

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙ-  
СКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**Основы кроссплатформенного программирования**

**Отчет по лабораторной работе №2.2**

**Условные операторы и циклы в языке Python**

Выполнил студент группы

ИВТ-б-о-21-1

Криворот В.Г. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил доцент

Кафедры инфокоммуникаций, старший  
преподаватель

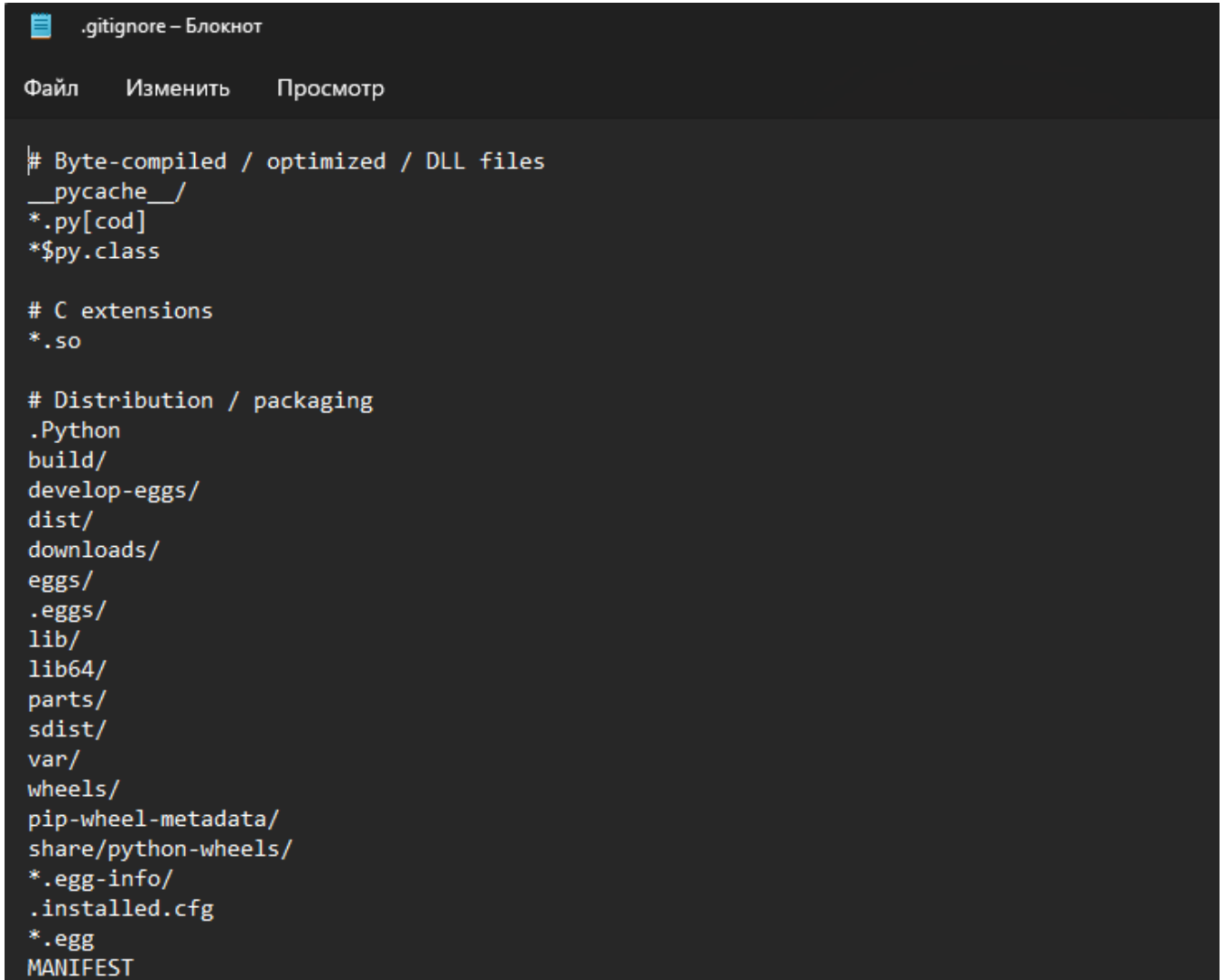
Воронкин Р.А.

\_\_\_\_\_  
(подпись)

Ставрополь 2022

## Тема: Условные операторы и циклы в языке Python

**№1.** Создать новый репозиторий и проработать примеры из лабораторной работы.

A screenshot of a text editor window titled ".gitignore – Блокнот". The window has a menu bar with "Файл", "Изменить", and "Просмотр". The main area contains the content of a .gitignore file, which lists various directories and file types to be ignored by Git. The text is as follows:

```
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
lib/
lib64/
parts/
sdist/
var/
wheels/
pip-wheel-metadata/
share/python-wheels/
*.egg-info/
.installed.cfg
*.egg
MANIFEST
```

Рисунок 1 – Изменённый файл .gitignore

```
c:\Users\Admin\Desktop\git\ForPrograms2.2>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/Admin/Desktop/git/ForPrograms2.2/.git/hooks]

c:\Users\Admin\Desktop\git\ForPrograms2.2>
```

Рисунок 2 – Организация репозитория согласно модели git-flow

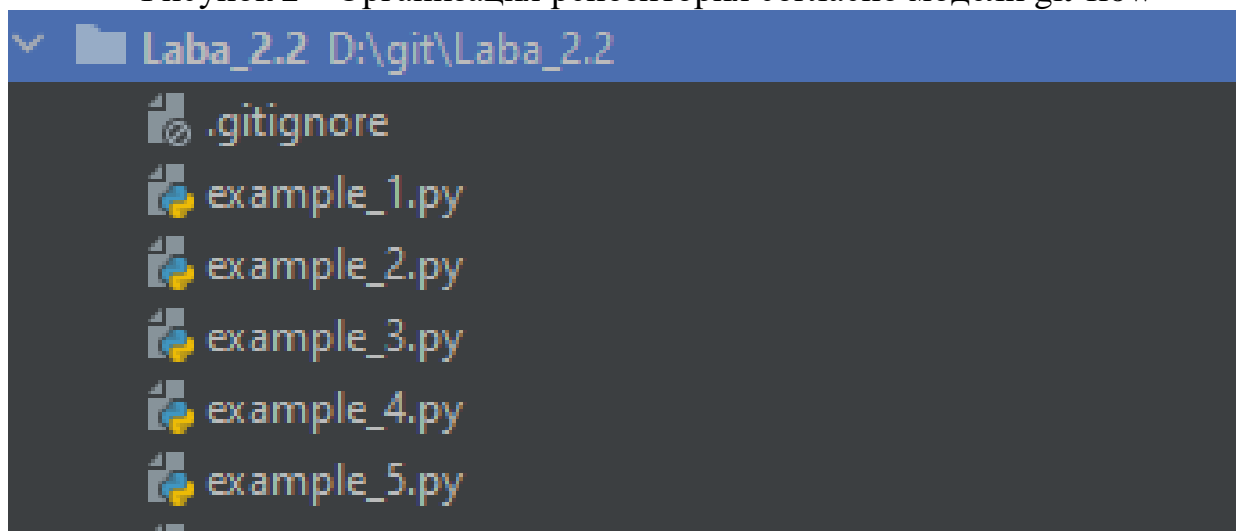


Рисунок 3 – Все проработанные примеры в лабораторной работе

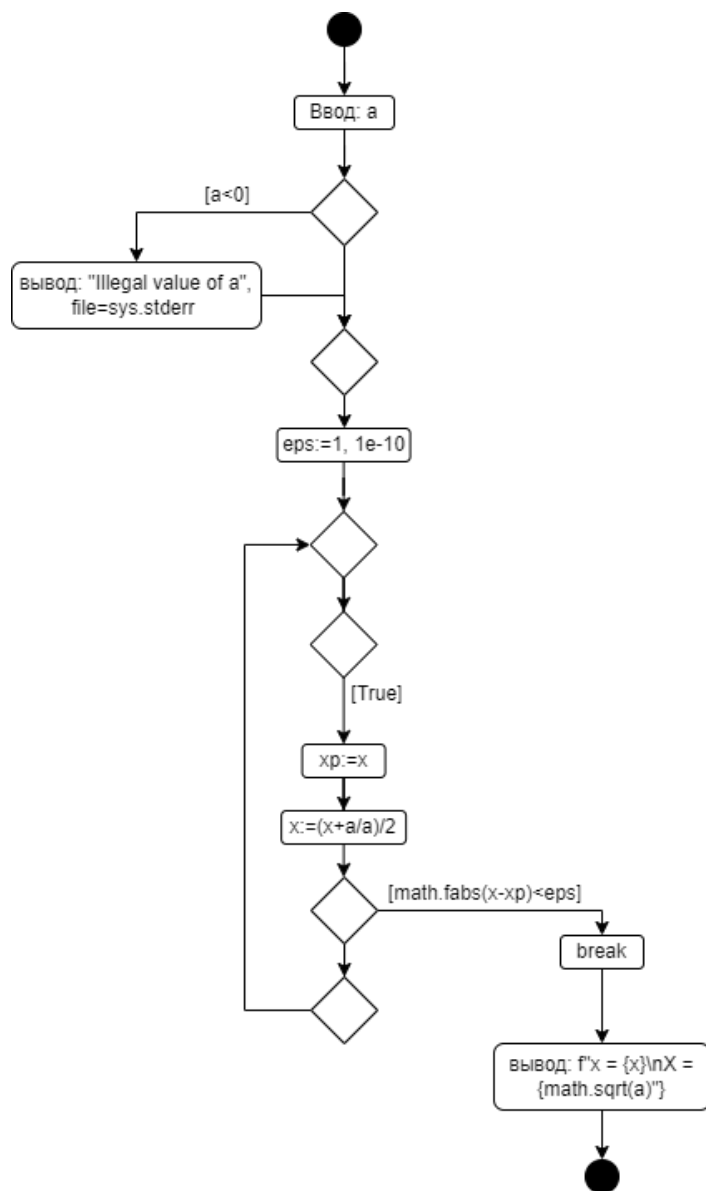


Рисунок 4 – UML-диаграмма для примера №4

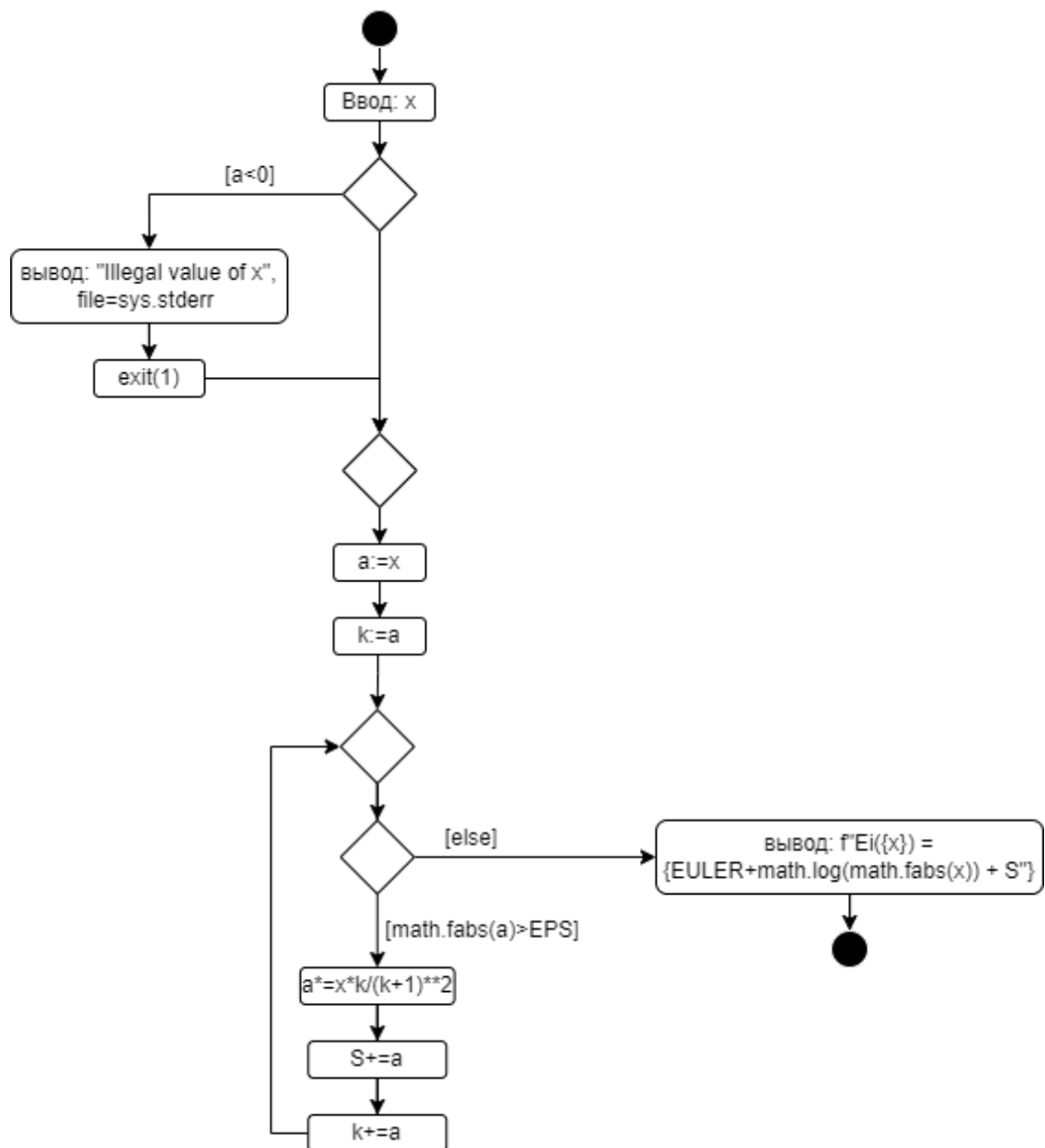


Рисунок 5 – UML-диаграмма для примера №5

**№2.** Выполнить индивидуальные задания и задание повышенной сложности согласно своему варианту(В-3). Построить UML-диаграммы написанных программ.

The image shows a Python IDE window titled 'Individual1.py'. The code is as follows:

```
1 """
2 3. Дано число m(1<=m<=7). Вывести на экран название дня недели, который соответствует
3 этому номеру.
4 """
5 import sys
6
7
8 if __name__ == '__main__':
9     m = int(input("Введите номер дня в недели: "))
10    if m == 1:
11        print("Понедельник")
12    elif m == 2:
13        print("Вторник")
14    elif m == 3:
```

Below the code editor is a 'Run' panel. It shows the command executed: `C:\Users\Admin\AppData\Local\Programs\Python\Python39\python.exe C:/Users/Admin/Desktop/Individual1.py`. The input provided is `Введите номер дня в недели: 2`, and the output is `Вторник`. The panel also indicates `Process finished with exit code 0`.

Рисунок 2.1 – Индивидуальное задание №1

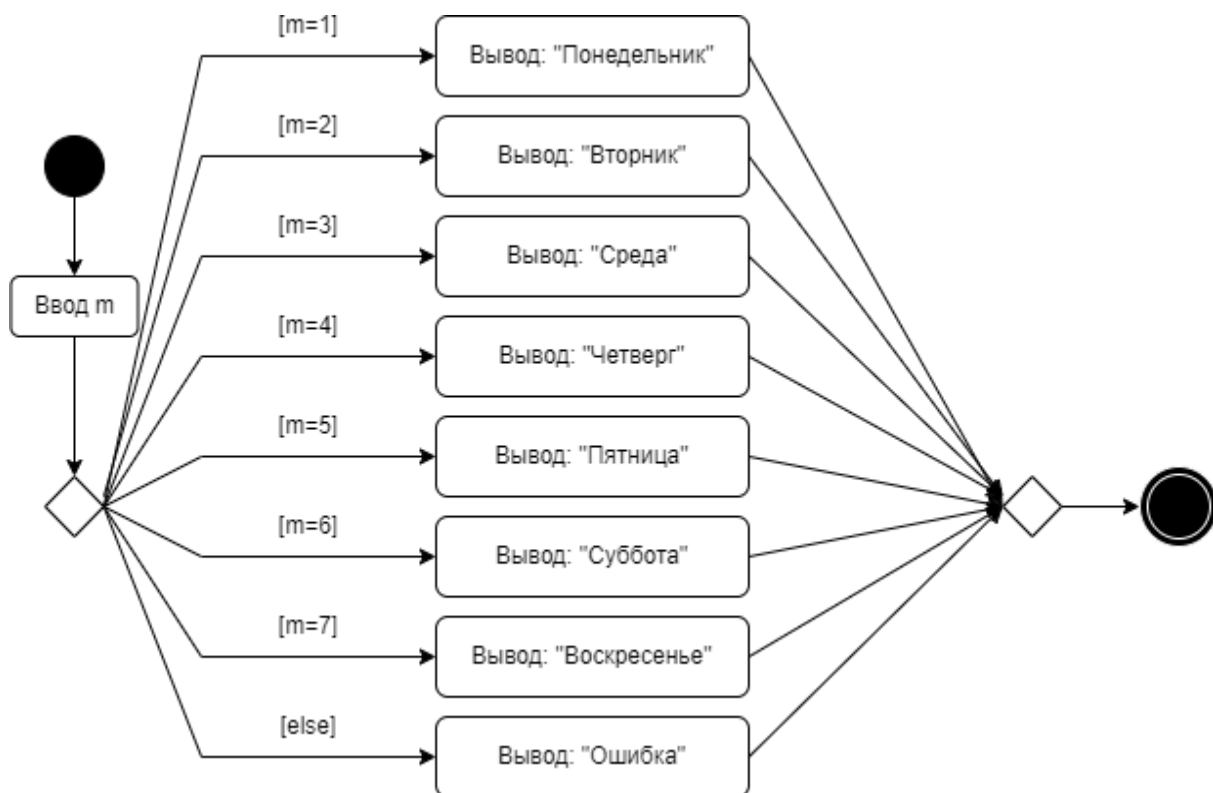


Рисунок 2.2 – UML-диаграмма для индивидуального задания №1

```
individual2.py x
1  """
2  Из трех действительных чисел a,b и c выбрать те, модули которых не меньше 4.
3  """
4
5
6  from math import fabs
7  a = int(input("Введите a: "))
8  b = int(input("Введите b: "))
9  c = int(input("Введите c: "))
10 if __name__ == '__main__':
11     if fabs(a) >= 4:
12         print("a=", fabs(a))
13     elif fabs(b) >= 4:
14         print("b=", fabs(b))
15     elif fabs(c) >= 4:
16         print("c=", fabs(c))
17     else:
18         print("Ошибка!")
19
Run: individual2 x
C:\Users\Admin\AppData\Local\Programs\Python\Python39\python.exe C:/Users/Ad
Введите a: -2
Введите b: 3
Введите c: -5
c= 5.0
Process finished with exit code 0
```

Рисунок 2.3 – Индивидуальное задание №2

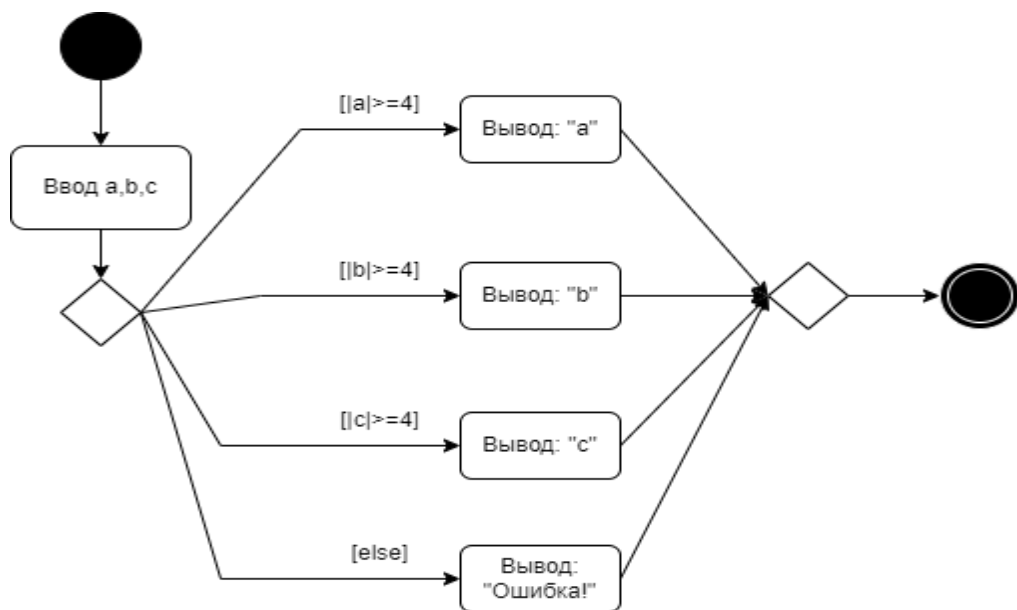


Рисунок 2.4 – UML-диаграмма для индивидуального задания №2



```

Individual3.py x
1 """
2 3. Начав тренировки, спортсмен пробежал 10 км. Каждый следующий день он увеличивал
3 дневную норму на 10% от нормы предыдущего дня. Какой суммарный путь пробежит
4 спортсмен за 7 дней?
5 """
6
7 dist = 0
8 day = 1
9 distance = 10
10 while day < 7:
11     distance = distance+((distance*10)/100)
12     dist += distance
13     day += 1
14 print(dist)
    while day < 7
Run: Individual3 x
C:\Users\Admin\AppData\Local\Programs\Python\Python39\python.exe C:/Users/Admin/Des
84.87171
Process finished with exit code 0

```

Рисунок 2.5 – Индивидуальное задание №3

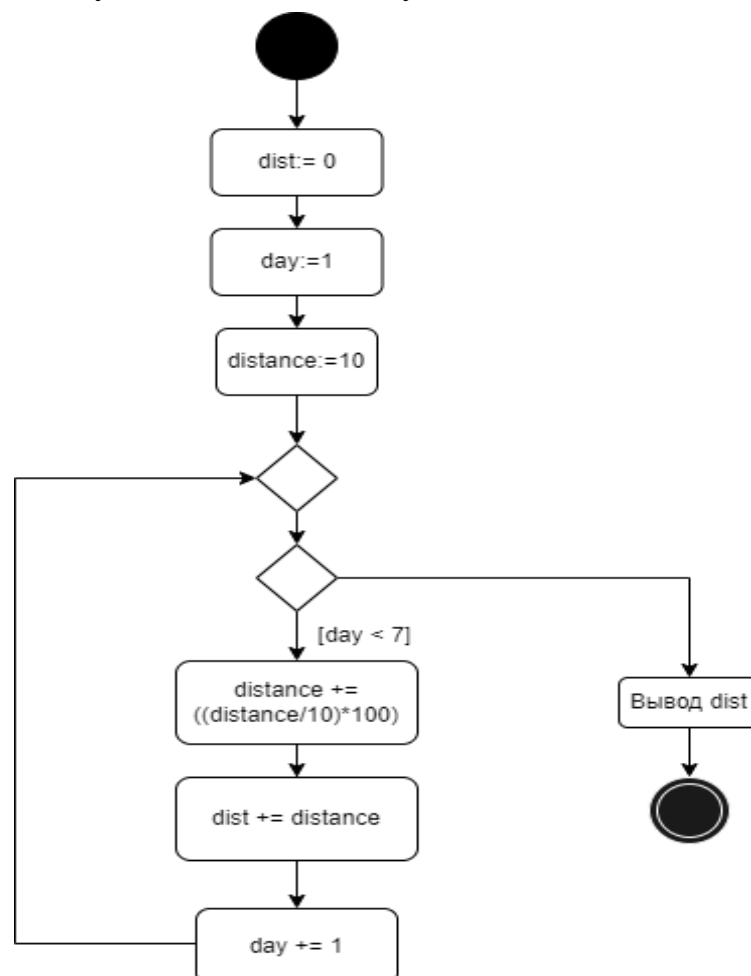


Рисунок 2.6 – UML-диаграмма для индивидуального задания №3

7. Функция Бесселя первого рода  $I_n(x)$ , значение  $n = 0, 1, 2, \dots$  также должно вводиться с клавиатуры

$$I_n(x) = \left(\frac{x}{2}\right)^n \sum_{k=0}^{\infty} \frac{(x^2/4)^k}{k!(k+n)!}.$$

Рисунок 2.7 – Задание повышенной сложности В-16

```
#!/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)
        S += a
        k += 1
    print(int(S))
```

Рисунок 2.8 – Задание повышенной сложности В-3

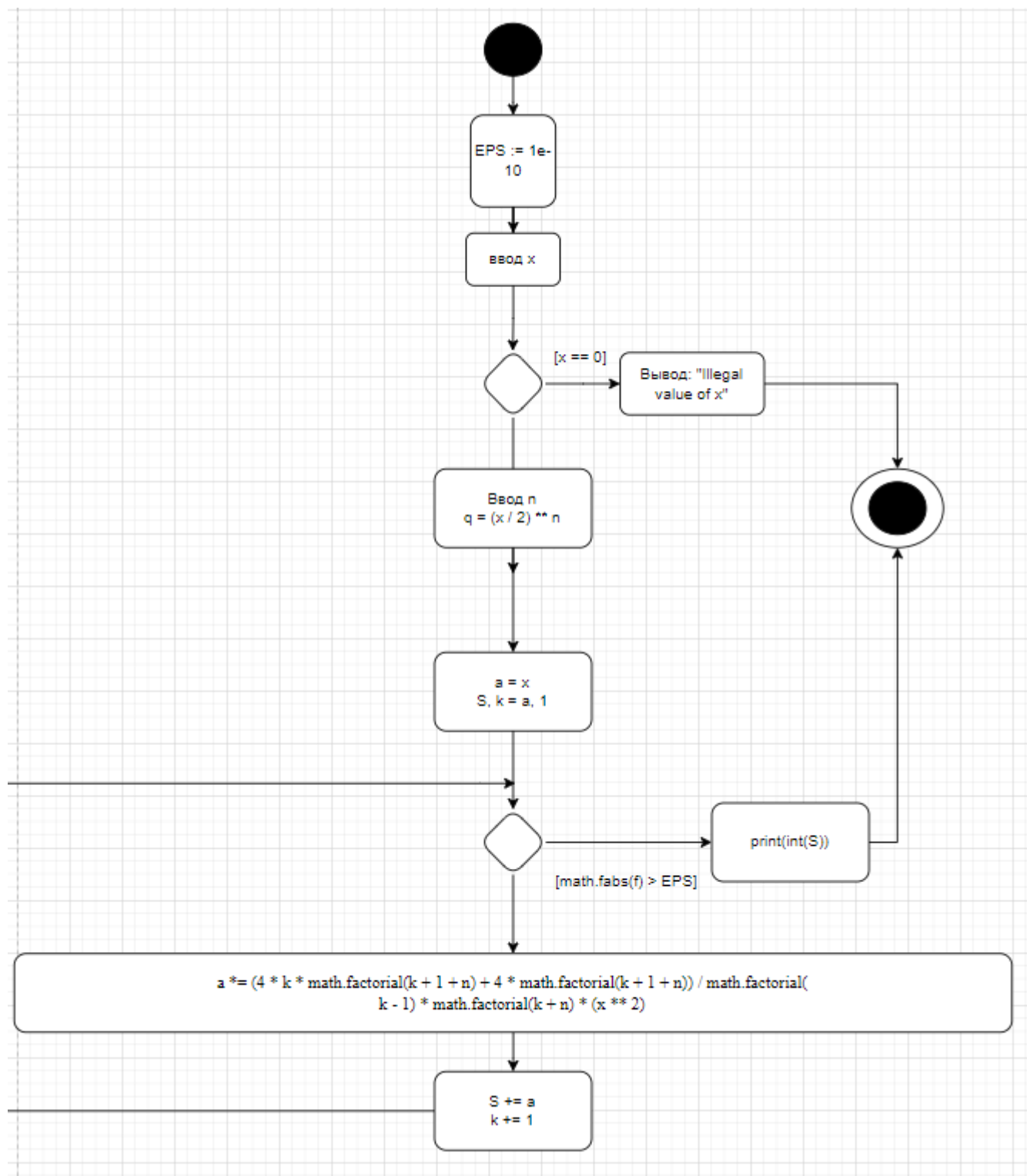


Рисунок 2.9 – UML-диаграмма для задачи повышенной сложности В-16

```

c:\Users\Admin\Desktop\git\ForPrograms2.2>git commit -m "1-Comm"
[develop 84f3db8] 1-Comm
24 files changed, 199 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/.name
create mode 100644 .idea/ForPrograms2.2.iml
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 Slognoe.py
create mode 100644 "UML-diag/1 \320\270\320\275\320\264\320\270\320\262\320\27
create mode 100644 "UML-diag/1 \320\270\320\275\320\264\320\270\320\262\320\27
create mode 100644 "UML-diag/\320\230\320\275\320\264\320\270\320\262\320\270\
create mode 100644 "UML-diag/\320\230\320\275\320\264\320\270\320\262\320\270\
create mode 100644 "UML-diag/\320\230\320\275\320\264\320\270\320\262\320\270\
create mode 100644 "UML-diag/\320\243\321\201\320\273\320\276\320\266\320\275\

create mode 100644 "UML-diag/\320\243\321\201\320\273\320\276\320\266\321\221\
create mode 100644 "UML-diag/\320\270\320\275\320\264\320\2622.png"
create mode 100644 individual/Individual1.py
create mode 100644 individual/Individual3.py
create mode 100644 individual/individual2.py
create mode 100644 primery/Primer1.py
create mode 100644 primery/Primer2.py
create mode 100644 primery/Primer3.py
create mode 100644 primery/Primer4.py
create mode 100644 primery/Primer5.py

c:\Users\Admin\Desktop\git\ForPrograms2.2>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

c:\Users\Admin\Desktop\git\ForPrograms2.2>git merge develop
Updating 2e71e72..84f3db8
Fast-forward
 .idea/.gitignore          | 3 +++
 .idea/.name               | 1 +
 .idea/ForPrograms2.2.iml  | 8 +++++++
 .idea/inspectionProfiles/profiles_settings.xml | 6 +++++

```

Рисунок 3.1 - Слияние ветки develop с веткой main

```

c:\Users\Admin\Desktop\git\ForPrograms2.2>git push
Enumerating objects: 32, done.
Counting objects: 100% (32/32), done.
Delta compression using up to 4 threads
Compressing objects: 100% (28/28), done.
Writing objects: 100% (31/31), 119.04 KiB | 7.94 MiB/s, done.
Total 31 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ItsMyLife1337/ForPrograms2.2.git
 2e71e72..84f3db8  main -> main

c:\Users\Admin\Desktop\git\ForPrograms2.2>

```

Рисунок 3.2 – Отправка изменений на удалённый репозиторий



MaksimUrusov test7

a7e7d82 7 hours ago

🕒 11 commits

📁 doc	test5	7 hours ago
📁 индивидуальные и повышенной	test4	7 hours ago
📁 примеры	test3	7 hours ago
📁 схемы	test7	7 hours ago
📄 .gitignore	Update .gitignore	8 hours ago
📄 LICENSE	Initial commit	8 hours ago
📄 README.md	Initial commit	8 hours ago

Рисунок 3.3 – Зафиксировал изменения на удалённом 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 представить 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.

**Вывод:** в результате выполнения работы были приобретены навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Также, освоены операторы языка Python версии 3.x `if` , `while` , `for` , `break` и `continue` , позволяющих реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.