

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 «Основы кроссплатформенного программирования»

Вариант ____

Выполнила:
Маньшина Дарья Алексеевна
1 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., канд. тех. 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.

Ход работы:

1. Проработаем примеры данные в методичке.

Пример 1. Составить UML-диаграмму деятельности и программу с использованием конструкции ветвления и вычислить значение функции

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

Рисунок 1 – Условие примера №1

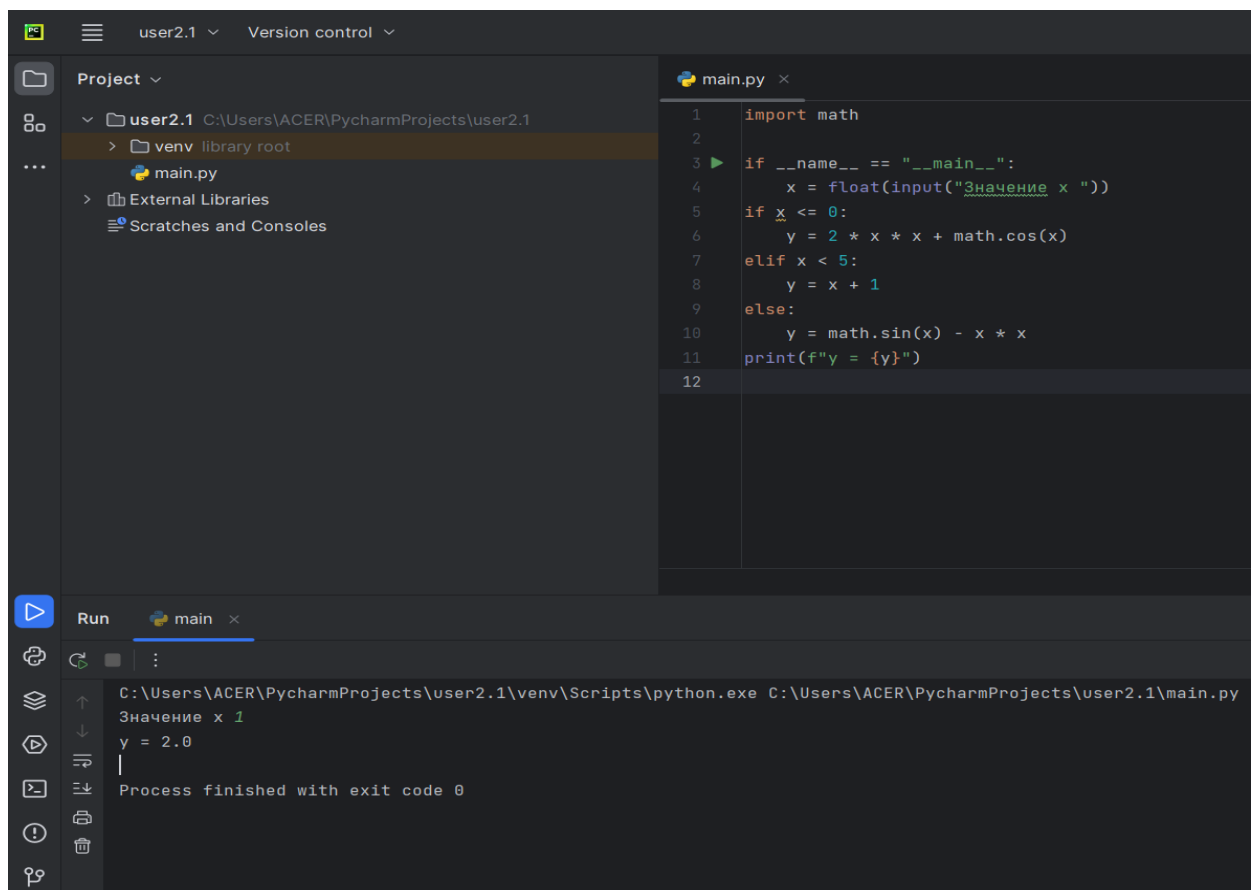
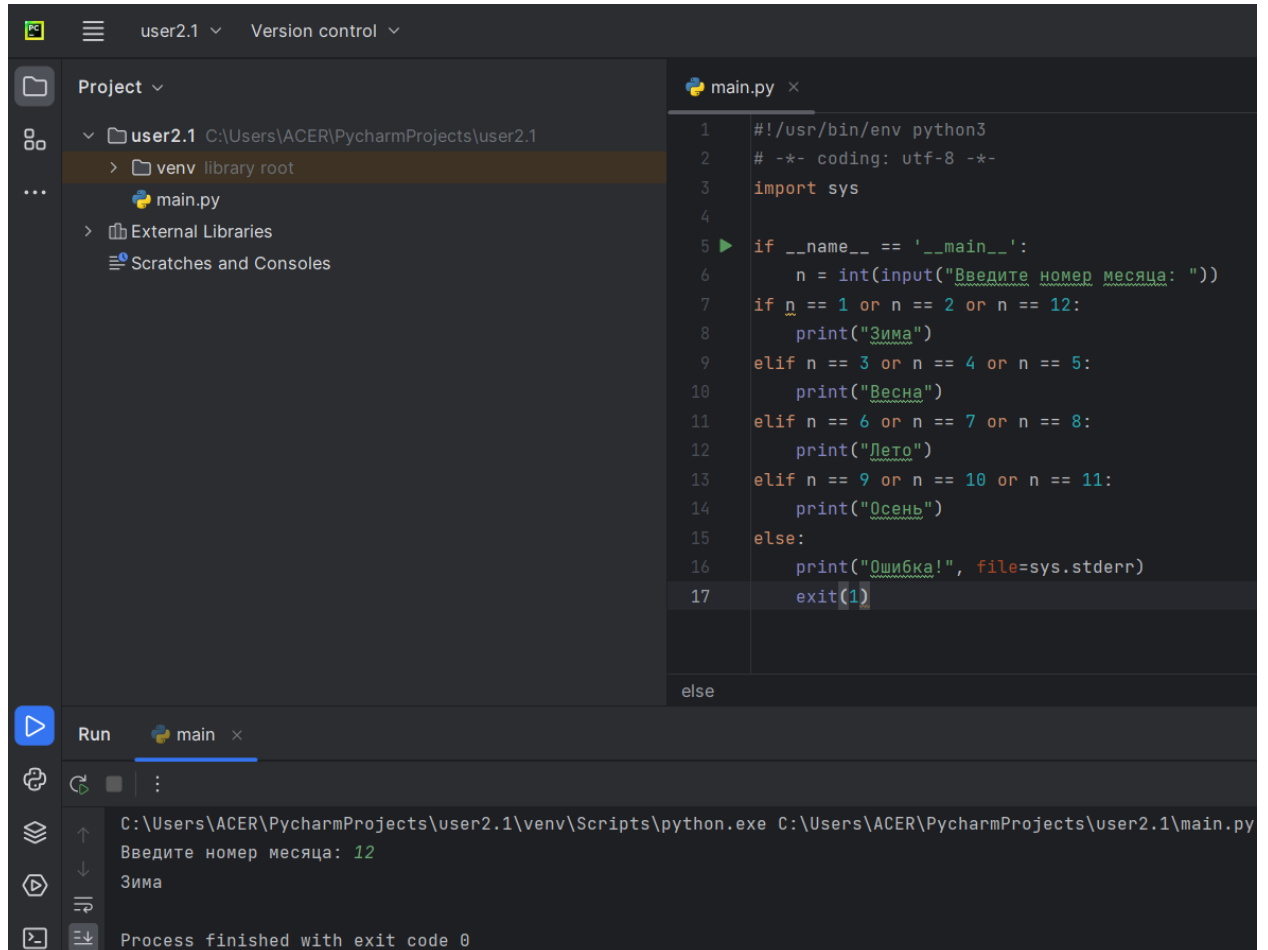


Рисунок 2 – Программа для примера №1

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.



The screenshot shows the PyCharm IDE interface. The left sidebar displays the project structure for 'user2.1', including a 'venv' directory and a 'main.py' file. The main editor window shows the code for 'main.py'. The code prompts the user to enter a month number, and based on the input, it prints the corresponding season: 'Зима' (Winter) for months 1-2, 'Весна' (Spring) for months 3-5, 'Лето' (Summer) for months 6-8, and 'Осень' (Autumn) for months 9-11. An error message is printed for invalid inputs. The bottom panel shows the program's execution, with the input '12' and the output 'Зима'.

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

Run main ×

C:\Users\ACER\PycharmProjects\user2.1\venv\Scripts\python.exe C:\Users\ACER\PycharmProjects\user2.1\main.py

Введите номер месяца: 12

Зима

Process finished with exit code 0

Рисунок 3 - Программа для примера №2

Пример 3. Составить UML-диаграмму деятельности и написать программу, позволяющую вычислить конечную сумму, где и вводятся с клавиатуры:

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

Рисунок 4 – Условие примера №3

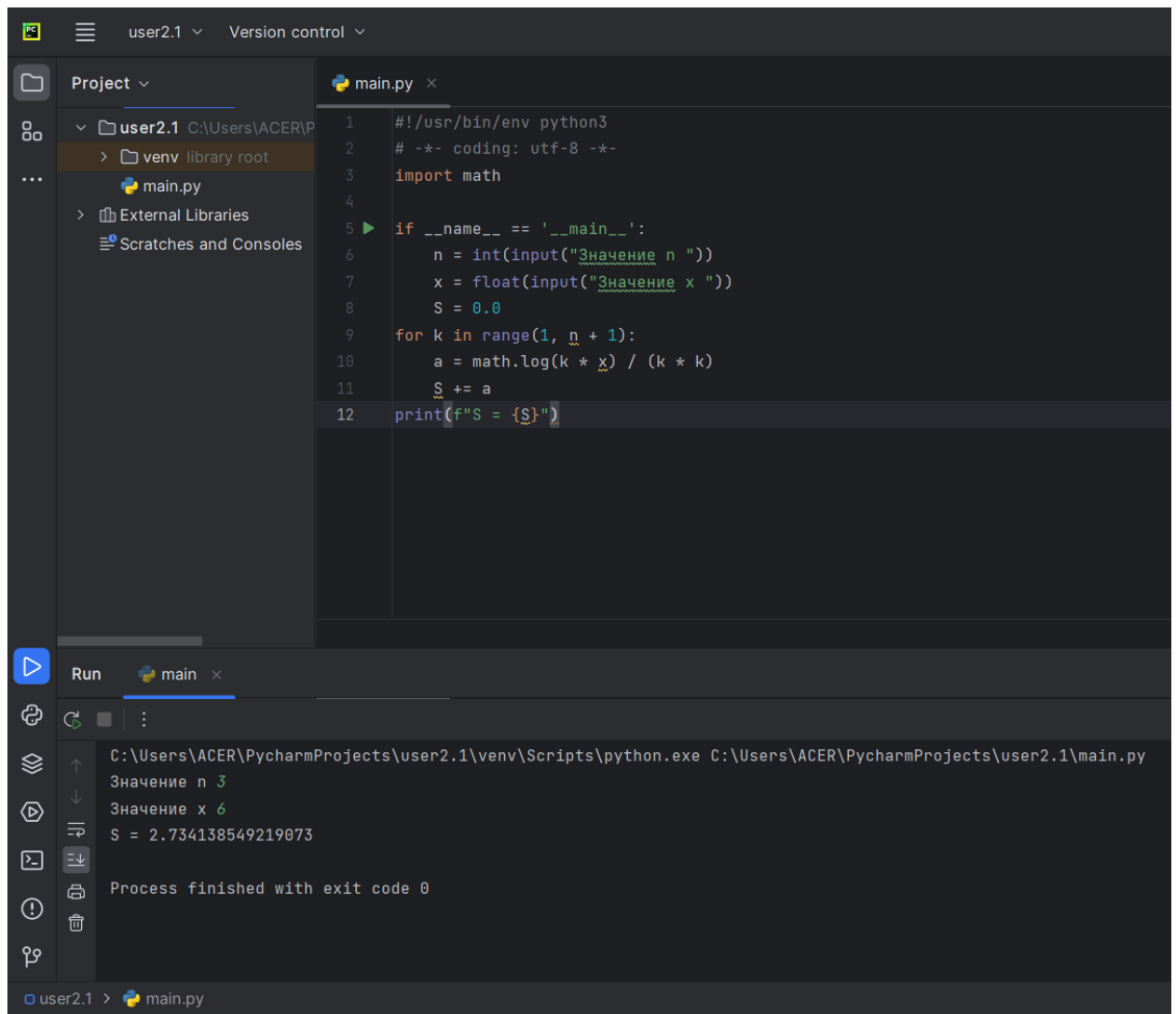


Рисунок 5 – Программа для решения примера №3

Пример 4.

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:

$$x_{n+1} = \frac{1}{2} \cdot \left(x_n + \frac{a}{x_n} \right). \quad (3)$$

В качестве начального значения примем $x_0 = 1$. Цикл должен выполняться до тех пор, пока не будет выполнено условие $|x_{n+1} - x_n| \leq \varepsilon$. Сравните со значением квадратного корня, полученным с использованием функций стандартной библиотеки. Значение $\varepsilon = 10^{-10}$.

Рисунок 6 – Условие примера №4

The image shows a screenshot of an IDE interface. The top bar displays 'user2.1' and 'Version control'. The left sidebar shows a project structure with 'user2.1' containing a 'venv' directory and a 'main.py' file. The main editor window shows the code in 'main.py'.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5  import sys
6
7  if __name__ == "__main__":
8      a = float(input("Значение a? "))
9      if a < 0:
10         print("Недопустимое значение a", file=sys.stderr)
11         exit(1)
12
13     x, eps = 1, 1e-10
14     while True:
15         xp = x
16         x = (x + a / x) / 2
17         if math.fabs(x - xp) < eps:
18             break
19
20     print(f"x = {x}\nX = {math.sqrt(a)}")
```

The bottom panel shows the 'Run' output for 'main'. It displays the input 'Значение a? -9', the error message 'Недопустимое значение a', and the final status 'Process finished with exit code 1'.

Рисунок 7 – Программа для примера №4. Решение с недопустимым значением

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5  import sys
6
7  if __name__ == "__main__":
8      a = float(input("Значение a? "))
9      if a < 0:
10         print("Недопустимое значение a", file=sys.stderr)
11         exit(1)
12
13         x, eps = 1, 1e-10
14         while True:
15             xp = x
16             x = (x + a / x) / 2
17             if math.fabs(x - xp) < eps:
18                 break
19
20         print(f"x = {x}\nX = {math.sqrt(a)}")
```

Run main

C:\Users\ACER\PycharmProjects\user2.1\venv\Scripts\python.exe C:\Users\ACER\PycharmProjects\user2.1\main.py

Значение a? 1

x = 1.0

X = 1.0

Рисунок 8 – Программа для примера №4. Решение с допустимым значением

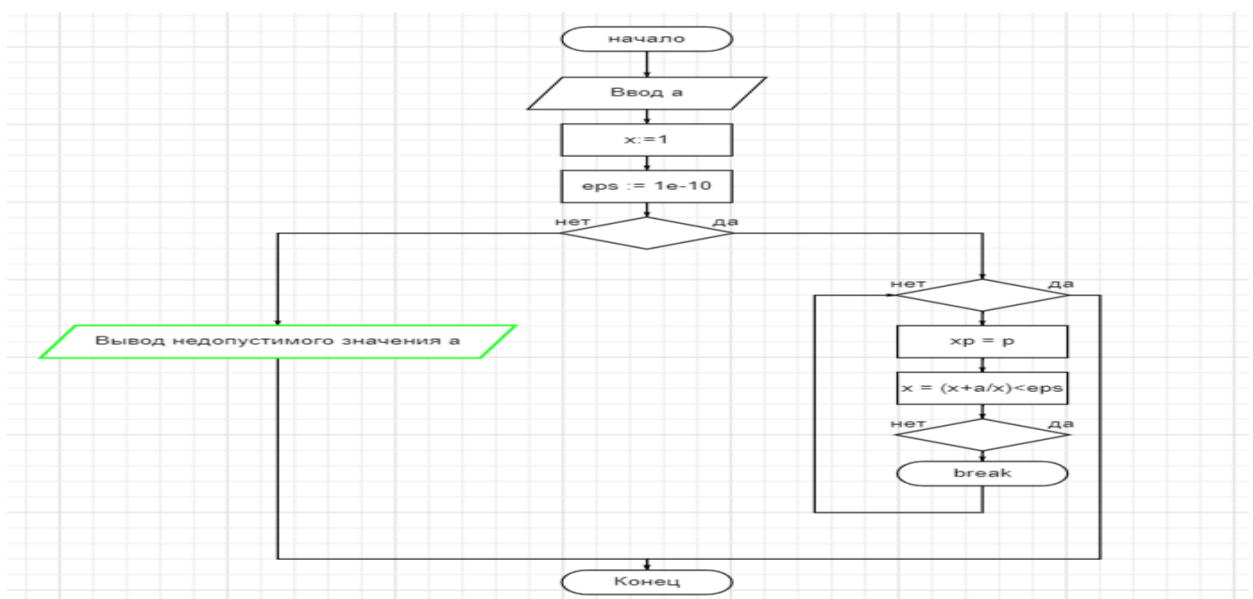


Рисунок 9 – UML-диаграмма деятельности примера №4

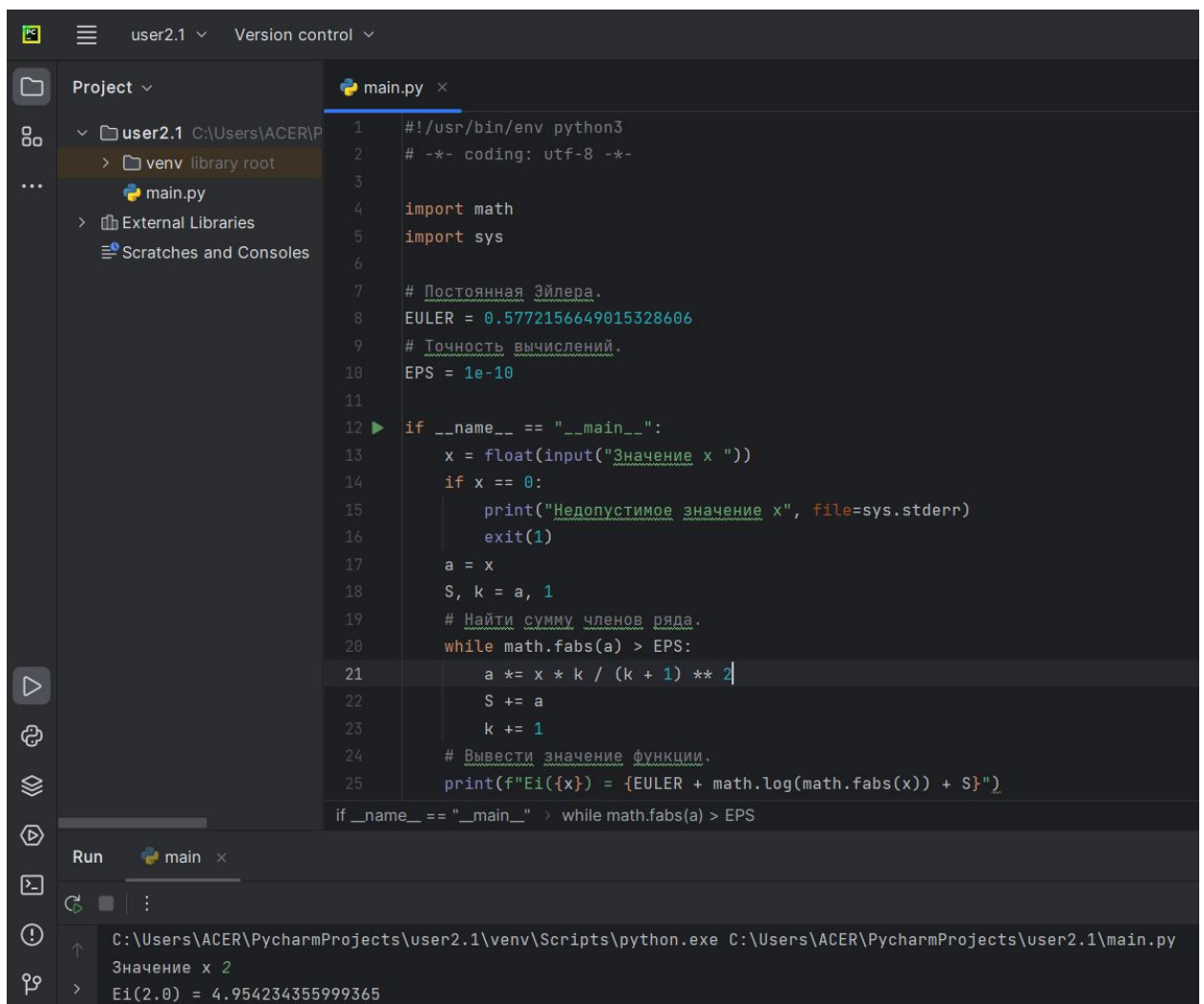
Пример 5.

Пример 5. Вычислить значение специальной (интегральной показательной) функции

$$\text{Ei}(x) = \int_{-\infty}^x \frac{\exp t}{t} dt = \gamma + \ln x + \sum_{k=1}^{\infty} \frac{x^k}{k \cdot k!}, \quad (4)$$

где $\gamma = 0.5772156649 \dots$ - постоянная Эйлера, по ее разложению в ряд с точностью $\varepsilon = 10^{-10}$, аргумент x вводится с клавиатуры.

Рисунок 10 – Условие для примера №5



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5  import sys
6
7  # Постоянная Эйлера.
8  EULER = 0.5772156649015328606
9  # Точность вычислений.
10 EPS = 1e-10
11
12 if __name__ == "__main__":
13     x = float(input("Значение x "))
14     if x == 0:
15         print("Недопустимое значение x", file=sys.stderr)
16         exit(1)
17     a = x
18     S, k = a, 1
19     # Найти сумму членов ряда.
20     while math.fabs(a) > EPS:
21         a *= x * k / (k + 1) ** 2
22         S += a
23         k += 1
24     # Вывести значение функции.
25     print(f"Ei({x}) = {EULER + math.log(math.fabs(x)) + S}")
if __name__ == "__main__" > while math.fabs(a) > EPS
```

Run main ×

C:\Users\ACER\PycharmProjects\user2.1\venv\Scripts\python.exe C:\Users\ACER\PycharmProjects\user2.1\main.py
Значение x 2
> Ei(2.0) = 4.95423435599365

Рисунок 11 – Решение и программа примера №5

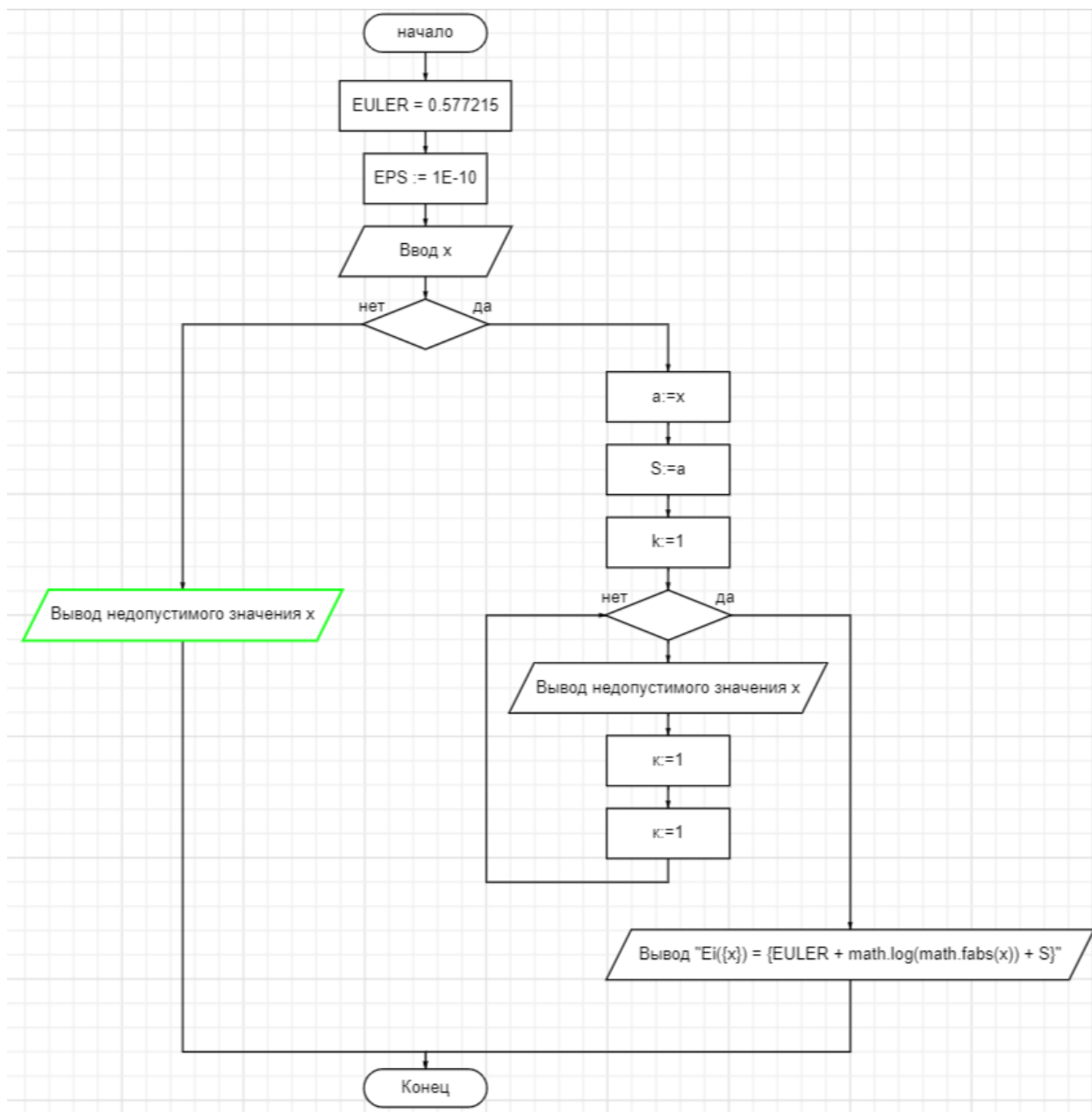


Рисунок 12 - UML-диаграмма деятельности примера №5

2. Решим индивидуальные задания. Вариант 9 (по списку подгруппы, так как заданий в первом индивидуальном задании всего 13) и вариант 15 (по списку группы).

Пример 1. Вводится число экзаменов. Напечатать фразу Мы успешно сдали N экзаменов, согласовав слово "экзамен" с числом.

project ▾

user2.1 C:\Users\ACER\PycharmProjects\user2.1

venv library root

main.py

External Libraries

Scratches and Consoles

main.py ×

```

1 N = int(input("Введите число меньше или равно 20: "))
2 if N == 0 or 5 <= N <= 20:
3     print("Мы сдали", N, "экзаменов")
4 elif 2 <= N <= 4:
5     print("Мы сдали", N, "экзамена")
6 elif N == 1:
7     print("Мы сдали", N, "экзамен")
8 else:
9     print("Ошибка")

```

main ×

```

C:\Users\ACER\PycharmProjects\user2.1\venv\Scripts\python.exe C:\Users\ACER\PycharmProjects\user2.1\main.py
Введите число меньше или равно 20: 5
Мы сдали 5 экзаменов

```

Рисунок 13 – Программа для первого индивидуального задания

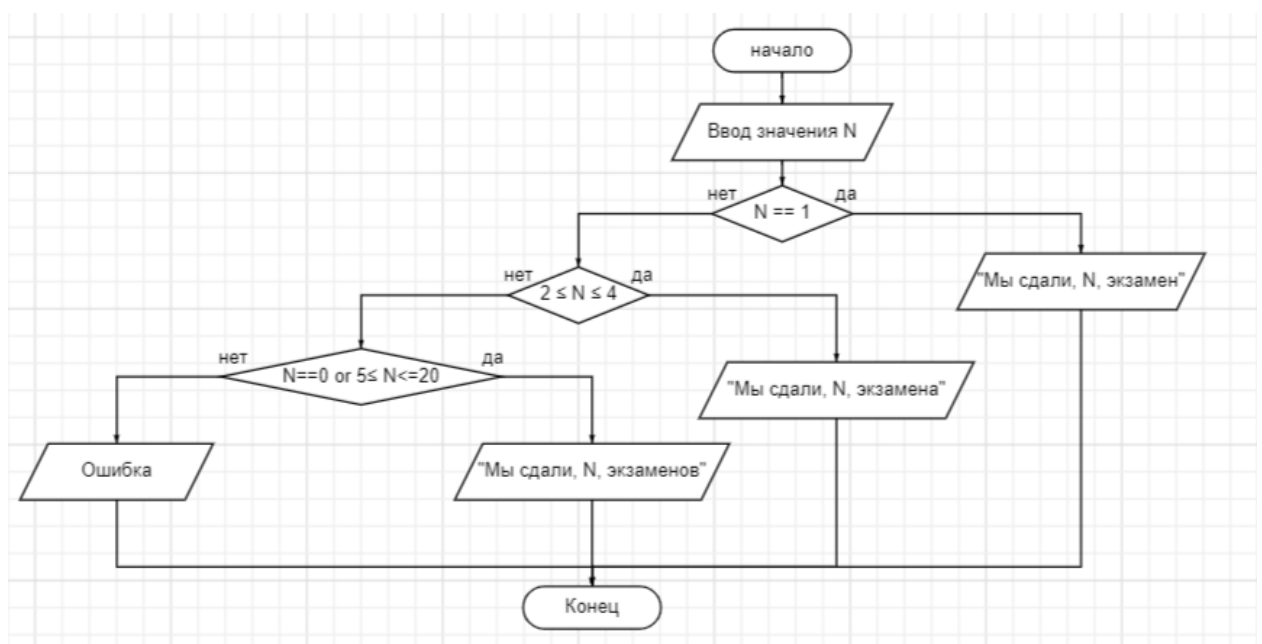
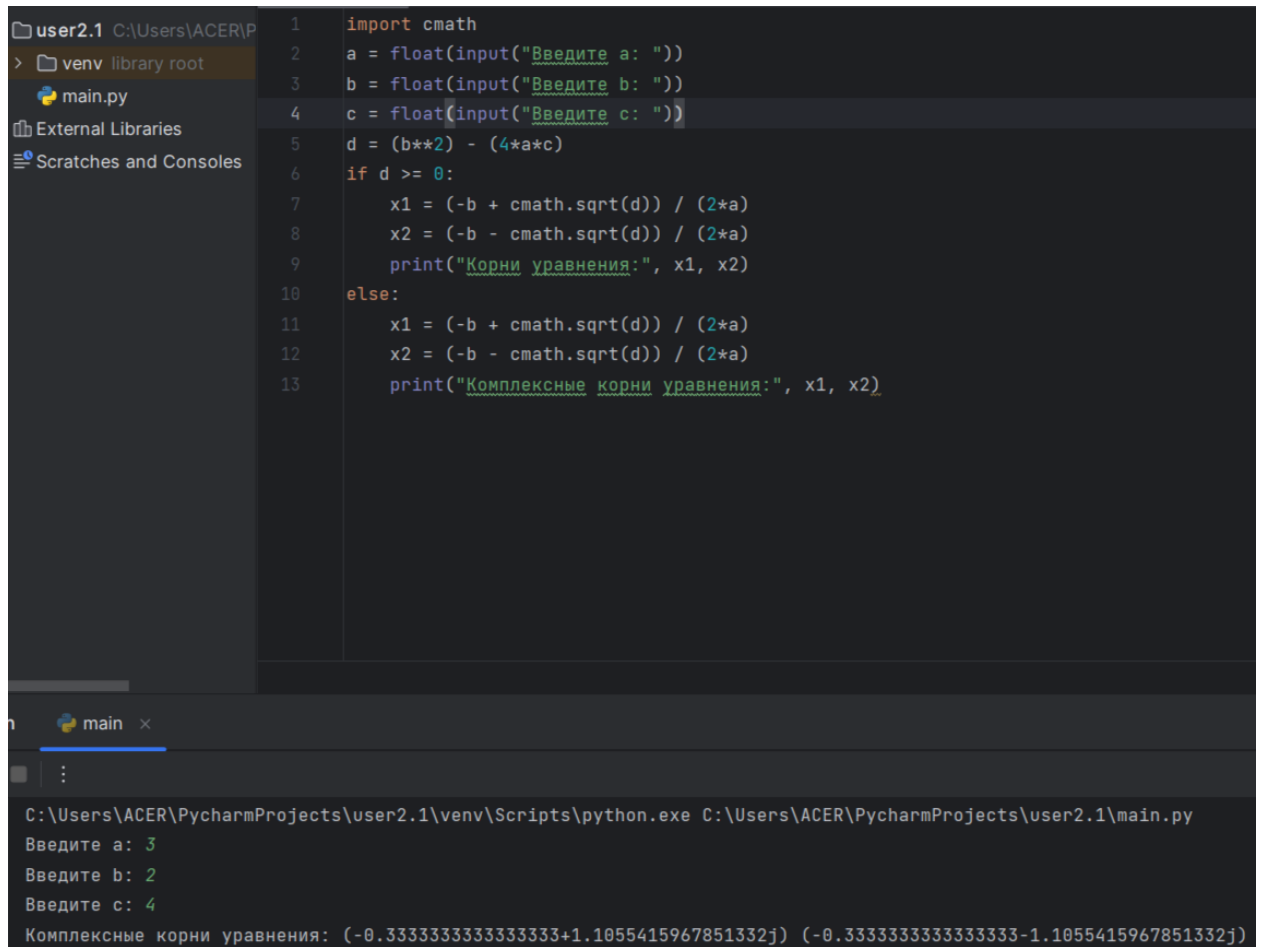


Рисунок 14 - UML-диаграмма деятельности первого индивидуального задания

Пример 2. Составить программу решения квадратного уравнения.
Выводить также комплексные решения.



```
1 import cmath
2 a = float(input("Введите a: "))
3 b = float(input("Введите b: "))
4 c = float(input("Введите c: "))
5 d = (b**2) - (4*a*c)
6 if d >= 0:
7     x1 = (-b + cmath.sqrt(d)) / (2*a)
8     x2 = (-b - cmath.sqrt(d)) / (2*a)
9     print("Корни уравнения:", x1, x2)
10 else:
11     x1 = (-b + cmath.sqrt(d)) / (2*a)
12     x2 = (-b - cmath.sqrt(d)) / (2*a)
13     print("Комплексные корни уравнения:", x1, x2)
```

main x

C:\Users\ACER\PycharmProjects\user2.1\venv\Scripts\python.exe C:\Users\ACER\PycharmProjects\user2.1\main.py
Введите a: 3
Введите b: 2
Введите c: 4
Комплексные корни уравнения: (-0.3333333333333333+1.1055415967851332j) (-0.3333333333333333-1.1055415967851332j)

Рисунок 15 – Программа и комплексное решение второго
индивидуального задания

```

1 import cmath
2 a = float(input("Введите a: "))
3 b = float(input("Введите b: "))
4 c = float(input("Введите c: "))
5 d = (b**2) - (4*a*c)
6 if d >= 0:
7     x1 = (-b + cmath.sqrt(d)) / (2*a)
8     x2 = (-b - cmath.sqrt(d)) / (2*a)
9     print("Корни уравнения:", x1, x2)
10 else:
11     x1 = (-b + cmath.sqrt(d)) / (2*a)
12     x2 = (-b - cmath.sqrt(d)) / (2*a)
13     print("Комплексные корни уравнения:", x1, x2)

```

if d >= 0

main

C:\Users\ACER\PycharmProjects\user2.1\venv\Scripts\python.exe C:\Users\ACER\PycharmProjects\user2.1\main.py

Введите a: 1

Введите b: 5

Введите c: 6

Корни уравнения: (-2+0j) (-3+0j)

Рисунок 16 - Программа и решение второго индивидуального задания

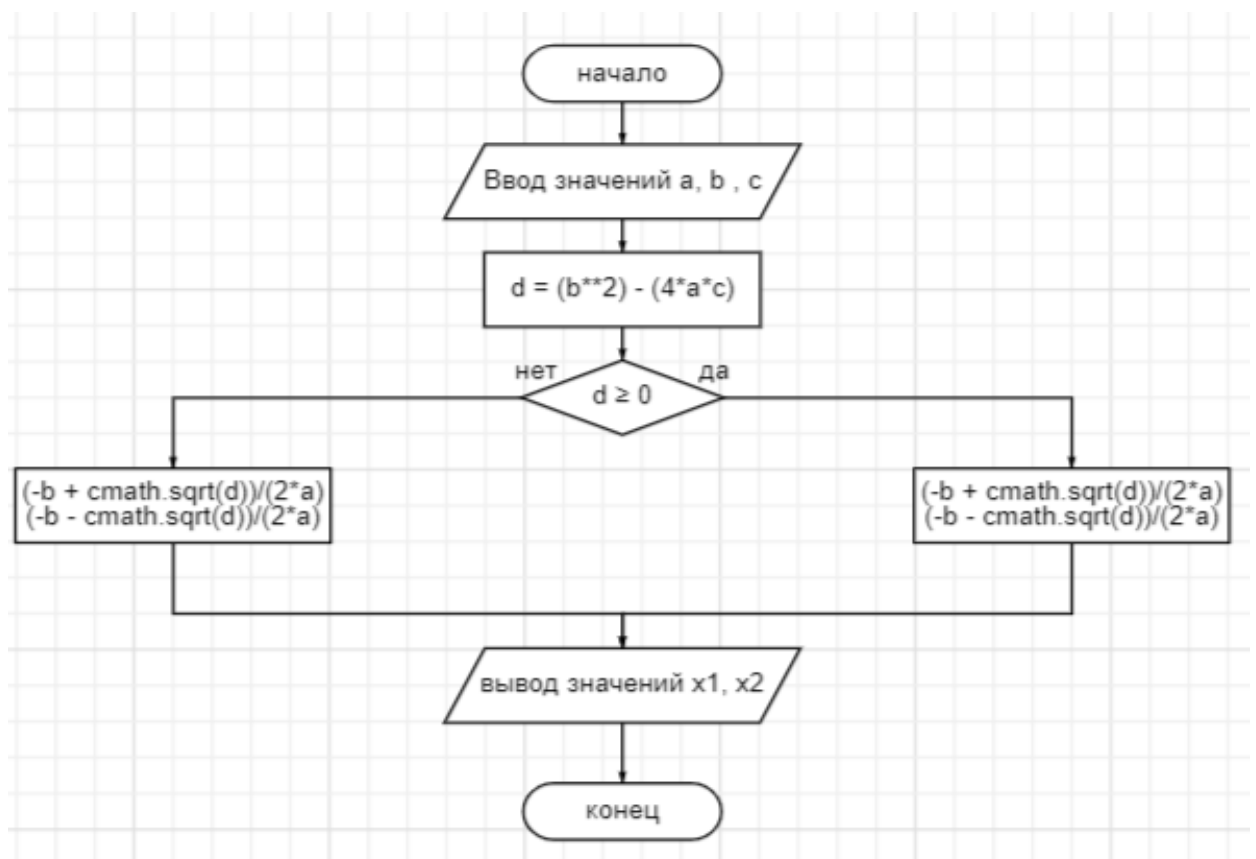
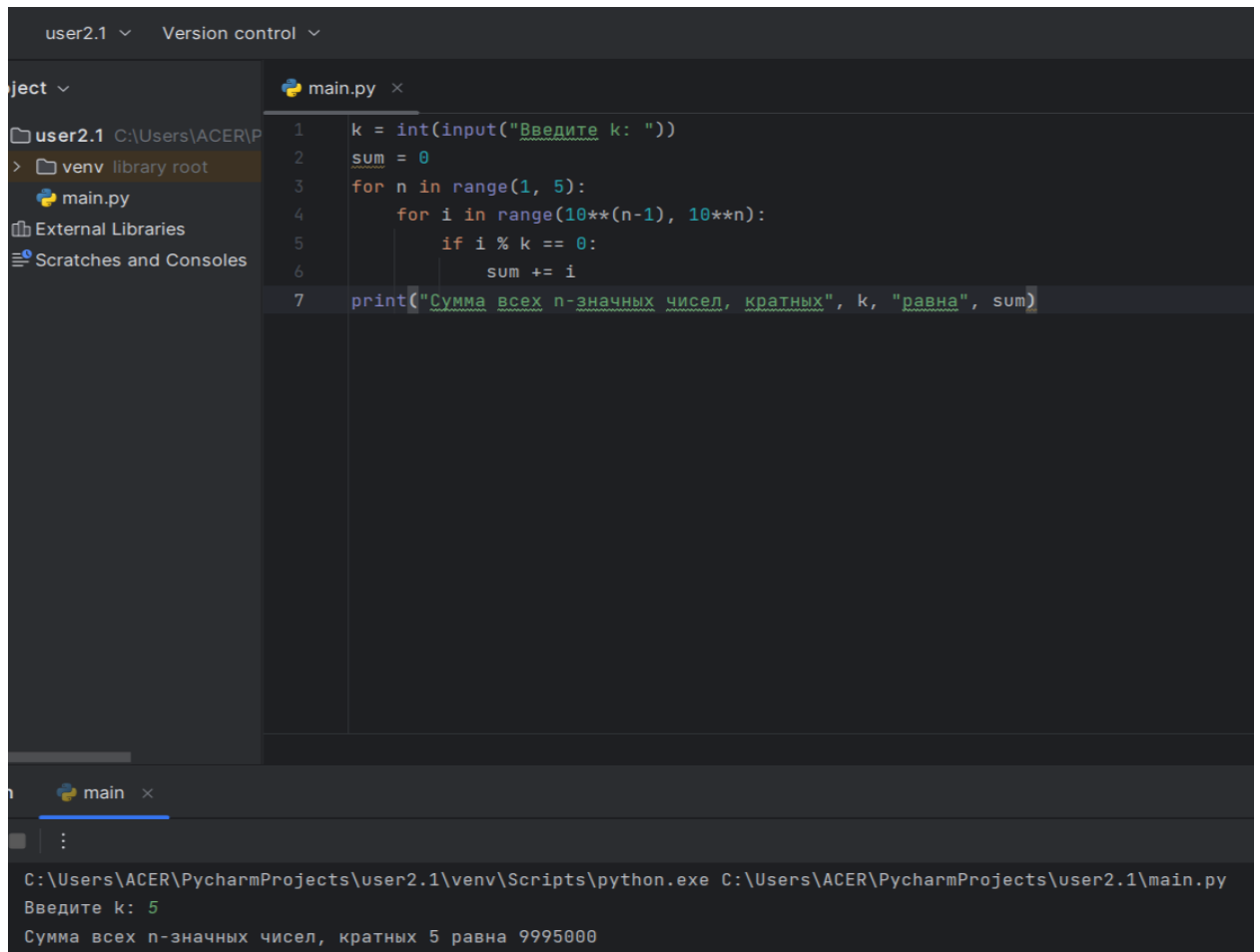


Рисунок 17 - UML-диаграмма деятельности второго индивидуального задания

Пример 3. Вычислить сумму всех n-значных чисел, кратных k ($1 \leq n \leq 4$).



The screenshot shows the PyCharm IDE interface. On the left, the project explorer shows a project named 'user2.1' with a file 'main.py'. The main editor displays the following Python code:

```
1 k = int(input("Введите k: "))
2 sum = 0
3 for n in range(1, 5):
4     for i in range(10**(n-1), 10**n):
5         if i % k == 0:
6             sum += i
7 print("Сумма всех n-значных чисел, кратных", k, "равна", sum)
```

Below the editor, the console window shows the execution output:

```
C:\Users\ACER\PycharmProjects\user2.1\venv\Scripts\python.exe C:\Users\ACER\PycharmProjects\user2.1\main.py
Введите k: 5
Сумма всех n-значных чисел, кратных 5 равна 9995000
```

Рисунок 18 - Программа и решение третьего индивидуального задания

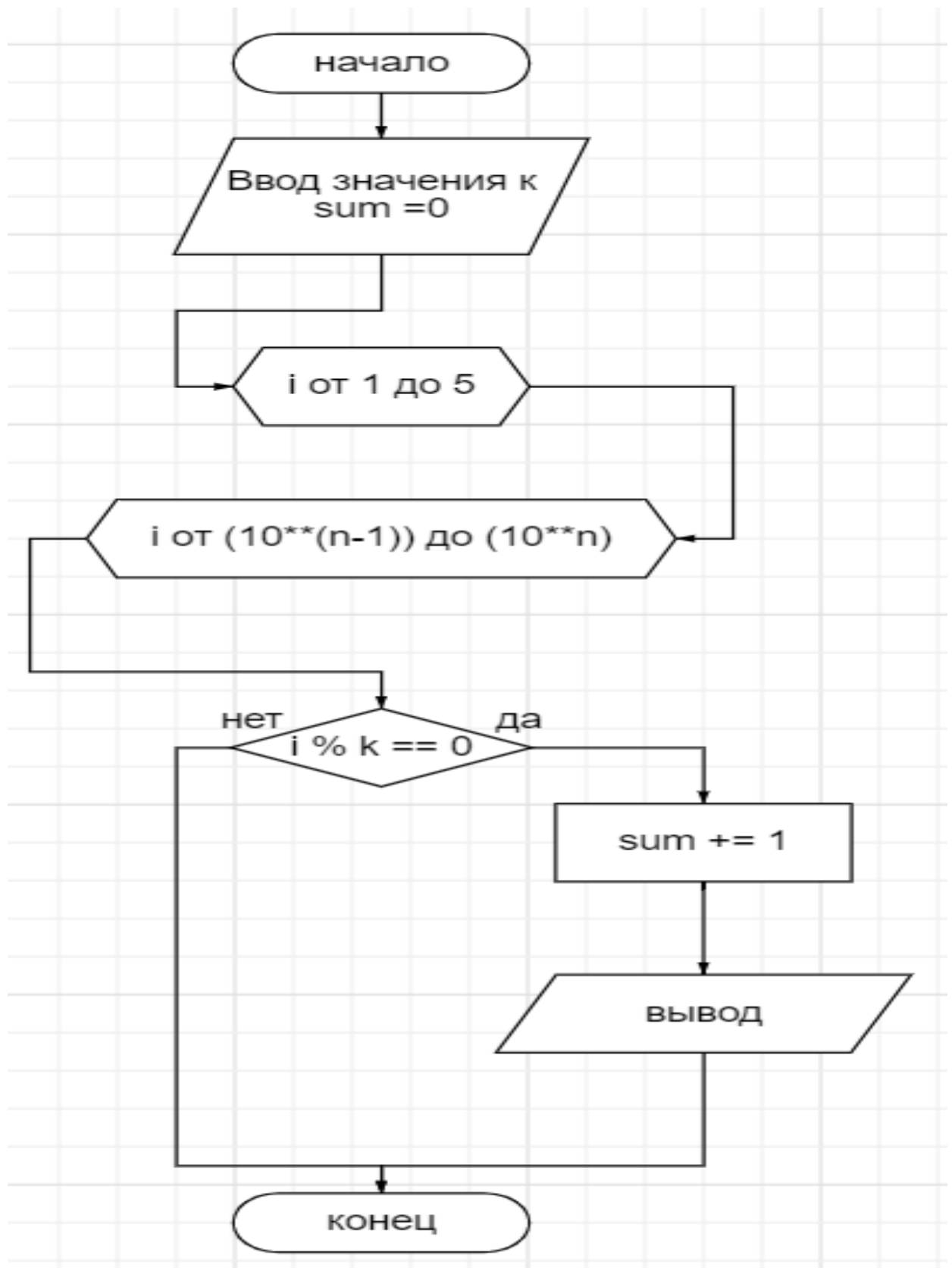


Рисунок 17 - UML-диаграмма деятельности третьего индивидуального задания

3. Клонирование репозитория. Использование команды git flow init.

```
C:\Users\ACER>git clone https://github.com/Dash-A1/2.2.git
Cloning into '2.2'...
remote: Enumerating objects: 36, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (32/32), done.
Receiving objects: 19% (7/36)used 0 (delta 0), pack-reused 0Receiving objects: 13% (5/36)
Receiving objects: 100% (36/36), 11.37 KiB | 3.79 MiB/s, done.
Resolving deltas: 100% (9/9), done.
```

Рисунок 18 – Результат клонирования

```
C:\Git\2.2-main>git flow init
Initialized empty Git repository in C:/Git/2.2-main/.git/
No branches exist yet. Base branches must be created now.
Branch name for production releases: [master]
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:/Git/2.2-main/.git/hooks]
```

Рисунок 19 – Результат git flow init

Ответы на контрольные вопросы

1. Для чего нужны диаграммы деятельности UML?

Диаграммы деятельности. Унифицированный язык моделирования (UML) является стандартным инструментом для создания «чертежей» программного обеспечения. С помощью UML можно визуализировать, специфицировать, конструировать и документировать артефакты программных систем. UML пригоден для моделирования любых систем: от информационных систем масштаба предприятия до распределенных Web-приложений и даже встроенных систем реального времени. Это очень выразительный язык, позволяющий рассмотреть систему со всех точек зрения, имеющих отношение к ее разработке и последующему разворачиванию.

2. Что такое состояние действия и состояние деятельности?

В потоке управления, моделируемом диаграммой деятельности, происходят различные события. Вы можете 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.

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

Существуют такие нотации, как: стрелки, решетки, декомпозиции, условные дуги.

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. 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.

5. Чем отличается разветвляющийся алгоритм от линейного?

В отличие от линейных алгоритмов, в которых команды выполняются последовательно одна за другой, в разветвляющиеся алгоритмы входит условие, в зависимости от выполнения или невыполнения которого выполняется та или иная последовательность команд (действий). В качестве условия в разветвляющемся алгоритме может быть использовано любое понятное исполнителю утверждение, которое может соблюдаться (быть истинно) или не соблюдаться (быть ложно).

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

Условный оператор или оператор ветвления — это оператор, конструкция языка программирования, обеспечивающая выполнение определённой команды (набора команд) только при условии истинности некоторого логического выражения, либо выполнение одной из нескольких команд (наборов команд) в зависимости от значения некоторого выражения.

Существует две основные формы условной инструкции, встречающиеся в реальных языках программирования: условный оператор (оператор if) и оператор многозначного выбора (переключатель, case, switch).

7. Какие операторы сравнения используются в Python?

В языке программирования Python для сравнения строк используют следующие операторы: оператор <, «меньше»; оператор <=, «меньше или равно»; оператор ==, «равно»; оператор !=, «не равно»; оператор >, «больше»; оператор >=, «больше или равно».

8. Что называется простым условием?

Простое условие - два выражения, связанные одним из знаков отношений: = (равно) > (больше) < (меньше) >= (больше либо равно) <= (меньше либо равно) <> (не равно). Примеры: $a > 0$; $a + c \leq 0$; $x \neq 0$; $a \leq d$.

9. Что такое составное условие?

Составные **условия - условия**, состоящие из двух или более **простых условий**, соединенных с помощью логических операций and (и), or (или), not (не).

10. Какие логические операторы допускаются при составлении сложных условий?

Три логических оператора, которые позволяют создавать сложные условия: and — логическое умножение; or — логическое сложение; not — логическое отрицание.

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

Так же внутри оператора ветвления **может** встречаться и другой оператор ветвления. Такой оператор ветвления называется вложенным. В этом случае важно не перепутать какая ветвь кода к какому оператору относится. Поэтому рекомендуется соблюдать отступы в исходном коде программы, чтобы не запутаться. `if <условие> then if <условие> then <оператор 1>`; Вложенный условный оператор.

12. Какой алгоритм является алгоритмом циклической структуры?

Алгоритмы, отдельные действия которых многократно повторяются, называются алгоритмами циклической структуры. Она позволяет существенно сократить объем алгоритма, представить его компактно за счет организации повторений большого числа одинаковых вычислений над разными данными для получения необходимого результата.

13. Типы циклов в языке Python.

В Python основных циклов всего два – «**while**» и «**for**». Первый используется тогда, когда заранее известно количество итераций, а второй – когда нужно выполнить перебор элементов. Но мало знать принципы работы циклов, в работе необходимы еще и операторы, которые делают «while» и «for» бесконечными или вложенными, прерывают их или меняют.

14. Назовите назначение и способы применения функции range

Функция range() возвращает объект, создающий последовательность чисел, начинающуюся с 0 (по умолчанию), последовательно увеличивающуюся (по умолчанию на 1) и останавливающуюся перед заданным числом (обязательный параметр). Обратите внимание, что range() возвращает не последовательность, а объект, вызывающий её.

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

```
range(15, 0 -2)
```

16. Могу ли быть циклы вложенными?

Вложенные циклы – это циклы, организованные в теле другого цикла. **Вложенный цикл** в тело другого цикла, называется **внутренним циклом**. Цикл, в теле которого существует **вложенный цикл**, называется **внешним**. Полное число исполнений **внутреннего цикла**, всегда равно произведению числа итераций **внутреннего цикла** на произведение чисел итераций всех **внешних циклов**.

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

Чтобы получить **бесконечный цикл** в Python надо просто задать условие, при котором всегда будет возвращаться True. Завершить работу программы можно будет, только принудительно закрыв программу или с помощью ключевого слова break.

18. Для чего нужен оператор break?

Инструкция break в языке программирования Python **прерывает выполнение блока кода.**

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

Оператор continue предназначен для **перехода к выполнению следующей итерации цикла.** Если в теле цикла встречается оператор continue, то: выполнение текущей итерации останавливается; происходит переход к следующей итерации цикла.

Оператор continue может быть применен во всех видах циклов: for, while, do-while.

В большинстве случаев вызов оператора continue осуществляется при выполнении некоторого условия в операторе if.

20. Для чего нужны стандартные потоки stdout и stderr?

Файловые объекты sys.stdin, sys.stdout и sys.stderr используются интерпретатором для стандартного ввода, вывода и ошибок: sys.stdin - используется для всех интерактивных входных данных, включая вызовы input(); sys.stdout - используется для вывода оператором print() и выражений, которые возвращают значение, а также для подсказок в функции input(); sys.stderr - сообщения об ошибках и собственные запросы переводчика.

21. Каково назначение функции exit?

Функция exit() модуля sys - это быстрый способ выйти из программы при возникновении ошибки. Эта **функция** реализуется путем вызова исключения SystemExit, и, следовательно, выполняются действия по очистке, указанные в предложениях finally операторов try. Так же можно перехватить попытку выхода на внешнем уровне.

Вывод: в ходе лабораторной работы приобрела навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Освоила операторы языка Python версии 3.x if, while, for, break и continue, позволяющих реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.