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 программной инженерии»

Тема: «Условные операторы и циклы в языке Python»

Выполнил: студент

2 курса

группы Пиж-б-о-21-1

Коныжев Максим

Викторович

Ставрополь 2022

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.

1. Был создан общедоступного репозиторий в GitHub в котором были добавлены gitignore, правила для работы с IDE PyCharm с ЯП Python и лицензия MIT, репозиторий был клонировал на локальный сервер и организован в соответствие с моделью ветвления git-flow.

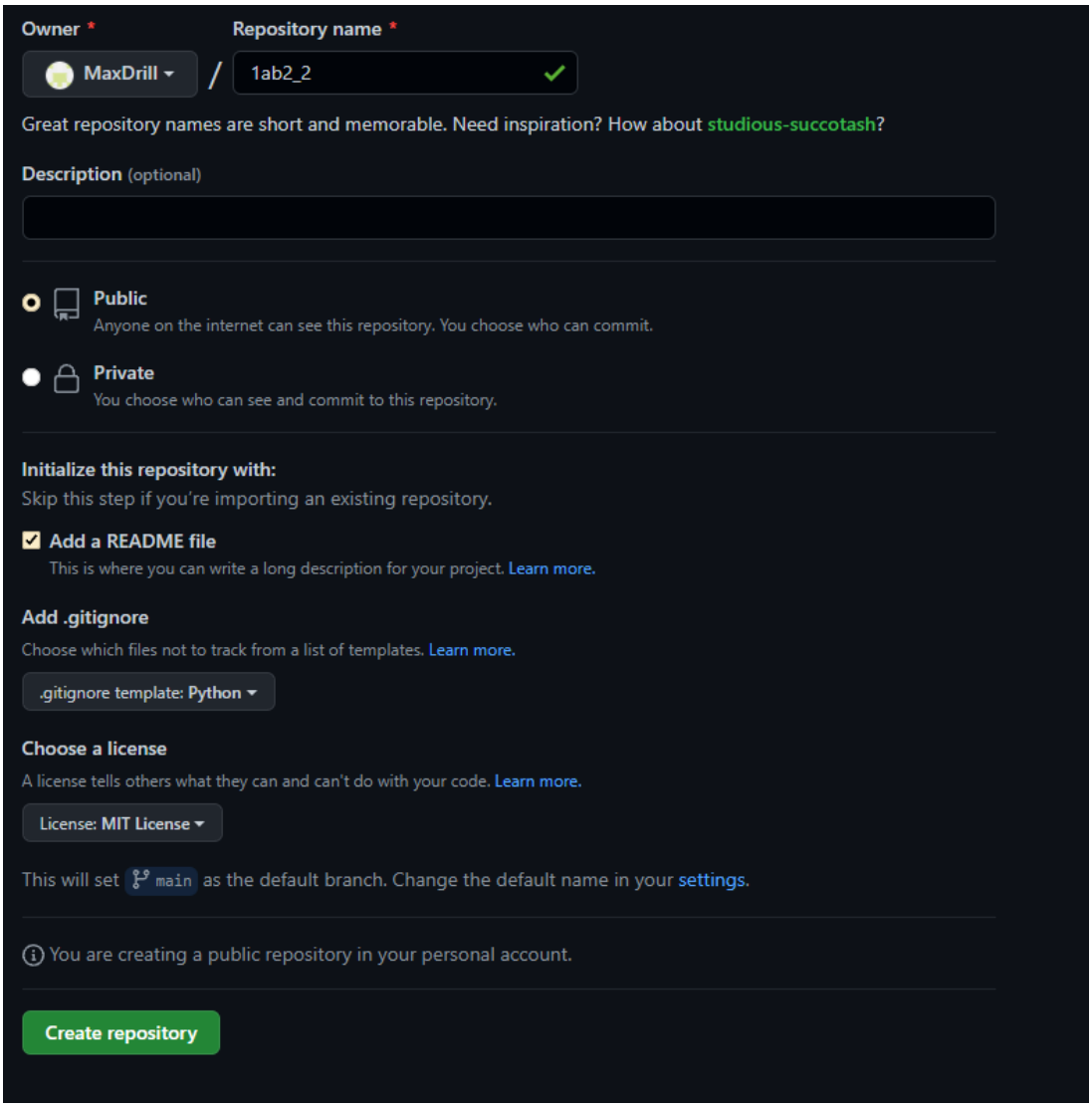

Рисунок 1.1 – Создание общедоступного репозитория

```
C:\Users\UESR\gitproj>git clone

C:\Users\UESR\gitproj>git clone https://github.com/MaxDrill/lab2_2.git
Cloning into 'lab2_2'...
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
Receiving objects: 100% (5/5), done.
```

Рисунок 1.2 – Клонирование созданного репозитория

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

Рисунок 1.3 – Организование репозитория по модели ветвления git flow

2. Была создана папка (proj), содержащая примеры из лабораторной работы

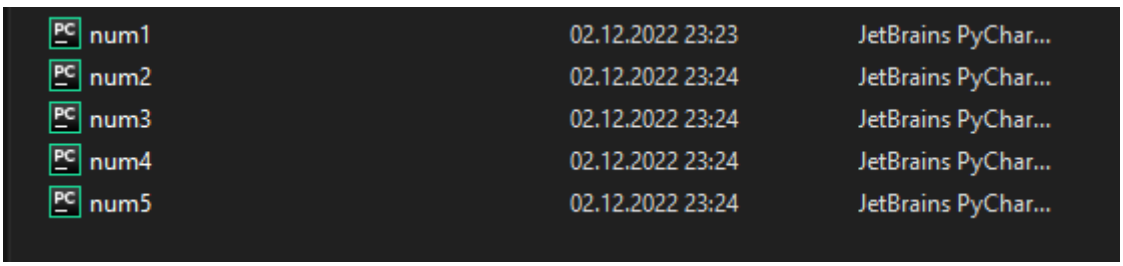

Рисунок 2.1 – Содержание папки proj

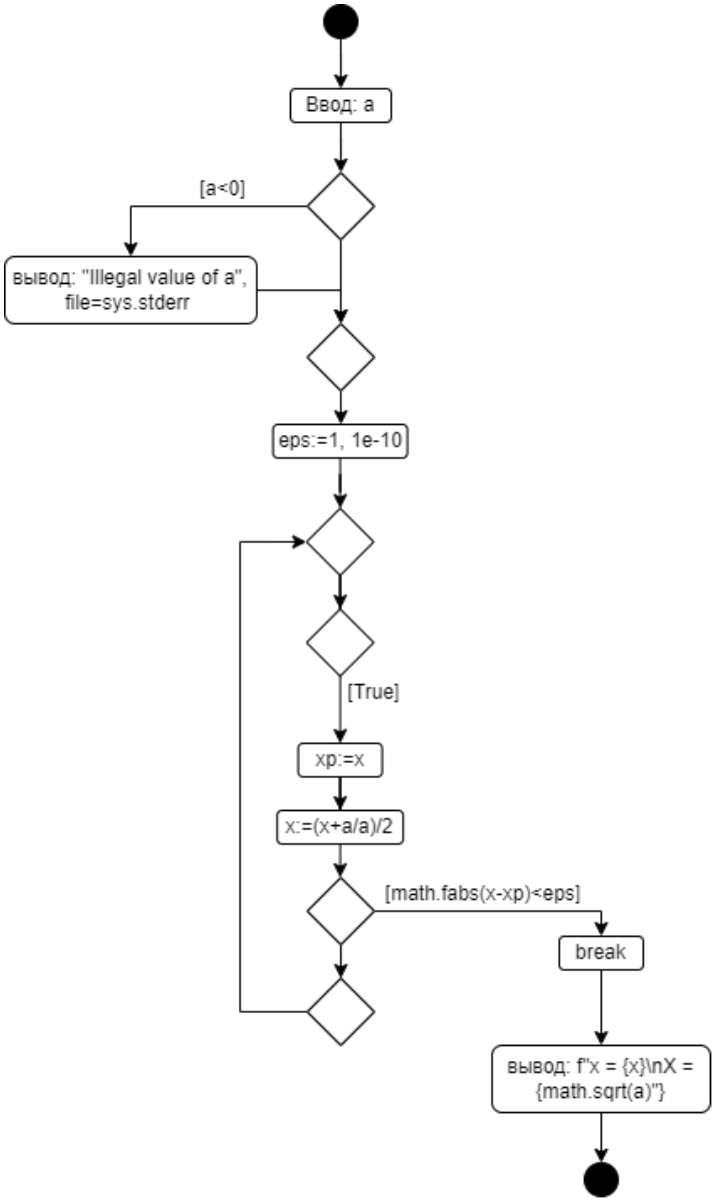

Рисунок 2.2 – UML-диаграмма программы для 4 примеры

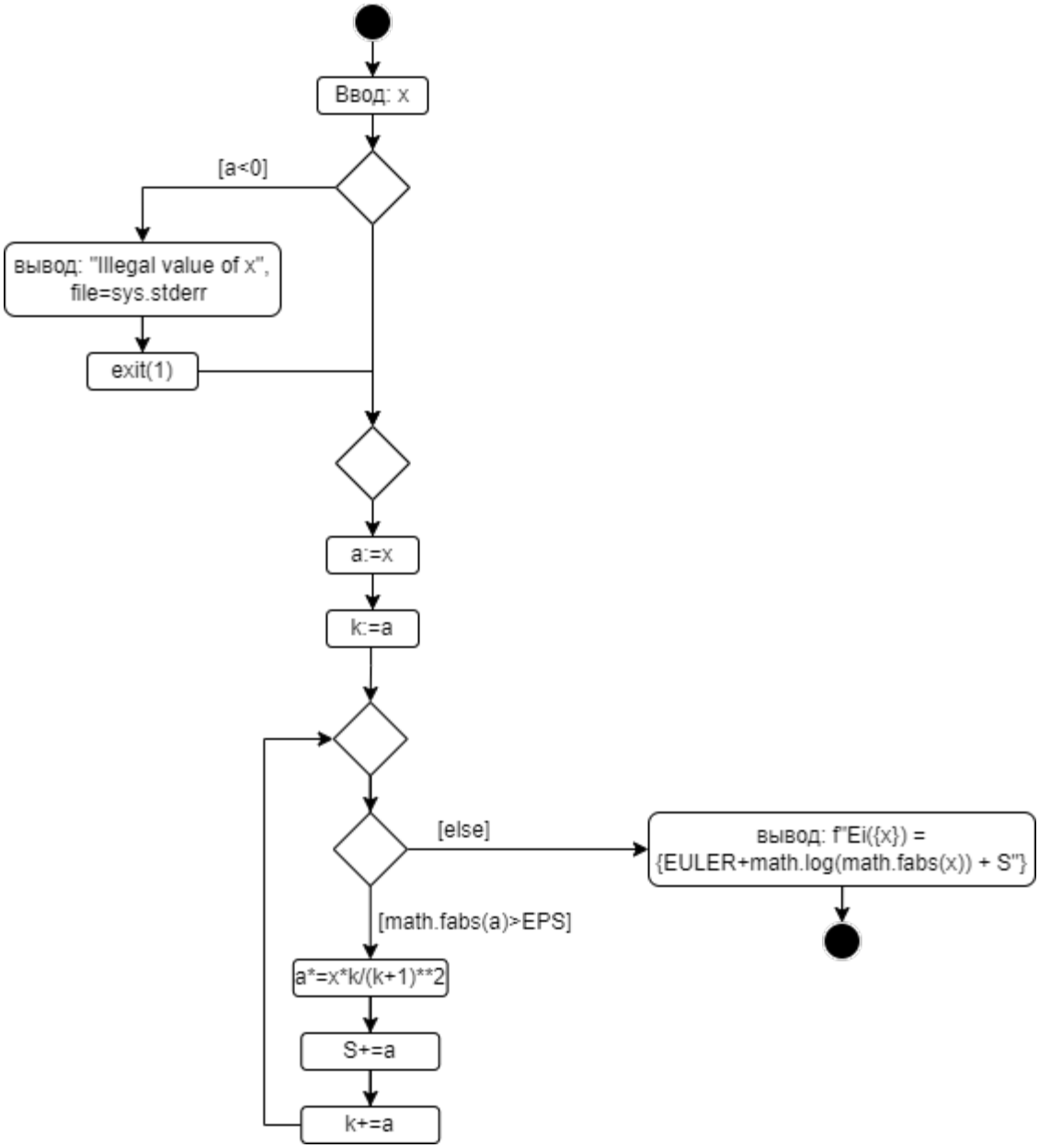

Рисунок 2.3 – UML-диаграмма для программы 5 примера

Индивидуальное задание

3 Было сделано индивидуальное задание согласно вариантам. Была построена UML диаграмма.

Вариант 7

Задание 1.

7. С клавиатуры вводится цифра m (от 1 до 12). Вывести на экран название месяца, соответствующего цифре.

Код программы:

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

if __name__ == '__main__':
    m = int(input("Введите число от 1 до 12: "))
    if m == 1:
        print("Январь")
    elif m == 2:
        print("Февраль")
    elif m == 3:
        print("Март")
    elif m == 4:
        print("Апрель")
    elif m == 5:
        print("Май")
    elif m == 6:
        print("Июнь")
    elif m == 7:
        print("Июль")
    elif m == 8:
        print("Август")
    elif m == 9:
        print("Сентябрь")
    elif m == 10:
        print("Октябрь")
    elif m == 11:
        print("Ноябрь")
    elif m == 12:
        print("Декабрь")
    else:
        print("Вы ввели число не от 1 до 12")
```

```
C:\Users\UESR\gitproj\lab2_2\proj\venv\Scripts\python.exe C:/User
Введите число от 1 до 12: 7
Июль

Process finished with exit code 0
```

Рисунок 3.1 – Результат работы программы

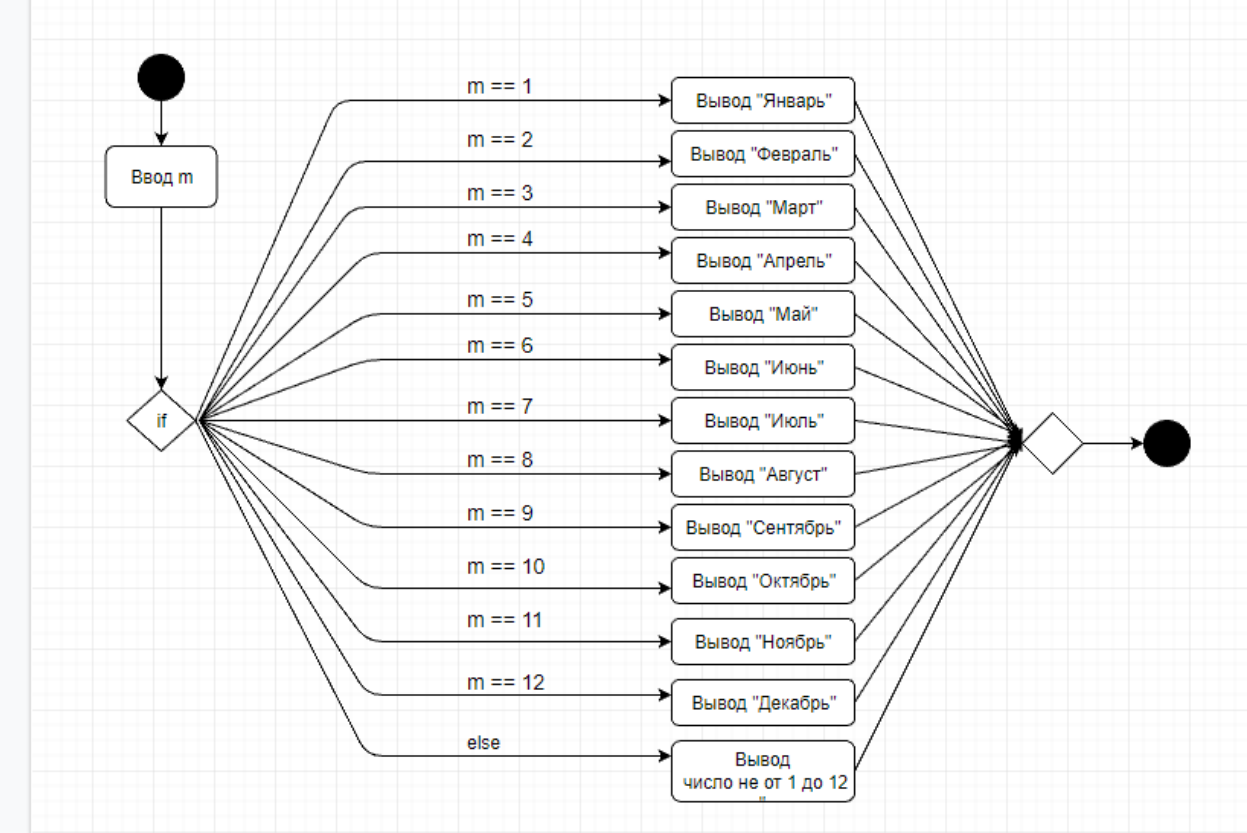

Рисунок 3.2 – UML диаграмма программы

Задание 2.

7. Провести исследование биквадратного уравнения $a\,x^4 + b\,x^2 + c = 0$ $(a \neq 0)$, где a, b и c - действительные числа. Если действительных корней нет, то об этом должно быть выдано сообщение, иначе должны быть выданы 2 или 4 действительных корня.

Код программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
if __name__ == '__main__':
    a = float(input("Введите коэф a: "))
    b = float(input("Введите коэф b: "))
    c = float(input("Введите коэф c: "))
    if a != 0:
        d = b**2 - 4 * a * c
        if d == 0:
            t = -b / (2 * a)
            x1 = t ** 0.5
            x2 = - (t ** 0.5)
            print(x1, x2)
        elif d > 0:
            t1 = (-b + d ** 0.5) / (2 * a)
            t2 = (-b - d ** 0.5) / (2 * a)
            x3 = t1 ** 0.5
            x4 = - (t1 ** 0.5)
            x5 = t2 ** 0.5
            x6 = - (t2 ** 0.5)
            print("Корни: ", x3, x4, x5, x6)
        elif d < 0:
            print("Корней нет")
    else:
        print("a не должно = 0")
```

```
Введите коэф a: 4
Введите коэф b: 4
Введите коэф c: 1
(4.329780281177467e-17+0.7071067811865476j) (-4.329780281177467e-17-0.7071067811865476j)
```

Рисунок 3.3 – Результат работы программы

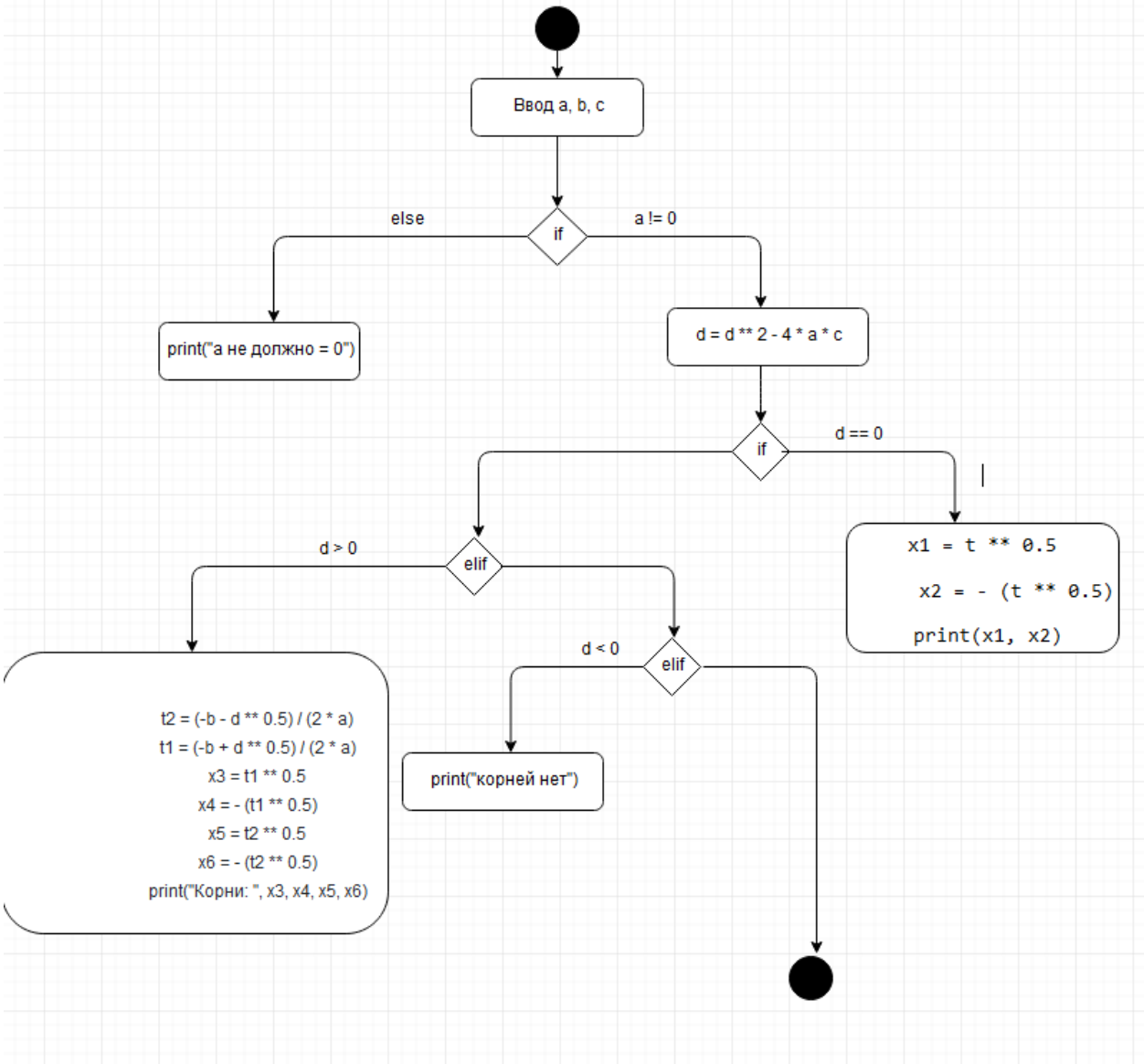

Рисунок 3.4 – UML диаграмма программы

Задание 3.

7. Определить среди всех двузначных чисел те, которые делятся на сумму своих цифр.

Код программы:

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

if __name__ == '__main__':
    N = int(input("Vvedite razmer spiska: "))
    lst = [int(input("Vvedite chis: ")) for i in range(N)]
    for i in range(N):
        a = lst[i] % 10
        b = lst[i] // 10
        summ = a + b
        if lst[i] % summ == 0:
            a = 0
            b = 0
            print(lst[i], "delitsa")
        else:
            print(lst[i], "ne delitsa")
```

```
Vvedite razmer spiska: 5
Vvedite chis: 12
Vvedite chis: 23
Vvedite chis: 34
Vvedite chis: 45
Vvedite chis: 56
12 delitsa
23 ne delitsa
34 ne delitsa
45 delitsa
56 ne delitsa
```

Рисунок 3.5 – Результат работы программы

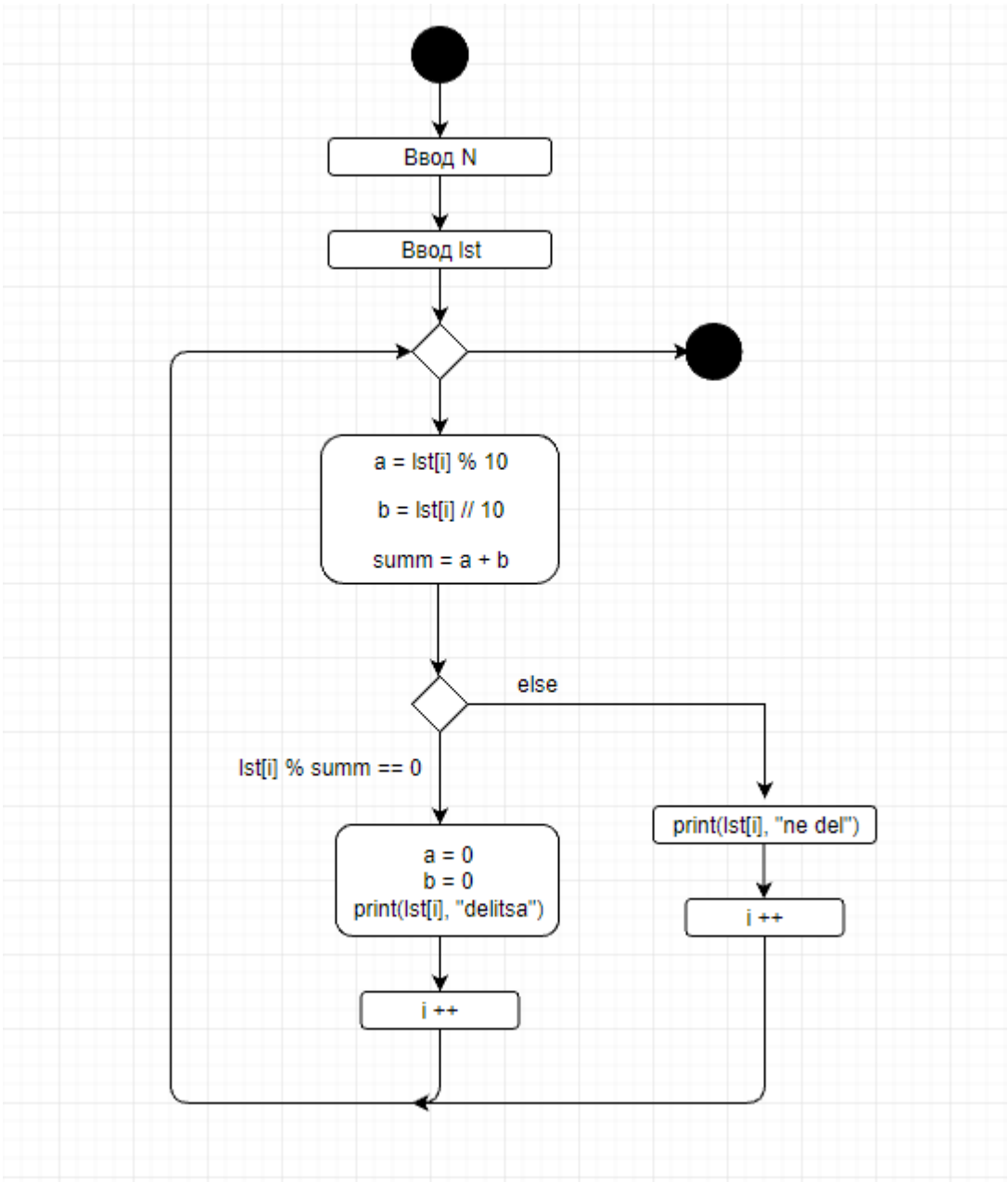

Рисунок 3.6 – UML диаграмма программы

```
C:\Users\UESR\gitproj\lab2_2>git add .

C:\Users\UESR\gitproj\lab2_2>git commit -m "Add readme and py files"
[main d3551b8] Add readme and py files
 9 files changed, 171 insertions(+), 1 deletion(-)
 create mode 100644 proj/idz1.py
 create mode 100644 proj/idz2.py
 create mode 100644 proj/idz3.py
 create mode 100644 proj/num1.py
 create mode 100644 proj/num2.py
 create mode 100644 proj/num3.py
 create mode 100644 proj/num4.py
 create mode 100644 proj/num5.py
```

Рисунок 3.7 – Был сделан коммит изменений

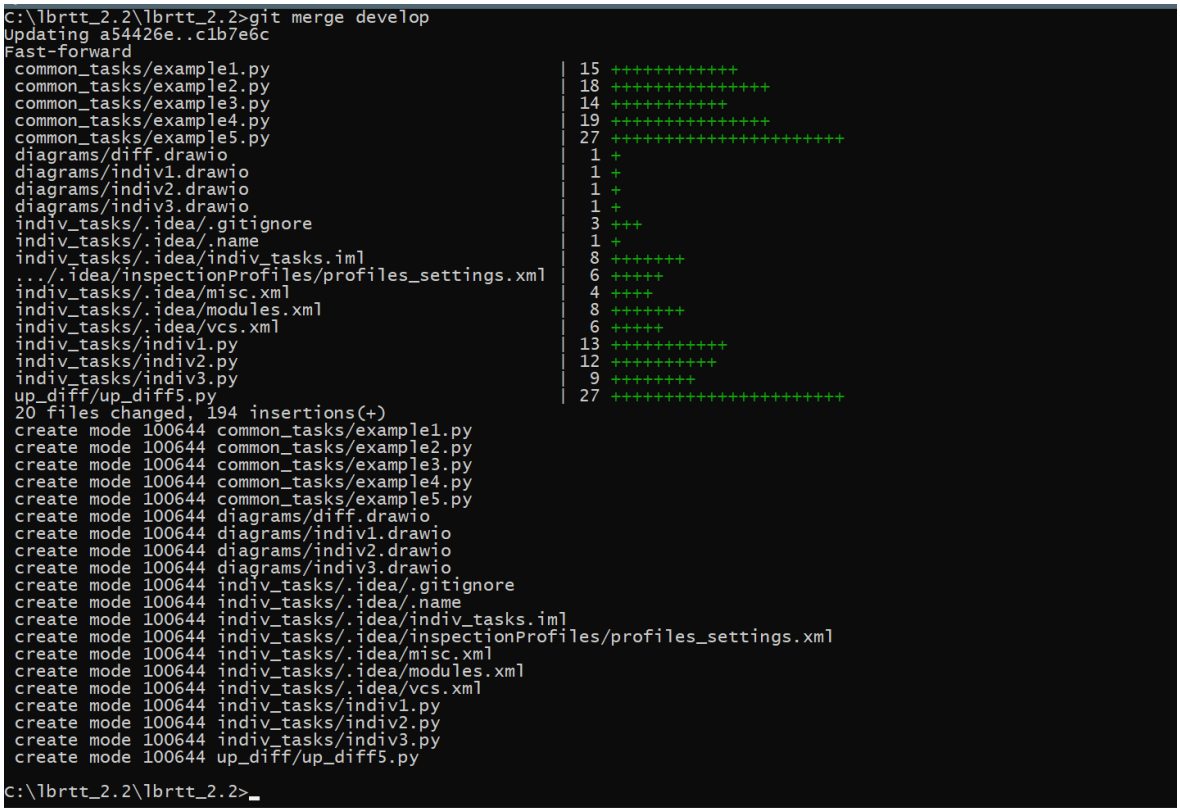

Рисунок 3.8 – Было осуществлено слияние веток main и develop

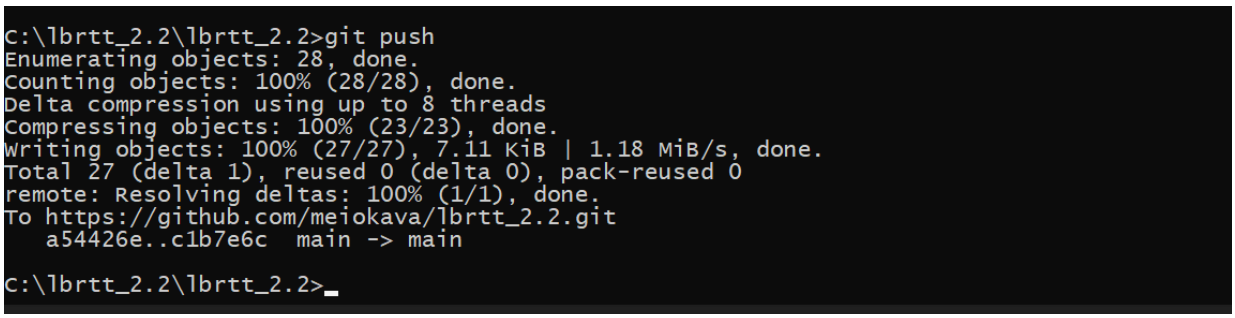

Рисунок 3.9 – Была осуществленная отправка изменений на удаленный сервер

Вывод: в результате лабораторной 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.

Контрольные вопросы

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.