

**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

Харченко Богдан Романович

Ставрополь 2022

Ход работы:

1. Создал репозиторий в GitHub «Lab_4» 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.

Owner * **Repository name ***

Konstellation / Lab_4 ✓

Great repository names are short and snappy. Your new repository will be created as Lab_4-. Learn more about [furry-chainsaw?](#)

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 ▼

ⓘ You are creating a public repository in your personal account.

Create repository

Рисунок 1.1 Создание репозитория

```
C:\>git clone https://github.com/Konstellation/Lab_4.git
Cloning into 'Lab_4'...
remote: Enumerating objects: 33, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (30/30), done.
remote: Total 33 (delta 3), reused 29 (delta 2), pack-reused 0
Receiving objects: 100% (33/33), 1.10 MiB | 412.00 KiB/s, done.
Resolving deltas: 100% (3/3), done.
```

Рисунок 1.2 Клонирование репозитория

```
C:\LB_4>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
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:/LB_4/.git/hooks]
```

Рисунок 1.3 Организация репозитория в соответствии с моделью ветвления
git-flow



```
.gitignore - Блокнот
Файл  Правка  Формат  Вид  Справка

.idea/
# Created by https://www.toptal.com/developers/gitignore/api/python,pycharm
# Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm

### PyCharm ###
# Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Andro:
# 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
```

Рисунок 1.4 Изменение .gitignore

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

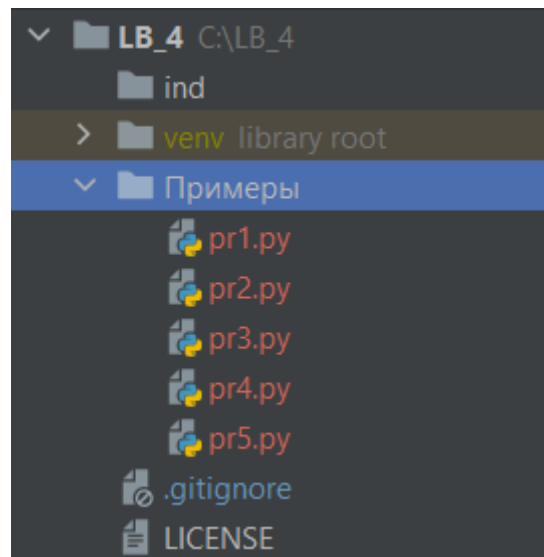


Рисунок 2.1 Примеры в проекте

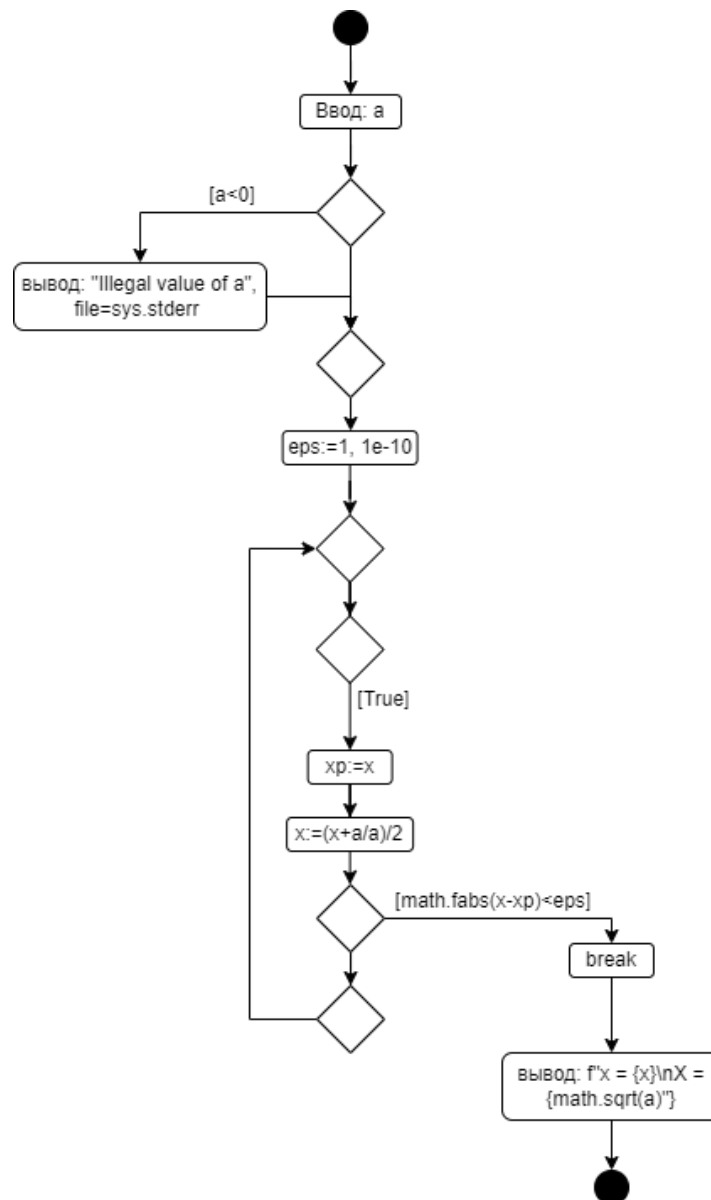


Рисунок 2.2 UML-диаграмма программы 4 примера

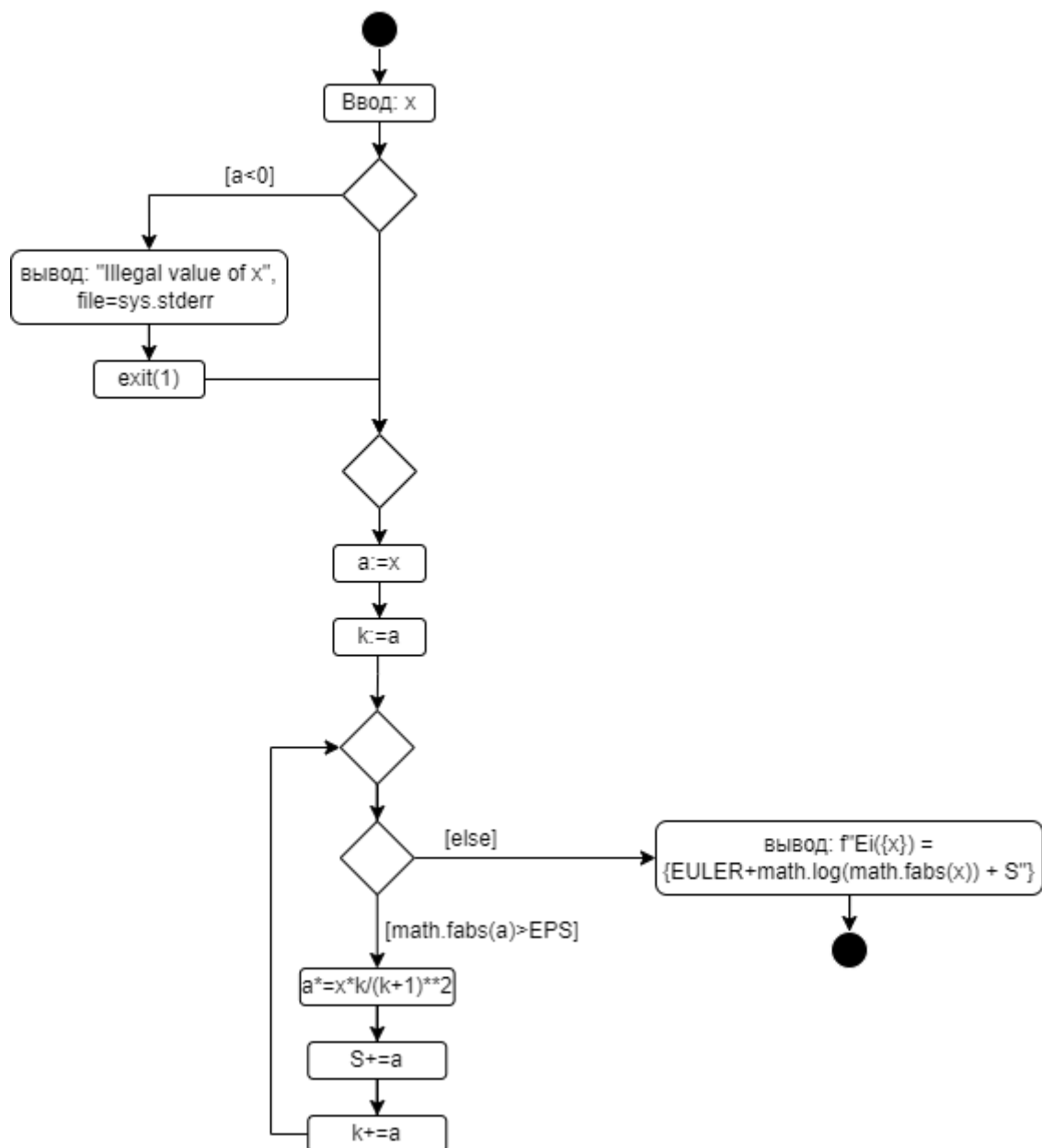


Рисунок 2.3 UML-диаграмма программы 5 примера

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

```

1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == '__main__':
5     # Ввод номера месяца
6     m = int(input('Введите номер месяца: '))
7
8     # Проверка на полугодие
9     if m <= 6:
10         print('Месяц приходится на первое полугодие.\n')
11     else:
12         print('Месяц приходится на второе полугодие.\n')
13
14     # Проверка на количество дней в месяце
15     if m == 2:
16         print('В месяце под номером ', m, ' 28 дней.')
17     elif m == 1 or m == 3 or m == 5 or m == 7 or m == 8 or m == 10 or m == 12:
18         print('В месяце под номером ', m, ' 31 дней.')
19     else:
20         print('В месяце под номером ', m, ' 30 дней.')
21

```

```

Введите номер месяца: 5
Месяц приходится на первое полугодие.

В месяце под номером 5 31 дней.

```

Рисунок 3.1 Программа к инд. заданию №1

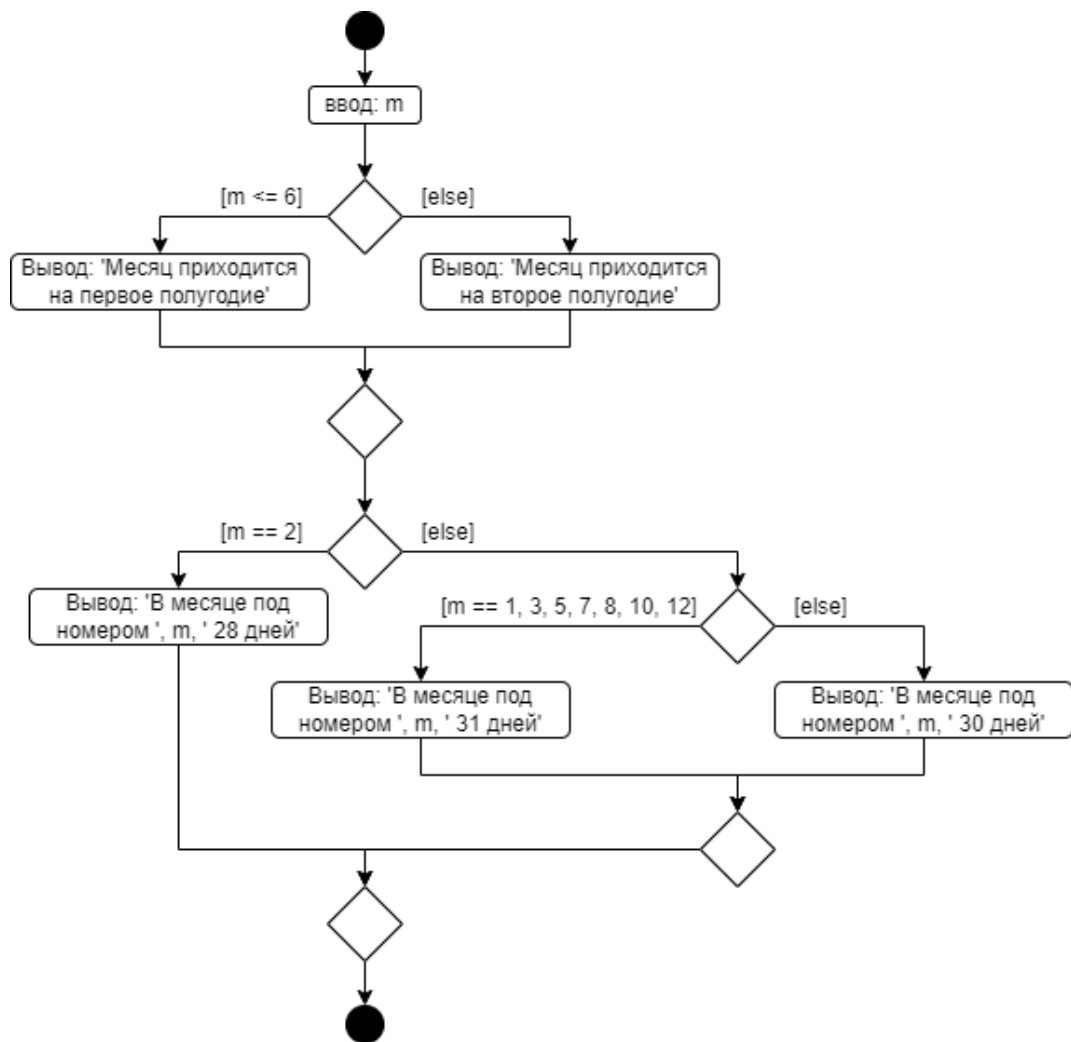


Рисунок 3.2 UML – диаграмма к программе инд. задания 1

```

1  ▶  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  ▶  if __name__ == '__main__':
5      print('Введите координаты точки M1')
6      x1 = int(input('x2 = '))
7      y1 = int(input('y2 = '))
8
9      print('Введите координаты точки M2')
10     x2 = int(input('x2 = '))
11     y2 = int(input('y2 = '))
12
13     if x1 == x2:
14         if y1 + y2 == 0:
15             print('Точки M1 и M2 симметричны относительно начала координат')
16         elif y1 == y2:
17             if x1 + x2 == 0:
18                 print('Точки M1 и M2 симметричны относительно начала координат')
19         else:
20             print('Точки не симметричны относительно начала координат')
21
  
```



```

Введите координаты точки M1
x2 = 2
y2 = 3
Введите координаты точки M2
x2 = -2
y2 = 3
Точки M1 и M2 симметричны относительно начала координат

```

Рисунок 3.3 Программа к инд. заданию №2

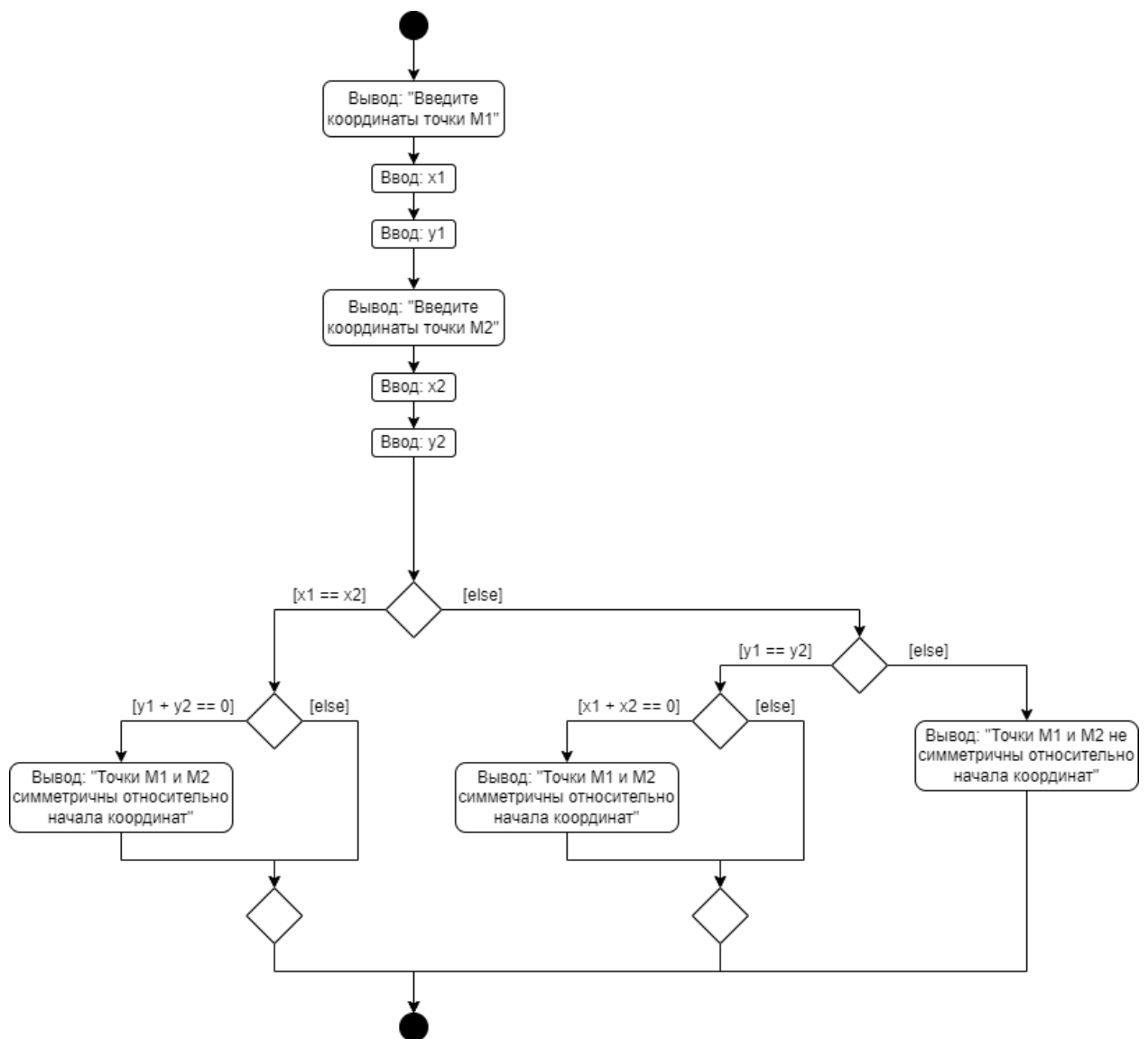


Рисунок 3.4 UML – диаграмма к программе инд. задания 2

```

1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == '__main__':
5     A = int(input('Введите стипендию: '))
6     B = int(input('Введите расходы: '))
7     s = B
8
9     for i in range(10):
10         B = (B*103) / 100
11         s += B
12     print(s)
13     c = s - A*10
14     print('Чтобы прожить учебный год, у родителей надо попросить ', c, ' р.')
15

```

```

Введите стипендию: 2000
Введите расходы: 10000
128077.95690814851
Чтобы прожить учебный год, у родителей надо попросить 108077.95690814851 р.

```

Рисунок 3.5 Программа к инд. заданию №3

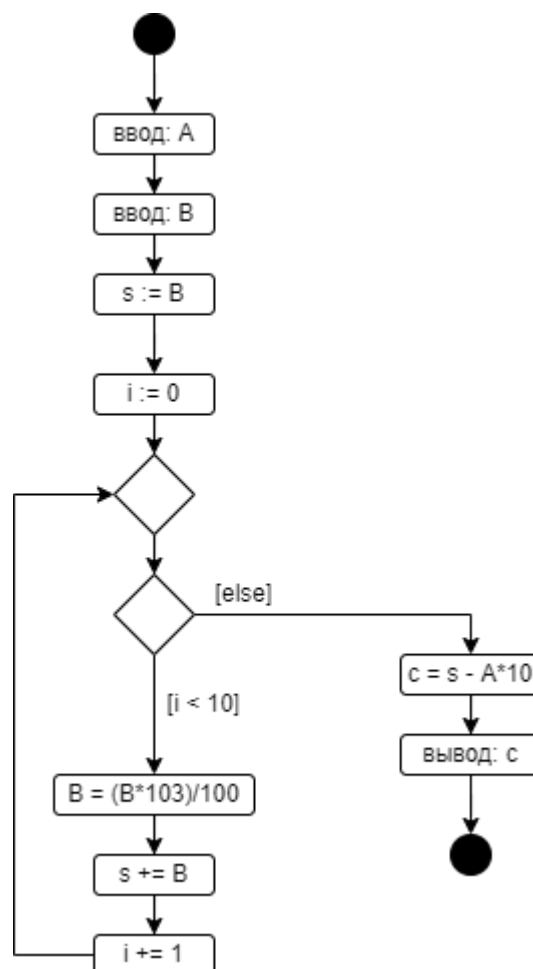


Рисунок 3.6 UML – диаграмма к программе инд. задания 3

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

Рисунок 3.7 Программа для задачи повышенной сложности.

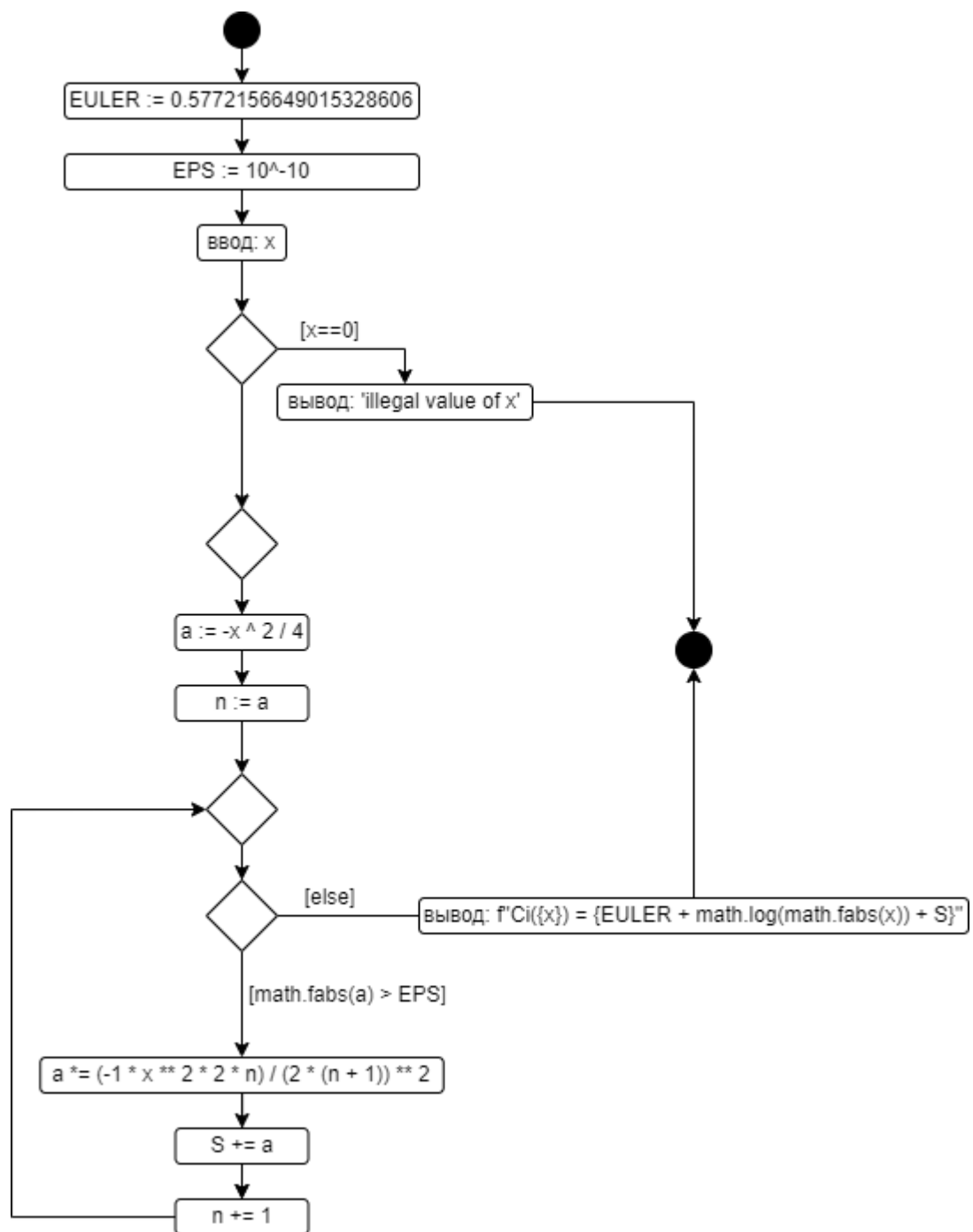


Рисунок 3.8 UML – диаграмма деятельности программы для усложненного задания

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

```

C:\LB_4>git add .
C:\LB_4>git commit -m "asd"
[develop 02ef1ba] asd
20 files changed, 325 insertions(+), 3 deletions(-)
create mode 100644 ind/ind1.py
create mode 100644 ind/ind2.py
create mode 100644 ind/ind3.py
create mode 100644 "uml \321\201\321\205\320\265\320\274\321\213\ind1.drawio"
create mode 100644 "uml \321\201\321\205\320\265\320\274\321\213\ind1.png"
create mode 100644 "uml \321\201\321\205\320\265\320\274\321\213\ind2.drawio"
create mode 100644 "uml \321\201\321\205\320\265\320\274\321\213\ind2.png"
create mode 100644 "uml \321\201\321\205\320\265\320\274\321\213\ind3.drawio"
create mode 100644 "uml \321\201\321\205\320\265\320\274\321\213\ind3.png"
create mode 100644 "uml \321\201\321\205\320\265\320\274\321\213\us1.drawio"
create mode 100644 "uml \321\201\321\205\320\265\320\274\321\213\us1.png"
create mode 100644 "uml \321\201\321\205\320\265\320\274\321\213~$ind2.drawio.bkp"
create mode 100644 "uml \321\201\321\205\320\265\320\274\321\213~$ind3.drawio.bkp"
create mode 100644 "\320\227\320\260\320\264\320\260\320\275\320\270\320\265 \320\277\320\276\320\262\321\213\321\210\3
20\265\320\275\320\276\320\271 \321\201\320\273\320\276\320\266\320\275\320\276\321\201\321\202\320\270.py"
create mode 100644 "\320\237\321\200\320\270\320\274\320\265\321\200\321\213\pr1.py"
create mode 100644 "\320\237\321\200\320\270\320\274\320\265\321\200\321\213\pr2.py"
create mode 100644 "\320\237\321\200\320\270\320\274\320\265\321\200\321\213\pr3.py"
create mode 100644 "\320\237\321\200\320\270\320\274\320\265\321\200\321\213\pr4.py"
create mode 100644 "\320\237\321\200\320\270\320\274\320\265\321\200\321\213\pr5.py"

```

Рисунок 4.1 Фиксация и коммит файлов

```

C:\LB_4>git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.

C:\LB_4>git merge develop
Updating 7e76459..02ef1ba
Fast-forward
 .gitignore | 153 ++++++
 ind/ind1.py | 20 +++
 ind/ind2.py | 20 +++
 ind/ind3.py | 14 ++
 .../ind1.drawio" | 1 +
 .../ind1.png" | Bin 0 -> 27672 bytes
 .../ind2.drawio" | 1 +
 .../ind2.png" | Bin 0 -> 43527 bytes
 .../ind3.drawio" | 1 +
 .../ind3.png" | Bin 0 -> 14549 bytes
 .../us1.drawio" | 1 +
 .../us1.png" | Bin 0 -> 30789 bytes
 .../~$ind2.drawio.bkp" | 1 +
 .../~$ind3.drawio.bkp" | 1 +
 ...266\320\275\320\276\321\201\321\202\320\270.py" | 22 +++
 .../pr1.py" | 15 ++
 .../pr2.py" | 18 +++
 .../pr3.py" | 14 ++
 .../pr4.py" | 19 +++
 .../pr5.py" | 27 +++++
 20 files changed, 325 insertions(+), 3 deletions(-)
 create mode 100644 ind/ind1.py

```

Рисунок 4.2 Слияние ветки develop с main

```

C:\LB_4>git push
info: please complete authentication in your browser...
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 8 threads
Compressing objects: 100% (25/25), done.
Writing objects: 100% (25/25), 109.36 KiB | 10.94 MiB/s, done.
Total 25 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/Konstelllation/LB_4.git
 7e76459..02ef1ba main -> main

```

Рисунок 4.3 Пуш коммитов

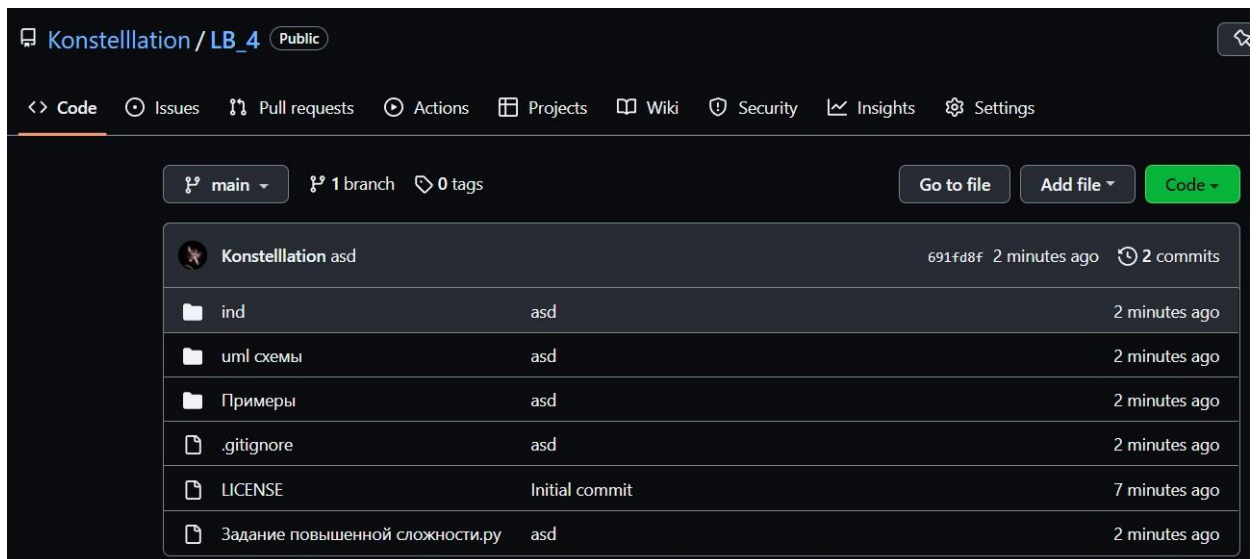


Рисунок 4.4 Изменения на уд. сервере

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 объединённых 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. 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?

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.