МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового

развития Кафедра

инфокоммуникаций

Дисциплина: «основы кроссплатформенного программирования»

ОТЧЕТ

по лабораторной работе №3

Выполнил: студент 1 курса группы ИТС-21-1

Якупов Эльдар

Алмазович Проверил: к.ф.-м.н., доцент кафедры

инфокоммуникаций

Воронкин Роман Александрович

Работа защищена с оценкой:_____

Ставрополь, 2022

Тема:

Условные операторы и циклы в языке Python

Цель работы:

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.

Ход работы:

Создадим общедоступный репозиторий - https://github.com/Blekroyt/berd3.git

Пример 1:

```
PS F:\visual code\3\berd3> & "F:/visual code/python.exe" "f:/visual code/3/berd3/Пример 1"
Value of x? 90
y = -8099.106003336399
PS F:\visual code\3\berd3>
```

Рисунок 1. Окно вывода для Примера 1.

Пример 2:

```
PS F:\visual code\3\berd3> & "F:/visual code/python.exe" "f:/visual code/3/berd3/Пример 2"
Введите номер месяца7
Summer
PS F:\visual code\3\berd3>
```

Рисунок 2. Окно вывода для Примера 2.

Пример 3:

```
PS F:\visual code\3\berd3> & "F:/visual code/python.exe" "f:/visual code/3/berd3/Пример 3"
Value of n?15
Value of x?5
S= 3.2398743390535354
```

Рисунок 3. Окно вывода для Примера 3.

Пример 4:

```
PS F:\visual code\3\berd3> & "F:/visual code/python.exe" "f:/visual code/3/berd3/Пример 4"
Value of a? 20
x = 4.47213595499958
X = 4.47213595499958
```

Рисунок 4. Окно вывода для Примера 4.

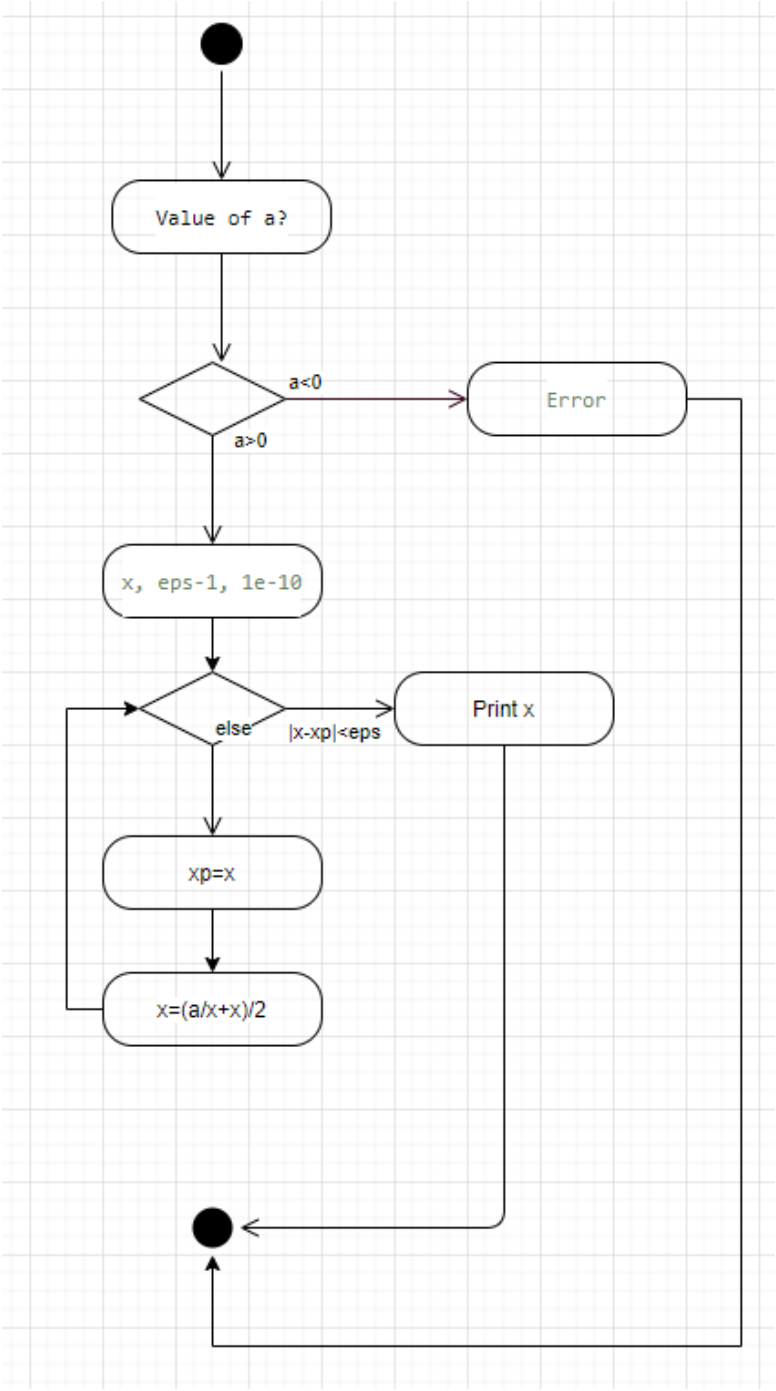

Рисунок 5. UML диаграмма для примера 4.

Пример 5:

```
PS F:\visual code\3\berd3> & "F:/visual code/python.exe" "f:/visual code/3/berd3/Пример 5"
Value of x?40
Ei(40.0) = 6039718263611241.0
```

Рисунок 6. Окно вывода для Примера 5.

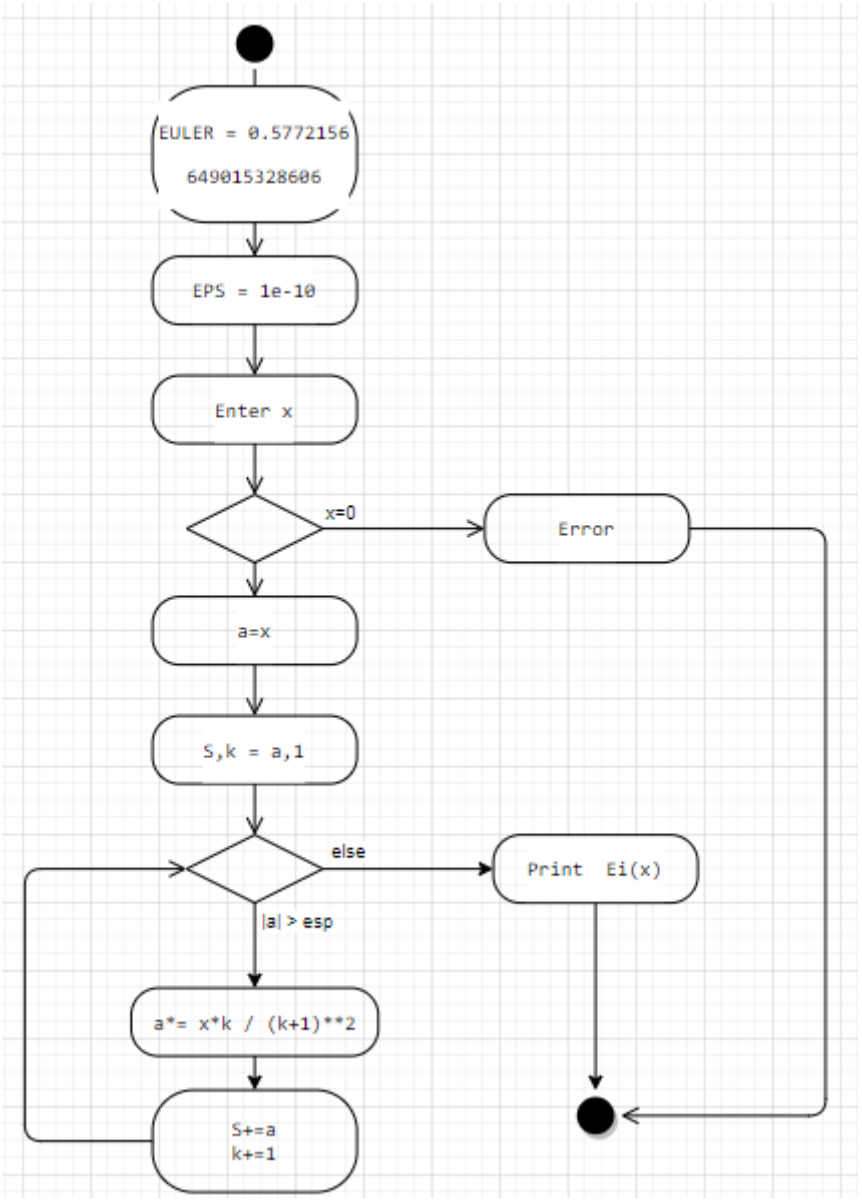

Рисунок 7. UML диаграмма для примера 5.

Индивидуальные задания:

Задание 1.

9. Вводится число экзаменов $N \leq 20$. Напечатать фразу `Мы успешно сдали N экзаменов`, согласовав слово "экзамен" с числом N.

Задание 1 > ...

```python
#!/usr/bin/env python3
# -*- кодировка: utf-8 -*-
import math
import sys

def new_func():
    ex = int(input("Введите число экзаменов: "))
    print(f"Мы успешно сдали { ex } экзаменов!")

new_func()

```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ

```
PS F:\visual code\3\berd3> & "F:/visual code/python.exe" "f:/visual code/3/berd3/Задание 1"
Введите число экзаменов: 20
Мы успешно сдали 20 экзаменов!
PS F:\visual code\3\berd3>
```

Рисунок 8. Окно вывода для Задачи 1.

Задача 2.

22. Треугольник задан координатами своих вершин. Определить принадлежит ли данная точка треугольнику. Координаты вершин треугольника и координаты точки задать самостоятельно.

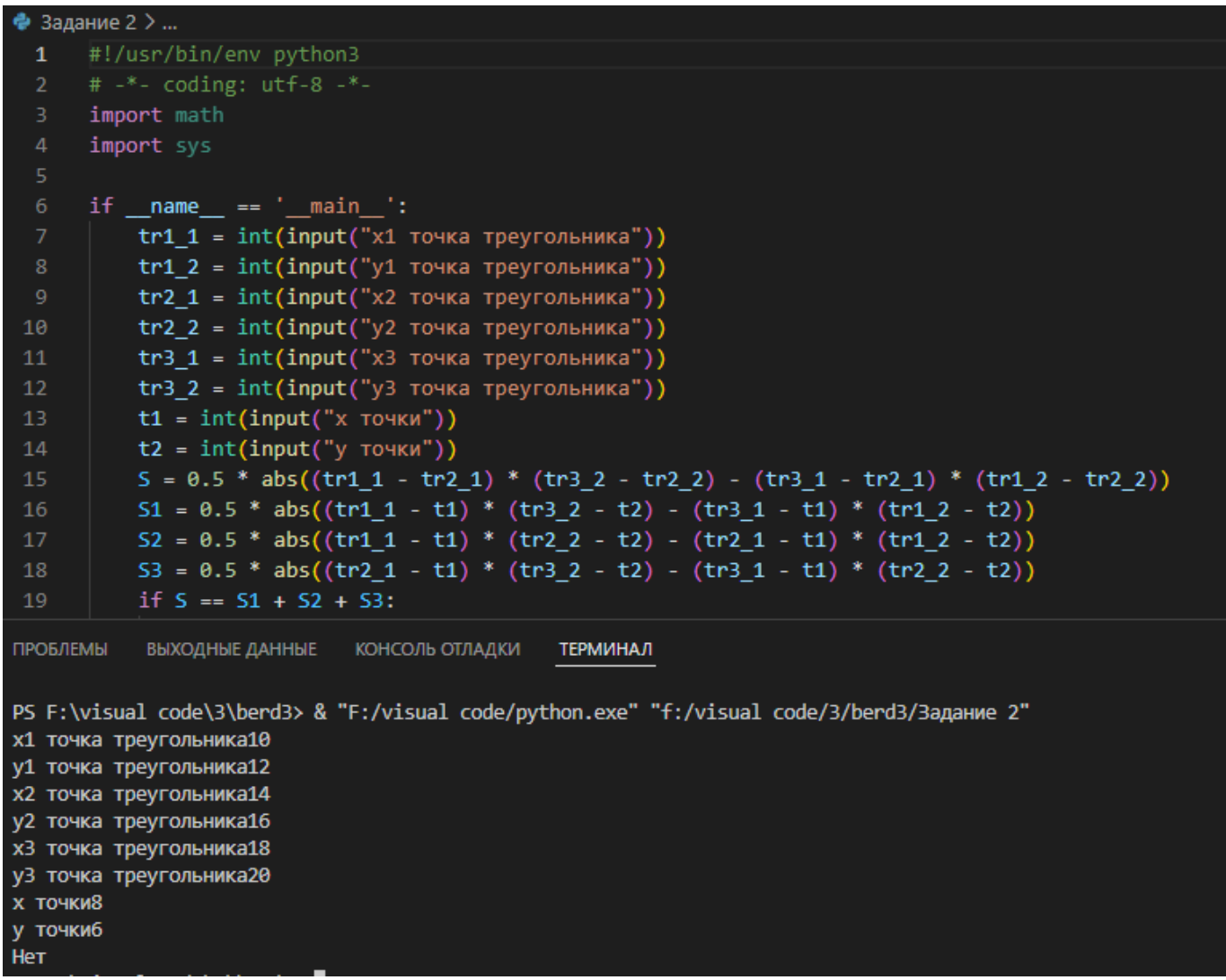

Рисунок 10. Окно вывода для задачи 2.

Рисунок 11. UML диаграмма для Задачи 2.

Задача 3. Составьте программу, которая по номеру дня в году выводит число и месяц в общепринятой форме (например, 33-й день года

- 2 февраля).

Задание 3 > ...

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

if __name__ == '__main__' :

    day = int(input("Введите день: "))
    if day > 0 and 31 >= day:
        m = 1

    elif day > 31 and 59 >= day:
        m = 2

    elif day > 59 and 90 >= day:
        m = 3

    elif day > 90 and 120 >= day:
        m = 4
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ

```
PS F:\visual code\3\berd3> & "F:/visual code/python.exe" "f:/visual code/3/berd3/Задание 3"
Введите день: 33
2 февраля
PS F:\visual code\3\berd3>
```

Рисунок 12. Окно вывода для задачи 3.

Рисунок 13. UML диаграмма для Задачи 3.

Задача повышенной сложности.

8. Интеграл вероятности:

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2)\, dt = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)n!}.$$

Рисунок 14. Условие задачи повышенной сложности

Рисунок 15. Окно вывода для задачи повышенной сложности.

Контрольные вопросы:

1. Для чего нужны диаграммы деятельности UML?

Унифицированный язык моделирования (UML) является стандартным инструментом для создания «чертежей» программного обеспечения. С помощью

UML можно визуализировать, специфицировать, конструировать и документировать артефакты программных систем.

2. Что такое состояние действия и состояние деятельности?

Также, например, можно выполнить операцию над объектом, послать ему сигнал или даже создать его или уничтожить. Все эти выполняемые атомарные вычисления называются состояниями действия.

Состояния деятельности не являются атомарными, то есть могут быть прерваны. Можно считать, что состояние действия - это частный вид состояния деятельности, а конкретнее – такое состояние, которое не может быть подвергнуто дальнейшей

декомпозиции.

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

В UML переход представляется простой линией со стрелкой, в точку ветвления может входить ровно один переход, а выходить – два или более.

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?

В программе разветвляющейся структуры имеется один или несколько условных операторов.

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

Условный оператор ветвления if

Оператор ветвления if позволяет выполнить определенный набор инструкций в зависимости от некоторого условия.

1) Конструкция if;
2) Конструкция if – else;
3) Конструкция if – elif – else;

7. Какие операторы сравнения используются в Python?

В языках программирования используются специальные знаки, подобные тем, которые используются в математике: > (больше), < (меньше), >= (больше или равно), <= (меньше или равно), == (равно), != (не равно).

8. Что называется простым условием? Приведите примеры.

Логические выражения типа (kByte >= 1023) являются

простыми, так как в них выполняется только одна логическая операция.

9. Что такое составное условие? Приведите примеры.

Простые инструкции описываются одной строкой когда, составные же – содержат вложенные инструкции.

```
if x <= 0:
y = 2 * x * x + math.cos(x)
elif x < 5:
y = x + 1
```

10. Какие логические операторы допускаются при составлении сложных условий?

В таких случаях используются специальные операторы, объединяющие два и более простых логических выражения. Широко используются два оператора – так называемые логические И (and) и ИЛИ (or).

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.

Оператор цикла while

Операторы break и continue

Оператор цикла for

14. Назовите назначение и способы применения функции range .

Функция range возвращает неизменяемую последовательность чисел в виде объекта range.

start - с какого числа начинается последовательность. По

умолчанию - 0

stop - до какого числа продолжается последовательность чисел.

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?

range(15, 0[, -2])

16. Могул ли быть циклы вложенными?

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

```
a = 0
while a >= 0:
if a == 7:
break
a += 1
print("A")
```

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

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?

В операционной системе по умолчанию присутствуют стандартных потока вывода на консоль: буферизованный поток stdout для вывода данных и информационных сообщений, а также не буферизованный поток stderr для вывода сообщений об

ошибках.

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

Для того, чтобы использовать поток stderr необходимо передать его в

параметре file функции print.

22. Каково назначение функции exit ?

В Python завершить программу и передать операционной системе заданный код возврата можно посредством функции exit.

Вывод:

Приобрел навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Освоил операторы языка Python.3 if, while, for, break, continue, позволяющих реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры