

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

Кафедра инфокоммуникаций

**Отчет по лабораторной работе № 2.2
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-22-1

Душин Александр Владимирович.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

Тема: Условные операторы и циклы в языке Python.

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.

Ход выполнения работы:


1. Создать общедоступный репозиторий на GitHub с использованием лицензии MIT и язык программирования Python:

Create a new repository

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

Required fields are marked with an asterisk (*).

Owner *

 MrPlatynum ▾

Repository name *

ProgrammEngineering5

✔ ProgrammEngineering5 is available.

Great repository names are short and memorable. Need inspiration? How about [turbo-guide](#) ?

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:

☐ **Add a README file**

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создание общедоступного репозитория на GitHub с заданными настройками

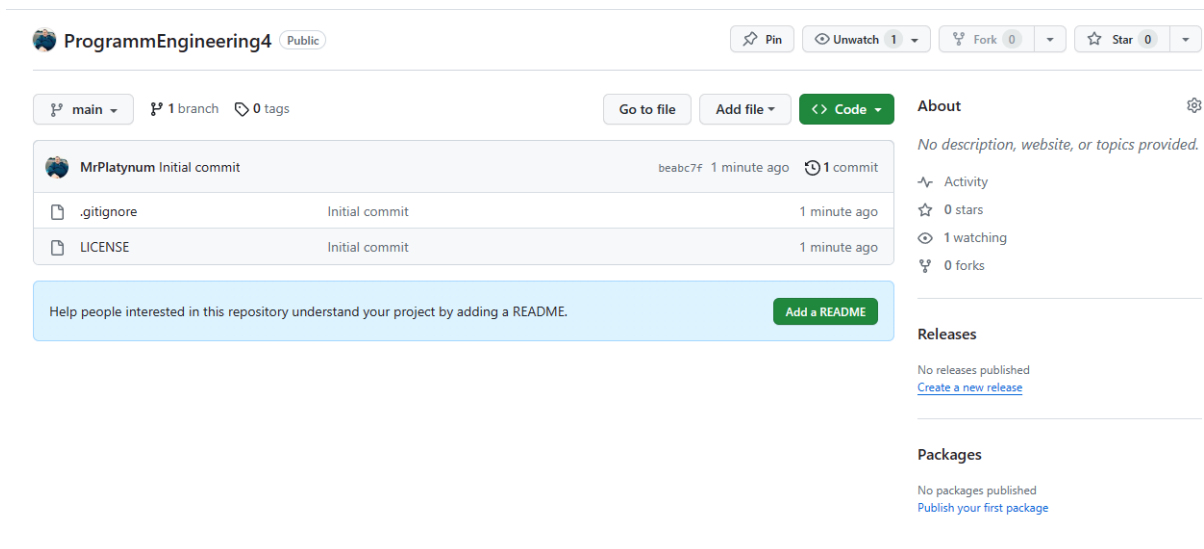


Рисунок 2 – Результат создания репозитория

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents
$ git clone https://github.com/MrPlatynum/ProgrammEngineering5.git
Cloning into 'ProgrammEngineering5'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Рисунок 3 – Клонирование созданного репозитория на локальный компьютер

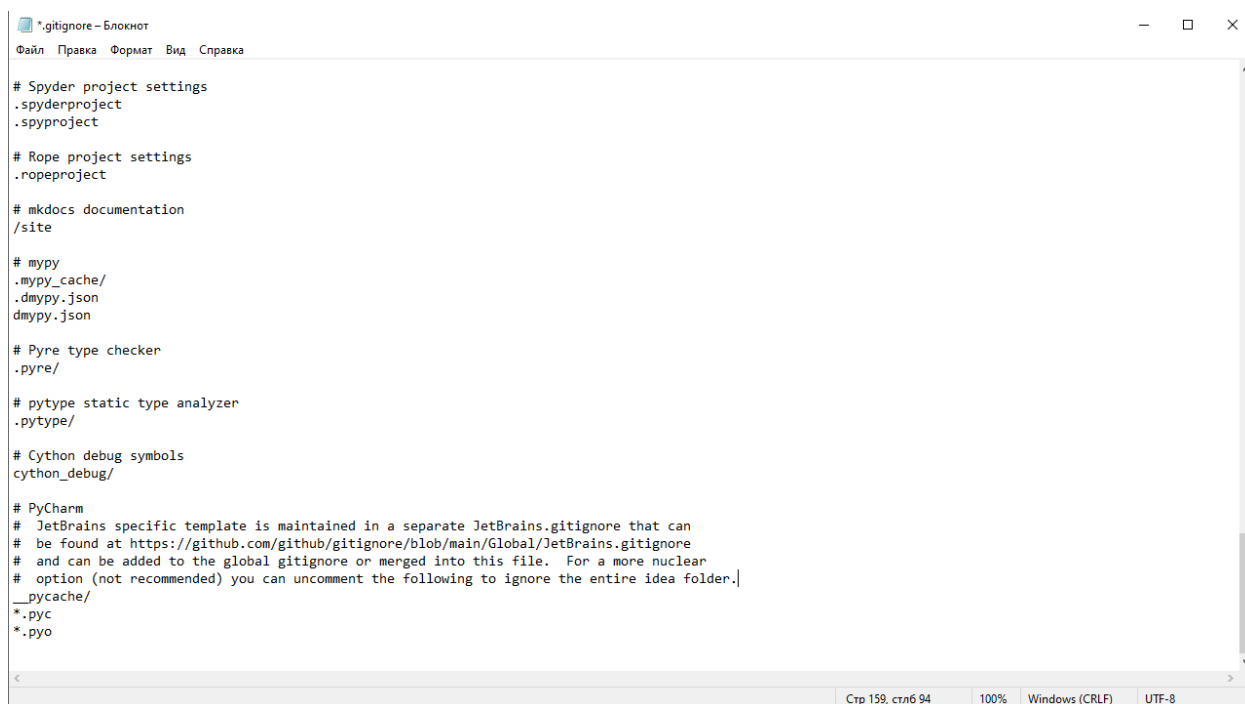


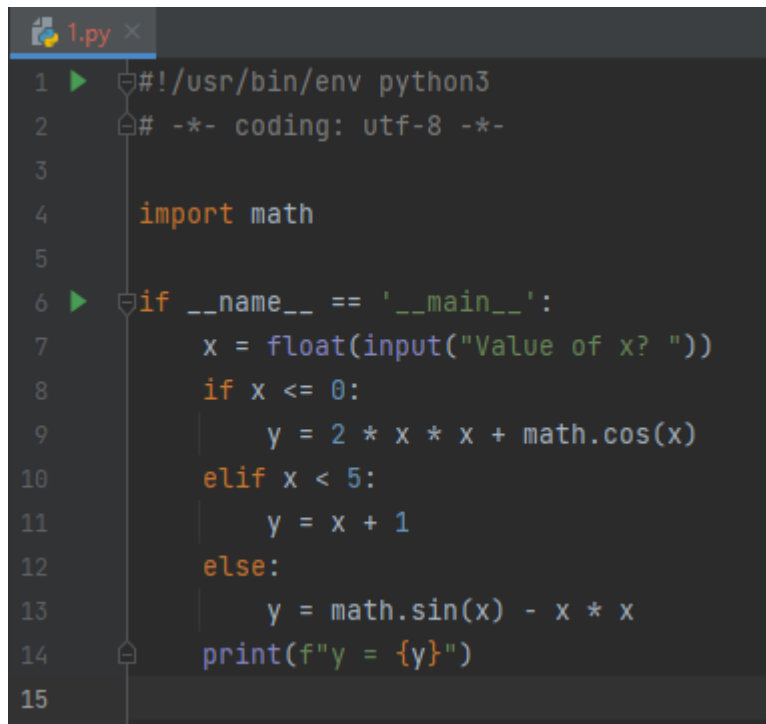
Рисунок 4 – файл .gitignore

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering5 (main)
$ git branch develop

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering5 (main)
$ git checkout develop
Switched to branch 'develop'
```

Рисунок 5 – организация репозитория в соответствии с моделью ветвления git flow

2. Проработать примеры лабораторной работы, оформляя код согласно PEP-8. Построить UML-диаграммы деятельности для 4 и 5 заданий:



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5
6  if __name__ == '__main__':
7      x = float(input("Value of x? "))
8      if x <= 0:
9          y = 2 * x * x + math.cos(x)
10     elif x < 5:
11         y = x + 1
12     else:
13         y = math.sin(x) - x * x
14     print(f"y = {y}")
15
```

Рисунок 6 – Нахождение значения функции (задание №1)

```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering5/1.py
Value of x? 5
y = -25.95892427466314
```

Рисунок 6 – Вывод программы (задание №1)

```
2.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6 ▶ if __name__ == '__main__':
7     n = int(input("Введите номер месяца: "))
8
9     if n == 1 or n == 2 or n == 12:
10         print("Зима")
11     elif n == 3 or n == 4 or n == 5:
12         print("Весна")
13     elif n == 6 or n == 7 or n == 8:
14         print("Лето")
15     elif n == 9 or n == 10 or n == 11:
16         print("Осень")
17     else:
18         print("Ошибка!", file=sys.stderr)
19         exit(1)
20
```

Рисунок 7 – Нахождение времени года по номеру месяца (задание №2)

```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering5/2.py
Введите номер месяца: 3
Весна
```

Рисунок 8 – Вывод времени года по номеру месяца (задание №2)

```

1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      import math
5
6  ▶  if __name__ == '__main__':
7      n = int(input("Value of n? "))
8      x = float(input("Value of x? "))
9      S = 0.0
10
11     for k in range(1, n + 1):
12         a = math.log(k * x) / (k * k)
13         S += a
14
15     print(f"S = {S}")
16

```

Рисунок 9 – Вычисление конечной суммы (задание №3)

```

"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering5/3.py
Value of n? 4
Value of x? 5
S = 2.6732119195688706

```

Рисунок 10 – Вывод конечной суммы (задание №3)

```
4.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import math
5 import sys
6
7 ▶ if __name__ == '__main__':
8     a = float(input("Value of a? "))
9     if a < 0:
10         print("Illegal value of a", file=sys.stderr)
11         exit(1)
12
13     x, eps = 1, 1e-10
14     while True:
15         xp = x
16         x = (x + a / x) / 2
17         if math.fabs(x - xp) < eps:
18             break
19
20     print(f"x = {x}\nX = {math.sqrt(a)}")
21
```

Рисунок 11 – Нахождение квадратного корня числа и его сравнение с результатом метода sqrt() стандартной библиотеки Python (задание №4)

```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering5/4.py
Value of a? 3
x = 1.7320508075688772
X = 1.7320508075688772
```

Рисунок 12 – Вывод программы (задание №4)

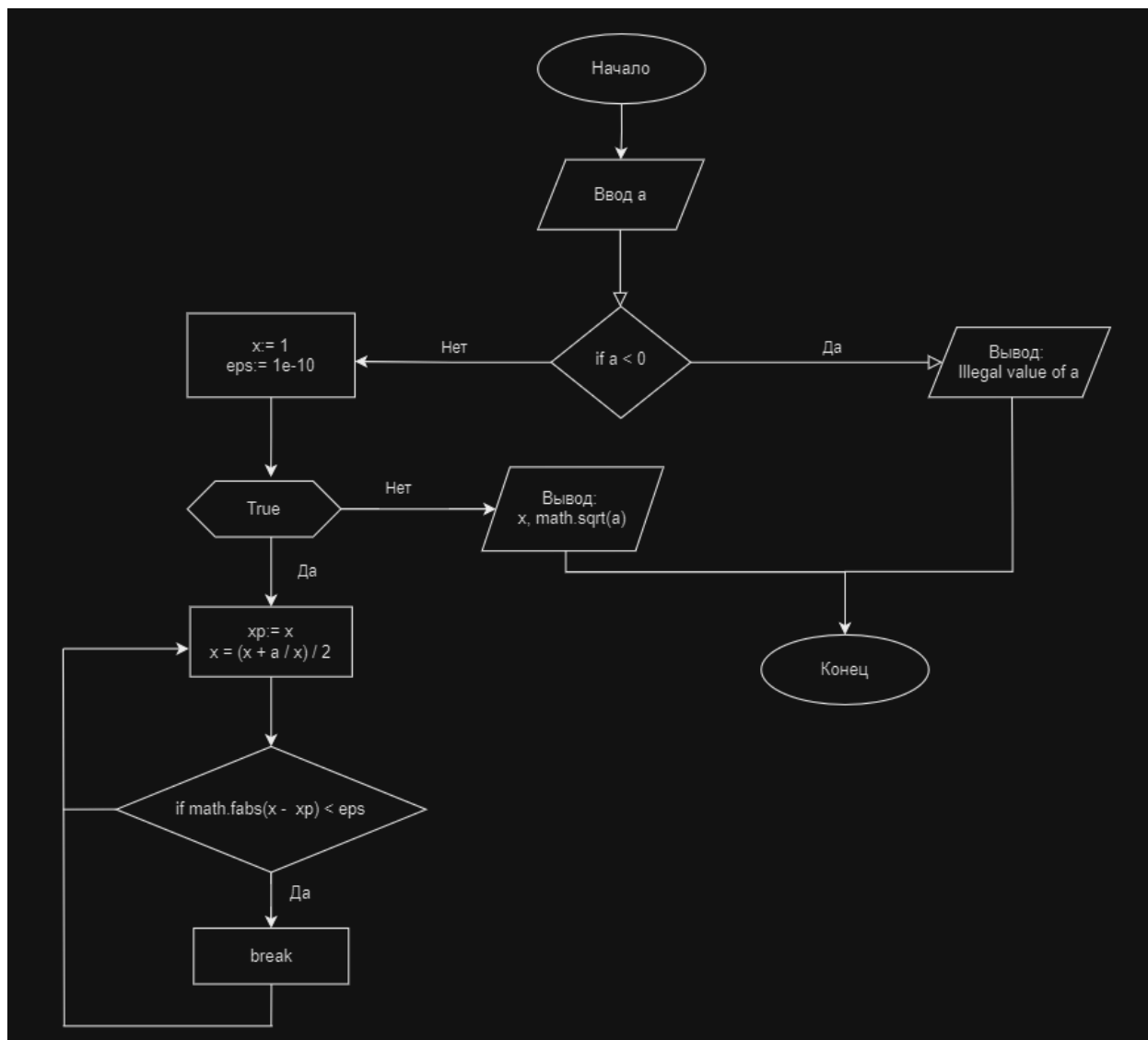


Рисунок 13 – Диаграмма для программы задания №4

```
5.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import math
5 import sys
6
7 # Постоянная Эйлера.
8 EULER = 0.5772156649015328606
9 # Точность вычислений.
10 EPS = 1e-10
11
12 ▶ if __name__ == '__main__':
13     x = float(input("Value of x? "))
14
15     if x == 0:
16         print("Illegal value of x", file=sys.stderr)
17         exit(1)
18
19     a = x
20     S, k = a, 1
21
22     # Найти сумму членов ряда.
23     while math.fabs(a) > EPS:
24         a *= x * k / (k + 1) ** 2
25         S += a
26         k += 1
27
28     # Вывести значение функции.
29     print(f"Ei({x}) = {EULER + math.log(math.fabs(x)) + S}")
30
```

Рисунок 14 – Вычисление значения специальной (интегральной показательной) функции (задание №5)

```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering5/5.py
Value of x? 4
Ei(4.0) = 19.63087447005282
```

Рисунок 15 – Вывод программы (задание №5)

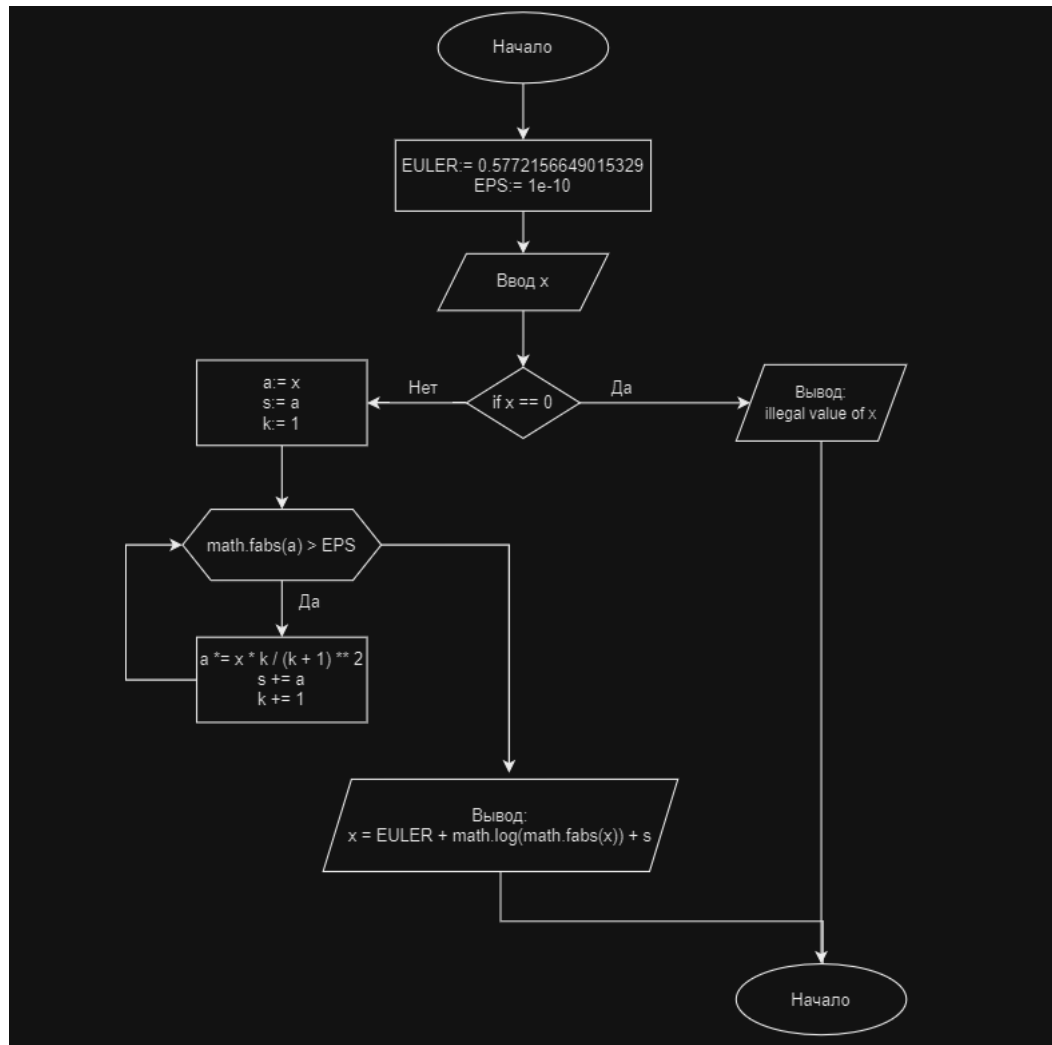


Рисунок 16 – Диаграмма для программы задания №5

3. Выполним индивидуальные задания и построим UML-диаграммы для них:

```
individual1.py ×
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == "__main__":
5     words = ["ноль", "один", "два", "три", "четыре", "пять", "шесть", "семь", "восемь"]
6
7     c = int(input("Введите целое число C, такое что |C| < 9: "))
8
9     if c < 0:
10         sign = "отрицательное"
11         c = abs(c)
12     else:
13         sign = "положительное"
14
15     word_form = words[c]
16     print(f"Словесная форма числа {c}: {sign} {word_form}")
17
```

Рисунок 17 –Нахождение словесной формы числа (задание №1)

```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering5/individual1.py
Введите целое число C, такое что |C| < 9: 4
Словесная форма числа 4: положительное четыре
```

Рисунок 18 – Вывод программы (задание №1)

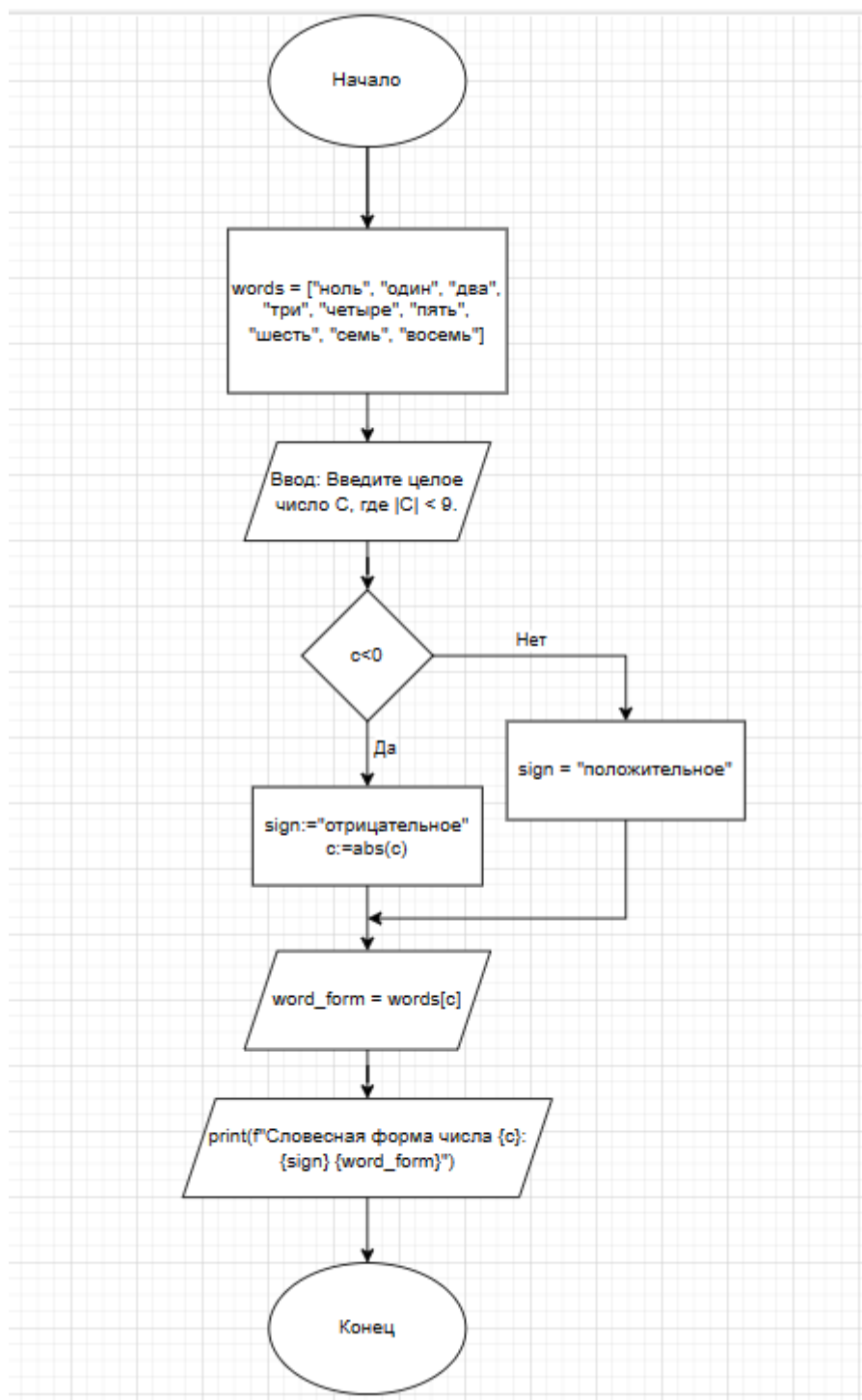


Рисунок 20 – Диаграмма для программы индивидуального задания №1

```
individual2.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 a = float(input("Введите коэффициент a: "))
5 b = float(input("Введите коэффициент b: "))
6 c = float(input("Введите коэффициент c: "))
7
8 discriminant = b ** 2 - 4 * a * c
9
10 if discriminant > 0:
11     x1 = (-b + discriminant ** 0.5) / (2 * a)
12     x2 = (-b - discriminant ** 0.5) / (2 * a)
13     if a > 0:
14         print(f"x < {x1} или x > {x2}")
15     else:
16         print(f"{x1} < x < {x2}")
17 elif discriminant == 0:
18     x = -b / (2 * a)
19     if a > 0:
20         print(f"x = {x}")
21     else:
22         print("Нет действительных корней")
23 else:
24     print("Пустое множество")
25
```

Рисунок 21 – Исследование квадратное неравенство $ax^2 + bx + c > 0$ (задание №2)

```
"C:\Program Files\Python312\python.exe" C:/Users/Ale.
Введите коэффициент a: 3
Введите коэффициент b: 5
Введите коэффициент c: 1
x < -0.2324081207560018 или x > -1.434258545910665
```

Рисунок 22 – Запуск программы

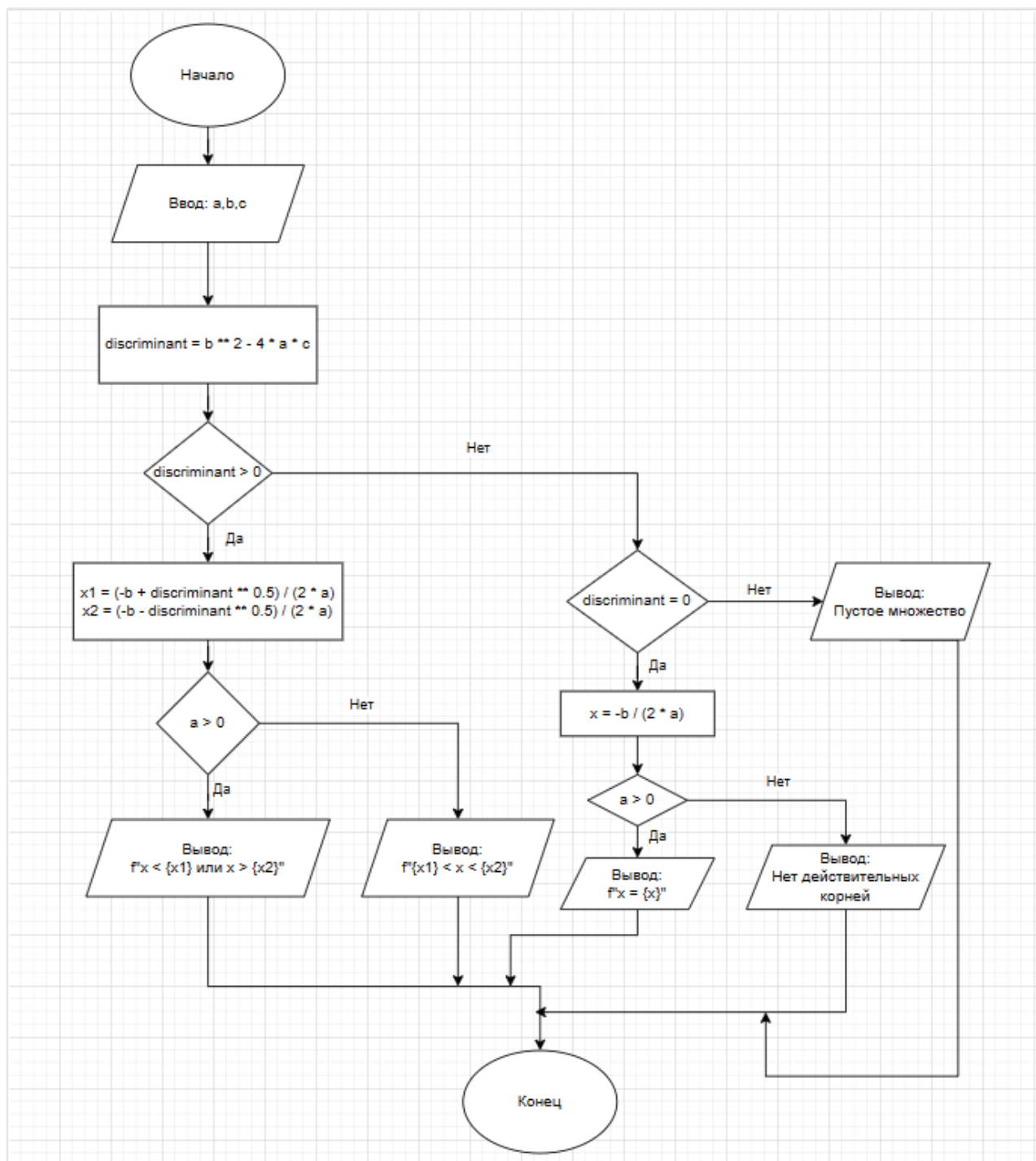
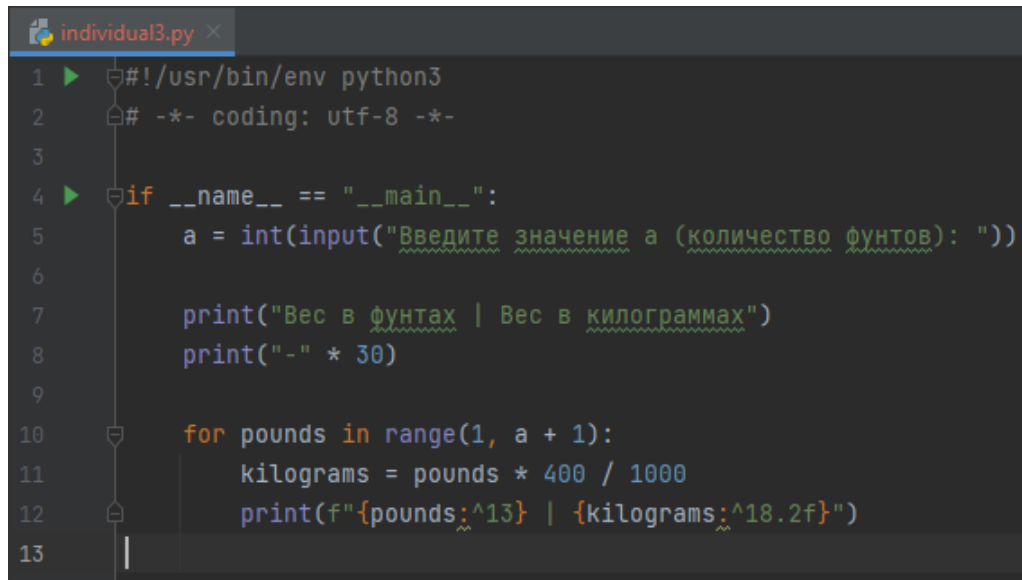


Рисунок 23 – Диаграмма для программы индивидуального задания №2

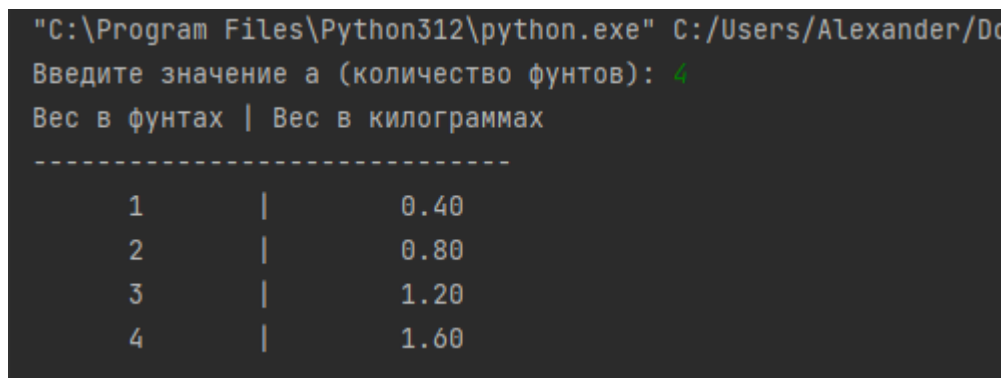


```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == "__main__":
5      a = int(input("Введите значение а (количество фунтов): "))
6
7      print("Вес в фунтах | Вес в килограммах")
8      print("-" * 30)
9
10     for pounds in range(1, a + 1):
11         kilograms = pounds * 400 / 1000
12         print(f"{pounds:^13} | {kilograms:^18.2f}")
13

```

Рисунок 24 – Программа, печатающая таблицу соответствия между весом в фунтах и весом в кг



```

"C:\Program Files\Python312\python.exe" C:/Users/Alexander/D...
Введите значение а (количество фунтов): 4
Вес в фунтах | Вес в килограммах
-----
1           |          0.40
2           |          0.80
3           |          1.20
4           |          1.60

```

Рисунок 25 – Запуск программы

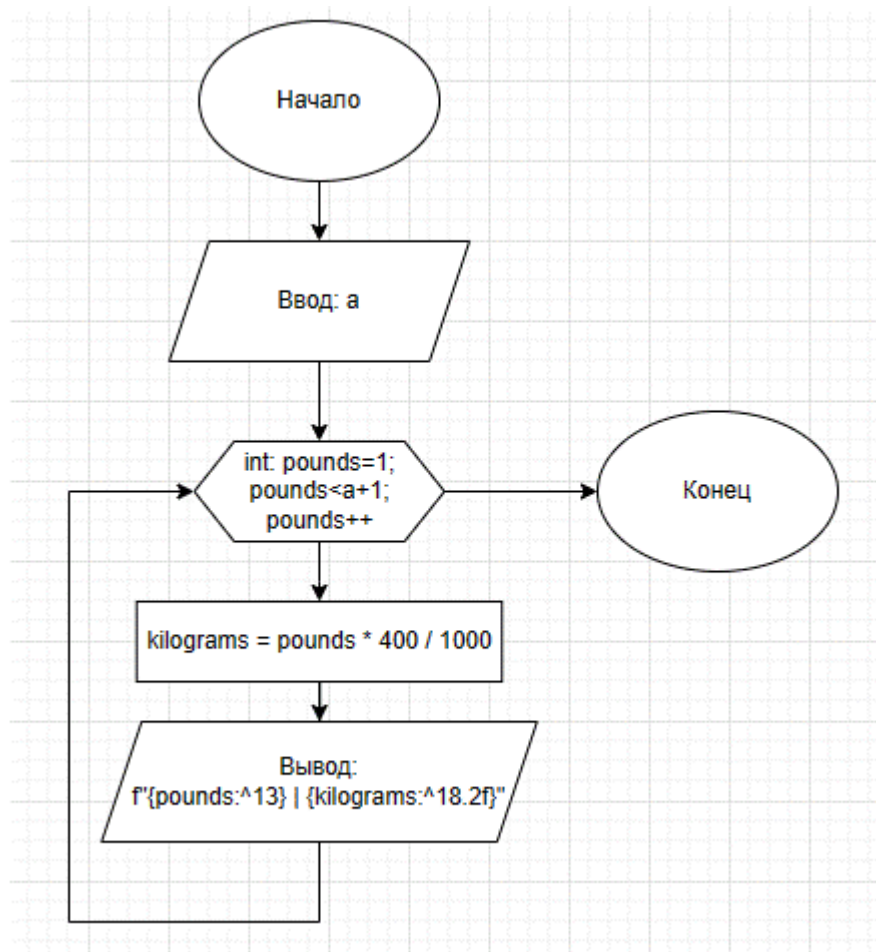


Рисунок 26 – Диаграмма программы индивидуального задания №3

```

individual4.py x
1  ▶  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5
6  ▶  if __name__ == "__main__":
7      n = int(input("Введите значение n для функции Бесселя первого рода: "))
8      x = float(input("Введите значение x: "))
9
10     result = 0.0
11     terms = 50
12
13     for k in range(terms):
14         numerator = ((-(x ** 2) / 4) ** k) / (math.factorial(k) * math.factorial(k + n))
15         result += numerator
16
17     besseL_value = result * (x / 2) ** n
18     print(besseL_value)
19

```

Рисунок 27 – Программа, вычисляющая функцию Бесселя первого рода

```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Do
Введите значение n для функции Бесселя первого рода: 3
Введите значение x: 5
0.36483123061366635
```

Рисунок 28 – Запуск программы

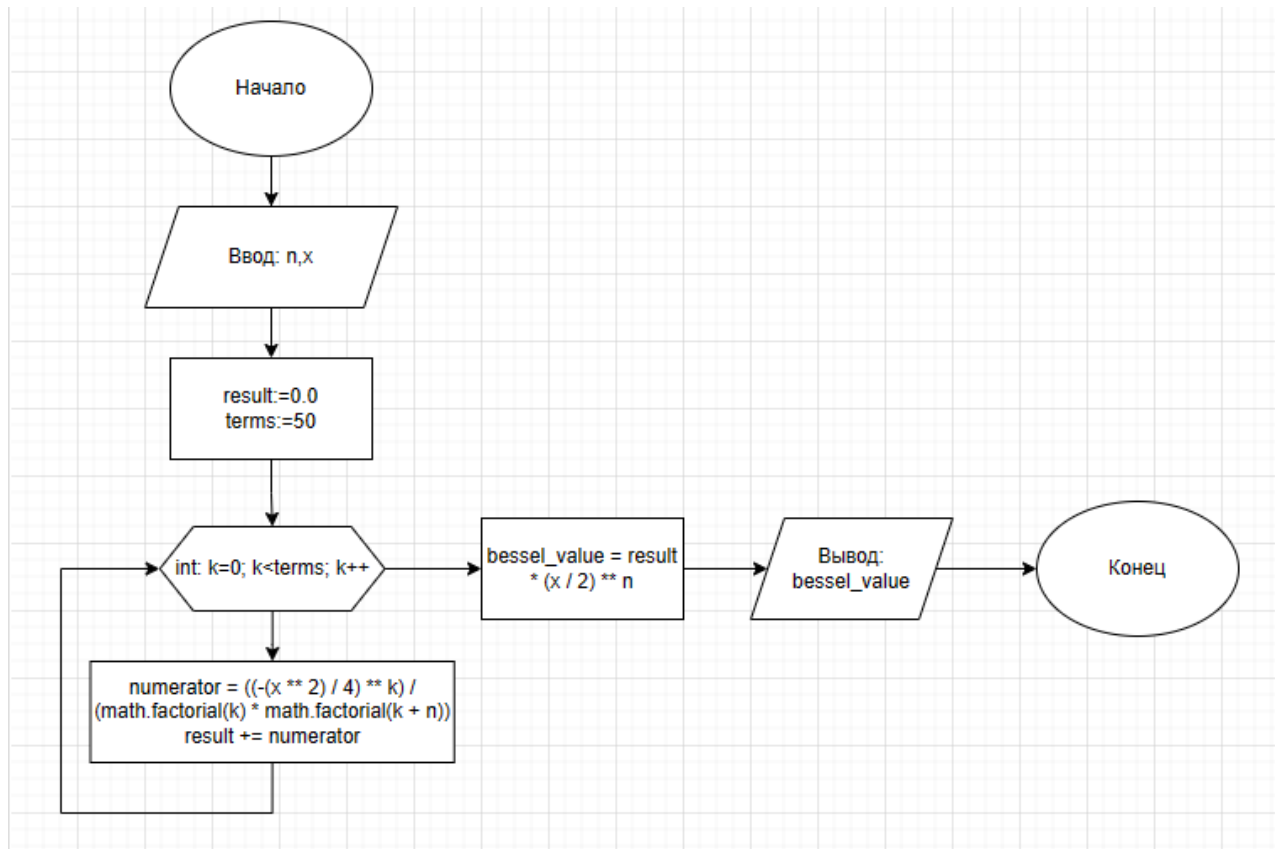


Рисунок 29 – Диаграмма программы индивидуального задания №4

4. Зафиксируем сделанные изменения, сольем ветки и отправим на удаленный репозиторий:

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering5 (develop)
$ git log
commit 17bfb88e2ab375988d6a0c167a74f90cac443a5a (HEAD -> develop)
Author: MrPlatynum <dushinsasha4@gmail.com>
Date: Fri Nov 24 20:10:59 2023 +0300

    финальные изменения

commit c749d16f08c95dd554677870d518ccd630f68cca (origin/main, origin/HEAD, main)
Author: MrPlatynum <71084177+MrPlatynum@users.noreply.github.com>
Date: Fri Nov 24 17:57:59 2023 +0300

    Initial commit
```

Рисунок 30 – Коммиты ветки develop во время выполнения лабораторной работы

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering5 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering5 (main)
$ git merge develop
Updating c749d16..17bfb88
Fast-forward
 .idea/.gitignore          | 0
 .idea/.name               | 1 +
 .idea/ProgrammEngineering5.iml | 8 ++
 .idea/inspectionProfiles/profiles_settings.xml | 6 +
 .idea/misc.xml            | 4 +
 .idea/modules.xml         | 8 ++
 .idea/vcs.xml             | 6 +
 .idea/workspace.xml       | 177 ++++++
 1.py                      | 15 +++
 2.py                      | 19 +++
 3.py                      | 15 +++
 4.py                      | 20 +++
 5.py                      | 29 ++++
 individual1.py            | 16 +++
 individual2.py            | 24 ++++
 individual3.py            | 12 ++
 individual4.py            | 18 +++
17 files changed, 378 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/.name
create mode 100644 .idea/ProgrammEngineering5.iml
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/misc.xml
```

Рисунок 31 – Слияние ветки develop в ветку main

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering5 (main)
$ git push origin main
Enumerating objects: 22, done.
Counting objects: 100% (22/22), done.
Delta compression using up to 12 threads
Compressing objects: 100% (18/18), done.
Writing objects: 100% (21/21), 5.93 KiB | 2.96 MiB/s, done.
Total 21 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/MrPlatynum/ProgrammEngineering5.git
 c749d16..17bfb88 main -> main
```

Рисунок 32 – Отправка на удаленный репозиторий

Ответы на контрольные вопросы:

1. Для чего нужны диаграммы деятельности UML?

Диаграмма деятельности (Activity diagram) показывает поток переходов от одной деятельности к другой. Деятельность (Activity) — это продолжающийся во времени неатомарный шаг вычислений в автомате. Деятельности в конечном счете приводят к выполнению некоего действия (Action), составленного из выполняемых атомарных вычислений, каждое из которых либо изменяет состояние системы, либо возвращает какое-то значение. Действие может заключаться в вызове другой операции, отправке сигнала, создании или уничтожении объекта либо в простом вычислении - скажем, значения выражения. Графически диаграмма деятельности представляется в виде графа, имеющего вершины и ребра.

2. Что такое состояние действия и состояние деятельности?

В потоке управления, моделируемом диаграммой деятельности, происходят различные события. Вы можете вычислить выражение, в результате чего изменяется значение некоторого атрибута или возвращается некоторое значение. Также, например, можно выполнить операцию над объектом, послать ему сигнал или даже создать его или уничтожить. Все эти выполняемые атомарные вычисления называются состояниями действия, поскольку каждое из них есть состояние системы, представляющее собой выполнение некоторого действия.

Состояния действия не могут быть подвергнуты декомпозиции. Кроме того, они атомарны. Это значит, что внутри них могут происходить различные события, но выполняемая в состоянии действия работа не может быть прервана. Обычно предполагается, что длительность одного состояния действия занимает неощутимо малое время. В противоположность этому состояния деятельности могут быть подвергнуты дальнейшей декомпозиции, вследствие чего выполняемую деятельность можно представить с помощью других диаграмм деятельности. Состояния деятельности не являются атомарными, то есть могут быть прерваны. Предполагается, что для их завершения требуется заметное

время. Можно считать, что состояние действия — это частный вид состояния деятельности, а конкретнее - такое состояние, которое не может быть подвергнуто дальнейшей декомпозиции. А 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?

Когда действие или деятельность в некотором состоянии завершается, поток управления сразу переходит в следующее состояние действия или деятельности. Для описания этого потока используются переходы, показывающие путь из одного состояния действия или деятельности в другое. В UML переход представляется простой линией со стрелкой.

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Это алгоритм, в котором вычислительный процесс осуществляется либо по одной, либо по другой ветви, в зависимости от выполнения некоторого условия. Программа разветвляющейся структуры реализует такой алгоритм. В программе разветвляющейся структуры имеется один или несколько условных операторов. Для программной реализации условия используется логическое выражение. В сложных структурах с большим числом ветвей применяют оператор выбора.

5. Чем отличается разветвляющийся алгоритм от линейного?

Разветвляющийся алгоритм отличается от линейного тем, что он позволяет программе принимать решения на основе различных условий.

6. Что такое условный оператор? Какие существуют его формы?

Оператор ветвления `if` позволяет выполнить определенный набор инструкций в зависимости от некоторого условия. Существуют: `if`, `if-else`, `if-elif-else`.

7. Какие операторы сравнения используются в Python?

`>`, `<`, `<=`, `>=`, `!=`, `==`

8. Что называется простым условием? Приведите примеры:

Простые условия – это те, в которых выполняется только одна логическая операция, например: `var >= 1023`, `var == 0`, `var != 3` и т. д.

9. Что такое составное условие? Приведите примеры:

Может понадобиться получить ответа `True` или `False` в зависимости от результата выполнения двух и более простых выражений. Для этого используется составное условие из операторов `or`, `and` и `not`. Например: `var < 100 and var > 0`, `not var < 10 or var == 100` и т. д.

10. Какие логические операторы допускаются при составлении сложных условий?

В Python допускаются три основных логических оператора для составления сложных условий: `and`, `or` и `not`.

11. Может ли оператор ветвления содержать внутри себя другие ветвления?

Да, может.

12. Какой алгоритм является алгоритмом циклической структуры?

Это алгоритм, в котором происходит многократное повторение одного и того же участка программы. Такие повторяемые участки вычислительного процесса называются циклами.

13. Типы циклов в языке Python:

Цикл со счетчиком (`for`), цикл с предусловием (`while`), цикл `for-each` (`for in`). Цикла с постусловием нет, но его можно сделать, используя другие алгоритмические конструкции и операторы.

14. Назовите назначение и способы применения функции `range`:

Функция `range` возвращает неизменяемую последовательность чисел в виде объекта `range`. Функция `range` хранит только информацию о значениях `start`, `stop` и `step` и вычисляет значения по мере необходимости. Это значит, что независимо от размера диапазона, который описывает функция `range`, она всегда будет занимать фиксированный объем памяти. Обычно применяется для цикла с счетчиком.

15. Как с помощью функции `range` организовать перебор значений от 15

до 0 с шагом 2?

```
range(15, -1, -2)
```

16. Могут ли быть циклы вложенными?

Могут.

17. Как образуется бесконечный цикл и как выйти из него?

Например, можно воспользоваться конструкцией `while True`. Чтобы выйти необходимо воспользоваться оператором `break`.

18. Для чего нужен оператор `break`?

Оператор `break` предназначен для досрочного прерывания работы цикла.

19. Где употребляется оператор `continue` и для чего он используется?

Оператор `continue` запускает цикл заново, при этом код, расположенный после данного оператора, не выполняется.

20. Для чего нужны стандартные потоки `stdout` и `stderr`?

В операционной системе по умолчанию присутствуют стандартных потока вывода на консоль: буферизованный поток `stdout` для вывода данных и информационных сообщений, а также небуферизованный поток `stderr` для вывода сообщений об ошибках.

21. Как в Python организовать вывод в стандартный поток `stderr`?

По умолчанию функция `print` использует поток `stdout`. Для того, чтобы использовать поток `stderr` необходимо передать его в параметре `file` функции `print`. Само же определение потоков `stdout` и `stderr` находится в стандартном пакете Python `sys`. Хорошим стилем программирования является наличие вывода ошибок в стандартный поток `stderr` поскольку вывод в потоки `stdout` и `stderr` может обрабатываться как операционной системой, так и сценариями пользователя по-разному.

22. Каково назначение функции `exit`?

В Python завершить программу и передать операционной системе заданный код возврата можно посредством функции `exit`.