Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5
дисциплины «Программирование на Python»

Выполнила:
Кубанова Ксения Олеговна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника», очная
форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р. А.

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: условные операторы и циклы в языке Python

Цель: приобретение навыков программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Освоить операторы языка Python версии 3.x if, while, for, break, continue, позволяющих реализовывать разветвляющие алгоритмы и алгоритмы циклической структуры.

Порядок выполнения

Пример 1.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import math
if __name__=='__main__':
    x=float(input("value of x? "))
    if x<=0:
        y=2*x*x+math.cos(x)
    elif x<5:
        y=x+1
    else:
        y=math.sin(x)-x*x
    print(f"y={y}")
```

Рисунок 1 – пример 1

```
value of x? 5
y=-25.95892427466314
```

Рисунок 2 – результат примера 1

Пример 2.

primer2.py > ...

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__=='__main__':
    n=int(input("month number: "))
    if n==1 or n==2 or n ==12:
        print("Winter")
    elif n==3 or n==4 or n==5:
        print("Spring")
    elif n==6 or n==7 or n==8:
        print("Summer")
    elif n==9 or n==10 or n==11:
        print("Autumn")
    else:
        print("Error!", file=sys.stderr)
        exit(1)
```

Рисунок 3 – пример 2

```
month number: 4
Spring
PS C:\Users\Сергей\Or
wsApps/python3.11.exe
month number: 13
Error!
```

Рисунок 4 – результат примера 2

Пример 3.

primer3.py > ...

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import math

if __name__=='__main__':
    n=int(input("value of n? "))
    x=float(input("value of x? "))
    s=0.0
    for k in range(1, n+1):
        a=math.log(k*x)/(k*k)
        s+=a
    print(f"s={s}")
```

Рисунок 5 – пример 3

```
value of n? 6
value of x? 3
s=2.1346049981654356
```

Рисунок 6 – результат примера 3

Пример 4.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import math
import sys

if __name__=='__main__':
    a=float(input("value of a? "))
    if a<0:
        print("illegal value of a", file=sys.stderr)
        exit(1)
    x,eps=1,1e-10
    while True:
        xp=x
        x=(x+a/x)/2
        if math.fabs(x-xp)<eps:
            break
    print(f"x={x}\nX={math.sqrt(a)}")
```

Рисунок 7 – пример 4

```
value of a? 4
x=2.0
X=2.0
PS C:\Users\Сергей\OneDrive\Рабочий стол\ДЗ\2 курс\пргрм на пит\лаб2.2\LABpyth2.2\prog> & C:/Users/Сергей/AppData/Local
wsApps/python3.11.exe "c:/Users/Сергей/OneDrive/Рабочий стол/ДЗ/2 курс/пргрм на пит/лаб2.2/LABpyth2.2/prog/primer4.py"
value of a? 7
x=2.6457513110645907
X=2.6457513110645907
PS C:\Users\Сергей\OneDrive\Рабочий стол\ДЗ\2 курс\пргрм на пит\лаб2.2\LABpyth2.2\prog>
```

Рисунок 8 – результат примера 4

UML-диаграмма

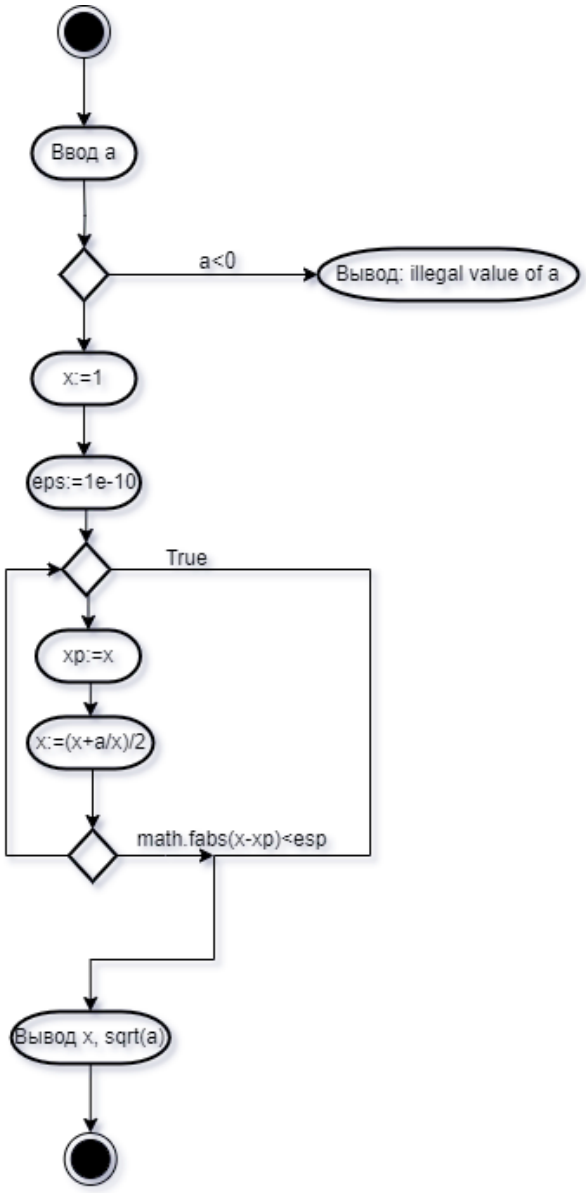

Рисунок 12 – UML-диаграмма примера 4

Пример 5.

primer5.py > ...

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import math
import sys

#Постоянная Эйлера
EULER=0.5772156649015328606
#Точность вычислений
EPS=1e-10

if __name__=='__main__':
    x=float(input("value of x? "))
    if x==0:
        print("illegal value of x", file=sys.stderr)
        exit(1)
    a=x
    s,k=a,1
    #Найти сумму членов ряда
    while math.fabs(a)>EPS:
        a*=x*k/(k+1)**2
        s+=a
        k+=1
    #Вывести значение функции
    print(f"Ei({x})={EULER + math.log(math.fabs(x))+s}")
```

Рисунок 10 – пример 5

```
3.11.exe  C:/Users/Сергей/OneD
value of x? 5
Ei(5.0)=40.185275355797 94
```

Рисунок 11 – результат примера 5

UML-диаграмма

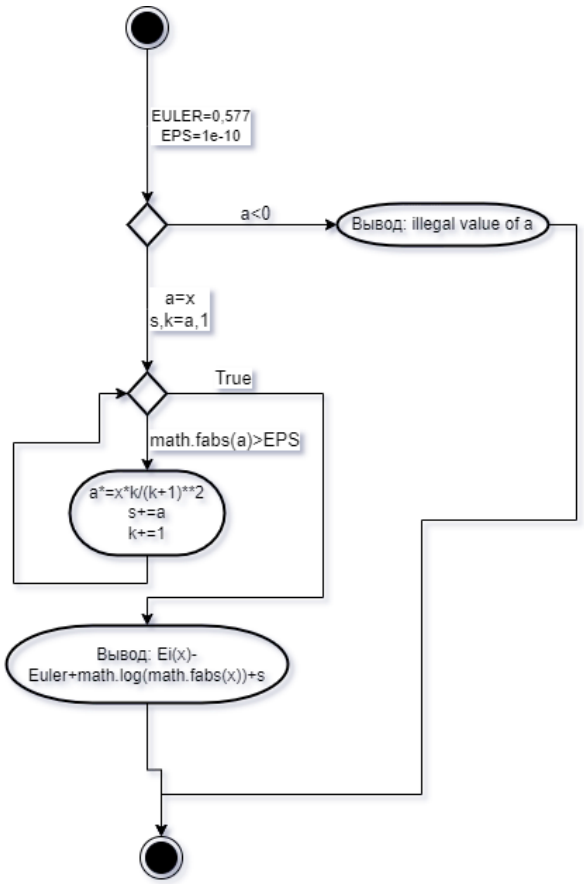

Рисунок 12 – UML-диаграмма примера 5

Индивидуальное задание 1.

6. Дано целое число c такое, что $|c| < 9$. Вывести это число в словесной форме, учитывая его знак.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import sys
if __name__=='__main__':
    c=int(input("value c? "))
    if c<0 or c>9:
        print("Error!", file=sys.stderr)
        exit(1)
    elif c==1:
        print("one")
    elif c==2:
        print("two")
    elif c==3:
        print("three")
    elif c==4:
        print("four")
    elif c==5:
        print("five")
    elif c==6:
        print("six")
    elif c==7:
        print("seven")
    elif c==8:
        print("eight")
    else:
        print("nine")

```

Рисунок 13 – код индивидуального задания 1

```
value c? -1
Error!
PS C:\Users\Сергей\OneDr
3.11.exe "c:/Users/Cepre
value c? 11
Error!
PS C:\Users\Сергей\OneDr
3.11.exe "c:/Users/Cepre
value c? 7
seven
```

Рисунок 14 – результат индивидуального задания 1

UML-диаграмма

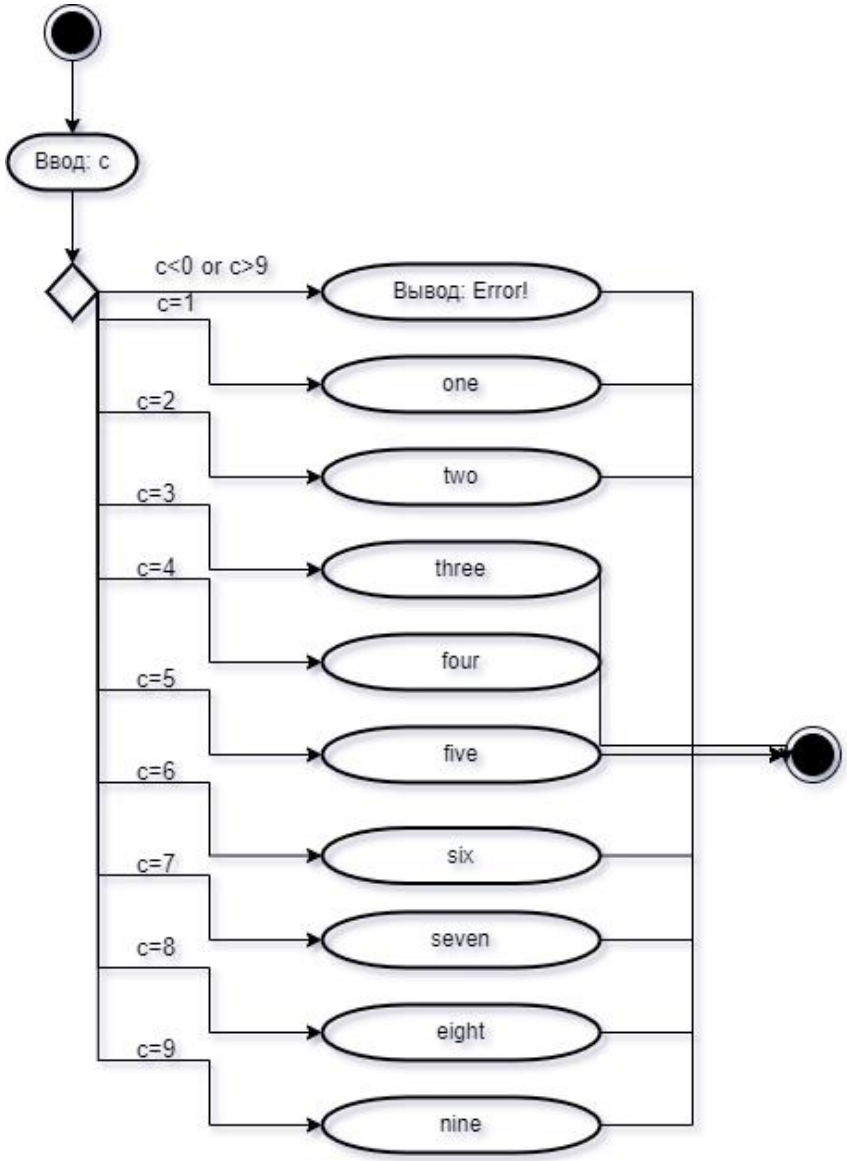

Рисунок 15 – UML-диаграмма индивидуального задания 1

Индивидуальное задание 2.

6. Решить квадратное неравенство $a\,x^2 + bx + c > 0$ ($a \neq 0$), где a, b и c - действительные числа.

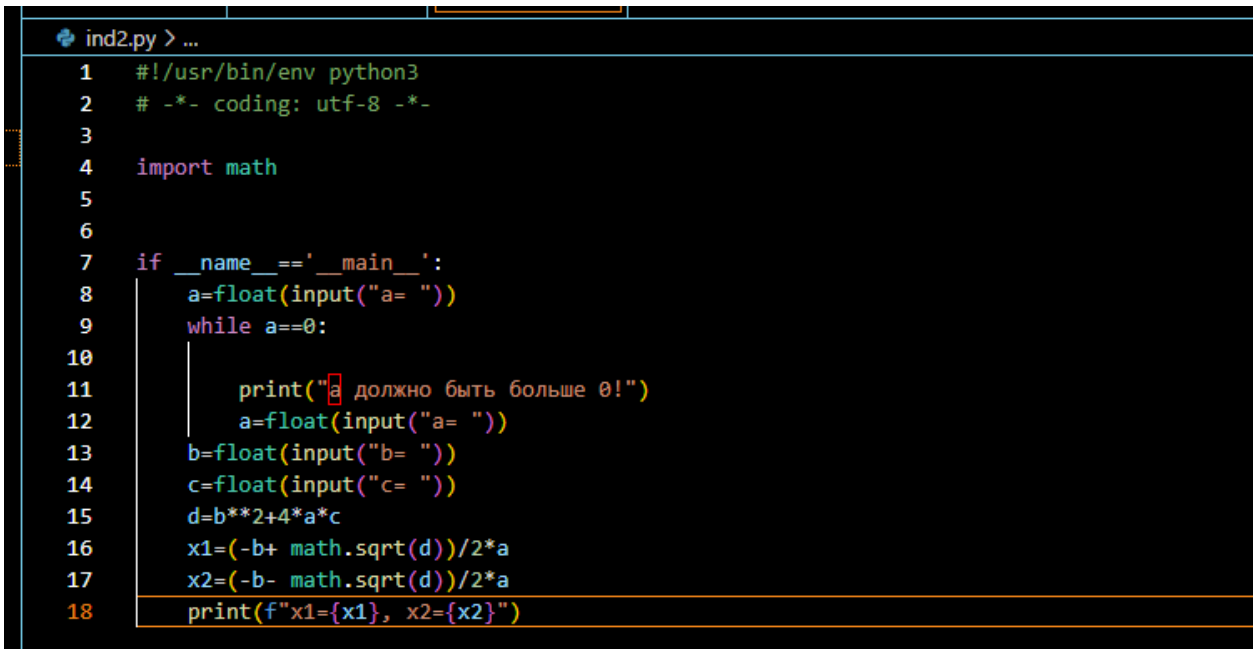

Рисунок 16 – код индивидуального задания 2

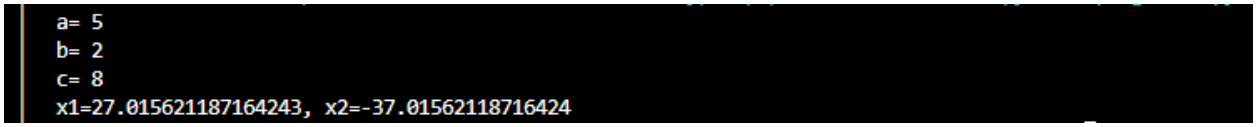

Рисунок 17 – результат индивидуального задания 1

UML-диаграмма

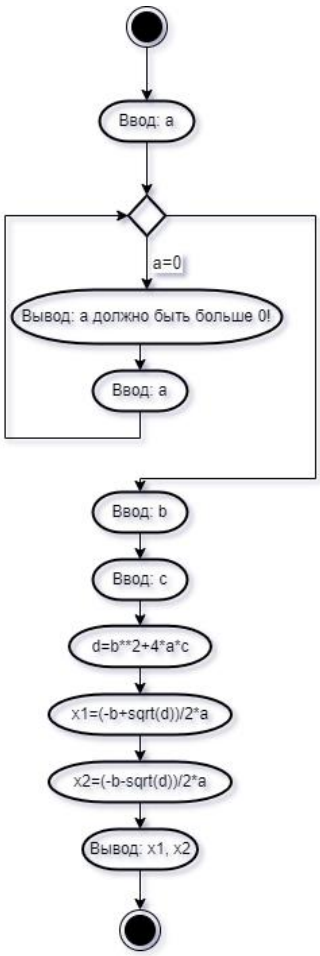

Рисунок 18 – UML-диаграмма индивидуального задания 2

Индивидуальное задание 3.

6. Напечатать таблицу соответствия между весом в фунтах и весом в кг для значений от 1 до a фунтов с шагом 1 фунт, если 1 фунт = 400 г.

ind3.py > ...

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__=='__main__':

 a=int(input("a= "))
 print("--------------")
 print("| funts | kg  |")
 print("--------------")
 for i in range (1, a, 1):
  f=i*400
  l=f/1000
  print("| {0}     | {1} |".format(i, l))
  print("--------------")
```

Рисунок 19 – код индивидуального задания 3

```
a= 8
-------------
| funts | kg  |
-------------
| 1     | 0.4 |
-------------
| 2     | 0.8 |
-------------
| 3     | 1.2 |
-------------
| 4     | 1.6 |
-------------
| 5     | 2.0 |
-------------
| 6     | 2.4 |
-------------
| 7     | 2.8 |
-------------
```

Рисунок 20 – результат индивидуального задания 3

UML-диаграмма

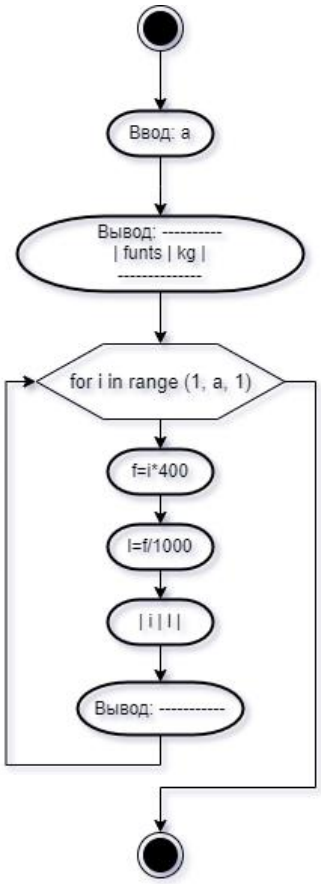

Рисунок 21 – UML-диаграмма индивидуального задания 3

Задание повышенной сложности.

6. Функция Бесселя первого рода $J_n(x)$, значение $n = 0, 1, 2, \ldots$ также должно вводиться с клавиатуры

$$J_n(x) = \left(\frac{x}{2}\right)^n \sum_{k=0}^{\infty} \frac{(-x^2/4)^k}{k!(k+n)!}.$$

Для написания программы функции Бесселя необходимо для начала вычислить рекуррентное соотношение. Для этого необходимо разделить следующий член ряда на текущий. Текущий член ряда:

$$a_к = \frac{(-\frac{x^2}{4})^k}{k!\,(k!+n)!}$$

Соответственно, следующий член ряда:

$$a_{k+1} = \frac{(-\frac{x^2}{4})^{k+1}}{(k+1)!\,\big((k+1)+n\big)!}$$

После разделения следующего члена ряда на текущий получается следующее рекуррентное соотношение:

$$a_{k+1} = \frac{-x^2/4}{(k+1)(k+1+n)} \cdot a_k$$

harmode.py > ...

```
1   #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3
4   import math
5   import sys
6
7   eps=1e-10
8
9   if __name__=='__main__':
10      n=int(input("n="))
11      if n==0:
12          print("Illegal value of n", file=sys.stderr)
13          exit(1)
14      x=int(input("x="))
15      if x==0:
16          print("Illegal value of x", file=sys.stderr)
17          exit(1)
18      a=x
19      s, k=a, 1
20      while math.fabs(a)<eps:
21          a*=((-x**2/4)/(k+1)*(k+1+n))
22          s+=a
23          k+=1
24      print(f"J {n}({x})={(x/2)**n*s}")
25
```

Рисунок 22 – код усложнённого задания

```
n=5
x=5
J 5(5)=488.28125
```

Рисунок 23 – результат усложнённого задания

UML-диаграмма

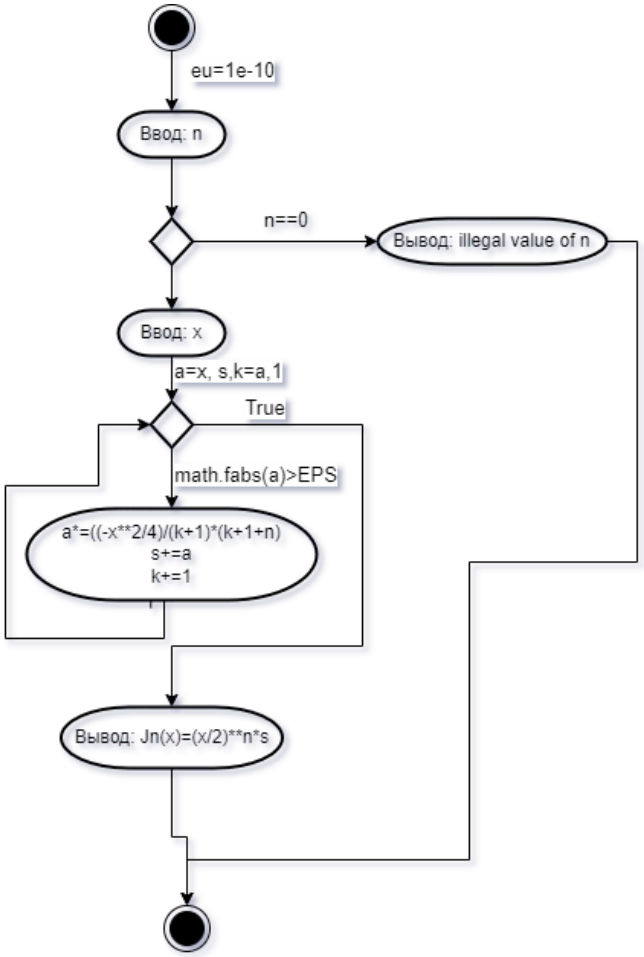

Рисунок 24 – UML-диаграмма усложнённого задания

Контрольные вопросы

1. Для чего нужны диаграммы деятельности UML?

Диаграммы деятельности UML используются для визуализации и моделирования последовательности действий и процессов в системе. Они позволяют описать, какие действия выполняются, какие решения принимаются и какие данные обрабатываются в рамках определенной деятельности или процесса. Диаграммы деятельности помогают лучше понять и описать логику работы системы, а также выявить потенциальные проблемы или улучшения в процессе выполнения задач.

2. Что такое состояние действия и состояние деятельности?

- **Состояние действия** в диаграммах деятельности UML представляет собой конкретное действие или операцию, которую выполняет объект или система в определенный момент времени. Состояние действия обычно представляется в виде прямоугольника с закругленными углами и содержит название действия.

- **Состояние деятельности** в диаграммах деятельности UML представляет собой группу связанных действий или операций, которые выполняются последовательно или параллельно. Состояние деятельности обычно представляется в виде прямоугольника с закругленными углами и содержит название деятельности.

3. Какие нотации существуют для обозначения переходов и ветвлений в диаграммах деятельности?

В диаграммах деятельности UML существуют следующие нотации для обозначения переходов и ветвлений:

- Стрелка с полным концом (->) обозначает переход от одного состояния к другому.

- Стрелка с пустым концом (--) обозначает переход к тому же состоянию.

- Ромб с надписью "да" или "нет" обозначает ветвление, где каждая ветвь представляет собой альтернативный путь выполнения.

- Стрелка с надписью "else" обозначает альтернативный путь выполнения в случае, если условие в ветвлении не выполняется.

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Алгоритм разветвляющейся структуры - это алгоритм, который включает в себя ветвления и условные операторы для принятия решений в зависимости от определенных условий. Один из примеров такого алгоритма - это условный оператор "if-else", который позволяет выполнить определенные действия, если условие истинно, и другие действия, если условие ложно.

5. Чем отличается разветвляющийся алгоритм от линейного?

Разветвляющийся алгоритм отличается от линейного алгоритма тем, что в разветвляющемся алгоритме присутствуют ветвления и условные операторы, которые позволяют принимать решения в зависимости от определенных условий. В результате выполнения разветвляющегося алгоритма может быть несколько возможных путей выполнения в зависимости от условий.

В линейном алгоритме, напротив, отсутствуют ветвления и условные операторы. Выполнение линейного алгоритма происходит последовательно, без возможности изменения пути выполнения в зависимости от условий.

6. Что такое условный оператор? Какие существуют его формы?

Условный оператор - это конструкция в программировании, которая позволяет выполнять определенные действия в зависимости от выполнения определенного условия. Условный оператор позволяет программе принимать решения и выбирать различные пути выполнения в зависимости от значения условия.

7. Какие операторы сравнения используются в Python?

В языке программирования Python используются следующие операторы сравнения:

- `==` (равно): проверяет, равны ли два значения.
- `!=` (не равно): проверяет, не равны ли два значения.
- `>` (больше): проверяет, является ли первое значение больше второго.
- `<` (меньше): проверяет, является ли первое значение меньше второго.
- `>=` (больше или равно): проверяет, является ли первое значение больше или равным второму.
- `<=` (меньше или равно): проверяет, является ли первое значение меньше или равным второму.

8. Что называется простым условием? Приведите примеры.

Простое условие - это условие, которое содержит один оператор сравнения и возвращает булево значение (True или False). Простое условие

проверяет отношение между двумя значениями и возвращает результат этой проверки.

9. Что такое составное условие? Приведите примеры.

Составное условие - это условие, которое содержит несколько простых условий, объединенных логическими операторами. Составное условие проверяет несколько условий одновременно и возвращает булево значение (True или False) в зависимости от результата проверки всех условий.

Примеры составных условий в Python:

- `x > 5 and y < 10` - проверяет, является ли значение переменной `x` больше 5 и значение переменной `y` меньше 10.

- `a == "hello" or b == "world"` - проверяет, равно ли значение переменной `a` строке "hello" или значение переменной `b`

10. Какие логические операторы допускаются при составлении сложных условий?

Логические операторы, допускаемые при составлении сложных условий, включают "и" (and), "или" (or) и "не" (not).

11. Может ли оператор ветвления содержать внутри себя другие ветвления?

Да, оператор ветвления может содержать внутри себя другие ветвления. Это называется вложенным оператором ветвления.

12. Какой алгоритм является алгоритмом циклической структуры?

Алгоритмом циклической структуры называется такой, который повторяется до истинного значения условия, повторяя одни и те же действия.

13. Типы циклов в языке Python.

Цикл while: выполняет блок кода, пока условие истинно.

Цикл for: выполняет блок кода для каждого элемента в последовательности (например, список, строка или диапазон чисел).

14. Назовите назначение и способы применения функции range .

Функция range в Python используется для создания последовательности чисел. Она принимает начальное значение, конечное значение и необязательный шаг. Например, range(0, 10, 2) создаст последовательность [0, 2, 4, 6, 8].

15. Как с помощью функции range организовать перебор значений от 15 до 0 с шагом 2?

```
for i in range(15, 0, -2):
    print(i)
```

16. Могул ли быть циклы вложенными?

Да, в Python можно использовать вложенные циклы. Это означает, что один цикл может находиться внутри другого цикла.

17. Как образуется бесконечный цикл и как выйти из него?

Бесконечный цикл образуется, когда условие цикла всегда остается истинным, и код внутри цикла выполняется бесконечно. Чтобы выйти из бесконечного цикла, можно использовать оператор break, который прерывает выполнение цикла и переходит к следующему участку кода за циклом.

18. Для чего нужен оператор break ?

Оператор break используется для прерывания выполнения цикла в Python. Когда оператор break достигается внутри цикла, выполнение цикла останавливается, и программа переходит к следующему участку кода после цикла.

19. Где употребляется оператор continue и для чего он используется?

Оператор continue используется в Python для пропуска текущей итерации цикла и перехода к следующей итерации. Он применяется внутри цикла и позволяет пропустить оставшуюся часть кода в текущей итерации и начать следующую итерацию.

20. Для чего нужны стандартные потоки stdout и stderr?

Стандартные потоки stdout (стандартный вывод) и stderr (стандартный поток ошибок) являются потоками вывода в операционной системе. stdout

используется для обычного вывода данных или результатов программы, а stderr используется для вывода сообщений об ошибках и предупреждений.

21. Как в Python организовать вывод в стандартный поток stderr?

Чтобы организовать вывод в стандартный поток stderr в Python, можно использовать модуль sys и его метод stderr.write().

22. Каково назначение функции exit ?

Функция exit в Python используется для выхода из программы. Она вызывает завершение программы и может принимать необязательный код возврата. Код возврата может быть использован для сообщения о результате выполнения программы другим программам или скриптам, которые вызывают ее.

Вывод: в ходе выполнения лабораторной работы были изучены типы и организация циклов, структура, её составляющие и организация ветвления в Python 3.x.