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»
Вариант 9

Выполнил:
Дудкин Константин Александрович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»
направление «Программное
обеспечение средств вычислительной
техники и автоматизированных
систем»,
очная форма обучения

_____
(подпись)

Руководитель практики:
Кандидат технических наук, доцент
кафедры инфокомуникаций, доцент
Воронкин Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: Условные операторы и циклы в языке Python

Цель: Приобретение навыков программирования разветвляющихся 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 пример 1: Составить UML-диаграмму деятельности и программу с использованием конструкции ветвления и вычислить занчение функции

$$y = \begin{cases} 2x^2 + \cos x, & x \le 3.5, \\ x + 1, & 0 < x < 5, \\ \sin 2x - x^2, & x \ge 5. \end{cases}$$

Рисунок 1. Уравнение примера 1

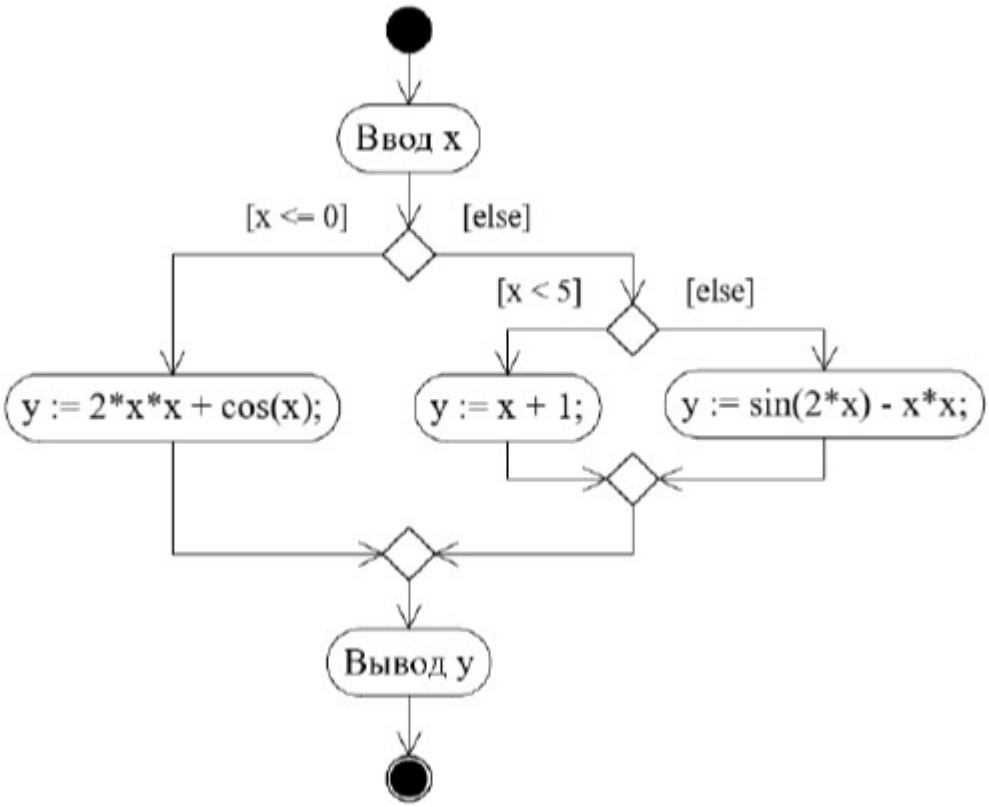


Рисунок 2. UML диаграмма примера 2

```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 3. Код примера 1

2. Проработал пример 2: Составить UML-диаграмму деятельности и программу для решения задачи: с клавиатуры вводится номер месяца от 1 до 12, необходимо для этого номера месяца вывести наименование времени года

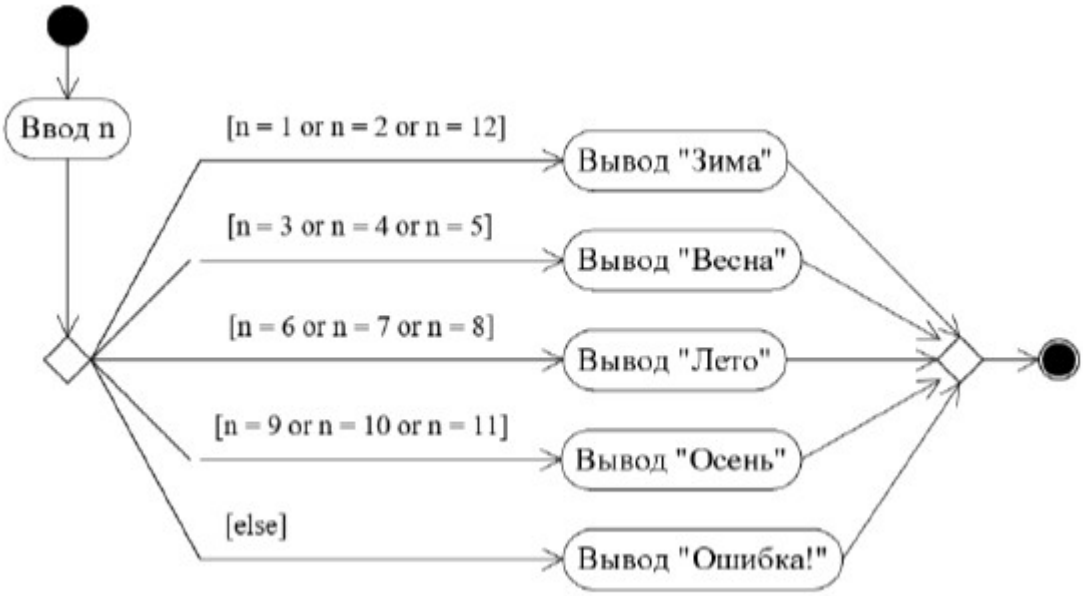


Рисунок 4. UML диаграмма примера 2

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

Рисунок 5. Код примера 2

3. Проработал пример 3: Составить UML-диаграмму деятельности и написать программу, позволяющую вычислить конечную сумму:

$$S = \sum_{k=1}^{n} \frac{\ln kx}{k^2},$$

Рисунок 6. Уравнение примера 3

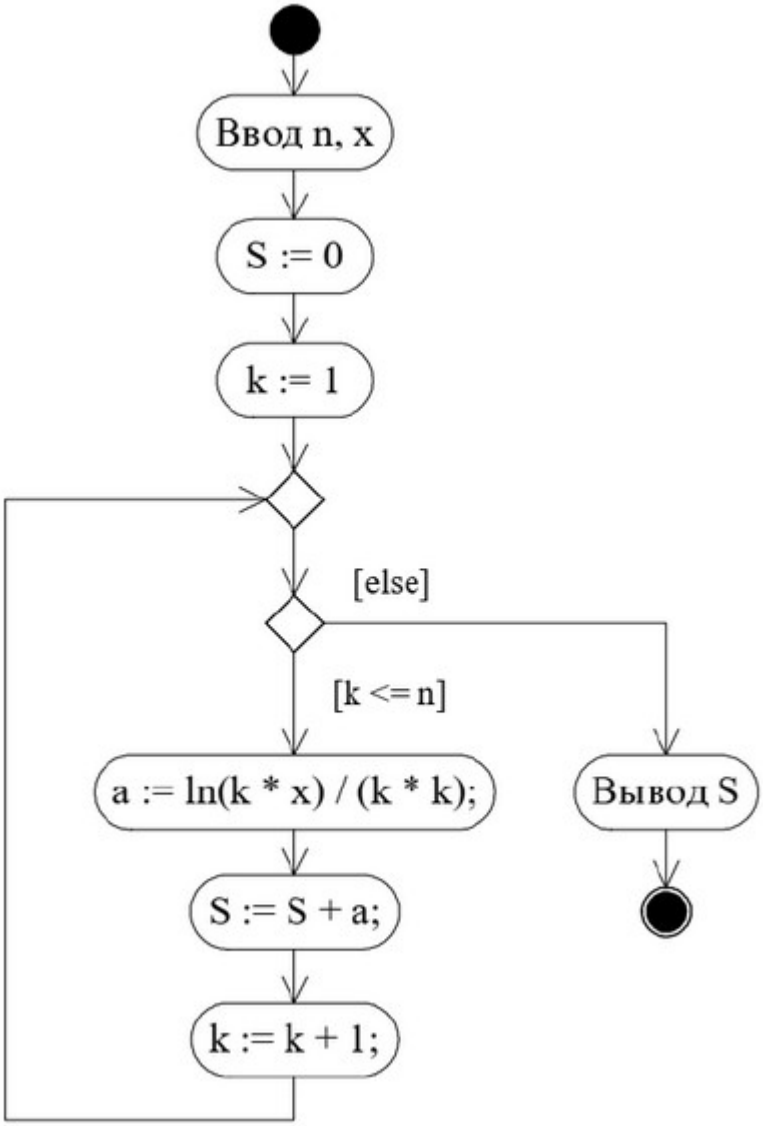


Рисунок 7. UML-диаграмма примера 3

```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
    for k in range(1, n + 1):
        a = math.log(k * x) / (k * k)
        S += a
    print(f"S = {S}")
```

Рисунок 8. Код примера 3

4. Проработал пример 4: Найти значение квадратного корня $x=\sqrt{a}$ из положительного числа a, вводимого с клавиатуры, с некоторой заданной точностью ε с помощью рекуррентного соотношения

$$x_{n+1} = \frac{1}{2} \cdot \left(x_n + \frac{a}{x_n} \right).$$

Рисунок 9. Уравнение

примера 4

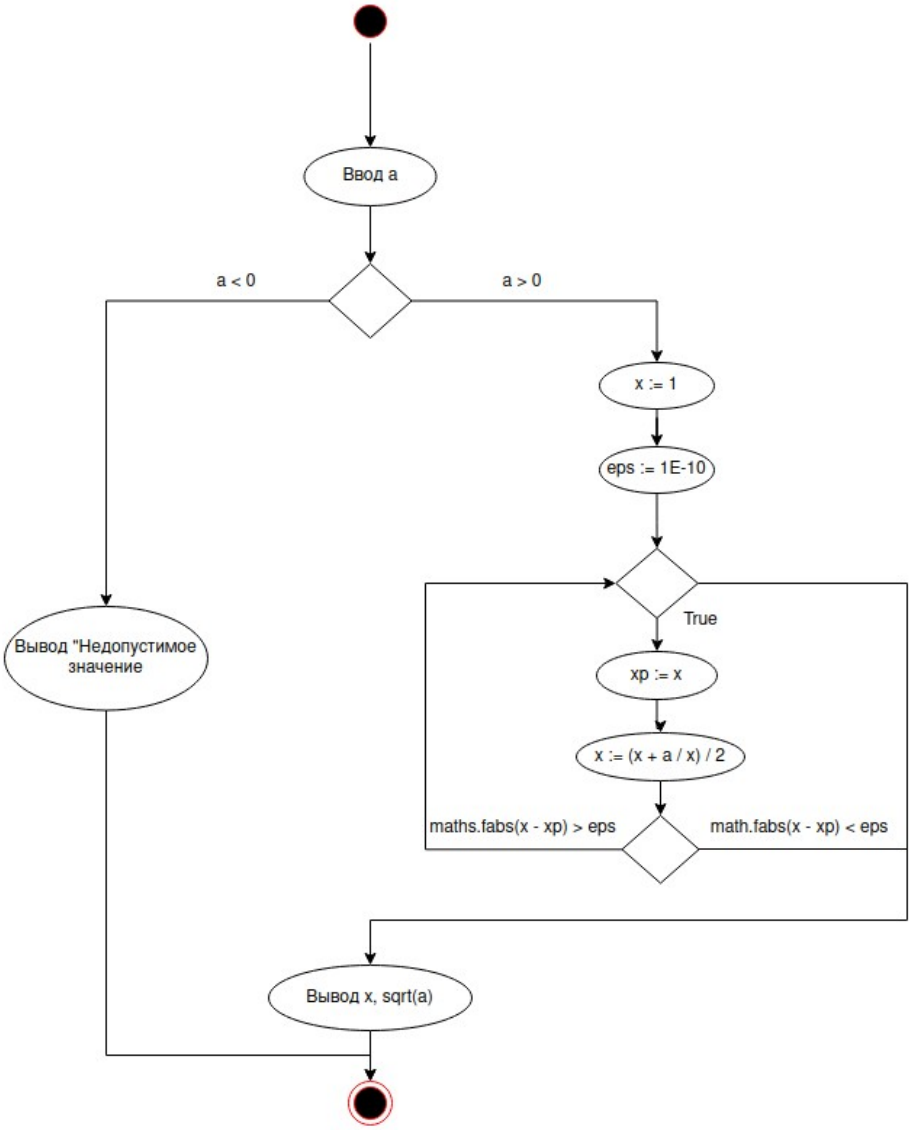
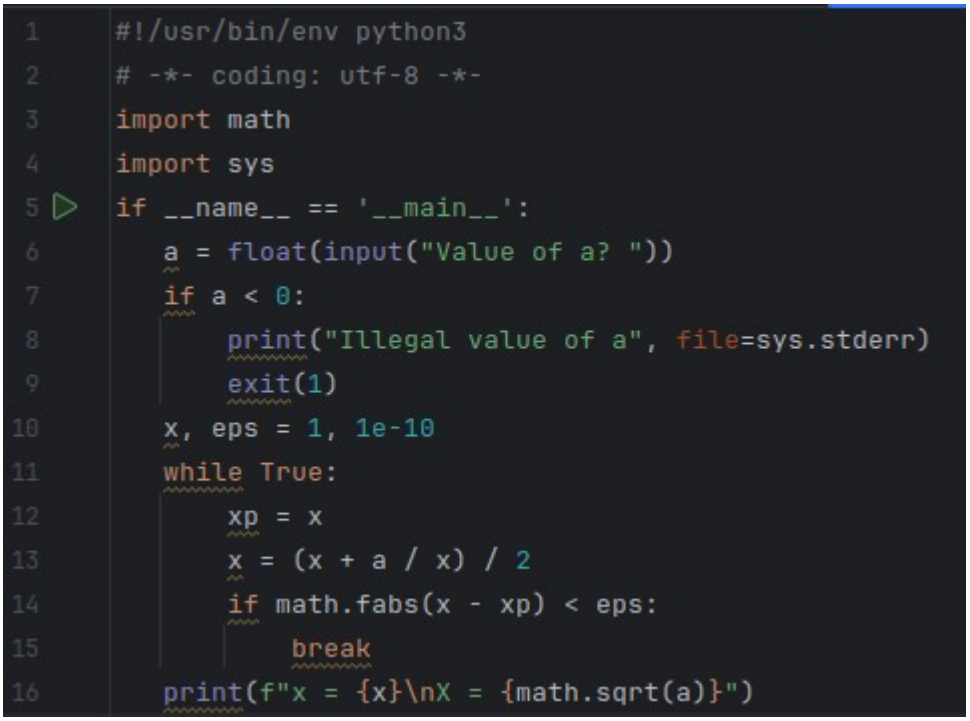


Рисунок 10. UML-диаграмма примера 4

Рисунок



```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)}")
```

11. Код примера 4

5. Проработал пример 5: Вычислить значение специальной (интегральной показательной) функции

$$\mathrm{Ei}(x) = \int_{-\infty}^{x} \frac{\exp t}{t}\,dt = \gamma + \ln x + \sum_{k=1}^{\infty} \frac{x^k}{k \cdot k!},$$

Рисунок 12. Уравнение примера 5

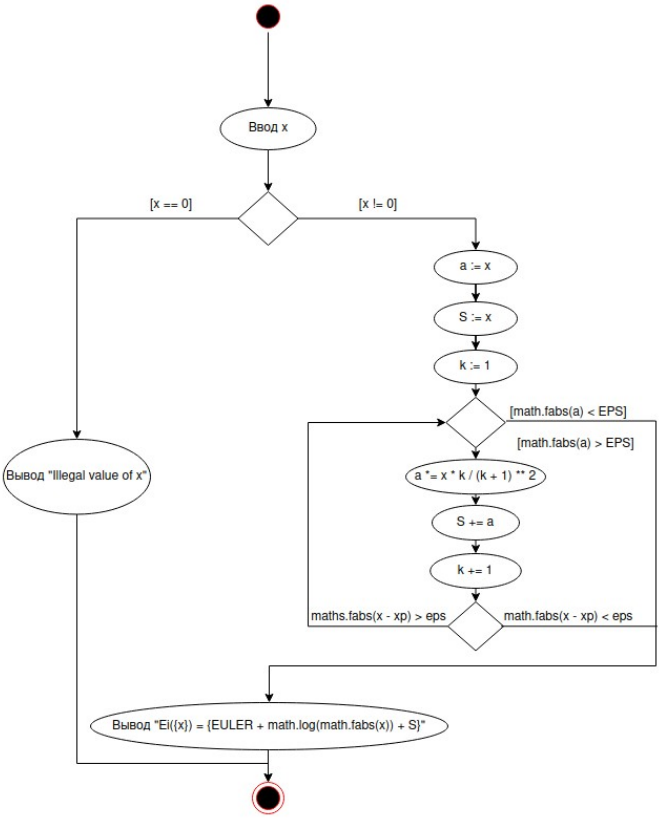


Рисунок 13. UML-диаграмма примера 5

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import math
import sys
# Постоянная Эйлера.
EULER = 0.5772156649015329
# 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 14. Код примера 5

6. Выполнил индивидуальное задание 1 в соответствии со своим вариантом: Вводится число экзаменов N <= 20. Напечатать фразу «Мы успешно сдали N экзаменов», согласовав слово «экзамен» с числом N

```python
exams = int(input("Сколько экзаменов мы сдали? "))

if exams > 20:
    print("Слишком много экзаменов!")
elif exams == 1:
    print("Мы успешно сдали 1 экзамен")
elif exams >= 2 and exams <= 4:
    print("Мы успешно сдали", exams, "экзамена")
elif exams >= 5:
    print("Мы успешно сдали", exams, "экзаменов")
```

Рисунок 16. Код задания 1

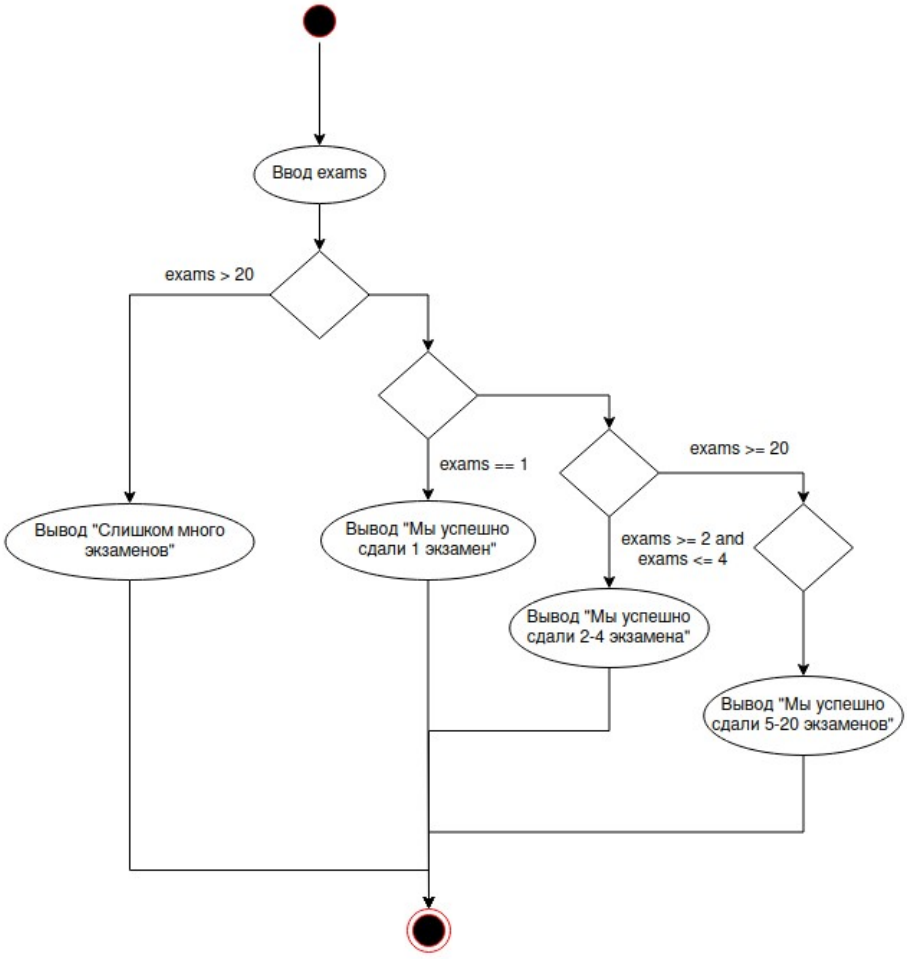


Рисунок 17. UML-диаграмма задания 1

7. Выполнил индивидуальное задание 2: Найти координаты точки пересечения прямых заданных уравнениями $a_1x+b_1y+c_1=0$ и $a_2x+b_2y+c_2=0$, либо сообщить совпадают, параллельны или не существуют

```python
a1 = int(input("Введите координату a1: "))
b1 = int(input("Введите координаду b1: "))
c1 = int(input("Введите координату c1: "))
a2 = int(input("Введите координату a2: "))
b2 = int(input("Введите координату b2: "))
c2 = int(input("Введите координату c2: "))

if a1 / a2 == b1 / b2:
    if c1 / a1 == c2 / a2:
        print("Прямые совпадают")
    else:
        print("Прямые параллельны")
else:
    x = (b1 * c2 - b2 * c1) / (a1 * b2 - a2 * b1)
    y = (a2 * c1 - a1 * c2) / (a1 * b2 - a2 * b1)
    print("Точка пересечения: (",x,",",y,")")
```

Рисунок 18. Код задания 2

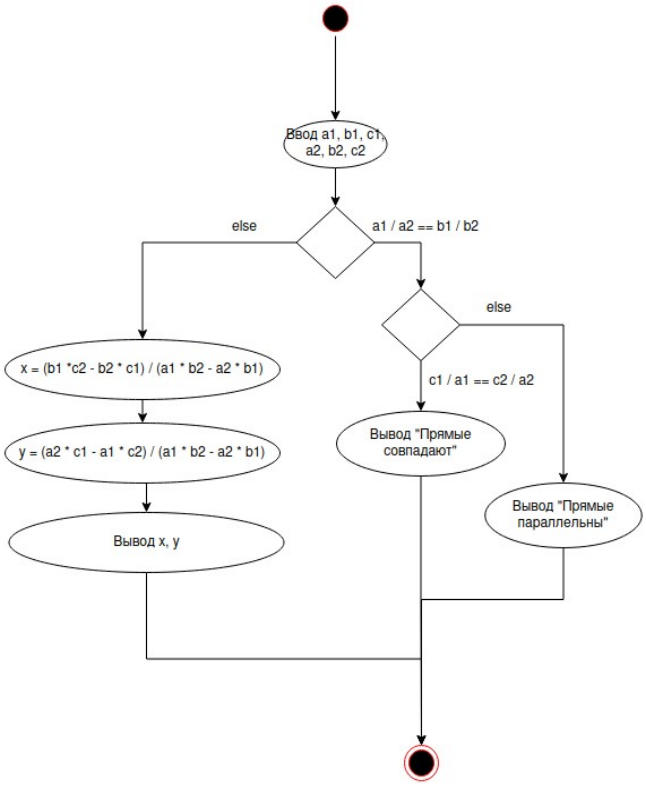


Рисунок 19. UML-диаграмма задания 2

8. Выполнил индивидуальное задание 3: Если к сумме цифр двухзначного числа прибавить квадрат этой суммы, то снова получится это двузначное число. Найти все эти числа

```
dec = 9
count = 0
for i in range (1, 9):
    for j in range (0, 9):
        dec += 1


print("                                    чисел")
```

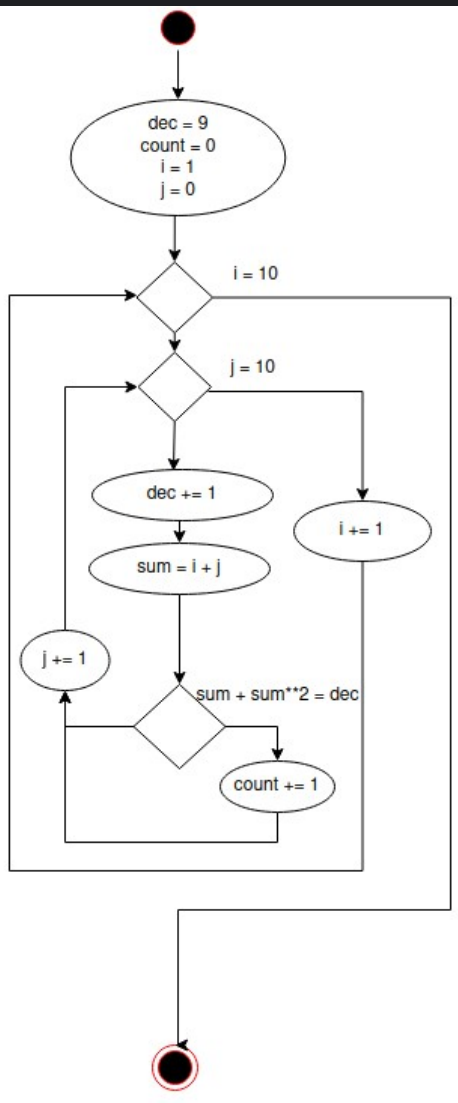


Рисунок 20. Код задания 3

Рисунок 21. UML-диаграмма задания 3

9. Сделал коммит сделанных заданий, выполнил слияние ветки Develop и main. Отправил коммит на GitHub

Ответы на вопросы

1. Диаграммы деятельности UML нужны для визуализации последовательности действий и процессов в системе. Они помогают описать, какие действия выполняются, какие ресурсы используются и какие условия должны быть соблюдены для перехода к другому состоянию. Чаще всего их применяют в бизнес-аналитике, в проектировании бизнес-процессов и в разработке ПО

2. Состояние действия в UML-диаграммах деятельности представляет собой конкретное действия, выполняемое объектом в определенный момент времени. Оно может быть представлено в виде прямоугольника с закругленными углами и действием внутри него

Состояние деятельности — это набор связанных действий, выполняемых последовательно ли параллельно. Оно может быть представлено так же, как и состояние действия

3. Для ветвлений и переходов в UML-диаграммах деятельности используются стрелки с направлением последующих действий и ромбы для ветвлений в случаях, где действия удовлетворяют или не удовлетворяют условиям задачи

4. Алгоритм разветвляющейся структуры — алгоритм, включающий в себя ветвления и условны операторы для принятия решений в зависимости он заданных условий. Он помогает решить вопрос неясности действий или же показать определённый результат при разных значениях переменных.

5. Разветвляющийся алгоритм отличается от линейного вариативностью действий при определенных условиях т.к. в линейном алгоритме все действия изначально заданы по порядку без вариантов другого исхода

6. Условный оператор — оператор, который позволяет программе принимать решения о последующих действиях на основе заданных условий. В Python условными операторами являются «if», «else» и «elif».

7. В Python существуют такие операторы сравнения, как:

- == (равно) — проверяет равность двух значений

- != (не равно) — проверяет неравность двух значений

- > (больше) — проверяет, является ли первое значение больше второго

- < (меньше) — проверяет, является ли первое значение меньше второго

- >= (больше или равно) — проверяет, является и первое значение больше или равным второму

- <= (больше или меньше) — проверяет, является ли первое значение меньше или равным второму

8. Простое условия — условие, содержащее всего одно сравнение или проверку. В них используются операторы сравнения для сравнения значений и принятия решения на их основе

9. Составное условие — условие, состоящее из нескольких простых условий. Оно позволяет проверить сразу несколько условий одновременно, принимая решение на основе их комбинации

10. В сложных условиях используются следующие логические операторы:

- and — логическое «и»

- or — логическое «или»

- not — логическое «не»

11. Оператор ветвления может включать в себя несколько других операторов ветвления. Такое ветвление называется сложным

12. Алгоритм циклической структуры — алгоритм, содержащий внутри себя цикл для повторений определённых действий. В Python для создания циклов используются операторы «for» и «while»

13. В Python представлены следующие операторы циклов:

- Цикл for — цикл выполняется до тех пор, пока заданное в нем значение не дойдет до заданного предела. В случае его преувелечения/преуменьшения/равности (зависит от условий) цикл завершается

- Цикл while — цикл выполняется до тех пор, пока указанное значение истинно. Обычно он используется тогда, когда неизвестно заранее количество необходимых повторений

14. Функция range используется для создания последовательности чисел. Она имеет следующий синтаксис: range(start, stop, step), где:

- start — начальное значение

- stop — конечное значение

- step — шаг изменения значения

Функция range может использоваться для создания циклов и перебора значений, а так же в тех задачах, где требуется последовательность чисел

15. Чтобы организовать перебор значений от 15 до 0 с шагом 2 нужно использовать следующий код: for i in range(15, -1, -2)

16. Циклы могут быть вложенными т.е. один цикл может выполняться внутри другого цикла

17. Бесконечный цикл образуется тогда, когда значение цикла всегда является истинным. Примером такого цикла является значение True в цикле while или отсутствие изменений переменной цикла, указанной в нем

Чтобы выйти из цикла нужно либо использовать оператор break или условия, которое станет ложным в определенный момент времени

18. Оператор break используется для выхода из цикла, вне зависимости от того, было ли значение цикла истинным или ложным. Его полезность заключается в завершении цикла тогда, когда изменения определенных значений больше не потребуется

19. Оператор continue используется в циклах для пропуска текущей итерации цикла и перехода к следующей. Он полезен тогда, когда в необходимом случае не требуется изменений значений внутри цикла, не прерывая при этом сам цикл

20. Стандартные потоки stdout и stderr используются для вывода определенной информации из программы:

stdout (стандартный вывод) используется для вывода обычной информации или результатов работы программы. Этот поток обычно направляется в консоль или другое устройство вывода

stderr (стандартный вывод ошибок) используется для вывода сообщений об ошибках или другой информации об ошибках. Он тоже направляется в консоль, но его также можно направить и в файл или же в другой поток

21. В Python для организации вывода в стандартный поток stderr можно использовать модуль sys. В нем присутствуют команды и операторы,

позволяющие выводить сообщения об ошибках на консоль или друге устройство вывода

22. Функция exit позволяет завершить программу. Когда эта функция вызвана, программа завершается, выводя код завершения

Данная функция полезна в случаях, когда в указанном случае действия программы больше не требуется или когда появляется ошибка, для предотвращения которой требуется срочные действия

Вывод: В данной работе я приобрел навыки программирования разветвляющихся и циклических алгоритмов, были подробно изучены операторы if, while, for, break и contunue