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 ЛАБОРАТОРНОЙ РАБОТЕ №5
дисциплины «Программирование на python»

Выполнил:
Кожуховский Виктор Андреевич
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной
техники и автоматизированных систем », очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

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.

Методика и порядок выполнения работы

1. Изучил теоретический материал работы.
2. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

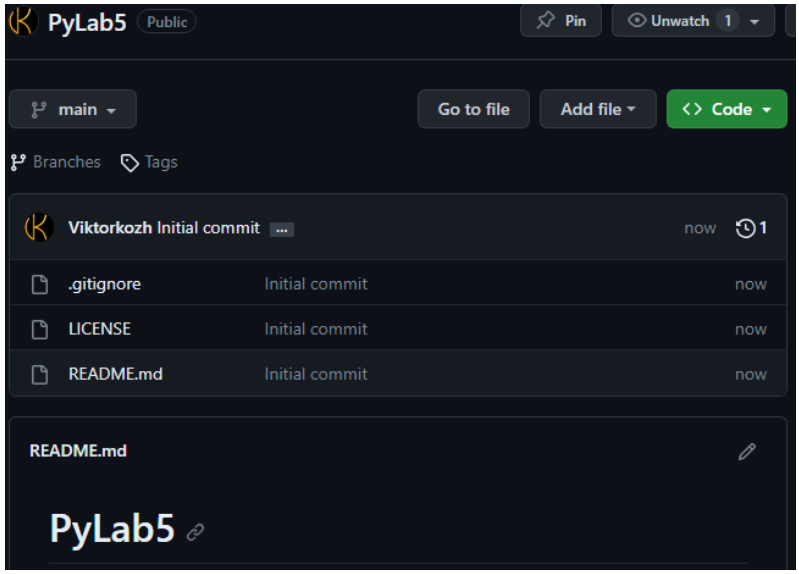


Рисунок 1. Созданный репозиторий

3. Выполнил клонирование созданного репозитория.

```
PS C:\Users\viktor\Desktop\скфу\python\gitstuff> git clone https://github.com/Viktorkozh/PyLab5.git
Cloning into 'PyLab5'...
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 2. Клонированный репозиторий

4. Организовал свой репозиторий в соответствие с моделью ветвления git-flow.

```
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab5> git branch developer
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab5> git checkout developer
Switched to branch 'developer'
```

Рисунок 3. Добавленная ветка developer

5. Проработал примеры лабораторной работы. Создал для каждого примера отдельный модуль языка Python. Зафиксировал изменения в репозитории.

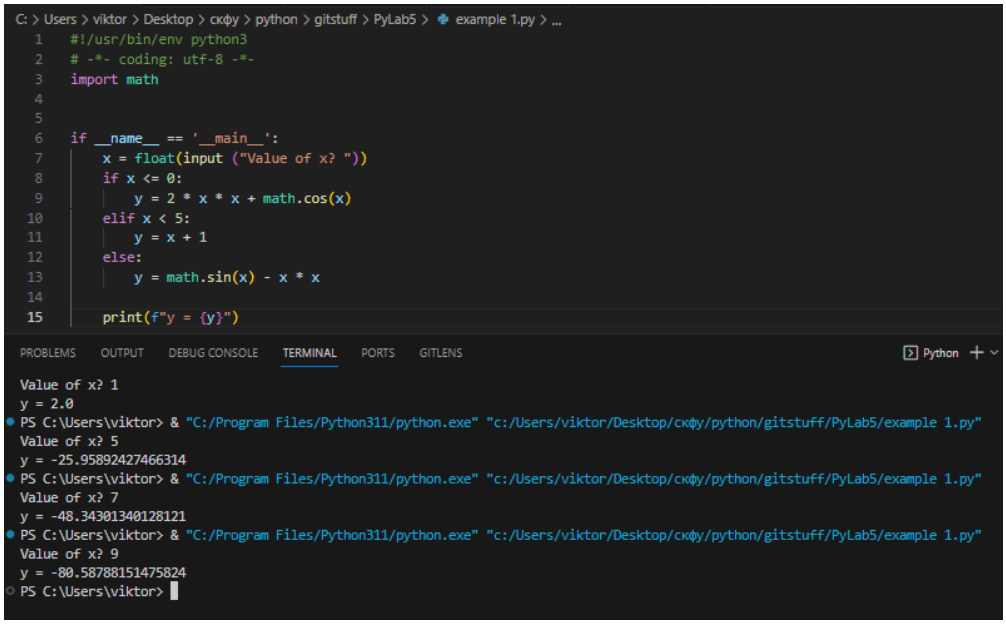


Рисунок 4. Код и выполнение примера 1

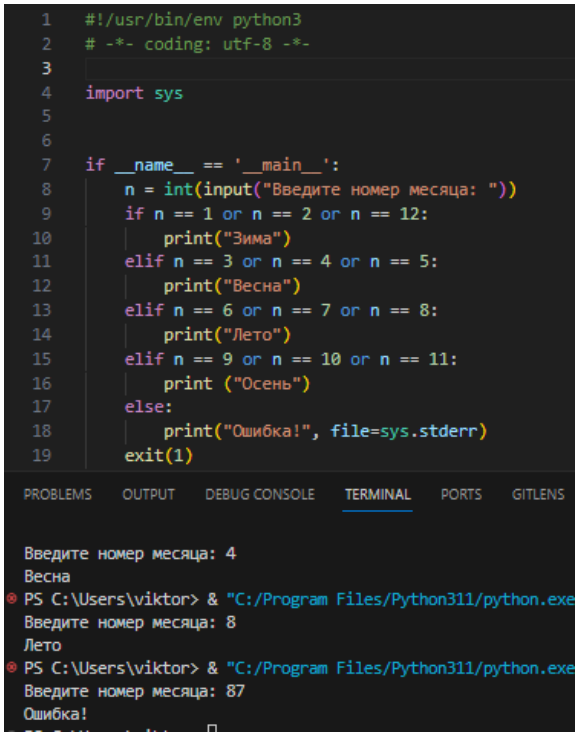
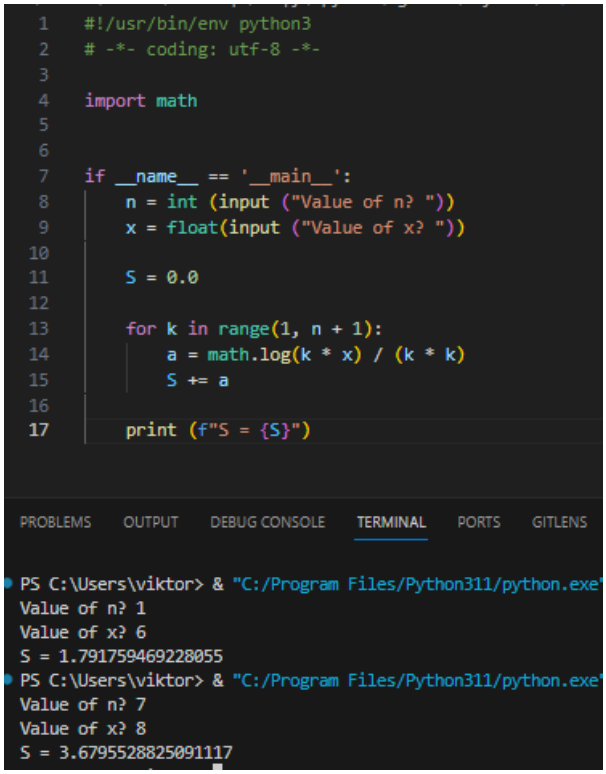


Рисунок 5. Код и выполнение примера 2



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

import math


if __name__ == '__main__':
    n = int (input ("Value of n? "))
    x = float(input ("Value of x? "))

    S = 0.0

    for k in range(1, n + 1):
        a = math.log(k * x) / (k * k)
        S += a

    print (f"S = {S}")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe"
Value of n? 1
Value of x? 6
S = 1.791759469228055
PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe"
Value of n? 7
Value of x? 8
S = 3.6795528825091117
```

Рисунок 6. Код и выполнение примера 3

C: > Users > viktor > Desktop > скфу > python > gitstuff > PyLab5 > example

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

import math
import sys


if __name__ == '__main__':
    a = float(input("Value of a? "))
    if a < 0:
        print("Illegal value of a", file=sys.stderr)
        exit(1)

    x, eps = 1, 1e-10
    while True:
        xp = x
        x = (x + a / x) / 2
        if math.fabs(x - xp) < eps:
            break

    print(f"x = {x}\nx = {math.sqrt(a)}")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" "c
Value of a? 7
x = 2.6457513110645907
x = 2.6457513110645907
PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" "c
Value of a? 8
x = 2.82842712474619
x = 2.8284271247461903
```

Рисунок 7. Код и выполнение примера 4

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

import math
import sys


# Постоянная Эйлера.
EULER = 0.5772156649015328606
# Точность вычислений.
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, k = a, 1

    # Найти сумму членов ряда.
    while math.fabs(a) > EPS:
        a *= x * k / (k + 1) ** 2
        S += a
        k += 1

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

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" "c:/Users/vi
Value of x? 8
Ei(8.0) = 440.3798995348318
PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" "c:/Users/vi
Value of x? 7
Ei(7.0) = 191.50474333549477
PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" "c:/Users/vi
Value of x? 10
Ei(10.0) = 2492.228976241855
```

Рисунок 8. Код и выполнение примера 5

6. Для примеров 4 и 5 построил UML-диаграмму деятельности. Для построения диаграмм деятельности использовал веб-сервис Google https://www.diagrams.net/.

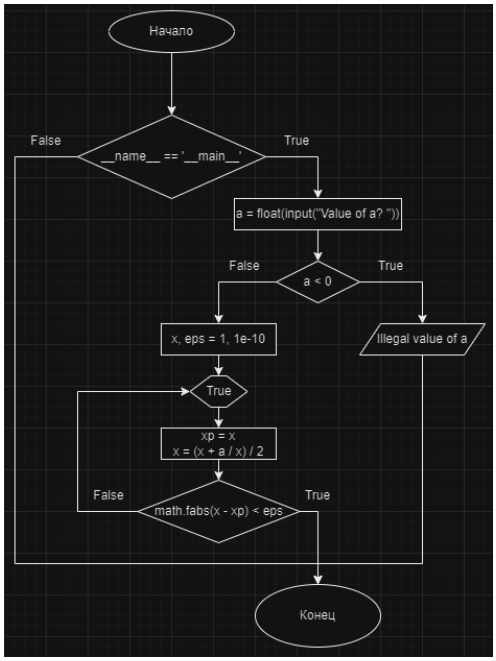


Рисунок 9. Схема примера 4

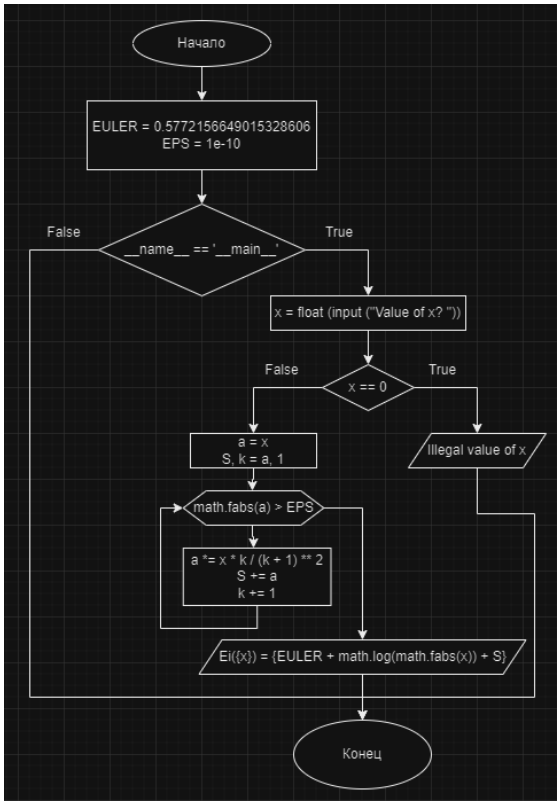


Рисунок 10. Схема примера 5

7. Выполнил индивидуальные задания, согласно своему варианту.

Дано число m (1<=m<=7). Вывести на экран название дня недели, который соответствует этому номеру.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# Дано число (1<=m<=7). Вывести на экран название

import sys


if __name__ == '__main__':
    n = int(input("Введите номер дня недели: "))

    if n == 1:
        print("Понедельник")
    elif n == 2:
        print("Вторник")
    elif n == 3:
        print("Среда")
    elif n == 4:
        print("Четверг")
    elif n == 5:
        print("Пятница")
    elif n == 6:
        print("Суббота")
    elif n == 7:
        print("Воскресенье")
    else:
        print("Ошибка!", file=sys.stderr)
        exit(1)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

Воскресенье
PS C:\Users\viktor> & "C:/Program Files/Python311/python.ex
Введите номер дня недели: 6
Суббота
PS C:\Users\viktor> & "C:/Program Files/Python311/python.ex
Введите номер дня недели: 9
Ошибка!
PS C:\Users\viktor> & "C:/Program Files/Python311/python.ex
Введите номер дня недели: 7
Воскресенье
```

Рисунок 11. Код и выполнение индивидуального задания 1

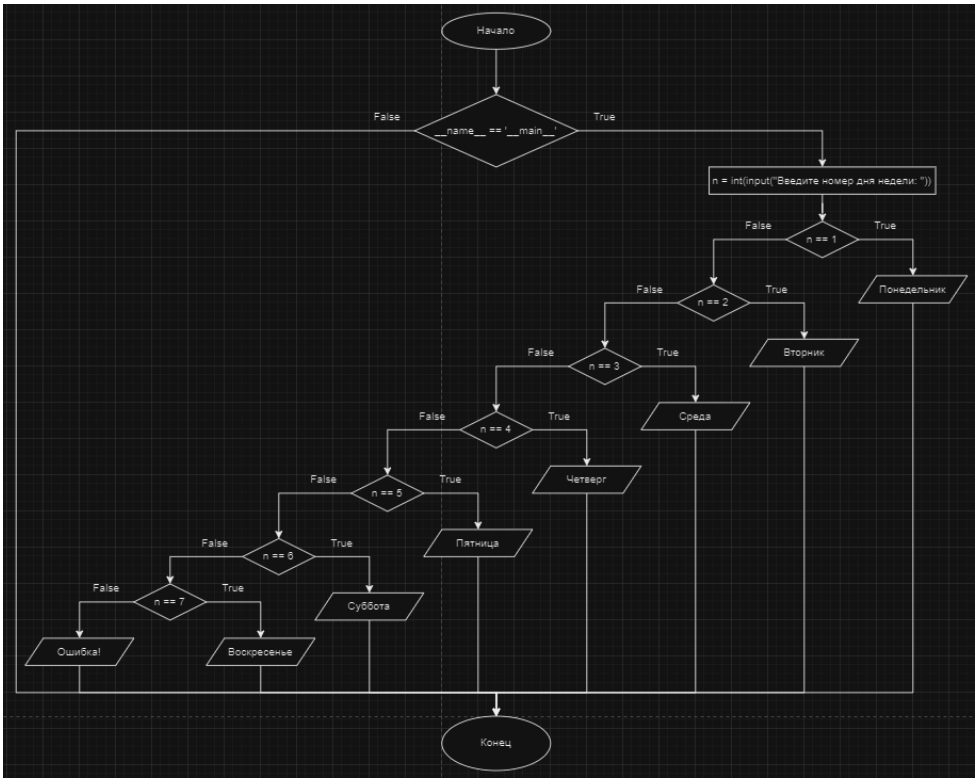


Рисунок 12. UML схема индивидуального задания 1

Даны три действительных числа. Составить программу с использованием конструкций ветвления, выбирающую из них те, которые принадлежат интервалу (0, 1).

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# Даны три действительных числа. Составить программ

if __name__ == '__main__':
    print("Введите три действительных числа:")
    a = float(input())
    b = float(input())
    c = float(input())

    if 0 < a < 1:
        print(f"{a} принадлежит интервалу (0, 1)")
    if 0 < b < 1:
        print(f"{b} принадлежит интервалу (0, 1)")
    if 0 < c < 1:
        print(f"{c} принадлежит интервалу (0, 1)")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe"
Введите три действительных числа:
3
0.4
2
0.4 принадлежит интервалу (0, 1)
```

Рисунок 13. Код и выполнение индивидуального задания 2

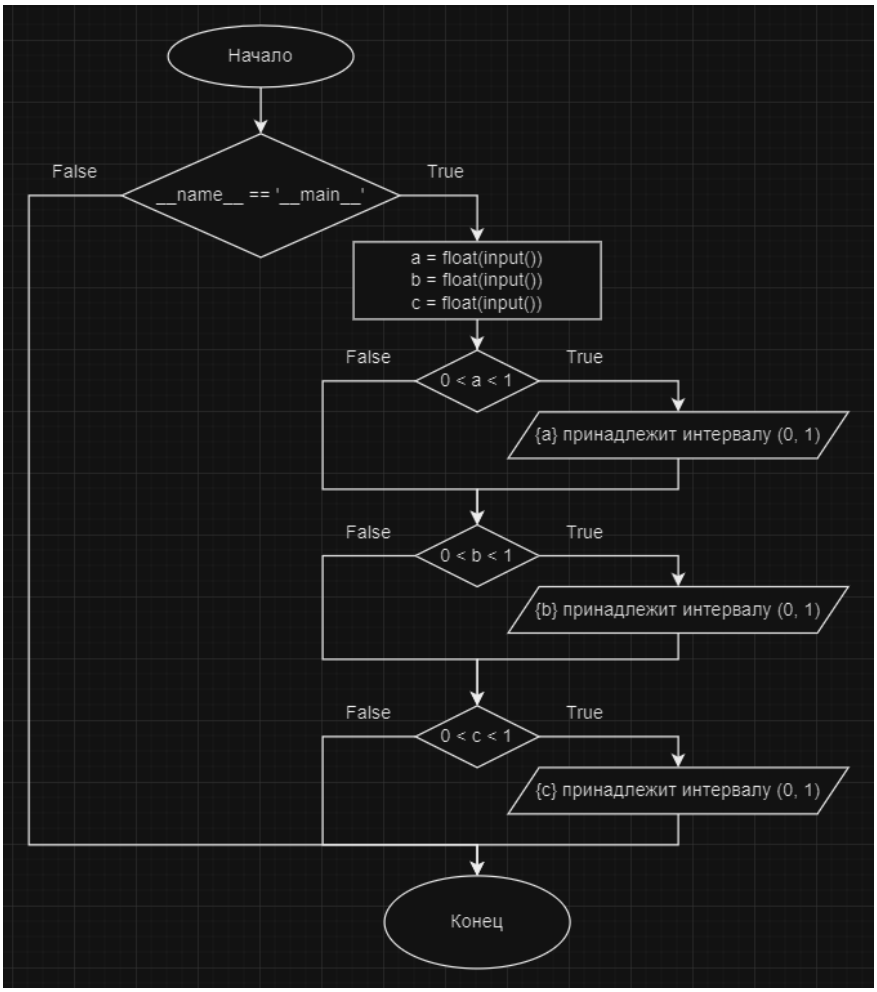


Рисунок 14. UML схема индивидуального задания 2

Ученик выучил в первый день 5 английских слов. В каждый следующий день он выучивал на 2 слова больше, чем в предыдущий. Сколько английских слов выучит ученик в 10-ый день занятий. Программа с использованием конструкций цикла.

```
You, 20 seconds ago | 1 author (You)
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# Ученик выучил в первый день 5 английских слов. В каждый следующий

words = 5
days = 10
sum = 0

    You, 1 second ago • Uncommitted changes
if __name__ == '__main__':
    for day in range(1, days + 1):
        sum += words
        words += 2

    print(f"Ученик выучит {sum} английских слов в {days}-й день.")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" c:/Users/viktor/D
Ученик выучит 140 английских слов в 10-й день.
```

Рисунок 15. Код и выполнение индивидуального задания 3

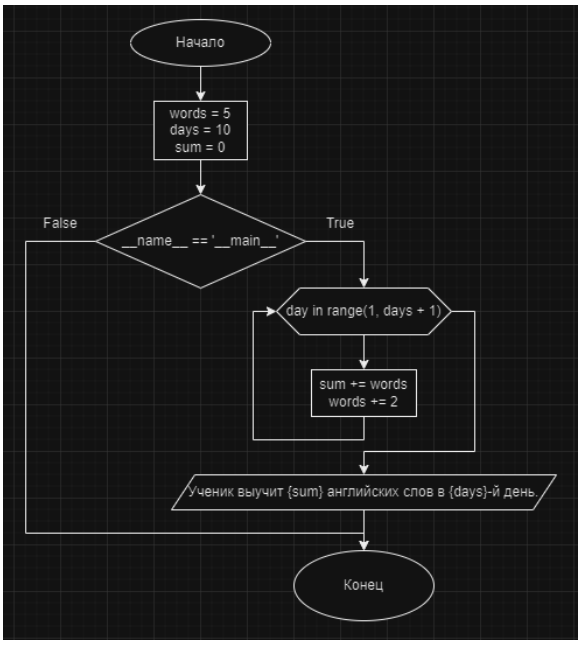


Рисунок 16. UML схема индивидуального задания 3

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

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

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

$$a_k = \frac{(x^2/4)^k}{k!*(k+n)!}$$

$$a_{k+1} = \frac{(x^2/4)^{k+1}}{(k+1)!*(k+1+n)!}$$

$$\frac{a_{k+1}}{a_k} = \frac{(x^2/4)^{k+1}*k!*(k+n)!}{(x^2/4)^k*(k+1)!*(k+n+1)!} = \frac{(x^2/4)}{(k+1)(k+n+1)}$$

$$a_{k+1} = \frac{x^2/4}{(k+1)(k+n+1)} * a_k$$

$$a_0 = \frac{(x^2/4)^0}{0!*(0+n)!} = \frac{1}{n!}$$

Рисунок 17. Разложение функции Бесселя первого рода

Рисунок 18. Код и выполнение усложненного задания

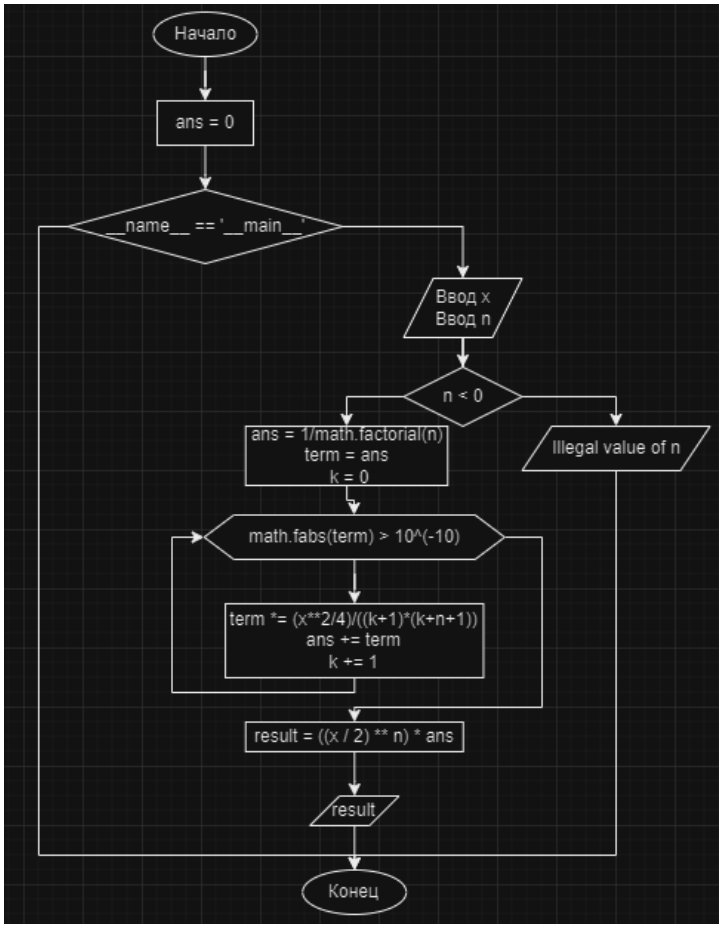


Рисунок 19. UML схема усложненного задания

8. Зафиксировал сделанные изменения в репозитории.
9. Выполнил слияние ветки для разработки с веткой main / master.
10. Отправил сделанные изменения на сервер GitHub.

Вопросы для защиты работы

1. Для чего нужны диаграммы деятельности UML?

Для облегченного наглядного анализа алгоритмов действий.

2. Что такое состояние действия и состояние деятельности?

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?

Процесс (активность):

Отображается прямоугольником с закругленными углами. Представляет собой выполняемую операцию или шаг в диаграмме.

Стрелка перехода:

Используется для обозначения перехода от одного шага к другому. Стрелка указывает направление выполнения.

Решающий узел (решение, ветвление):

Представляется ромбовидным элементом. Используется для обозначения точек ветвления в процессе, где выполнение может идти по разным путям в зависимости от условия.

Исключающее решение (решение с условием):

Этот элемент также ромбовидный, но с дополнительным тремя стрелками, указывающими на возможные пути выполнения в зависимости от условий.

Слияние (соединение, объединение):

Используется для объединения потоков выполнения после ветвления. Это часто представлено полукруглым элементом, указывающим на слияние разветвленных потоков.

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?

Разветвляющийся алгоритм отличается от линейного тем, что в разветвляющейся структуре вычислительный процесс может принимать различные пути выполнения в зависимости от условий. То есть программа

может выбирать, какие блоки кода выполнять, исходя из значений переменных или результатов проверок условий.

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

Условный оператор в Python - это конструкция, которая выполняет различные блоки кода в зависимости от истинности или ложности выражения (условия). В Python основным условным оператором является if. Дополнительно, можно использовать elif (сокращение от "else if") и else, чтобы предоставить альтернативные варианты выполнения кода.

Формы условного оператора:

Простая форма с if:

```
x = 10
if x > 5:
    print("x больше 5")
```

Условие с if и else:

```
x = 3
if x > 5:
    print("x больше 5")
else:
    print("x меньше или равен 5")
```

Цепочка условий с if, elif и else:

```
x = 5
if x > 5:
    print("x больше 5")
elif x < 5:
    print("x меньше 5")
else:
    print("x равен 5")
```

Вложенные условия:

```
x = 10
y = 5
if x > 5:
    if y > 2:
        print("Оба условия выполняются.")
```

7. Какие операторы сравнения используются в Python?

В Python используются специальные знаки, подобные тем, которые используются в математике: > (больше), < (меньше), >= (больше или равно), <= (меньше или равно), == (равно), != (не равно).

8. Что называется простым условием? Приведите примеры.

Простые инструкции описываются одной строкой кода.

```
number = 7

if number % 2 == 0:
    print("Число четное.")
else:
    print("Число нечетное.")
```

9. Что такое составное условие? Приведите примеры.

Составные условия - состоят из двух или нескольких простых отношений (условий), которые объединяются с помощью логических операций:

И - логическое умножение - на языке Python записывается как and,

ИЛИ - логическое сложение - на языке Python записывается как or,

НЕ - логическое отрицание -на языке Python записывается как not.

```
a = True
b = False
print(a or b)
```

10. Какие логические операторы допускаются при составлении сложных условий?

И (and), ИЛИ (or) и унарный логический оператор not, т. е. отрицание.

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

Оператор ветвления может содержать внутри себя сколько угодно других ветвлений.

12. Какой алгоритм является алгоритмом циклической структуры?

Алгоритм 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.

Циклы где проверка условия выполняется на каждой итерации либо до тела цикла (тогда говорят о цикле с предусловием), либо после тела цикла (цикл с постусловием).

14. Назовите назначение и способы применения функции range.

Функция range возвращает неизменяемую последовательность чисел в виде объекта range.

15. Как с помощью функции range организовать перебор значений от 15 до 0 с шагом 2?

For i in range(0, 15, 2)

16. Могут ли быть циклы вложенными?

Могут.

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

Бесконечный цикл образуется если изменяемая переменная никогда не доходит до значения, при котором цикл останавливается. Выйти из такого цикла помогает оператор break.

18. Для чего нужен оператор break?

Оператор break предназначен для досрочного прерывания работы цикла while.

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?

Stdout предназначен для обычного вывода, stderr обычно используется для вывода ошибок.

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

Для этого можно использовать строку file=sys.stderr в print.

22. Каково назначение функции exit?

exit(код возврата) приводит к немедленному завершению программы и операционной системе передается код возврата.