

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 № 2.2
дисциплины «Программирование на Python»

Вариант 15

Выполнил:
Кенесбаев Хилол Куат улы
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

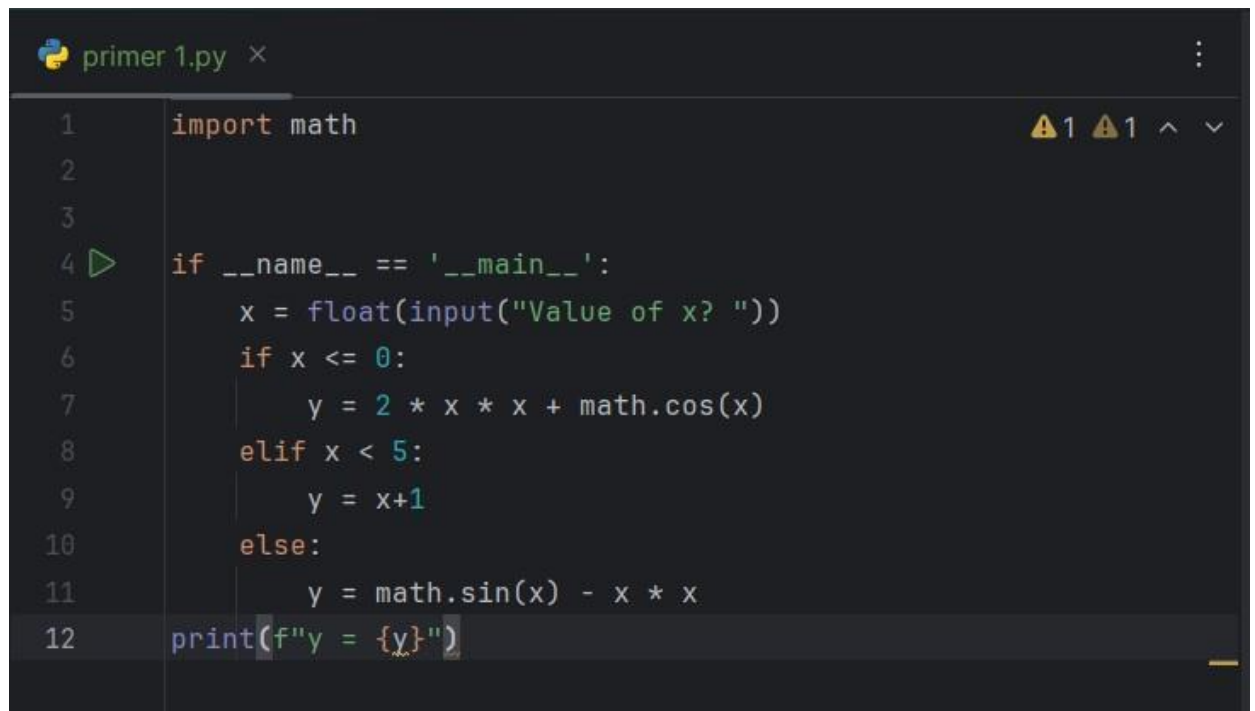
Ставрополь, 2023 г.

Тема: Условные операторы и циклы в языке питон

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

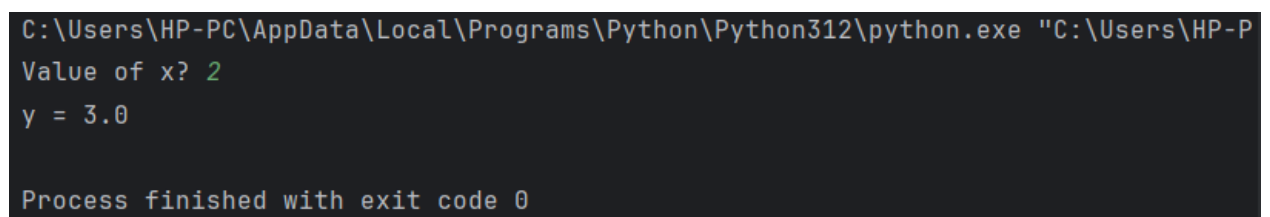
Порядок выполнения работы:

1. Проработал примеры лабораторной работы:



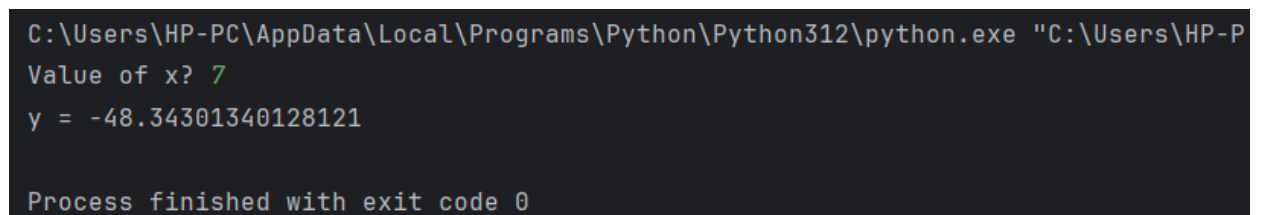
```
1 import math
2
3
4 if __name__ == '__main__':
5     x = float(input("Value of x? "))
6     if x <= 0:
7         y = 2 * x * x + math.cos(x)
8     elif x < 5:
9         y = x+1
10    else:
11        y = math.sin(x) - x * x
12    print(f'y = {y}')
```

Рисунок 1 Пример 1



```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\HP-P
Value of x? 2
y = 3.0
Process finished with exit code 0
```

Рисунок 2 Первый тест для примера 1



```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\HP-P
Value of x? 7
y = -48.34301340128121
Process finished with exit code 0
```

Рисунок 3 Второй тест для примера 1



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

Рисунок 4 Пример 2

```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\HP-P
Введите номер месяца: 7
Лето

Process finished with exit code 0
```

Рисунок 5 Первый тест для примера 2

```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\HP-P
Введите номер месяца: 12
Зима

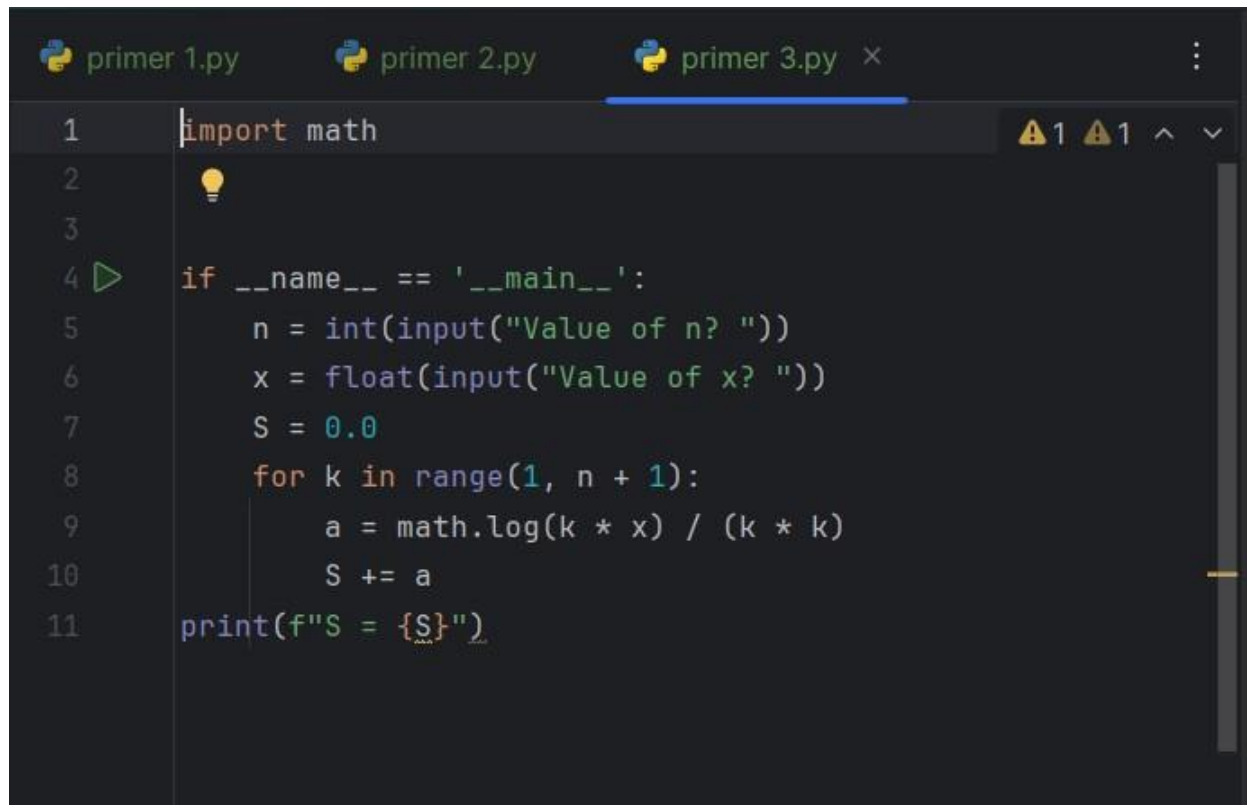
Process finished with exit code 0
```

Рисунок 6 Второй тест для примера 2

```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\HP-P
Введите номер месяца: 5
Весна

Process finished with exit code 0
```

Рисунок 7 Третий тест для примера 2



```
1 import math
2
3
4 if __name__ == '__main__':
5     n = int(input("Value of n? "))
6     x = float(input("Value of x? "))
7     S = 0.0
8     for k in range(1, n + 1):
9         a = math.log(k * x) / (k * k)
10        S += a
11    print(f"S = {S}")
```

Рисунок 8 Пример 3

```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\HP-P
Value of n? 7
Value of x? 5
S = 2.9690027813136357

Process finished with exit code 0
```

Рисунок 9 Первый тест для примера 3

```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\HP-P
Value of n? 9
Value of x? 7
S = 3.591726448373983

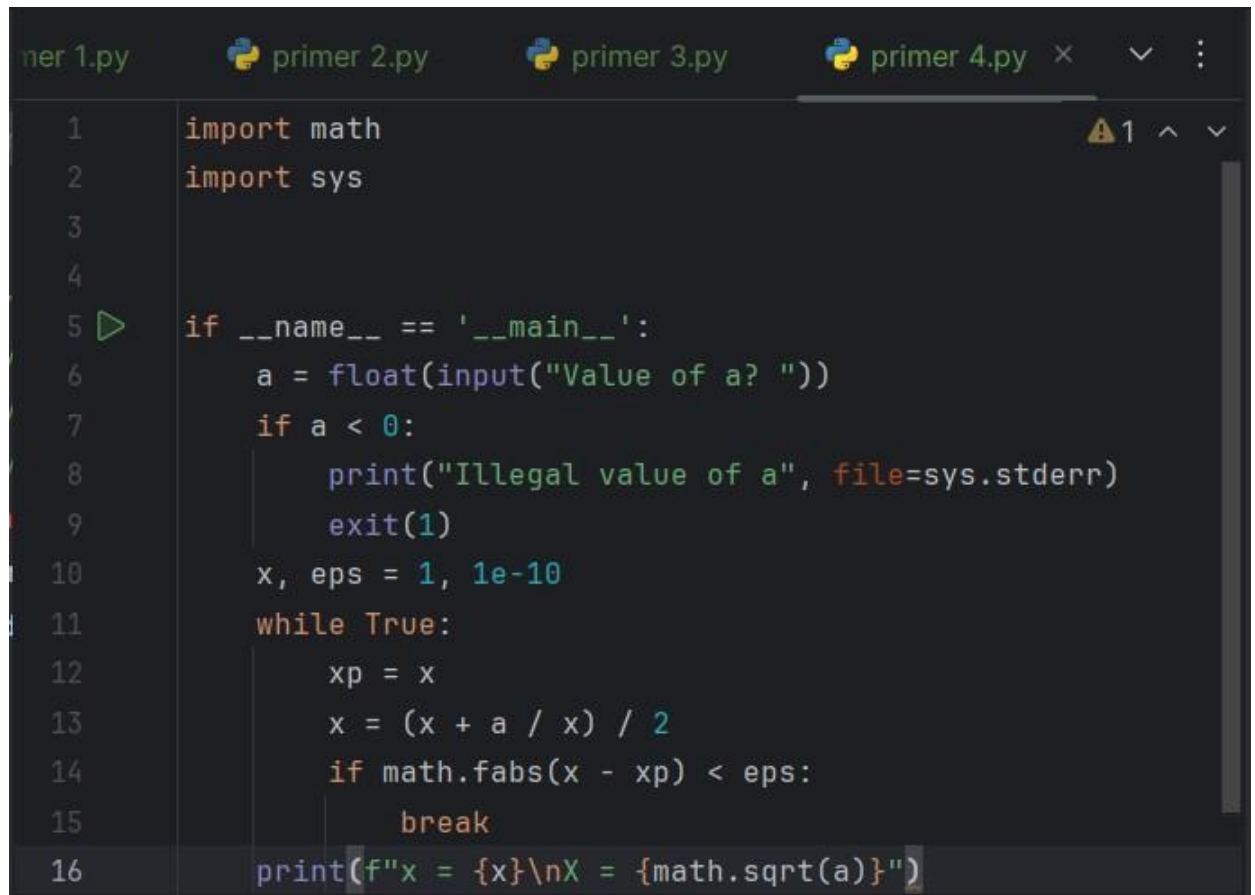
Process finished with exit code 0
```

Рисунок 10 Второй тест для примера 3

```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\HP-P
Value of n? 1
Value of x? 5
S = 1.6094379124341003

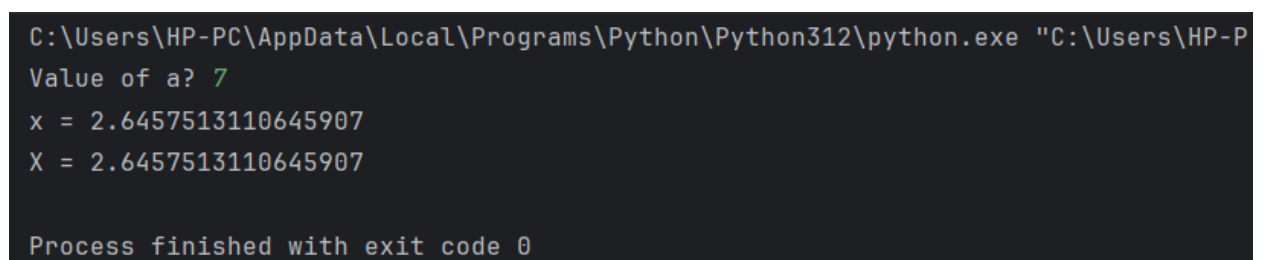
Process finished with exit code 0
```

Рисунок 11 Третий тест для примера 3



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

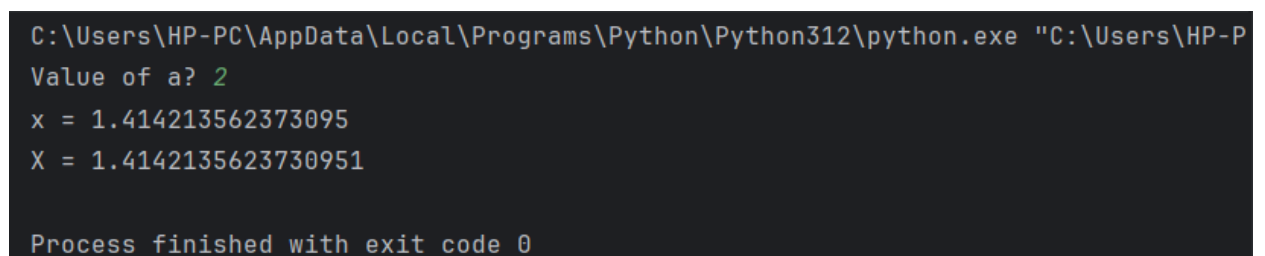
Рисунок 12 Пример 4



```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\HP-P
Value of a? 7
x = 2.6457513110645907
X = 2.6457513110645907

Process finished with exit code 0
```

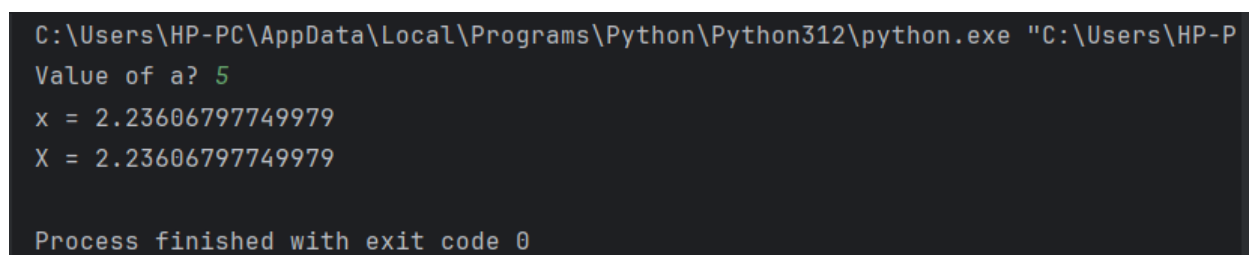
Рисунок 13 Первый тест для примера 4



```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\HP-P
Value of a? 2
x = 1.414213562373095
X = 1.4142135623730951

Process finished with exit code 0
```

Рисунок 14 Второй тест для примера 4



```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\HP-P
Value of a? 5
x = 2.23606797749979
X = 2.23606797749979

Process finished with exit code 0
```

Рисунок 15 Третий тест для примера 5

```

1  import math
2  import sys
3
4  # Постоянная Эйлера.
5  EULER = 0.5772156649015328606
6  # Точность вычислений.
7  EPS = 1e-10
8  if __name__ == '__main__':
9      x = float(input("Value of x? "))
10     if x == 0:
11         print("Illegal value of x", file=sys.stderr)
12         exit(1)
13     a=x
14     S, k = a, 1
15     # Найти сумму членов ряда.
16     while math.fabs(a) > EPS:
17         a *= x * k / (k + 1) ** 2
18         S += a
19         k += 1
20     # Вывести значение функции.
21     print(f"Ei({x}) = {EULER + math.log(math.fabs(x)) + S}")

```

Рисунок 16 Пример 5

```

C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\HP-P
Value of x? 7
Ei(7.0) = 191.50474333549477

```

Рисунок 17 Первый тест для примера 5

```

C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\HP-P
Value of x? 2
Ei(2.0) = 4.954234355999365

Process finished with exit code 0

```

Рисунок 18 Второй тест для примера 5

```

C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\HP-P
Value of x? 6
Ei(6.0) = 85.98976214243285

Process finished with exit code 0

```

Рисунок 19 Третий тест для примера 5

2. Диаграммы для примеров 4 и 5

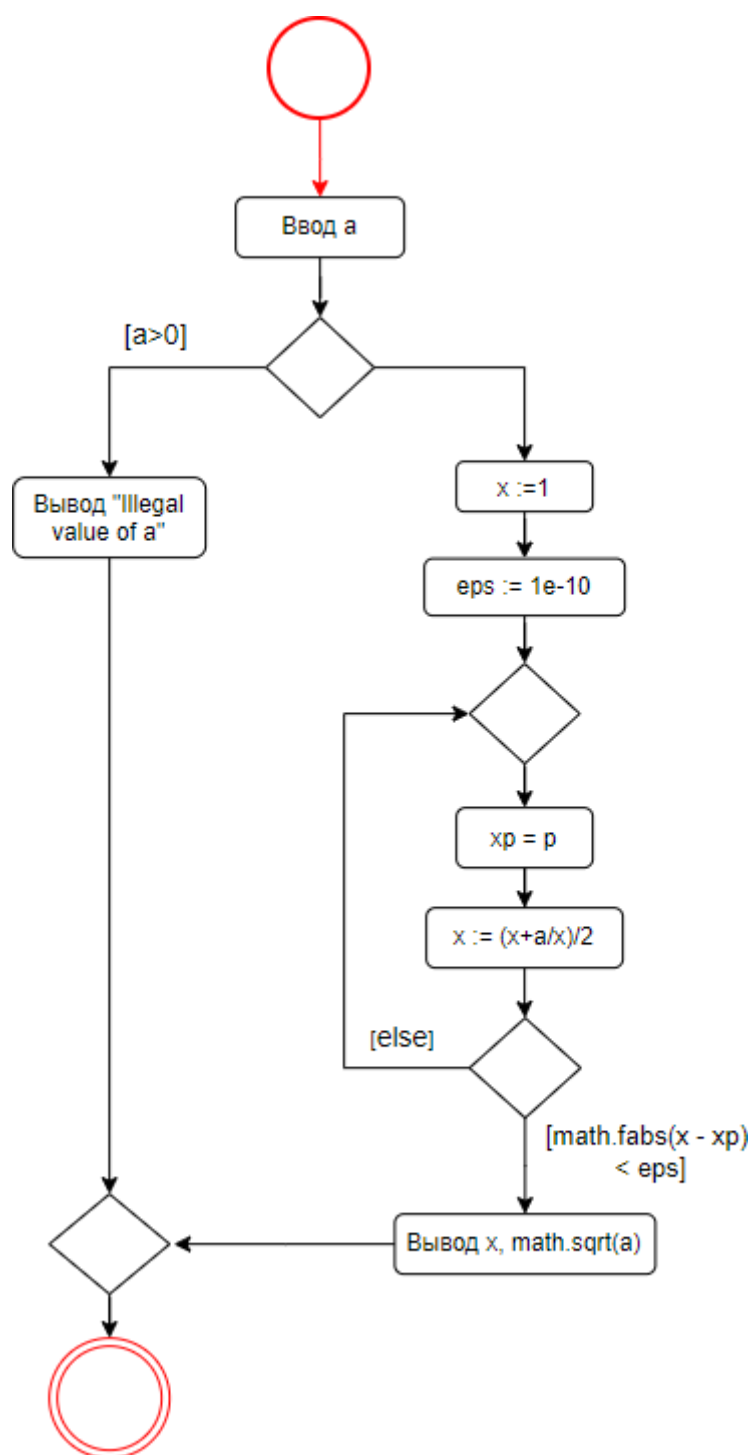


Рисунок 20 UML диаграмма примера 4

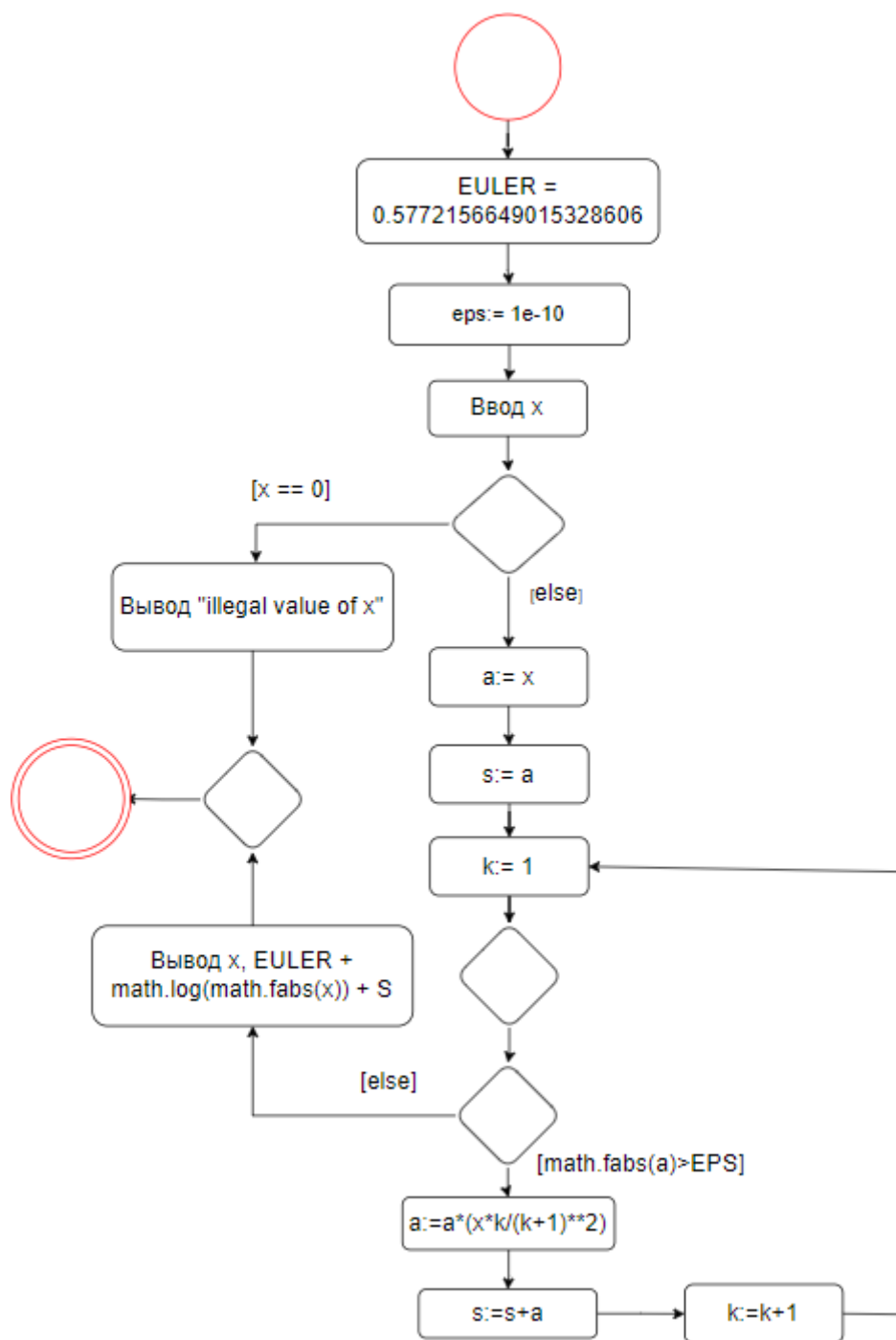


Рисунок 21 UML диаграмма примера 5

3. Выполнил индивидуальное задание 1:

Дано число m ($1 \leq m \leq 12$). Определить, сколько дней в месяце с номером m

Код программы:

```
import calendar
```

```
# Введите номер месяца от 1 до 12
```

```
m = int(input("Введите номер месяца (от 1 до 12): "))
```

```
# Проверка корректности введенного номера месяца
```

```
if m < 1 or m > 12:
```

```
    print("Некорректный номер месяца. Введите число от 1 до 12.")
```

```
else:
```

```
# Определение количества дней в указанном месяце
```

```
days_in_month = calendar.monthrange(2024, m)[1]
```

```
print(f"В месяце с номером {m} количество дней: {days_in_month}")
```

```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe C:\Users\HP-PC\Desktop\study\python\пересдача\Lab2.2\код\individual1.py
Введите номер месяца (от 1 до 12): 7
В месяце с номером 7 количество дней: 31

Process finished with exit code 0
|
```

Рисунок 22 Результат работы программы индивидуального задания 1

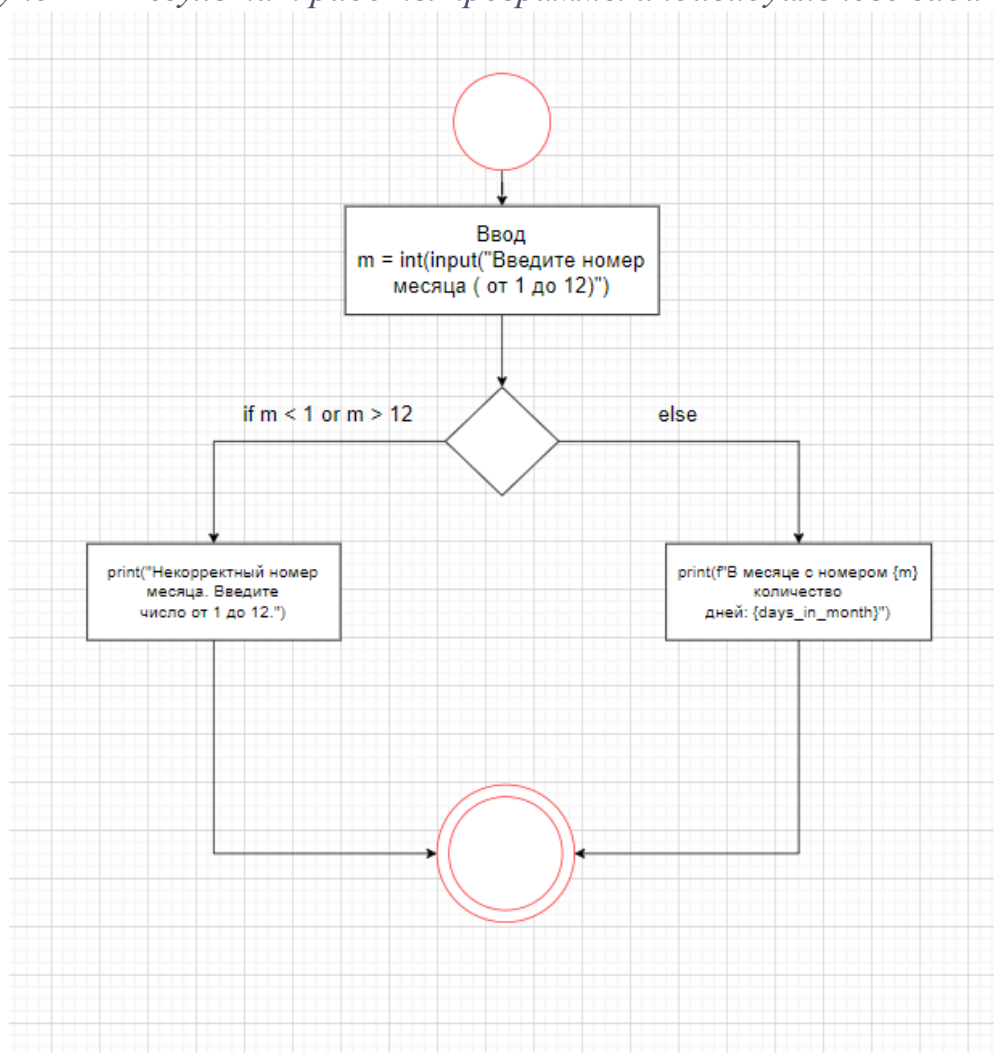


Рисунок 23 UML диаграмма индивидуального задания 1

Выполнил индивидуальное задание 2:

Составить программу решения квадратного уравнения. Выводить также комплексные решения.

Код программы:

```
import cmath

def solve_quadratic(a, b, c):
    # Вычисляем дискриминант
    discriminant = (b ** 2) - (4 * a * c)

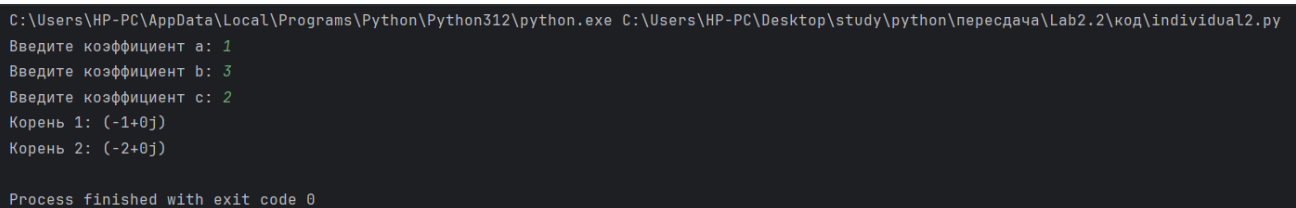
    # Вычисляем корни уравнения
    root1 = (-b + cmath.sqrt(discriminant)) / (2 * a)
    root2 = (-b - cmath.sqrt(discriminant)) / (2 * a)

    return root1, root2

# Ввод коэффициентов квадратного уравнения
a = float(input("Введите коэффициент a: "))
b = float(input("Введите коэффициент b: "))
c = float(input("Введите коэффициент c: "))

# Решение квадратного уравнения
solution1, solution2 = solve_quadratic(a, b, c)

print(f"Корень 1: {solution1}")
print(f"Корень 2: {solution2}")
```



```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe C:\Users\HP-PC\Desktop\study\python\пересдача\Lab2.2\код\individual2.py
Введите коэффициент a: 1
Введите коэффициент b: 3
Введите коэффициент c: 2
Корень 1: (-1+0j)
Корень 2: (-2+0j)
Process finished with exit code 0
```

Рисунок 24 Результат работы программы индивидуального задания 2

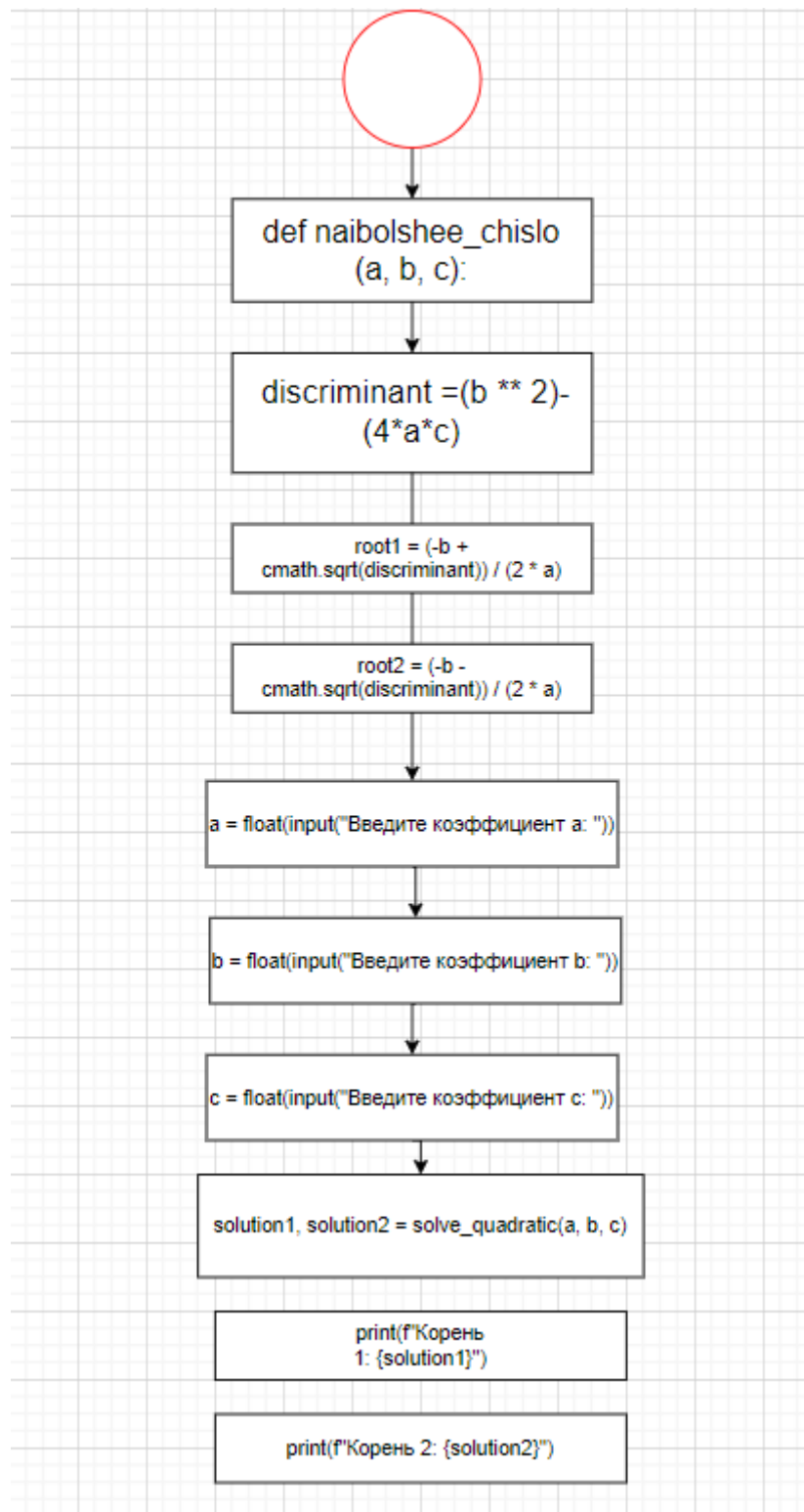


Рисунок 25 UML диаграмма индивидуального задания 2

Выполнил индивидуальное задание 3:

Вычислить сумму всех n-значных чисел, кратных k ($1 \leq n \leq 4$)

Код программы:

```
def sum_of_multiples(n, k):
    total_sum = 0
    for i in range(10**(n-1), 10**n): # Генерируем все n-значные числа
        if i % k == 0: # Проверяем, кратно ли число k
            total_sum += i
    return total_sum

# Ввод значений n и k
n = int(input("Введите значение n (от 1 до 4): "))
k = int(input("Введите значение k: "))

# Вычисление суммы всех n-значных чисел, кратных k
result = sum_of_multiples(n, k)
print(f"Сумма всех {n}-значных чисел, кратных {k}: {result}")
```

```
C:\Users\HP-PC\AppData\Local\Programs\Python\Python312\python.exe C:\Users\HP-PC\Desktop\study\python\пересдача\Lab2.2\код\individual3.py
Введите значение n (от 1 до 4): 3
Введите значение k: 5
Сумма всех 3-значных чисел, кратных 5: 98550
```

Рисунок 26 Результат работы программы индивидуального задания 3

Ответы на контрольные 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.

2. Что такое состояние действия и состояние деятельности?

В потоке управления, моделируемом диаграммой деятельности, происходят различные события. Можно вычислить выражение, в результате чего изменяется значение некоторого атрибута или возвращается некоторое значение. Также, например, можно выполнить операцию над объектом, послать ему сигнал или даже создать его или уничтожить. Все эти выполняемые атомарные вычисления называются состояниями действия, поскольку каждое из них есть состояние системы, представляющее собой выполнение некоторого действия.

Состояния действия не могут быть подвергнуты декомпозиции. Кроме того, они атомарны. Это значит, что внутри них могут происходить различные события, но выполняемая в состоянии действия работа не может быть прервана. Обычно предполагается, что длительность одного состояния действия занимает неощутимо малое время.

В противоположность этому состояния деятельности могут быть подвергнуты дальнейшей декомпозиции, вследствие чего выполняемую деятельность можно представить с помощью других диаграмм деятельности. Состояния деятельности не являются атомарными, то есть могут быть прерваны. Предполагается, что для их завершения требуется заметное время. Можно считать, что состояние действия – это частный вид состояния

деятельности, а конкретнее – такое состояние, которое не может быть подвергнуто дальнейшей декомпозиции. А состояние деятельности можно представлять себе как составное состояние, поток управления которого включает только другие состояния деятельности и действий.

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

Когда действие или деятельность в некотором состоянии завершается, поток управления сразу переходит в следующее состояние действия или деятельности. Для описания этого потока используются переходы, показывающие путь из одного состояния действия или деятельности в другое. В UML переход представляется простой линией со стрелкой.

Поток управления должен где-то начинаться и заканчиваться (разумеется, если это не бесконечный поток, у которого есть начало, но нет конца). Можно задать как начальное состояние (закрашенный кружок), так и конечное (закрашенный кружок внутри окружности).

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Простые последовательные переходы встречаются наиболее часто, но их одних недостаточно для моделирования любого потока управления. Как и в блок-схеме, вы можете включить в модель ветвление, которое описывает различные пути выполнения в зависимости от значения некоторого булевского выражения. Как видно из рис. 4.3, точка ветвления представляется ромбом. В точку ветвления может входить ровно один переход, а выходить – два или более. Для каждого исходящего перехода задается булевское выражение, которое вычисляется только один раз при входе в точку ветвления. Ни для каких двух исходящих переходов эти сторожевые условия не должны одновременно принимать значение «истина», иначе поток управления окажется неоднозначным. Но эти условия должны покрывать все возможные варианты, иначе поток остановится.

Для удобства разрешается использовать ключевое слово `else` для пометки того из исходящих переходов, который должен быть выбран в случае,

если условия, заданные для всех остальных переходов, не выполнены.

Реализовать итерацию можно, если ввести два состояния действия - в

первом устанавливается значение счетчика, во втором оно увеличивается - и точку ветвления, вычисление в которой показывает, следует ли прекратить итерации.

5. Чем отличается разветвляющийся алгоритм от линейного?

Линейный алгоритм – алгоритм, все этапы которого выполняются однократно и строго последовательно.

Разветвляющийся алгоритм – это алгоритм, в котором вычислительный процесс осуществляется либо по одной, либо по другой ветви, в зависимости от выполнения некоторого условия. Программа разветвляющейся структуры реализует такой алгоритм. В программе разветвляющейся структуры имеется один или несколько условных операторов. Для программной реализации условия используется логическое выражение. В сложных структурах с большим числом ветвей применяют оператор выбора.

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

Условные операторы – это специальные конструкции, благодаря которым в программе происходит ветвление в зависимости от условий.

Условный оператор имеет полную и краткую формы.

7. Какие операторы сравнения используются в Python?

If, elif, else.

8. Что называется простым условием? Приведите примеры.

Простое условие в программировании – это выражение, которое может быть истинным или ложным. Оно используется для принятия решений в коде на основе значения переменных или других условий.

Пример: $x > 5$

9. Что такое составное условие? Приведите примеры.

Составное условие – логическое выражение, содержащее несколько простых условий, объединенных логическими операциями. Это операции not, and, or.

Пример: `a == b or a == c`

10. Какие логические операторы допускаются при составлении сложных условий?

Not, and, or.

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

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.

В Python есть 2 типа циклов: – цикл while, – цикл for.

Оператор for выполняет указанный набор инструкций заданное количество раз, которое определяется количеством элементов в наборе. Оператор цикла while выполняет указанный набор инструкций до тех пор, пока условие цикла истинно.

14. Назовите назначение и способы применения функции range. Функция range возвращает неизменяемую последовательность чисел в виде объекта range.

Параметры функции:

start - с какого числа начинается последовательность. По умолчанию – 0

stop - до какого числа продолжается последовательность чисел.

Указанное число не включается в диапазон

step - с каким шагом растут числа. По умолчанию 1.

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

for i in range(15, 0, -2).

16. Могут ли быть циклы вложенными?

Могут.

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

Бесконечный цикл в программировании – цикл, написанный таким образом, что условие выхода из него никогда не выполняется. Чтобы выйти из цикла нужно использовать оператор 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?

Ввод и вывод распределяется между тремя стандартными потоками: stdout – стандартный вывод (экран), stderr – стандартная ошибка (вывод ошибок на экран)

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

Во-первых нужно импортировать `sys`, а дальше использовать `print(..., file=sys.stderr)`.

22. Каково назначение функции `exit`?

Функция `exit()` модуля `sys` – выход из Python.

Вывод: в ходе выполнения данной лабораторной работы были приобретены навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры, освоены операторы языка Python версии 3.x `if`, `while`, `for`, `break` и `continue`, позволяющие реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.