

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего
образования
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ
УНИВЕРСИТЕТ»**

**Кафедра
инфокоммуникаций
Институт цифрового
развития**

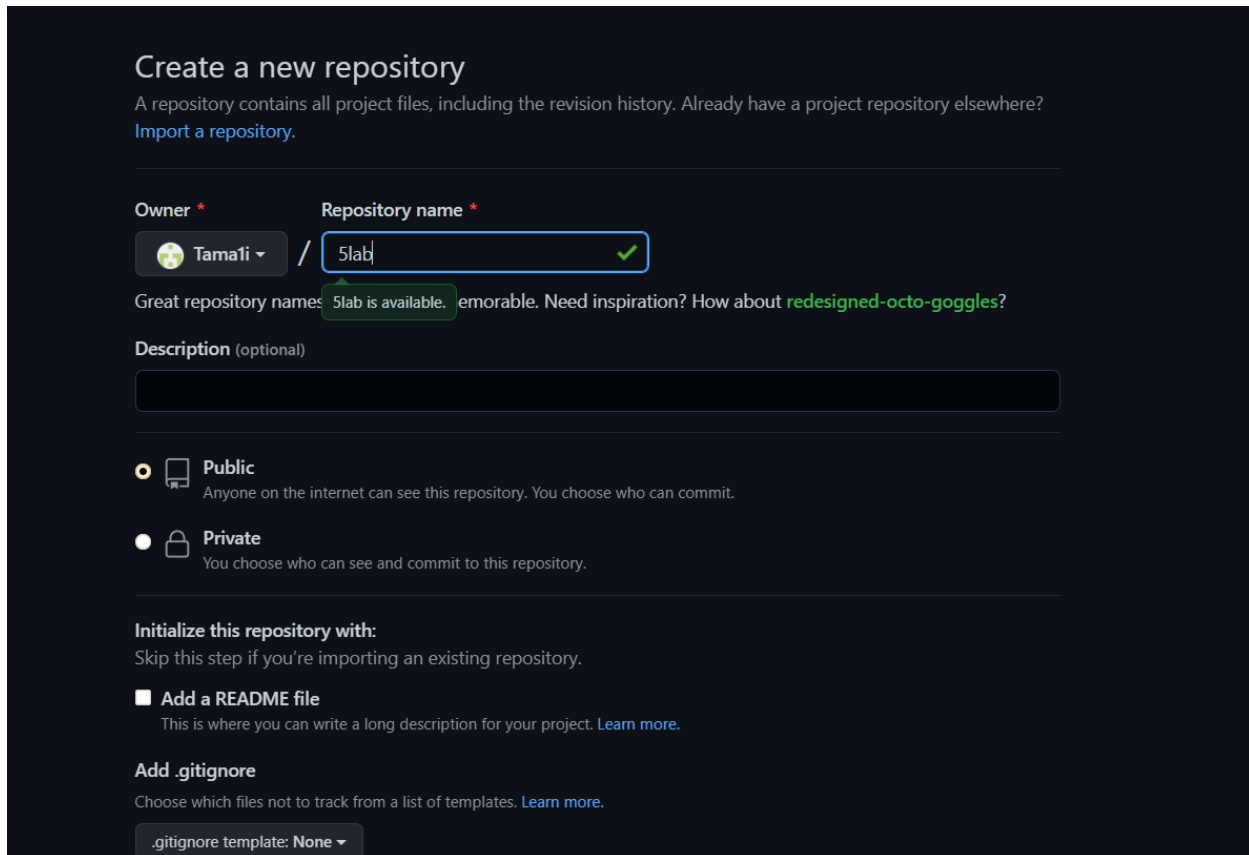
ОТЧЁТ
по лабораторной работе №5
Дисциплина: «Основы программной инженерии»
Тема: «Условные операторы и циклы в языке Python»

Выполнил: студент 2
курса группы Пиж-б-о-
21-1
Рязанцев Матвей
Денисович

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.

Выполнение работы



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * / Repository name * 5lab ✓

Great repository names 5lab is available. memorable. Need inspiration? How about redesigned-octo-goggles?

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: None

Рисунок 1 -создание репозитория

```
D:\gite>cd/d D:\gite\5lab
D:\gite\5lab>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? [D:/gite/5lab/.git/hooks]
D:\gite\5lab>
```

Рисунок 2 - Организация репозитория по модели ветвления git glow

```
D:\gite\51lab>git merge develop
Updating 2cc8ac7..5931771
Fast-forward
 idz/1.py | 10 ++++++++
 idz/2.py | 10 ++++++++
 idz/3.py | 6 +++++
 ob/PythonApplication1.py | 16 ++++++++
 ob/PythonApplication2.py | 19 ++++++++
 ob/PythonApplication3.py | 14 ++++++++
 ob/PythonApplication4.py | 20 ++++++++
 ob/PythonApplication5.py | 28 ++++++++
 pov.py | 28 ++++++++
 9 files changed, 151 insertions(+)
 create mode 100644 idz/1.py
 create mode 100644 idz/2.py
 create mode 100644 idz/3.py
 create mode 100644 ob/PythonApplication1.py
 create mode 100644 ob/PythonApplication2.py
 create mode 100644 ob/PythonApplication3.py
 create mode 100644 ob/PythonApplication4.py
 create mode 100644 ob/PythonApplication5.py
 create mode 100644 pov.py
```

Рисунок 2 – объединение веток

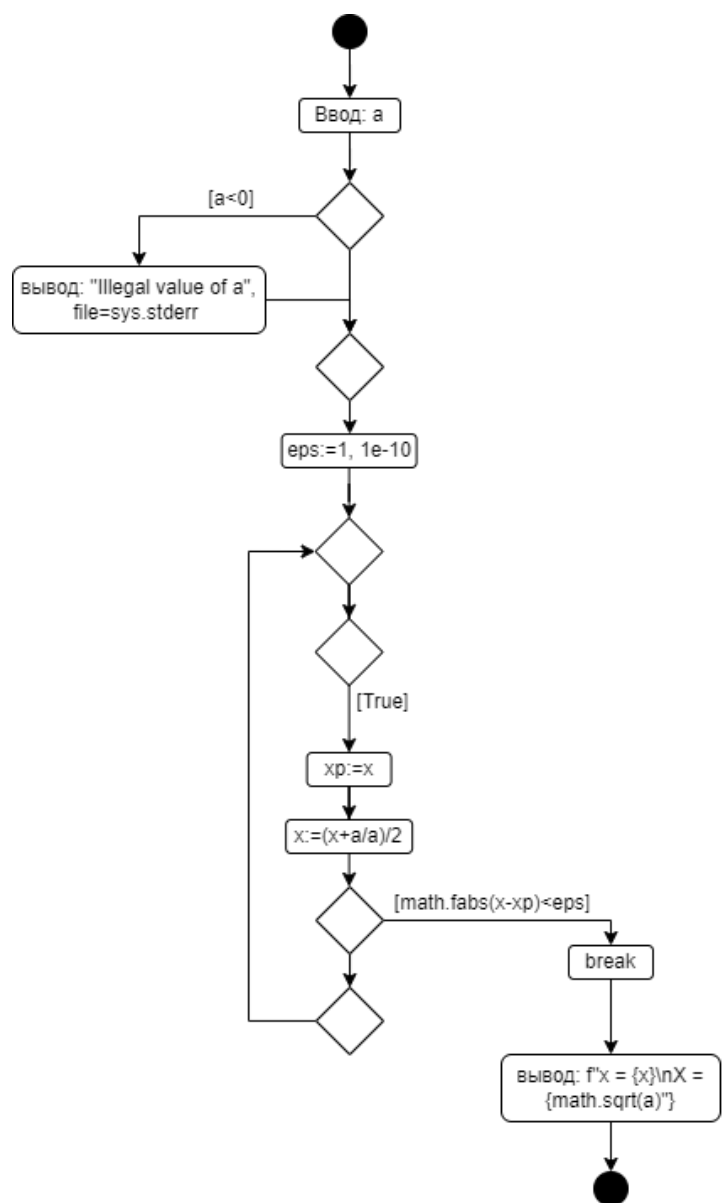


Рисунок 3 – UML-диаграмма программы для 4 примеры

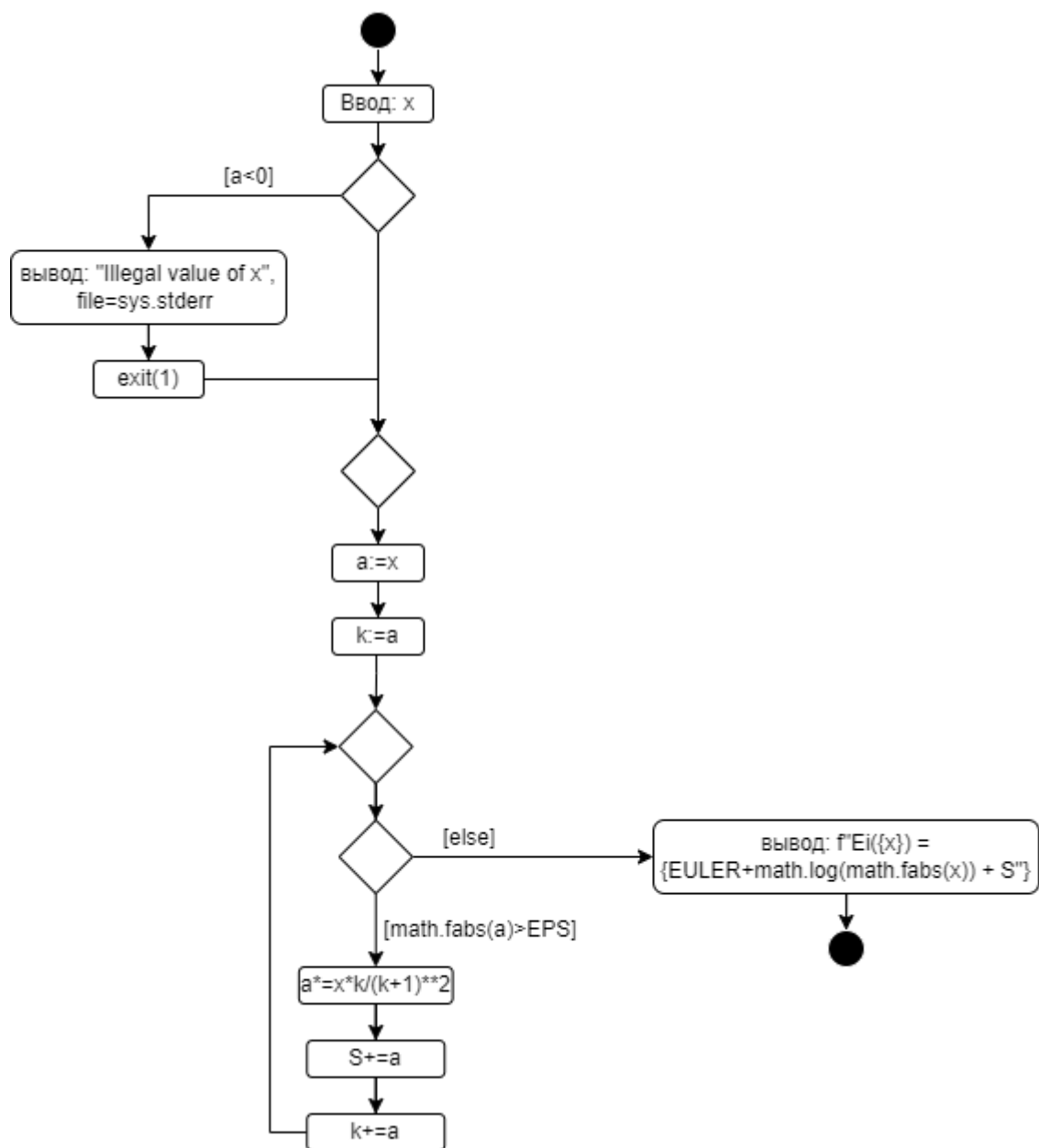


Рисунок 4 – UML-диаграмма для программы 5 примера

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

if __name__ == '__main__':
    a = int(input("enter kVt/h "))

    if (a <= 250):
        print("k oplate ", a * 7)
    elif a > 250 and a <= 300:
        print("k oplate ", a * 17)
    elif a > 300:
        print("k oplate ", a * 20)
```

Рисунок 5 – код программы

```
C:\Users\ryaza\AppData\Local\Programs\Python\Python39\python.exe
enter kVt/h 252
k oplate 4284
Press any key to continue . . .
```

Рисунок 6 - результат работы программы

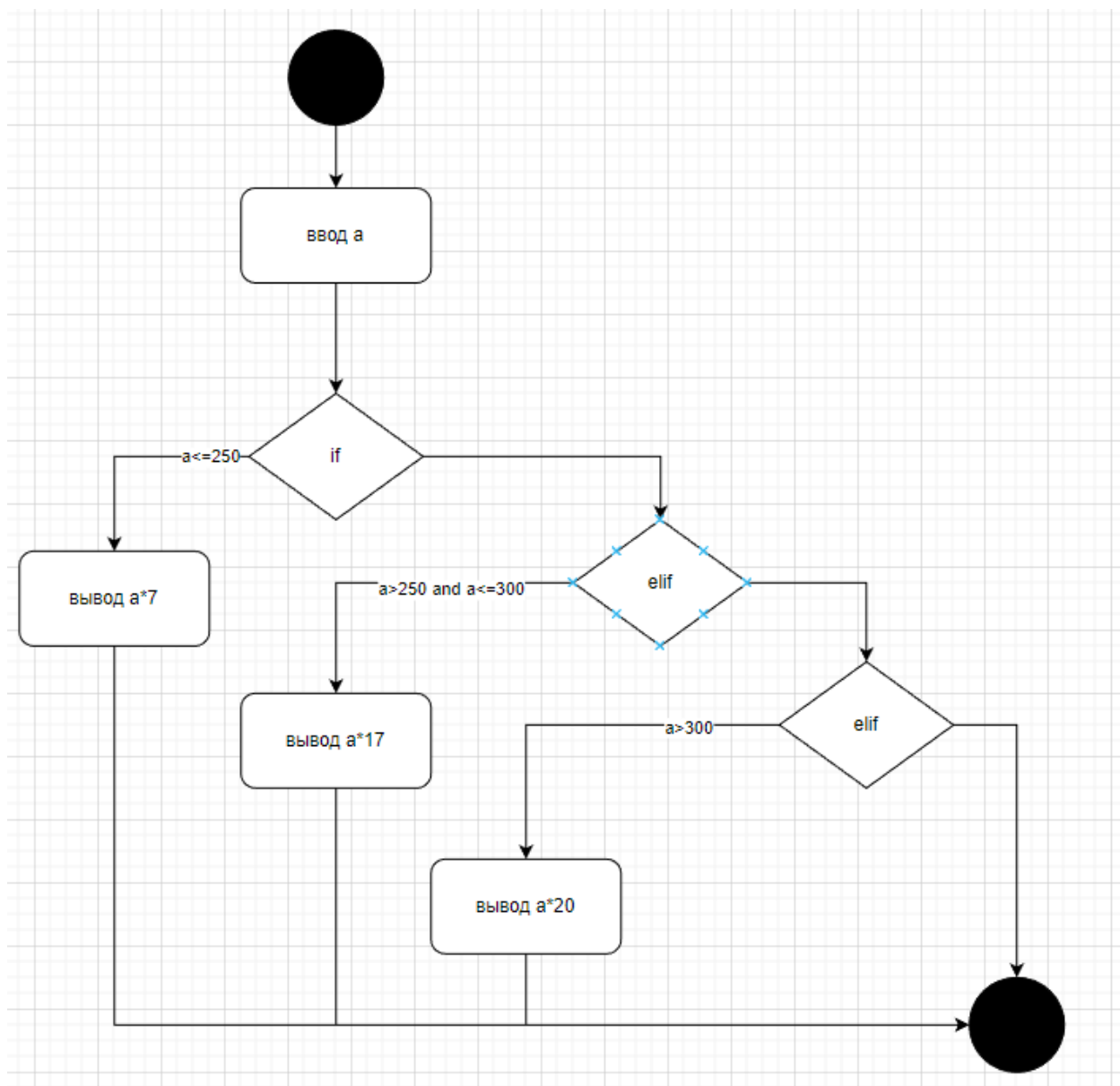


Рисунок 7 – Uml диаграмма

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

if __name__ == '__main__':
    a = int(input())
    b = int(input())
    c = int(input())

    if a % 2 == 0 or b % 2 == 0 or c % 2 == 0:
        print("ye")
    else:
        print("no")
```

Рисунок 8 – код программы

```
2
33
3
ye
Press any key to continue . . .
```

Рисунок 9 – результат работы программы

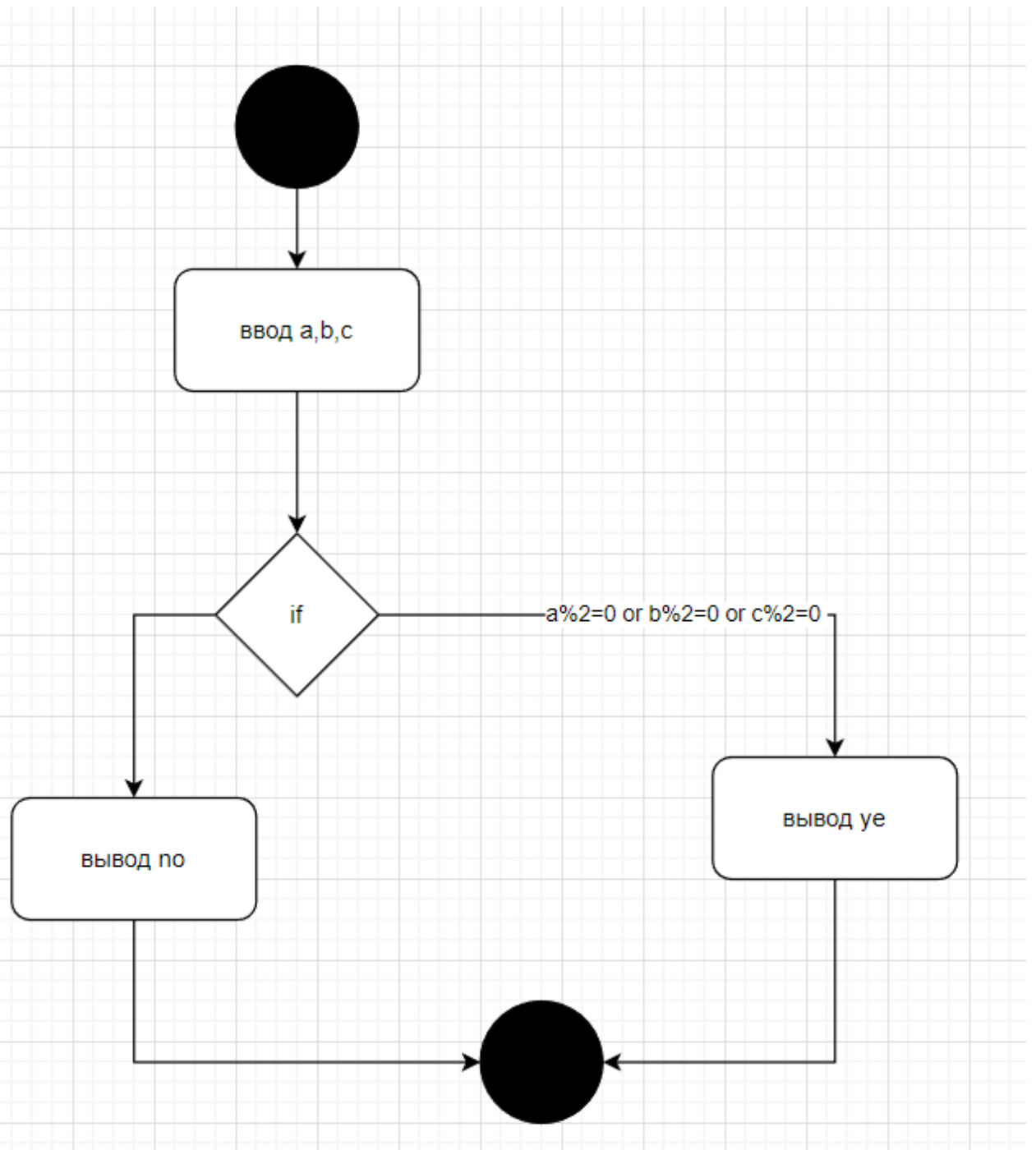


Рисунок 10 – uml диаграмма

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

if __name__ == '__main__':
    for i in range(1, 10):
        for j in range(1, 10):
            print(i * j, end = "\t")
        print("\n")
```

Рисунок 11 – код программы

C:\Users\ryaza\AppData\Local\Programs\Python\Python310\python.exe

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

Press any key to continue . . .

Рисунок 12 – результат работы программы

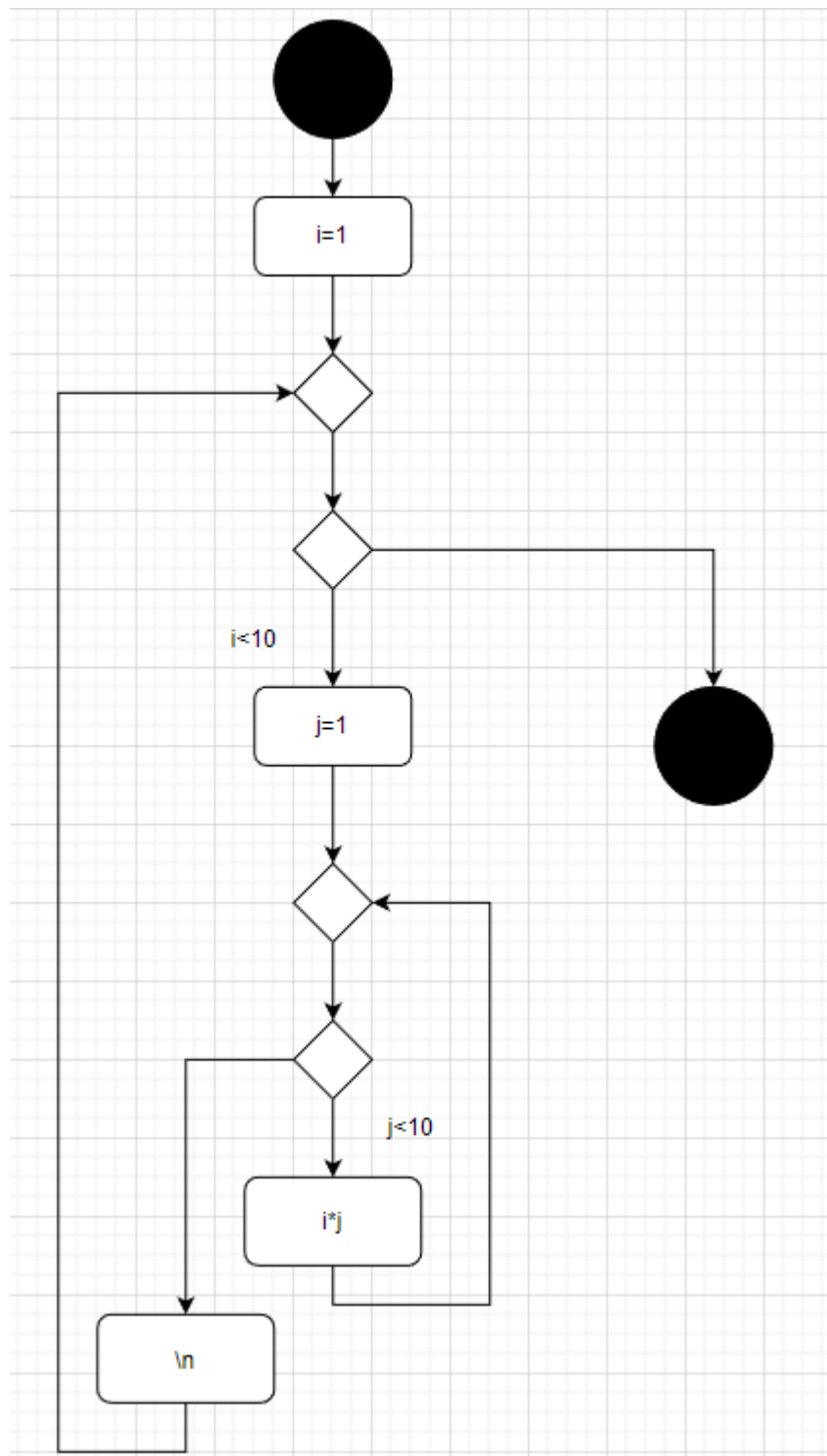


Рисунок 13 – uml диаграмма

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

import math
import sys

EULER = 0.5772156649015328606
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)
    a = x
    S, n = a, 1
    while math.fabs(a) > EPS:
        a *= -(((x**2) * ((2 * n) + 1)) / (((2 * n) + 3)**2 * ((2 * n) + 2)))
        S += a
        n += 1

    print(f"Ei({x}) = {EULER + math.log(math.fabs(x)) + S}")
```

Рисунок 15 – код программы идз

```
Value of x? 3
Ei(3.0) = 3.430055627614241
Press any key to continue . . .
```

Рисунок 16 – результат работы программы

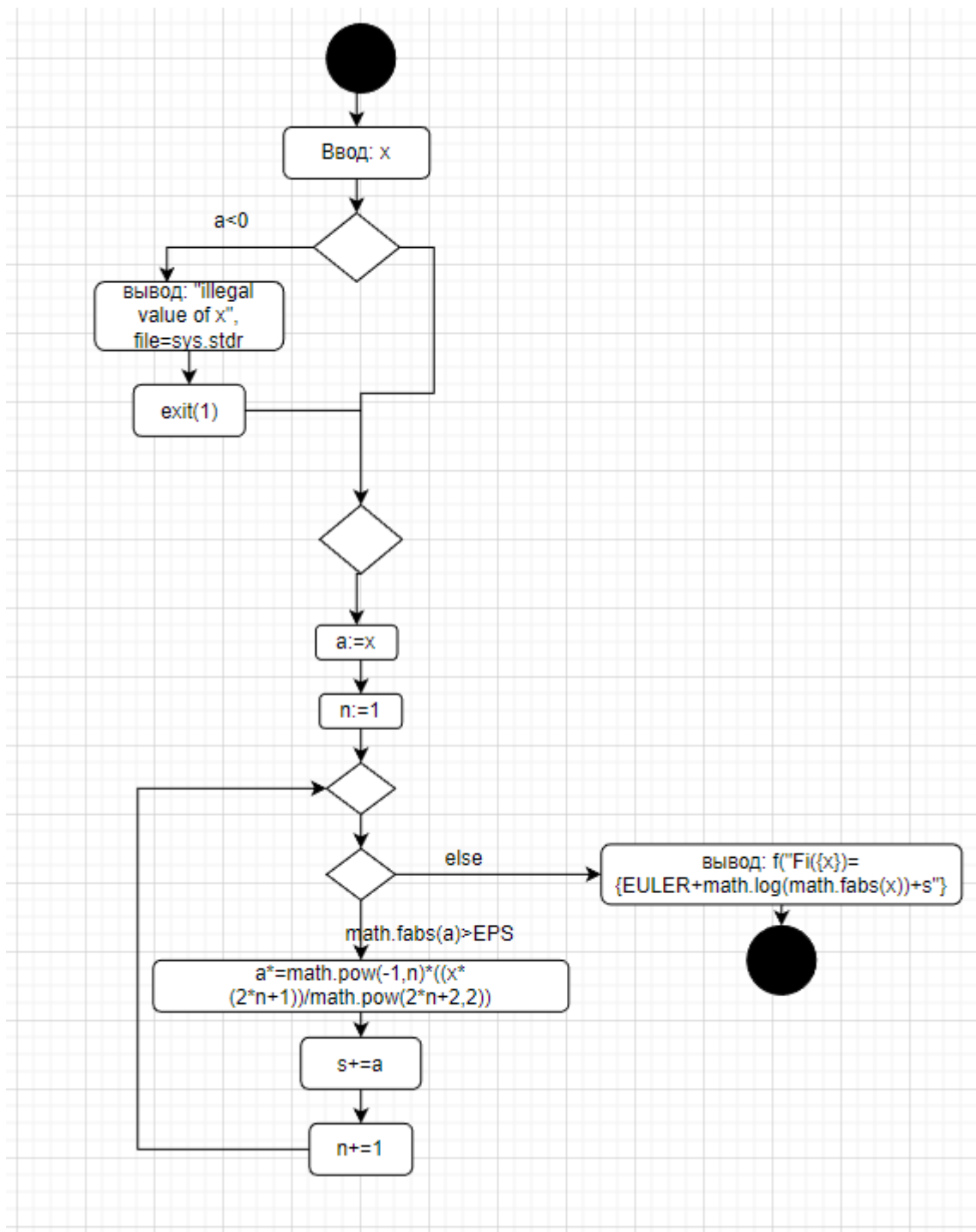


Рисунок 17 -Uml диаграмма

Вывод: в ходе выполнения работы исследован процесс установки и базовых возможностей языка Python

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.