Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2.2
дисциплины
«Основы кроссплатформенного программирования»

Выполнила студентка группы
ИВТ-б-о-21-1
Яковлева Е.А. « »_____20__г.
Подпись студента_____
Работа защищена
« »_____20__г.

Проверил доцент
Кафедры инфокоммуникаций,
старший преподаватель
Воронкин Р.А.
_____
(подпись)

Ставрополь, 2022 г.

Тема: условные операторы и циклы в языке Python

Порядок выполнения работы:

Создала общедоступный репозиторий на GitHub, в котором использованы лицензия MIT и язык программирования Python.

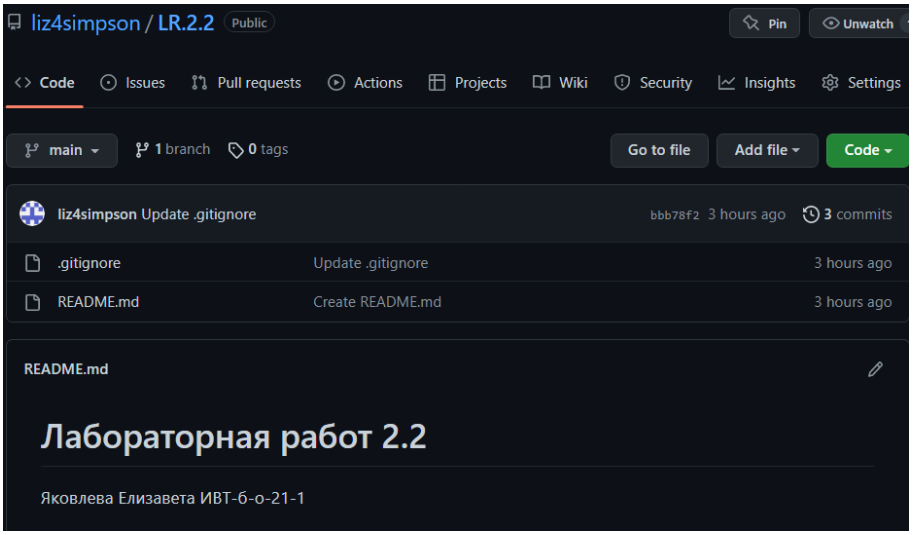

Рисунок 1. Создание репозитория

Выполнила клонирование созданного репозитория.

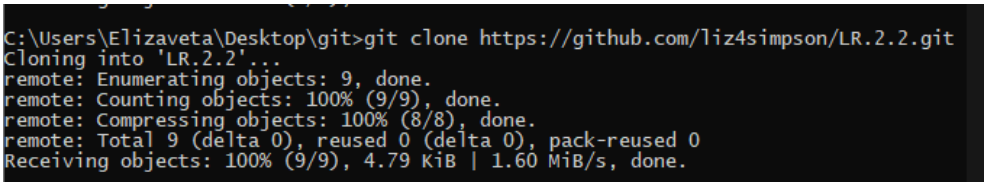

Рисунок 2. Клонирование репозитория

Организовала свой репозиторий в соответствие с моделью ветвления git-flow.

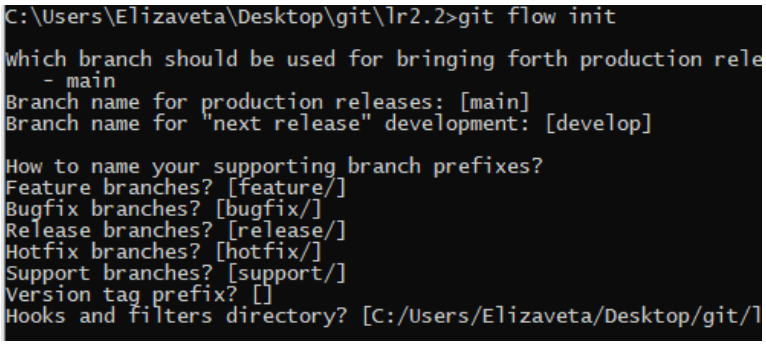

Рисунок 3. Организация репозитория согласно модели ветвления get-flow

2. Создала проект PyCharm в папке репозитория, проработала примеры из лабораторной работы.

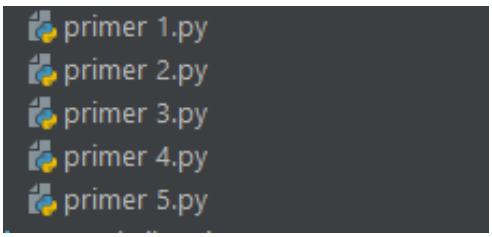

Рисунок 4. Проработанные примеры

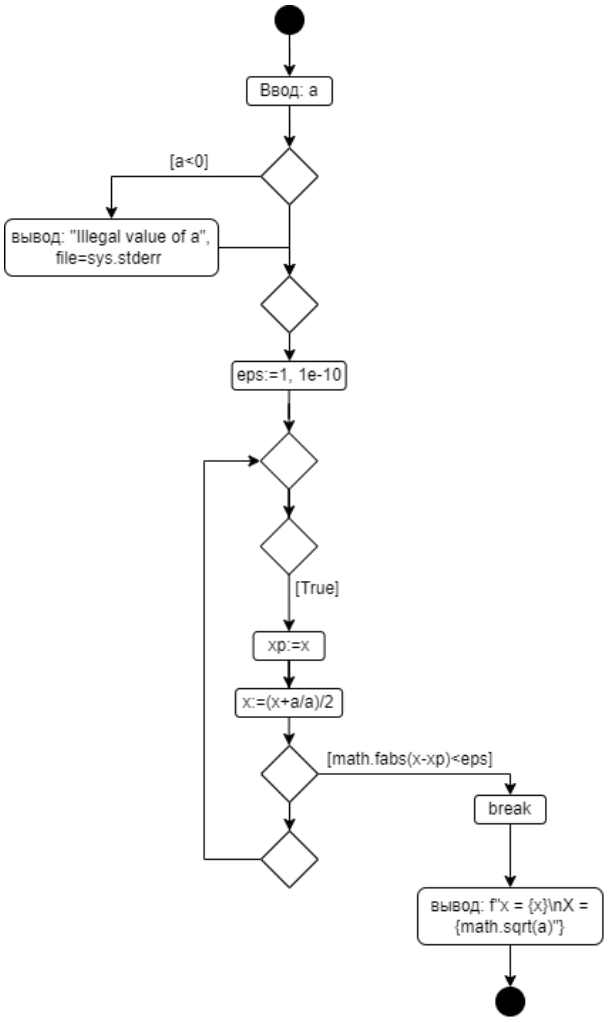

Рисунок 5. UML-диаграмма программы 4 примера

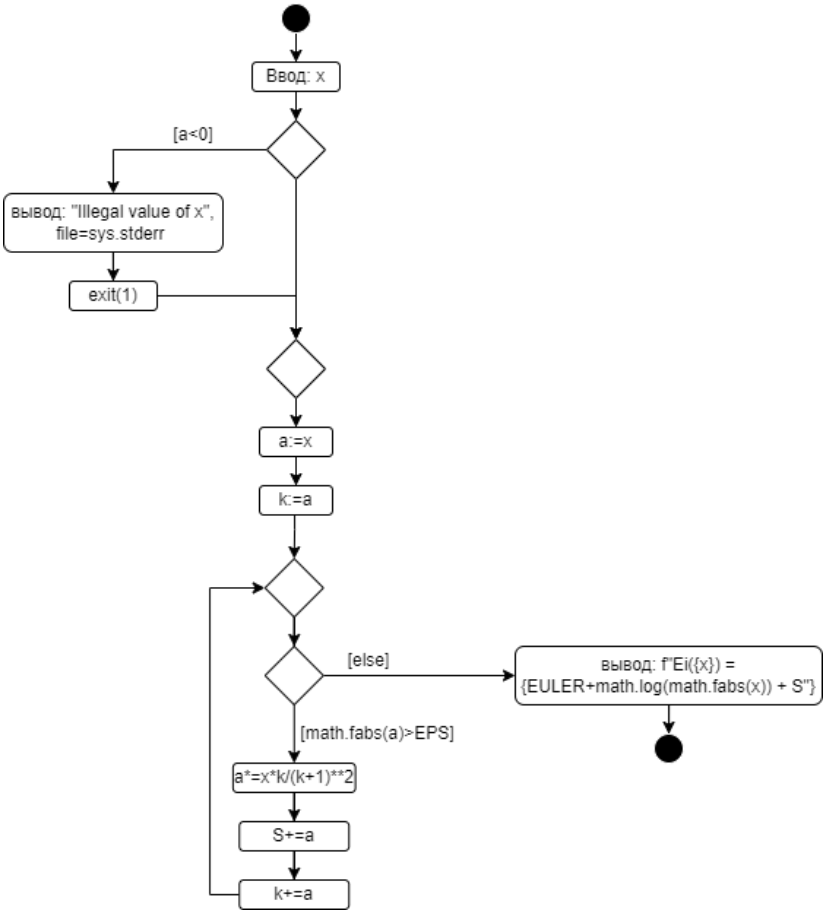

Рисунок 6. UML-диаграмма программы 5 примера

3. Выполнила индивидуальные задания. Построила UML диаграммы программ.

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

if __name__ == '__main__':
    # Ввод номера месяца
    m = int(input('Введите номер месяца: '))

    # Проверка на полугодие
    if m <= 6:
        print('Месяц приходится на первое полугодие.\n')
    else:
        print('Месяц приходится на второе полугодие.\n')

    # Провека на количество дней в месяце
    if m == 2:
        print('В месяце под номером ', m, ' 28 дней.')
    elif m == 1 or m == 3 or m == 5 or m == 7 or m == 8 or m == 10 or m == 12:
```

```
if __name__ == '__main__'
Terminal:  4.1.py  +

C:\Users\Elizaveta\PycharmProjects\pythonProject6>C:/Users/Elizaveta/PycharmProjects/pytho
Введите номер месяца: 3
Месяц приходится на первое полугодие.

В месяце под номером  3  31 дней.
```

Рисунок 7. Индивидуальное задание №1

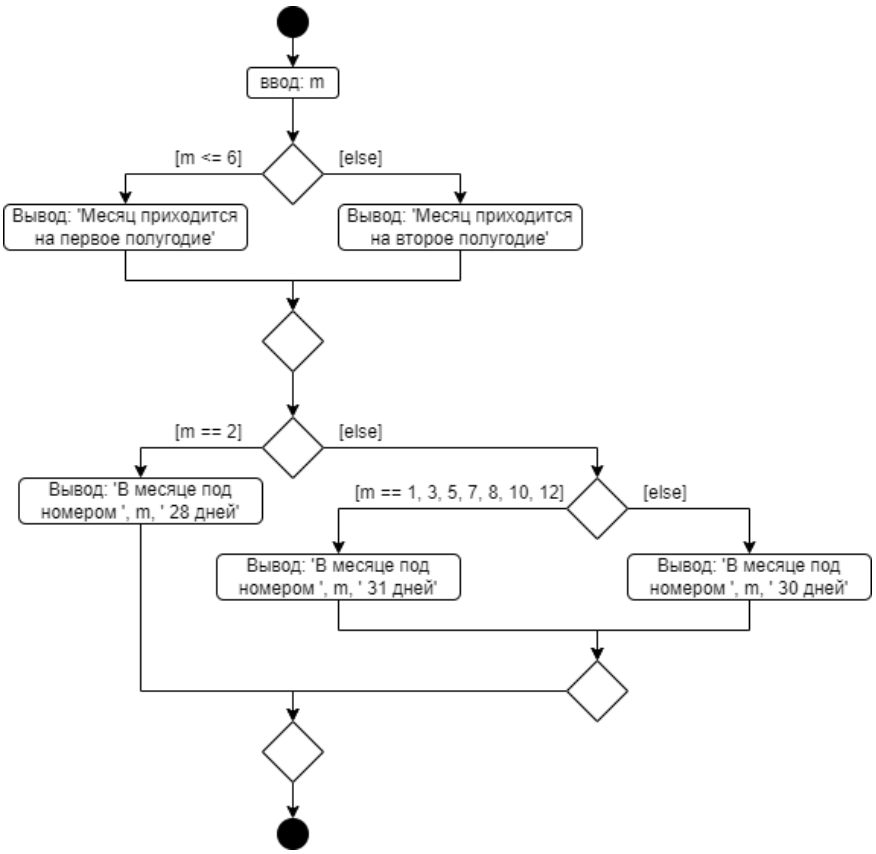

Рисунок 8. UML-диаграмма для 1 задания

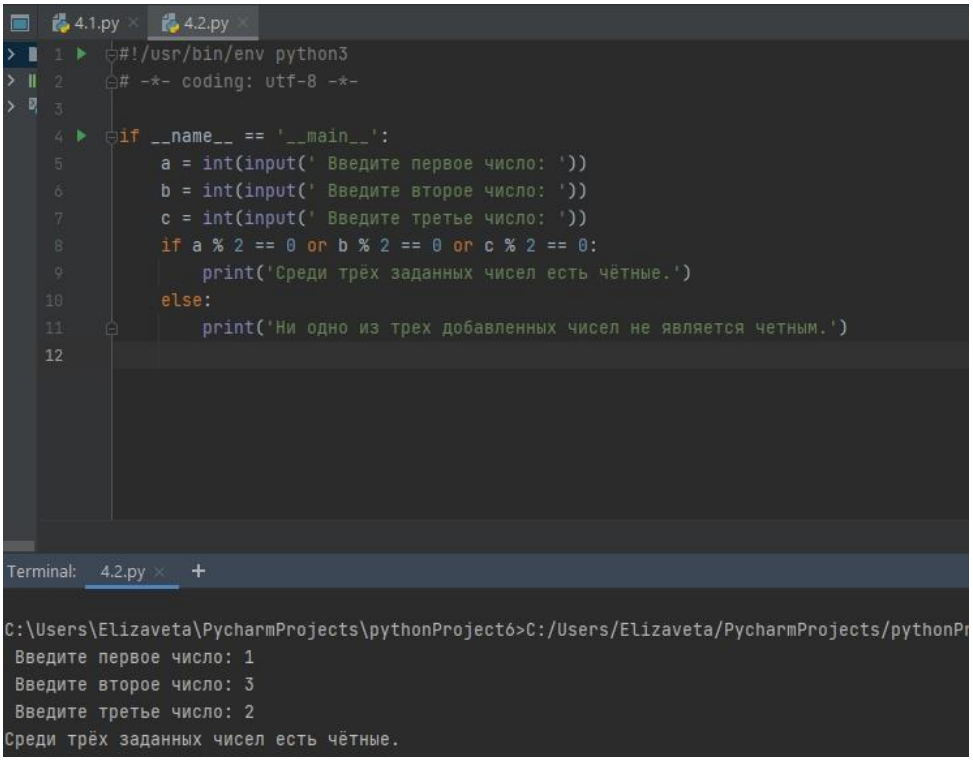

Рисунок 9. Индивидуальное задание №2

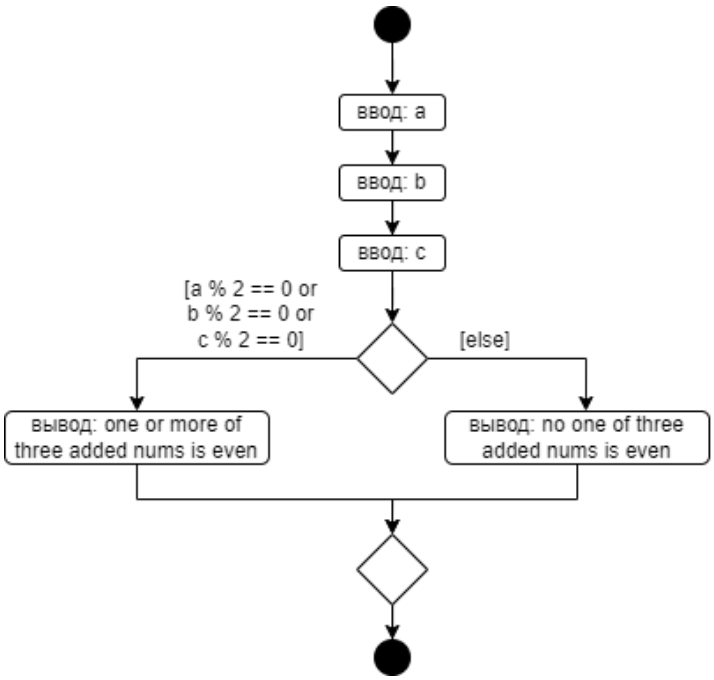

Рисунок 10. UML-диаграмма для 2 задания

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

if __name__ == '__main__':
    sum = 0
    for i in range(20, 100):
        if i % 3 == 0:
            sum = sum + i
    print(sum)
```

Рисунок 11. Индивидуальное задание №3

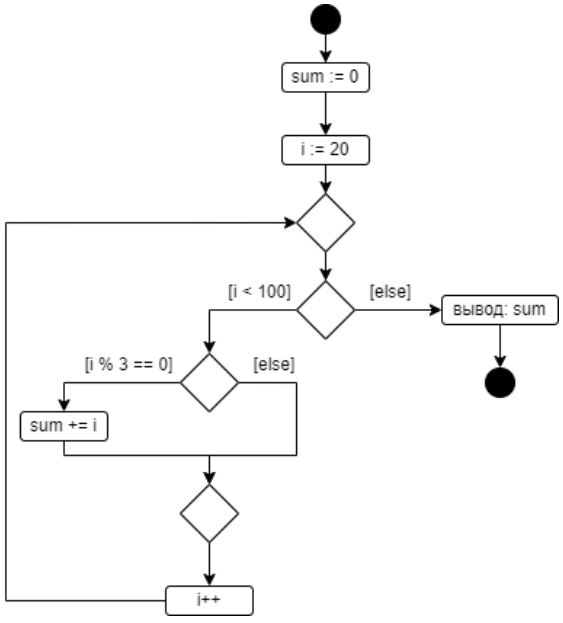

Рисунок 12. UML-диаграмма для 3 задания

```
4.1.py  4.2.py  4.3.py  povishennoe.py
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import math
import sys

EPS = 1e-10
if __name__ == '__main__':
    x = float(input("Value of x? "))
    if x == 0:
        print("Illegal value of x", file=sys.stderr)
        exit(1)
    n = float(input("Value of n -> "))
    q = (x / 2) ** n

    a = x
    S, k = a, 1
    while math.fabs(a) > EPS:
        a *= (4 * k * math.factorial(k + 1 + n) + 4 * math.factorial(k + 1 + n)) / \
             math.factorial(k - 1) * math.factorial(k + n) * (x ** 2)
```

Рисунок 13. Индивидуальное задание повышенной сложности

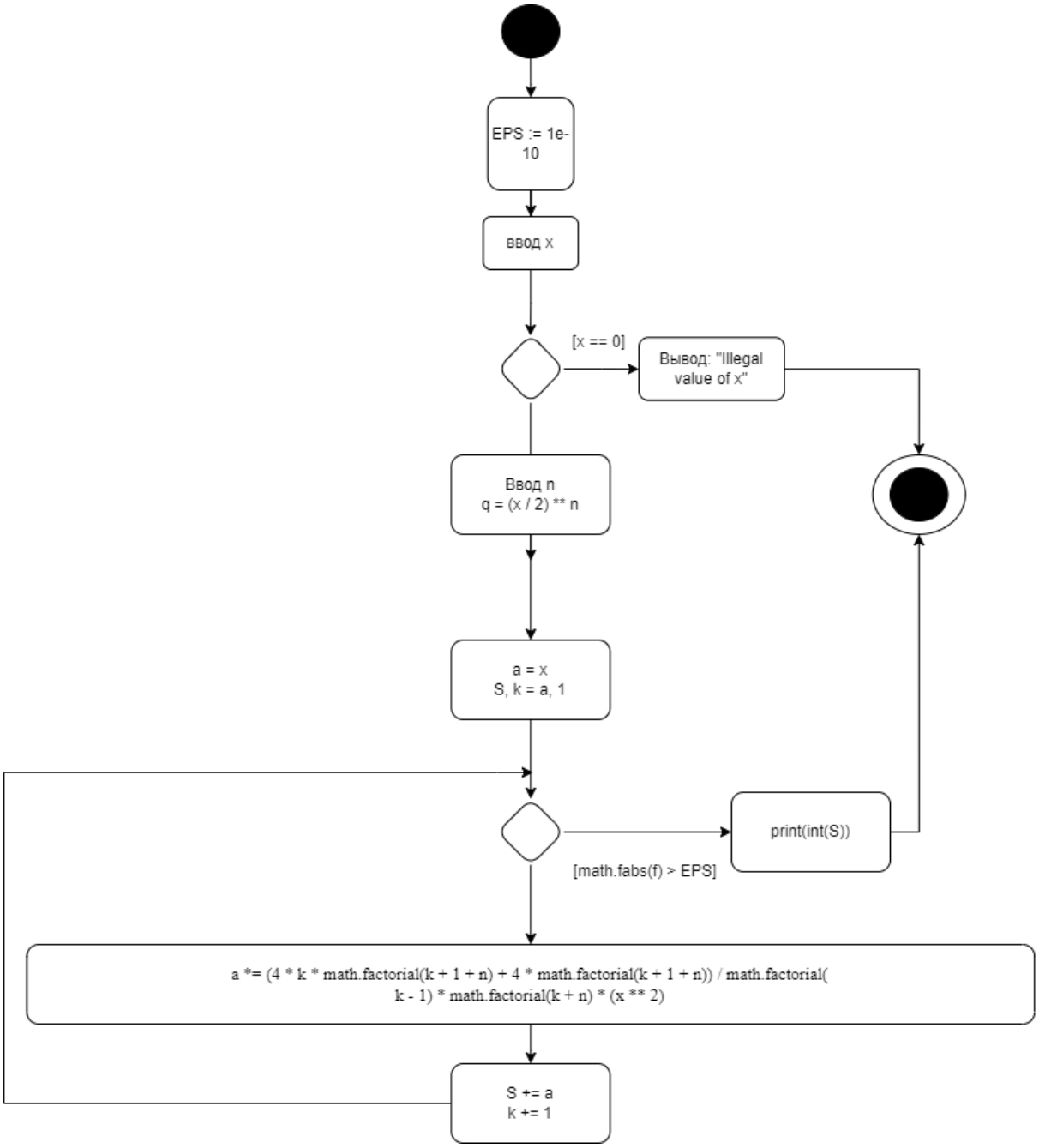

Рисунок 14. UML-диаграмма для задания повышенной сложности

4. Сделала коммит, выполнила слияние с веткой main, и запушила изменения в удаленный репозиторий.

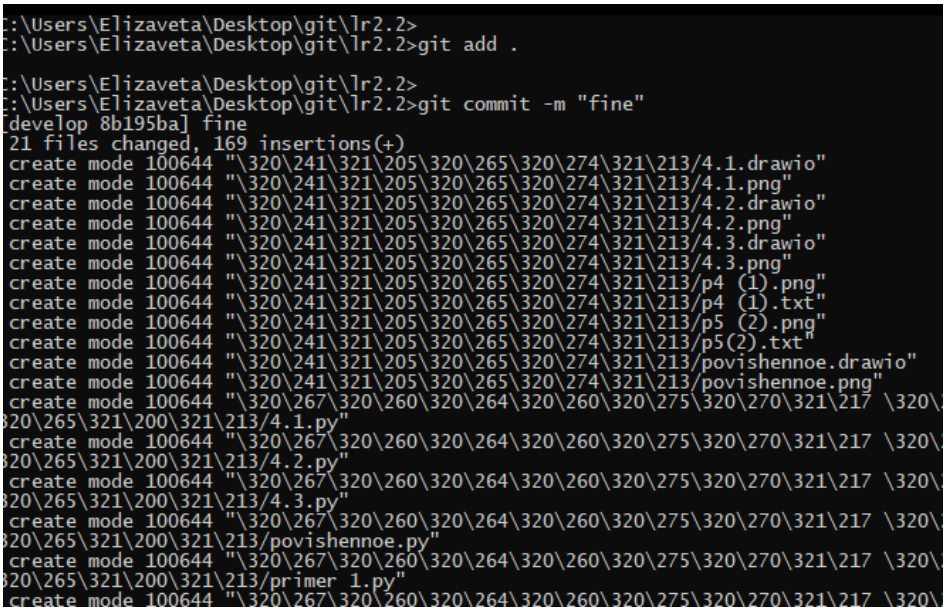

Рисунок 15. Фиксация и коммит файлов

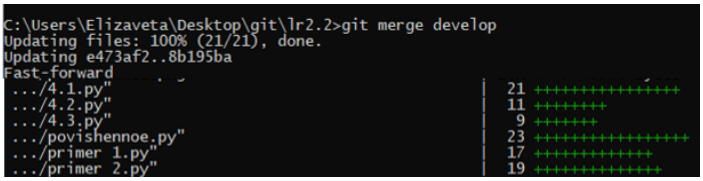

Рисунок 16. Слияние ветки develop с main

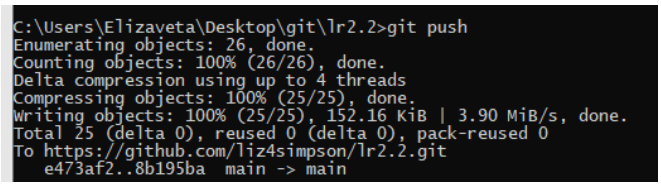

Рисунок 17. Отправка изменений на удаленный репозиторий

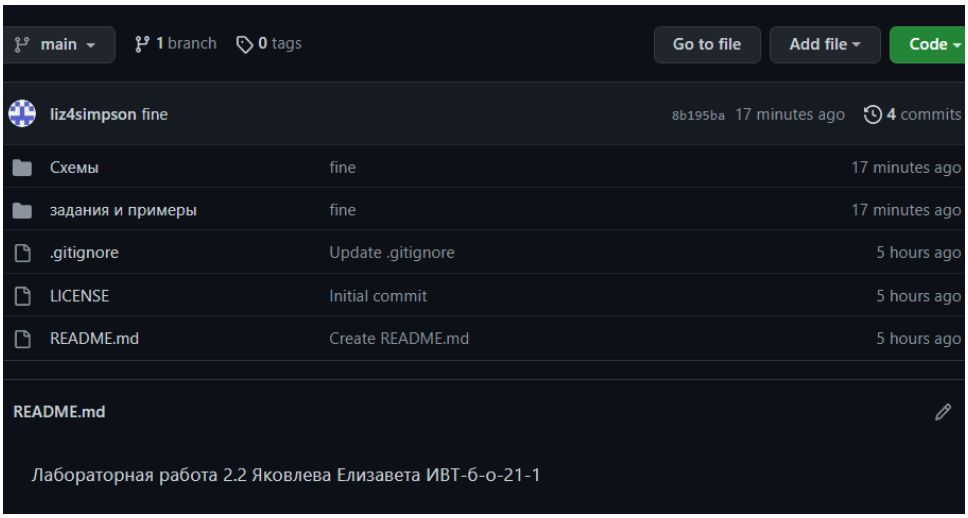

Рисунок 18. Изменения на удаленном репозитории

Ответы на 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. 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.

Вывод: в результате выполнения работы были приобретены навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры.