

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций
«Исследование возможностей Git для работы с
локальными репозиториями»**

**Отчет по лабораторной работе № 2.2
по дисциплине «Основы программной инженерии»**

Выполнил студент группы
ПИЖ-б-о-21-1
Трушева В. О. .« » сентября
2022г.

Подпись студента _____
Работа защищена «
» _____ 20__ г.

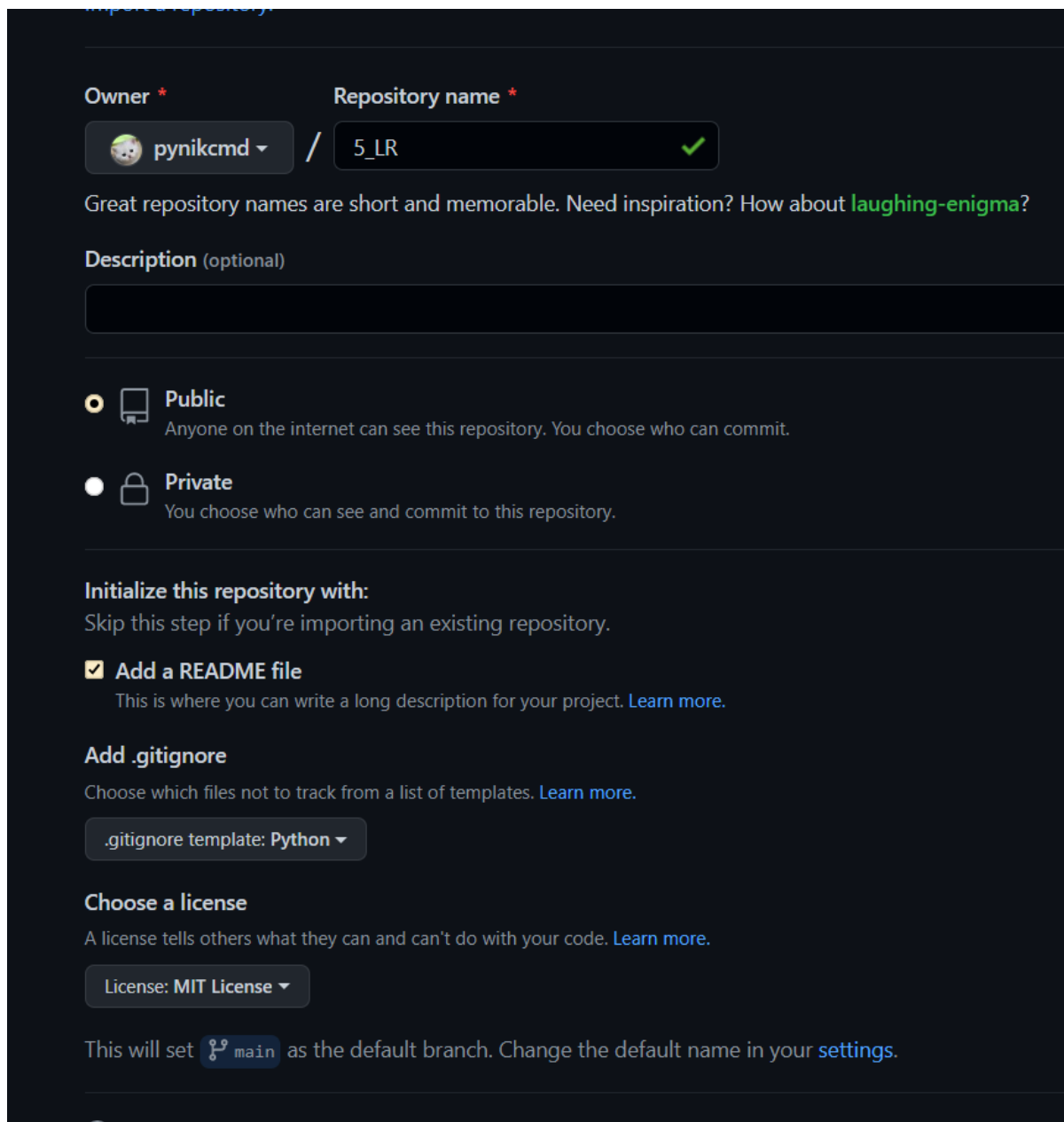
Проверила Воронкин Р.А.

(подпись)

Ставрополь 2022

Методика и порядок выполнения работы

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



Owner * / Repository name *

Great repository names are short and memorable. Need inspiration? How about [laughing-enigma?](#)

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: **Python**

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

License: **MIT License**

This will set **main** as the default branch. Change the default name in your [settings](#).

Рисунок 1 – Создание репозитория

3. Выполните клонирование созданного репозитория.

```
D:\fgit>git clone https://github.com/pynikcmd/5_LR.git
Cloning into '5_LR'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 11 (delta 2), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (11/11), 4.52 KiB | 2.26 MiB/s, done.
Resolving deltas: 100% (2/2), done.
```

Рисунок 2 – Клонирование репозитория

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



Рисунок 3 – Дополнен файл .gitignore

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

```
d:\fgit\5_LR>git flow init
Already initialized for gitflow.
To force reinitialization, use: git flow init -f
```

Рисунок 4 – 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 заданий в соответствии с PEP-8.

7. Создайте проект PyCharm в папке репозитория.

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

9. Приведите в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных вводимых с клавиатуры.

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

Run: 1_Primer x

D:\fgit\5_LR\PyCharm\Scripts\python.exe D:\fgit\5_LR\Tasks\1_Primer.py

Value of x? 54

y = -2916.5587890488514

Рисунок 5 – Первый пример

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

Run: 2_Primer x

D:\fgit\5_LR\PyCharm\Scripts\python.exe D:\fgit\5_LR\Tasks\2_Primer.py

Введите номер месяца: 12

Зима

Process finished with exit code 0

Рисунок 6 – Второй пример

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

Run: 3_Primer x

D:\fgit\5_LR\PyCharm\Scripts\python.exe D:\fgit\5_LR\Tasks\3_

Value of n? 5

Value of x? 4

S = 2.4753715714873286

Рисунок 7 – Третий пример

```
1 import math
2 import sys
3
4 if __name__ == '__main__':
5     a = float(input("Value of a? "))
6     if a < 0:
7         print("Illegal value of a", file=sys.stderr)
8         exit(1)
9
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
17    print(f"x = {x}\nX = {math.sqrt(a)}")
```

Run: 4_Primer x

D:\fgit\5_LR\PyCharm\Scripts\python.exe D:\fgit\5_LR\Tasks\4_

Value of a? 54

x = 7.3484692283495345

X = 7.3484692283495345

Рисунок 8 – Четвертый пример:

```
1 import math
2 import sys
3
4 # Постоянная Эйлера.
5 EULER = 0.5772156649015328606
6 # Точность вычислений.
7 EPS = 1e-10
8
9 if __name__ == '__main__':
10     x = float(input("Value of x? "))
11     if x == 0:
12         print("Illegal value of x", file=sys.stderr)
13         exit(1)
14
15     a = x
16     S, k = a, 1
17
18     # Найти сумму членов ряда.
19     while math.fabs(a) > EPS:
20         a *= x * k / (k + 1) ** 2
21         S += a
22         k += 1
23
24     # Вывести значение функции.
25     print(f"Ei({x}) = {EULER + math.log(math.fabs(x)) + S}")
26
27 if __name__ == '__main__' > while math.fabs(a) > EPS
```

Run: 5_Primer x

D:\fgit\5_LR\PyCharm\Scripts\python.exe D:\fgit\5_LR\Tasks\5_Primer.py

Value of x? 78

Ei(78.0) = 9.73989168877175e+31

Рисунок 9 – Пятый пример

10. Для примеров 4 и 5 постройте UML-диаграмму деятельности. Для построения диаграмм деятельности использовать веб-сервис Google <https://www.diagrams.net/>.

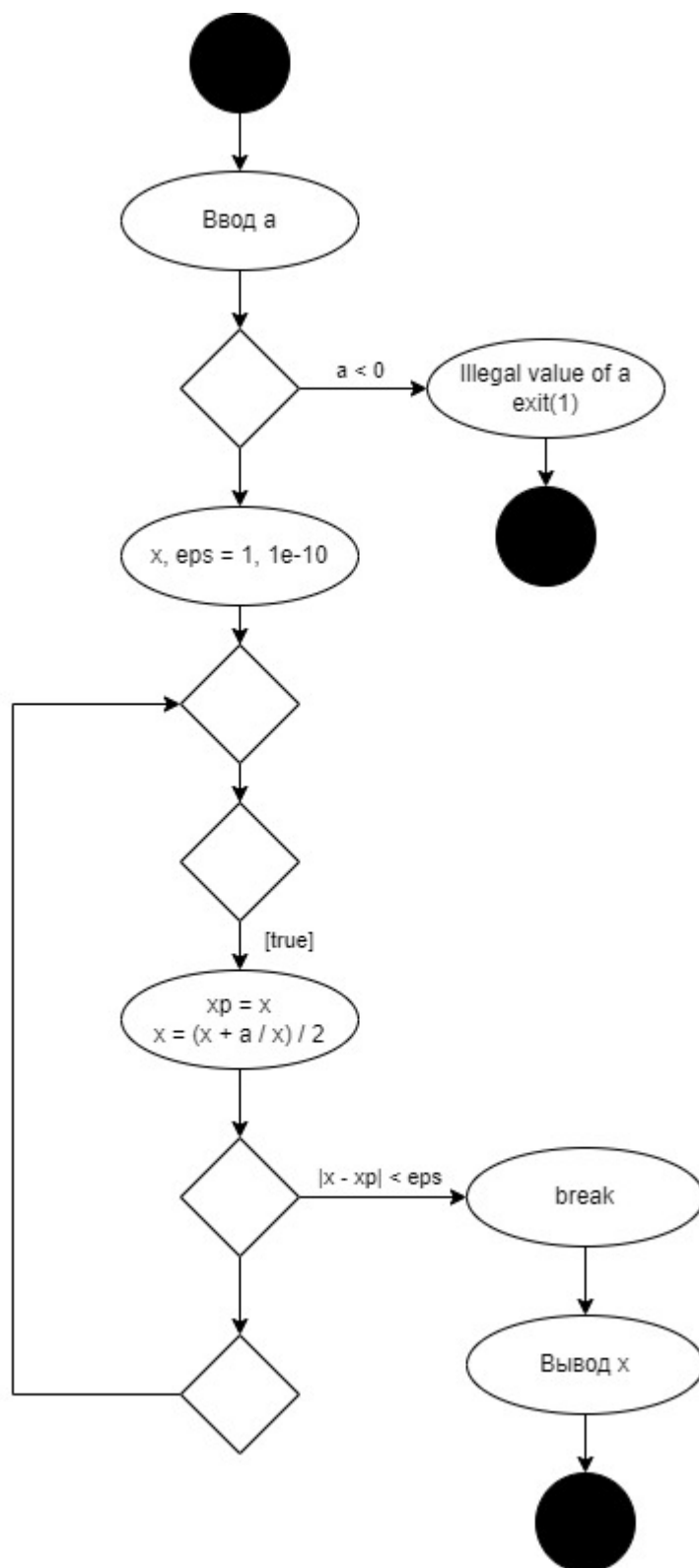


Рисунок 10 – UML-диаграмма для 4 примера

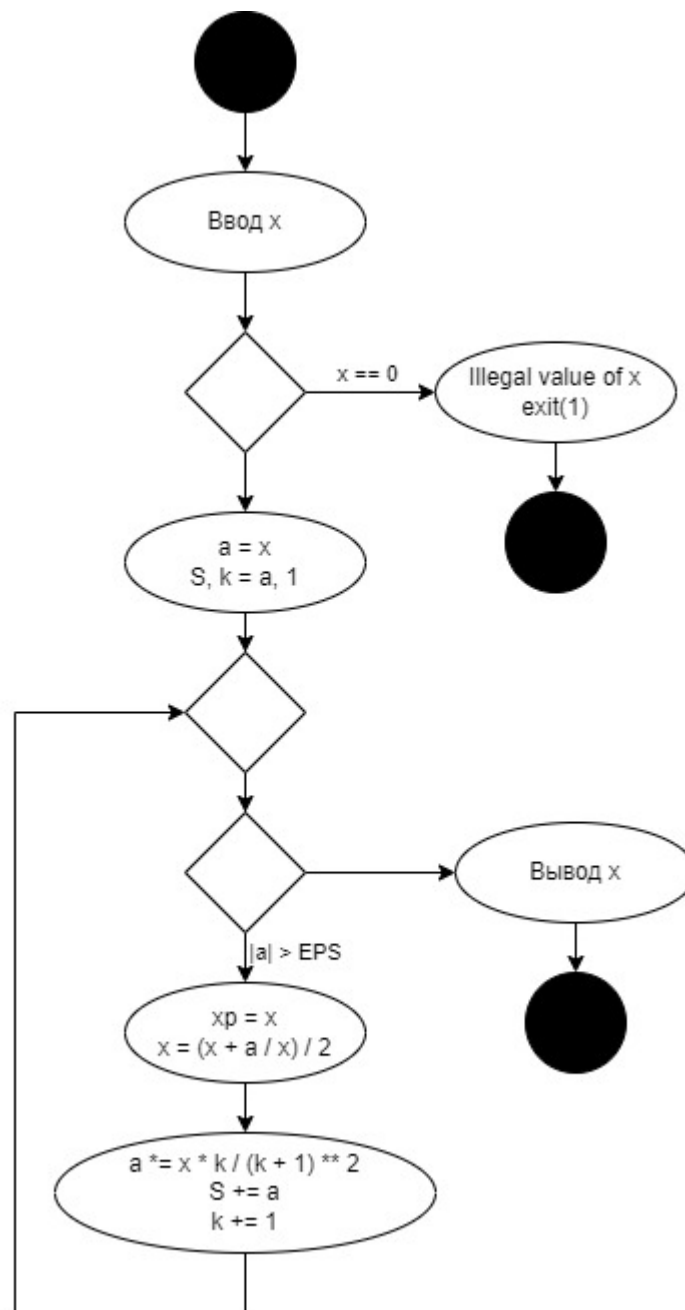
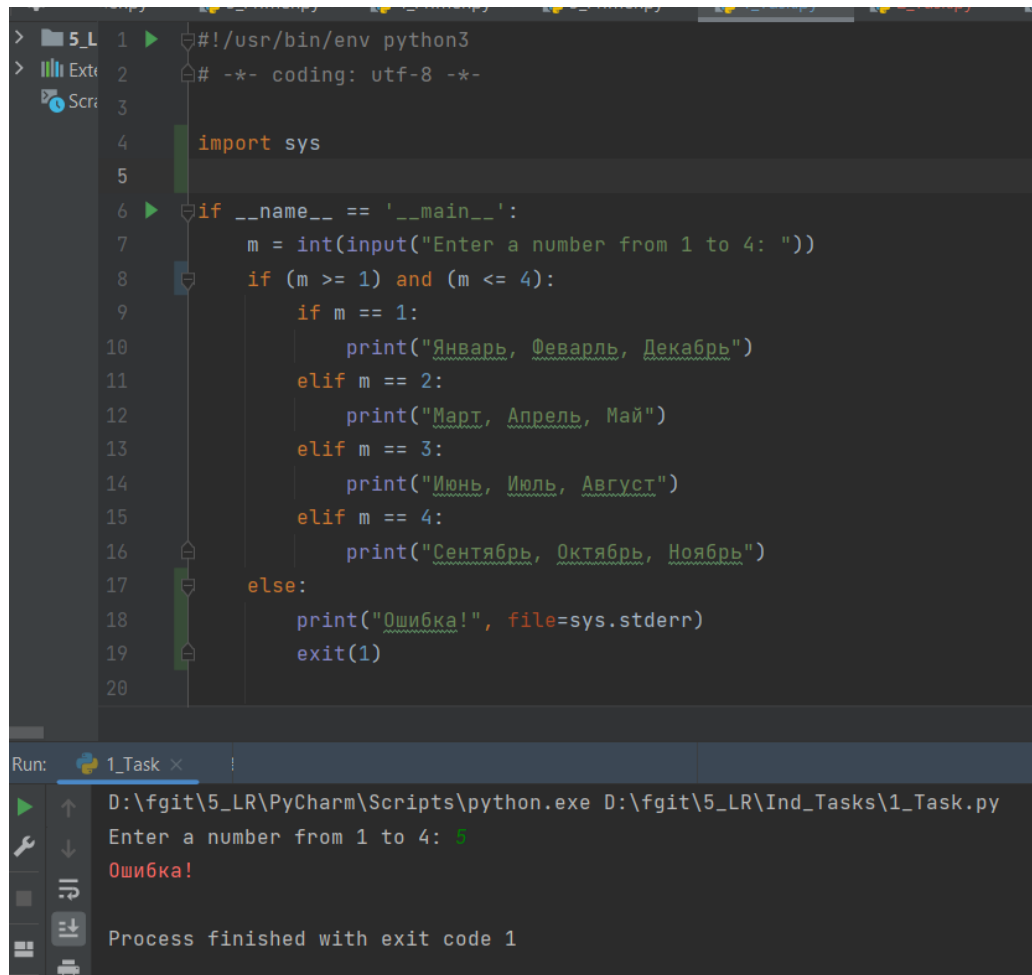


Рисунок 11 – UML-диаграмма для 5 примера

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

Индивидуальное задание 1. Вариант – 5.

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



```
1  > 1  > #!/usr/bin/env python3
2  > 2  > # -*- coding: utf-8 -*-
3
4  > 3  > import sys
5
6  > 4  > if __name__ == '__main__':
7  > 5  >     m = int(input("Enter a number from 1 to 4: "))
8  > 6  >     if (m >= 1) and (m <= 4):
9  > 7  >         if m == 1:
10 > 8  >             print("Январь, Феварль, Декабрь")
11 > 9  >         elif m == 2:
12 > 10 >             print("Март, Апрель, Май")
13 > 11 >         elif m == 3:
14 > 12 >             print("Июнь, Июль, Август")
15 > 13 >         elif m == 4:
16 > 14 >             print("Сентябрь, Октябрь, Ноябрь")
17 > 15 >     else:
18 > 16 >         print("Ошибка!", file=sys.stderr)
19 > 17 >         exit(1)
20 > 18 >
```

Run: 1_Task x

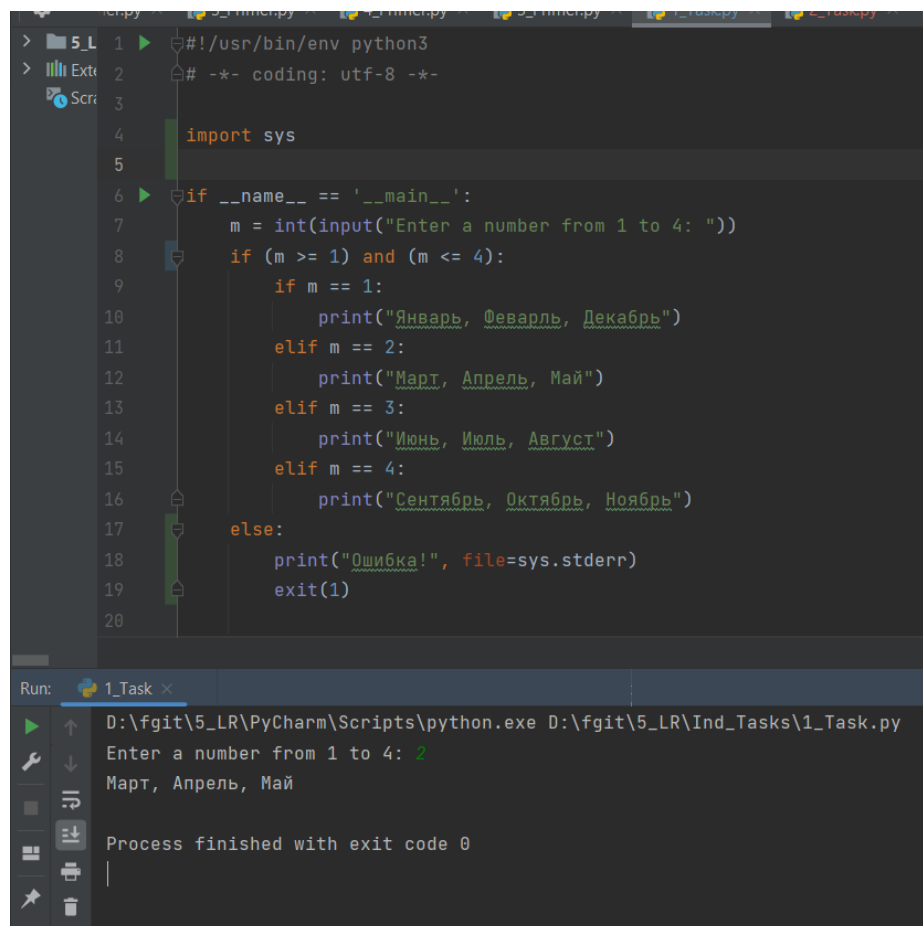
D:\fgit\5_LR\PyCharm\Scripts\python.exe D:\fgit\5_LR\Ind_Tasks\1_Task.py

Enter a number from 1 to 4: 5

Ошибка!

Process finished with exit code 1

Рисунок 12 – Индивидуальное задание 1



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6 if __name__ == '__main__':
7     m = int(input("Enter a number from 1 to 4: "))
8     if (m >= 1) and (m <= 4):
9         if m == 1:
10             print("Январь, Февраль, Декабрь")
11         elif m == 2:
12             print("Март, Апрель, Май")
13         elif m == 3:
14             print("Июнь, Июль, Август")
15         elif m == 4:
16             print("Сентябрь, Октябрь, Ноябрь")
17     else:
18         print("Ошибка!", file=sys.stderr)
19         exit(1)
20
```

Run: 1_Task x

D:\fgit\5_LR\PyCharm\Scripts\python.exe D:\fgit\5_LR\Ind_Tasks\1_Task.py

Enter a number from 1 to 4: 2

Март, Апрель, Май

Process finished with exit code 0

Рисунок 13 – Индивидуальное задание 1

Индивидуальное задание 2. Вариант – 7.

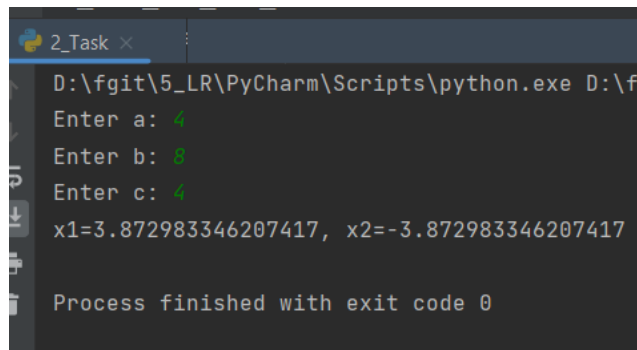
Условия. Провести исследование биквадратного уравнения $ax^2 + bx^2 + c = 0$ ($a \neq 0$), где a , b , c – действительные числа. Если действительных корней нет, то об этом должно быть выдано сообщение, иначе должны быть выданы 2 или 4 действительных корня.

```

7  ▶ if __name__ == '__main__':
8      a = float(input("Enter a: "))
9      b = float(input("Enter b: "))
10     c = float(input("Enter c: "))
11     if a == 0:
12         print("Ошибка!", file=sys.stderr)
13         exit(1)
14     else:
15         D = b * b + 4 * a * c
16         if D > 0:
17             t1 = (-b + D) / (2 * a)
18             t2 = (-b + (-D)) / (2 * a)
19             if t1 >= 0:
20                 x1 = math.sqrt(t1)
21                 x2 = -math.sqrt(t1)
22                 print(f"x1={x1}, x2={x2}")
23             if t2 >= 0:
24                 x3 = math.sqrt(t2)
25                 x4 = -math.sqrt(t2)
26                 print(f"x3={x3}, x4={x4}")
27             else:
28                 print("Действительных корней нет")
29         elif D == 0:
30             t1 = -b / (2 * a)
31             if t1 >= 0:
32                 x1 = math.sqrt(t1)
33                 x2 = -math.sqrt(t1)
34                 print(f"x1={x1}, x2={x2}")
35             else:
36                 print("Действительных корней нет")
37         elif D < 0:
38             print("Действительных корней нет")
39

```

Рисунок 14 – Индивидуальное задание 2



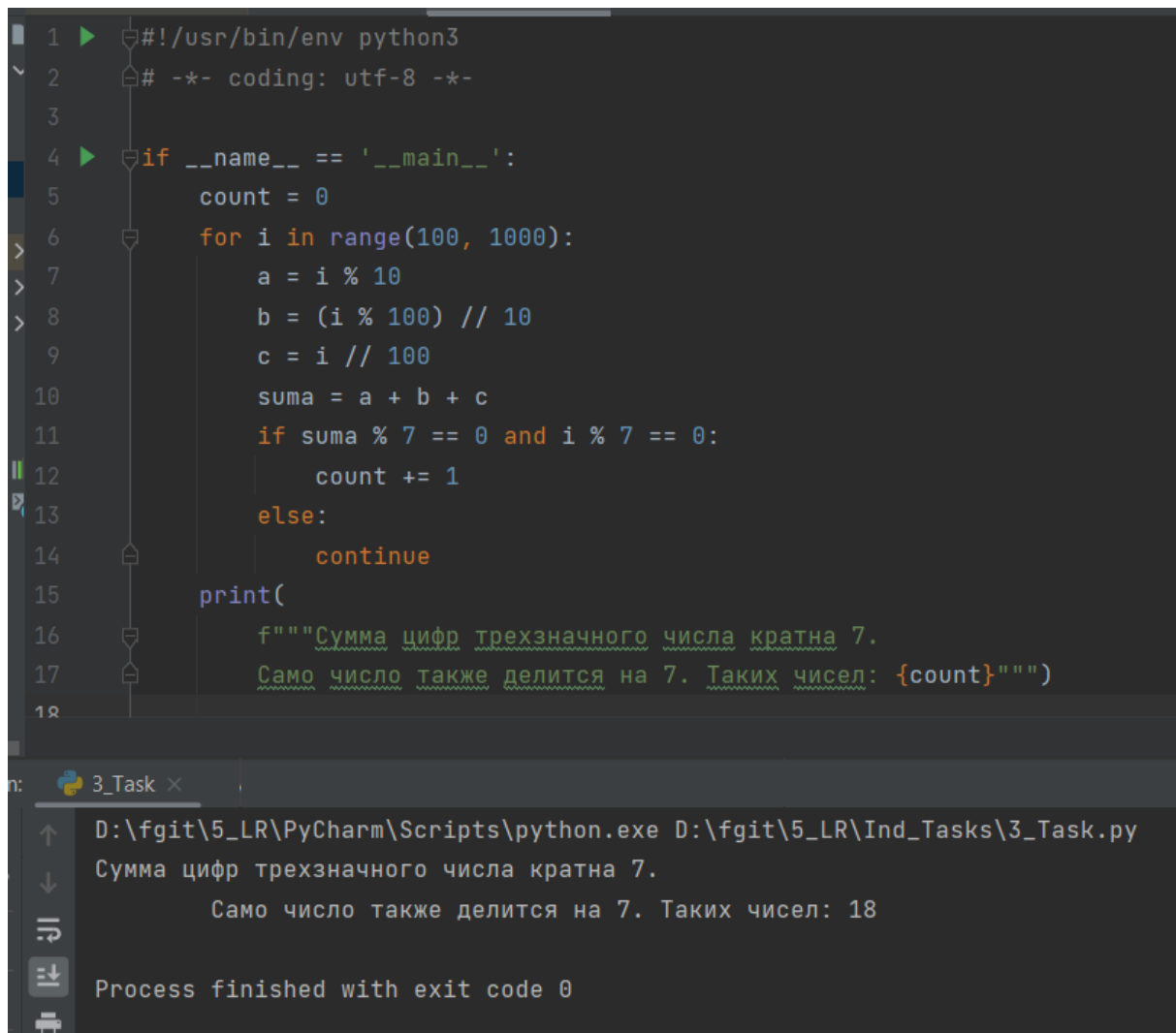
```
2_Task x
D:\fgit\5_LR\PyCharm\Scripts\python.exe D:\f
Enter a: 4
Enter b: 8
Enter c: 4
x1=3.872983346207417, x2=-3.872983346207417

Process finished with exit code 0
```

Рисунок 15 – Индивидуальное задание 2

Индивидуальное задание 3. Вариант – 8.

Условия. Сумма цифр трехзначного числа кратна 7. Само число также делится на 7. Найти все такие числа.



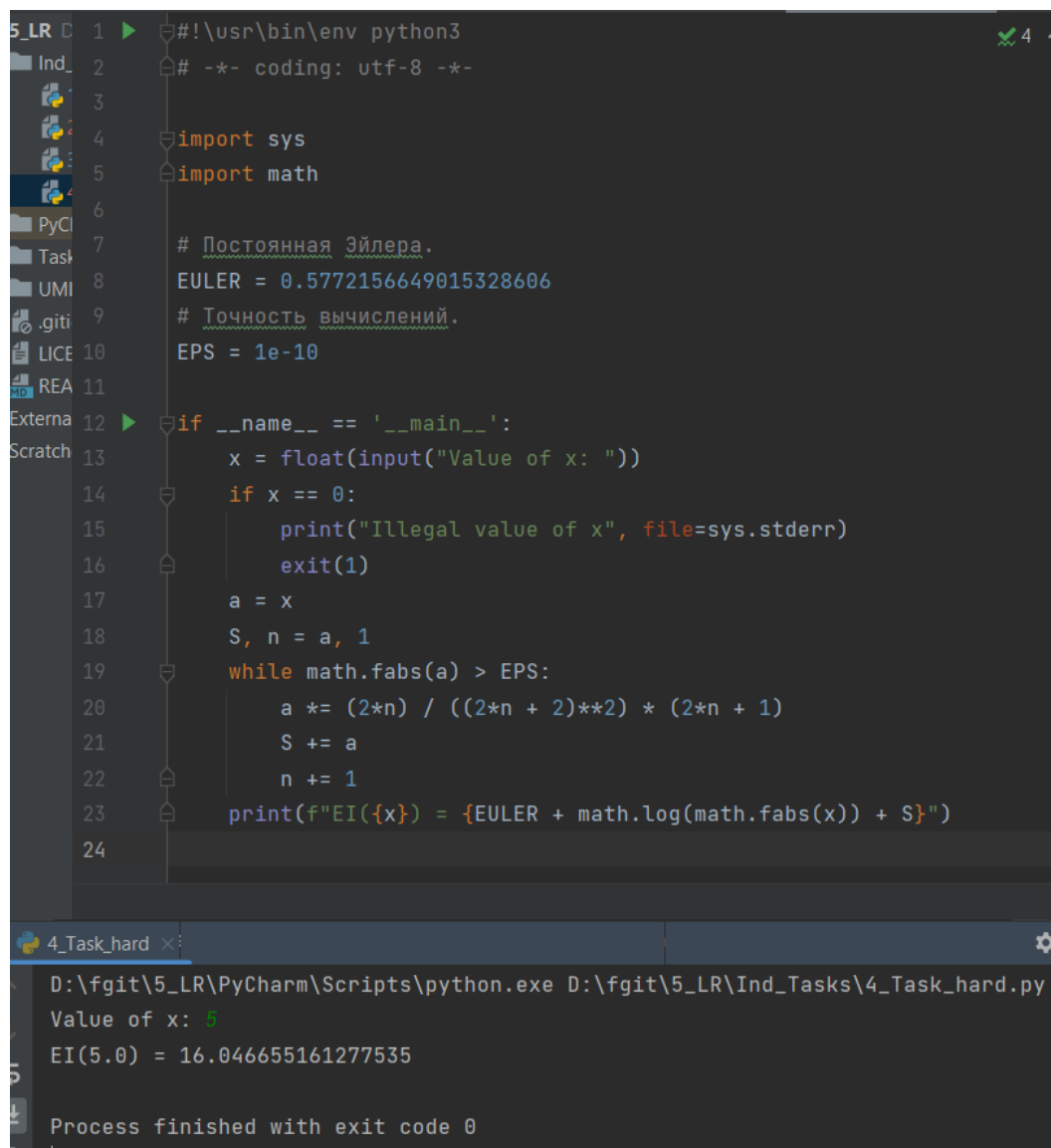
```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     count = 0
6     for i in range(100, 1000):
7         a = i % 10
8         b = (i % 100) // 10
9         c = i // 100
10        suma = a + b + c
11        if suma % 7 == 0 and i % 7 == 0:
12            count += 1
13        else:
14            continue
15    print(
16        f"Сумма цифр трехзначного числа кратна 7.
17        Само число также делится на 7. Таких чисел: {count}")
18
```

```
3_Task x
D:\fgit\5_LR\PyCharm\Scripts\python.exe D:\fgit\5_LR\Ind_Tasks\3_Task.py
Сумма цифр трехзначного числа кратна 7.
    Само число также делится на 7. Таких чисел: 18

Process finished with exit code 0
```

Рисунок 16 – Индивидуальное задание 3

Индивидуальное задание 4 (повышенная сложность). Вариант – 4.



```
1  #!\usr\bin\env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5  import math
6
7  # Постоянная Эйлера.
8  EULER = 0.5772156649015328606
9  # Точность вычислений.
10 EPS = 1e-10
11
12 if __name__ == '__main__':
13     x = float(input("Value of x: "))
14     if x == 0:
15         print("Illegal value of x", file=sys.stderr)
16         exit(1)
17     a = x
18     S, n = a, 1
19     while math.fabs(a) > EPS:
20         a *= (2*n) / ((2*n + 2)**2) * (2*n + 1)
21         S += a
22         n += 1
23     print(f"EI({x}) = {EULER + math.log(math.fabs(x)) + S}")
24
```

4_Task_hard x

D:\fgit\5_LR\PyCharm\Scripts\python.exe D:\fgit\5_LR\Ind_Tasks\4_Task_hard.py

Value of x: 5

EI(5.0) = 16.046655161277535

Process finished with exit code 0

Рисунок 17 – Индивидуальное задание 4

12. Приведите в отчете скриншоты работы программ и UML-диаграммы деятельности решения индивидуальных заданий.

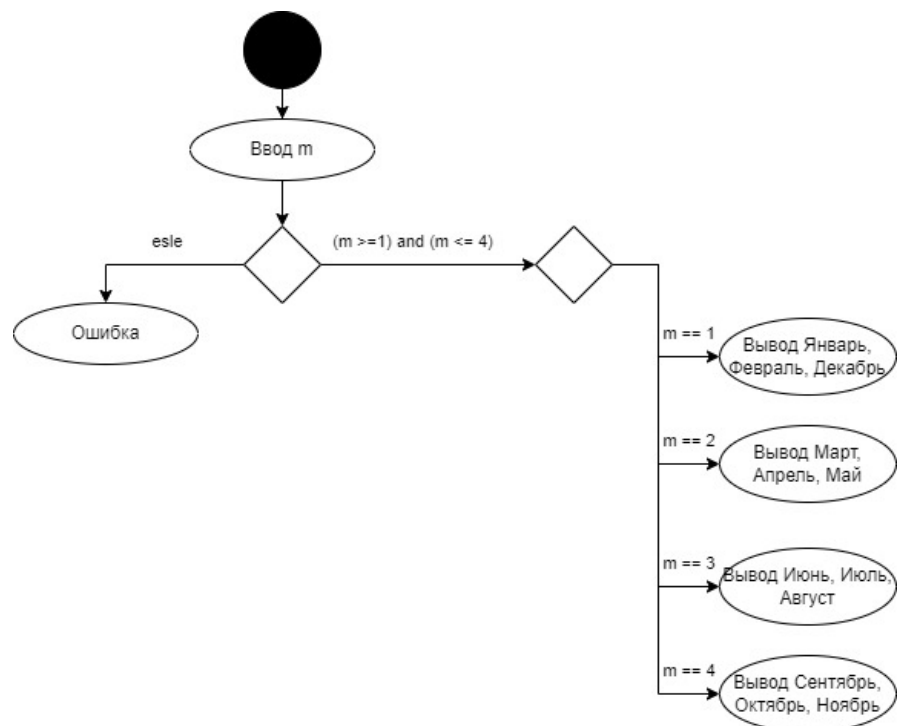


Рисунок 18 – UML-диаграмма для 1 индивидуального задания

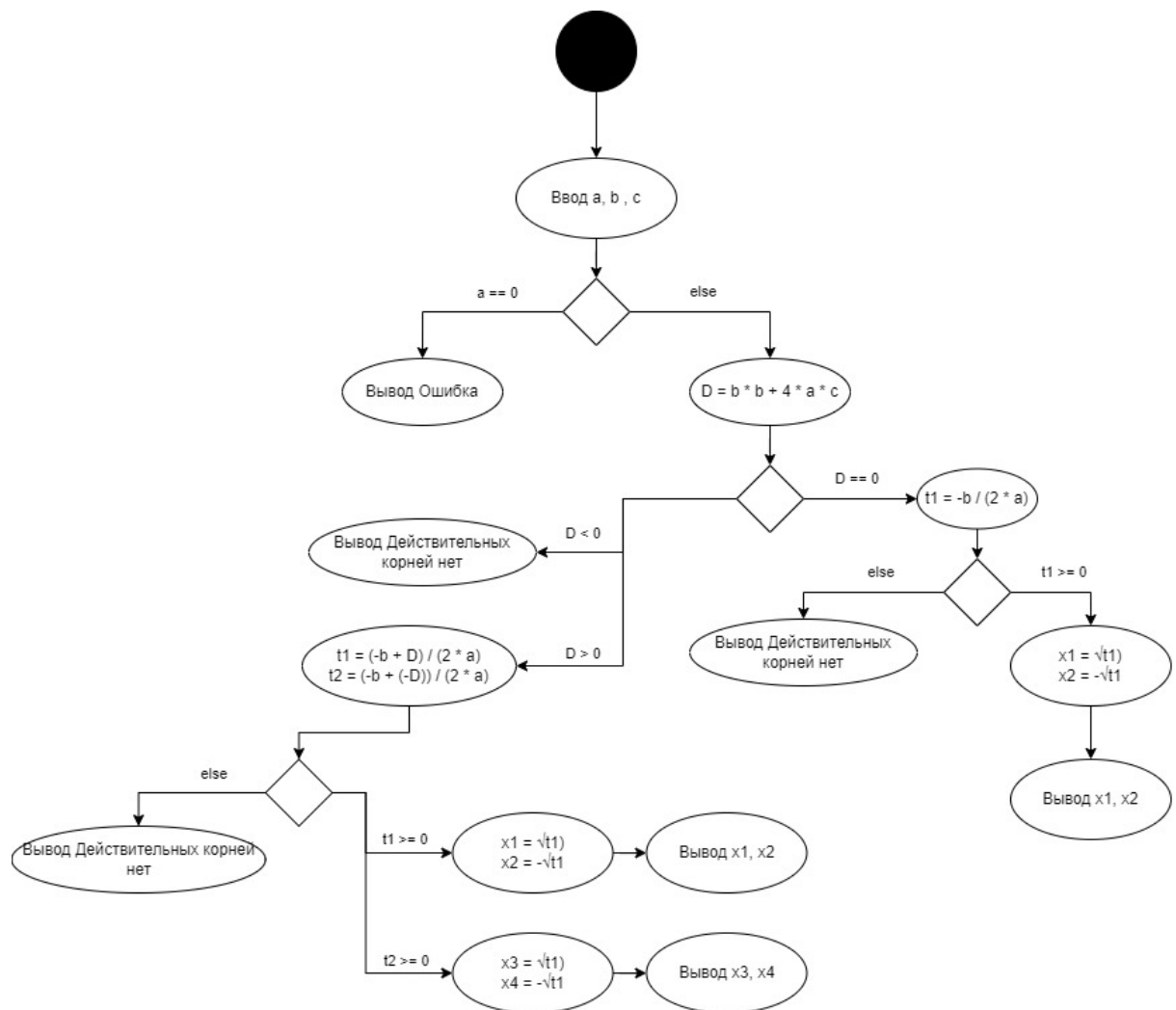


Рисунок 19 – UML-диаграмма для 2 индивидуального задания

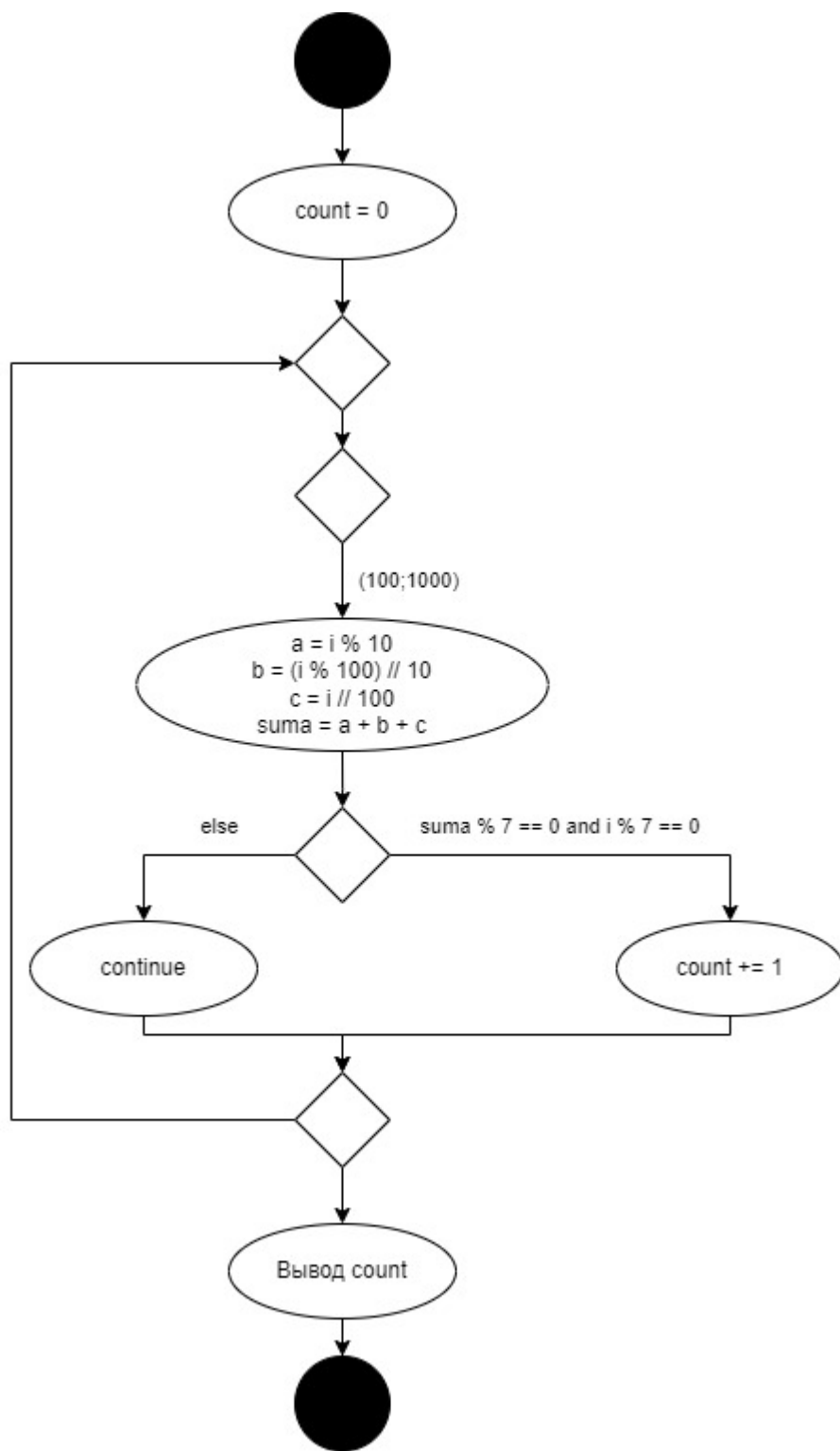


Рисунок 20 – UML-диаграмма для 3 индивидуального задания

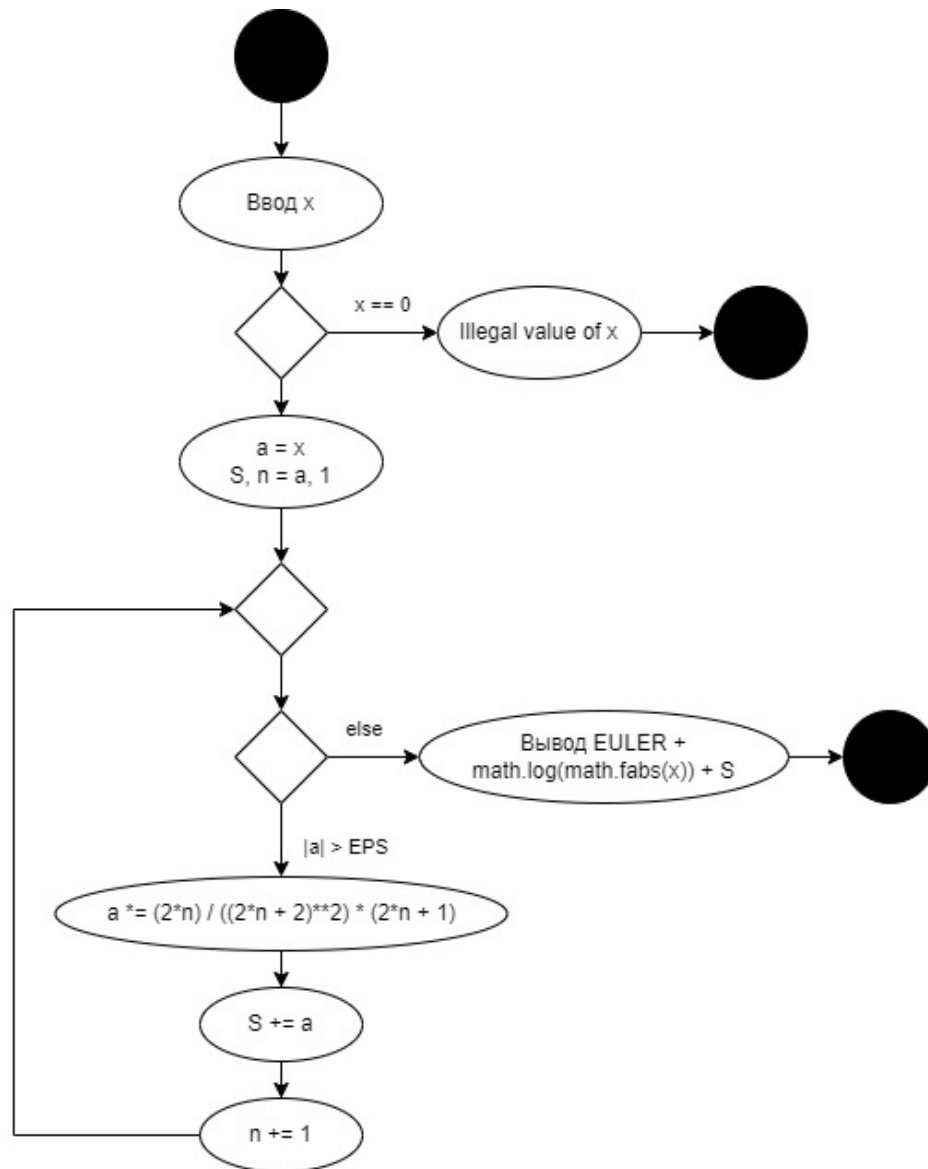


Рисунок 21 – UML-диаграмма для 4 индивидуального задания

13. Зафиксируйте сделанные изменения в репозитории.
14. Выполните слияние ветки для разработки с веткой main / master.
15. Отправьте сделанные изменения на сервер GitHub.
16. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

Вопросы для защиты работы

1. Для чего нужны диаграммы деятельности UML?

Диаграммы деятельности представляют собой графическое представление рабочих процессов поэтапных действий и действий с поддержкой выбора, итерации и параллелизма. Они описывают поток управления целевой системой, такой как исследование сложных бизнесправил и операций, а также описание прецедентов и бизнес-процессов. В UML диаграммы деятельности предназначены для моделирования как вычислительных, так и организационных процессов.

2. Что такое состояние действия и состояние деятельности?

Состояние действия — это состояние, внутренняя активность которого является действием.

Состояние деятельности — это состояние, внутренняя активность которого является деятельностью.

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

Переходы, ветвление, алгоритм разветвляющейся структуры, алгоритм циклической структуры.

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?

Линейный алгоритм - алгоритм, все этапы которого выполняются однократно и строго последовательно. Разветвляющийся алгоритм - алгоритм, содержащий хотя бы одно условие, в результате проверки которого ЭВМ обеспечивает переход на один из нескольких возможных шагов.

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

Оператор, конструкция языка программирования, обеспечивающая выполнение определённой команды (набора команд) только при условии истинности некоторого логического выражения, либо выполнение одной из нескольких команд. Условный оператор имеет полную и краткую формы.

7. Какие операторы сравнения используются в Python?

If, elif, else

8. Что называется простым условием? Приведите примеры.

Простым условием называется выражение, составленное из двух арифметических выражений или двух текстовых величин.

Пример: `x == y`

9. Что такое составное условие? Приведите примеры.

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

Пример: `(a == b or c == d)`

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?

Циклический алгоритм — алгоритм, предусматривающий многократное повторение одного и того же действия (одних и тех же операций) над новыми исходными данными.

13. Типы циклов в языке Python.

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

14. Назовите назначение и способы применения функции `range`.

Функция `range` генерирует серию целых чисел, от значения `start` до `stop`, указанного пользователем. Мы можем использовать его для цикла `for` и обходить весь диапазон как список.

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?

```
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.

18. Для чего нужен оператор `break`?

Оператор `break` предназначен для досрочного прерывания работы цикла.

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

Оператор `continue` используется только в циклах. В операторах `for`, `while` 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:
stdin – стандартный ввод (клавиатура), stdout – стандартный вывод (экран),
stderr – стандартная ошибка (вывод ошибок на экран)

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

Указать в `print(..., file=sys.stderr)`.

22. Каково назначение функции `exit`?

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