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

Кафедра инфокоммуникаций

Институт цифрового развития

ОТЧЁТ

по лабораторной работе №2.2

Дисциплина: «Основы кроссплатформенного программирования»

Тема: «Условные операторы и циклы в языке Python»

Выполнил: студент 1 курса

группы ИВТ-б-о-21-1

Горшков Виталий Игоревич

Ставрополь 2022

Выполнение работы.

1. Создал репозиторий в GitHub «rep 2.2» в который добавил .gitignore, который дополнил правила для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на лок. сервер и организовал в соответствие с моделью ветвления git-flow.

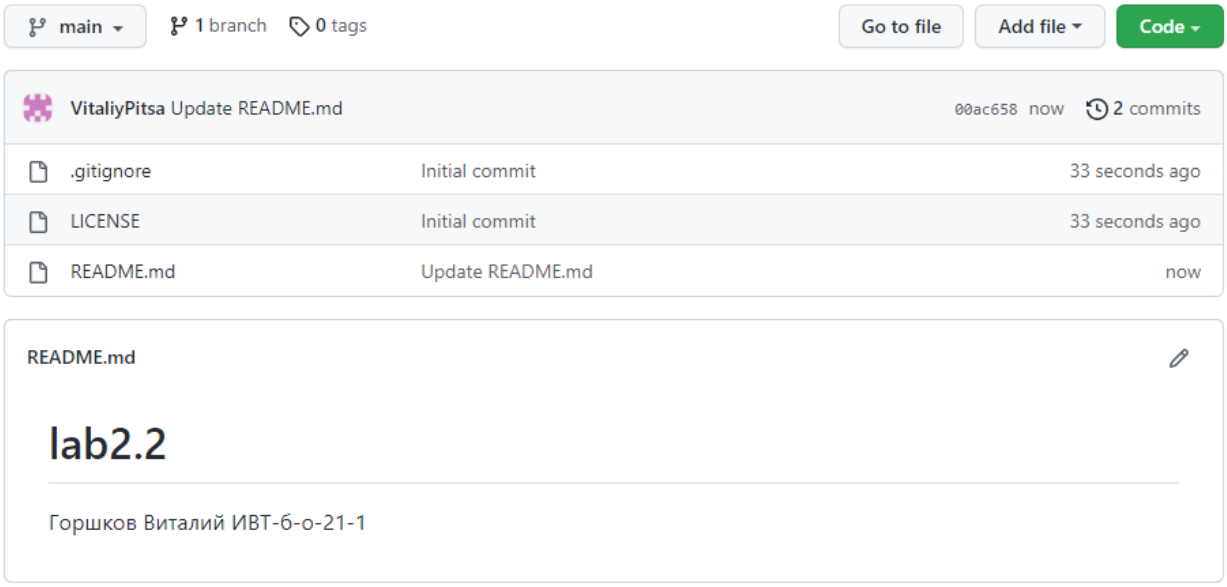


Рисунок 1 Создание репозитория

267 lines (213 sloc) | 5.26 KB

```
# Created by https://www.toptal.com/developers/gitignore/api/python,pycharm
# Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm
.idea/
### PyCharm ###
# Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio, WebStorm and Rider
# Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839

# User-specific stuff
.idea/**/workspace.xml
.idea/**/tasks.xml
.idea/**/usage.statistics.xml
.idea/**/dictionaries
.idea/**/shelf

# AWS User-specific
.idea/**/aws.xml

# Generated files
.idea/**/contentModel.xml

# Sensitive or high-churn files
.idea/**/dataSources/
.idea/**/dataSources.ids
.idea/**/dataSources.local.xml
.idea/**/sqlDataSources.xml
.idea/**/dynamic.xml
.idea/**/uiDesigner.xml
.idea/**/dbnavigator.xml

# Gradle
.idea/**/gradle.xml
.idea/**/libraries

# Gradle and Maven with auto-import
# When using Gradle or Maven with auto-import, you should exclude module files,
# since they will be recreated, and may cause churn.  Uncomment if using
# auto-import.
# .idea/artifacts
```

Рисунок 2 Добавление правил в .gitignore

```
C:\Users\adamk\OneDrive\Рабочий стол\ЛР 2.2>git clone https://github.com/AdamKh/rep_2.2.git
Cloning into 'rep_2.2'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), 4.25 KiB | 1.06 MiB/s, done.
Resolving deltas: 100% (1/1), done.

C:\Users\adamk\OneDrive\Рабочий стол\ЛР 2.2>git flow init
Initialized empty Git repository in C:/Users/adamk/OneDrive/Рабочий стол/ЛР 2.2/.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:/Users/adamk/OneDrive/Рабочий стол/ЛР 2.2/.git/hooks]
```

Рисунок 3 Клонирование и организация репозитория согласно модели ветвления git-flow

2. Создал проект PyCharm в папке репозитория, проработал примеры ЛР.

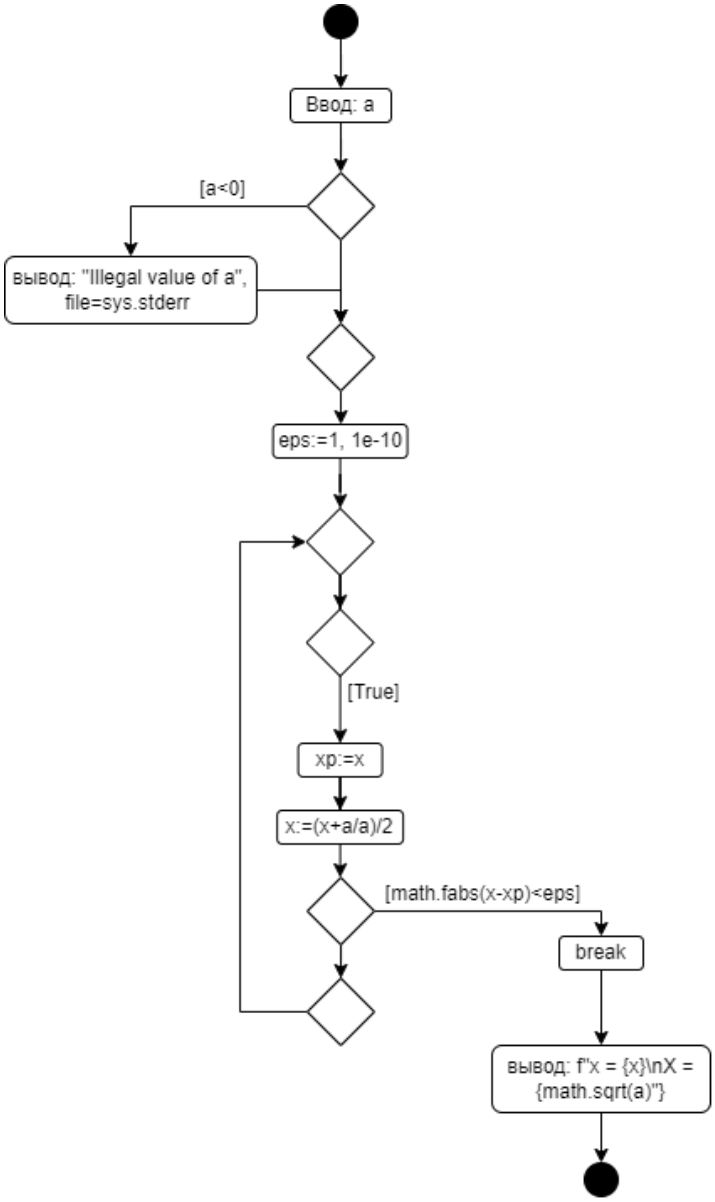


Рисунок 4 UML-диаграмма программы 4 примера

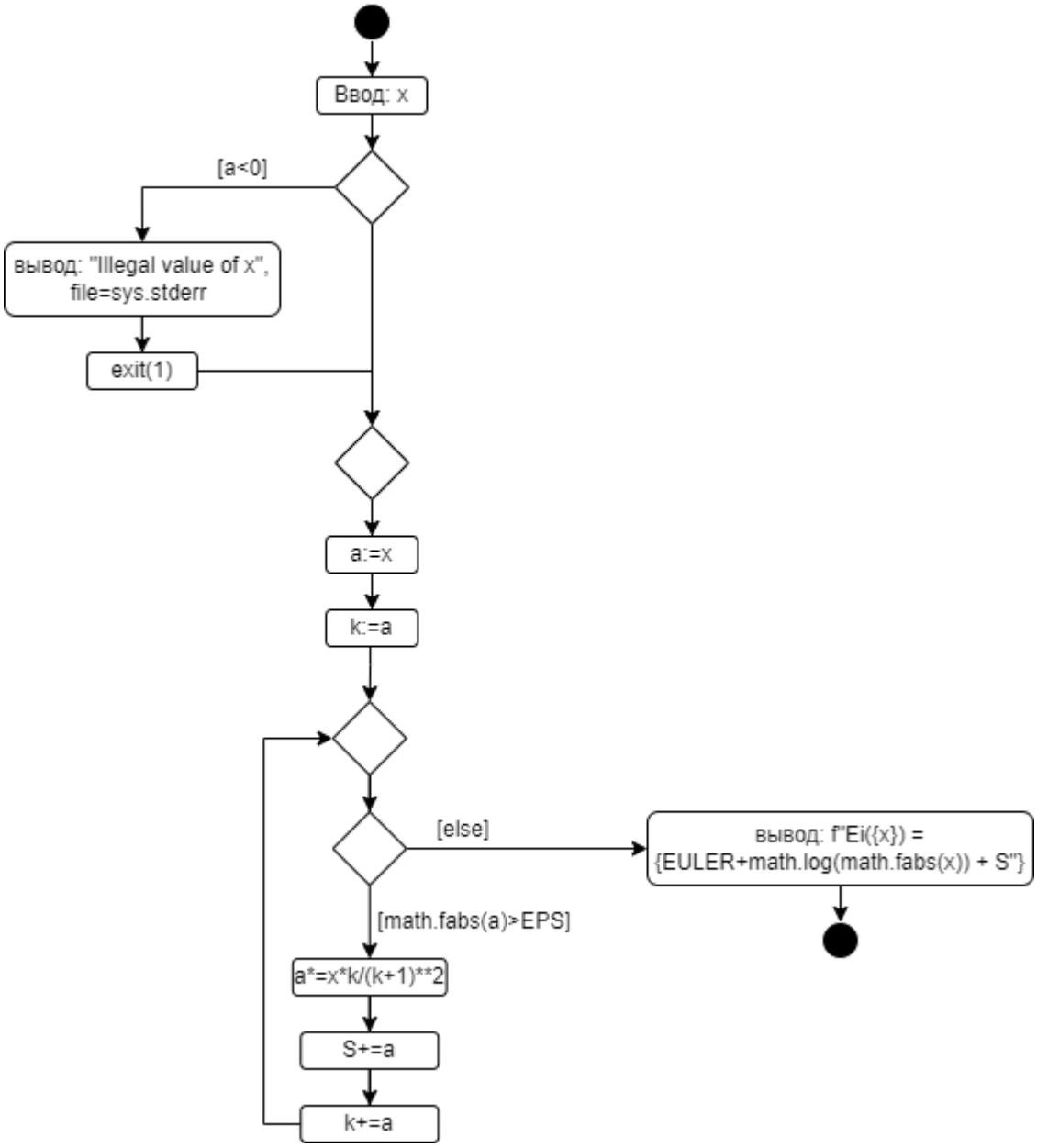


Рисунок 5 UML-диаграмма программы 5 примера

3. Выполнил индивидуальные задания и задание повышенной сложности согласно своему варианту. Построил UML диаграммы программ.

Рисунок 6 UML – диаграмма к программе инд. задания 1

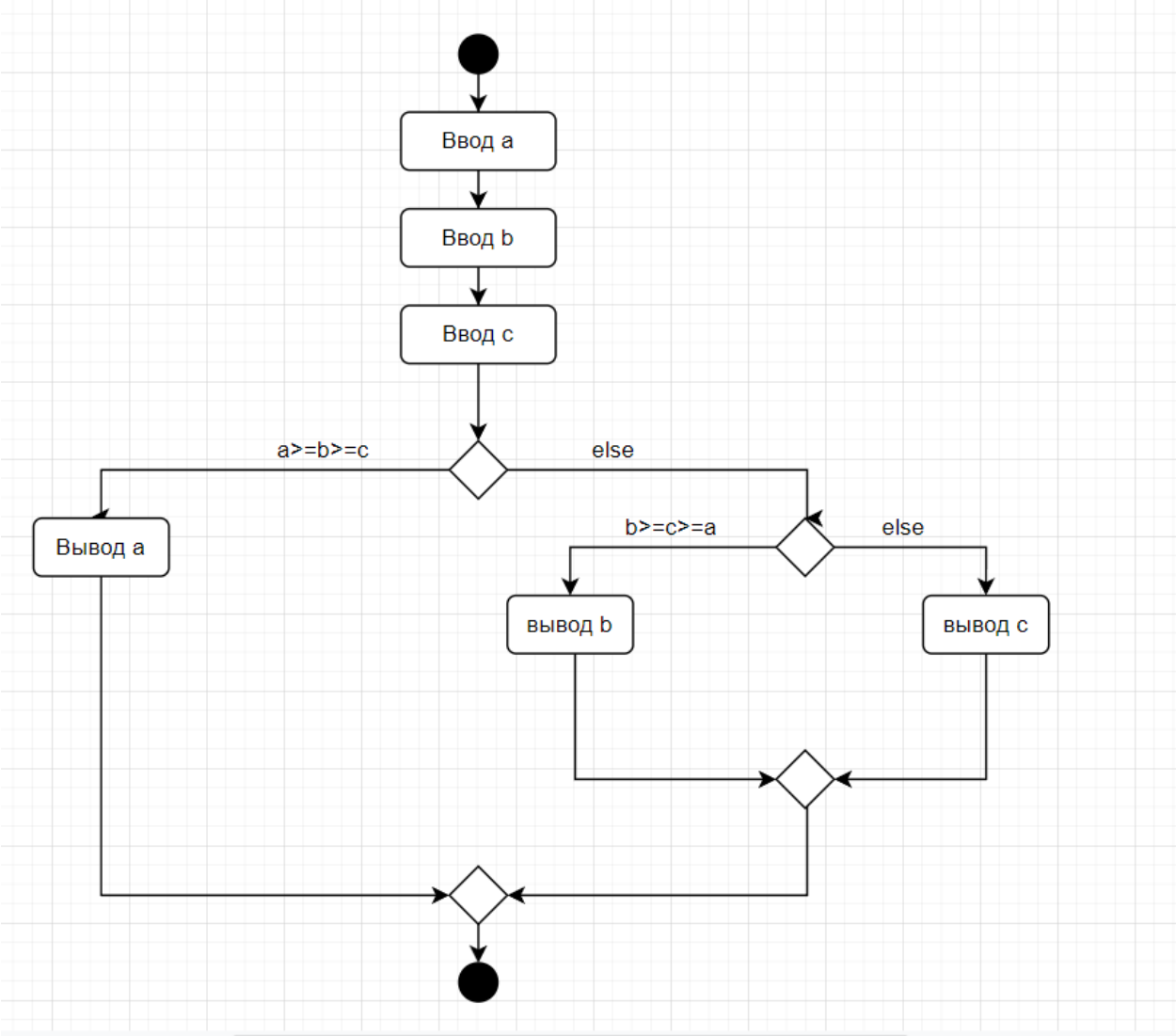


Рисунок 7 UML – диаграмма к программе инд. задания 2

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

if __name__ == '__main__':
    n = int(input("Введите номер месяца: "))

    if n == 1:
        print("первое полугодие, 31 день")
    elif n == 2:
        print("первое полугодие, 28 дней")
    elif n == 3:
        print("первое полугодие, 31 день")
    elif n == 4:
        print("первое полугодие, 30 дней")
    elif n == 5:
        print("первое полугодие, 31 день")
    elif n == 6:
        print("первое полугодие, 30 дней")
    elif n == 7:
        print("второе полугодие, 31 день")
    elif n == 8:
        print("второе полугодие, 30 дней")
    elif n == 9:
        print("второе полугодие, 31 день")
    elif n == 10:
        print("второе полугодие, 30 дней")
    elif n == 11:
        print("второе полугодие, 31 день")
    elif n == 12:
        print("второе полугодие, 30 дней")
    else:
        print("Ошибка!", file=sys.stderr)
        exit(1)
```

Рисунок 8 Программа к инд. заданию №1

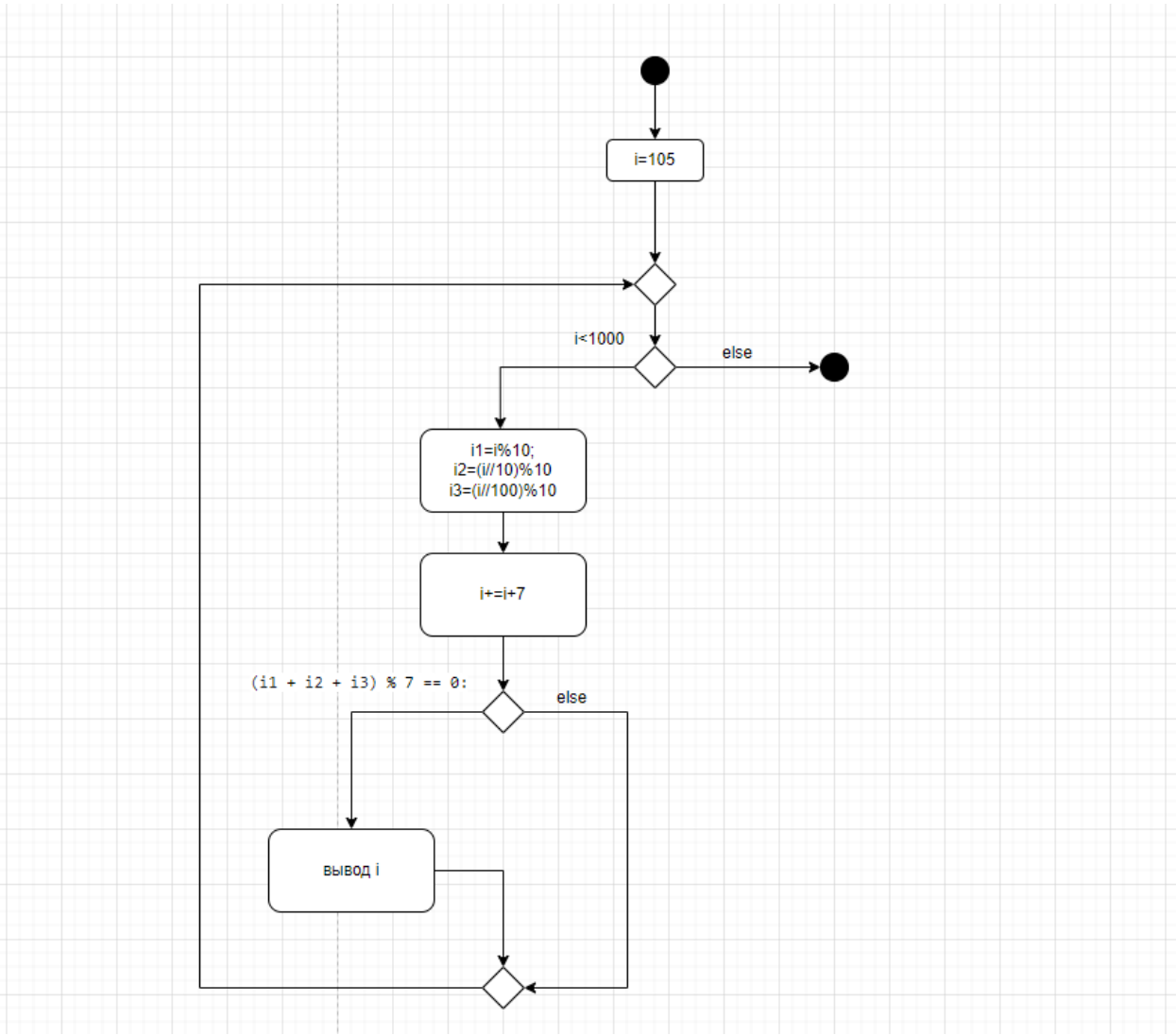
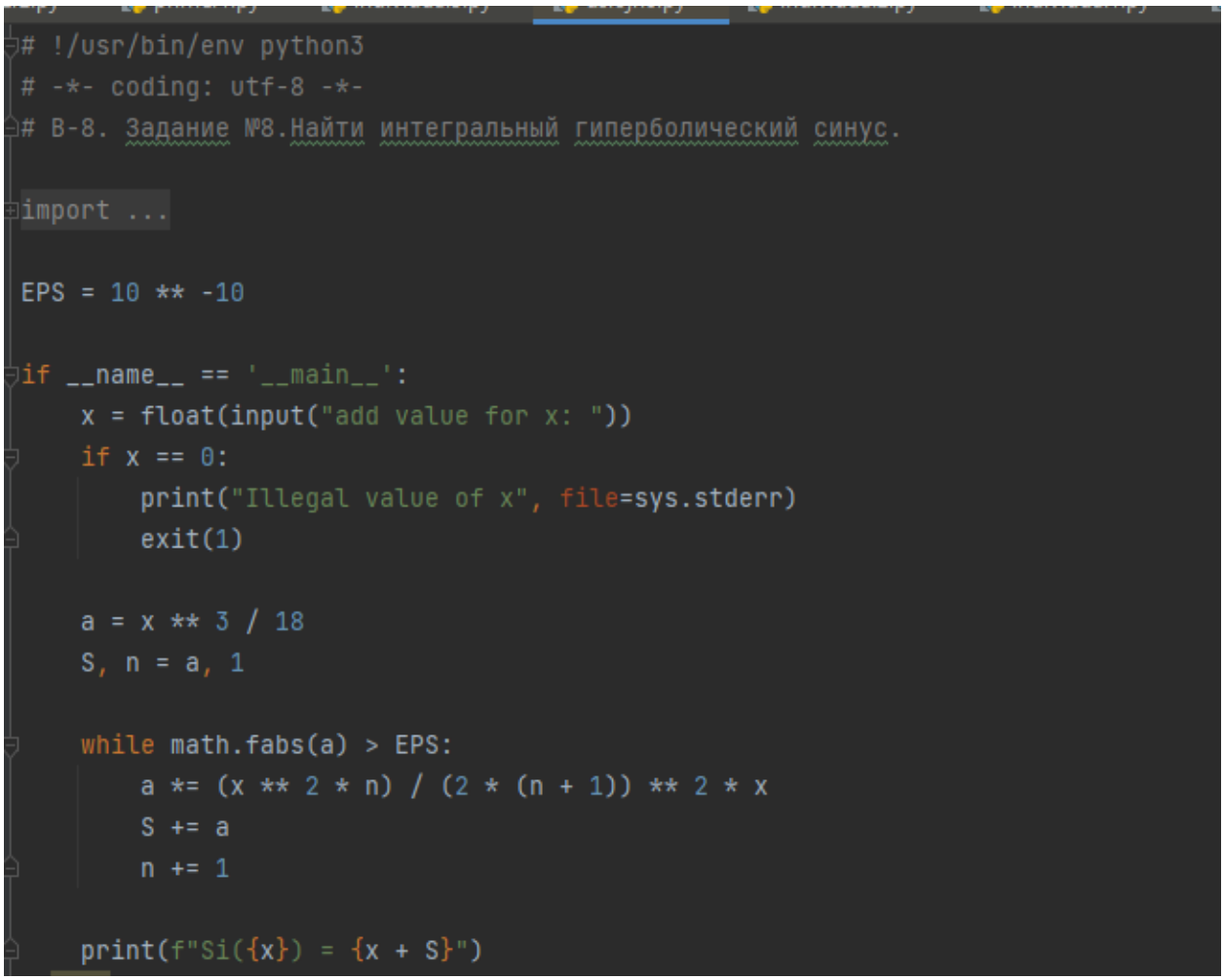


Рисунок 3.6 UML – диаграмма к программе инд. задания 3



```python
# !/usr/bin/env python3
# -*- coding: utf-8 -*-
# B-8. Задание №8.Найти интегральный гиперболический синус.

import ...

EPS = 10 ** -10

if __name__ == '__main__':
    x = float(input("add value for x: "))
    if x == 0:
        print("Illegal value of x", file=sys.stderr)
        exit(1)

    a = x ** 3 / 18
    S, n = a, 1

    while math.fabs(a) > EPS:
        a *= (x ** 2 * n) / (2 * (n + 1)) ** 2 * x
        S += a
        n += 1

    print(f"Si({x}) = {x + S}")
```

Рисунок 3.7 Программа для задачи повышенной сложности.

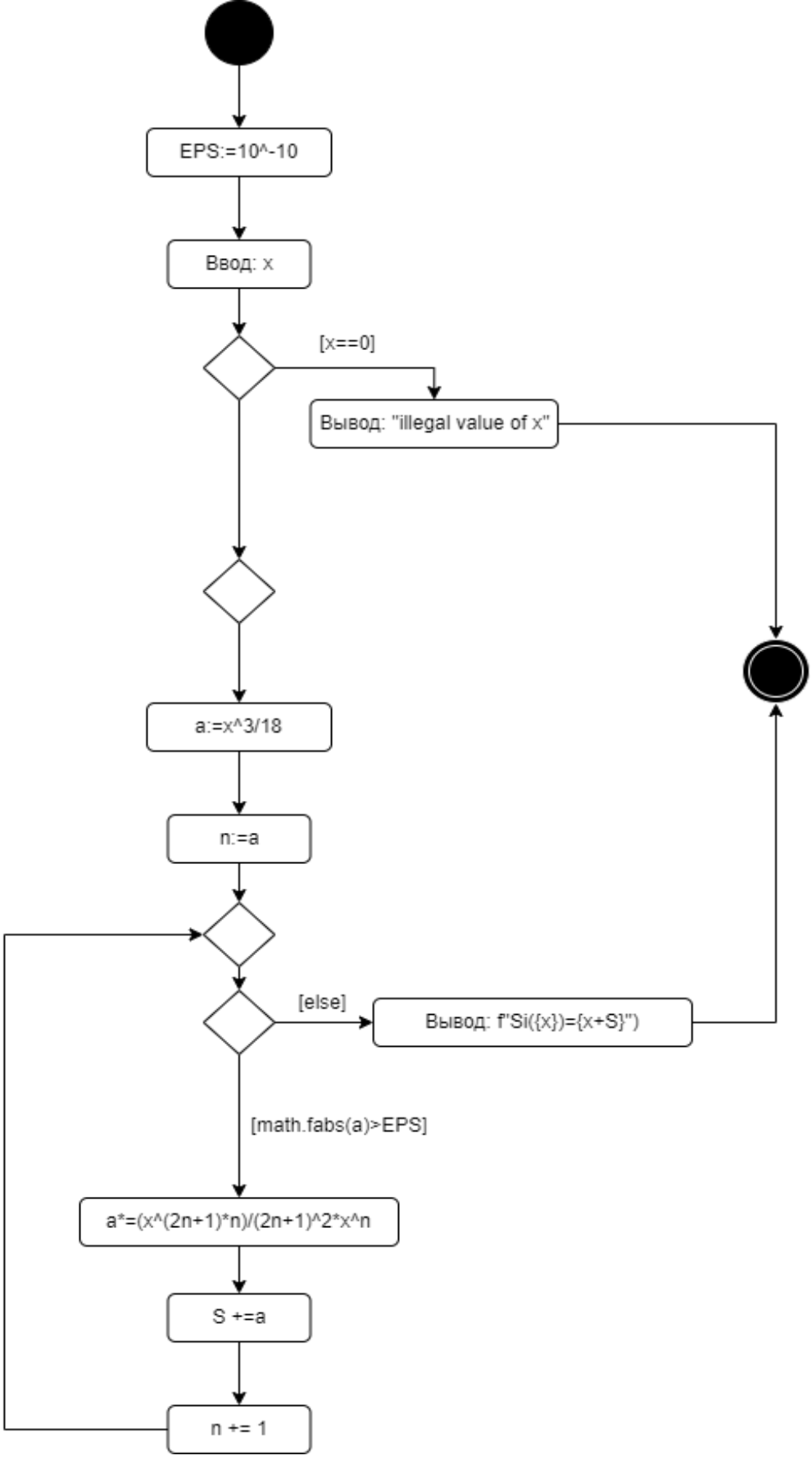


Рисунок 3.8 UML – диаграмма деятельности программы для усложненного задания

4. Сделал коммит, выполнил слияние с веткой main, и запушил изменения в уд. репозиторий.

Рисунок 4.2 Изменения на уд. сервере

1. Для чего нужны диаграммы деятельности UML?

Позволяет наглядно визуализировать алгоритм программы.

2. Что такое состояние действия и состояние деятельности?

Состояние действия - частный вид состояния деятельности, а конкретнее – такое состояние, которое не может быть подвергнуто дальнейшей декомпозиции.

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?

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: a == b

9. Что такое составное условие? Приведите примеры.

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

Пример: (a == b or a == c)

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 — это вид алгоритма, в процессе выполнения которого одно или несколько действий нужно повторить.

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.

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

range(15, 0, 2)

16. Могул 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?

Используется для выхода из цикла.

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

Оператор continue используется только в циклах. В операторах for , while , do while , оператор continue выполняет пропуск оставшейся части кода тела цикла и переходит к следующей итерации цикла.

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.