

**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 №4**

Дисциплина: «Основы кроссплатформенного программирования»

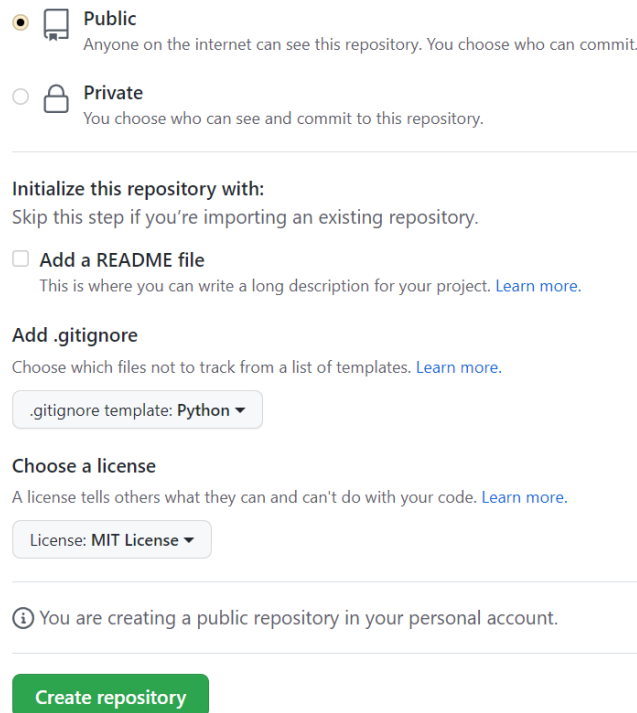
Тема: «Условные операторы и циклы в языке Python»

Выполнил: студент 1 курса  
группы ИВТ-б-о-21-1  
Уланбекова Айканыш

Ставрополь 2022

## Выполнение работы.

1. Создала репозиторий в GitHub «lab3» в который добавила .gitignore, который дополнила правила для работы с IDE PyCharm с ЯП Python, выбрала лицензию MIT, клонировала его на лок. сервер и организовала в соответствие с моделью ветвления git-flow.



☒ **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

```
276 строк (219 sloc) | 7.96 КБ
1
2 # Создано https://www.toptal.com/developers/gitignore/api/pycharm ,python
3 # Редактировать по адресу https://www.toptal.com/developers/gitignore?templates=pycharm ,python
4
5 ### PyCharm ###
6 # Охватывает IDE JetBrains: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio, WebStorm и Rider
7 # Ссылка: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839
8
9 # Пользовательские вещи
10 .идея /**/workspace.xml
11 .идея /**/tasks.xml
12 .идея /**/usage.statistics.xml
13 .idea /**/словари
14 .idea /**/полка
15
16 # AWS для конкретного пользователя
```

Рисунок 1.2 Добавление правил в .gitignore

```
C:\Users\User>git clone https://github.com/aikanyshkaukanbekova/lab4.git
Cloning into 'lab4'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), 6.14 KiB | 1.02 MiB/s, done.

C:\Users\User>git flow init
Initialized empty Git repository in C:/Users/User/.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/User/.git/hooks]
```

Рисунок 1.3 Клонирование и организация репозитория согласно модели ветвления git-flow

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

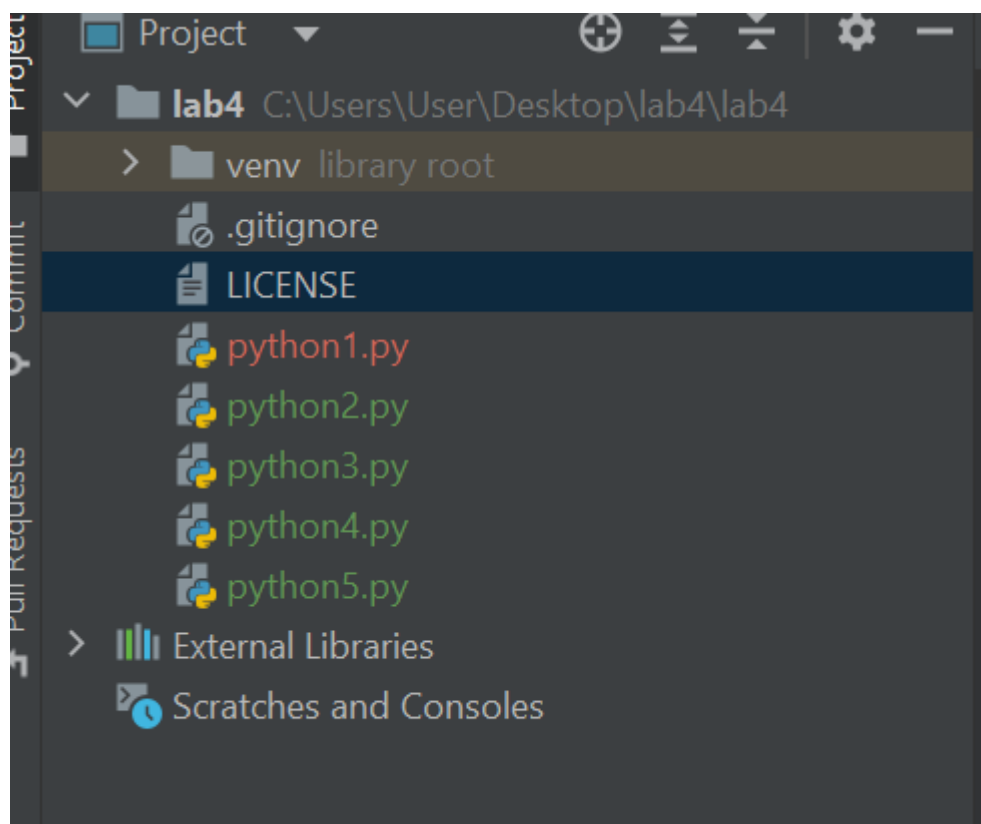


Рисунок 2.1 Примеры в проекте

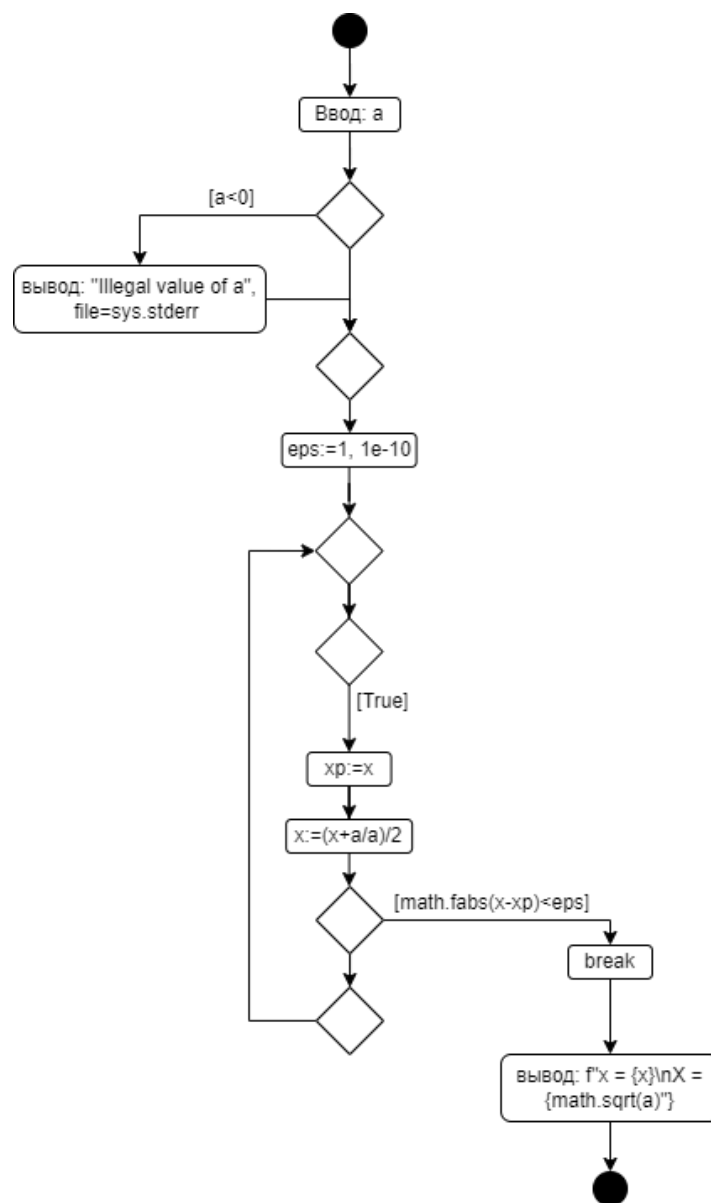


Рисунок 2.2 UML-диаграмма программы 4 примера

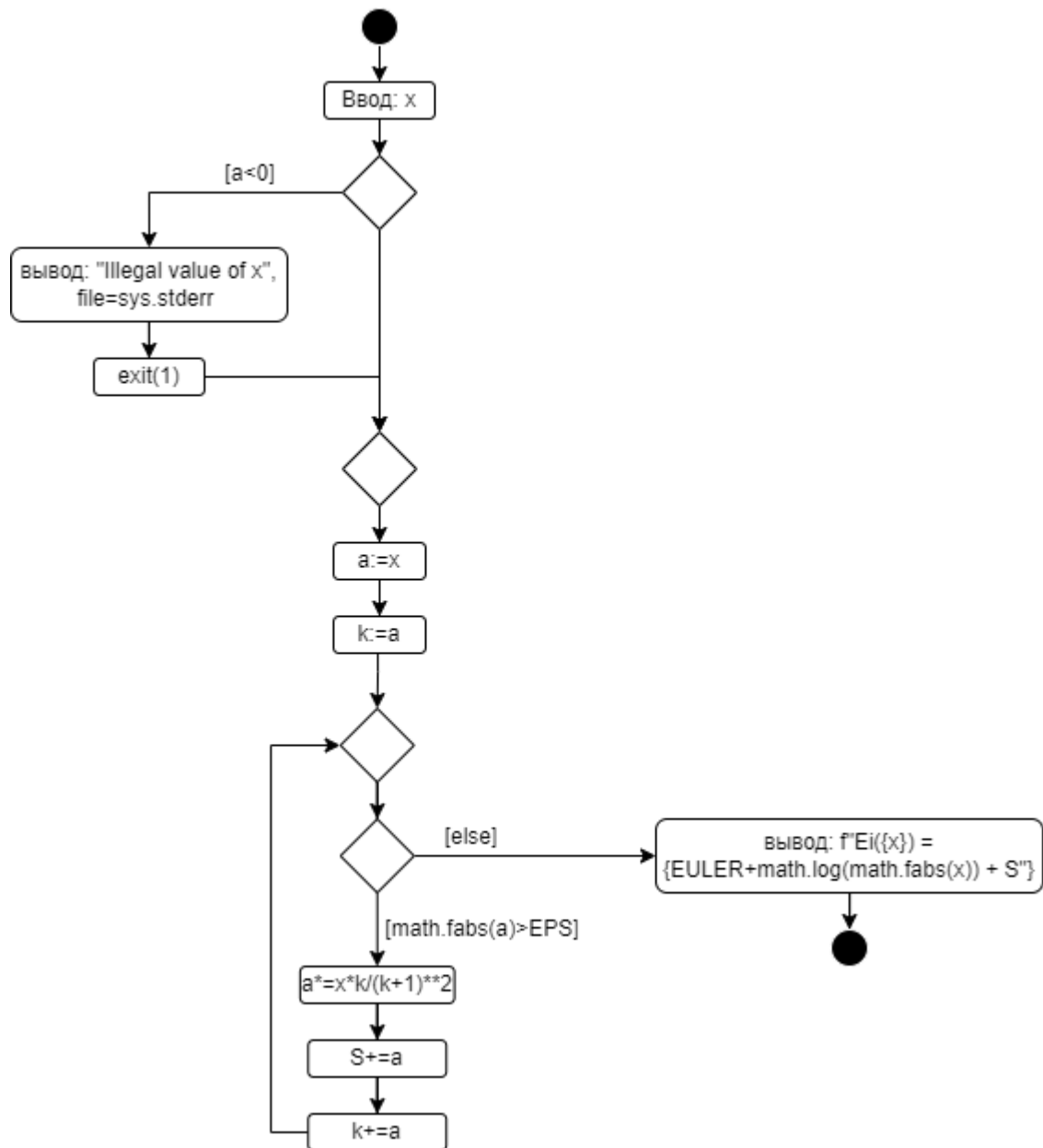


Рисунок 2.3 UML-диаграмма программы 5 примера

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

```

"""
Вводится число N<=20 экзаменов . Напечатать фразу "Мы успешно сдали N экзаменов",
согласовав слово "экзамен" с числом .
"""
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == '__main__':
    N = int(input("Введите число сданных экзаменов: "))
    print("Мы успешно сдали ", N, " экзаменов")
  
```

Рисунок 3.1 Программа к инд. заданию №1



Рисунок 3.2 UML – диаграмма к программе инд. задания 1

```
Треугольник задан координатами своих вершин. Определить принадлежит ли данная точка
треугольнику. Координаты вершин треугольника и координаты точки задать самостоятельно.
"""
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == '__main__':
    ax = int(input('Введите координаты первой вершины:\n'))
    ay = int(input())
    bx = int(input('Введите координаты второй вершины:\n'))
    by = int(input())
    cx = int(input('Введите координаты третьей вершины:\n'))
    cy = int(input())
    xx = int(input('Введите координаты точки:\n'))
    xy = int(input())

    """
    Если точка лежит внутри треугольника, то сумма площадей образованных ей треугольников
    равна площади данного треугольника
    """

    if(abs((ax - xx)*(by - xy) - (bx - xx)*(ay - xy)) +
    if(abs((ax - xx)*(by - xy) - (bx - xx)*(ay - xy)) +
        abs((ax - cx)*(xy - cy) - (xx - cx)*(ay - cy)) +
        abs((xx - cx)*(by - cy) - (bx - cx)*(xy - cy)) ==
        abs((ax - cx)*(by - cy) - (bx - cx)*(ay - cy))):
        print('Точка ВХОДИТ в треугольник')
    else:
        print('Точка НЕ ВХОДИТ в треугольник')
```

Рисунок 3.3 Программа к инд. заданию №1

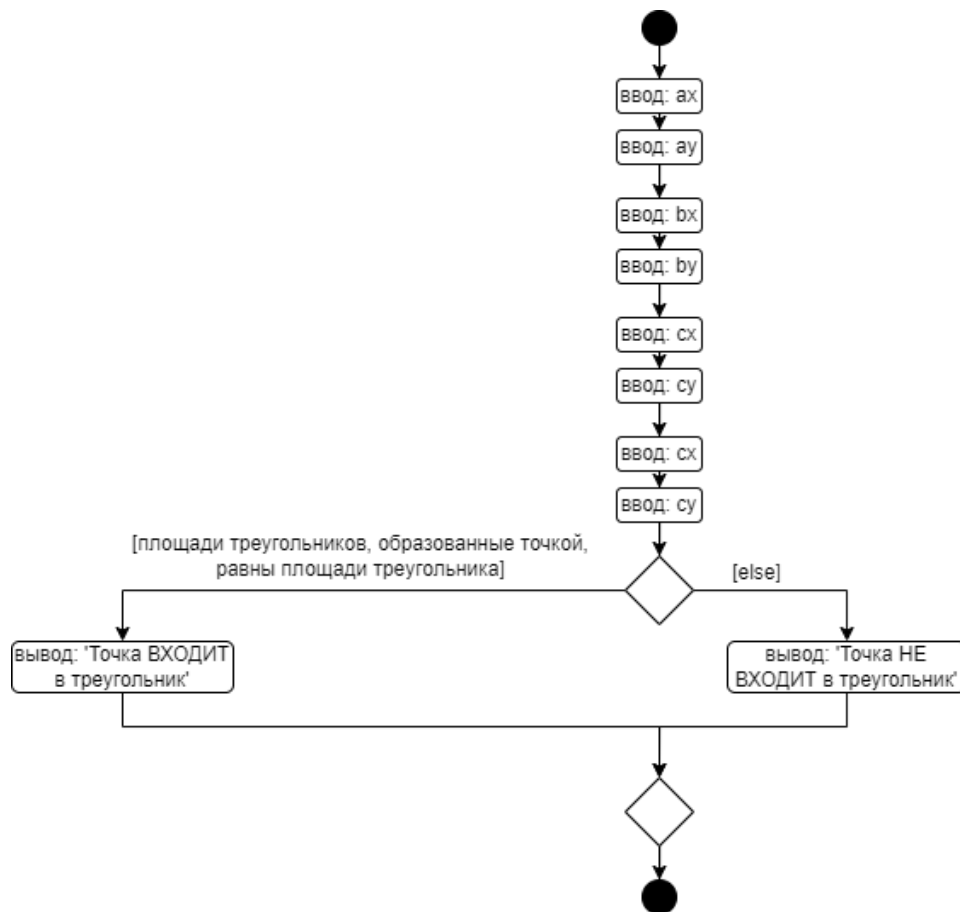


Рисунок 3.4 UML – диаграмма к программе инд. задания 2

```

Составьте программу, которая по номеру дня в году выводит число и месяц
в общепринятой форме (например, 33-й день года - 2 февраля).
"""
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == '__main__':
    NumDay = int(input('add number day'))
    if NumDay <= 31:
        print(NumDay, f'-й день года - ', NumDay, '.01', sep='')
    else:
        m = 0
        month = 0
        day = 0
        for month in range(1, 13):
            if month == 2:
                a = 28
            elif (month == 1 or month == 3 or month == 5 or month == 7 or
                  month == 8 or month == 10 or month == 12):
                a = 31
            else:
                a = 30
            m += a
            if NumDay - m <= 31:
                day = NumDay - m
                if NumDay > 31:
                    month += 1
                break

        if month < 10:
            print(NumDay, f'-й день года - ', day, '.0', month, sep='')
        else:
            print(NumDay, f'-й день года - ', day, '.', month, sep='')
  
```

Рисунок 3.5 Программа к инд. заданию №3

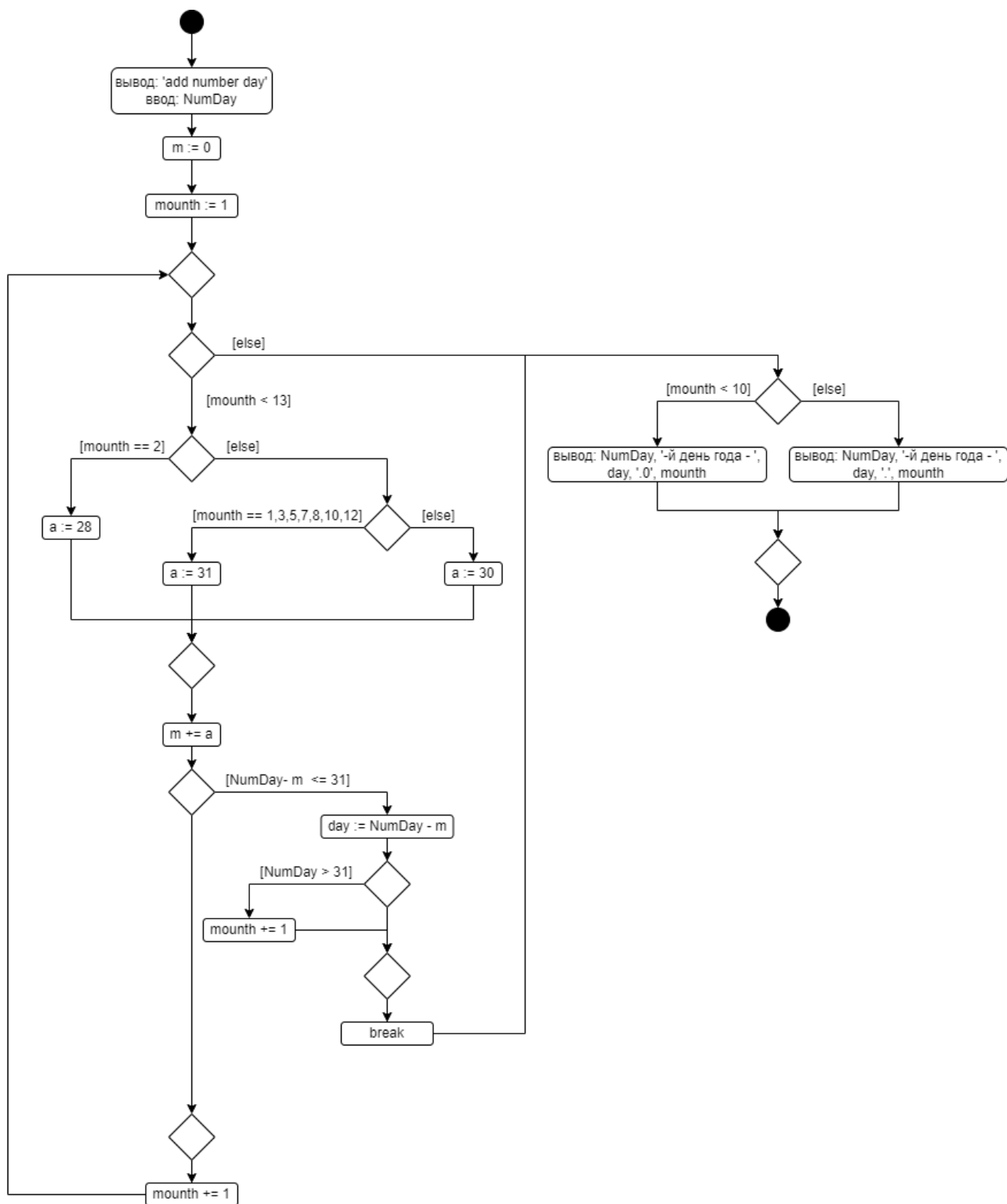


Рисунок 3.6 UML – диаграмма к программе инд. задания 3



```

EULER = 0.5772156649015328606
EPS = 1e-10

if __name__ == '__main__':
    x = float(input("x = "))
    if x == 0:
        print("Error", file=sys.stderr)
        exit(1)
    a = -x ** 2 / 4
    S, n = a, 1
    while math.fabs(a) > EPS:
        a *= (-1 * x ** 2 * 2 * n) / (2 * (n + 1)) ** 2
        S += a
        n += 1
    print(f"f'Ci({x}) = {EULER + math.log(math.fabs(x)) + S}")

```

Рисунок 3.7 Программа для задачи повышенной сложности.

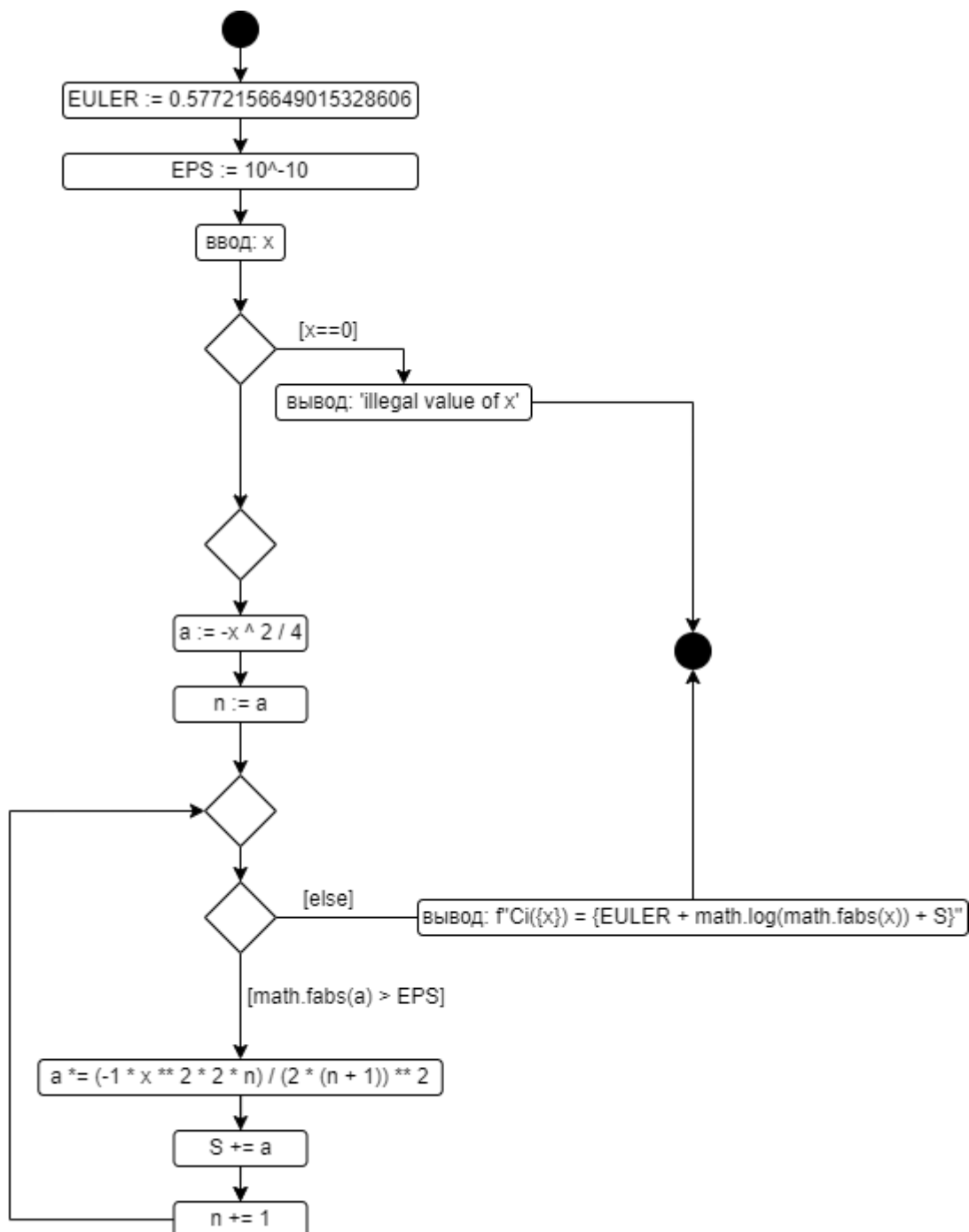


Рисунок 3.8 UML – диаграмма деятельности программы для усложненного задания

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

```
C:\Users\User\Desktop\lab4\lab4>git commit -m "first com"
[main e75e9d8] first com
7 files changed, 54 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/lab4.iml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 python1.py

C:\Users\User\Desktop\lab4\lab4>git checkout main
Already on 'main'
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

C:\Users\User\Desktop\lab4\lab4>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

C:\Users\User\Desktop\lab4\lab4>git merge develop
Merge made by the 'ort' strategy.
 indiv1.py      | 10 ++++++++
 indiv2.py      | 28 ++++++++
 indiv3.py      | 34 ++++++++
 primer1.py     | 17 ++++++++
 python2.py     | 20 ++++++++
 python3.py     | 16 ++++++++
 python4.py     | 21 ++++++++
 python5.py     | 29 ++++++++
 uslagnennoe.py | 22 ++++++++
9 files changed, 197 insertions(+)
create mode 100644 indiv1.py
create mode 100644 indiv2.py
create mode 100644 indiv3.py
create mode 100644 primer1.py
create mode 100644 python2.py
create mode 100644 python3.py
create mode 100644 python4.py
create mode 100644 python5.py
create mode 100644 uslagnennoe.py

C:\Users\User\Desktop\lab4\lab4>git push
Enumerating objects: 24, done.
Counting objects: 100% (24/24), done.
Delta compression using up to 8 threads
Compressing objects: 100% (21/21), done.
Writing objects: 100% (23/23), 4.96 KiB | 564.00 KiB/s, done.
Total 23 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/aikanyshkaukanbekova/lab4.git
 ad28a22..6bd0124  main -> main
```

Рисунок 4.1 Работа в GIT CMD
















|                                                                                                                                                                                                    |                       |                |                                                                                                                   |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|----------------|-------------------------------------------------------------------------------------------------------------------|
|  aikanyshkaukanbekova Объединить филиал "развитие" <span>6bd0124 2 минуты назад</span> <span>🗨️ 5 коммитов</span> |                       |                | Нет описания, веб-сайта или тем.<br>MIT license<br>0 звезд<br>1 смотреть<br>0 вилков                              |
|  .идея                                                                                                            | первый ком            | 10 минут назад |                                                                                                                   |
|  .gitignore                                                                                                       | Обновить .gitignore   | 3 часа назад   |                                                                                                                   |
|  ЛИЦЕНЗИЯ                                                                                                         | Первоначальный коммит | 3 часа назад   |                                                                                                                   |
|  indiv1.py                                                                                                        | первый ком            | 30 минут назад |                                                                                                                   |
|  indiv2.py                                                                                                        | первый ком            | 30 минут назад |                                                                                                                   |
|  indiv3.py                                                                                                        | первый ком            | 30 минут назад |                                                                                                                   |
|  primer1.py                                                                                                       | первый ком            | 30 минут назад |                                                                                                                   |
|  python1.py                                                                                                       | первый ком            | 10 минут назад |                                                                                                                   |
|  python2.py                                                                                                       | первый ком            | 30 минут назад |                                                                                                                   |
|  python3.py                                                                                                       | первый ком            | 30 минут назад |                                                                                                                   |
|  python4.py                                                                                                       | первый ком            | 30 минут назад |                                                                                                                   |
|  python5.py                                                                                                       | первый ком            | 30 минут назад |                                                                                                                   |
|  uslagnennoe.py                                                                                                   | первый ком            | 30 минут назад |                                                                                                                   |
|                                                                                                                                                                                                    |                       |                | <b>РЕЛИЗЫ</b><br>Релизы не опубликованы<br><a href="#">Создать новый выпуск</a>                                   |
|                                                                                                                                                                                                    |                       |                | <b>Пакетов</b><br>Нет опубликованных пакетов <a href="#">Опубликуйте свой первый пакет</a>                        |
|                                                                                                                                                                                                    |                       |                | <b>Языки</b><br> Python 100.0% |

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. 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.