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»

Выполнил:
Горбунов Данила Евгеньевич
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», очная 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 , позволяющие реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.

Ход работы

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python. Выполнил клонирование созданного репозитория.
2. Дополнил файл .gitignore необходимыми правилами.
3. Организовал созданный репозиторий в соответствие с моделью ветвления git-flow.
4. Изучил рекомендации к оформлению исходного кода на языке Python PEP-8.
5. Проработал примеры лабораторной работы. Создал для каждого примера отдельный модуль языка Python. Привел в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных вводимых с клавиатуры. Для примеров 4 и 5 построил UML-диаграммы деятельности.

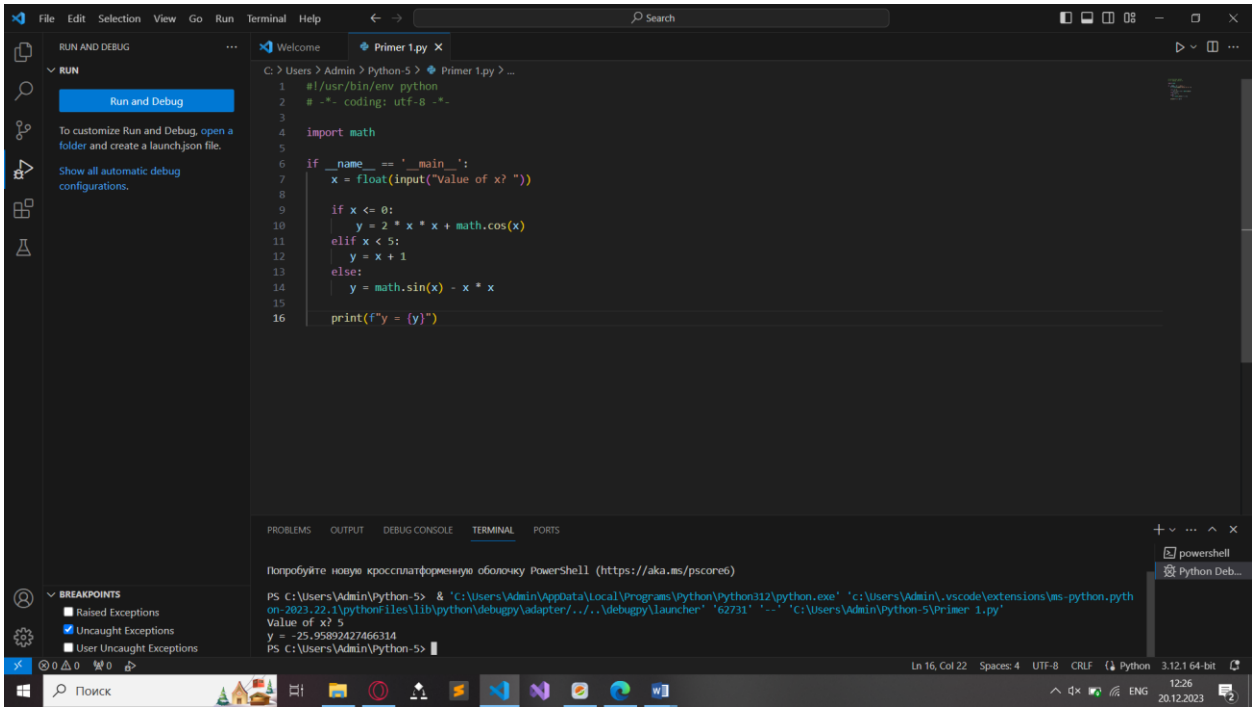

Рисунок 1. Результат работы программы из примера 1

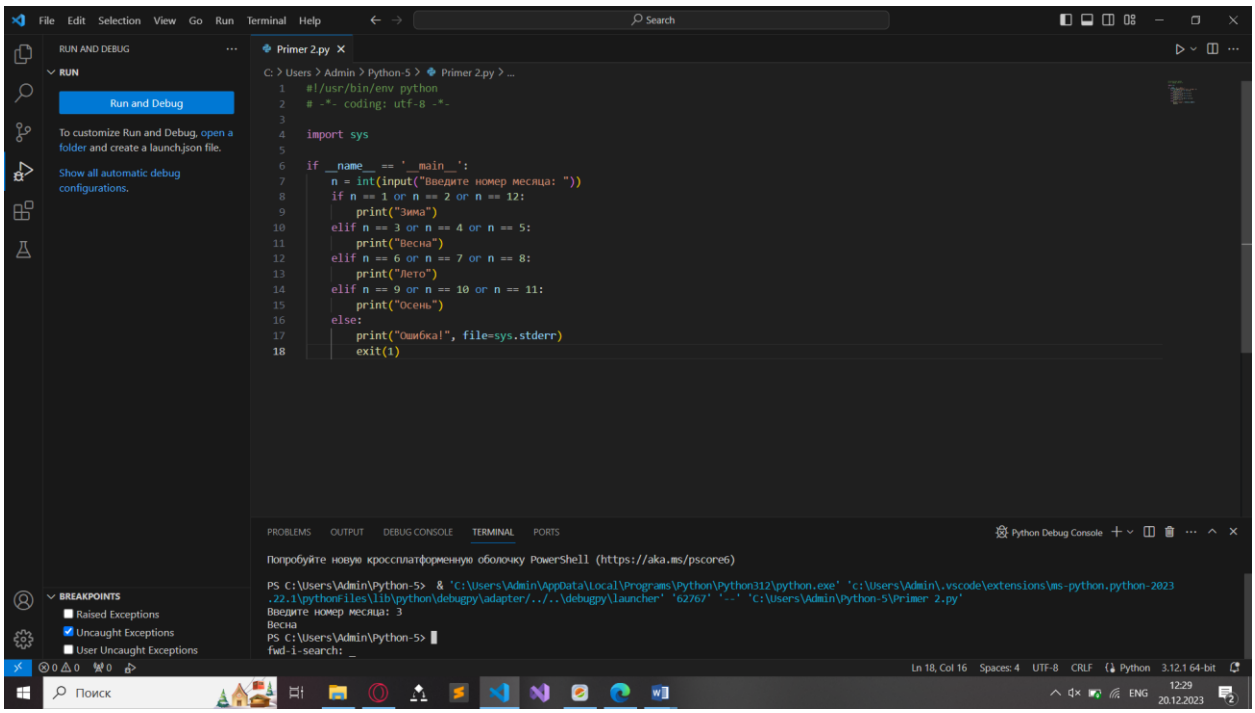

Рисунок 2. Результат работы программы из примера 2

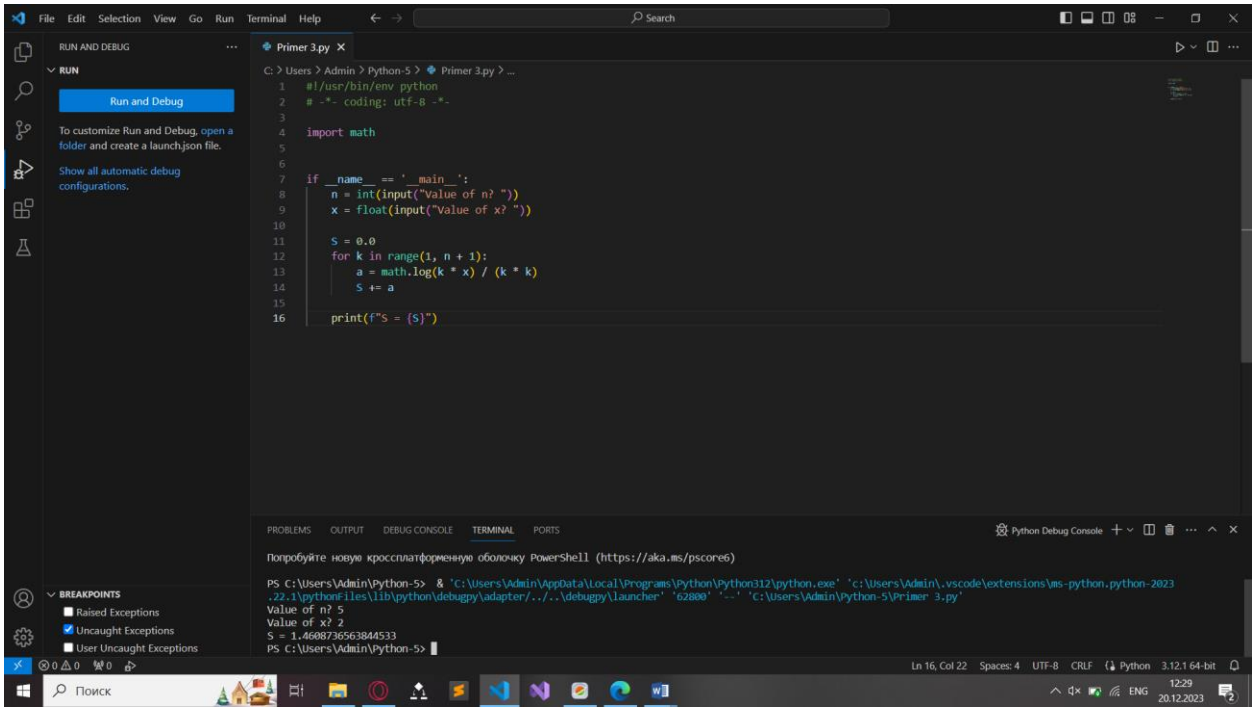

Рисунок 3. Результат работы программы из примера 3

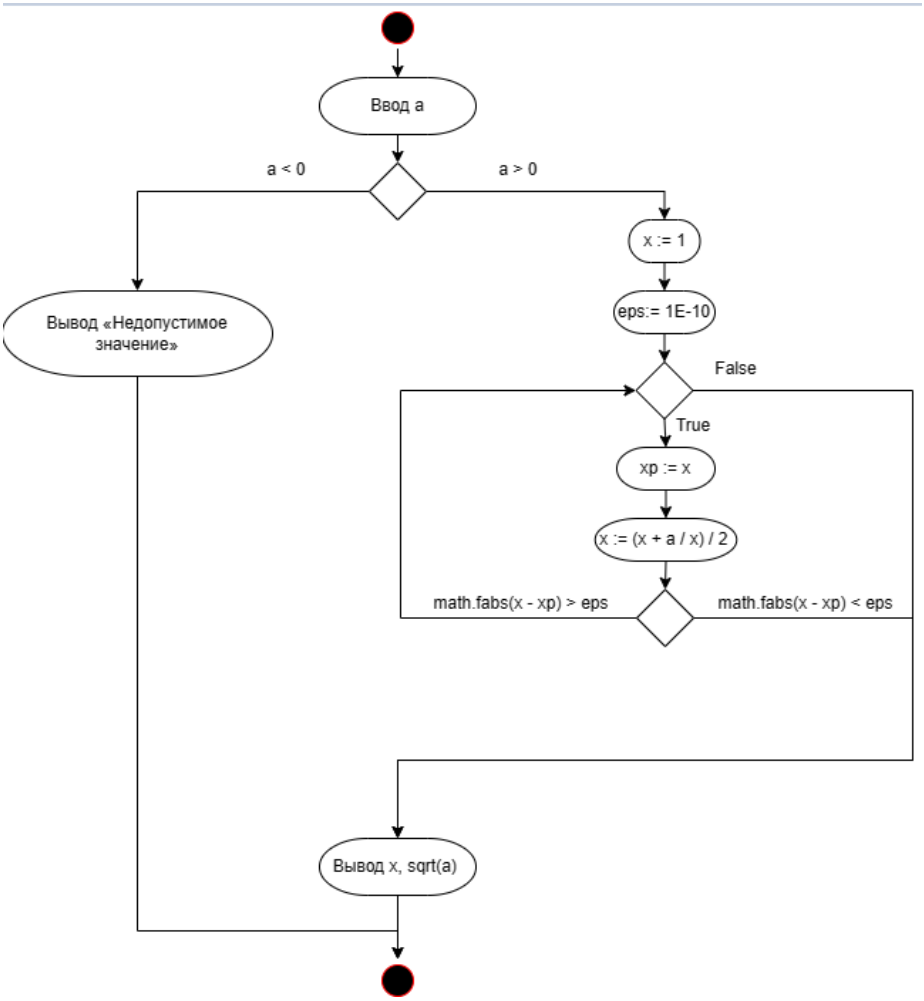

Рисунок 4. UML-диаграмма для программы из примера 4

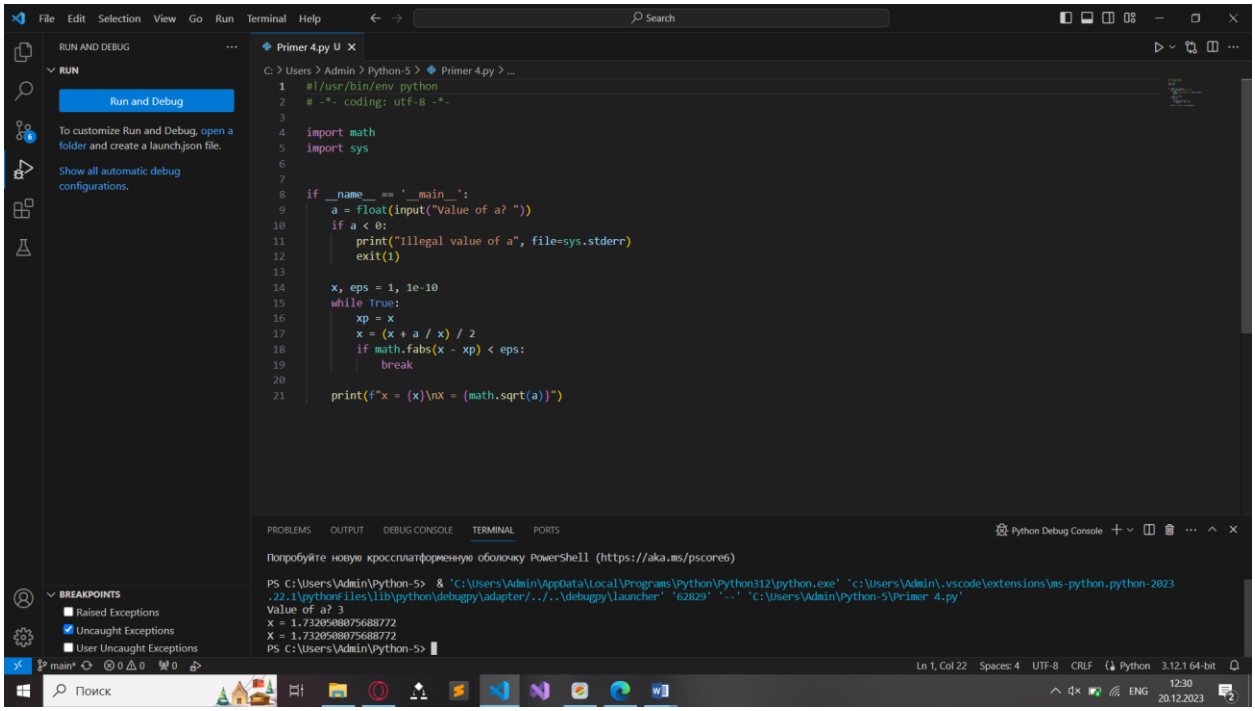

Рисунок 5. Результат работы программы из примера 4

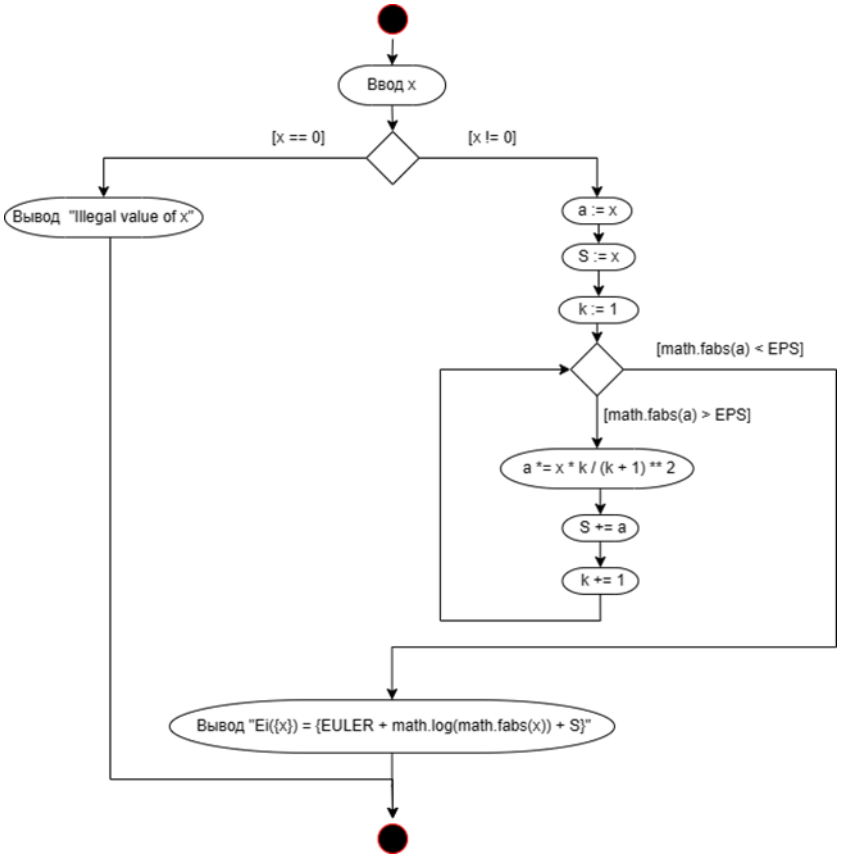

Рисунок 6. UML-диаграмма для программы из примера 5

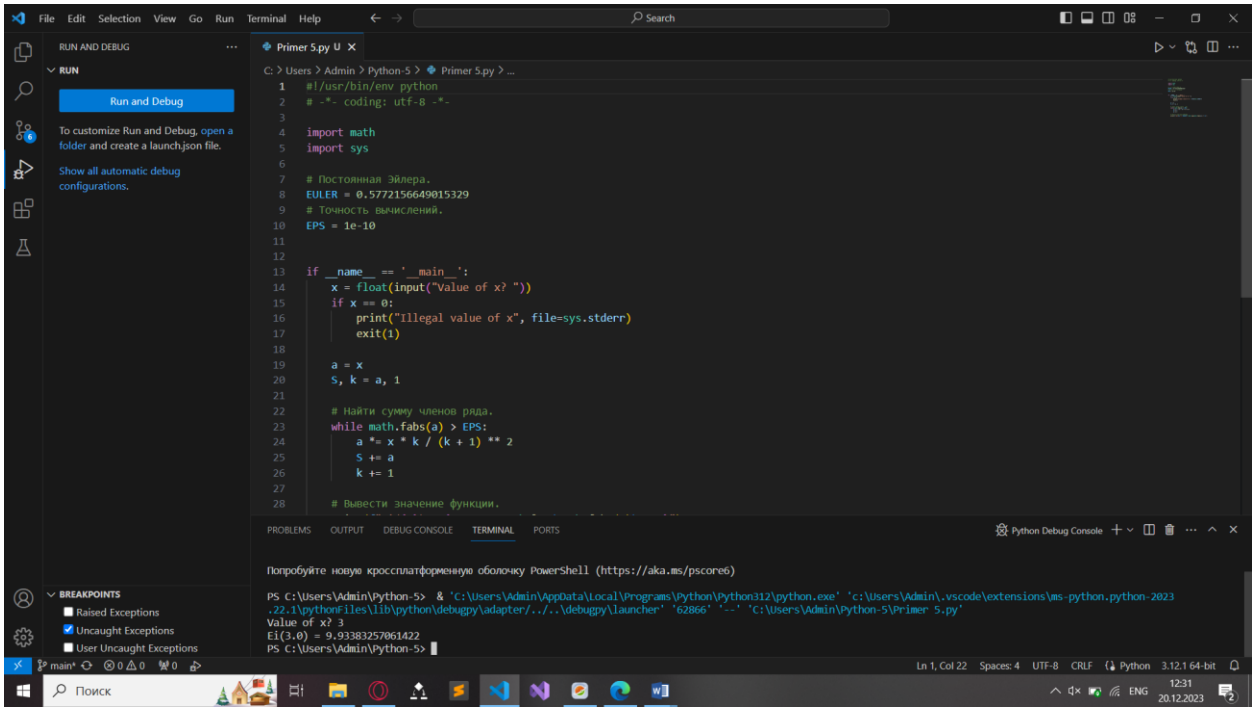

Рисунок 7. Результат работы программы из примера 5

6. Выполнил индивидуальные задания. А также задание повышенной сложности. Привёл в отчете скриншоты работ программ и UML-диаграммы деятельности решения индивидуальных заданий.

Задание 1. Дано число m (1 <= m <= 12). Определить полугодие, на которое приходится месяц с номером m и количество дней в том месяце (год не високосный).

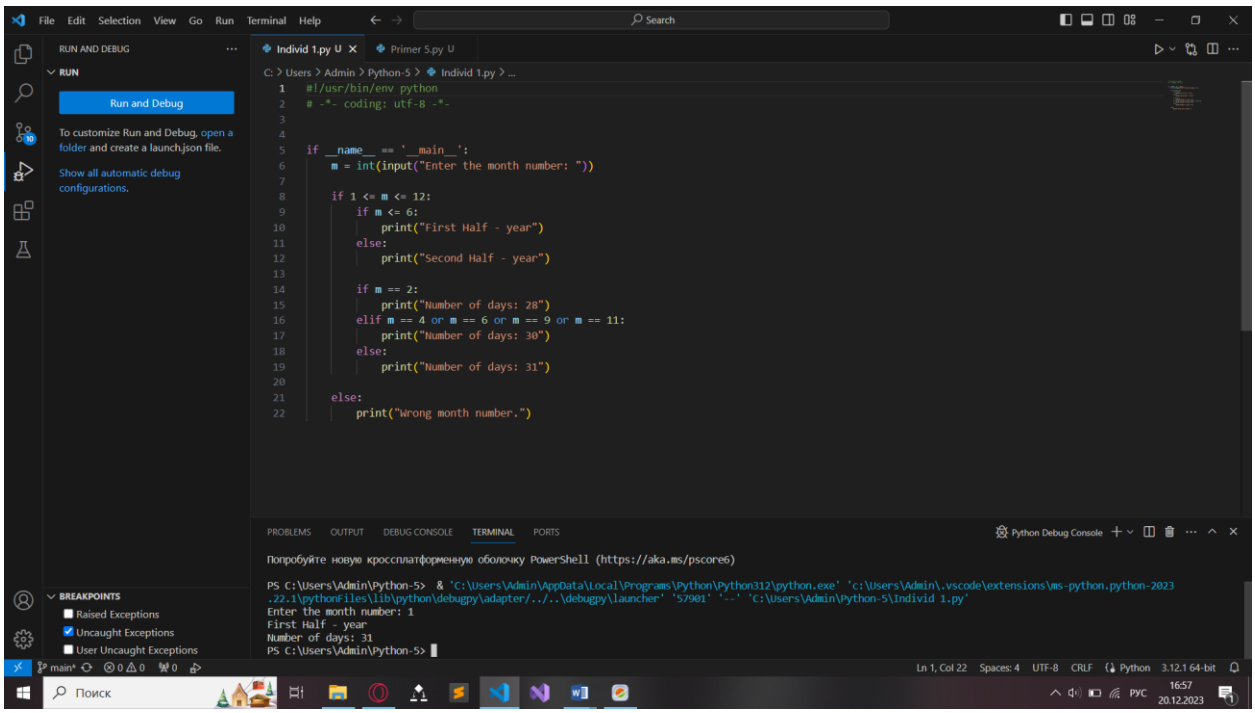

Рисунок 8. Результат работы программы из 1 индивидуального задания

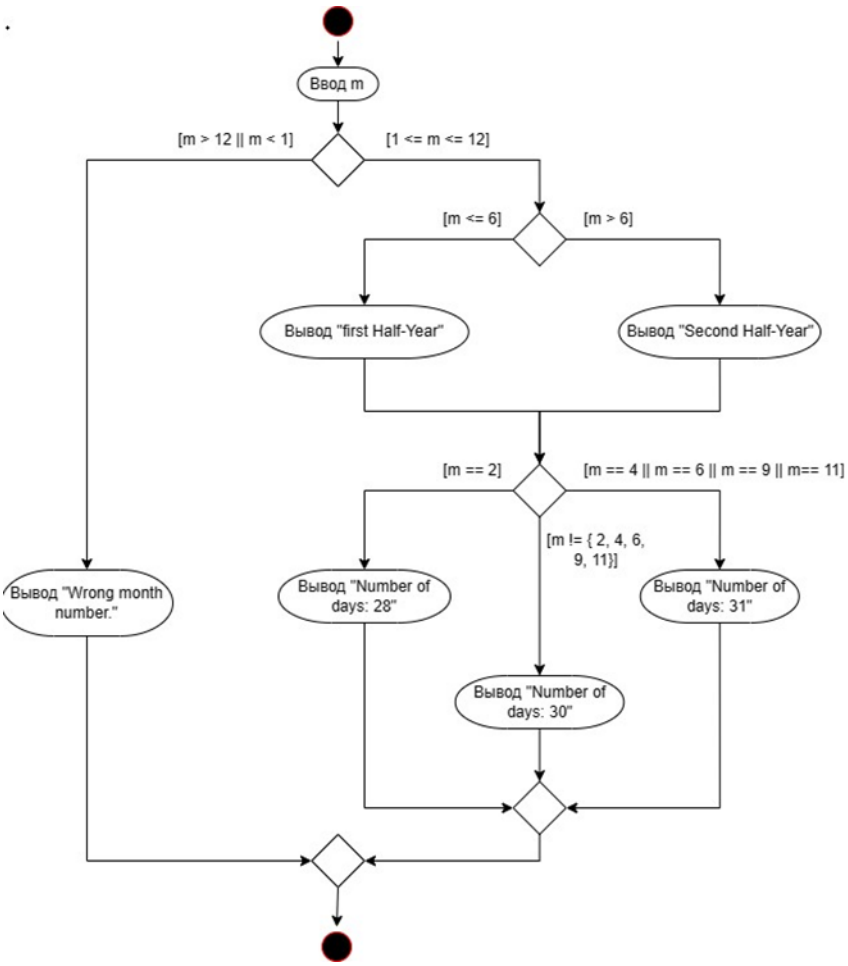

Рисунок 9. UML-диаграмма для программы из 1 индивидуального задания

Задание 2. Найти координаты точки пересечения прямых заданных уравнениями a1x + b1y + c1 = 0 и a2x + b2y + c2 = 0, либо сообщить совпадают, параллельны или не существуют.

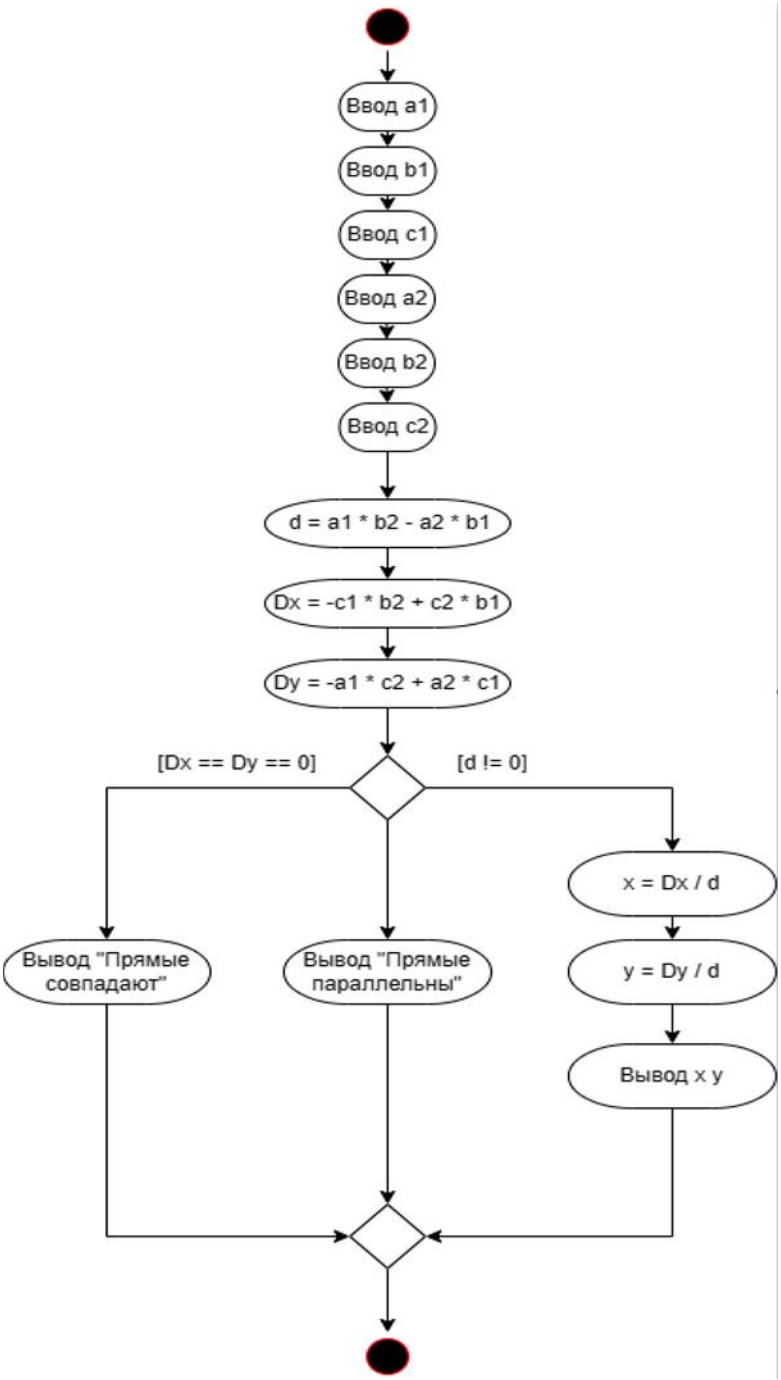

Рисунок 10. UML-диаграмма для программы из 2 индивидуального задания

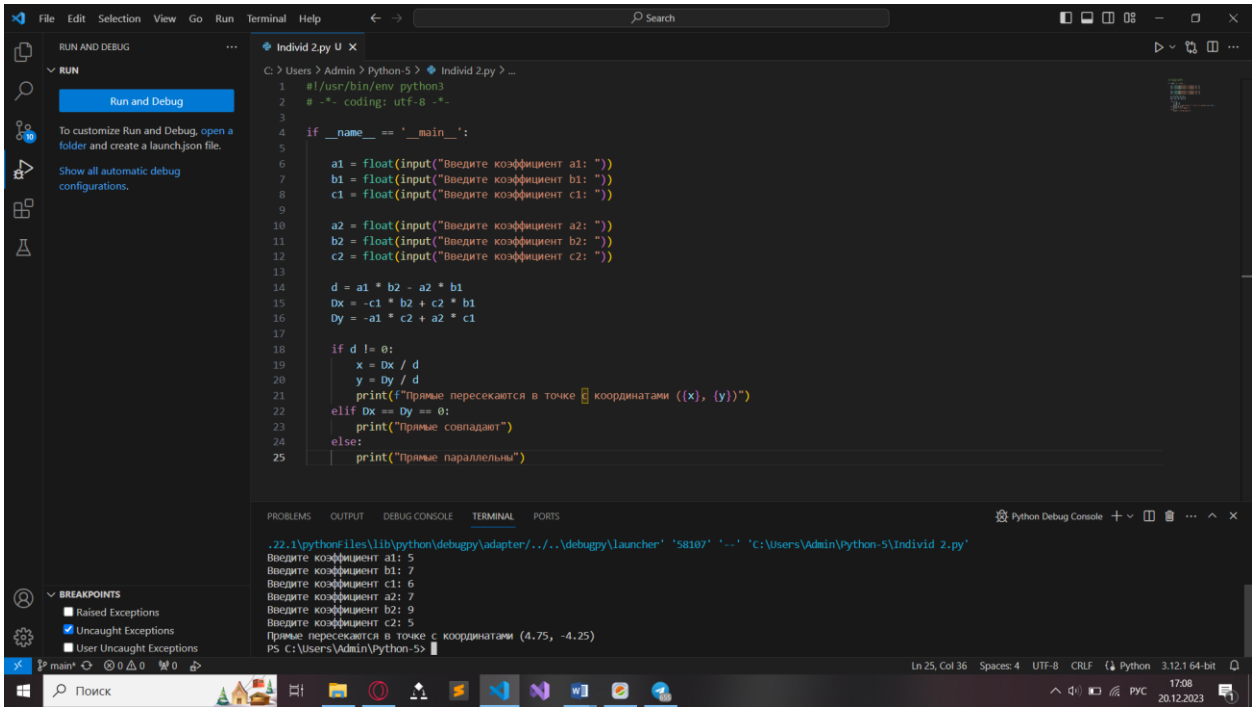

Рисунок 11. Результат работы программы из 2 индивидуального задания

Задание 3. Сумма цифр трехзначного числа кратна 7. Само число также делится на 7 Найти все такие числа.

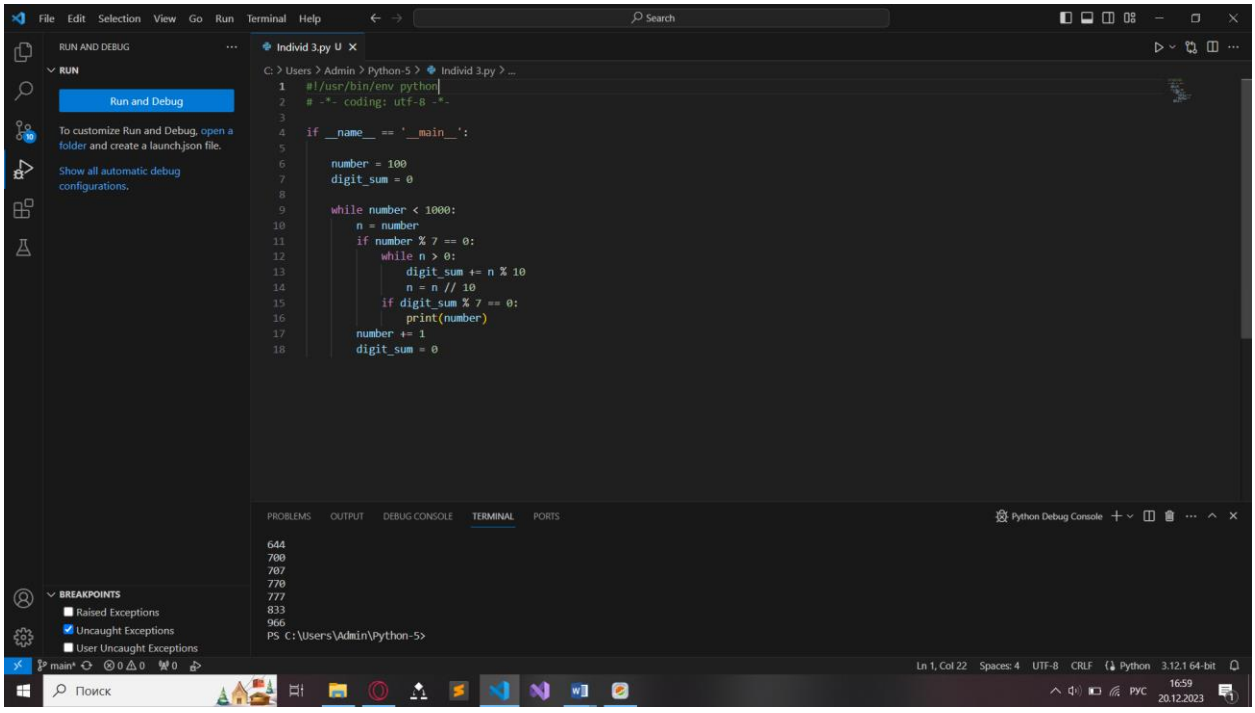

Рисунок 12. Результат работы программы из 3 индивидуального задания

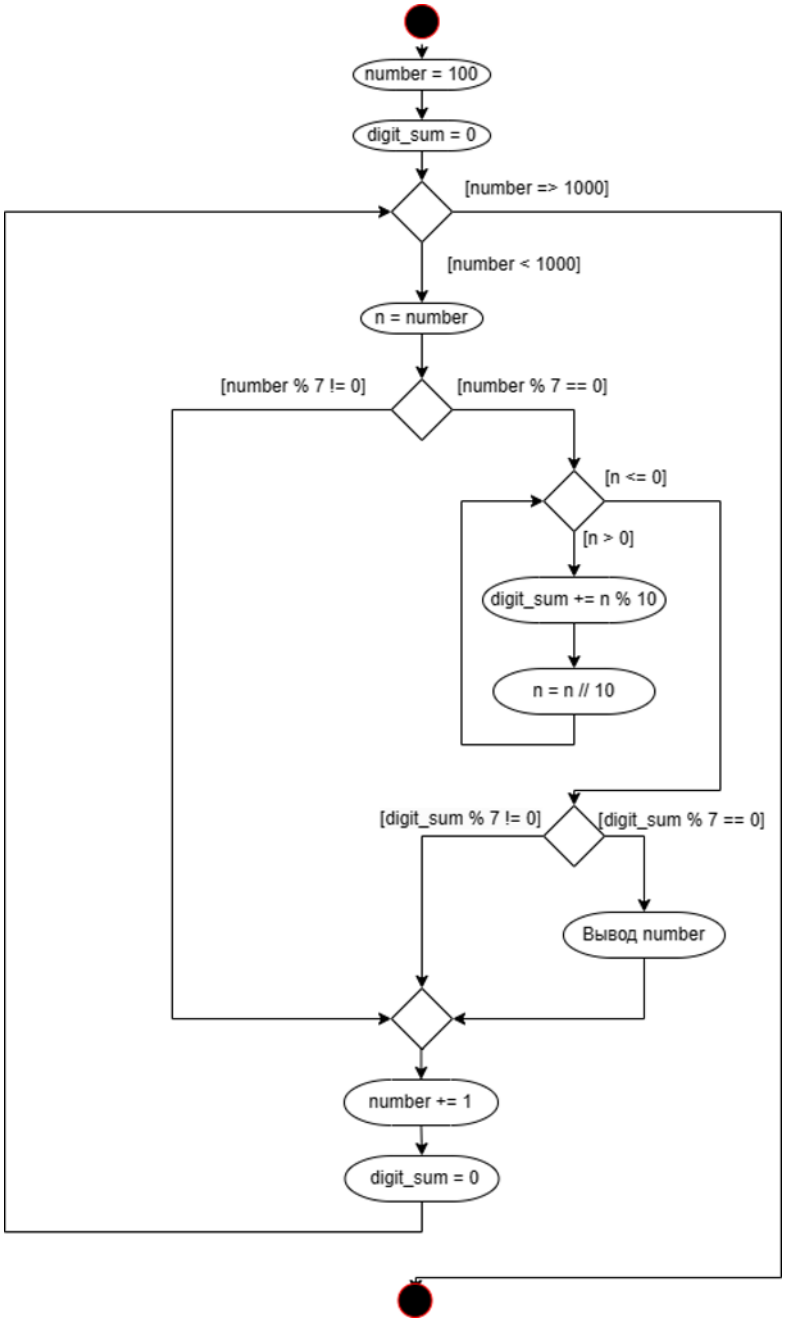

Рисунок 13. UML-диаграмма для программы из 3 индивидуального задания

Задание повышенной сложности. Составить UML-диаграмму деятельности, программу и произвести вычисления вычисление значения специальной функции по ее разложению в ряд с точностью , аргумент функции вводится с клавиатуры. Исследуемая функция:

$$J_n(x) = \left(\frac{x}{2}\right)^n \sum_{k=0}^{\infty} \frac{(-x^2/4)^k}{k!(k+n)!}.$$

Общий член ряда: k $\frac{(-\frac{x^2}{\_})^k}{k!(k+n)!}$

Следующий член ряда: $k+1$ $\frac{(-\frac{x^2}{\_})^{k+1}}{(k+1)!((k+1)+n)!}$

Неупрощѐнное отношение: $\frac{ak+1}{a_k}$ $\frac{(-1)x^{2(k+1)}}{4^{k+1}(k+1)!((k+1)+n)!} * \frac{k!(k+n)!4^k}{(-1)^k x^{2k}}$

Отношение: $\frac{ak+1}{a_k}$ $\frac{x^2}{4(k+1)(k+n+1)}$

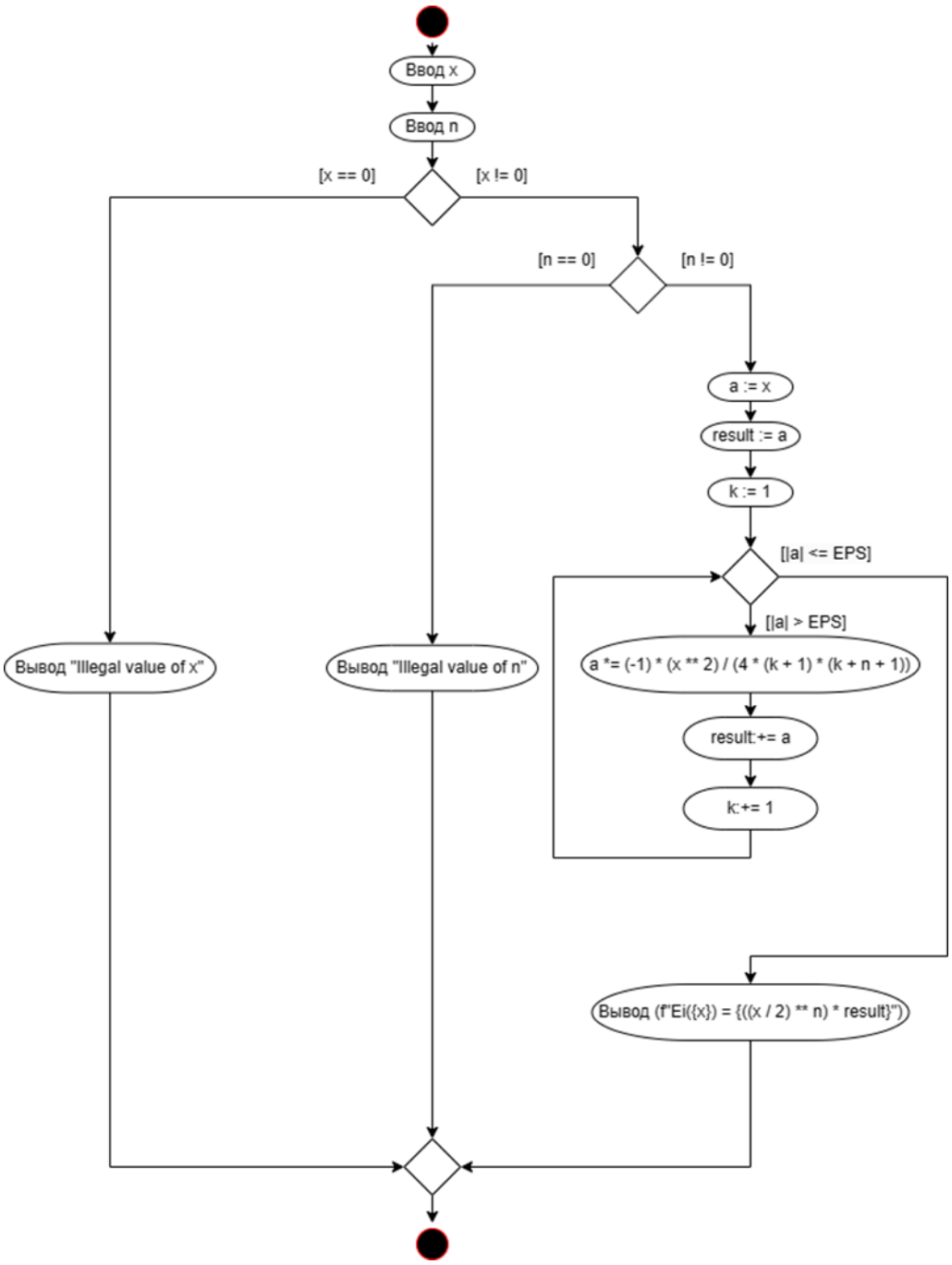

Рисунок 14. UML-диаграмма для программы из индивидуального задания повышенной сложности

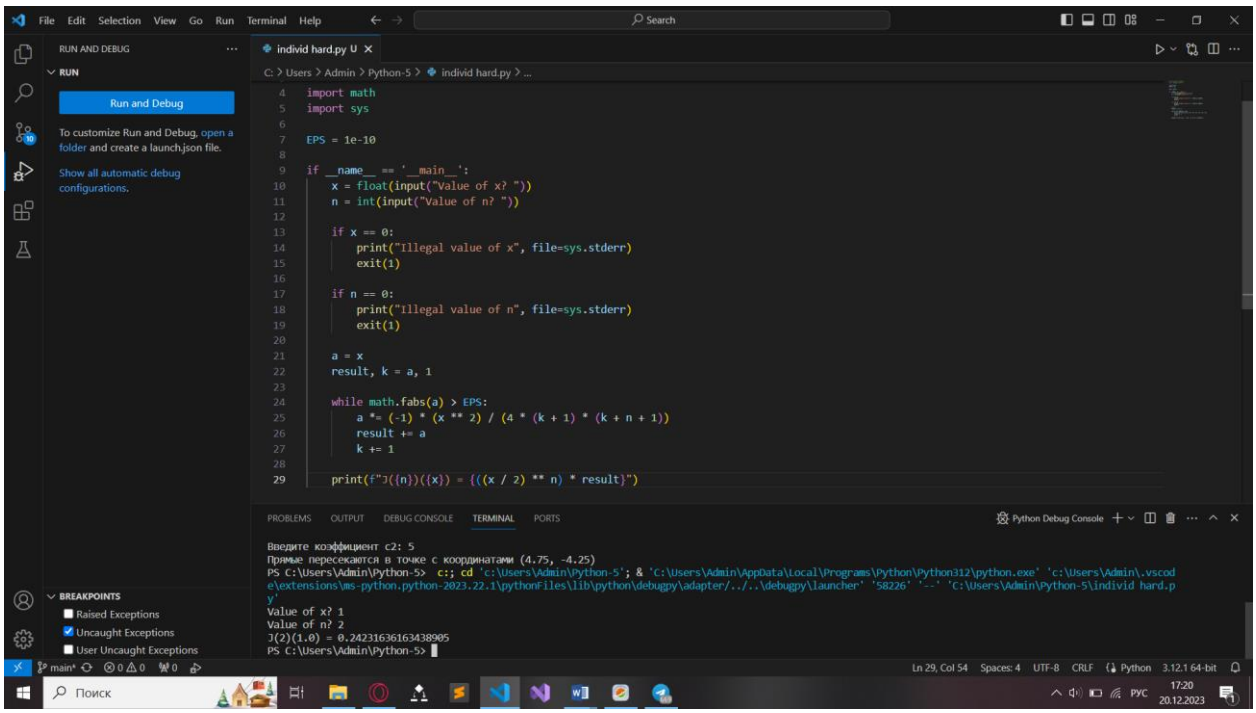

Рисунок 15. Результат работы программы из индивидуального задания повышенной сложности

Контрольные вопросы

1. Для чего нужны диаграммы деятельности UML?

Диаграммы деятельности UML используются для визуализации и моделирования последовательности действий и процессов в системе. Они помогают описать, какие действия выполняются, какие ресурсы используются, и какие условия должны быть выполнены для перехода от одного состояния к другому. Диаграммы деятельности UML широко применяются в разработке программного обеспечения, бизнес-анализе и проектировании бизнес-процессов.

2. Что такое состояние действия и состояние деятельности?

1) Состояние действия в диаграммах деятельности UML представляет конкретное действие или операцию, которую выполняет объект или система в определенный момент времени. Состояние действия может быть представлено в виде прямоугольника с закругленными углами и названием действия внутри.

2) Состояние деятельности в диаграммах деятельности UML представляет набор связанных действий, которые выполняются

последовательно или параллельно. Состояние деятельности может быть представлено в виде прямоугольника с закругленными углами и названием деятельности внутри.

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

В диаграммах деятельности UML существуют следующие нотации для обозначения переходов и ветвлений:

1) Стрелки с направлением указывают на переходы между состояниями или действиями.

2) Ромбы используются для обозначения ветвлений и объединений. Ветвление происходит, когда одно состояние или действие может привести к нескольким возможным состояниям или действиям. Объединение происходит, когда несколько состояний или действий объединяются в одно состояние или действие.

3) Условия переходов обычно указываются рядом со стрелками или ромбами, чтобы показать, какие условия должны быть выполнены для перехода.

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Алгоритм разветвляющейся структуры – это алгоритм, который включает в себя ветвления и условные операторы для принятия решений в зависимости от определенных условий. Он позволяет программе выполнять различные действия в зависимости от того, выполняется ли определенное условие или нет. Примером алгоритма разветвляющейся структуры может быть условный оператор "if-else" в языке программирования Python.

5. Чем отличается разветвляющийся алгоритм от линейного? Разветвляющийся алгоритм отличается от линейного алгоритма тем,

что разветвляющийся алгоритм содержит ветвления и условные операторы, которые позволяют программе принимать решения и выполнять различные действия в зависимости от определенных условий. Линейный

алгоритм, с другой стороны, выполняет последовательные действия без ветвлений или

условных операторов. В линейном алгоритме действия выполняются одно за другим в определенном порядке без возможности изменения последовательности выполнения.

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

Условный оператор – это конструкция в программировании, которая позволяет программе принимать решения и выполнять различные действия в зависимости от определенных условий. В языке программирования Python существуют две формы условного оператора: if-else – это форма условного оператора, которая позволяет программе выполнить одно действие, если условие истинно, и другое действие, если условие ложно. if-elif-else – это форма условного оператора, которая позволяет программе проверить несколько условий последовательно и выполнить соответствующее действие для первого истинного условия. Если ни одно из условий не является истинным, выполняется блок кода в разделе else.

7. Какие операторы сравнения используются в Python?

В языке программирования Python используются следующие операторы сравнения:

1) == (равно) - проверяет, равны ли два значения.

2) != (не равно) - проверяет, не равны ли два значения.

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

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

5) >= (больше или равно) - проверяет, является ли первое значение больше или равным второму.

6) <= (меньше или равно) - проверяет, является ли первое значение меньше или равным второму.

Эти операторы могут использоваться в условных операторах для сравнения значений и принятия решений.

8. Что называется простым условием? Приведите примеры.

Простое условие – это условие, которое содержит одно сравнение или проверку. В простых условиях используются операторы сравнения для сравнения значений и принятия решений на основе результатов сравнения.

9. Что такое составное условие? Приведите примеры.

Составное условие – это условие, которое состоит из нескольких простых условий, объединенных логическими операторами. Составное условие позволяет программе проверять несколько условий одновременно и принимать решения на основе их комбинации. Примеры составных условий в языке программирования Python:

10. Какие логические операторы допускаются при составлении сложных условий?

При составлении сложных условий в языке программирования Python допускаются следующие логические операторы:

1) and - логическое "и". Возвращает True, если оба условия истинны.

2) or - логическое "или". Возвращает True, если хотя бы одно из условий истинно.

3) not - логическое "не". Инвертирует значение условия.

Эти операторы позволяют комбинировать простые условия в составные условия и принимать решения на основе их комбинации.

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

Да, оператор ветвления в языке программирования Python может содержать внутри себя другие ветвления. Это называется вложенными ветвлениями или вложенными условными операторами. Вложенные ветвления позволяют программе выполнять различные действия в зависимости от нескольких условий, включая проверку условий внутри других условий.

12. Какой алгоритм является алгоритмом циклической структуры?

Алгоритм циклической структуры – это алгоритм, который включает в себя циклы для повторения определенных действий или блоков кода. Циклическая структура позволяет программе выполнять одни и те же действия несколько раз до выполнения определенного условия. Примером алгоритма циклической структуры является цикл "for" или цикл "while" в языке программирования Python.

13. Типы циклов в языке Python.

В языке программирования Python существуют два основных типа циклов:

1) Цикл for - выполняет набор действий для каждого элемента в заданной последовательности. Цикл for обычно используется, когда заранее известно количество повторений или когда нужно перебрать элементы в списке, кортеже или другой последовательности.

2) Цикл while - выполняет набор действий до тех пор, пока условие истинно. Цикл while обычно используется, когда количество повторений неизвестно заранее или когда нужно повторять действия до выполнения определенного условия.

14. Назовите назначение и способы применения функции range.

Функция range в языке программирования Python используется для создания последовательности чисел. Она имеет следующий синтаксис:

range(start, stop, step)

1) start (необязательный) - начальное значение последовательности (по умолчанию 0).

2) stop (обязательный) - конечное значение последовательности (не включается в последовательность).

3) step (необязательный) - шаг, с которым генерируются числа (по умолчанию 1).

Функция range может использоваться для создания циклов, перебора значений, генерации списков и других задач, где требуется последовательность чисел.

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

Для организации перебора значений от 15 до 0 с шагом 2 с помощью функции range, можно использовать следующий код:

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

16. Могут ли быть циклы вложенными?

Да, циклы в языке программирования Python могут быть вложенными, то есть один цикл может находиться внутри другого цикла. Вложенные циклы позволяют выполнять повторяющиеся действия внутри других повторяющихся действий.

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

Бесконечный цикл образуется, когда условие цикла всегда остается истинным, и цикл продолжает выполняться бесконечно. Например, если условие цикла всегда равно True или если условие никогда не изменяется, то цикл будет выполняться бесконечно.

Чтобы выйти из бесконечного цикла, можно использовать операторы break или условие, которое станет ложным в определенный момент времени. Оператор break позволяет прервать выполнение цикла и выйти из него досрочно, даже если условие цикла остается истинным.

18. Для чего нужен оператор break?

Оператор break используется в циклах для прерывания выполнения цикла и выхода из него досрочно. Когда оператор break достигается внутри

цикла, выполнение цикла немедленно прекращается, и управление передается к следующей инструкции после цикла.

Оператор break полезен, когда требуется прервать выполнение цикла, когда определенное условие выполняется, или когда достигнута определенная точка в программе, и дальнейшее выполнение цикла не требуется.

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

Оператор continue используется в циклах для пропуска текущей итерации цикла и перехода к следующей итерации. Когда оператор continue достигается внутри цикла, оставшаяся часть текущей итерации пропускается, и управление передается к следующей итерации цикла.

Оператор continue полезен, когда требуется пропустить выполнение определенных действий в цикле, но продолжить выполнение цикла с следующей итерации. Например, можно использовать оператор continue для пропуска выполнения некоторых действий в цикле, если определенное условие выполняется.

20. Для чего нужны стандартные потоки stdout и stderr?

Стандартные потоки stdout (стандартный вывод) и stderr (стандартный поток ошибок) являются каналами, используемыми для вывода информации из программы.

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

2) stderr - используется для вывода сообщений об ошибках или другой информации об ошибках, которые могут возникнуть в программе. Этот поток обычно также направляется в консоль или другое устройство вывода, но может быть перенаправлен в файл или другой поток.

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

В Python для организации вывода в стандартный поток stderr можно использовать модуль sys. Вот пример кода:

```
import sys
sys.stderr.write("Это сообщение об ошибке\n")
```

В этом примере используется функция write из модуля sys, чтобы написать сообщение в стандартный поток ошибок stderr. Сообщение будет выведено в консоль или другое устройство вывода, на которое направлен поток ошибок.

22. Каково назначение функции exit?

Функция exit используется в Python для немедленного завершения программы. Когда функция exit вызывается, выполнение программы прекращается, и программа выходит из программы с указанным кодом завершения.

Назначение функции exit заключается в том, чтобы предоставить возможность явно завершить программу в определенных ситуациях. Например, если возникла критическая ошибка или требуется принудительно остановить выполнение программы, можно вызвать функцию exit с соответствующим кодом завершения.

Вывод: был приобретен набор навыков программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Были освоены операторы языка Python версии 3.x if, while, for, break и continue, позволяющие реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.