

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 29

Выполнил:
Саенко Андрей Максимович
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», заочная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., канд. технических
наук, доцент кафедры
инфокоммуникаций

(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 и continue, позволяющих реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.

Порядок выполнения работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

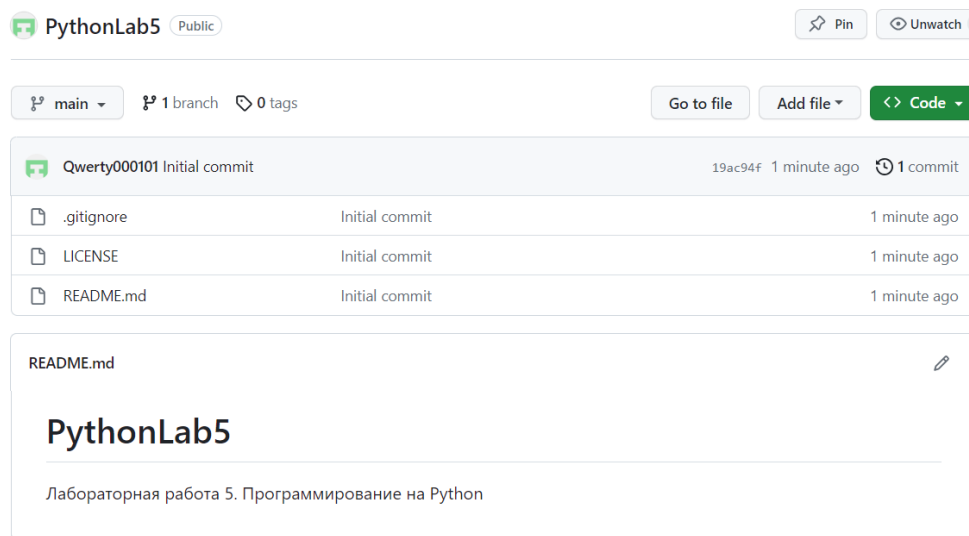


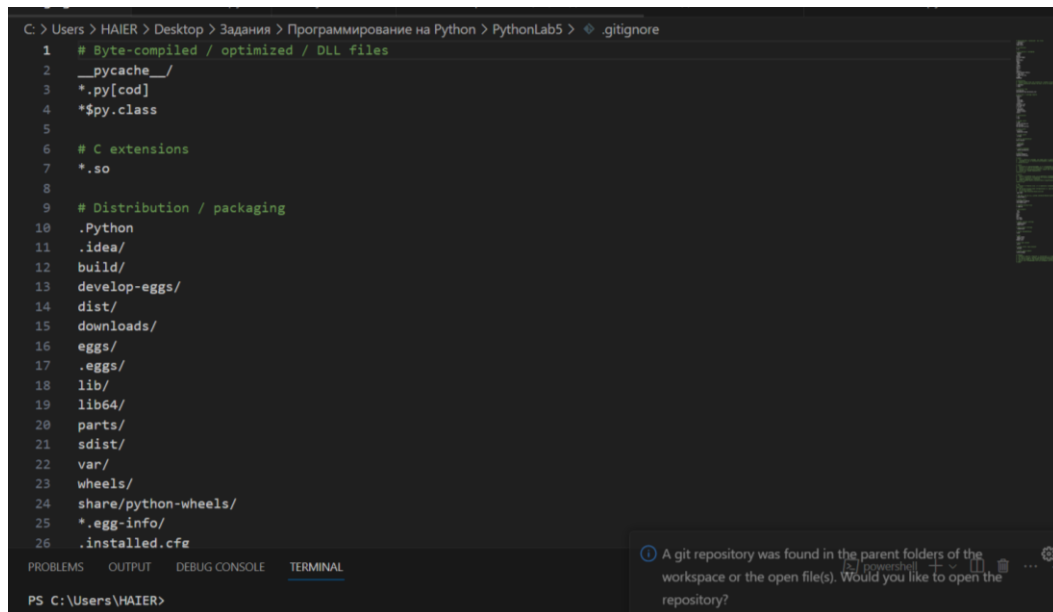
Рисунок 1 – Общедоступный репозиторий на GitHub

3. Выполните клонирование созданного репозитория.

```
Git CMD
C:\Users\HAIER>cd C:\Users\HAIER\Desktop\Задания\Программирование на Python
C:\Users\HAIER\Desktop\Задания\Программирование на Python>git clone https://github.com/Qwerty000101/PythonLab5.git
Cloning into 'PythonLab5'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
C:\Users\HAIER\Desktop\Задания\Программирование на Python>
```

Рисунок 2 – Клонирование репозитория

4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.

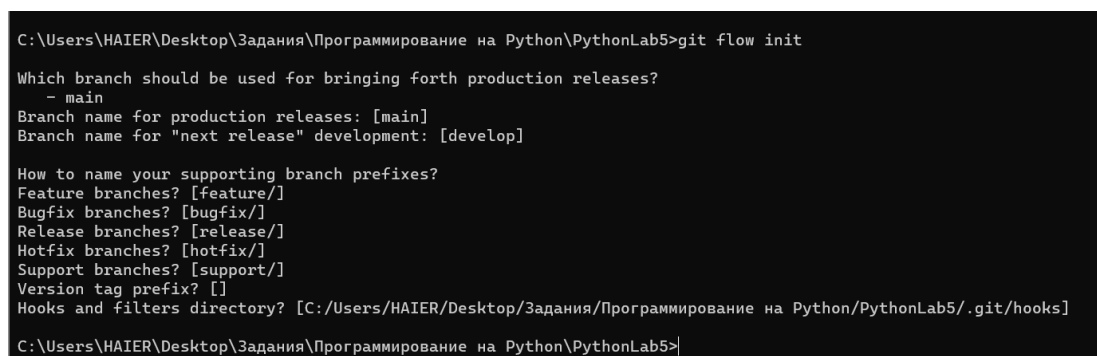


```
C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab5> .gitignore
1 # Byte-compiled / optimized / DLL files
2 __pycache__ /
3 *.py[.cod]
4 *$py.class
5
6 # C extensions
7 *.so
8
9 # Distribution / packaging
10 .Python
11 .idea/
12 build/
13 develop-eggs/
14 dist/
15 downloads/
16 eggs/
17 .eggs/
18 lib/
19 lib64/
20 parts/
21 sdist/
22 var/
23 wheels/
24 share/python-wheels/
25 *.egg-info/
26 .installed.cfg
```

At the bottom of the editor, a notification bubble states: "A git repository was found in the parent folders of the workspace or the open file(s). Would you like to open the repository?"

Рисунок 3 – Дополнение файла .gitignore

5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.



```
C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab5>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/HAIER/Desktop/Задания/Программирование на Python/PythonLab5/.git/hooks]

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab5>
```

Рисунок 4 – Организация репозитория в соответствии с git-flow

6. Самостоятельно изучите рекомендации к оформлению исходного кода на языке Python PEP-8 (<https://pep8.org/>). Выполните оформление исходного примеров лабораторной работы и индивидуальных созданий в соответствии с PEP-8.

7. Создайте проект PyCharm в папке репозитория.

Имя	Дата изменения	Тип	Размер
.git	22.11.2023 10:03	Папка с файлами	
PyCharm_Project	22.11.2023 10:06	Папка с файлами	
.gitignore	22.11.2023 0:23	Исходный файл G...	4 КБ
LICENSE	22.11.2023 9:53	Файл	2 КБ
README.md	22.11.2023 9:53	Исходный файл ...	1 КБ

Рисунок 5 – Проект PyCharm в папке репозитория

8. Проработайте примеры лабораторной работы. Создайте для каждого примера отдельный модуль языка Python. Зафиксируйте изменения в репозитории.

Пример 1.

Результат работы программы:

```
Value of x? 4
y = 5.0

Process finished with exit code 0
```

Рисунок 6 – Результат работы программы

```
Value of x? 10
y = -100.54402111088937

Process finished with exit code 0
```

Рисунок 7 – Результат работы программы

Код программы:

```
#!/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

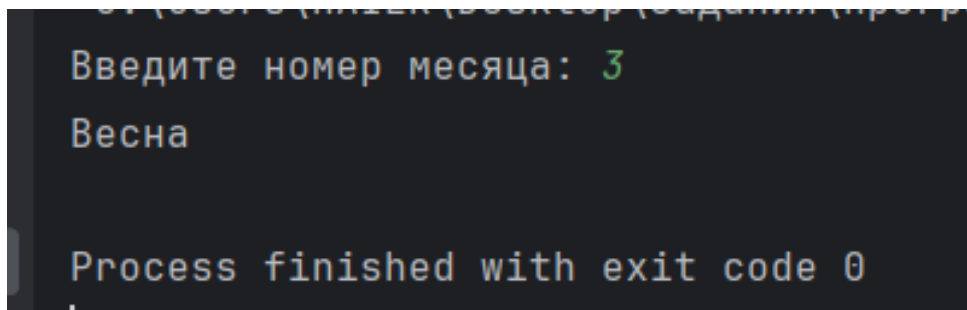
    else:
        y = math.sin(x) - x*x

    print(f'y = {y}')

```

Пример 2.

Результат работы программы:



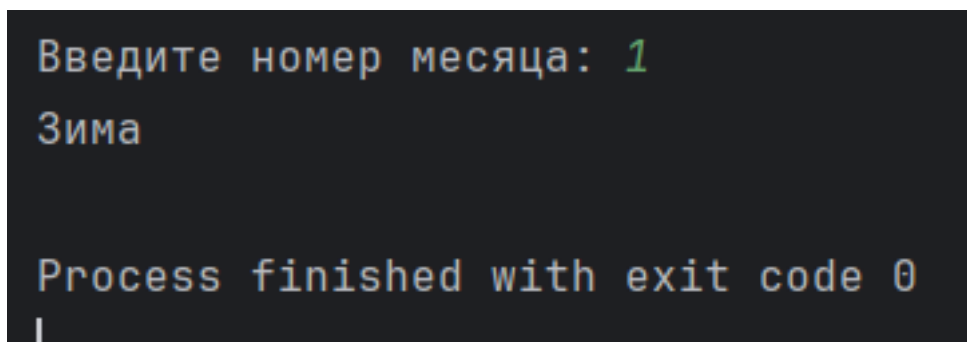
```

Введите номер месяца: 3
Весна

Process finished with exit code 0

```

Рисунок 8 – Результат работы программы



```

Введите номер месяца: 1
Зима

Process finished with exit code 0

```

Рисунок 9 – Результат работы программы

Код программы:

```

#!/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:

```

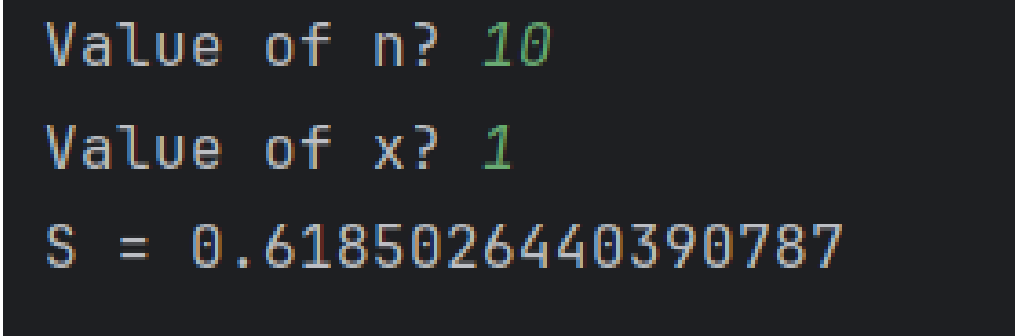
```

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)

```

Пример 3.

Результат работы программы:

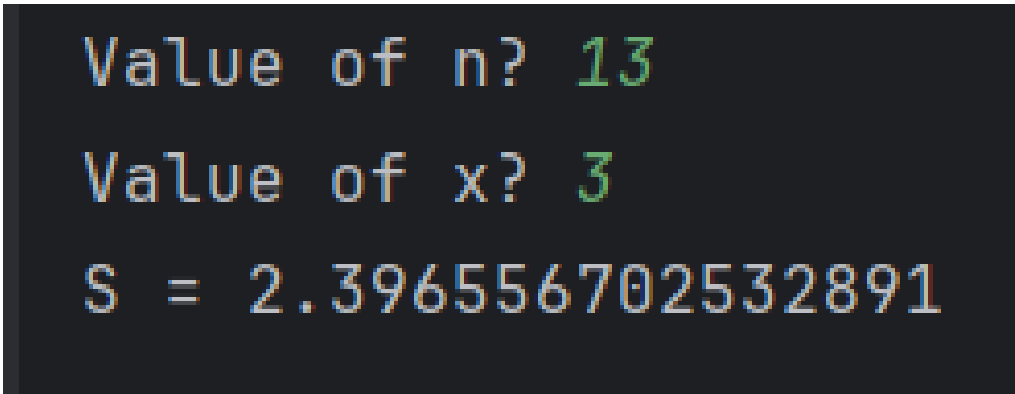


```

Value of n? 10
Value of x? 1
S = 0.6185026440390787

```

Рисунок 10 – Результат работы программы



```

Value of n? 13
Value of x? 3
S = 2.396556702532891

```

Рисунок 11 – Результат работы программы

Код программы:

```

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

import math

if __name__ == '__main__':
    n = int(input("Value of n? "))
    x = float(input("Value of x? "))

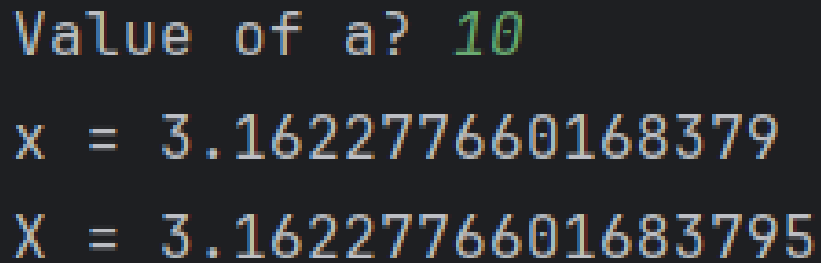
    S = 0.0
    for k in range(1, n + 1):
        a = math.log(k * x) / (k * k)
        S += a

```

```
print(f'S = {S}')
```

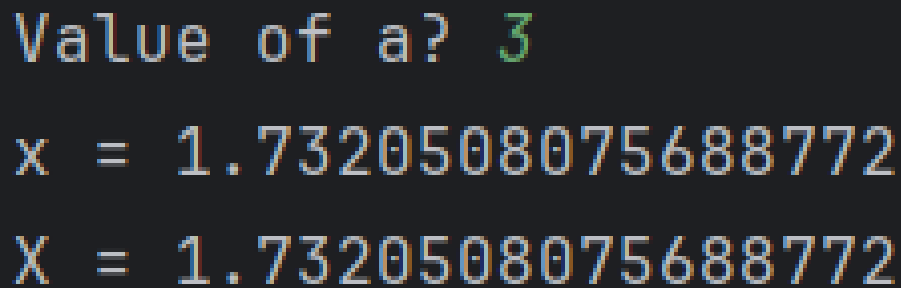
Пример 4.

Результат работы программы:

A screenshot of a terminal window with a dark background. It shows the output of a program where the variable 'a' is 10. The text is displayed in a monospaced font with a rainbow-colored shadow effect. The output consists of three lines: 'Value of a? 10', 'x = 3.162277660168379', and 'X = 3.1622776601683795'.

```
Value of a? 10  
x = 3.162277660168379  
X = 3.1622776601683795
```

Рисунок 12 – Результат работы программы

A screenshot of a terminal window with a dark background. It shows the output of a program where the variable 'a' is 3. The text is displayed in a monospaced font with a rainbow-colored shadow effect. The output consists of three lines: 'Value of a? 3', 'x = 1.7320508075688772', and 'X = 1.7320508075688772'.

```
Value of a? 3  
x = 1.7320508075688772  
X = 1.7320508075688772
```

Рисунок 13 – Результат работы программы

Код программы:

```
#!/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)}')
```

UML-диаграмма:

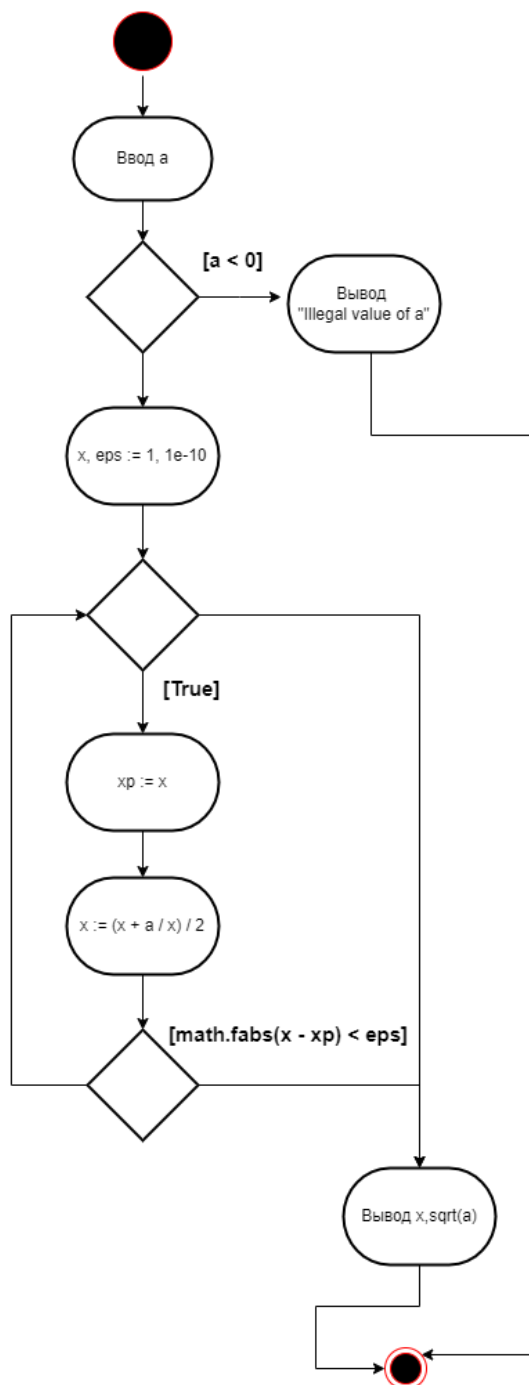
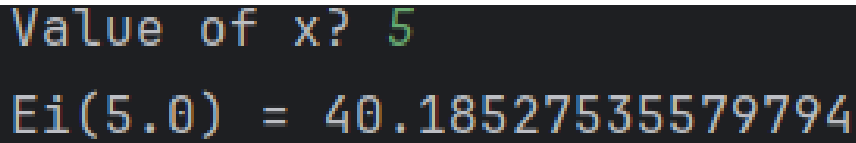


Рисунок 10 – UML-диаграмма для примера 4

Пример 5.

Результат работы программы:

A screenshot of a terminal window with a dark background. It shows the text "Value of x? 5" on the first line and "Ei(5.0) = 40.18527535579794" on the second line. The text is rendered in a monospaced font with a light blue/cyan color.

```
Value of x? 5
Ei(5.0) = 40.18527535579794
```

Рисунок 14 – Результат работы программы

A screenshot of a terminal window with a dark background. It shows the text "Value of x? 2" on the first line and "Ei(2.0) = 4.954234355999365" on the second line. The text is rendered in a monospaced font with a light blue/cyan color.

```
Value of x? 2
Ei(2.0) = 4.954234355999365
```

Рисунок 15 – Результат работы программы

Код программы:

```
#!/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}')
```

UML-диаграмма:

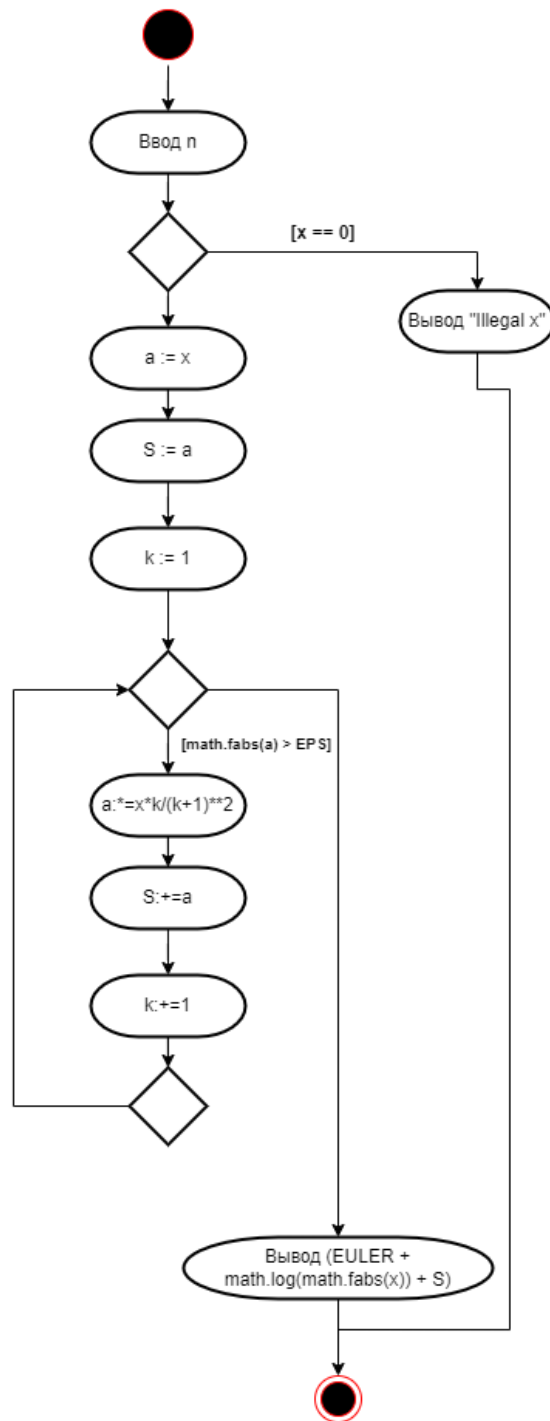


Рисунок 16 – UML-диаграмма для примера 5

9. Выполните индивидуальные задания, согласно своего варианта. Для заданий повышенной сложности номер варианта должен быть получен у преподавателя.

Индивидуальные задания:

Задание 1.

3. Дано число ($1 \leq m \leq 7$). Вывести на экран название дня недели, который соответствует этому номеру.

Решение:

Результат работы программы:

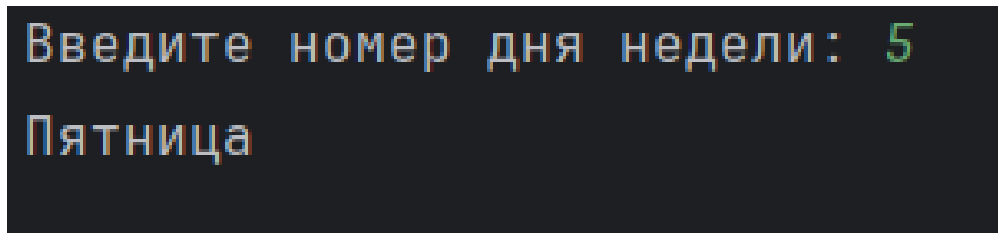


Рисунок 13 – Результат работы программы

Код программы:

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

import sys

if __name__ == '__main__':
    n = int(input("Введите номер дня недели: "))
    if n == 1:
        print("Понедельник")
    elif n == 2:
        print("Вторник")
    elif n == 3:
        print("Среда")
    elif n == 4:
        print("Четверг")
    elif n == 5:
        print("Пятница")
    elif n == 6:
        print("Суббота")
    elif n == 7:
        print("Воскресенье")
    else:
        print("Ошибка!", file=sys.stderr)
        exit(1)
```

UML-диаграмма:

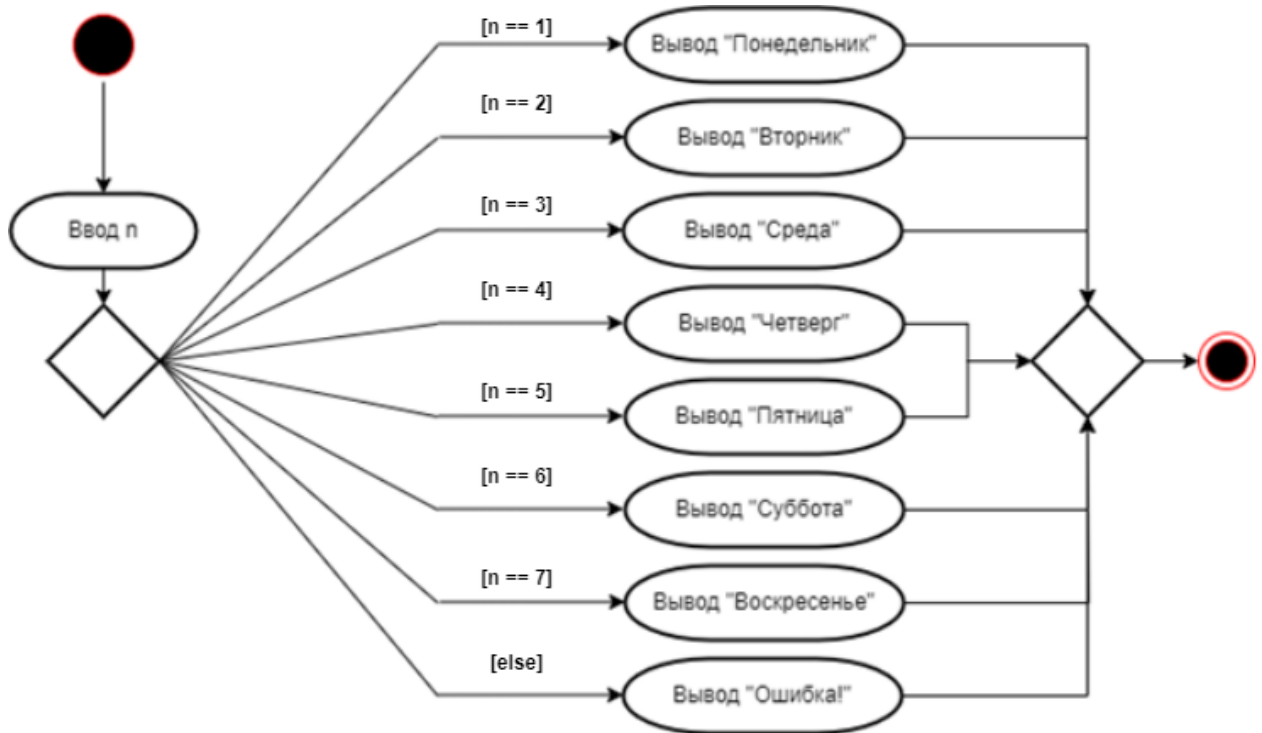


Рисунок 14 – UML-диаграмма для задания 1

Задание 2.

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

Результат работы программы:

```
Введите a: 1
Введите b: -3
Введите c: -18
Ответ: (-inf,-3.0) and (6.0;inf)

Process finished with exit code 0
```

Рисунок 15 – Результат работы программы

Код программы:

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

import math

if __name__ == '__main__':
    a = int(input("Введите a: "))
    b = int(input("Введите b: "))
    c = int(input("Введите c: "))
    D = b * b - 4 * a * c

    if D < 0 and c > 0:
        print("(-inf;inf)")
    elif D < 0 and c < 0:
        print("Пустое множество")
    else:
        x_first = (-b - math.sqrt(D))/(2*a)
        x_second = (-b + math.sqrt(D)) / (2 * a)
        if x_first > x_second:
            x_first, x_second = x_second, x_first
        if a < 0:
            print(f"Ответ: ({x_first};{x_second})")
        else:
            print(f"Ответ: (-inf,{x_first}) and ({x_second};inf)")
```

UML-диаграмма:

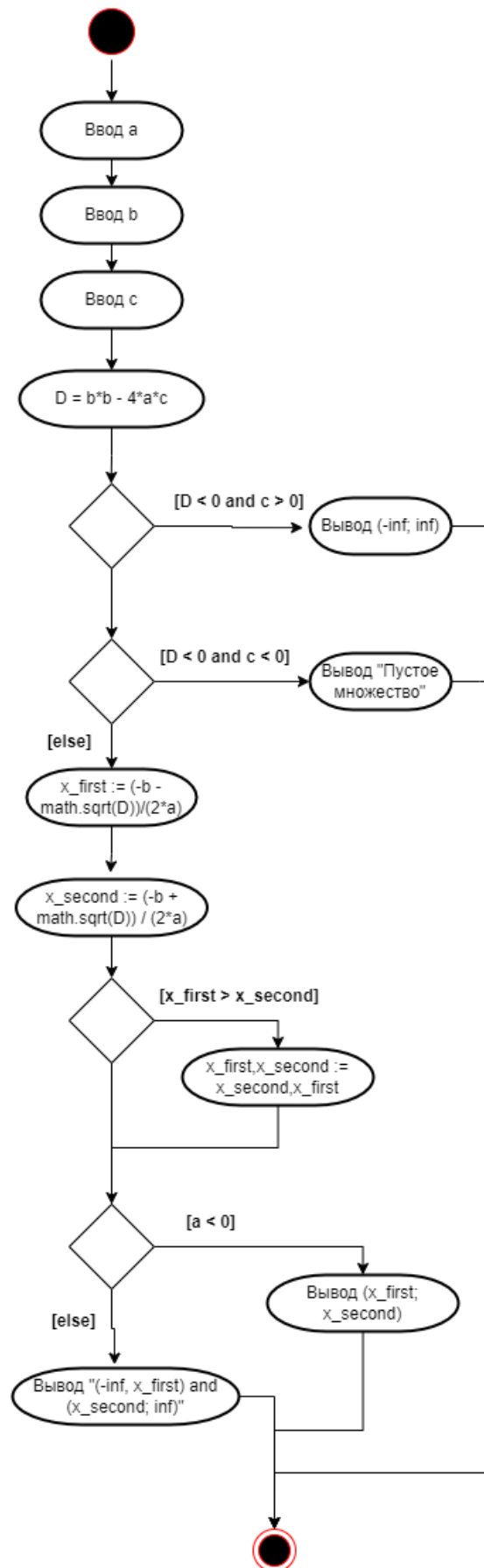
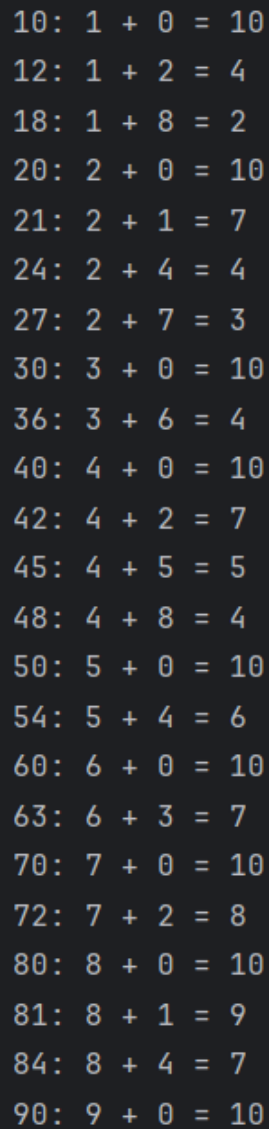


Рисунок 16 – UML-диаграмма для задания 2

Задание 3.

7. Определить среди всех двузначных чисел те, которые делятся на сумму своих цифр.

Результат работы программы:



```
10: 1 + 0 = 10
12: 1 + 2 = 4
18: 1 + 8 = 2
20: 2 + 0 = 10
21: 2 + 1 = 7
24: 2 + 4 = 4
27: 2 + 7 = 3
30: 3 + 0 = 10
36: 3 + 6 = 4
40: 4 + 0 = 10
42: 4 + 2 = 7
45: 4 + 5 = 5
48: 4 + 8 = 4
50: 5 + 0 = 10
54: 5 + 4 = 6
60: 6 + 0 = 10
63: 6 + 3 = 7
70: 7 + 0 = 10
72: 7 + 2 = 8
80: 8 + 0 = 10
81: 8 + 1 = 9
84: 8 + 4 = 7
90: 9 + 0 = 10
```

Рисунок 17 – Результат работы программы

Код программы:

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

import math

if __name__ == '__main__':
    i = 10
    while i <= 100:
        if i%(int(str(i)[0])+int(str(i)[1])) == 0:
            result = int(i / (int(str(i)[0]) + int(str(i)[1])))
```

```
print(f"{i}: {int(str(i)[0])} + {int(str(i)[1])} = {result}")  
i += 1
```

UML-диаграмма:

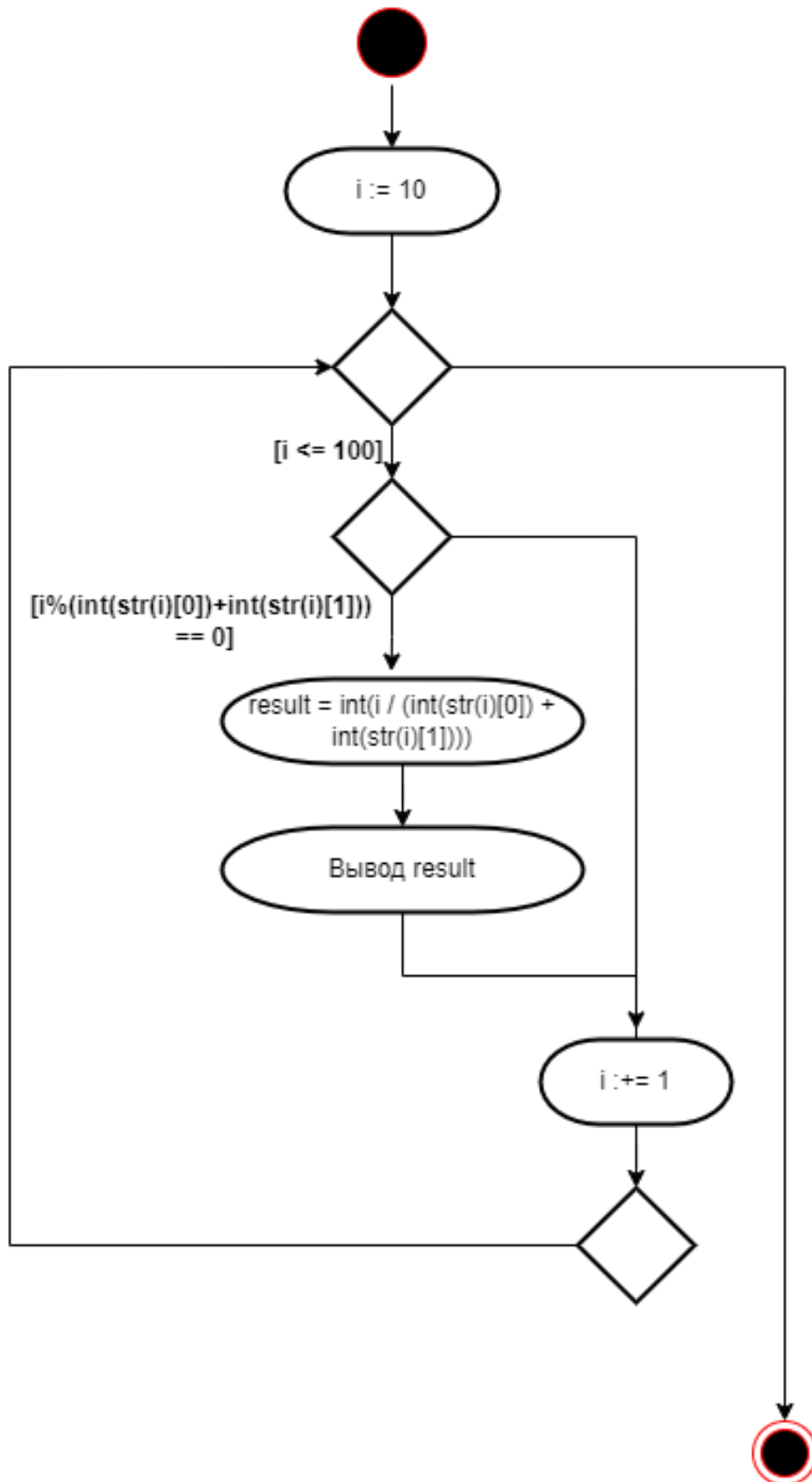


Рисунок 18 – UML-диаграмма для задания 3

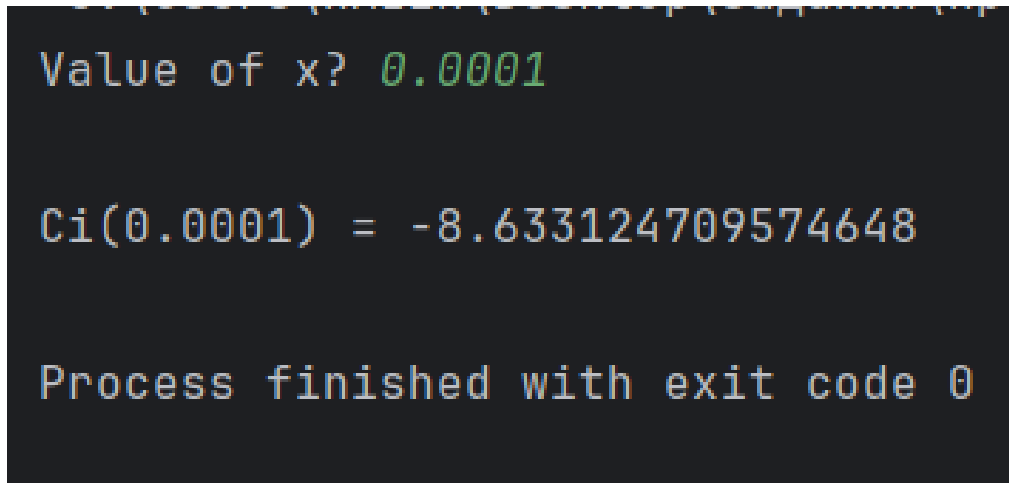
Задание повышенной сложности:

2. Интегральный косинус:

$$\text{Ci}(x) = \gamma + \ln x + \int_0^x \frac{\cos t - 1}{t} dt = \gamma + \ln x + \sum_{n=1}^{\infty} \frac{(-1)^n x^{2n}}{(2n)(2n)!}.$$

Рисунок 19 – Функция из задания повышенной сложности

Результат работы программы:



```
Value of x? 0.0001

Ci(0.0001) = -8.633124709574648

Process finished with exit code 0
```

Рисунок 20 – Результат работы программы

Код программы:

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

from decimal import *
import math
import sys

getcontext().prec = 4096
EULER = 0.5772156649015328606
EPS = 1e-10

if __name__ == '__main__':
    x = Decimal(input("Value of x? "))

    if x <= 0:
        print("Illegal value of x", file=sys.stderr)
        exit(1)

    S = Decimal(0)
    a = Decimal(1)
```

```

n = Decimal(1)

while math.fabs(a) > EPS:
    e0 = (-1)**n
    e1 = x**(2*n)
    e2 = 2*n
    e3 = Decimal(1)
    temp = 2*n
    while temp > 1:
        e3 *= temp
        temp -= 1
    a = (e0*e1)/(e2*e3)
    n += 1
    S += a

print(f"\nCi(x) = {EULER + math.log(x) + float(S)}")

```

UML-диаграмма:

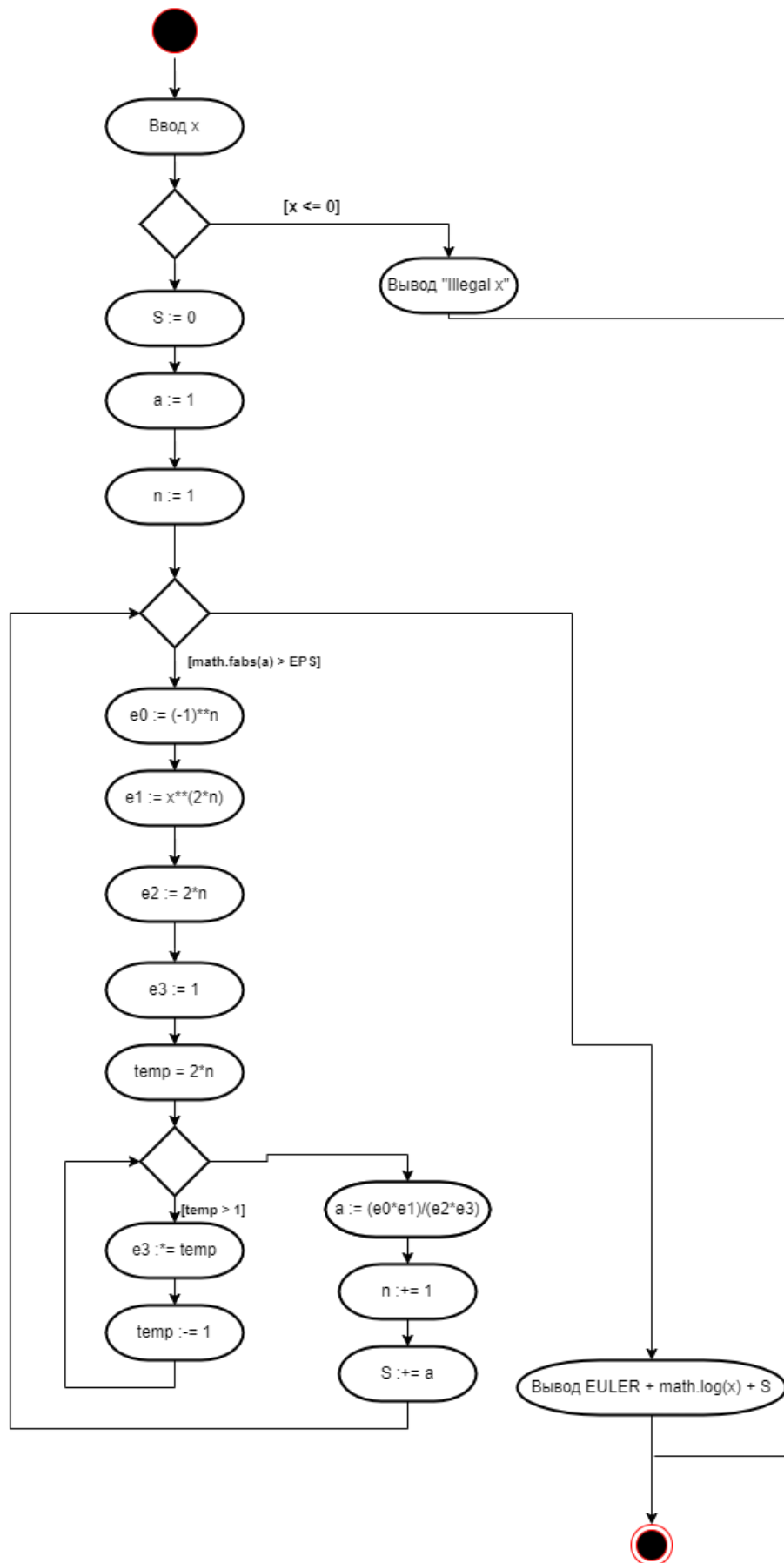


Рисунок 21 – UML-диаграмма для задания повышенной сложности

Ответы на 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?

Выполняемые атомарные вычисления называются состояниями действия, поскольку каждое из них есть состояние системы, представляющее собой выполнение некоторого действия.

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?

В диаграммах деятельности существуют следующие нотации:

- 1) Стрелки, которые указывают на переход между состояниями;
- 2) Ромбы, которые используются для обозначения ветвлений и объединений;
- 3) Условия переходов (указываются рядом со стрелками и ромбами).

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Алгоритм разветвляющейся структуры – алгоритм, который содержит ветвление, в языке Python оно может осуществляться с помощью конструкции «if-else»

5. Чем отличается разветвляющийся алгоритм от линейного?

Разветвляющийся алгоритм отличается от линейного тем, что содержит ветвление. Линейный алгоритм не содержит подобных конструкций и выполняется линейно.

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

Условный оператор – оператор, который в зависимости от некоторого условия выполняет определённый набор инструкций. Существуют следующие формы условного оператора:

- 1) Оператор «if».
- 2) Конструкция «if-else».
- 3) Конструкция «if-elif-else».
7. Какие операторы сравнения используются в Python?

В языке программирования Python используются следующие операторы сравнения:

- 1) == (значения равны);
- 2) != (значения не равны);
- 3) > (первое значение больше второго);
- 4) < (второе значение больше первого);
- 5) >= (первое значение больше второго или равно ему);
- 6) <= (второе значение больше первого или равно ему);
8. Что называется простым условием? Приведите примеры.

Простое условие – условие, которое содержит одну проверку. Пример: «a > b».

9. Что такое составное условие? Приведите примеры.

Составное условие – условие, которое содержит больше одной проверки. Пример: «a > b and b > c».

10. Какие логические операторы допускаются при составлении сложных условий?

При составлении сложных условий в языке программирования Python используются следующие операторы:

- 1) and (логическое «и»);
- 2) or (логическое «или»);
- 3) not (логическое «нет»).

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

Оператор ветвления может содержать внутри себя другие ветвления.

12. Какой алгоритм является алгоритмом циклической структуры?

Алгоритм циклической структуры – алгоритм, который содержит циклы.

13. Типы циклов в языке Python.

В языке программирования Python существуют следующие типы циклов:

1) Оператор `for`. Данный оператор выполняет указанный набор инструкций заданное количество раз, которое определяется количеством элементов в наборе;

2) Оператор цикла `while`. Данный оператор 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`.

Синтаксис функции:

`range(stop)`; `range(start, stop, step)`.

Параметры функции `range`:

1) Параметр `start` указывает, с какого числа начинается последовательность. По умолчанию – 0;

2) Параметр `stop` указывает, до какого числа продолжается последовательность чисел. Указанное число не включается в диапазон;

3) Параметр `step` определяет шаг. По умолчанию – 1.

Данная функция генерирует возрастающие или убывающие последовательности чисел с определённым шагом, которые могут использоваться для создания циклов, а также массивов и списков чисел.

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

Чтобы организовать перебор значений от 15 до 0 с шагом 2, то необходимо создать цикл for и передать в функцию range() следующие параметры:

1) start = 15;

2) stop = -1;

3) step = -2.

for i in range(15, -1, -2):

 print(i)

16. Могут ли быть циклы вложенными?

Циклы могут быть вложенными.

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

Бесконечный цикл образуется тогда, когда возникает ситуация, в которой условие цикла всегда остаётся истинным. Чтобы выйти из бесконечного цикла, можно использовать оператор break, который будет срабатывать при определённых условиях, не давая циклу выполняться бесконечно.

18. Для чего нужен оператор break ?

Оператор break нужен для досрочного завершения выполнения цикла.

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

Оператор continue используется в циклах, он необходим для пропуска определённой итерации.

20. Для чего нужны стандартные потоки stdout и stderr?

Поток stdout нужен для вывода данных и информационных сообщений, а поток stderr используется 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. Само же определение потока stderr находится в стандартном пакете Python sys.

Пример кода:

```
import sys
```

```
print("Ошибка!", file=sys.stderr)
```

22. Каково назначение функции exit?

Функция `exit` используется для того, чтобы завершить программу и передать операционной системе заданный код возврата.

Вывод

В ходе выполнения 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.