**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

Гайибов Хасан
Мамадиерович

Ставрополь 2022

Выполнение работы.

1. Создал репозиторий в GitHub «lab3» 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.

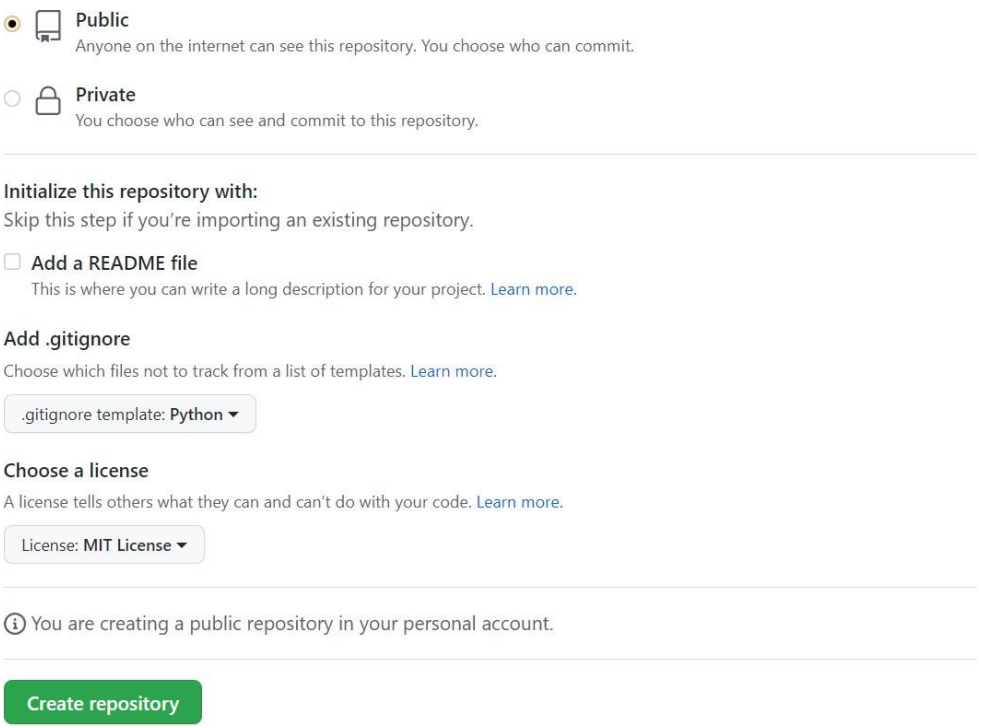


Рисунок 1.1 Создание репозитория

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>cd C:\Users\User\Desktop\laba3

C:\Users\User\Desktop\laba3>git clone https://github.com/Balkhievhusein/laba3.git
Cloning into 'laba3'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), 6.01 KiB | 879.00 KiB/s, done.
```

```
C:\Users\User\Desktop\laba3\laba3>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:/Users/User/Desktop/laba3/laba3/.git/hooks]
```

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

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

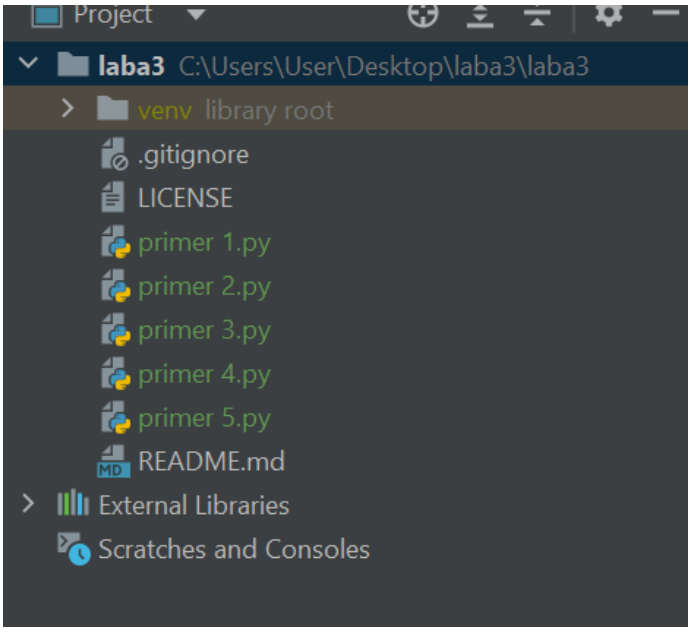


Рисунок 2.1 Примеры в проекте

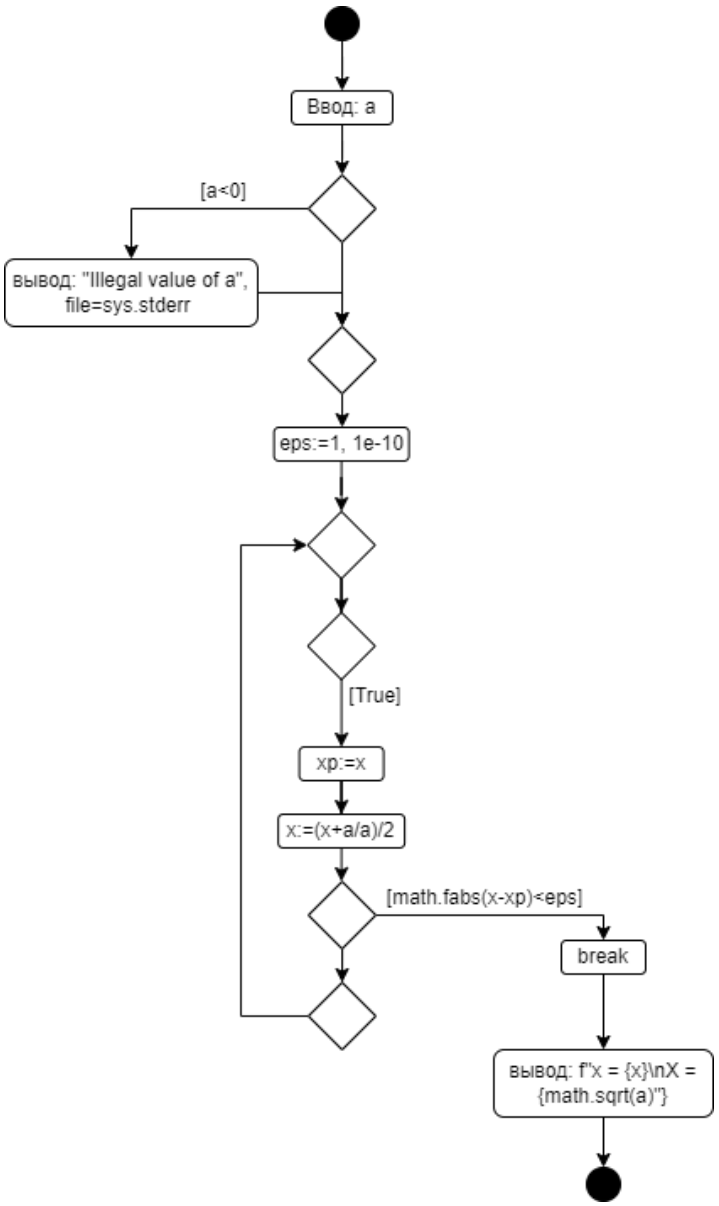


Рисунок 2.2 UML-диаграмма программы 4 примера

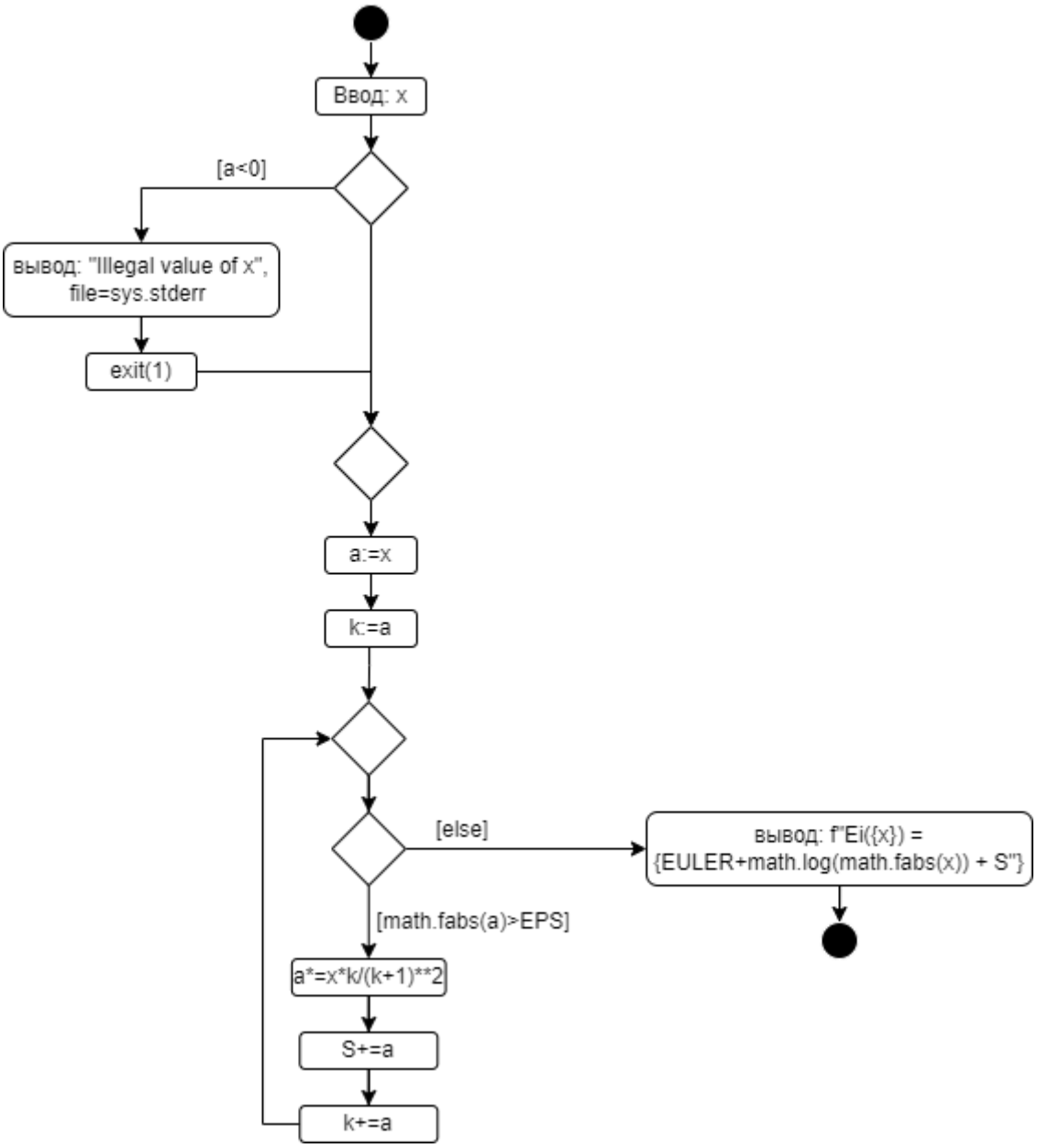


Рисунок 2.3 UML-диаграмма программы 5 примера

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

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
nb=int(input())
m_th={1:'Январь',2:'Февраль',3:'Март',4:'Апрель',5:'Май',6:'Июнь',7:'Июль',8:'Август',9:'Сентябрь',10:'
if nb<13:
    print(m_th[nb])
else:print("Error")
```

Рисунок 3.1 Программа к инд. заданию №1

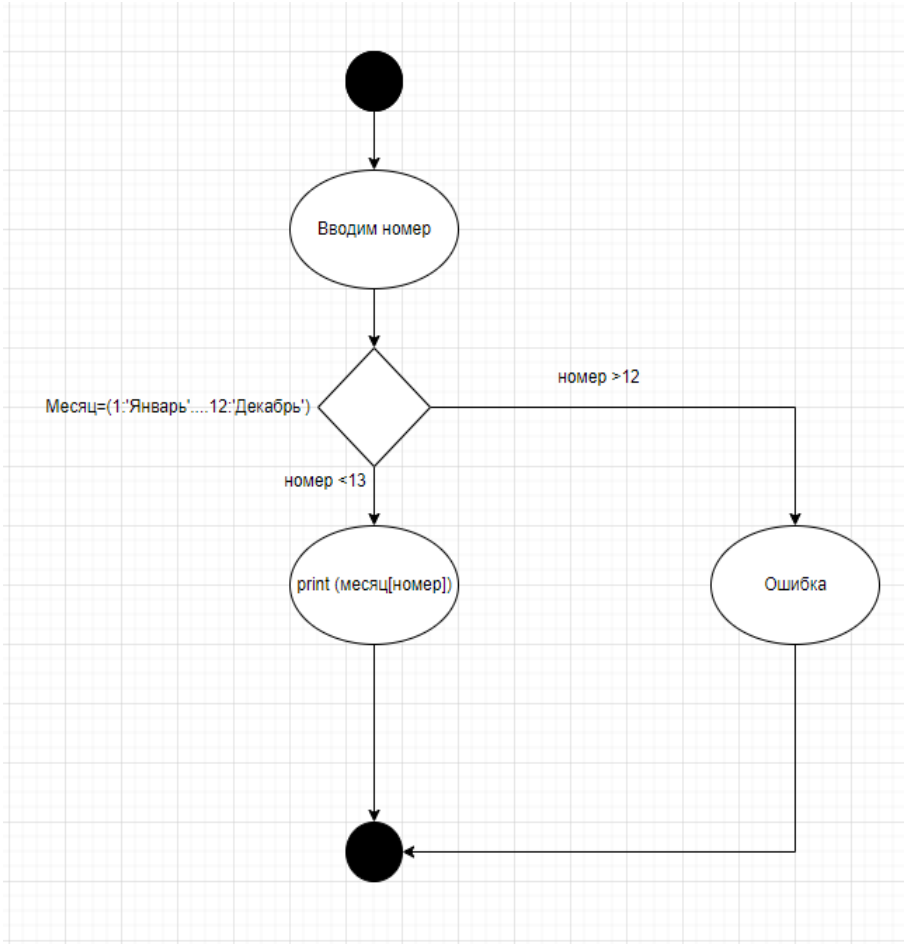


Рисунок 3.2 UML – диаграмма к программе инд. задания 1

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
a, b, c = map(int, input('Введите a, b, c через пробел: ').spl
D = b*b - 4*a*c

x = []

if D < 0:
    print('Действительных корней нет')
else:
    t1 = (-b+D**(1/2))/(2*c)
    t2 = (-b-D**(1/2))/(2*c)

    if t1 >= 0: x.append(t1**(1/2))
    if t2 >= 0: x.append(t2**(1/2))

    print('Действительные корни:', *x, sep=' '+chr(177))
```

Рисунок 3.3 Программа к инд. заданию №2

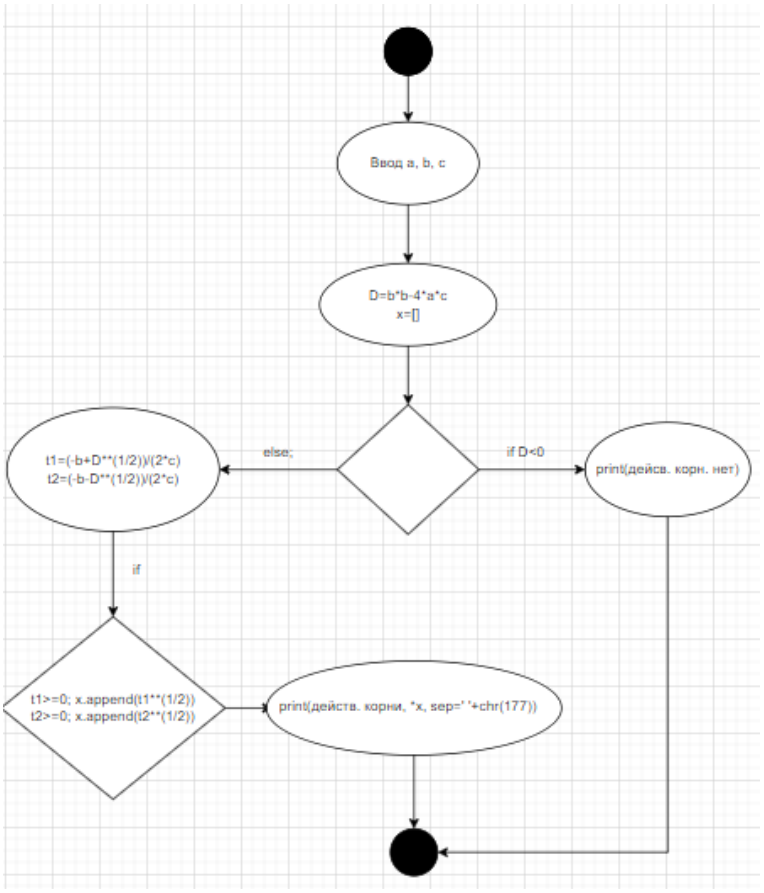


Рисунок 3.4 UML – диаграмма к программе инд. задания 2

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
i=input('Введите число: ')
print('Да' if int(i)%sum(map(int, list(i)))==0 else 'Нет')
```

Рисунок 3.5 Программа к инд. заданию №3

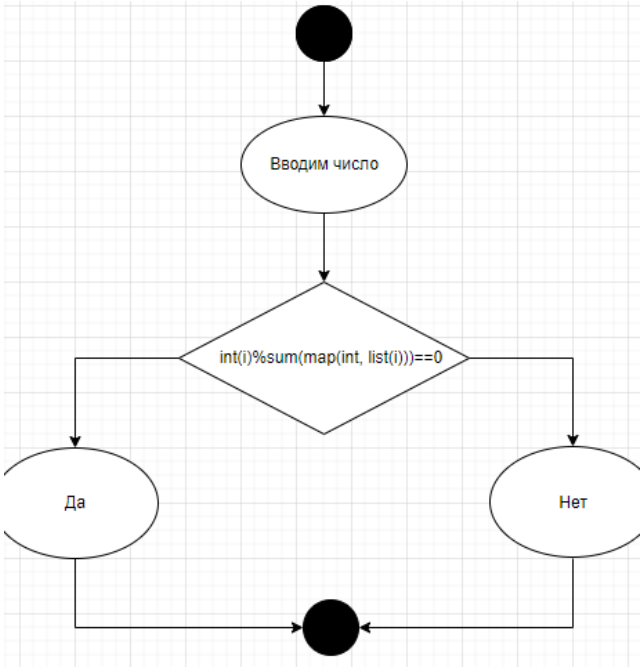


Рисунок 3.6 UML – диаграмма к программе инд. задания 3

```
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"Ci({x}) = {EULER + math.log(math.fabs(x)) + S}")
```

Рисунок 3.7 Программа для задачи повышенной сложности.

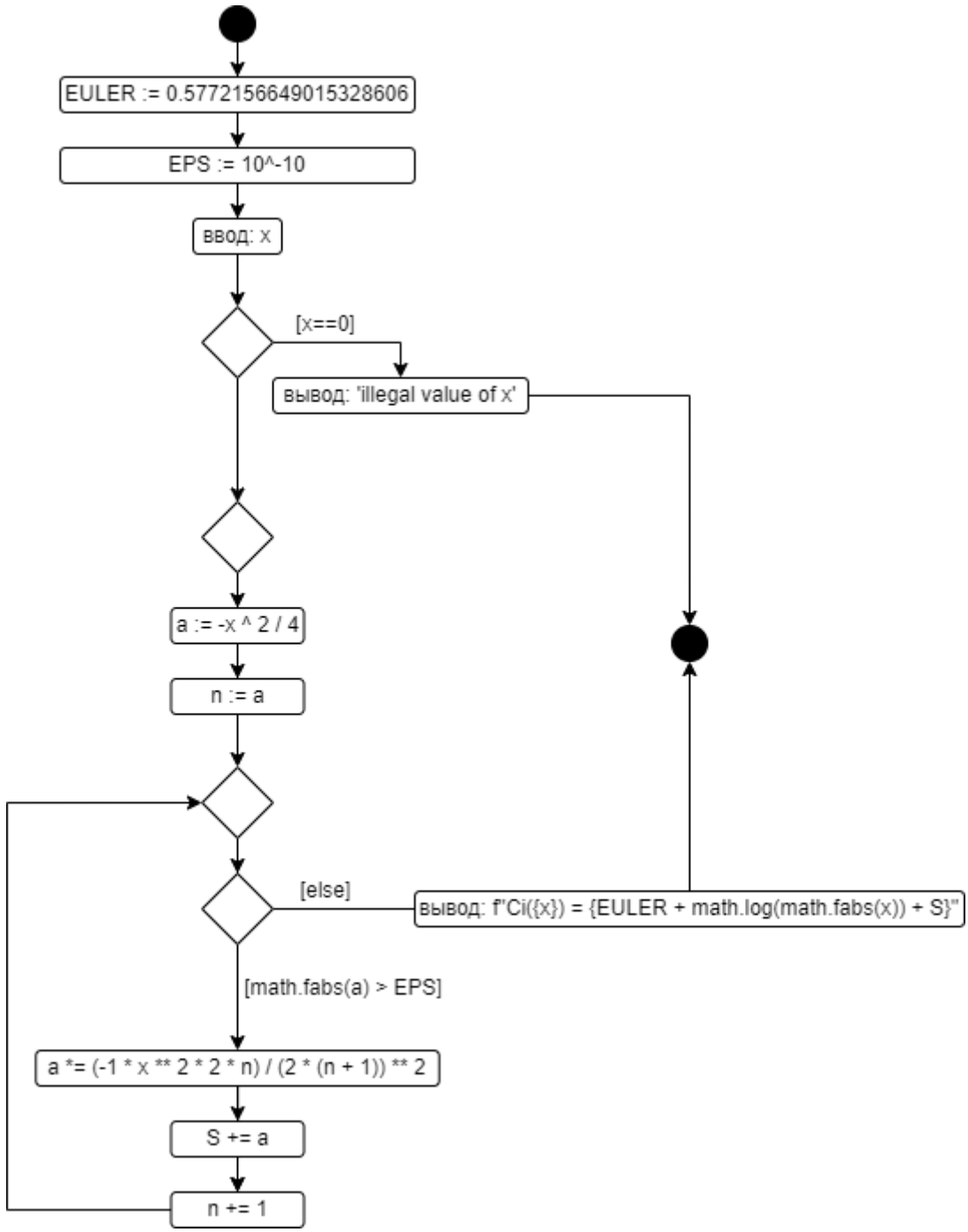


Рисунок 3.8 UML – диаграмма деятельности программы для усложненного задания

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

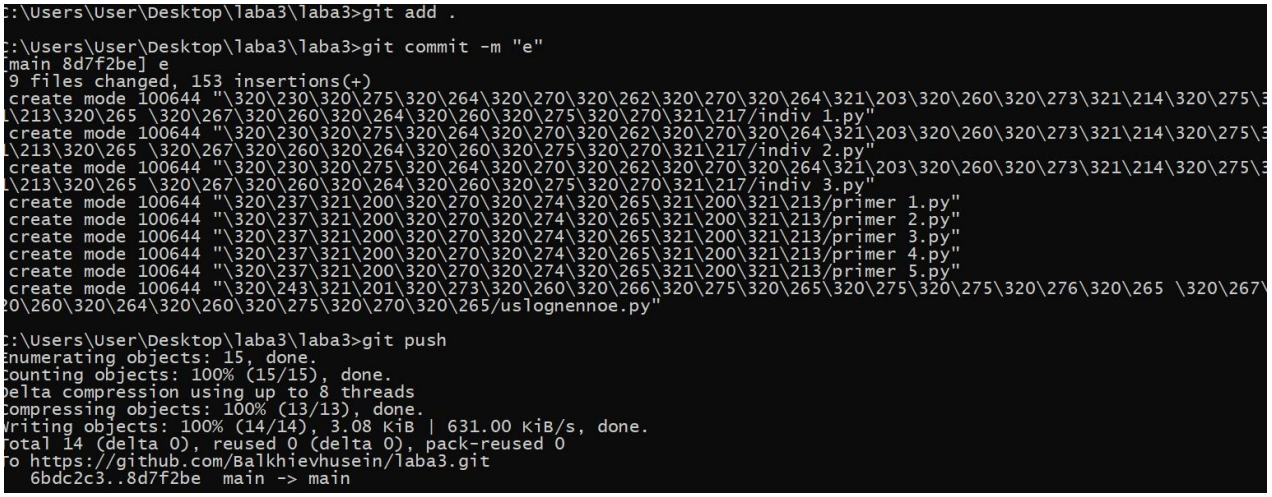


Рисунок 4.1 Работа в GIT CMD

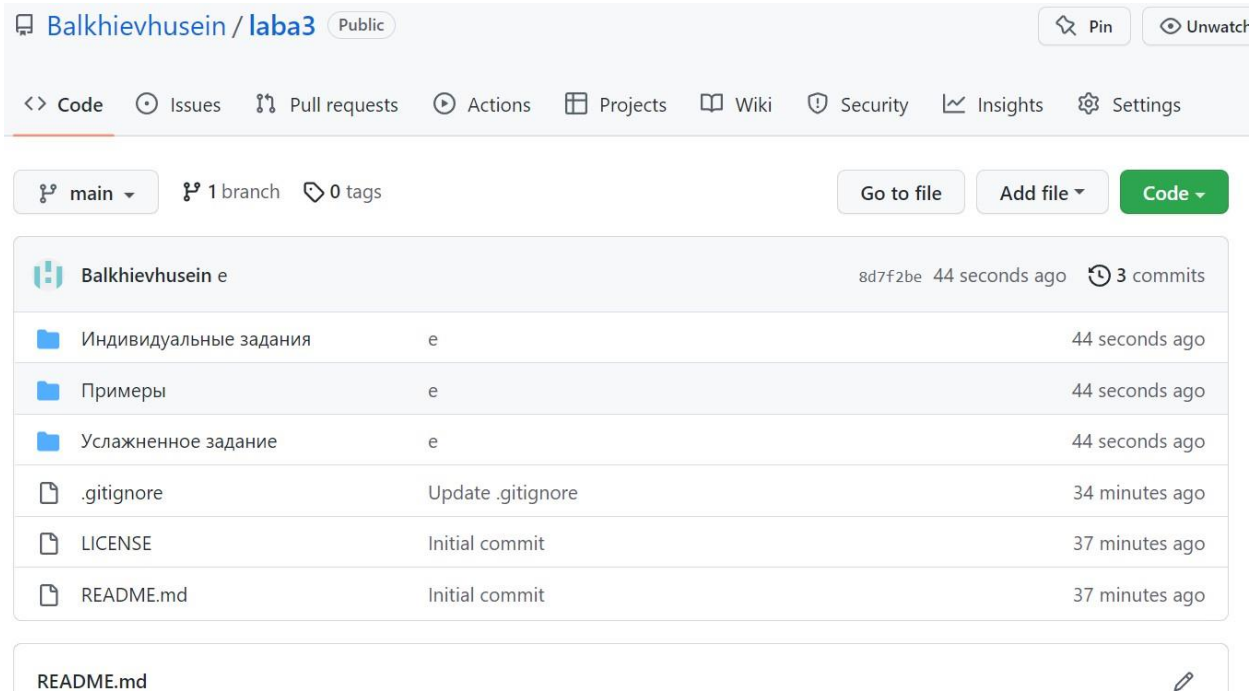


Рисунок 4.2 Изменения на уд. сервере

**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.