксис и простейшие операторы языка Python.
3. Научиться сохранять набранные тексты программ в файлы.
4. Научиться загружать сохраненные файлы с текстами программ в окно редактирования среды IDLE (другой среды).
5. Получить результаты работы программ в соответствии с примерами 1-2.
6. В примерах 1-3 произвольно изменить фразу и объект-контейнер, произвольно изменить индексы для работы цикла for.
8. Показать результаты работы преподавателю.
9. Ответить на контрольные вопросы.
10. Оформить отчет.

2.7. Контрольные вопросы

1. Опишите синтаксис языка Python.
2. Опишите простейшие операторы языка Python.
3. Опишите структуру ветвления (выбора) языка Python.
4. Что позволяет осуществлять функция len в языке Python?
5. Что такое объект-контейнер?
6. Что означает запись end= ‘ ’ на языке Python?

2.8. Правила оформления отчета

Отчет должен содержать:

1. Номер, название и цель лабораторной работы.
2. Описание процесса выполнения примеров 1-3 (коды и результаты работы программ).
3. Описание процесса изменения примеров 1-3 в соответствии с заданием (коды и результаты работы программ).
4. Вывод по работе.

Практическая работа №3
Основы алгоритмизации и программирования.
Вычисление значений функций
в интегрированной среде программирования Python
(изучение структуры «ветвление»)

3.1. Цель работы

Получение навыков по вычислению значений функций при помощи написания программ на языке Python (изучение ветвления или выбора).

3.2. Теоретические положения

Для вычисления значений функций при помощи написания программ на языке Python необходимо изучить вышеизложенную теорию и проделать вышеизложенные работы.

В работе в основу составления схем программ для вычисления значения функций заложено понятие «ветвление» или «выбор».

На рисунке 7 показаны базовые структуры, представляющие собой ветвление или выбор с одним или двумя действиями.

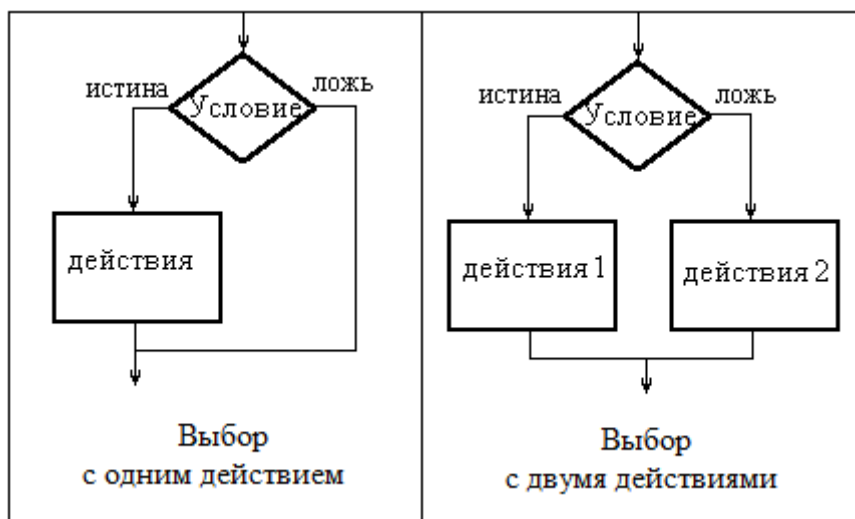


Рисунок 7 – Ветвление или выбор с одним или двумя действиями

В тексте программ ветвление осуществляется при помощи оператора условия if then (если то). В программе указывается условие при помощи if, затем действие, которое должно выполняться при соблюдении условия при помощи then. Такая структура в программе соответствует схеме выбора с одним действием.

Также может указываться действие, которое выполниться при несоблюдении условия при помощи else. Такая структура в программе соответствует схеме выбора с двумя действиями.

Примечание: Перед и после else запятая не ставиться.

3.3. Примеры для составления схем и программ

Пример 3: Составить программу, вычисляющую функцию

$$y(x) = \begin{cases} x^2, & x > 0 \\ x^3, & x \leq 0 \end{cases}$$

Схема работы программы для примера 3 представлена на рисунке 11.

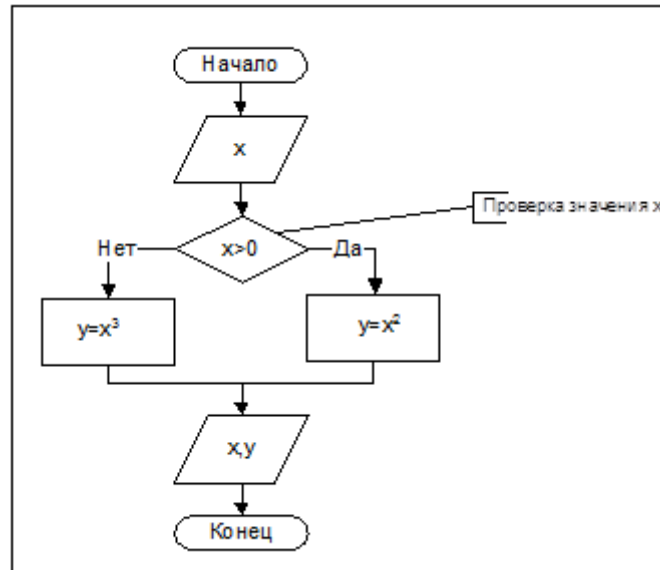


Рисунок 8 – Схема работы программы для примера 3

Текст программы для примера 3 представлен на рисунке 9.

`x = int(input())`

`if x > 0:`

`y = x**2`

`else:`

`y = x**3`

`print(f'x = {x}, y = {y}')`

а)

`x = int(input('Введите значение x: '))`

`if x > 0:`

`y = x**2`

`else:`

`y = x**3`

`print(f'x = {x}, y = {y}')`

б)

Рисунок 9 – Текст программы для примера 3

Здесь реализовано ветвление или выбор с двумя действиями при помощи структуры `if then else`.

Разница между программы под а) и под б) в том, что во втором случае пользователь программы при ее запуске увидит запись 'Введите значение x' и, таким образом, будет знать, что необходимо ввести значение переменной с клавиатуры.

Примечание: Если при выводе результатов работы программы после переменных `x` и `y` ввести значения цифр, например, `3:2`, то количество знаков после запятой в полученном значении будет ограничено.

3.4. Оборудование и программное обеспечение

1. Персональная ЭВМ.
2. Операционная система Windows/Linux.

3. Среда программирования IDLE или другая среда, содержащая язык программирования Python.

3.5 Задание на работу

1. Изучить теоретические положения лабораторной работы.
2. Набрать и отладить два варианта программы в соответствии с примером 3.
3. Получить результаты работы программы.
4. Показать результаты работы программ преподавателю.
5. Ответить на контрольные вопросы.
6. Оформить отчет.

3.6. Контрольные вопросы

1. Что такое ветвление или выбор?
2. Какие виды ветвления или выбора существуют?
3. Для чего нужен оператор if?
4. Какие функции выполняет then?
5. Какие функции выполняет else?
6. Какими особенностями обладает структура if then else?

3.7. Правила оформления отчета

Отчет должен содержать:

1. Номер, название и цель лабораторной работы.
2. Описание процесса выполнения примеров (коды, схемы и результаты работы программ).
3. Вывод по работе.

Практическая работа №4
Основы алгоритмизации и программирования.
Вычисление значений функций при различных переменных
в интегрированной среде программирования Python
(изучение структуры «цикл»)

4.1. Цель работы

Получение навыков вычисления значений функций при различных переменных на языке Python (изучение циклов).

4.2. Теоретические положения

Для вычисления значений функций при различных переменных при помощи написания программ на языке Python необходимо изучить вышеизложенную теорию и проделать вышеизложенные работы.

Действия, неоднократно повторяющиеся в программе при различных переменных, называются циклом. Возможны различные способы организации циклов:

- цикл с предусловием (условие стоит перед действием – телом цикла);
- цикл с постусловием (условие стоит после действия – тела цикла);
- цикл с параметром (тело цикла повторяется указанное количество раз для переменного цикла).

На рисунке 10 показаны базовые структуры схем программ с различными циклами.

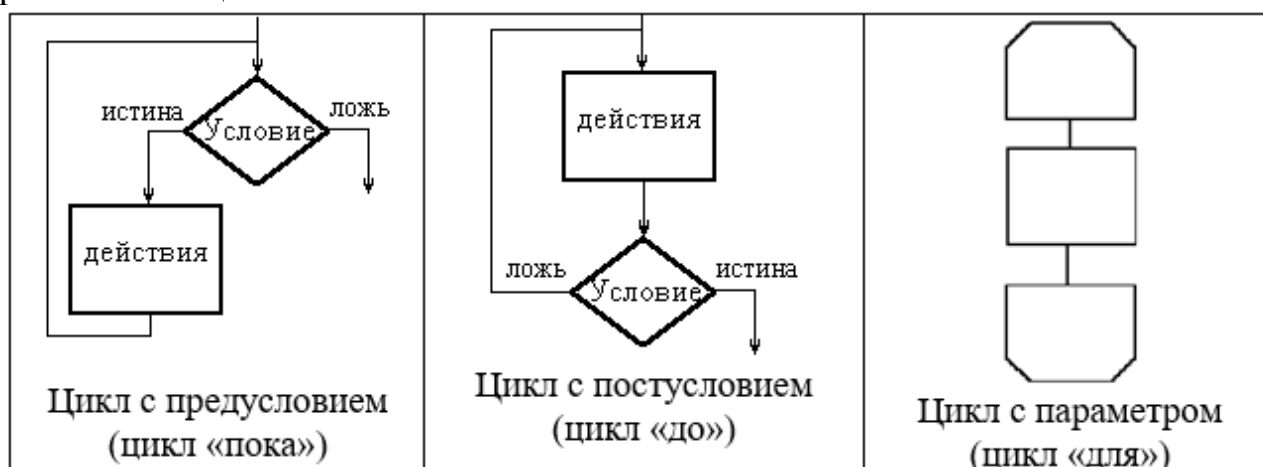


Рисунок 10 – Базовые структуры схем программ с различными циклами

На языке Python (в среде Free Pascal):

- цикл с предусловием организуется при помощи оператора while (пока);
- цикл с постусловием организуется при помощи оператора until (до тех пор);
- цикл с параметром организуется при помощи оператора for (для).

Выбор базовой структуры определяется условиями задачи и навыками разработчика программ.

4.3. Примеры для составления схем и программ

Пример 4: Составить программу табуляции функции

$$y(x) = \begin{cases} x^2, & x > 0 \\ x^3, & x \leq 0 \end{cases},$$

при $x \in [-2, 3]$ с шагом 1.

Процесс табуляции – это процесс табличного группирования вычисляемых данных. Иными словами, в примере необходимо получить таблицу, содержащую значения входной переменной x и выходной переменной y , вычисленной при соответствующей переменной x .

То есть в примере требуется получить данные определённого вида, как показано в таблице 4.

Таблица 4 – Пример табуляции функции

x	y
-2	-8
-1	-1
0	0
1	1
2	4
3	9

Схема работы программы для примера 4 показана на рисунке 11.

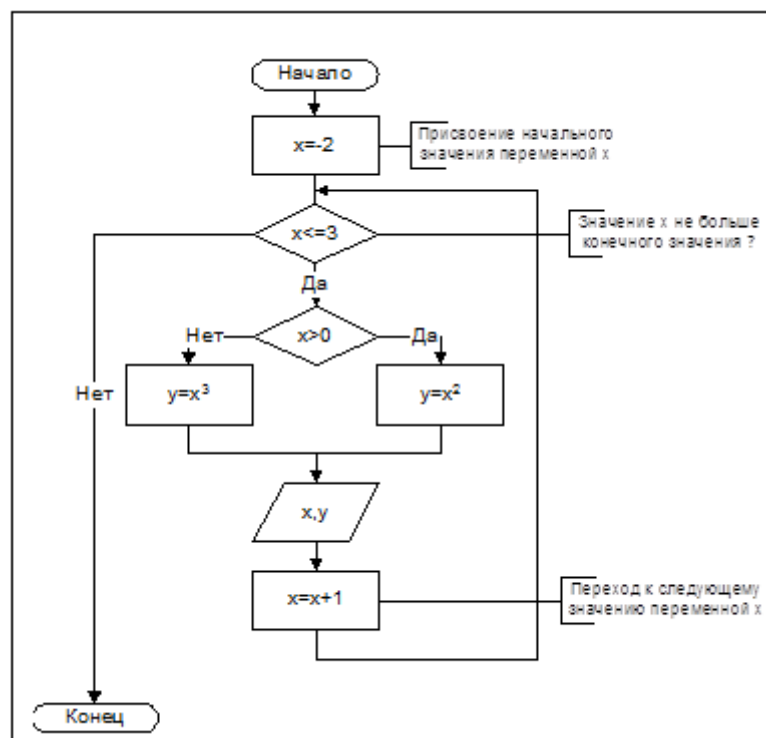


Рисунок 11 – Схема работы программы для примера 4

Текст программы для примера 4 представлен на рисунке 12.

<code>x = -2</code>	<code>x = int(input('Введите значение x: '))</code>
<code>while x <= 3:</code>	<code>x = -2</code>
<code>if x > 0:</code>	<code>while x <= 3:</code>
<code>y = x**2</code>	<code>if x > 0:</code>
<code>else:</code>	<code>y = x**2</code>
<code>y = x**3</code>	<code>else:</code>
<code>print(x, y)</code>	<code>y = x**3</code>
<code>x+= 1</code>	<code>print(x, y)</code>
	<code>x+= 1</code>

a)

б)

Рисунок 12 – Текст программы для примера 4

В указанном примере реализован цикл с предусловием `while` (пока).

Примечание: Строка программы `y:=sqr(x)*x` вычисляет значение куба переменной `x` и записывает это значение в переменную `y`. Оператор `writeln` может быть заменен на оператор `write`. Перед и после `else` запятая не ставится. Если при выводе результатов программы после переменных `x` и `y` ввести значения цифр, например, 3:2, то количество знаков после запятой в полученном значении будет ограничено.

4.4. Оборудование и программное обеспечение

1. Персональная ЭВМ.
2. Операционная система Windows/Linux.
3. Среда программирования IDLE или другая среда, содержащая язык программирования Python

4.5 Задание на работу

1. Изучить теоретические положения лабораторной работы.
2. Набрать и отладить два варианта программы в соответствии с примером 4.
3. Получить результаты работы программы.
4. Показать результаты работы программ преподавателю.
5. Ответить на контрольные вопросы.
6. Оформить отчет.

4.6. Контрольные вопросы

1. Что такое цикл?
2. Какие виды циклов существуют?
3. Что такое табуляция?
4. Как в среде Python реализовать цикл «пока»?
5. Как в среде Python реализовать цикл «до тех пор»?
6. Как в среде Python реализовать цикл «для»?

4.7. Правила оформления отчета

Отчет должен содержать:

1. Номер, название и цель лабораторной работы.
2. Описание процесса выполнения примеров (коды, схемы и результаты работы программ).
3. Вывод по работе.

Практическая работа №5

Основы алгоритмизации и программирования.

Вычисление значений функций при различных переменных, введенных с клавиатуры, в интегрированной среде программирования Python (изучение структуры «цикл»)

5.1. Цель работы

Получение навыков вычисления значений функций при различных переменных на языке Python, введенных с клавиатуры, (изучение циклов).

5.2. Теоретические положения

Необходимо смотреть вышеизложенную теорию. При табуляции функции, т.е. при вычислении значений функций при различных переменных, введенных с клавиатуры, необходимо, чтобы введенные переменные записывались в некоторые переменные. Иными словами, необходимо организовать процесс записи введенных с клавиатуры значений в переменные.

5.3. Примеры для составления схем и программ

Пример 5: Составить программу табуляции функции

$$y(x) = \begin{cases} x^2, & x > 0 \\ x^3, & x \leq 0 \end{cases}$$

при $x \in [-a, b]$ с шагом h . Значения a, b, h ввести с клавиатуры.

Схема работы программы для примера 5 представлена на рисунке 13.

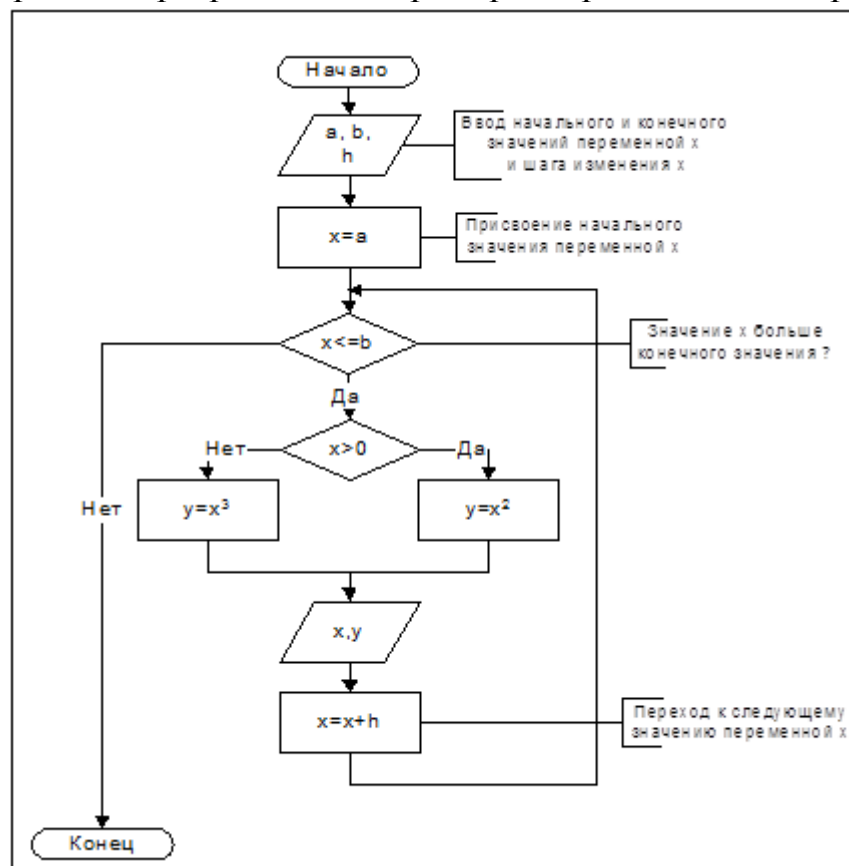


Рисунок 13 – Схема работы программы для примера 5

Текст программы для примера 5 представлен на рисунке 14.

```
print('a, b, h')
a = int(input())
b = int(input())
h = int(input())

x = a

while x <= b:
    if x > 0:
        y = x**2
    else:
        y = x**3
    print(x, y)
    x = x + h
```

Рисунок 14 – Текст программы для примера 5

5.4. Оборудование и программное обеспечение

1. Персональная ЭВМ.
2. Операционная система Windows/Linux.
3. Среда программирования IDLE или другая среда, содержащая язык программирования Python.

5.5 Задание на работу

1. Изучить теоретические положения лабораторной работы.
2. Набрать и отладить вариант программы в соответствии с примером 5.
3. Получить результаты работы программы.
4. Показать результаты работы программы преподавателю.
5. Ответить на контрольные вопросы.
6. Оформить отчет.

5.6. Контрольные вопросы

1. Как организовать ввод переменных с клавиатуры?
2. Чем отличаются программы, работающие с переменными, значения которых прописаны в программе от программ, где значения переменных вводятся с клавиатуры?

5.7. Правила оформления отчета

Отчет должен содержать:

1. Номер, название и цель лабораторной работы.
2. Описание процесса выполнения примеров (коды, схемы и результаты работы программ).
3. Вывод по работе.

Практическая работа №6

Организация массивов на языке Python

6.1. Цель лабораторной работы

Научиться организовывать массивы на языке Python

6.2 Теоретические положения

Служебное слово *row* на языке Python необходимо для осуществления манипуляций со строками матриц.

Пример 6

Сформировать массив, имеющий шесть строк и пять столбцов, причем, чтобы каждый элемент массива, записывающийся при движении по строке этого массива, был равен произведению текущего номера строки на текущий номер столбца.

Выполнение примера 6

На рисунке 15 показан код программы к примеру 6.

```
m=6 # количество записей в строке
n=5 # количество строк
a=[[ (i) * (j) for j in range(m)] for i in range(n)] # каждому элементу строки присвоить i*j для каждой строки
for row in a: # для каждой строки массива
    print(' '.join([str(elem) for elem in row])) #печатать значение каждого элемента текущей строки
    # ' '.join убирает кавычки и [] при выводе
    # str(elem) переводит значение элемента строки в тип string

# отсчет нумерации элементов начинается с 0 т.е. первые i и j будут = 0
# [(i+1) * (j+1) for j in range(m)] for i in range(n)] для отсчета с 1
```

Рисунок 15 – Код программы к примеру 7

Пример 7

Сформировать массив, имеющий шесть строк и пять столбцов, причем, чтобы каждый элемент массива, записывающийся при движении по строке этого массива, был равен произведению текущего номера строки на текущий номер столбца и, кроме того, если значение элемента массива больше 5, заменить этот элемент на элемент 44.

Выполнение примера 7

На рисунке 16 показан код программы к примеру 7.

```
m=6 # количество записей в строке
n=5 # количество строк
a=[[ (i) * (j) for j in range(m)] for i in range(n)] # каждому элементу строки присвоить i*j для каждой строки
for i in range(n):
    for j in range(m):
        if a[i][j]>5:
            a[i][j]=44
for row in a: # для каждой строки массива
    print(' '.join([str(elem) for elem in row])) #печатать значение каждого элемента текущей строки
    # ' '.join убирает кавычки и [] при выводе
    # str(elem) переводит значение элемента строки в тип string

# отсчет нумерации элементов начинается с 0 т.е. первые i и j будут = 0
# [(i+1) * (j+1) for j in range(m)] for i in range(n)] для отсчета с 1
```

Рисунок 16 – Код программы к примеру 8

6.3. Оборудование и программное обеспечение

1. Персональная ЭВМ.
2. Операционная система Windows/Linux.

3. Среда программирования IDLE или другая среда, содержащая язык программирования Python.

6.4. Задание на работу

1. Изучить теоретические положения лабораторной работы.

2. Получить результаты работы программ в соответствии с примерами 7-

9

3. Для примера 7:

а) произвольно изменить количество записей по строке и столбцу и проверить правильность работы программы путем расчетов;

б) организовать заполнение массива элементами при условии, что подсчет индекса строки и столбца начинается с 1 (проверить правильность работы программы путем расчетов);

в) организовать заполнение массива элементами при условии, что подсчет индекса строки и столбца начинается с 2 (проверить правильность работы программы путем расчетов).

4. Для примера 8:

а) произвольно изменить количество записей по строке и столбцу и проверить правильность работы программы путем расчетов.

б) осуществить замену элемента, значение которого равно 0, на любой произвольный элемент;

в) осуществить замену элемента, значение которого меньше b , на элемент (i, j) , если b, i, j – произвольные параметры (отсчет индекса строки и столбцов начинается с 0).

6. Разработать программу для формирования элементов матрицы путем сложения индексов строк и столбцов, выполнить в программе произвольные замены элементов.

7. Изобразить схемы работы программ в соответствии с ГОСТ 19.701-90 (использовать графический редактор).

8. Показать результаты работы преподавателю.

9. Ответить на контрольные вопросы.

10. Оформить отчет.

6.5. Контрольные вопросы

1. Что означает запись `row` на языке Python?

2. Что означает запись `join` на языке Python?

3. Что означает запись `[str(elem) for elem in row]` на языке Python?

3. Как сформировать двумерный массив (матрицу) на языке Python?

4. Какими способами можно заполнить матрицу элементами?

5. Как реализовать замены элементов?

6. Как осуществляется подсчет индексов строк и столбцов матриц в языке Python?

7. Что необходимо сделать, чтобы подсчет индексов строк и столбцов матриц в языке Python начинался не с 0?

8. Какие операторы были использованы для решения примеров в текущей лабораторной работе?

9. Какие циклы были использованы для решения примеров в текущей лабораторной работе?

10. Как оформляется комментарий в программах, написанных на языке Python?

6.6. Правила оформления отчета

Отчет должен содержать:

1. Номер, название и цель лабораторной работы.
2. Описание процесса выполнения примеров (коды, схемы и результаты работы программ).
3. Вывод по работе.

ПРАКТИЧЕСКАЯ РАБОТА № 7

Основы алгоритмизации и программирования.

Суммирование значений

в интегрированной среде программирования IDLE Python

(изучение структуры «цикл»)

7.1. Цель работы

Получение навыков по суммированию значений в программе, составленной на языке Python (составление сложных программ).

7.2. Теоретические положения

В языке Python существуют различные способы суммирования значений.

Первый и самый простой это функция *sum()*. Встроенная функция *sum()* – это эффективный и питонический способ суммирования списка числовых значений. Еще с помощью *sum()* можно объединять списки и кортежи. Функция *sum()* используется и с другими числовыми типами Python, такими как *float*, *complex* и т.д.

Следующий способ с помощью применения циклов *for* и *while*.

Цикл *for* используется в том случае, когда необходимо выполнить некоторую часть кода до тех пор, пока не будет выполнено заданное условие. Цикл *for* также называют циклом с предусловием. Лучше использовать цикл *for*, если количество итераций известно заранее.

Цикл *while* используется в сценарии, когда мы не знаем заранее количество итераций. Блок операторов в цикле *while* выполняется до тех пор, пока не будет выполнено условие, указанное в цикле *while*. Его также называют циклом с предварительной проверкой условия.

При использовании циклов в Python вводится переменная, значение которой в самом начале обнуляется (переменная содержит значение, равное 0). Если в дальнейшем к нулю прибавить требуемое значение, а потом к полученной сумме еще раз прибавить требуемое значение и так в цикле некоторое количество раз, то будет определена необходимая сумма.

Примечание: При необходимости вычисления произведения, вводится переменная, значение которой равно 1. Если 1 умножить на значение в цикле некоторое количество раз, то будет определено требуемое произведение.

7.3. Примеры для составления схем и программ

Пример 6: Несколько примеров использования *sum()* со значениями разных числовых типов.

Текст программы для примера 6 представлен на рисунке 18.

```
File Edit Format Run Options Window Help
# Сумма чисел с плавающей запятой
print (sum([10.2, 12.5, 11.8]))

# Сумма комплексных чисел
print(sum([3 + 2j, 5 + 6j]))
```

Рисунок 18 – Текст программы для примера 6

Результат выполнения программы для примера 6 представлена на рисунке 19.

```
Сумма чисел с плавающей запятой: 34.5  
Сумма комплексных чисел: (8+8j)
```

Рисунок 19 – Результат работы программы для примера 6

Пример 7: Известны оценки абитуриента на четырех экзаменах. Определить сумму набранных им баллов. Оценки вводятся в цикле.

Текст программы для примера 7 представлен на рисунке 20.

```
summa=0 #обнуление переменной суммы  
for i in range(1,5): #цикл для ввода только четырёх оценок  
    oценка =int(input('Введите оценку: ')) #ввод целочисленных оценок  
    summa=summa+oценка #суммирование всех оценок  
print('Сумма баллов абитуриента: ', summa) #вывод суммы на экран
```

Рисунок 20 – Текст программы для примера 7

Примечание 1: Строка суммирования оценок может быть представлена и в таком виде: *summa+=oценка*, смысл суммирования от этого не меняется. Для умножения также есть два варианта: *summa*=oценка* и *summa= summa*oценка*.

Примечание 2: Текст после знака # - это комментарий к программе.

Схема работы программы для примера 7 представлена на рисунке 21.

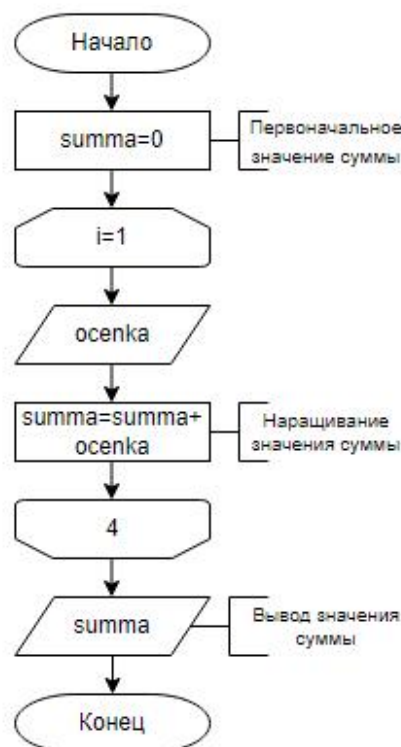


Рисунок 21 – Схема работы программы для примера 7

Здесь переменная, которая накапливает сумму, обозначена *summa*. Значение, содержащееся в этой переменной, подается на вывод после окончания работы программы.

Пример 8: Дано натуральное число. Определить сумму цифр в нем.

Текст программы для примера 8 представлен на рисунке 22.

```

n = 356 #инициализация переменной с натуральным числом
print ('Натуральное число=', n) #вывод на экран натурального числа
summa = 0 #обнуление переменной суммы
while n > 0: #цикл проверяющий является ли натуральное число положительным
    summa += n % 10
    n = n // 10 #делит натуральное число на 10 без остатка
print('Сумма цифр: ', summa) #вывод на экран суммы
#summa += n % 10 - n%10 определяет остаток от деления n и суммирует их

```

Рисунок 22 – Текст программы для примера 8

Схема работы программы для примера 8 представлена на рисунке 23.

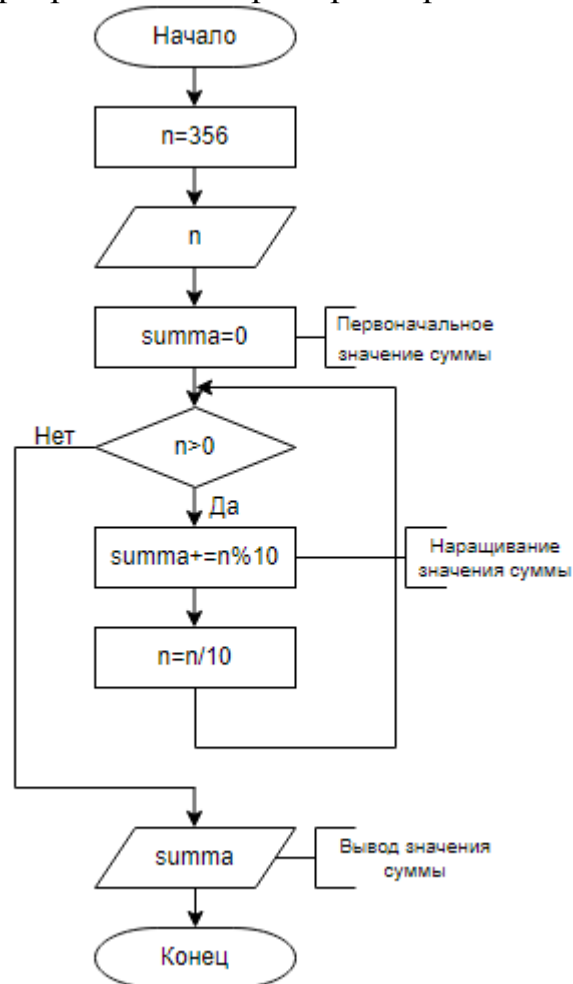


Рисунок 23 – Схема работы программы для примера 8

Здесь переменная, которая накапливает сумму, обозначена summa. Значение, содержащееся в этой переменной, подается на вывод после окончания работы программы.

Примечание: Необходимо изучить вышеизложенную теорию и проделать вышеизложенные работы.

7.4. Оборудование и программное обеспечение

1. ПЭВМ IBM PC.
2. Операционная система Windows.
3. Интегрированная среда IDLE Python.

7.5 Задание на работу

1. Изучить теоретические положения лабораторной работы.

2. Набрать и отладить вариант программы в соответствии с примерами 6,7,8.
3. Получить результаты работы программы.
4. Показать результаты работы программы преподавателю.
5. Ответить на контрольные вопросы.
6. Оформить отчет.

7.6. Контрольные вопросы

1. Как в программе получить сумму некоторых значений?
2. Как в программе получить произведение некоторых значений?

7.7. Правила оформления отчета

Отчет должен содержать:

1. Номер работы, цель и название.
2. Сформулированные примеры, для которых составляются программы.
3. Схемы работы программ, изображенные в соответствии с ГОСТ 19.701-90.
4. Тексты введенных и отлаженных программ (фото текстов программ).
5. Результаты работы программ (фото результатов работы программ).
6. Описание выполненных в лабораторной работе действий (коротко как набирались и отлаживались программы).
7. Вывод по работе.

ПРАКТИЧЕСКАЯ РАБОТА № 8

Основы алгоритмизации и программирования.

Вычисление значений факториалов и степеней в интегрированной среде программирования IDLE Python

8.1. Цель работы

Получение навыков по вычислению значений факториалов и степеней в программе, составленной на языке Python.

8.2. Теоретические положения

В языке Python ранее изученные циклы также могут применяться для нахождения факториала и степеней числа. Однако, в данном языке программирования существуют рекурсия и встроенные математические модули.

Математическая модуль существенно сокращает код и обеспечивает простой способ вычисления факториала любого положительного целого числа. Конечно, в модуле есть предопределенный метод *factorial()*, который принимает целое число в качестве аргумента и возвращает факториал числа. Для того, чтобы использовать различные математические модули необходима библиотека *math*. Библиотека *Math* в Python обеспечивает доступ к некоторым популярным математическим функциям и константам, которые можно использовать в коде для более сложных математических вычислений. Библиотека является встроенным модулем Python. Чтобы подключить модуль, необходимо в начале программы прописать следующую инструкцию:

```
import math
```

Рекурсия — это когда функция вызывает сама себя. Рекурсивные решения лучше всего подходят, когда проблема имеет чёткие подзадачи, которые необходимо повторить, и, если вы не уверены, сколько раз вам нужно будет выполнить цикл с итеративным решением.

8.3. Примеры для составления схем и программ

Пример 9: Вычислить факториал числа n ($n!=1*2*3*...*n$).

1) Текст программы через рекурсию для примера 9 представлен на рисунке 24.

```
n=int(input('Введите число для нахождения факториала: ')) #ввод целочисленного числа
def fact(n):
    if(n==1 or n==0): #цикл проверки числа
        return 1 #возвращает единицу
    else: #иначе
        return n*fact(n-1) #возвращает вычисление факториала от числа
print('факториал числа ',n,' : ',fact(n)) #вывод на экран факториала числа
#def fact(n)- функция нахождения факториала
```

Рисунок 24 – Текст программы для примера 9

2) Текст программы через *factorial()* для примера 9 представлен на рисунке 25.

```
import math #библиотека для использования стандартных математических функций
n=int(input('Введите число для нахождения факториала: ')) #ввод целочисленного числа
print('факториал числа ',n,' : ',math.factorial(n)) #вывод на экран факториала числа
#math.factorial(n) - встроенный модуль в библиотеке math для вычисления факториала
```

Рисунок 25 – Текст программы для примера 9

Схема работы программы для примера 9 представлена на рисунке 26.

1)

2)

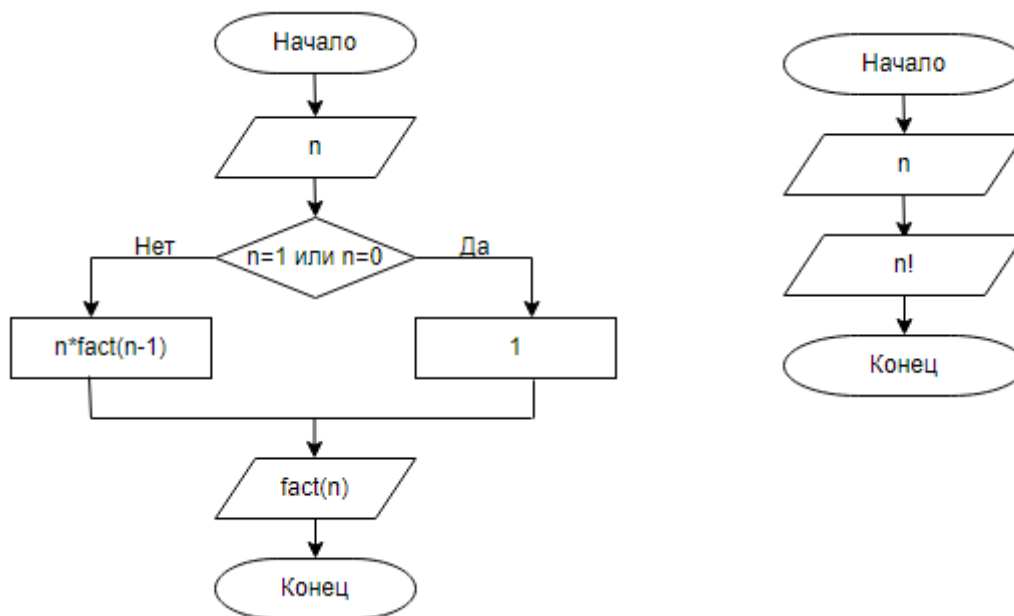


Рисунок 26 – Схема работы программы для примера 9

- 1) Входные данные для работы программы – n – целое число, факториал которого необходимо вычислить. Выходные данные – переменная $fact$, содержащая значение факториала для ранее введенного числа n . Промежуточные данные – функция $def\ fact(n)$, содержащая цикл if определяющий, если n равно 1 или 0, то вернется значение 1. Если введено любое, отличное от 1 и 0, натуральное число, то функция $fact$ будет содержать значение, равное накопленному произведению с шагом $n-1$, когда функция становится больше n , функция закрывается и последнее значение, записанное в переменную $fact$ выводится на экран.
- 2) Входные данные для работы программы – n – целое число, факториал которого необходимо вычислить. Выходные данные – переменная n , вычисленная с помощью математического модуля $math.factorial()$.

Пример 10: Вычислить a^n ($n > 0$).

Текст программы для примера 10 представлен на рисунке 27.

```

a=int(input('Введите основание степени a: ')) #ввод целочисленного числа
n=int(input('Введите показатель степени n: ')) #ввод целочисленного числа
print ('Число', a, 'в степени', n, '=', pow(a,n)) #вывод на экран число в степени
#pow(a,n) - встроенный математический модуль вычисляющий число в степени
  
```

Рисунок 27 – Текст программы для примера 10

Схема работы программы для примера 10 представлена на рисунке 28.

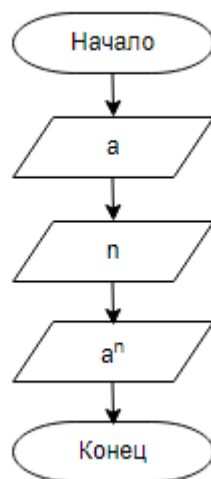


Рисунок 28 – Схема работы программы для примера 10

Входные данные для работы программы – a – целое число, которое необходимо возвести в целую положительную степень n . Выходные данные – переменные a и n , вычисленные с помощью математического модуля `pow()`.

Известно, что для того, чтобы получить целую степень n числа a , необходимо умножить число a само на себя n раз.

8.4. Оборудование и программное обеспечение

1. ПЭВМ IBM PC.
2. Операционная система Windows.
3. Интегрированная среда IDLE Python.

8.5 Задание на работу

1. Изучить теоретические положения лабораторной работы.
2. Набрать и отладить вариант программы в соответствии с примерами 9,10.
3. Получить результаты работы программы.
4. Показать результаты работы программы преподавателю.
5. Ответить на контрольные вопросы.
6. Оформить отчет.

8.6. Контрольные вопросы

1. Какими способами можно вычислить факториал и степень числа в Python?
2. Что такое математические модули?
3. Что надо сделать, чтобы математические модули работали?
4. Что такое рекурсия?
5. Какие ещё математические модули есть в Python?

8.7. Правила оформления отчета

Отчет должен содержать:

1. Номер работы, цель и название.
2. Сформулированные примеры, для которых составляются программы.
3. Схемы работы программ, изображенные в соответствии с ГОСТ 19.701-90.
4. Тексты введенных и отлаженных программ (фото текстов программ).

5. Результаты работы программ (фото результатов работы программ).
6. Описание выполненных в лабораторной работе действий (коротко как набирались и отлаживались программы).
7. Вывод по работе.

Практическая работа №9
Основы алгоритмизации и программирования.
Обработка массивов (часть 1) на языке Python
(в среде IDLE Python)

9.1. Цель работы

Освоить основные приемы обработки массивов на языке Python.

9.2. Теоретические положения

Часто для работы с множеством однотипных данных оказывается удобным использовать массивы.

В Python массивы — это контейнеры, способные хранить более одного элемента одновременно.

В частности, они представляют собой упорядоченный набор элементов, каждое значение которого относится к одному и тому же типу данных.

Массивы разделяются на одномерные и многомерные.

Например, последовательность чисел 2,4,6,8 является одномерным массивом.

Последовательность чисел, записанных в виде таблицы – в виде строк и столбцов, называется многомерным массивом, например, такой массив как

2	4
6	8

называется двумерным (в двумерном массиве строки – первое измерение, столбцы – второе измерение).

В Python процедура объявления и вывода массива может выполняться следующими способами:

1.

```
mass=[1, 2, 3, 4, 5]
print('Вывод:', mass)
print('Вывод[0]:', mass[0])
```
2.

```
mass = list('итерируемый объект')
print('Вывод:', mass)
```
3.

```
mass = [i for i in range(5)]
print('Вывод:', mass)
print('Вывод[3]:', mass[3])
```
4. Массив можно создать в Python, импортировав модуль массива в программу.

```
from array import *
arrayName = array(typecode, [initializers])
```
5. Двухмерный массив в Python можно объявить следующим образом.

```
example_array = [[-1, 0, 0, 1], [2, 3, 5, 8]]
print(example_array[0])
```

```
print(example_array[0][3])
```

6. Так же можно определить трехмерный массив или многомерный массив.

```
example_array = [[[-1, 0], [0, 1]], [[2, 3], [5, 8]]]
print(example_array[1])
print(example_array[0][1][0])
```

Массивы подвержены следующим операциям:

- ввод/вывод элементов массива;
- вычисление суммы или произведения элементов массива;
- поиск минимального или максимального элемента в массиве;
- сортировка элементов в массиве (различными методами);
- добавление или удаление элементов массива.

9.3. Задания для составления схем и программ

Пример 11: Дан массив А, состоящий из К целых положительных чисел. Записать все четные числа из массива А в массив В.

Решение задачи заключается в следующем: последовательно перебираются элементы массива А. Если среди них находятся четные, то они записываются в массив В.

В таблице 5 показан исходный массив А.

Таблица 5 – Исходный массив А

i	1	2	3	4	5	6	7	8
A	3	7	4	9	2	8	1	6

Переменная i в массиве А указывает позицию элемента. А – значения элементов в массиве.

В таблице 6 показан формируемый массив В.

Таблица 6 – Формируемый массив В

B	4	2	8	6
m	1	2	3	4

Переменная m в массиве В указывает позицию элемента. В – значения элементов в массиве, которые выбраны из массива А в соответствии со свойством четности.

Первый четный элемент хранится в массиве А в позиции 3, второй и третий в позиции 5 и 6 соответственно, четвертый – в позиции 8. В массиве В этим элементам соответствуют совершенно иные позиции, а именно 1,2,3,4, для формирования чего необходимо определить новую переменную m.

Текст программы для примера 11 представлен на рисунке 29.

```
Mass_A=[3,7,4,9,2,8,1,6]#инициализация массива А
print ("Массив А:\n ", (Mass_A), "\n")#вывод массива А на печать
Mass_B= [i for i in Mass_A if i % 2 == 0]#создание массива В
print("Массив В:\n ", (Mass_B))#вывод массива В на печать
#[i for i in Mass_A if i % 2 == 0] проверяет каждый элемент
#массива А чтобы его остаток при делении на 2 был = 0, то есть ищет четные числа
```

Рисунок 29 – Текст программы для примера 11

Схема работы программы представлена на рисунке 30.

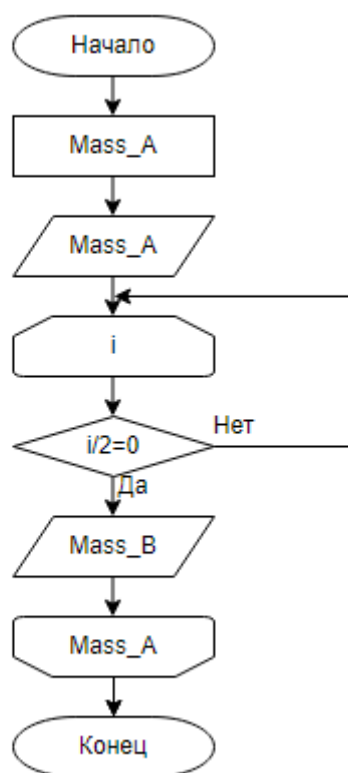


Рисунок 30 – Схема работы программы для примера 11

9.4. Оборудование и программное обеспечение

1. ПЭВМ IBM PC.
2. Операционная система Windows.
3. Интегрированная среда IDLE Python.

9.5 Задание на работу

1. Изучить теоретические положения лабораторной работы.
2. Набрать и отладить программу в соответствии с примером 11.
3. Получить результаты работы программы.
4. Показать результаты работы программы преподавателю.
5. Ответить на контрольные вопросы.
6. Оформить отчет.

9.6. Контрольные вопросы

1. Что такое массив?
2. Какие массивы существуют?
3. Как задается одномерный массив?
4. Как задается двумерный массив?
5. Какие операции можно осуществлять над массивами?
6. Что такое однородность данных?
7. Какими способами задается одномерный массив на языке Python?
8. Какими способами задается двумерный массив на языке Python?

9.7. Правила оформления отчета

Отчет должен содержать:

1. Номер работы, цель и название.
2. Сформулированные примеры, для которых составляются программы.

3. Схемы работы программ, изображенные в соответствии с ГОСТ 19.701-90.

4. Тексты введенных и отлаженных программ (фото текстов программ).

5. Результаты работы программ (фото результатов работы программ).

6. Описание выполненных в лабораторной работе действий (коротко как набирались и отлаживались программы).

7. Вывод по работе.

Практическая работа №10
Основы алгоритмизации и программирования.
Обработка массивов (часть 2) на языке Python
(в среде IDLE Python)

10.1. Цель работы

Освоить основные приемы обработки массивов на языке Python.

10.2. Теоретические положения

Над массивами, как уже говорилось ранее, можно реализовывать самые различные операции, в том числе и перезапись элементов в другой массив. Запись элементов в другой массив осуществляется на основании проверки условий, в соответствии с которыми будет выбран элемент для записи его в другой массив. Ввод массивов был рассмотрен выше. Структуры схем для программ по обработке массивов содержат циклы, как правило, с параметром и различного рода условия.

10.3. Задания для составления схем и программ

Пример 12: Задан массив Y из n целых чисел. Сформировать массив Z таким образом, чтобы в начале шли отрицательные элементы массива Y, затем положительные и наконец нулевые.

Текст программы для примера 12 представлен на рисунке 31.

```
def task(Mass_Y):  
  
    Mass_Z = [] #инициализация массива A  
    index = 0 #инициализация переменной для счёта элементов  
    for a in Mass_Y: #инициализация массива A  
        if a < 0:  
            Mass_Z.insert(index, a)  
            index += 1  
        elif a > 0:  
            Mass_Z.insert(index, a)  
        else:  
            Mass_Z.append(a)  
    return Mass_Z  
  
print('Введите количество элементов массива: ') #вывод на экран  
N=int(input()) #ввод целочисленного числа элементов массива с клавиатуры  
  
print('Введите массив C через пробел: ') #вывод на экран  
Mass_Y=[] #инициализация массива  
Mass_Y=list(map(int, input().split()[:N]))  
print('Массив Y: ') #вывод на экран  
print(Mass_Y) #вывод на экран введенного массива  
  
print('Mass_Z: ') #вывод на экран  
print(task(Mass_Y)) #вывод на экран результат выполнения функции  
#list(map(int, input().split()[:N])) - list-создает список,  
#map-передает каждому введенному элементу функцию,  
#то есть введенное число должно быть целым и не больше N  
###insert()- ставит элемент списка на указанную позицию  
###append()- добавляет элемент в конец списка
```

Рисунок 31 – Текст программы для примера 12

Схема работы программы представлена на рисунке 32.

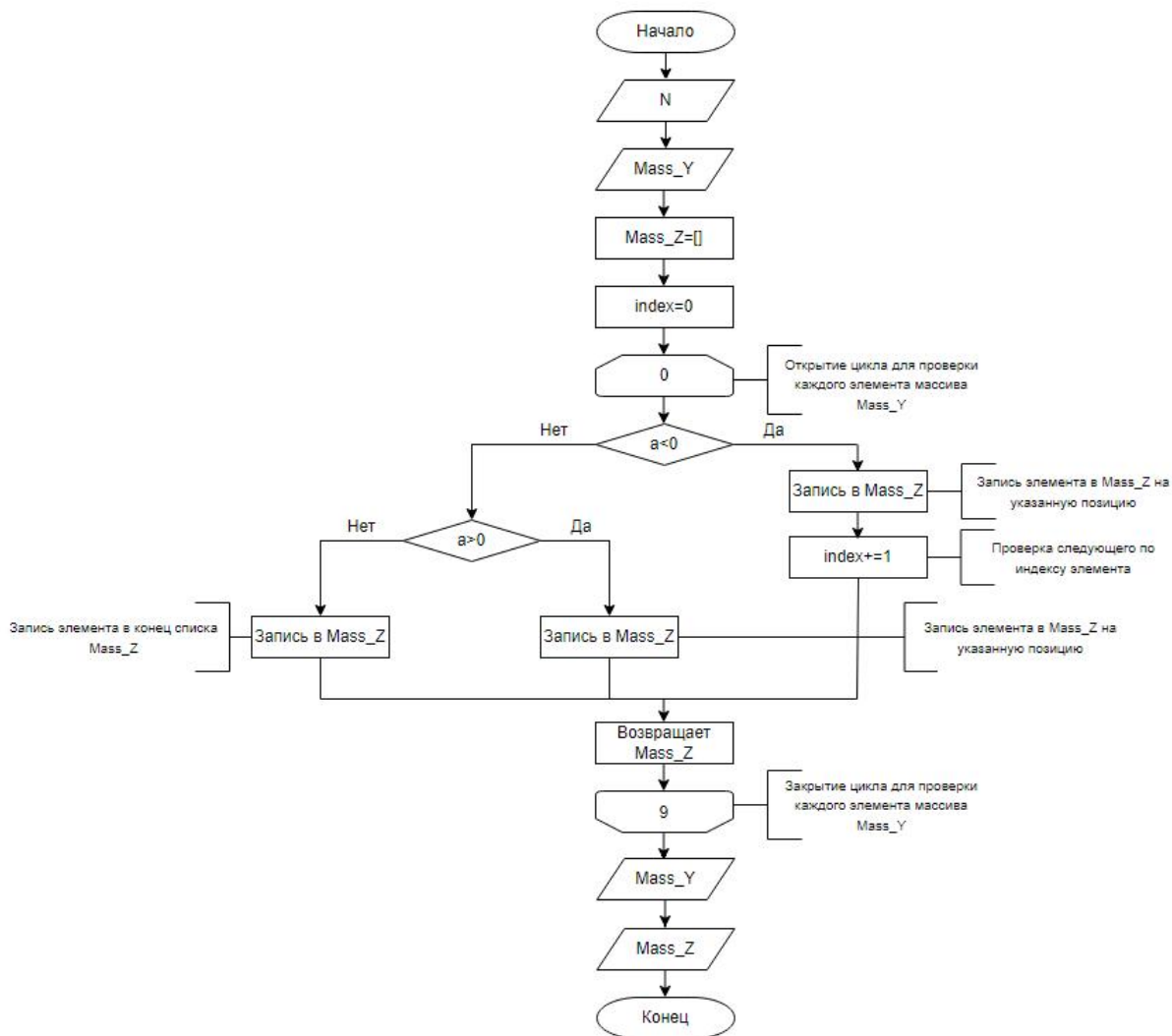


Рисунок 32 – Схема работы программы для примера 12

10.4. Оборудование и программное обеспечение

1. ПЭВМ IBM PC.
2. Операционная система Windows.
3. Интегрированная среда IDLE Python.

10.5 Задание на работу

1. Изучить теоретические положения лабораторной работы.
2. Набрать и отладить программу в соответствии с примером 12.
3. Получить результаты работы программы.
4. Показать результаты работы программы преподавателю.
5. Ответить на контрольные вопросы.
6. Оформить отчет.

10.6. Контрольные вопросы

1. Как реализовать запись из одного массива в другой отрицательных элементов массива?
2. Как реализовать запись из одного массива в другой положительных элементов массива?
3. Как реализовать запись из одного массива в другой нулевых элементов массива?

4. Как присвоить элементу, записываемому в новый массив, нового индекса?

10.7. Правила оформления отчета

Отчет должен содержать:

1. Номер работы, цель и название.
2. Сформулированные примеры, для которых составляются программы.
3. Схемы работы программ, изображенные в соответствии с ГОСТ 19.701-90.
4. Тексты введенных и отлаженных программ (фото текстов программ).
5. Результаты работы программ (фото результатов работы программ).
6. Описание выполненных в лабораторной работе действий (коротко как набирались и отлаживались программы).
7. Вывод по работе.

Практическая работа №11
Основы алгоритмизации и программирования.
Обработка массивов (часть 3) на языке Python
(в среде IDLE Python)

11.1. Цель работы

Освоить основные приемы обработки массивов на языке Python.

11.2. Теоретические положения

Часто при работе с массивами требуется знание математических основ работы с числами. Именно эти знания позволяют определить те или иные свойства элементов массивов.

11.3. Задания для составления схем и программ

Пример 13: Массив целых чисел C состоит из N элементов. Найти сумму простых чисел, входящих в него.

Простое число – это число, которое имеет только два делителя – 1 и само число (делится только на единицу и на само себя).

Идея написания программы заключается в следующем:

Сначала сумма равна 0 (вводится переменная, изначально принимающая значение 0). Далее последовательно перебираются все элементы массива, если очередной элемент простой, то он прибавляется к сумме. Сумма каждый раз возрастает на величину простого элемента (от первоначального нулевого значения до суммы всех простых элементов в представленном массиве).

Таким образом, сложность написания программы заключается только в определении простого числа. Для этого можно использовать операции:

- дивергенция – получение целой части от деления;
- модуляция – получение остатка от деления.

Дивергенция (в Python – `//`) работает следующим образом: $5:3=1$, так как целых «троек» в числе «пять» только одна.

Модуляция (в Python – `%`) работает следующим образом: $5:3=2$, так как остаток от целочисленного деления числа «пять» на число «три» это число «два».

Модуляцию и дивергенцию совместно можно записать так: $5:3=1$ остаток 2.

Текст программы для примера 13 представлен на рисунке 33.

Примечание 1: Текст после знака `#` – это комментарии, которые в тесте программы можно не указывать.

```

import math #подключение библиотеки для вычисления математических функций
def Sumprost(n):
    for i in range(2, int(math.sqrt(n) + 1)):
        if n % i == 0: #проверяет, чтобы при делении числа получался 0
            return False #возвращает False
    return n > 1 or n==1 #возвращает значение больше или равное единице

print('Введите количество элементов массива: ') #вывод на экран
N=int(input()) #ввод целочисленного числа элементов массива с клавиатуры

print('Введите массив C через пробел: ') #вывод на экран
Mass_C=[] #инициализация массива
Mass_C=list(map(int, input().split()[:N]))
print('Массив C: ') #вывод на экран
print(Mass_C) #вывод на экран введенного массива

Summa = sum(filter(Sumprost, Mass_C)) #суммирует отфильтрованный список
                                     #элементов простых чисел массива C
print('Сумма простых чисел массива = ', Summa) #вывод на экран суммы простых чисел
#range(2, int(math.sqrt(n) + 1)) - определяет диапазон от 2 до корня n+1
#list(map(int, input().split()[:N])) - list-создает список,
#map-передает каждому введенному элементу функцию,
#то есть введенное число должно быть целым и не больше N

```

Рисунок 33 – Текст программы для примера 13

В указанном коде программы для подсчёта суммы простых чисел используется встроенный математический модуль `sum`. Открывается функция для просмотра всех введенных элементов массива. Далее открывается цикл для проверки каждого введенного числа массива. Начиная с числа два (это первый возможный делитель после 1) до значения $\sqrt{n+1}$ происходит поиск делителей текущего числа n , а именно, если остаток от деления текущего числа n равен 0, значит число имеет хотя бы один делитель, отличный от 1, в этом случае возвращается логическое значение «ложь», так как число, имеющее делители, отличные от 1, не является простым.

Схема работы программы представлена на рисунке 34.

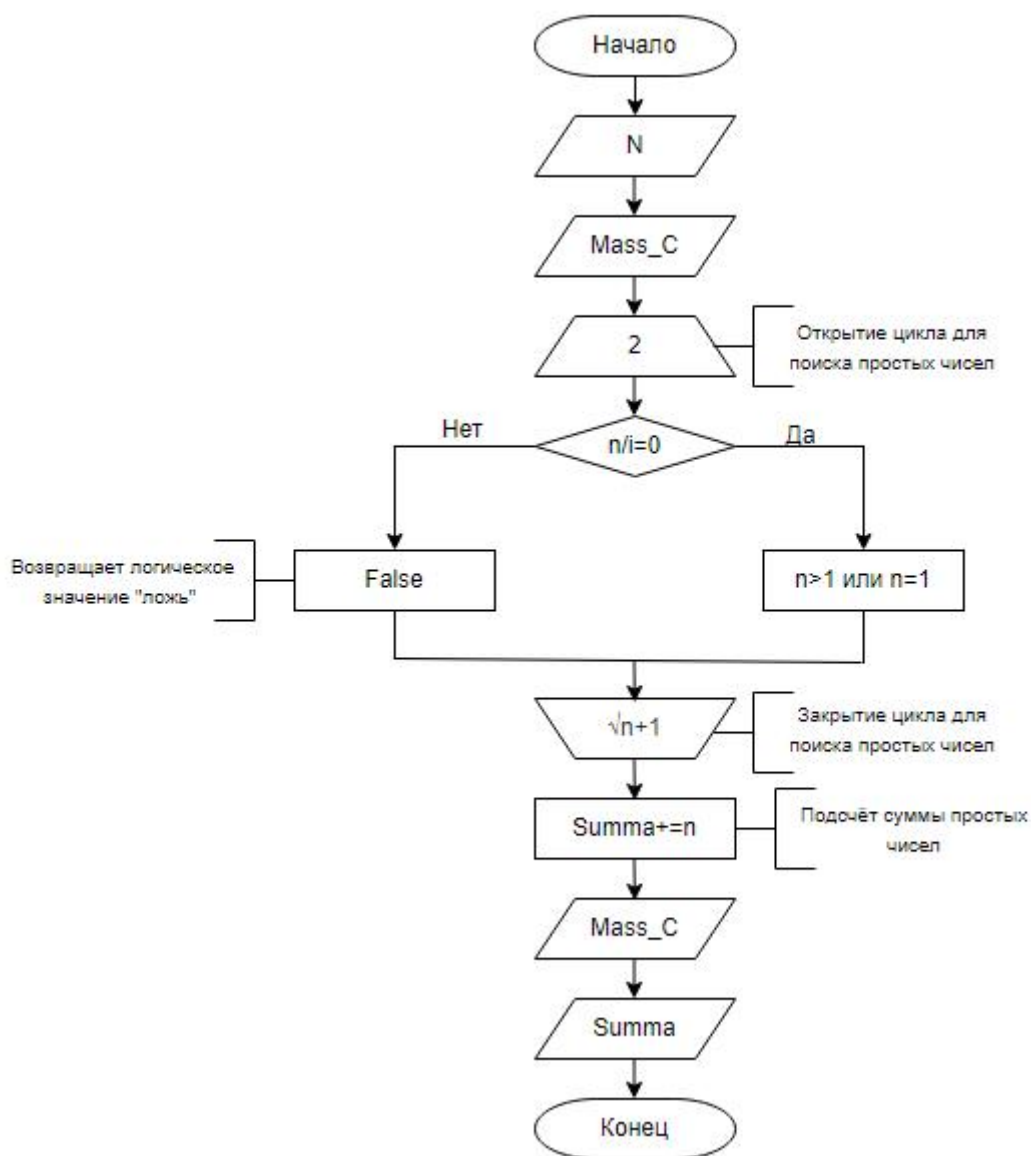


Рисунок 34 – Схема работы программы для примера 13

Рассмотрим, например, следующий массив чисел: 6, 7, 8, 3, 5 (предполагается, что этот линейный массив был загружен с клавиатуры).

Берем первый элемент, это число 6. Далее начиная с числа 2 до числа $\sqrt{6}+1$ осуществляется проверка первого элемента массива – числа 6.

Так как рассматриваемое условие $6 \% 2 = 0$ выполняется, то возвращается логическое значение «ложь», так как 6 это не простое число (оно делится на 2,3, помимо того, что делится на 1 и на само себя). После этого осуществляется выход из цикла (от 2 до $\sqrt{6}+1$).

Далее действия повторяются в цикле от 2 до $\sqrt{n}+1$.

Входим второй раз в цикл от 2 до $\sqrt{n}+1$, то есть берем второй элемент: берем второй элемент – число 7. Далее открывается еще один цикл рассматривается условие – $7 \% 2 \neq 0$ ($7 \% 2 = 1$), тогда логическое условие будет «истинной», далее $7 \% 3 \neq 0$ ($7 \% 3 = 1$), тогда логическое условие снова не меняется, оставаясь «истинным». Так как логическое условие «истинна», следовательно, $Summa = 0 + 7 = 7$.

Далее действия повторяются для всех элементов массива:

- для третьего элемента массива – $Summa=7$, берем элемент, равный 8, от 2 до $\sqrt{8+1}$ проверяем его на делители, $8 \% 2=0$, логическое условие будет «ложь». Осуществляется досрочный выход из цикла, так как число 8 не является простым (у него обнаружен первый делитель, отличный от 1, это число 2). При этом значение переменной $Summa$ не меняется: $Summa=7$;

- для четвертого элемента массива – $Summa=7$, берем элемент, равный 3, от 2 до $\sqrt{3+1}$ проверяем его на делители, $3 \% 2 \neq 0$ ($3 \% 2=1$), логическое условие будет «истина». Далее $3 \% 1 \neq 0$ ($3 \% 1=3$), логическое условие останется «истинным». Так как логическое условие «истинно», то $Summa=7+3=10$;

- для пятого элемента массива – $Summa=10$, берем элемент, равный 5, от 2 до $\sqrt{5+1}$ проверяем его на делители, $5 \% 2 \neq 0$ ($5 \% 2=1$), логическое условие имеет значение «истина». Далее $5 \% 2 \neq 0$ ($5 \% 2=1$), переменная логическое условие имеет значение «истина». Так как логическое условие «истина», то $Summa=10+5=15$.

Так как все элементы массива пересмотрены (достигнуто число $N=5$), то осуществляется выход из цикла.

Результат работы программы: Сумма простых чисел массива $Summa=15$.

11.4. Оборудование и программное обеспечение

1. ПЭВМ IBM PC.
2. Операционная система Windows.
3. Интегрированная среда IDLE Python.

11.5 Задание на работу

1. Изучить теоретические положения лабораторной работы.
2. Набрать и отладить программу в соответствии с примером 13.
3. Получить результаты работы программы.
4. Показать результаты работы программы преподавателю.
5. Ответить на контрольные вопросы.
6. Оформить отчет.

11.6. Контрольные вопросы

1. Как найти сумму элементов (любого свойства) массива?
2. Как работает операция дивергенция ($//$)?
3. Как работает операция модуляция ($\%$)?
4. Как осуществить досрочный выход из цикла?

11.7. Правила оформления отчета

Отчет должен содержать:

1. Номер работы, цель и название.
2. Сформулированные примеры, для которых составляются программы.
3. Схемы работы программ, изображенные в соответствии с ГОСТ 19.701-90.
4. Тексты введенных и отлаженных программ (фото текстов программ).
5. Результаты работы программ (фото результатов работы программ).
6. Описание выполненных в лабораторной работе действий (коротко как набирались и отлаживались программы).
7. Вывод по работе.

Практическая работа №12
Основы алгоритмизации и программирования.
Разработка программы на языке Python
(в среде IDLE Python)

12.1. Цель работы

Освоить основные приемы алгоритмизации и самостоятельного составления программ на языке Python.

12.2. Теоретические положения

Необходимо изучить вышеизложенную теорию и проделать вышеизложенные работы.

Примечание: задания сформулированы таким образом, что схема работы составляемой программы будет содержать только ветвление (выбор).

12.3. Задания для составления схем и программ

Вариант 1

Задана цена товара C , количество проданного товара K в день, номинал заработной платы продавца в день N . Рассчитать заработную плату продавца в день Z . Если $K \leq 10$, то заработная плата равна номиналу $Z = N$. Если $10 < K \leq 50$, то к номиналу заработной платы прибавляется двадцать процентов от суммы дневной выручки. Если $K > 50$, то к номиналу заработной платы прибавляется тридцать процентов от суммы дневной выручки.

Вариант 2

Заданы длины четырех сторон a основания правильной четырехугольной пирамиды, апофема f , высота пирамиды h . Вычислить боковую поверхность пирамиды $S_{\text{бок}}$ и ее объем V . Если $S_{\text{бок}} > 40$, то вычислить площадь одной боковой грани пирамиды $S_{\text{бок гр}}$.

Вариант 3

Заданы значения расстояния S и времени t скорости движения транспортного средства. Вычислить значения скорости движения транспортного средства V и расход бензина R . Если $V \leq 90$ и $V \geq 120$ км/ч, то $R = 8 * \frac{S}{100}$. Если $90 < V < 120$, то $R = 6 * \frac{S}{100}$.

Вариант 4

Заданы размеры сторон большего a и меньшего b оснований правильной усеченной пирамиды, ее апофема f и высота h . Вычислить боковую поверхность $S_{\text{бок}}$ правильной усеченной пирамиды и ее объем V . Если $V < 40$, то вычислить сумму площадей оснований пирамиды $S_{\text{осн}}$.

Вариант 5

Заданы размеры сторон a, b, c некоторой детали в виде параллелепипеда, количество штук деталей K , размеры сторон тары для упаковки деталей, имеющей также форму параллелепипеда A, B, C . Вычислить объем детали v , объем тары V , необходимое количество тары N для упаковки деталей количеством K штук. Если $N > 10$, то вычислить отношение объема детали к объему тары $\frac{v}{V}$.

Вариант 6

Заданы радиус основания круглого конуса r , и длина ее образующей l и высота конуса h . Вычислить боковую поверхность круглого конуса $S_{\text{бок}}$ и его объем V . Если $V < 20$, то вычислить длину окружности основания конуса $C_{\text{осн}}$.

Вариант 7

Вычислить значение функции $y=f(x)$. Если $x \leq 5$, то $y=x^2+3x-4$. Если $5 < x \leq 10$, то $y=\ln x$. Если $x > 10$, то $y=\frac{x^3}{3}$.

Вариант 8

Заданы радиус шара R и высота шарового сегмента n . Вычислить кривую поверхность шарового сегмента $S_{\text{шар сег}}$ и его объем V . Если $V < 20$, то вычислить отношение радиуса шара к высоте шарового сегмента $\frac{R}{n}$.

Вариант 9

Заданы массы некоторых двух тела m_1, m_2 . Тела движутся и проходят за время t следующие расстояния S_1 и S_2 соответственно. Вычислить импульсы тел P_1 и P_2 . Если $P_1 < P_2$, вычислить разность этих импульсов. Если $P_1 > P_2$, вычислить сумму этих импульсов. Если $P_1 = P_2$, вычислить произведение импульсов.

Вариант 10

Заданы поверхность шарового сегмента $S_{\text{шар сег}}$ и поверхность конуса $S_{\text{кон}}$, вместе образующие шаровый сектор. Заданы радиус шара R и высота шарового сегмента n , принадлежащая шаровому сектору. Вычислить поверхность шарового сектора $S_{\text{шар сект}}$ и его объем V . Если $V < 50$, то вычислить отношение поверхностей шарового сегмента к поверхности конуса $\frac{S_{\text{шар сег}}}{S_{\text{кон}}}$.

Вариант 11

Заданы первый член арифметической прогрессии a , разность прогрессии d и количество членов прогрессии k . Вычислить $k-1$ и k -ые члены арифметической прогрессии и сумму первых n членов S_n . Если $S_n \leq 25$, то вычислить процентное отношение первого и последнего членов арифметической прогрессии $\frac{a}{k} \cdot 100\%$.

Вариант 12

Заданы радиусы внутренней и наружной поверхностей полого шара $R_{\text{вн}}, R_{\text{нар}}$. Вычислить объем полого шара V и его полную поверхность $S_{\text{полн пов}}$. Если $S_{\text{полн пов}} < 40$, то вычислить толщину стенки шара t .

Вариант 13

Задан ежедневный доход индивидуального предпринимателя в течение месяца (двадцать шесть рабочих дней) $d_1, d_2, d_3, \dots, d_{25}, d_{26}$. Рассчитать месячный доход индивидуального предпринимателя $D = \sum_{i=1}^{i=26} d_i$. Если $D \geq 15\,000\,000$, то выдать сообщение о том, что «необходимо перейти к полной системе налогообложения». Если $D < 15\,000\,000$, то заданы базовая доходность индивидуального предпринимателя $БД=1800$, арендуемая индивидуальным предпринимателем площадь помещения F , корректирующий коэффициент $K_1=1,096$, поправочные коэффициенты $K_2=1$; $K_3=0,7$; $K_4=0,97$, процент налогообложения для

индивидуального предпринимателя $n=15\%$. Рассчитать сумму S , с которой будет заплачен налог $S = БД * F * K_1 * K_2 * K_3 * K_4 * 3$. Рассчитать сумму единого вмененного налога $N = \frac{S * n}{100}$.

Вариант 14

Задано значение синуса угла альфа $\sin \alpha$. Вычислить значение косинуса $\cos \alpha$, тангенса $\operatorname{tg} \alpha$ и котангенса $\operatorname{ctg} \alpha$, значение $\sin 2\alpha$. Если угол α острый, то вычислить $\cos 2\alpha$.

Вариант 15

Заданы при производстве некоторого изделия затраты на материалы M , электроэнергию $\mathcal{E}$, инструмент I , заработную плату рабочих Z . Вычислить себестоимость изделия Π_c . Заданы затраты на транспортировку изделия T . Вычислить стоимость изделия в магазине Π_m . Если $T \leq 1000$, то цена изделия в магазине возрастает на 30% по сравнению с себестоимостью. Если $1000 < T \leq 1500$, то цена в магазине возрастает на 40% по сравнению с себестоимостью. Если $T > 1500$, то цена в магазине возрастает на 50% по сравнению с себестоимостью изделия.

Вариант 16

Задан тангенс половины угла альфа $\operatorname{tg} \frac{\alpha}{2}$. Если $\alpha \leq 0,5$ рад, то вычислить значения синуса и косинуса полного угла α . Если $\alpha > 0,5$ рад, то вычислить значение тангенса полного угла α .

Вариант 17

Заданы значения расстояния S и времени t скорости движения транспортного средства. Вычислить значения скорости движения транспортного средства V и расход бензина R . Если $V \leq 90$ и $V \geq 120$ км/ч, то $R = 8 * \frac{S}{100}$. Если $90 < V < 120$, то $R = 6 * \frac{S}{100}$.

Вариант 18

Заданы поверхность $S_{\text{пов цилиндра}}$ и объем $V_{\text{цилиндра}}$ цилиндра, описанного вокруг шара. Вычислить поверхность $S_{\text{пов шара}}$ и объем шара $V_{\text{шара}}$. Если $S_{\text{пов шара}} \geq 120$, то вычислить отношение объема шара к его поверхности $\frac{V_{\text{шара}}}{S_{\text{пов шара}}}$.

Вариант 19

Заданы размеры сторон a, b, c некоторой детали в виде параллелепипеда, количество штук деталей K , размеры сторон тары для упаковки деталей, имеющей также форму параллелепипеда A, B, C . Вычислить объем детали v , объем тары V , необходимое количество тары N для упаковки деталей количеством K штук. Если $N > 10$, то вычислить отношение объема детали к объему тары $\frac{v}{V}$.

Вариант 20

Задано значение тангенса угла бета $\operatorname{tg} \beta$. Вычислить значение котангенса $\operatorname{ctg} \beta$, косинуса $\cos \beta$ и синуса $\sin \beta$, значение $\cos 2\beta$. Если угол β тупой, то вычислить $\sin 2\beta$.

Вариант 21

Задан объем партии изделий N , количество бракованных изделий в этой партии B , допустимое количество бракованных изделий в партии, выраженное в процентах $D_{\text{пр}}$. Вычислить проценты бракованных $B_{\text{пр}}$ и годных $G_{\text{пр}}$ изделий в партии. Если $B_{\text{пр}} \leq D_{\text{пр}}$, то вычислить количество годных изделий в партии G .

Вариант 22

Заданы длины трех сторон треугольника a, b, c . Вычислить периметр p и площадь треугольника S . Если $10 \leq S \leq 100$, вычислить отношение квадрата площади треугольника к его периметру $\frac{S}{p}$.

Вариант 23

Задана масса m груза, лежащего на плоскости. Сечение груза, параллельное плоскости, имеет форму прямоугольника. Заданы размеры сечения a, b . Определить давление груза на плоскость P . Задано допустимое давление на плоскость $P_{\text{доп}}$. Если $P \leq P_{\text{доп}}$, выдать сообщение о том, что «плоскость выдержит груз». Если $P > P_{\text{доп}}$, выдать сообщение о том, что «плоскость груза не выдержит». Если $P > P_{\text{доп}}$, определить разницу между фактическим P и допустимым $P_{\text{доп}}$ давлениями.

Вариант 24

Заданы длины двух катетов прямоугольного треугольника a, b . Вычислить длину гипотенузы c и величины двух углов треугольника α, β . Если $\alpha > 50^\circ$, то вычислить площадь треугольника. Если $\alpha < 50^\circ$, то вычислить сумму длин его сторон.

Вариант 25

Задан вид квадратного уравнения $ax^2 + bx + c = 0$, коэффициенты a, b, c . Найти корни уравнения x_1, x_2 .

Вариант 26

Задана масса груза, подвешенного на канате m и предельная сила натяжения каната $F_{\text{пр}}$. Определить силу тяжести груза, подвешенного на канате F . Если $F \leq F_{\text{пр}}$, то выдать сообщение о том, что «канат выдержит груз». Если $F > F_{\text{пр}}$, выдать сообщение, что «канат оборвется». Подсчитать запас прочности каната как отношение предельной силы каната к силе его натяжения $\frac{F_{\text{пр}}}{F}$.

Вариант 27

Задан диаметр шара D . Вычислить объем V , площадь основания $S_{\text{осн}}$, боковую поверхность $S_{\text{бок}}$ и полную поверхность полушария $S_{\text{полн пов}}$. Если $20 < S_{\text{осн}} < 100$, то вычислить радиус шара R .

Вариант 28

Заданы масса тела m , имеющего форму шара, и его диаметр D . Вычислить плотность материала тела $P_{\text{т}}$. Задана плотность жидкости $P_{\text{ж}}$, куда будет опущено тело. Если $P_{\text{т}} < P_{\text{ж}}$, вывести сообщение «тело будет плавать». Если $P_{\text{т}} > P_{\text{ж}}$, вывести сообщение «тело утонет». Если $P_{\text{т}} < P_{\text{ж}}$, вычислить процентное отношение плотностей тела и жидкости $\frac{P_{\text{т}}}{P_{\text{ж}}} 100\%$.

Вариант 29

Заданы радиус шара R и высота шарового слоя m , радиусы оснований шарового слоя r_1, r_2 . Вычислить кривую поверхность шарового слоя $S_{\text{шар сл}}$ и его объем V . Если $S_{\text{шар сл}} > 60$, то вычислить разность между радиусами оснований шарового слоя $r_1 - r_2$.

Вариант 30

Заданы первый член a , знаменатель q , количество членов k геометрической прогрессии. Вычислить $k-1$ и k -ые члены геометрической прогрессии и сумму первых n членов S_n . Если $S_n \geq 10$, то вычислить процентное отношение первого и последнего членов геометрической прогрессии $\frac{a}{k} 100\%$.

12.4. Оборудование и программное обеспечение

1. ПЭВМ IBM PC.
2. Операционная система Windows.
3. Интегрированная среда IDLE Python.

12.5 Задание на работу

1. Изучить теоретические положения лабораторной работы.
2. Получить вариант примера у преподавателя.
3. Составить схему работы программы в соответствии с полученным вариантом (по ГОСТ 19.701-90).
4. Составить на языке Python и отладить программу в соответствии с полученным вариантом.
5. Получить результаты работы программы.
6. Показать результаты работы программы преподавателю.
7. Ответить на контрольные вопросы.
8. Оформить отчет.

12.6. Контрольные вопросы

1. Какие структуры вы использовали для составления схемы работы программы в соответствии с полученным вариантом?
2. Какие операторы Вы использовали для составления программы на языке Python в среде IDLE Python?

12.7. Правила оформления отчета

Отчет должен содержать:

1. Номер работы, цель и название.
2. Сформулированные примеры, для которых составляются программы.
3. Схемы работы программ, изображенные в соответствии с ГОСТ 19.701-90.
4. Тексты введенных и отлаженных программ (фото текстов программ).
5. Результаты работы программ (фото результатов работы программ).
6. Описание выполненных в лабораторной работе действий (коротко как набирались и отлаживались программы).
7. Вывод по работе.

Практическая работа №13
Основы алгоритмизации и программирования.
Разработка программы на языке Python
(в среде IDLE Python) с использованием строгого цикла

13.1. Цель работы

Освоить основные приемы алгоритмизации и самостоятельного составления программ на языке Python с использованием строгого цикла.

13.2. Теоретические положения

Необходимо изучить вышеизложенную теорию и проделать вышеизложенные работы.

Примечание: задания сформулированы таким образом, что схема работы составляемой программы будет содержать цикл с параметром.

13.3. Задания для составления схем и программ

1. Вычислить $A = k! * k$ ($k! = 1 * 2 * 3 * \dots * k$, k -номер студента в списке группы).
2. Вычислить $B = k! / k$ ($k! = 1 * 2 * 3 * \dots * k$, k -номер студента в списке группы).
3. Вычислить $C = k! - k$ ($k! = 1 * 2 * 3 * \dots * k$, k -номер студента в списке группы).
4. Вычислить $D = k! + k$ ($k! = 1 * 2 * 3 * \dots * k$, k -номер студента в списке группы).
5. Вычислить $E = k! * k$ ($k! = 1 * 2 * 3 * \dots * k$, k -номер студента в списке группы).
6. Вычислить $F = a^k$ (k -номер студента в списке группы, a – номер старосты в списке группы).
7. Вычислить $G = a^{k-1}$ (a -номер студента в списке группы старосты, k – номер старосты в списке группы).
8. Вычислить $H = a^{k+k}$ (k -номер студента в списке группы, a – произвольное число в диапазоне от 1 до 12).
9. Вычислить $L = a^{k+1}$ (k -номер студента в списке группы, a – дата рождения студента (число месяца)).
10. Вычислить $M = a^{k*2}$ (a -номер студента в списке группы, k – номер дома, в котором проживает студент).

13.4. Оборудование и программное обеспечение

1. ПЭВМ IBM PC.
2. Операционная система Windows.
3. Интегрированная среда IDLE Python.

13.5 Задание на работу

1. Изучить теоретические положения лабораторной работы.
2. Получить вариант примера у преподавателя.
3. Составить схему работы программы в соответствии с полученным вариантом (по ГОСТ 19.701-90).
4. Составить на языке Python и отладить программу в соответствии с полученным вариантом.

5. Получить результаты работы программы.
6. Показать результаты работы программы преподавателю.
7. Ответить на контрольные вопросы.
8. Оформить отчет.

13.6. Контрольные вопросы

1. Какие типы данных Вы использовали для составления программы на языке Python в среде IDLE Python?
2. Какие структуры вы использовали для составления схемы работы программы в соответствии с полученным вариантом?
3. Какие операторы Вы использовали для составления программы на языке Python в среде IDLE Python?
4. Что составило тело цикла программы?
5. Какие ошибки были выявлены при компиляции?

13.7. Правила оформления отчета

Отчет должен содержать:

1. Номер работы, цель и название.
2. Сформулированные примеры, для которых составляются программы.
3. Схемы работы программ, изображенные в соответствии с ГОСТ 19.701-90.
4. Тексты введенных и отлаженных программ (фото текстов программ).
5. Результаты работы программ (фото результатов работы программ).
6. Описание выполненных в лабораторной работе действий (коротко как набирались и отлаживались программы).
7. Вывод по работе.

Практическая работа №14
Основы алгоритмизации и программирования.
Разработка программы на языке Python
(в среде IDLE Python) с использованием средств обработки массивов

14.1. Цель работы

Освоить основные приемы алгоритмизации и самостоятельного составления программ на языке Python с использованием средств обработки массивов.

14.2. Теоретические положения

Необходимо изучить вышеизложенную теорию и проделать вышеизложенные работы.

Примечание: задания сформулированы таким образом, что схема работы составляемой программы будет содержать различного рода циклы и условия.

14.3. Задания для составления схем и программ

Вариант 1.

а) Модернизировать программу для примера 9 так, чтобы в массив В записывались нечетные элементы массива А;

б) Модернизировать программу для примера 10 так, чтобы в массив Z из массива Y записывались сначала нулевые элементы, затем положительные, а потом отрицательные;

в) Модернизировать программу для примера 11 так, чтобы суммировались числа с делителем 3.

Вариант 2.

а) Модернизировать программу для примера 9 так, чтобы в массив В записывались элементы массива А, стоящие на нечетных позициях.

б) Модернизировать программу для примера 10 так, чтобы в массив Z из массива Y записывались сначала отрицательные элементы, затем нулевые, а потом положительные;

в) Модернизировать программу для примера 11 так, чтобы суммировались числа с делителем 5.

Вариант 3

а) Модернизировать программу для примера 9 так, чтобы в массив В записывались элементы массива А, стоящие на четных позициях.

б) Модернизировать программу для примера 10 так, чтобы в массив Z из массива Y записывались только положительные элемента массива.

в) Модернизировать программу для примера 11 так, чтобы суммировались числа с делителем 4.

Вариант 4

а) Модернизировать программу для примера 9 так, чтобы в массив В записывались элементы массива А, в обратном порядке от к-го до первого.

б) Модернизировать программу для примера 10 так, чтобы в массив Z из массива Y записывались только отрицательные элемента массива.

в) Модернизировать программу для примера 11 так, чтобы суммировались числа с делителем 6.

Вариант 5

а) Модернизировать программу для примера 9 так, чтобы в массив В записывались три первых элемента массива А.

б) Модернизировать программу для примера 10 так, чтобы в массив Z из массива Y записывались только целые элемента массива.

в) Модернизировать программу для примера 11 так, чтобы суммировались числа с делителем 7.

Вариант 6

а) Модернизировать программу для примера 9 так, чтобы в массив В записывались три последних элемента массива А.

б) Модернизировать программу для примера 10 так, чтобы в массив Z из массива Y записывались только нулевые элемента массива.

в) Модернизировать программу для примера 11 так, чтобы суммировались числа с делителем 8.

Вариант 7

а) Модернизировать программу для примера 9 так, чтобы в массив В записывались три первых элемента и один последний элемент массива А.

б) Модернизировать программу для примера 10 так, чтобы в массив Z из массива Y записывались подряд нулевые и положительные элемента массива.

в) Модернизировать программу для примера 11 так, чтобы суммировались числа с делителем 2 или 5.

Вариант 8.

а) Модернизировать программу для примера 9 так, чтобы в массив В записывался второй и предпоследний элемент массива А.

б) Модернизировать программу для примера 10 так, чтобы в массив Z из массива Y записывались подряд отрицательные и положительные элемента массива.

в) Модернизировать программу для примера 11 так, чтобы суммировались числа с делителем 3 или 6.

Вариант 9.

а) Модернизировать программу для примера 9 так, чтобы в массив В записывался пятый и седьмой элемент массива А.

б) Модернизировать программу для примера 10 так, чтобы в массив Z из массива Y записывались подряд отрицательные и нулевые элемента массива.

в) Модернизировать программу для примера 11 так, чтобы суммировались числа с делителем 4 или 9.

Вариант 10.

а) Модернизировать программу для примера 9 так, чтобы в массив В записывались третий и шестой элемент массива А.

б) Модернизировать программу для примера 10 так, чтобы в массив Z из массива Y записывались подряд положительные элементы массива, за исключением первого положительного элемента.

в) Модернизировать программу для примера 11 так, чтобы суммировались числа с делителем 5 или 6.

14.4. Оборудование и программное обеспечение

1. ПЭВМ IBM PC.
2. Операционная система Windows.
3. Интегрированная среда IDLE Python.

14.5 Задание на работу

1. Изучить теоретические положения лабораторной работы.
2. Получить вариант примера у преподавателя.
3. Составить схему работы программы в соответствии с полученным вариантом (по ГОСТ 19.701-90).
4. Составить на языке Python и отладить программу в соответствии с полученным вариантом.
5. Получить результаты работы программы.
6. Показать результаты работы программы преподавателю.
7. Ответить на контрольные вопросы.
8. Оформить отчет.

14.6. Контрольные вопросы

1. Что необходимо изменить в программе примеров 11,12,13 для решения Вашего варианта текущей лабораторной работы?
2. Как изменится схема алгоритма?

14.7. Правила оформления отчета

Отчет должен содержать:

1. Номер работы, цель и название.
2. Сформулированные примеры, для которых составляются программы.
3. Схемы работы программ, изображенные в соответствии с ГОСТ 19.701-90.
4. Тексты введенных и отлаженных программ (фото текстов программ).
5. Результаты работы программ (фото результатов работы программ).
6. Описание выполненных в лабораторной работе действий (коротко как набирались и отлаживались программы).
7. Вывод по работе.