

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Основы кроссплатформенного программирования

Отчет по лабораторной работе №3

Выполнила студентка группы ИТС-б-о-
20-1 (2)

Маслова А.В. « » _____ 2021г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил доцент
Кафедры инфокоммуникаций Воронкин
Р.А.

(подпись)

г. Ставрополь, 2021

Лабораторная работа 3.

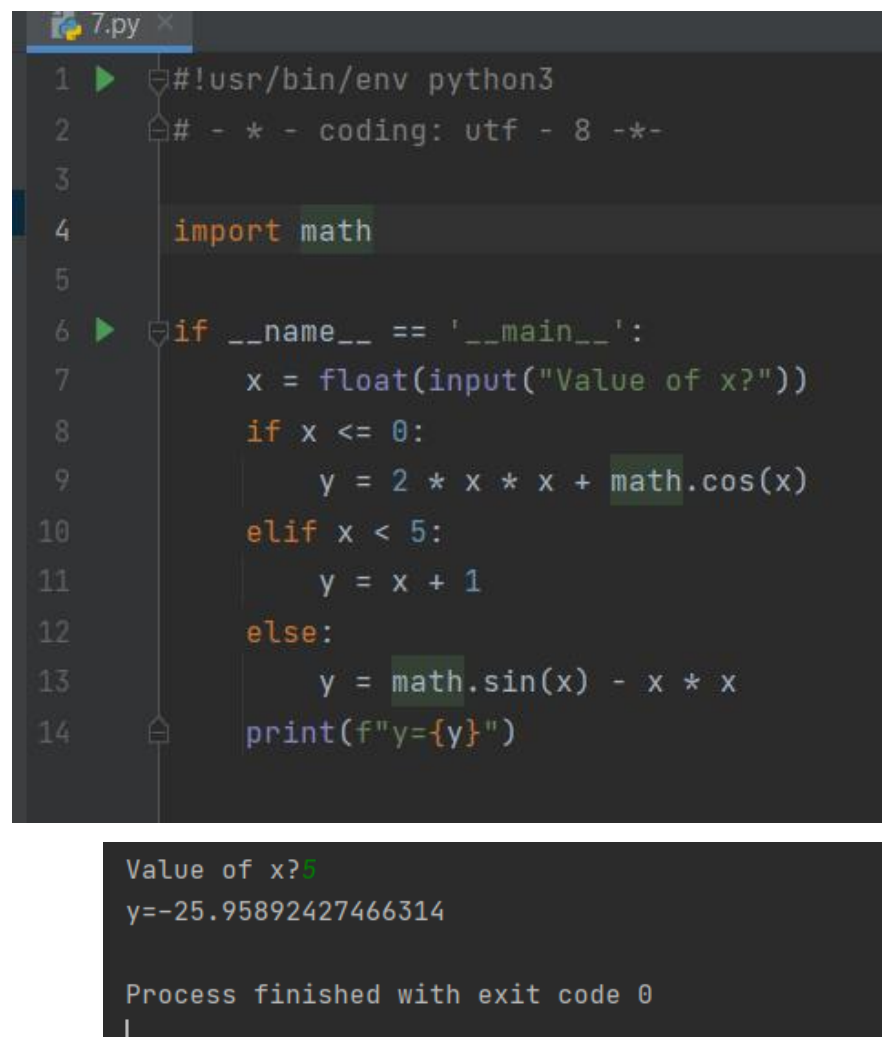
Условные операторы и циклы в языке Python3

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

<https://github.com/alina-j/lab3>

Порядок выполнения работы:

Пример 1. Составить UML-диаграмму деятельности и программу с использованием конструкции ветвления и вычислить значение функции.



```
7.py x
1  ▶  #!/usr/bin/env python3
2      # - * - coding: utf - 8 -*-
3
4      import math
5
6  ▶  if __name__ == '__main__':
7          x = float(input("Value of x?"))
8          if x <= 0:
9              y = 2 * x * x + math.cos(x)
10         elif x < 5:
11             y = x + 1
12         else:
13             y = math.sin(x) - x * x
14         print(f"y={y}")
```

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

Process finished with exit code 0
```

Рисунок 1 – программа 1

Пример 2

```
1  ▶  #!/usr/bin/env python3
2  #  -*- coding: utf-8 -*-
3
4  import sys
5
6
7  ▶  if __name__ == '__main__':
8      n = int(input("Введите номер месяца:
9      if n == 1 or n == 2 or n == 12:
10         print("Зима")
11     elif n == 3 or n == 4 or n == 5:
12         print("Весна")
13     elif n == 6 or n == 7 or n == 8:
14         print("Лето")
15     elif n == 9 or n == 10 or n == 11:
16         print("Осень")
17     else:
18         print("Ошибка!", file=sys.stderr)
19     exit(1)
```

```
Введите номер месяца: 8
Лето
Process finished with exit code 0
```

Рисунок 2 – работа программы

```
Введите номер месяца: 13
Ошибка!
Process finished with exit code 1
```

Рисунок 3.– работа программы

Пример 3

```
7.py x 8.py x 9.py x
1  ▶  #!/usr/bin/env python3
2  #  -*- coding: utf-8 -*-
3
4  import math
5
6  ▶  if __name__ == '__main__':
7      n = int(input("Value of n? "))
8      x = float(input("Value of x? "))
9      S = 0.0
10     for k in range(1, n + 1):
11         a = math.log(k * x) / (k * k)
12         S += a
13     print(f"S = {S}")
```

```

Value of n? 2
Value of x? 6
S = 0.6212266624470001

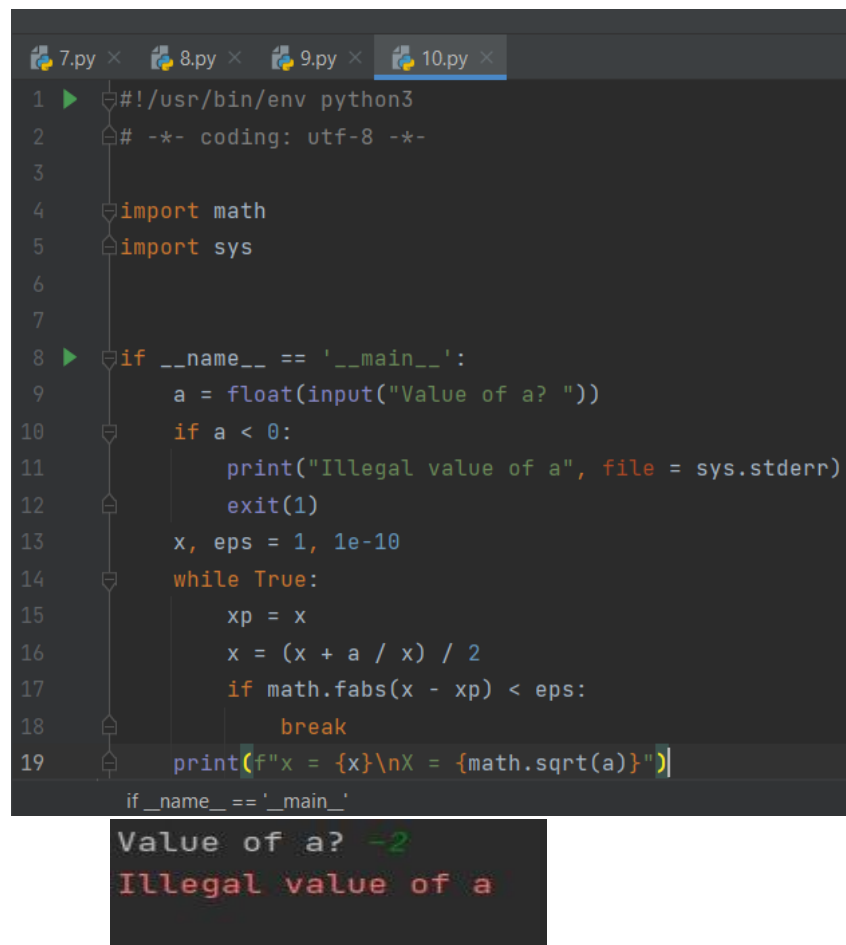
Process finished with exit code 0

```

Рисунок 4.— Пример работы программы при вводе «2» и «6»

Для примеров 4 и 5 строим UML-диаграмму деятельности.

Пример 4



The image shows a screenshot of a Python IDE with a dark theme. The top part displays the source code of a file named 10.py. The code includes a shebang, a UTF-8 encoding declaration, imports for math and sys, and a main function that prompts the user for a value 'a'. It checks if 'a' is less than 0, and if so, prints an error message and exits with code 1. Otherwise, it calculates the square root of 'a' using a Newton-Raphson method and prints the result.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5  import sys
6
7
8  if __name__ == '__main__':
9      a = float(input("Value of a? "))
10     if a < 0:
11         print("Illegal value of a", file = sys.stderr)
12         exit(1)
13     x, eps = 1, 1e-10
14     while True:
15         xp = x
16         x = (x + a / x) / 2
17         if math.fabs(x - xp) < eps:
18             break
19     print(f"x = {x}\nX = {math.sqrt(a)}")

```

Below the code, the execution output is shown in a separate window. It displays the prompt "Value of a? -2" and the response "Illegal value of a" in red text, indicating an error.

```

Value of a? -2
Illegal value of a

```

Рисунок 5.— работа программы при вводе «-2»

```

Value of a? 2
x = 1.414213562373095
X = 1.4142135623730951

Process finished with exit code 0

```

Рисунок 6 .— Пример работы программы при вводе «2»

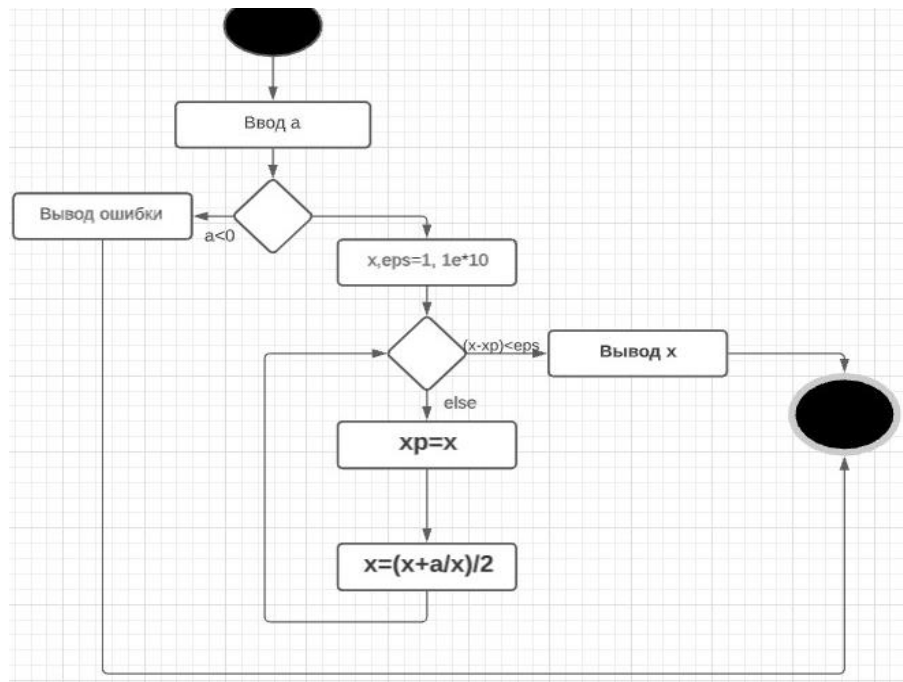


Рисунок 7.– UML диаграмма для примера 4

Пример 5

```

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

import math
import sys
EULER = 0.5772156649015328606
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

    while math.fabs(a) > EPS:
        a *= x * k / (k + 1) ** 2
        S += a
        k += 1

    print(f"Ei({x}) = {EULER + math.log(math.fabs(x)) + S}")
  
```

```

Value of x? 10
Ei(10.0) = 2492.228976241855
  
```

Рисунок 8.– Пример работы программы при вводе «10»

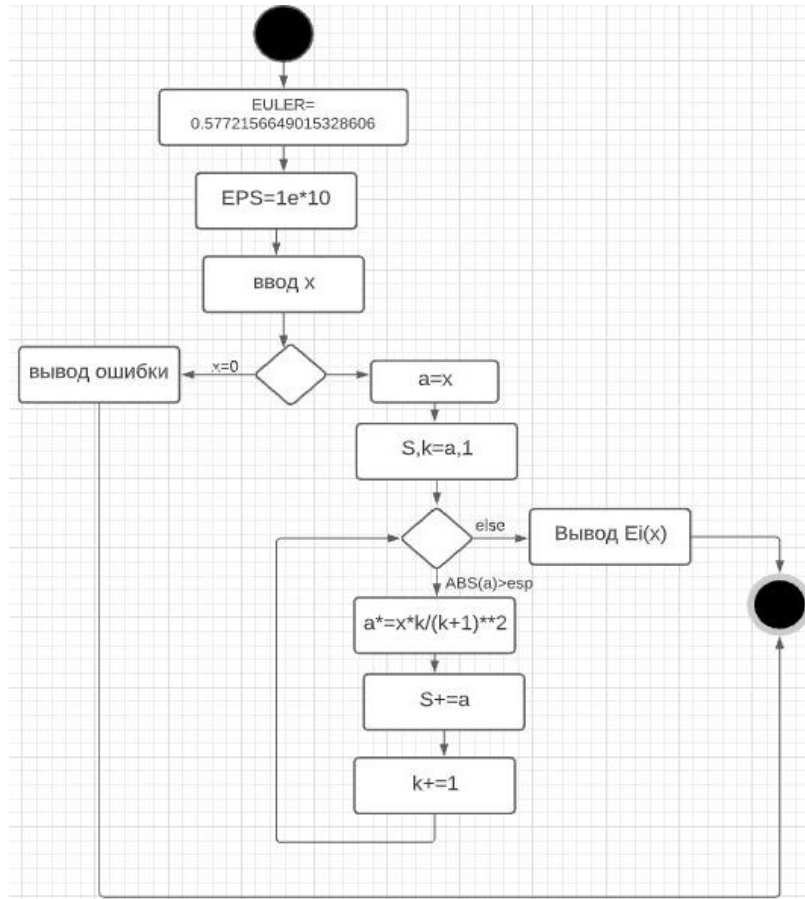


Рисунок 9.– UML диаграмма для примера 5

Выполнение индивидуальных заданий по вариантам:

Задание 1 (Вариант 5)

С клавиатуры вводится цифра m (от 1 до 4). Вывести на экран названия месяцев, соответствующих времени года с номером (считать зиму временем года № 1).

```
8.py x 9.py x 10.py x 11.py x o1.py x
#!/usr/bin/env python3
# - * - coding: utf - 8 -*-
#С клавиатуры вводится цифра m (от 1 до 4). Вывести на экран названи

import sys

if __name__ == '__main__':
    m= int(input("Введите номер времени года:"))
    if m == 1:
        print("Декабрь, Январь, Февраль")
    elif m == 2:
        print("Март, Апрель, Май")
    elif m == 3:
        print("Июнь, Июль, Август")
    elif m == 4:
        print("Сентябрь, Октябрь, Ноябрь")
    else:
        print("Ошибка!", file = sys.stderr)
        exit(1)
```

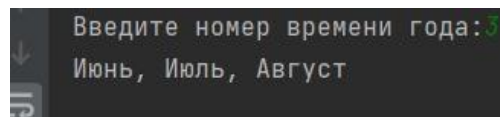


Рисунок 11.— работа программы при вводе «3»

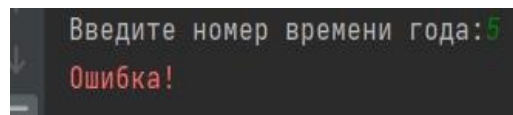


Рисунок 12.— работа программы при вводе «5»

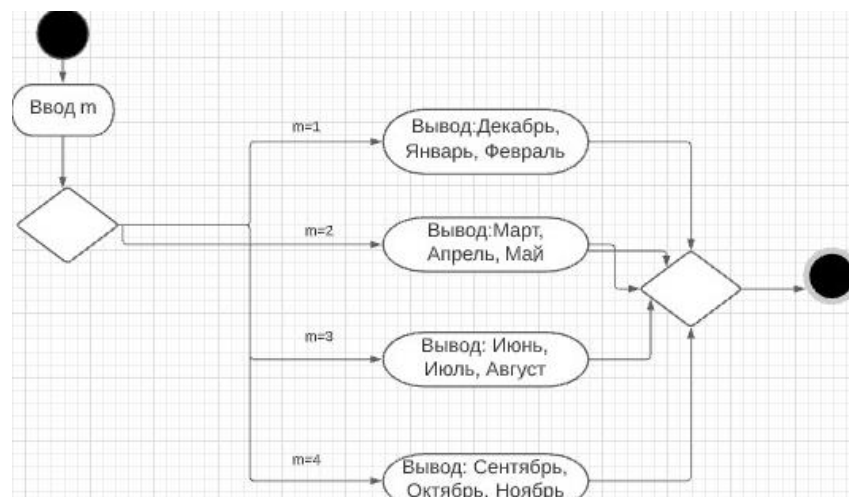


Рисунок 13.— UML диаграмма для индивидуального задания 1
Задание 2(Вариант 17)

Определить, есть ли среди трёх заданных чисел нечётные.

```
7.py x 8.py x 9.py x 10.py x 11.py x o1.py x o2.py x
1  #!usr/bin/env python3
2  # - * - coding: utf - 8 -*
3  #определить, есть ли среди трёх заданных чисел нечётные.
4
5  import sys
6
7
8  if __name__ == '__main__':
9      a = int(input('a-любое натуральное число ='))
10     b = int(input('b- любое натуральное число ='))
11     c = int(input('c- лбое натуральное число ='))
12     if a % 2 == 0:
13         print('Четное число')
14     else:
15         print('Нечетное число')
16     if b % 2 == 0:
17         print('Четное число')
18     else:
19         print('Нечетное число')
20     if c % 2 == 0:
21         print('Четное число')
22     else:
23         print('Нечетное число')
24     exit(1)
```

```
a-любое натуральное число =5
b- любое натуральное число =2
c- лбое натуральное число =4
Нечетное число
Четное число
Четное число
```

Рисунок 14.— работа программы при вводе «5» «2» «4»

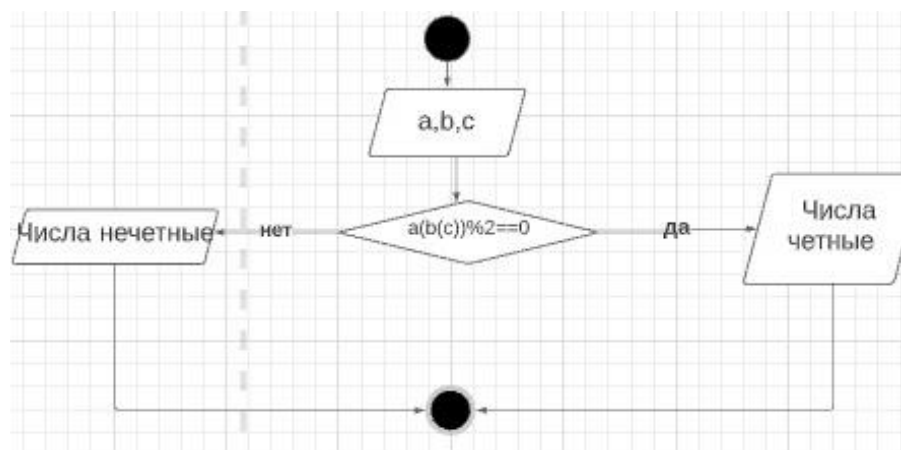


Рисунок 15. – UML диаграмма для индивидуального задания 2

Задание 3

Вариант 16

Илья выучил в первый день 5 английских слов. В каждый следующий день он выучивал на 2 слова больше, чем в предыдущий. Сколько английских слов выучит Илья в 10-ый день занятий.

```

1  #!/usr/bin/env python3
2  # - * - coding: utf - 8 -*-
3  #Илья выучил в первый день 5 английских слов. В каждый следующий день он
4  import math
5  import sys
6
7
8  if __name__ == '__main__':
9      d = int(input('Количество дней='))
10     A = int(input('Количество выученных за каждый день слов='))
11     if d <= 10:
12         print(A + 2 * 9)
13     else:
14         print('A')

```

Количество дней=1
Количество выученных за каждый день слов=5
23

Рисунок 16.— Результаты работы

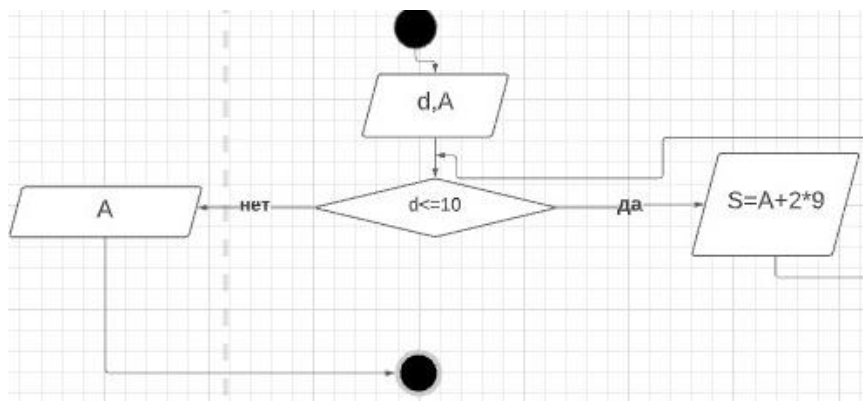


Рисунок 17.— UML диаграмма для задания 3.

Ответы на вопросы:

1. Для чего нужны диаграммы деятельности UML?

Диаграммы деятельности - это один из пяти видов диаграмм, применяемых в UML для моделирования динамических аспектов поведения системы. Диаграмма деятельности - это, по существу, блок-схема, которая показывает, как поток управления переходит от одной деятельности к другой, однако, по сравнению с последней, у ней есть явные преимущества:

поддержка многопоточности и объектно-ориентированного проектирования.

Диаграмма деятельности (Activity diagram) показывает поток переходов от одной деятельности к другой.

2. Что такое состояние действия и состояние деятельности?

Состояние действия – это состояние, которое не может быть подвергнуто дальнейшей декомпозиции.

Состояние деятельности – это составное состояние, поток управления которого включает только другие состояния деятельности и действий.

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

Переход представляется, как простая линия со стрелкой.

Ветвление представляется в виде ромба

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Алгоритм разветвляющейся структуры – это алгоритм, в котором вычислительный процесс осуществляется либо по одной, либо по другой ветви, в зависимости от выполнения некоторого условия.

5. Чем отличается разветвляющийся алгоритм от линейного?

Линейные алгоритмы выполняются команда за командой, а в разветвляющихся алгоритмах путь программы зависит от условия.

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

Условный оператор – элемент компьютерной программы, осуществляющий ветвление операций.

Условный оператор может иметь две формы (структуры) – полную или неполную.

Полная форма условного оператора имеет вид: if–then–else Неполная форма: if –then

7. Какие операторы сравнения используются в Python?

==(Проверяет равны ли оба операнда. Если да, то условие становится истинным)

!=(Проверяет равны ли оба операнда. Если нет, то условие становится истинным.)

<>(Проверяет равны ли оба операнда. Если нет, то условие становится истинным)

<,>,>=,<=

8. Что называется простым условием? Приведите примеры.

Простое условие — это два выражения, связанные одним из знаков отношений: = (равно), > (больше), < (меньше), >= (больше либо равно), <= (меньше либо равно).

9. Что такое составное условие? Приведите примеры.

Составные условия — это условия, состоящие из двух или более простых условий, соединенных с помощью логических операций: and , or , not

10. Какие логические операторы допускаются при составлении сложных условий?–True, False

11. Может ли оператор ветвления содержать внутри себя другие ветвления?– Да, может

12. Какой алгоритм является алгоритмом циклической структуры?

Алгоритм циклической структуры - это алгоритм, в котором происходит многократное повторение одного и того же участка программы.

Такие повторяемые участки вычислительного процесса называются циклами.

13. Типы циклов в языке Python.– цикл while, - цикл for.

14. Назовите назначение и способы применения функции range .

Функция range возвращает неизменяемую последовательность чисел в виде объекта range.

Функция range хранит только информацию о значениях start, stop и step и вычисляет значения по мере необходимости. Это значит, что независимо от размера диапазона, который описывает функция range, она всегда будет занимать фиксированный объем памяти.

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

16. Могут ли быть циклы вложенными?– Могут, это циклы, которые располагаются внутри других циклов

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

Чтобы организовать бесконечный цикл, используют конструкцию while (true). При этом он, как и любой другой цикл, может быть прерван директивой break.

18. Для чего нужен оператор break ?

Оператор break предназначен для досрочного прерывания работы цикла while.

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

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

20. Для чего нужны стандартные потоки stdout и stderr?

STDOUT - стандартный вывод - то, куда выводят данные команды echo/print, консоль или сокет отправляющий данные браузеру.

STDERR - поток сообщений об ошибках.

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

Если вы хотите перенаправить stderr, просто добавьте следующую строку после `sys.stdout=redir: sys.stderr=redir`.

22. Каково назначение функции `exit` ?— `exit()` поможет не просто прервать выполнения цикла, но и полностью останавливает программу, код далее не читается. `exit()` является помощником для интерактивной оболочки (консоли).

Вывод: в ходе лабораторной работы были изучены различные алгоритмы: разветвляющиеся и циклические, также были ознакомлены с такими операторами как: `if` , `while` , `for` , `break` и `continue`, которые позволяют реализовать алгоритмы; были построены UML-диаграммы.