

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

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №3
дисциплины «Программирование на Python»
Вариант 7

Выполнила:
Еремина Татьяна Евгеньевна
2 курс, группа ИВТ-б-о-24-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Проверил:
Воронкин Р.А., доцент департамента
цифровых, робототехнических систем
и электроники института
перспективной инженерии

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2025 г.

Тема: условные операторы и циклы в языке Python.

Цель: приобретение навыков программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Освоить операторы языка Python версии 3.x if, while, for, break continue, позволяющих реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.

Практическая часть:

Исходный репозиторий: <https://github.com/TL-hash/labs3>

Пример 1. Написать программу с использованием конструкции ветвления и вычислить значение функции:

$$y = \begin{cases} 2x^2 + \cos x, & x \leq 3.5, \\ x + 1, & 0 < x < 5, \\ \sin 2x - x^2, & x \geq 5. \end{cases}$$

Листинг программы:

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

import sys

if __name__ == '__main__':
    m = int(input("Введите номер месяца (от 1 до 12): "))

    if m == 1:
        print("Январь")
    elif m == 2:
        print("Февраль")
    elif m == 3:
        print("Март")
    elif m == 4:
        print("Апрель")
    elif m == 5:
        print("Май")
    elif m == 6:
        print("Июнь")
    elif m == 7:
        print("Июль")
    elif m == 8:
        print("Август")
    elif m == 9:
        print("Сентябрь")
    elif m == 10:
        print("Октябрь")
```

```

elif m == 11:
    print("Ноябрь")
elif m == 12:
    print("Декабрь")
else:
    print("Ошибка!", file=sys.stderr)
    exit(1)

```

Результат выполнения:

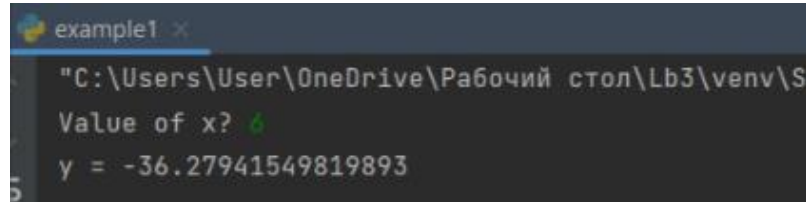


Рисунок 1 - Результат выполнения примера 1

Пример 2. Написать программу для решения задачи: с клавиатуры вводится номер месяца от 1 до 12, необходимо для этого номера месяца вывести наименование времени года.

Листинг программы:

```

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

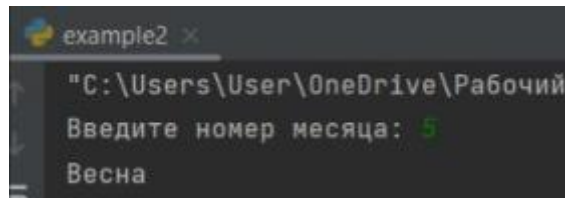
import sys

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

    if n == 1 or n == 2 or n == 12:
        print("Зима")
    elif n == 3 or n == 4 or n == 5:
        print("Весна")
    elif n == 6 or n == 7 or n == 8:
        print("Лето")
    elif n == 9 or n == 10 or n == 11:
        print("Осень")
    else:
        print("Ошибка!", file=sys.stderr)
        exit(1)

```

Результат выполнения:



```
example2 x
"C:\Users\User\OneDrive\Рабочий
Введите номер месяца: 5
Весна
```

Рисунок 2 - Результат выполнения примера 2

Пример 3. Написать программу, позволяющую написать конечную сумму:

$$S = \sum_{k=1}^n \frac{\ln kx}{k^2},$$

Листинг программы:

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

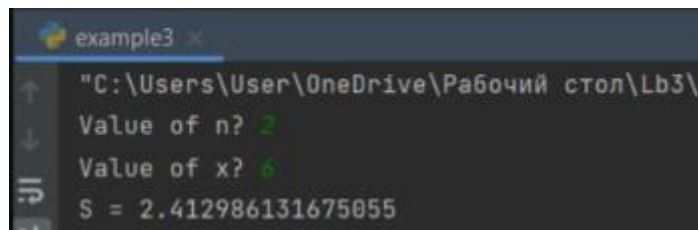
import math

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}")
```

Результат выполнения:



```
example3 x
"C:\Users\User\OneDrive\Рабочий стол\Лб3\
Value of n? 2
Value of x? 6
S = 2.412986131675055
```

Рисунок 3 - Результат выполнения примера 3

Пример 4. Найти значение квадратного корня из положительного числа a вводимого с клавиатуры, с некоторой заданной точностью ϵ с помощью рекуррентного соотношения:

$$x_{n+1} = \frac{1}{2} \cdot \left(x_n + \frac{a}{x_n} \right).$$

UML-диаграмма деятельности:

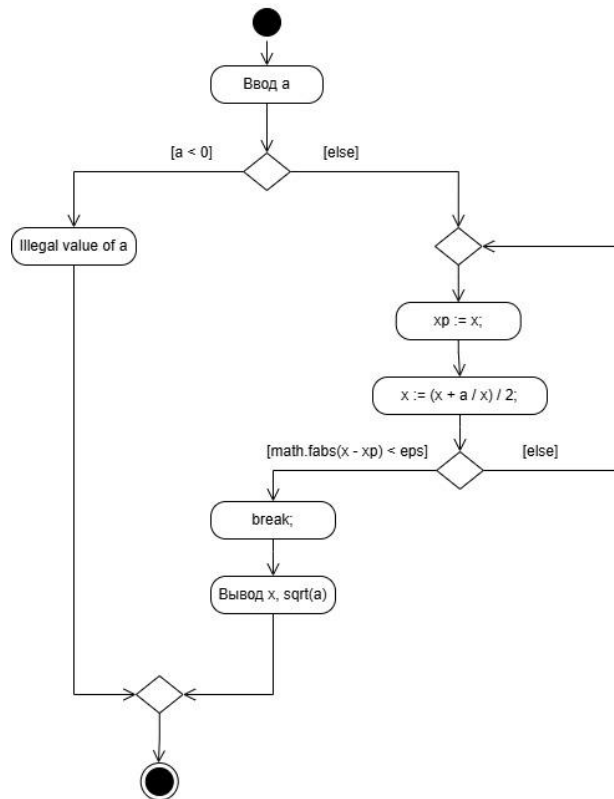


Рисунок 4 - UML-диаграмма для примера 4

Листинг программы:

```
#!/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)}")
```

Результат выполнения:

```

example4 x
"C:\Users\User\OneDrive\Рабочий стол\Lb3\ven
Value of a? 1
x = 1.0
X = 1.0

```

Рисунок 5 - Результат выполнения примера 4

Пример 5. Вычислить значение специальной (интегральной показательной) функции:

$$Ei(x) = \int_{-\infty}^x \frac{\exp t}{t} dt = \gamma + \ln x + \sum_{k=1}^{\infty} \frac{x^k}{k \cdot k!},$$

UML-диаграмма деятельности:

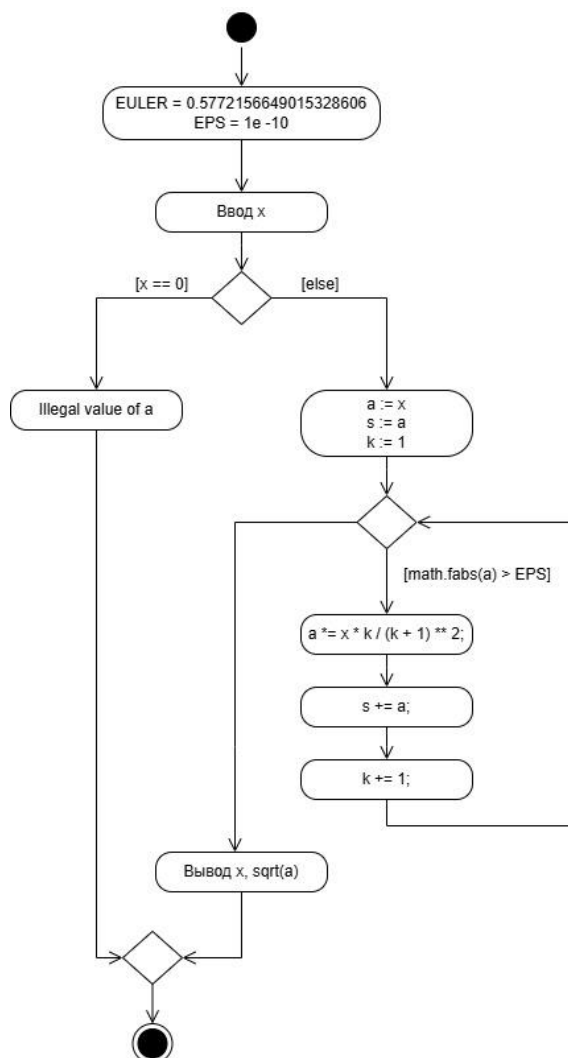


Рисунок 6 - UML-диаграмма деятельности для примера 5

Листинг программы:

```

#!/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}")

```

Результат выполнения программы:

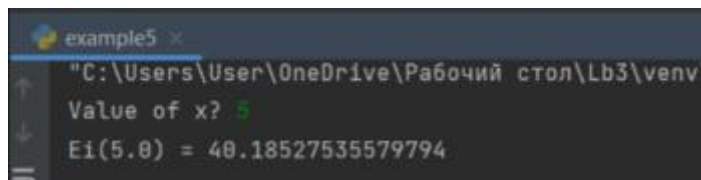


Рисунок 7 - Результат выполнения программы для примера 5

Задание 1. С клавиатуры вводится цифра m (от 1 до 12). Вывести на экран название месяца, соответствующего цифре.

Решение: для решения данного задания воспользуемся конструкцией `if...elif...else`, где условием будет являться цифра, а выводится название месяца.

UML-диаграмма деятельности:

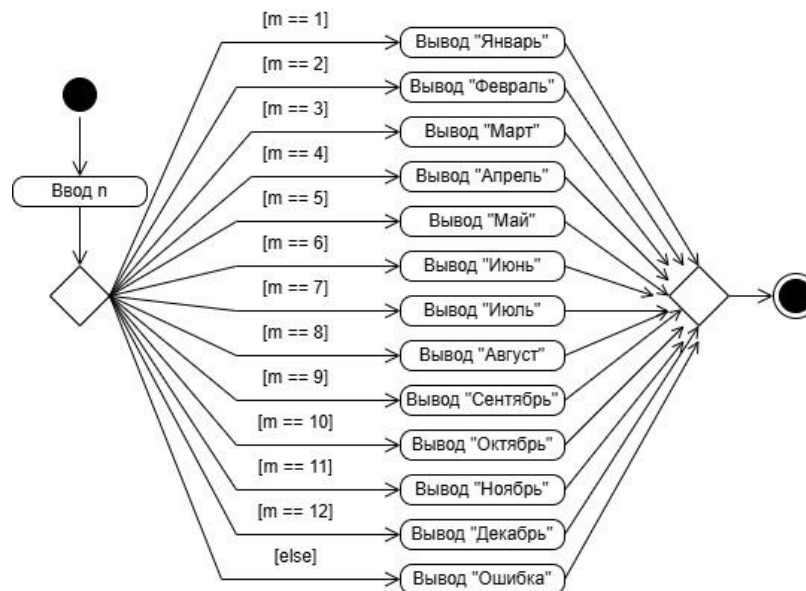


Рисунок 8 - UML-диаграмма для задания 1

Листинг программы:

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

import sys

if __name__ == '__main__':
    m = int(input("Введите номер месяца (от 1 до 12): "))

    if m == 1:
        print("Январь")
    elif m == 2:
        print("Февраль")
    elif m == 3:
        print("Март")
    elif m == 4:
        print("Апрель")
    elif m == 5:
        print("Май")
    elif m == 6:
        print("Июнь")
    elif m == 7:
        print("Июль")
    elif m == 8:
        print("Август")
    elif m == 9:
        print("Сентябрь")
    elif m == 10:
        print("Октябрь")
    elif m == 11:
        print("Ноябрь")
    elif m == 12:
        print("Декабрь")
    else:
        print("Ошибка")
```



```
print("Ошибка!", file=sys.stderr)
exit(1)
```

Результат выполнения программы:

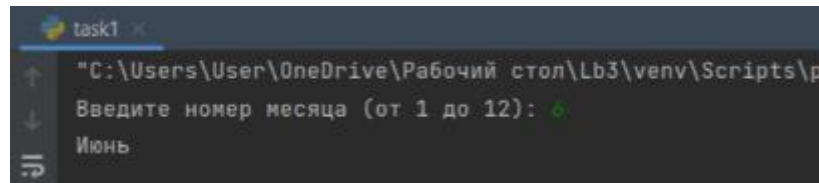


Рисунок 9 - Результат выполнения программы задания 1

Задание 2. Провести исследование биквадратного уравнения $ax^4 + bx^2 + c = 0$ ($a \neq 0$), где a, b, c – действительные числа. Если действительных корней нет, то об этом должно быть выдано сообщение, иначе должны быть выданы 2 или 4.

Решение: чтобы найти корни данного уравнения сделаем замену в уравнении $x^2 = y$ и решим относительно y . Для этого пропишем все условия конструкцией `if...elif...else` и запишем соответствующие формулы нахождения корней.

UML-диаграмма деятельности:

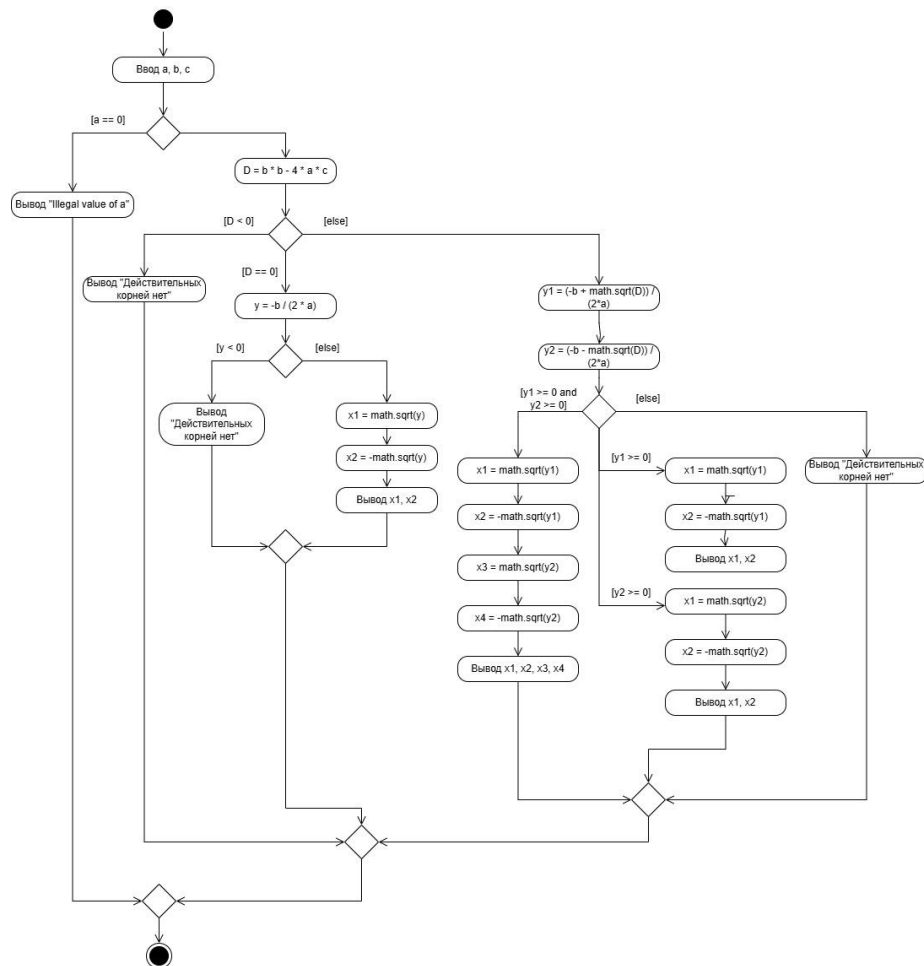


Рисунок 10 - UML-диаграмма для задания 2

Листинг программы:

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

```
import math
import sys
```

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

```
#Делаем замену в уравнении  $x^2 = y$  и решаем относительно  $y$ 
```

```
D = b * b - 4 * a * c
```

```
if D < 0:
    print("Действительных корней нет")
elif D == 0:
    y = -b / (2 * a)
    if y < 0:
```

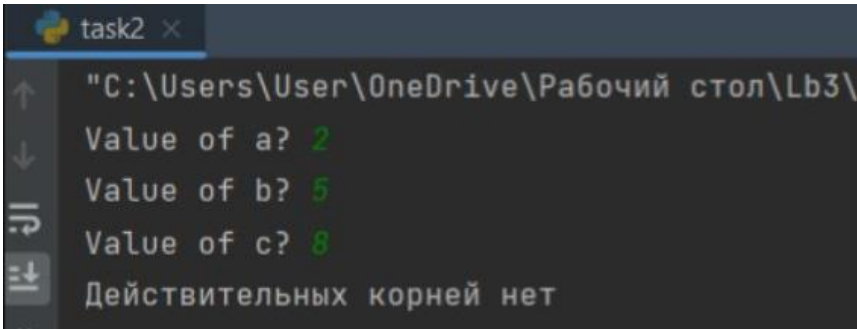
```

        print("Действительных корней нет")
    else:
        x1 = math.sqrt(y)
        x2 = -math.sqrt(y)
        print(f"x1 = {x1}")
        print(f"x2 = {x2}")
    else:
        y1 = (-b + math.sqrt(D)) / (2 * a)
        y2 = (-b - math.sqrt(D)) / (2 * a)

        if y1 >= 0 and y2 >= 0:
            x1 = math.sqrt(y1)
            x2 = -math.sqrt(y1)
            x3 = math.sqrt(y2)
            x4 = -math.sqrt(y2)
            print(f"x1 = {x1}")
            print(f"x2 = {x2}")
            print(f"x3 = {x3}")
            print(f"x4 = {x4}")
        elif y1 >= 0:
            x1 = math.sqrt(y1)
            x2 = -math.sqrt(y1)
            print(f"x1 = {x1}")
            print(f"x2 = {x2}")
        elif y2 >= 0:
            x1 = math.sqrt(y2)
            x2 = -math.sqrt(y2)
            print(f"x1 = {x1}")
            print(f"x2 = {x2}")
        else:
            print("Действительных корней нет")

```

Результат выполнения программы.

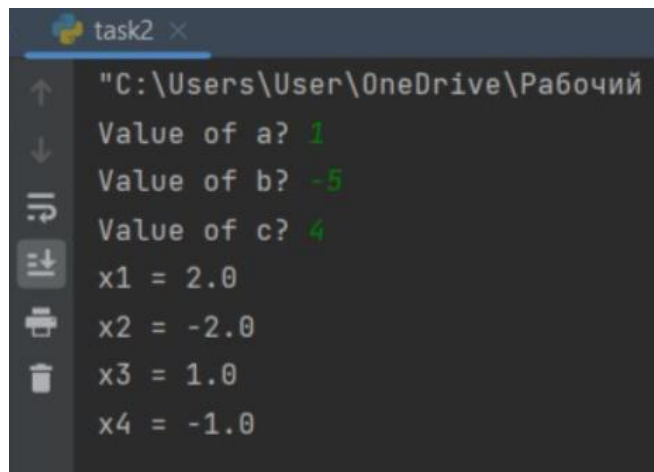


```

task2 x
"C:\Users\User\OneDrive\Рабочий стол\Lb3\
Value of a? 2
Value of b? 5
Value of c? 8
Действительных корней нет

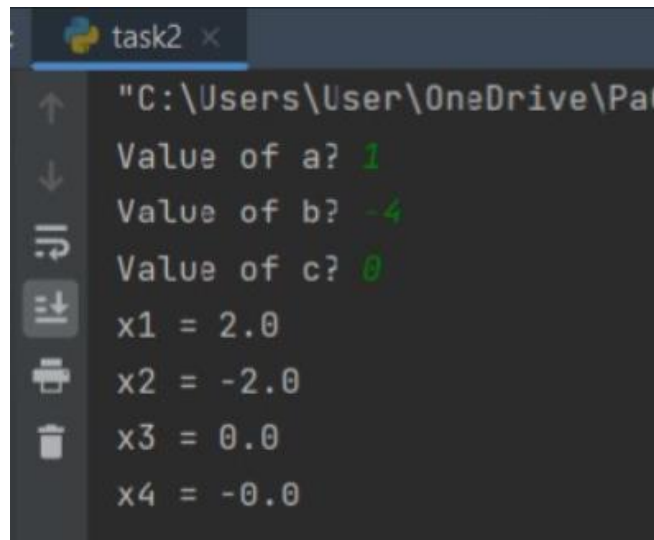
```

Рисунок 11 - Результат выполнения (1)



```
task2 x
"C:\Users\User\OneDrive\Рабочий
Value of a? 1
Value of b? -5
Value of c? 4
x1 = 2.0
x2 = -2.0
x3 = 1.0
x4 = -1.0
```

Рисунок 12 - Результат выполнения (2)



```
task2 x
"C:\Users\User\OneDrive\Раб
Value of a? 1
Value of b? -4
Value of c? 0
x1 = 2.0
x2 = -2.0
x3 = 0.0
x4 = -0.0
```

Рисунок 13 - Результат выполнения (3)

Задание 3. Определить среди всех двузначных чисел те, которые делятся на сумму своих цифр.

Решение: чтобы найти цифр числа, воспользуемся свойствами целочисленного деления и деления с остатком, с помощью цикла for переберём двузначные числа.

UML-диаграмма деятельности:

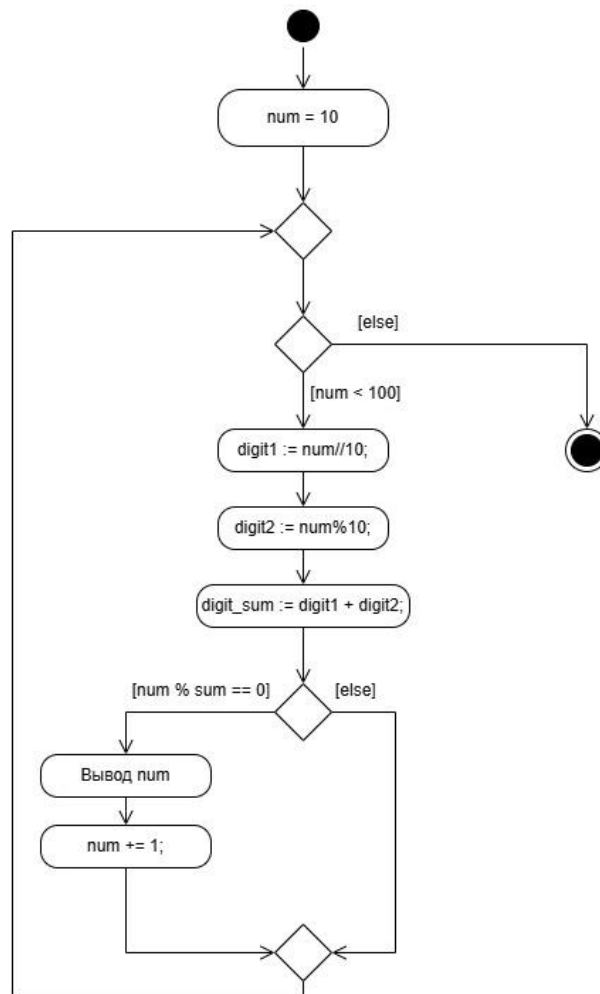


Рисунок 14 - UML-диаграмма для задания 3

Листинг программы:

```

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

if __name__ == '__main__':
    for num in range(10, 100):
        digit1 = num // 10
        digit2 = num % 10
        digit_sum = digit1 + digit2

        if num % digit_sum == 0:
            print(num)
  
```

Результат выполнения:

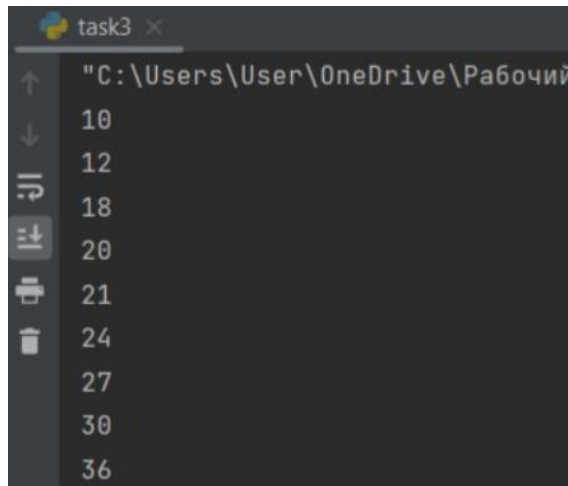


Рисунок 15 - Результат выполнения программы для задания 3

Задание 4 (повышенной сложности). Составить UML-диаграмму деятельности, программу и произвести вычисления значения специальной функции по ее разложению в ряд с точностью $\epsilon = 10^{-10}$, аргумент функции x вводится с клавиатуры.

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

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

Решение: для написания программы упростим второе слагаемое, то есть сумму от $k = 0$ до бесконечности. Для этого выпишем общий текущий член ряда, следующий член ряда и найдём их рекуррентное соотношение. После нахождения соотношения можно выразить следующий член ряда через текущий. Также найдём первое слагаемое, подставив 0 в текущий член ряда. С помощью цикла `while` и условия точности находим сумму ряда. В конце умножаем на первое слагаемое функции.

Текущий член ряда

$$a_k := \frac{\left(\frac{x^2}{4}\right)^k}{k! \cdot (k+n)!}$$

Следующий член ряда

$$a_{k+1} := \frac{\left(\frac{x^2}{4}\right)^{k+1}}{(k+1)! \cdot (k+1+n)!}$$

Рекуррентное соотношение

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

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

Первый член ряда

$$a_0 := \frac{1}{n!}$$

Рисунок 16 - Упрощение и преобразование ряда

UML-диаграмма деятельности:

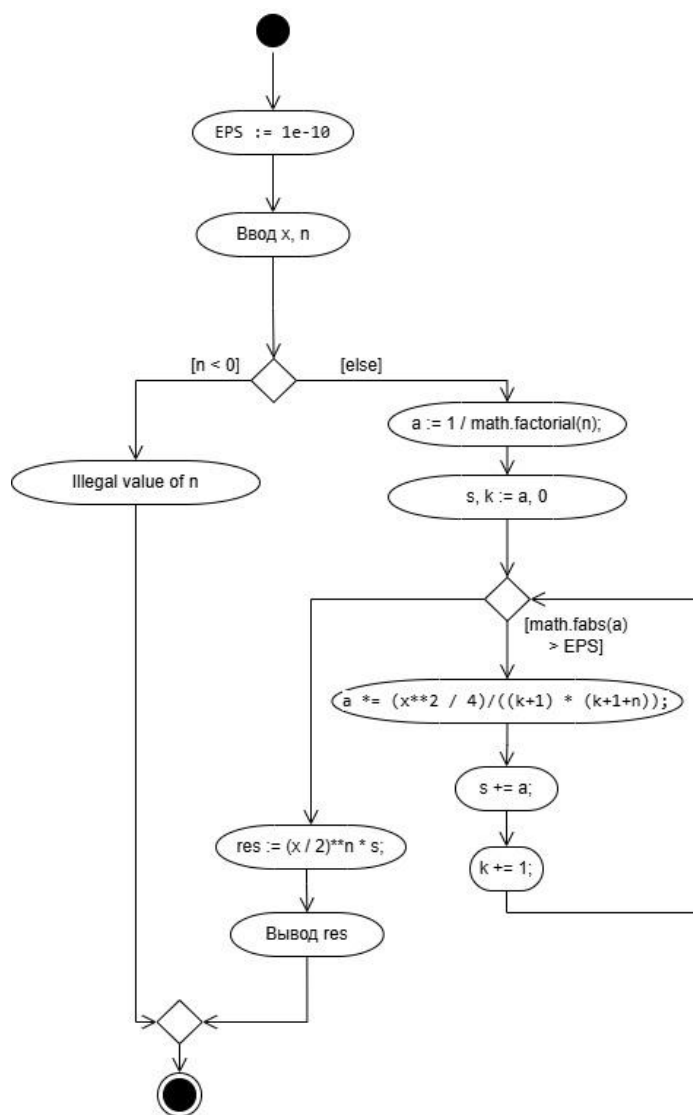


Рисунок 17 - UML-диаграмма для задания 4

Листинг программы:

```

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

import math
import sys

EPS = 1e-10

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

    a = 1 / math.factorial(n)
    S, k = a, 0
  
```



```

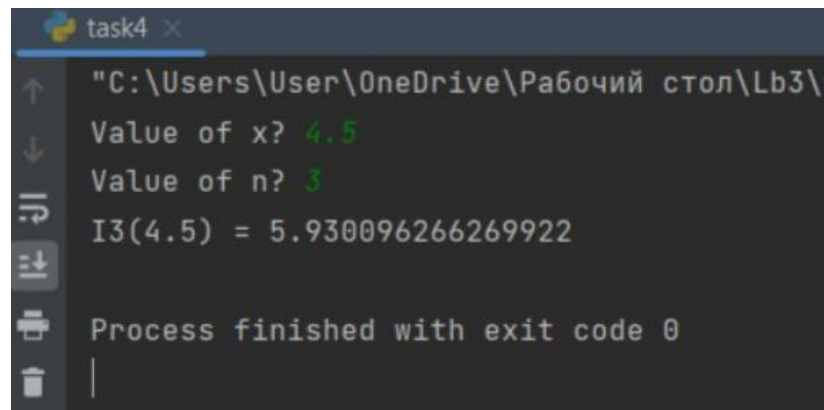
while math.fabs(a) > EPS:
    a *= (x**2 / 4) / ((k + 1)*(k + 1 + n))
    S += a
    k += 1

res = (x / 2) ** n * S

print(f"I{n}({x}) = {res}")

```

Результат выполнения:



```

task4 x
"C:\Users\User\OneDrive\Рабочий стол\Lb3\
Value of x? 4.5
Value of n? 3
I3(4.5) = 5.930096266269922
Process finished with exit code 0

```

Рисунок 18 - Результат выполнения программы для задания 4

Контрольные вопросы:

1. Для чего нужны диаграммы деятельности UML?

Диаграммы деятельности UML используются для моделирования бизнес-процессов, рабочих потоков и алгоритмов. Они показывают последовательность действий, условия перехода между ними и параллельные процессы.

2. Что такое состояние действия и состояние деятельности?

Состояние действия – элементарное неделимое действие в процессе выполнения. Состояние деятельности – составное состояние, которое может содержать вложенные действия и подпроцессы.

3. Какие нотации существуют для обозначения переходов и ветвлений в диаграммах деятельности?

- Ромб – узел ветвления/слияния
- Стрелки – переходы между действиями
- Условия [в квадратных скобках] – условия перехода

- Горизонтальные линии – ветвление/слияние потоков

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Алгоритм, в котором выполнение различных блоков команд зависит от выполнения определенных условий.

5. Чем отличается разветвляющийся алгоритм от линейного?

Линейный – команды выполняются последовательно одна за другой

Разветвляющийся – выполнение различных веток кода зависит от условий

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

Оператор, выполняющий различные действия в зависимости от условия.

Формы в Python:

if условие:

 действие

if условие:

 действие1

else:

 действие2

if условие1:

 действие1

elif условие2:

 действие2

else:

 действие3

7. Какие операторы сравнения используются в Python?

- == - равно
- != - не равно

- < - меньше
- > – больше
- <= – меньше или равно
- >= – больше или равно
- is – идентичность объектов
- in – принадлежность

8. Что называется простым условием? Приведите примеры.

Простое условие содержит одно логическое выражение.

Примеры:

`x > 0`

`age >= 18`

`name == "John"`

9. Что такое составное условие? Приведите примеры.

Составное условие объединяет несколько простых условий логическими операторами.

Примеры:

`x > 0 and x < 10`

`not is_weekend`

10. Какие логические операторы допускаются при составлении сложных условий?

- and – логическое И
- or – логическое ИЛИ
- not – логическое НЕ

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

Да, это называется вложенными условными операторами.

`if x > 0:`

`if y > 0:`

`print("Первая четверть")`

`else:`

```
print("Четвертая четверть")
```

12. Какой алгоритм является алгоритмом циклической структуры?

Алгоритм, в котором некоторая последовательность действий выполняется многократно до выполнения определенного условия.

13. Типы циклов в языке Python.

- for – для итерации по последовательностям
- while – выполняется пока условие истинно

14. Назовите назначение и способы применения функции range.

Функция range используется для генерации последовательности чисел.

Способы применения:

```
range(stop)
```

```
range(start, stop)
```

```
range(start, stop, step)
```

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

```
for i in range(15, -1, -2):
```

```
    print(i)
```

16. Могут ли быть циклы вложенными?

Да, циклы могут быть вложенными друг в друга.

```
for i in range(3):
```

```
    for j in range(3):
```

```
        print(f"i={i}, j={j}")
```

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

Бесконечный цикл – это цикл, который выполняется постоянно, без остановки, пока программа не будет прервана извне. Бесконечный цикл образуется, когда условие выхода из цикла никогда не становится ложным. Выйти из него можно с помощью оператора break или изменения условия.

18. Для чего нужен оператор break?

Оператор break в Python нужен для досрочного выхода из цикла, когда дальнейшее выполнение не имеет смысла.

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

Оператор `continue` употребляется внутри циклов. Он позволяет прервать текущую итерацию цикла и перейти к следующей. Оператор `continue` используется, когда нужно пропустить определённые значения или условия в цикле. Например, если необходимо пропустить обработку определённых данных или условий, `continue` позволяет сделать это без завершения всего цикла.

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

`stdout` нужен для вывода выходных данных, чаще всего в формате текста, что является результатом выполнения программы. По умолчанию поток нацелен на запись на устройство отображения (монитор); `stderr` нужен для вывода диагностических или отладочных сообщений, а также информации об ошибках в работе программы. Таким образом, `stdout` отвечает за основную работу программы, а `stderr` – за выявление и отображение возможных проблем в её работе.

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

Вывод в стандартный поток `stderr` можно организовать с помощью функции `print`, по умолчанию она выводит сообщения в `stdout`. Но с помощью аргумента `file` можно перенаправить вывод в любой другой поток, включая `stderr`. Пример: `import sys; print("Произошла ошибка", file=sys.stderr)`.

22. Каково назначение функции `exit`?

Назначение функции `exit()` в Python – немедленное завершение выполнения кода. Также функция `exit()` позволяет передать необязательный аргумент, который представляет состояние завершения программы. По соглашению, значение 0 указывает на успешное выполнение, а любое ненулевое значение – на ошибку или ненормальное завершение.

Вывод: были приобретены навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Освоены операторы языка Python версии 3.x `if`, `while`, `for`, `break` `continue`,

позволяющие реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.