МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Департамент цифровых, робототехнических систем и электроники

«Условные операторы и циклы в языке Python»
Отчет по лабораторной работе № 3
по дисциплине «Программирование на Python»
Вариант 5

Выполнил студент группы ИВТ-б-о-24-1
Грабарь Артемий Павлович
«__» октября 2025г.
Подпись студента_____
Работа защищена « »_____20__г.
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2025

Цель работы: 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.

Ссылка на GitHub: https://github.com/Arhi258/Laba_3.git

Задание:

1. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.
2. Выполнить клонирование данного репозитория.
3. Дополнить файл .gitignore необходимыми правилами для работы с IDE PyCharm.
4. Самостоятельно изучить рекомендации к оформлению исходного кода на языке Python PEP-8. Выполнить оформление исходных примеров лабораторной работы и индивидуальных созданий в соответствии с PEP-8.
5. Создать проект PyCharm в папке репозитория.
6. Проработать примеры лабораторной работы. Создать для каждого примера отдельный модуль языка Python. Зафиксировать изменения в репозитории.
7. Для примеров 4 и 5 построить UML-диаграмму деятельности.
8. Выполните индивидуальные задания, согласно своего варианта. Задание 1: с клавиатуры вводится цифра m (от 1 до 4). Вывести на экран названия месяцев, соответствующих времени года с номером m (считать зиму временем года № 1). Задание 2: определить принадлежит ли точка A(a, b) кольцу определяемому окружностями $x^2+y^2=1$ и $x^2+y^2=0.25$. Задание 3: одноклеточная амеба каждые три часа делится на 2 клетки. Определить, сколько будет клеток через 6 часов. Задание повышенной сложности: первый интеграл Френеля: $C(x) = \int_0^x \cos\left(\frac{\pi}{2}t^2\right) dt = \sum_{n=0}^{\infty} \frac{(-1)^n (\pi/2)^{2n}}{(2n)!(4n+1)}$.

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

10. Зафиксируйте сделанные изменения в репозитории.

11. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.

12. Отправьте сделанные изменения на сервер GitHub.

Ход работы:

1. Был создан общедоступный репозиторий на GitHub, с использованием лицензии MIT и языка программирования Python (рис 1).

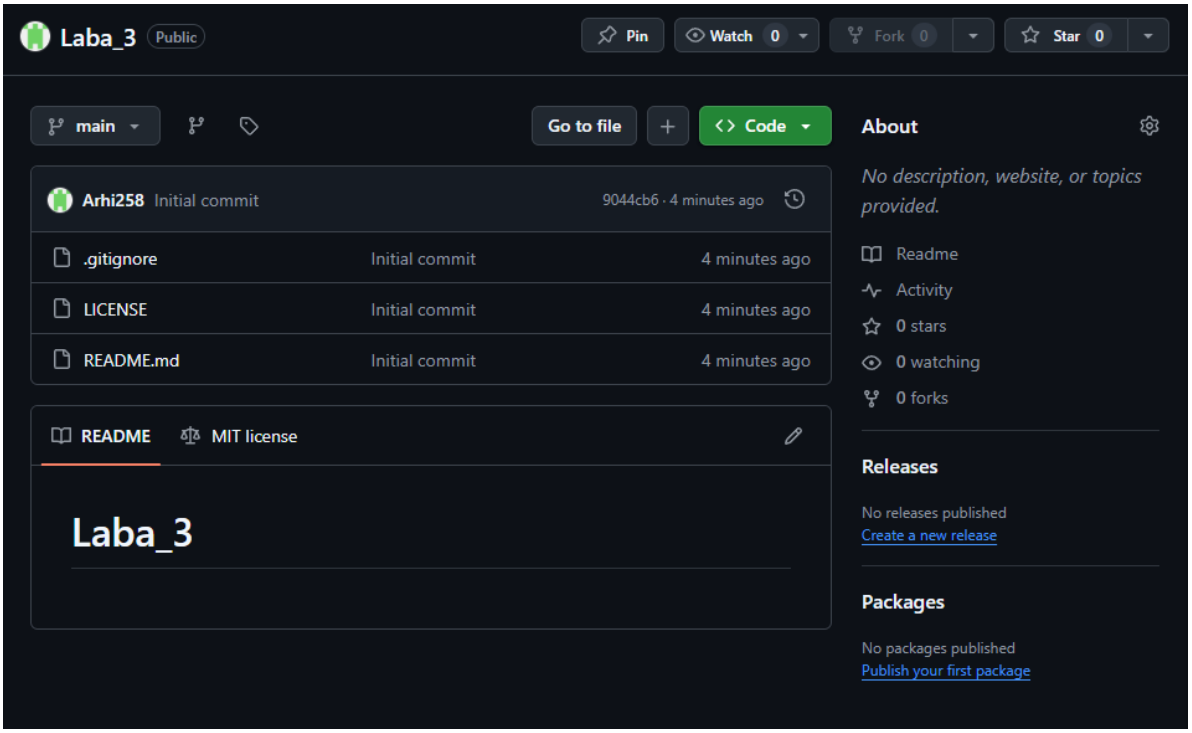


Рисунок 1. Общедоступный репозиторий

2. Было выполнено клонирование данного репозитория (рис 2).

```
arhi2@DESKTOP-25N997A MINGW64 /z/Репозиторий/Laba_3
$ cd Z:\Репозиторий

arhi2@DESKTOP-25N997A MINGW64 /z/Репозиторий
$ git clone https://github.com/Arhi258/Laba_3.git
Cloning into 'Laba_3'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (5/5), done.
```

Рисунок 2. Клонирование репозитория

3. Был дополнен файл .gitignore и изменен файл README.

4. Был создан проект PyCharm в папке репозитория.

5. Были проработаны примеры лабораторной работы. Для каждого примера был создан отдельный модуль языка Python. Изменения были зафиксированы в репозитории.

6. Были построены UML-диаграммы деятельности для примера 4 (рис 3) и примера 5 (рис 4).

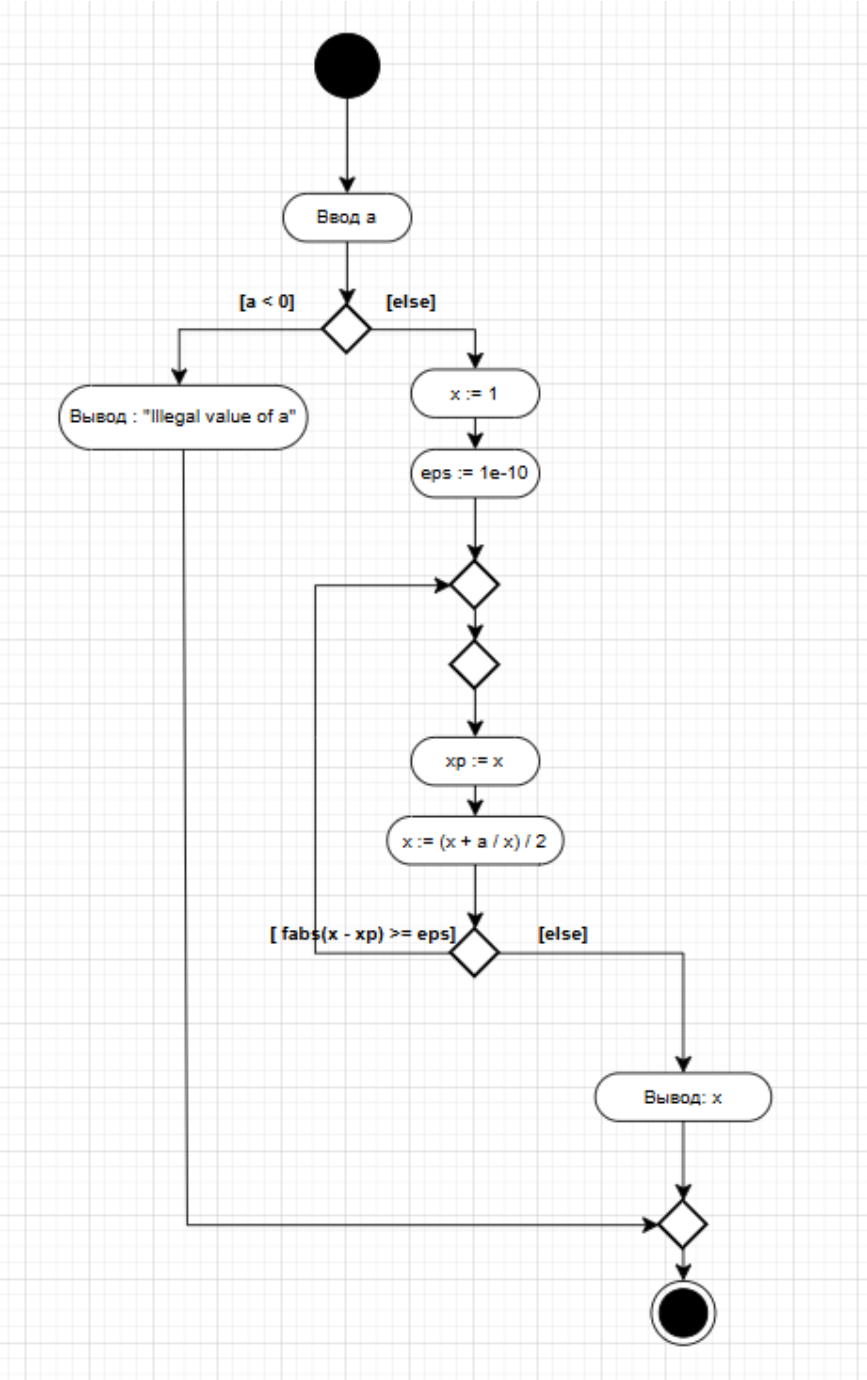


Рисунок 3. UML-диаграмма деятельности

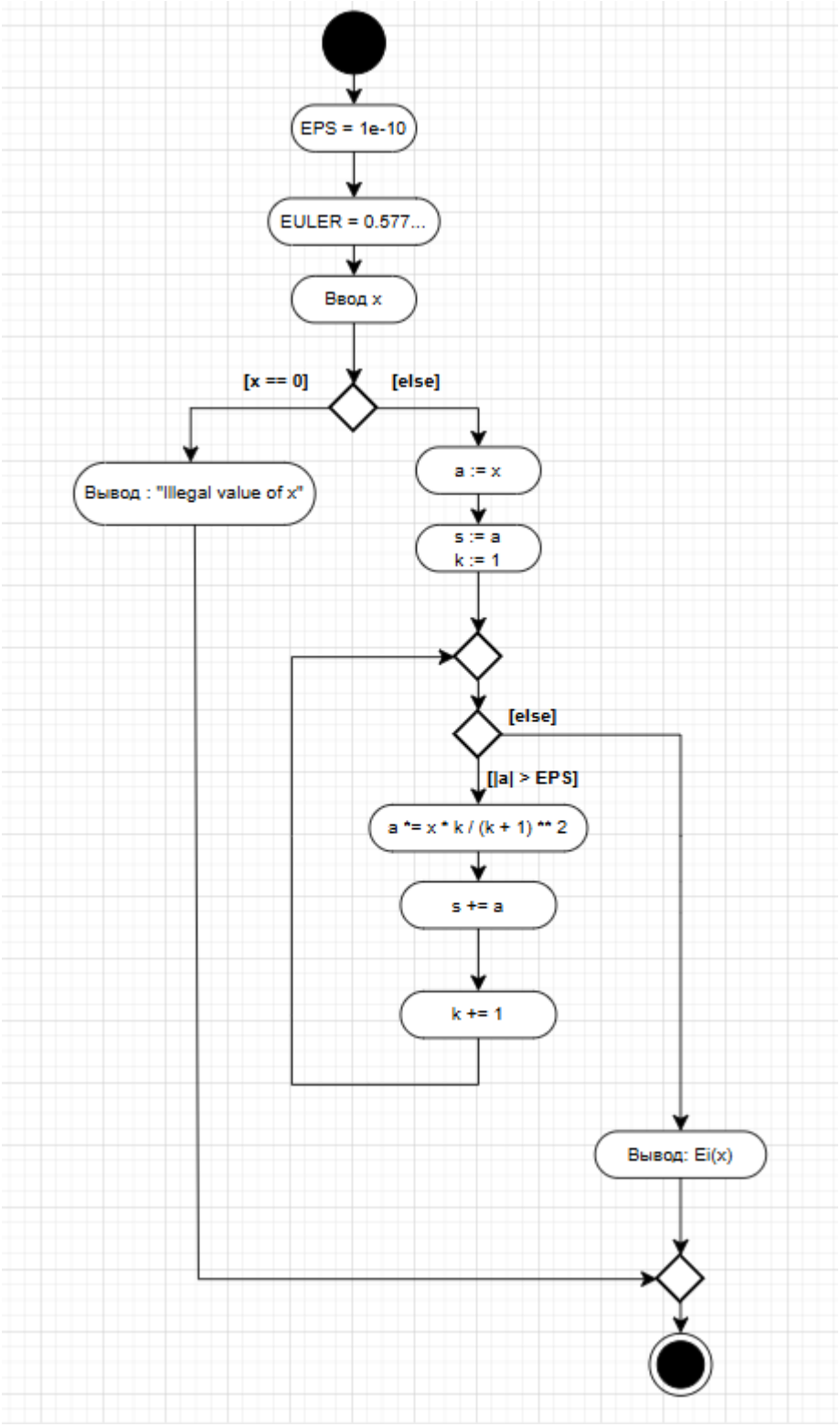


Рисунок 4. UML-диаграмма деятельности

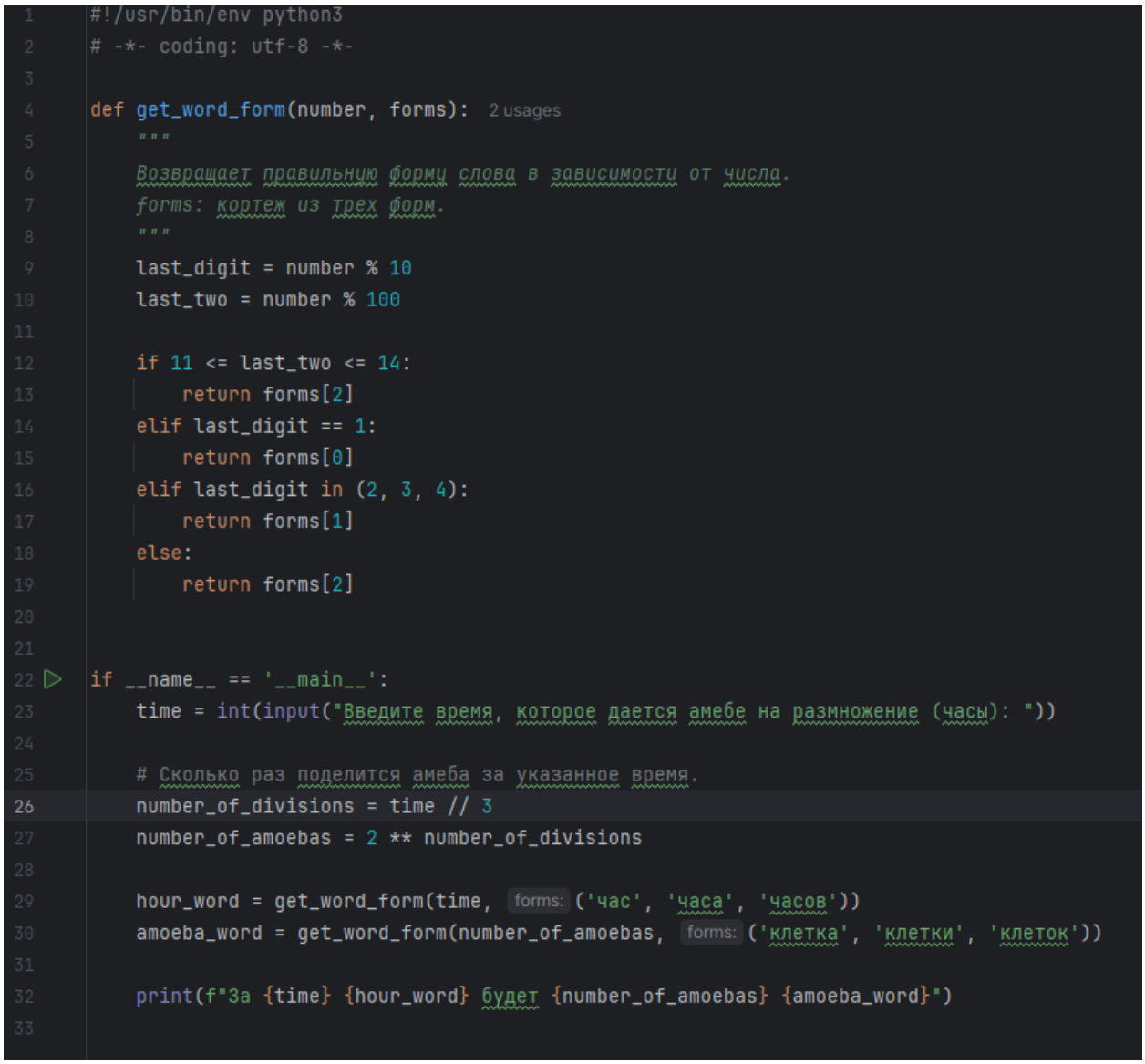
7. Были выполнены индивидуальные задания: задание 1 (рис 5), задание 2 (рис 6), задание 3 (рис 7), задание повышенной сложности (рис 8). Были сделаны UML-диаграммы деятельности для задания 1 (рис 9), задания 2 (рис 10), задания 3 (рис 11), задания повышенной сложности (рис 12).

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

import sys

if __name__ == '__main__':
    season_number = int(input("Введите номер времени года (1-4): "))

    if season_number == 1:
        print("Декабрь, Январь, Февраль")
    elif season_number == 2:
        print("Март, Апрель, Май")
    elif season_number == 3:
        print("Июнь, Июль, Август")
    elif season_number == 4:
        print("Сентябрь, Октябрь, Ноябрь")
    else:
        print("Неверно введённое число!", file=sys.stderr)
        sys.exit(1)
```

Рисунок 5. Задание 1

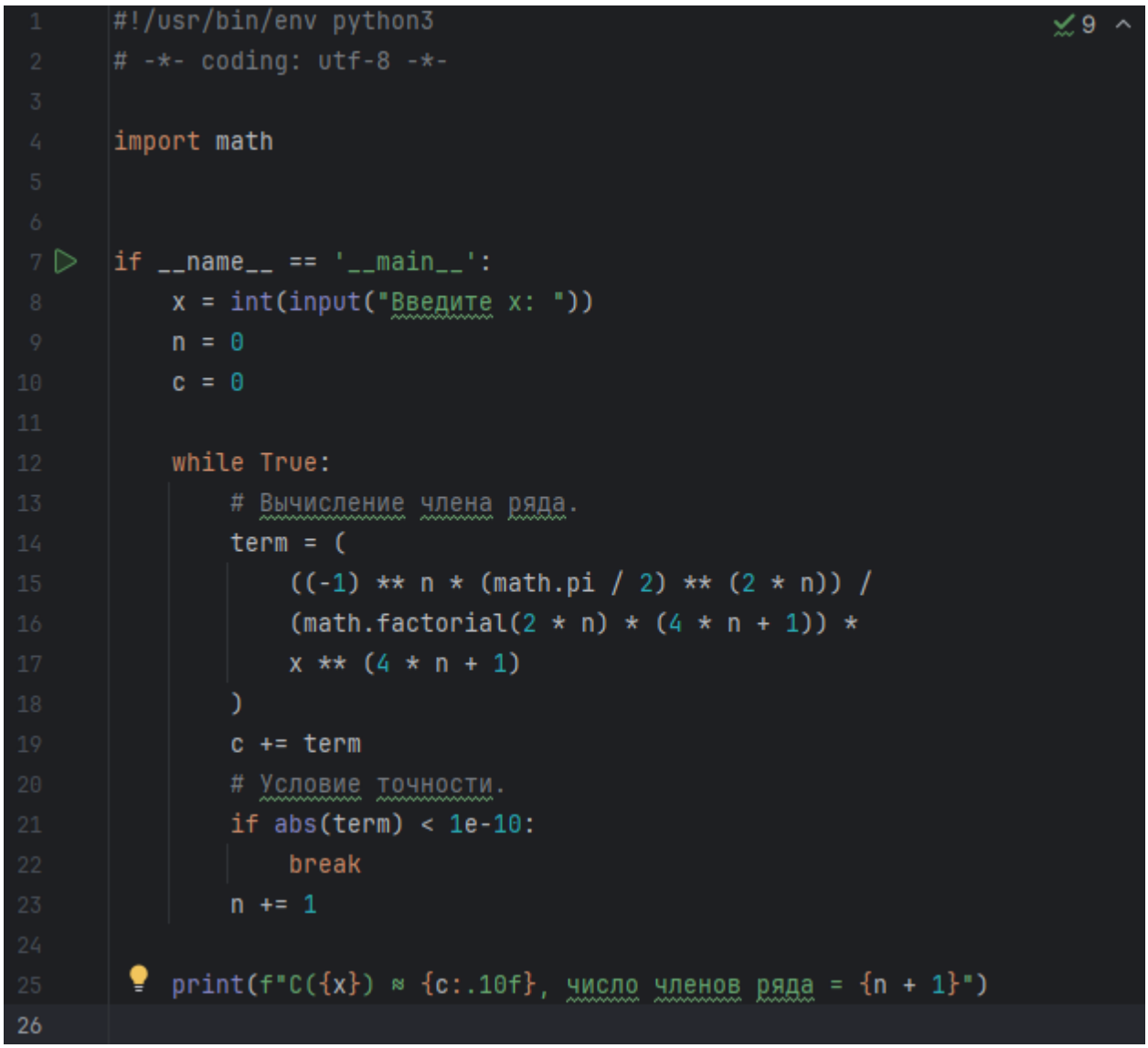
```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import math


if __name__ == '__main__':
    a = float(input("Введите координату a = "))
    b = float(input("Введите координату b = "))
    r = math.sqrt(a ** 2 + b ** 2)
    if 0.5 <= r <= 1:
        print(f"Точка A({a}, {b}) принадлежит кольцу")
    else:
        print(f"Точка A({a}, {b}) не принадлежит кольцу")
```

Рисунок 6. Задание 2



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

def get_word_form(number, forms):  2 usages
    """
    Возвращает правильную форму слова в зависимости от числа.
    forms: кортеж из трех форм.
    """
    last_digit = number % 10
    last_two = number % 100

    if 11 <= last_two <= 14:
        return forms[2]
    elif last_digit == 1:
        return forms[0]
    elif last_digit in (2, 3, 4):
        return forms[1]
    else:
        return forms[2]


if __name__ == '__main__':
    time = int(input("Введите время, которое дается амебе на размножение (часы): "))

    # Сколько раз поделится амеба за указанное время.
    number_of_divisions = time // 3
    number_of_amoebas = 2 ** number_of_divisions

    hour_word = get_word_form(time, forms: ('час', 'часа', 'часов'))
    amoeba_word = get_word_form(number_of_amoebas, forms: ('клетка', 'клетки', 'клеток'))

    print(f"За {time} {hour_word} будет {number_of_amoebas} {amoeba_word}")

```

Рисунок 7. Задание 3



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

import math


if __name__ == '__main__':
    x = int(input("Введите x: "))
    n = 0
    c = 0

    while True:
        # Вычисление члена ряда.
        term = (
            ((-1) ** n * (math.pi / 2) ** (2 * n)) /
            (math.factorial(2 * n) * (4 * n + 1)) *
            x ** (4 * n + 1)
        )
        c += term
        # Условие точности.
        if abs(term) < 1e-10:
            break
        n += 1

    print(f"C({x}) ≈ {c:.10f}, число членов ряда = {n + 1}")
```

Рисунок 8. Задание повышенной сложности

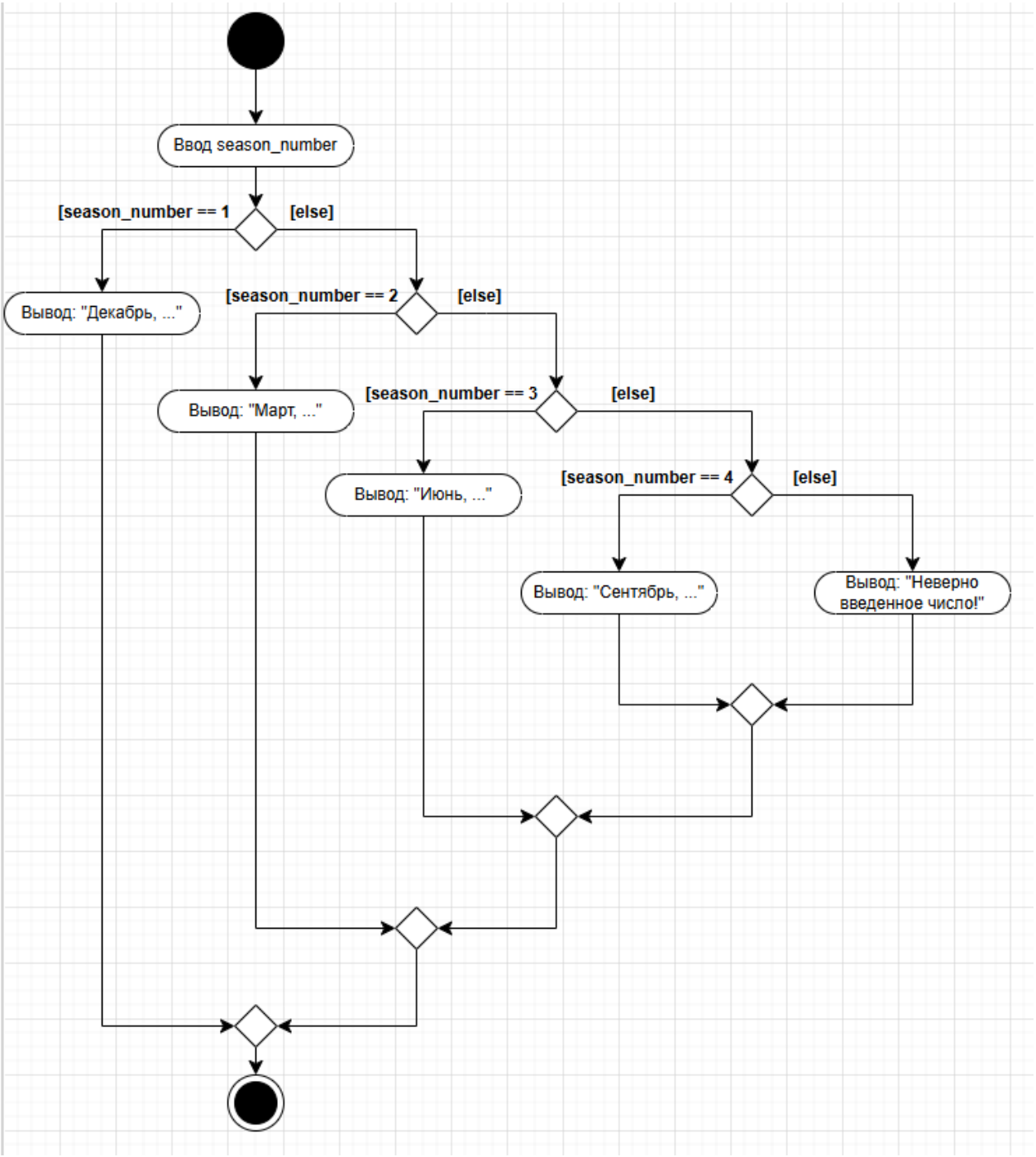


Рисунок 9. UML-диаграмма деятельности для задания 1

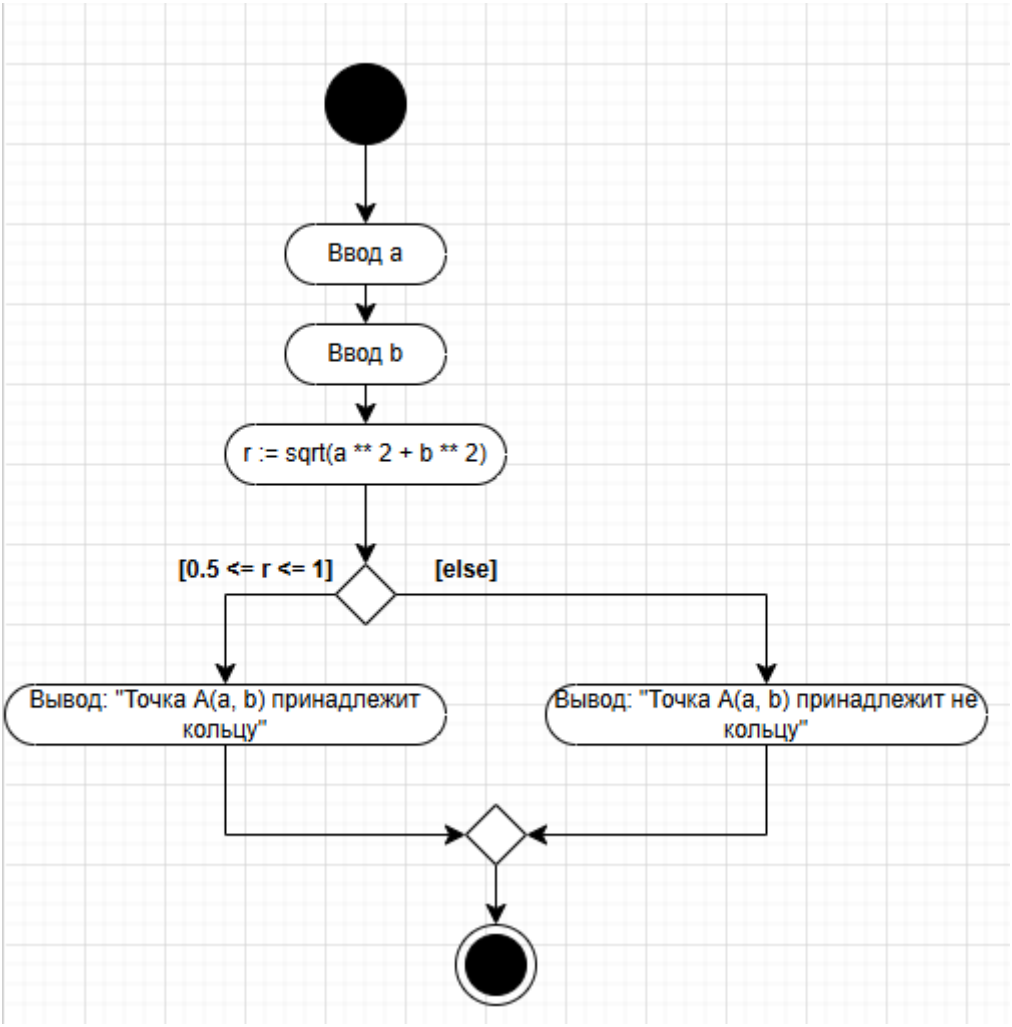


Рисунок 10. UML-диаграмма деятельности для задания 2

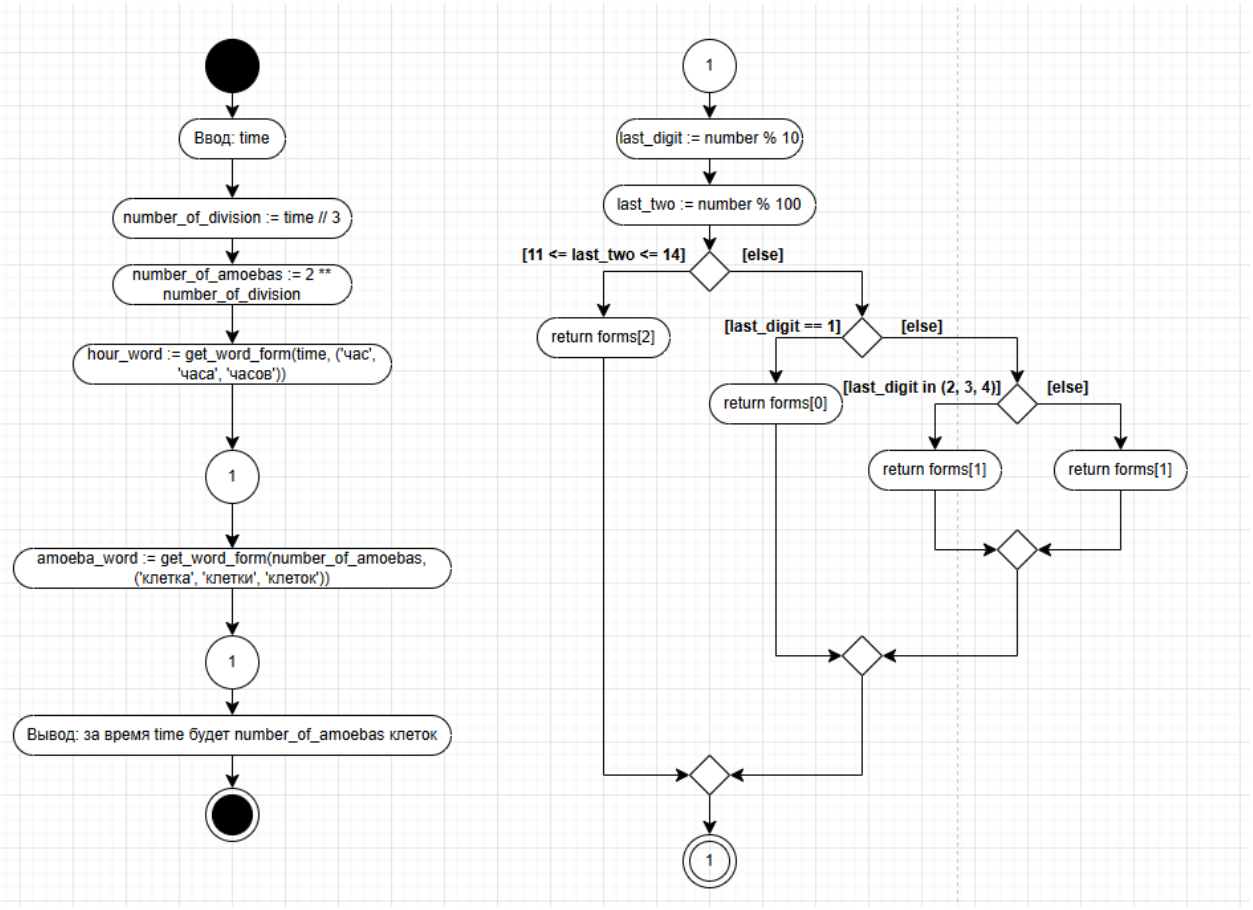


Рисунок 11. UML-диаграмма деятельности для задания 3

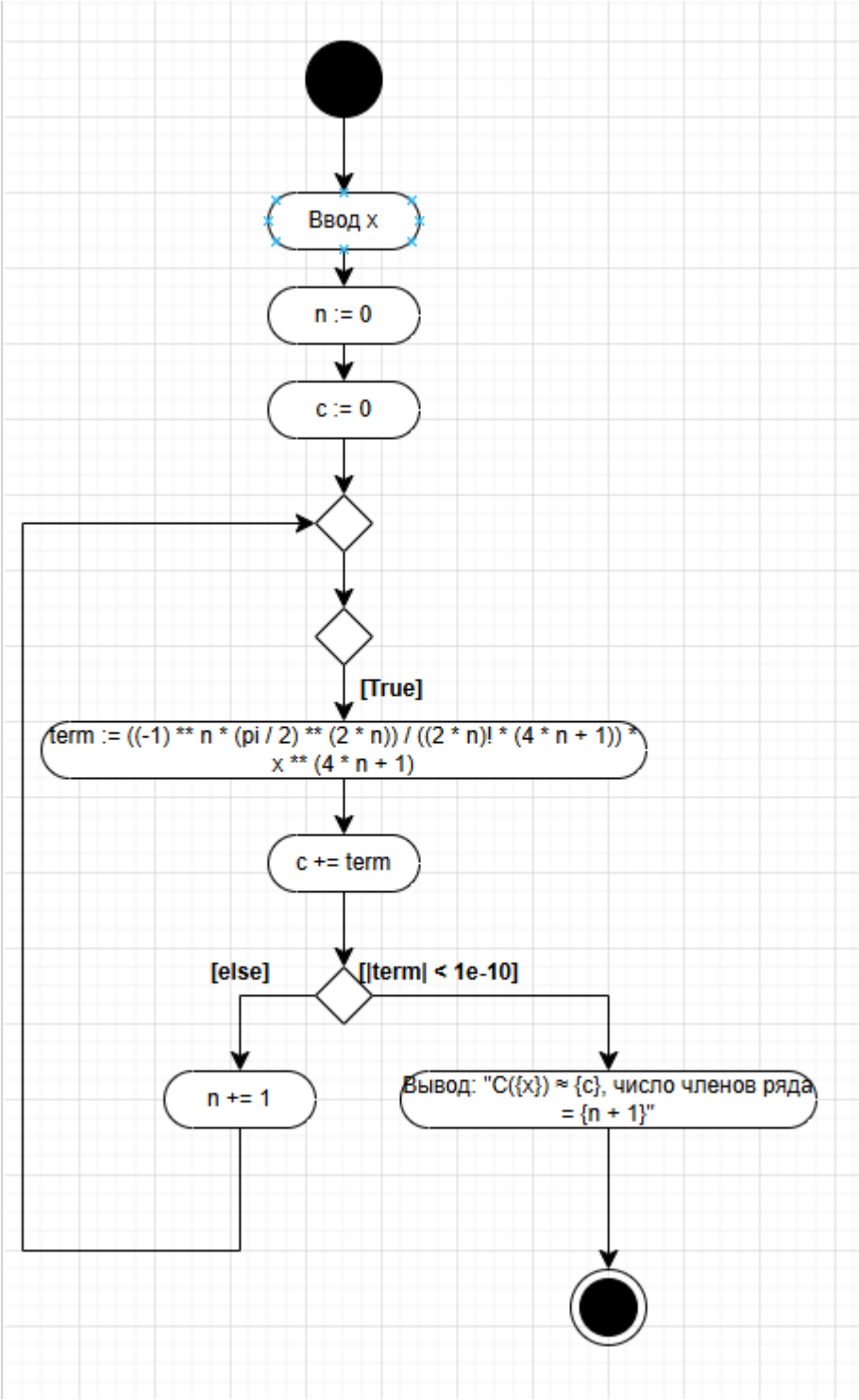


Рисунок 12. UML-диаграмма деятельности для задания повышенной сложности

8. Изменения были зафиксированы в репозитории (рис 13).

```
arhi2@DESKTOP-25N997A MINGW64 /z/Репозиторий/Laba_3 (main)
$ git commit -m "Выполнены индивидуальные задания"
[main 2501061] Выполнены индивидуальные задания
 4 files changed, 90 insertions(+)
 create mode 100644 Tasks/Difficult_task.py
 create mode 100644 Tasks/Task 1.py
 create mode 100644 Tasks/Task 2.py
 create mode 100644 Tasks/Task 3.py

arhi2@DESKTOP-25N997A MINGW64 /z/Репозиторий/Laba_3 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

arhi2@DESKTOP-25N997A MINGW64 /z/Репозиторий/Laba_3 (main)
$ git push origin main
Enumerating objects: 30, done.
Counting objects: 100% (30/30), done.
Delta compression using up to 16 threads
Compressing objects: 100% (25/25), done.
Writing objects: 100% (27/27), 6.91 KiB | 1.73 MiB/s, done.
Total 27 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/Arhi258/Laba_3.git
   9044cb6..2501061  main -> main
```

Рисунок 13. Фиксация изменений в репозитории

9. Был добавлен отчет в папку doc, изменения в репозитории были зафиксированы. Сделанные изменения были отправлены на сервер GitHub (рис 14).

```
arhi2@DESKTOP-25N997A MINGW64 /z/Репозиторий/Laba_3 (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 750.28 KiB | 37.51 MiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Arhi258/Laba_3.git
   0d48c46..db719aa  main -> main

arhi2@DESKTOP-25N997A MINGW64 /z/Репозиторий/Laba_3 (main)
$
```

Рисунок 14. Отправка изменений на сервер

Контрольные вопросы.

1. Диаграммы деятельности UML нужны для наглядного описания потока работ и алгоритмов: что происходит, в каком порядке, при каких условиях и что выполняется параллельно. Ими удобно фиксировать бизнес-процессы и поведение системы на уровне сценариев.

2. Состояние действия – выполнение одного атомарного шага. Состояние деятельности – выполнение длительной или составной деятельности.

3. В UML-диаграммах деятельности переходы – это ребра (flows), а ветвления – специальные узлы (decision/merge и fork/join) с охранными условиями.

4. Алгоритм разветвляющейся структуры – это алгоритм, в котором выполнение дальнейших шагов выбирается в зависимости от условия исполняется одна из альтернативных ветвей.

5. Линейный алгоритм – последовательность шагов без выбора. Разветвляющийся алгоритм – содержит проверку условий и альтернативные ветви исполнения.

6. Условный оператор – конструкция, которая выбирает, какой блок действий выполнить, в зависимости от истинности логического условия. Основные формы: неполная форма, полная форма, множественный выбор, тернарный условный оператор.

7. Операторы сравнения в Python: равенство (==), не равно (!=), больше (>), меньше (<), меньше или равно (<=), больше или равно (>=).

8. Простым условием в Python называется условие в операторе if, которое проверят истинность или ложность выражения. Пример: if x >10; print (x).

9. Составное условие в Python – группа операторов, объединенных логическими операторами and, or и not. Также к составным условиям относятся конструкции if-elif-else. Пример: if x > 10 and y < 2; print (x).

10. При составлении сложных условий допускаются операторы and, or, not.

11. Оператор ветвления может содержать внутри себя другие ветвления.

12. Алгоритм циклической структуры – это такой алгоритм, в котором одна или несколько последовательностей действий повторяются многократно до выполнения определенного условия.

13. Типы циклов: цикл с предусловием (while), цикл с параметром (for).

14. Функция rangeиспользуется для генерации последовательностей целых чисел, часто применяется в циклах, особенно в циклах for.

15. Чтобы организовать перебор значений от15 до 0 с шагом 2 нужно написать: range(0, 15, 2).

16. Циклы могут быть вложенными.

17. Бесконечный цикл образуется тогда, когда условие его выполнения всегда остается истинным, и цикл не имеет механизма завершения. Для выхода из бесконечного цикла используют оператор break.

18. Оператор break используется для немедленной остановки цикла.

19. Оператор continue запускает цикл заново, при этом код, расположенный после данного оператора не выполняется.

20. stdout – стандартный поток вывода, используется для вывода данных. stderr – стандартный поток ошибок, предназначен для сообщений об ошибках и диагностической информации.

21. В Python вывод в стандартный поток ошибок можно организовать несколькими способами: через print() с параметром file, напрямую через sys.stderr.write(), через логирование.

22. Функция exit() используется в Python для завершения выполнения программы.

Вывод: в результате работы были приобретены навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Были освоены операторы языка Python версии 3.x if, while, for,

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