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»**

**Вариант 23**

|  |  |  |
| --- | --- | --- |
|  | Выполнил:  Омонкулов Исомиддин Валижон угли  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 Р. А.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2024

**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.

Листинг для примера №1:

#!/usr/bin/env python 3  
# -\*- coding: utf-8 -\*-  
  
import math  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 x = float(input('Value of x? '))  
  
 if x <= 0:  
 y = 2 \* x \* x + math.cos(x)  
 elif x < 5:  
 y = x + 1  
 else:  
 y = math.sin(x) - x \* x  
  
 print(f'y = {y}')

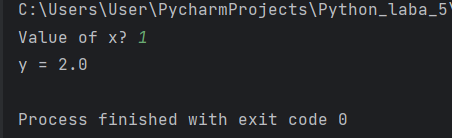


Рисунок 1. 1-ый тест для примера №1

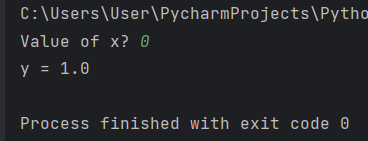


Рисунок 2. 2-ой тест для примера №1

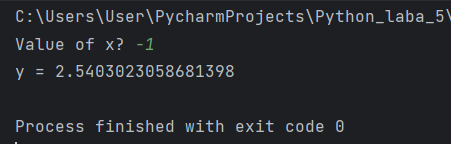


Рисунок 3. 3-ий тест для примера №1

Листинг для примера №2:

#!/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)

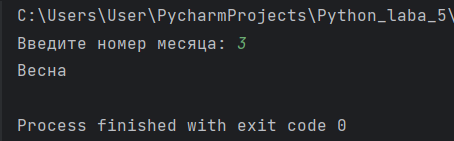


Рисунок 4. 1-ый тест для примера №2

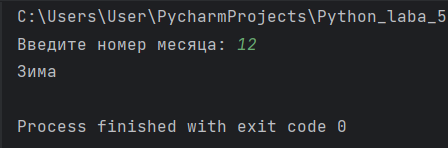


Рисунок 5. 2-ой тест для примера №2

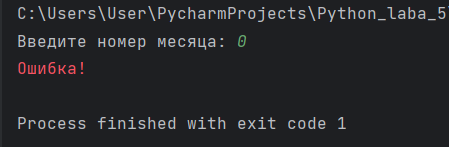


Рисунок 6. 3-ий тест для примера №2

Листинг для примера №3:

#!/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}')

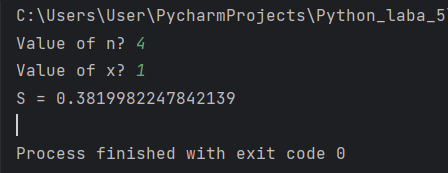


Рисунок 7. 1-ый тест для примера №3

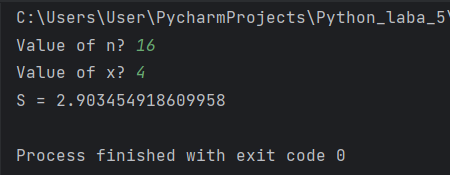


Рисунок 8. 2-ой тест для примера №3

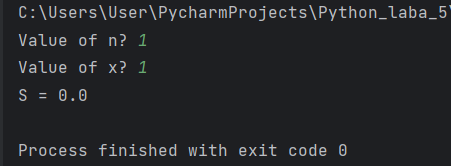


Рисунок 9. 3-ий тест для примера №3

Листинг для примера №4:

#!/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)}')

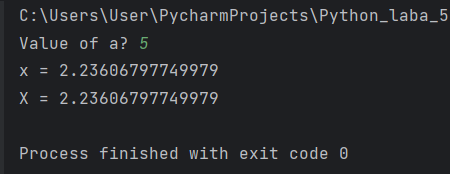


Рисунок 10. 1-ый тест для примера №4

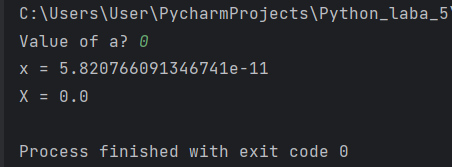


Рисунок 11. 2-ой тест для примера №4

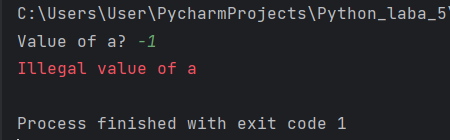


Рисунок 12. 3-ий тест для примера №4

Листинг для примера №5:

#!/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}')

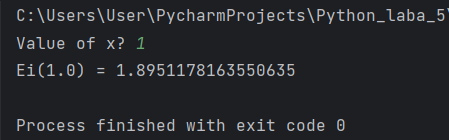


Рисунок 13. 1-ый тест для примера №5

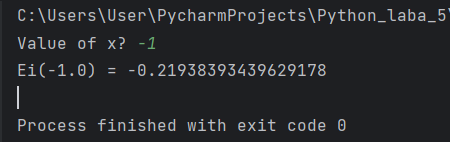


Рисунок 14. 2-ой тест для примера №5

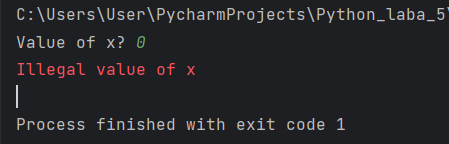


Рисунок 15. 3-ий тест для примера №5

Листинг к индивидуальному заданию №1:

#!/usr/bin/env python3  
# -\*- coding: utf-8 -\*-  
  
import sys  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 n = int(input('Enter the month number: '))  
 if n == 1:  
 print('It is January')  
 elif n == 2:  
 print('It is February')  
 elif n == 3:  
 print('It is March')  
 elif n == 4:  
 print('It is April')  
 elif n == 5:  
 print('It is May')  
 elif n == 6:  
 print('It is June')  
 elif n == 7:  
 print('It is July')  
 elif n == 8:  
 print('It is August')  
 elif n == 9:  
 print('It is September')  
 elif n == 10:  
 print('It is October')  
 elif n == 11:  
 print('It is November')  
 elif n == 12:  
 print('It is December')  
 else:  
 print('Ошибка!', file=sys.stderr)  
 exit(1)

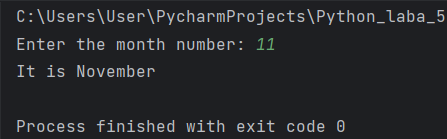


Рисунок 16. Результат индивидуальной программы №1

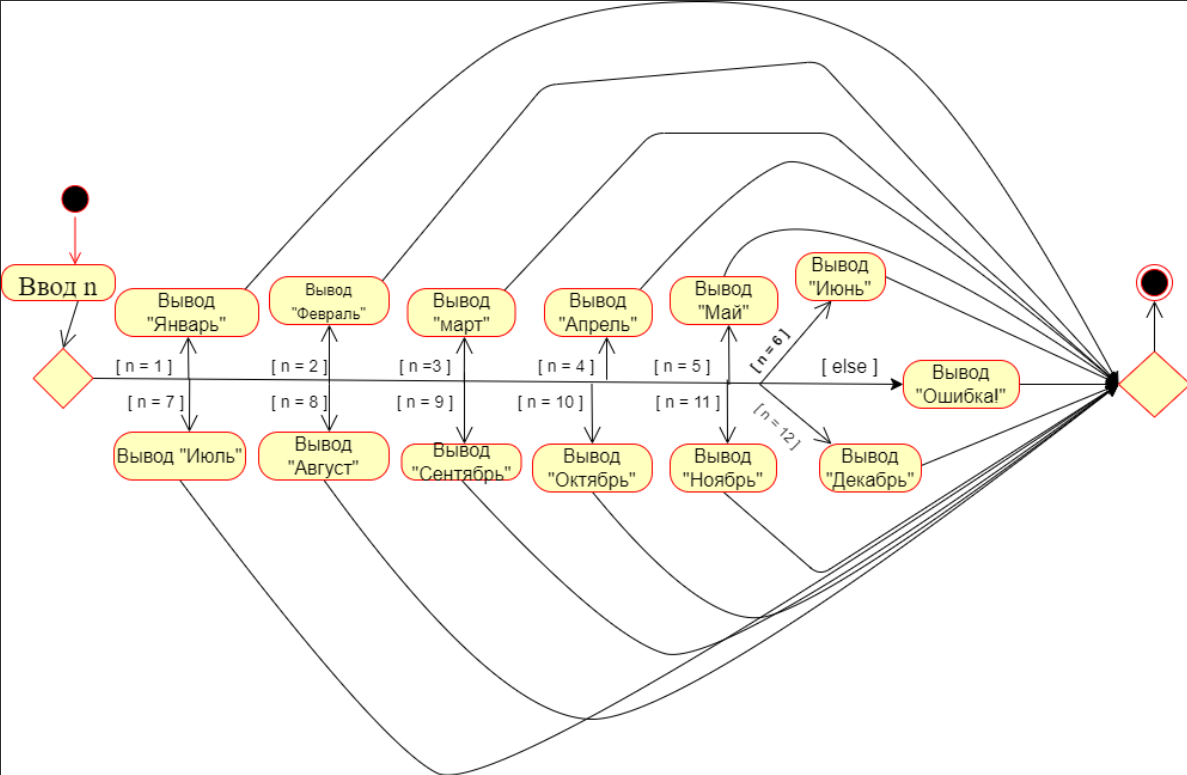


Рисунок 17. UML-диаграмма для индивидуальной программы №1

Листинг к индивидуальному заданию №2:

#!/usr/bin/env python3  
# -\*- coding: utf-8 -\*-  
  
from math import \*  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 a = float(input('Enter the coefficient "a": '))  
 b = float(input('Enter the coefficient "b": '))  
 c = float(input('Enter the coefficient "c": '))  
  
 if a == 0:  
 print('The equation is not biquadrate.')  
 else:  
 discriminant = b \*\* 2 - 4 \* a \* c  
  
 if discriminant < 0:  
 print('The equation has no real roots.')  
 else:  
 t\_1 = (-b + sqrt(discriminant)) / (2 \* a)  
 t\_2 = (-b - sqrt(discriminant)) / (2 \* a)  
  
 if discriminant == 0:  
 print('The equation has two matching real roots:')  
 x\_1 = sqrt(t\_1)  
 x\_2 = -sqrt(t\_1)  
 print('x\_1 =', x\_1)  
 print('x\_2 =', x\_2)  
 else:  
 print('The equation has four real roots:')  
 x\_1 = sqrt(t\_1)  
 x\_2 = -sqrt(t\_1)  
 x\_3 = sqrt(t\_2)  
 x\_4 = -sqrt(t\_2)  
 print('x\_1 =', x\_1)  
 print('x\_2 =', x\_2)  
 print('x\_3 =', x\_3)  
 print('x\_4 =', x\_4)

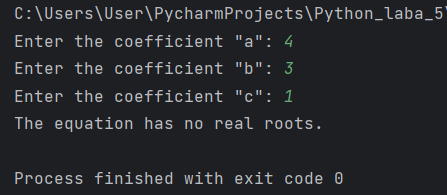


Рисунок 18. Тест №1

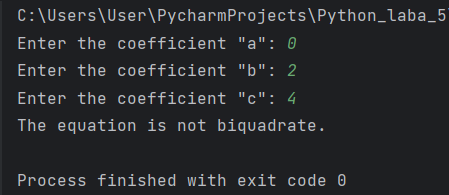


Рисунок 19. Тест №2

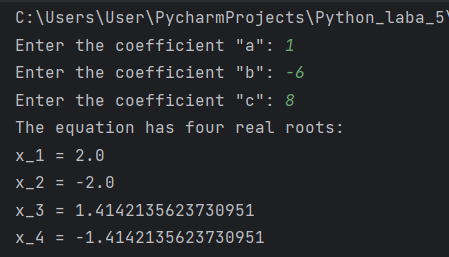


Рисунок 20. Тест №3

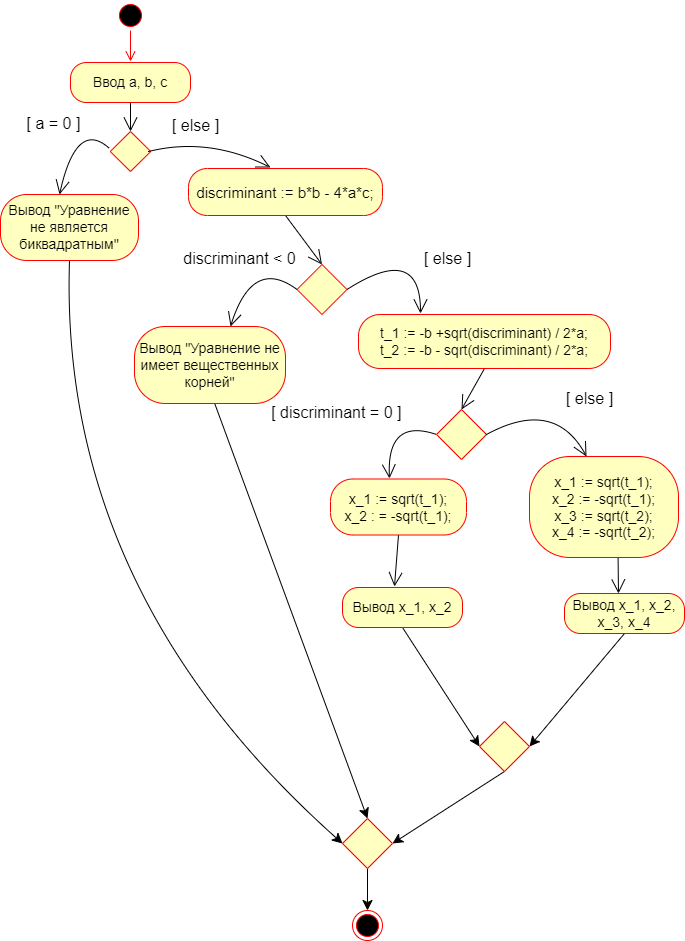


Рисунок 21. UML-диаграмма для индивидуального задания №2

Листинг к индивидуальному заданию №3:

#!/usr/bin/env python3  
# -\*- coding: utf-8 -\*-  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print('All two-digit numbers that are divisible by the sum of the digits they consist of:')  
  
 for i in range(10, 100):  
 last\_digit = i % 10  
 first\_digit = i // 10  
  
 if i % (last\_digit + first\_digit) == 0:  
 print(i)

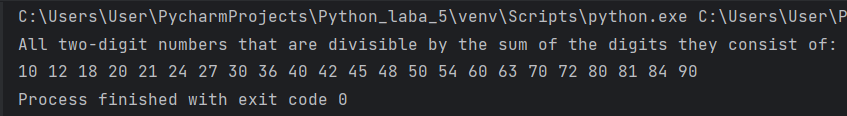


Рисунок 22. Результат индивидуальной программы №3

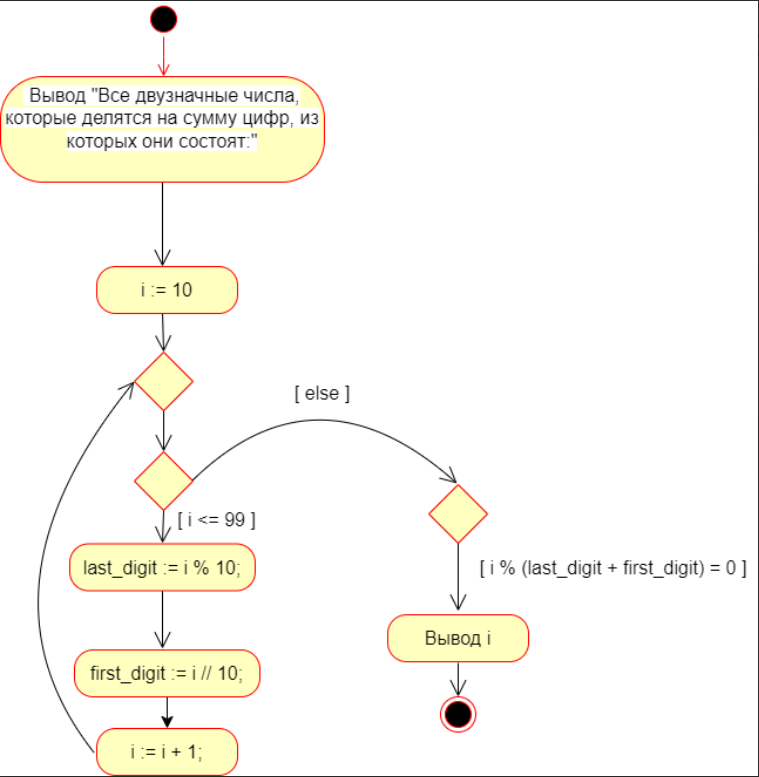


Рисунок 23. UML-диаграмма для индивидуального задания №3

Листинг к программе повышенной сложности:

#!/usr/bin/env python3  
# -\*- coding: utf-8 -\*-  
  
import math  
import sys  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 n = int(input("Enter the value n: "))  
 x = float(input("Enter the value of the function argument: "))  
  
 if x == 0:  
 print('Недопустимое значение x', file=sys.stderr)  
 exit(1)  
 else:  
 EPS = float(input("Enter the accuracy value: "))  
 result = 0  
 k = 0  
 tekushee\_slagaemoe = 1 / math.factorial(n)  
  
 while math.fabs(tekushee\_slagaemoe) > EPS:  
 tekushee\_slagaemoe \*= x \*\* 2 / (4 \* (k + n + 1) \* (k + 1))  
 result += tekushee\_slagaemoe  
 k += 1  
  
 print(f'I{n}({x}) = {((x / 2) \*\* n) \* result}')

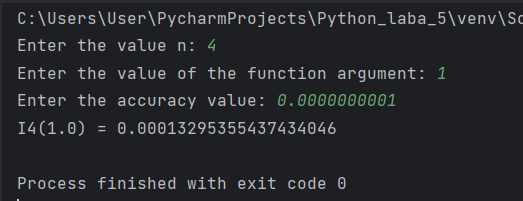


Рисунок 25. Тест №1

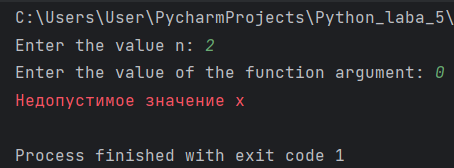


Рисунок 26. Тест №2

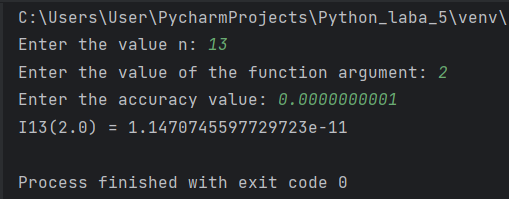


Рисунок 27. Тест №3

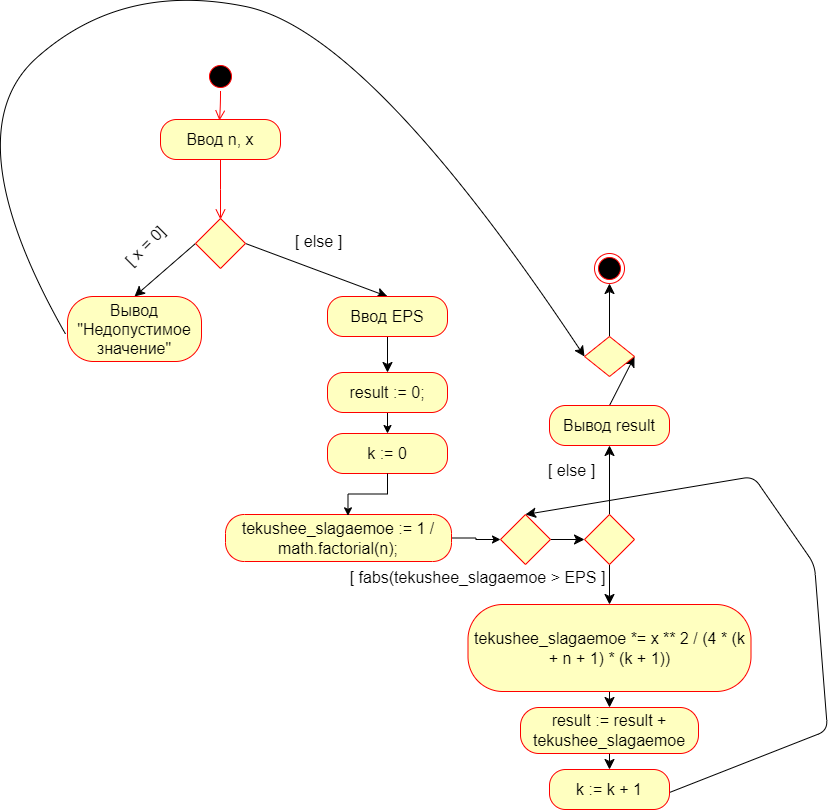


Рисунок 28. UML-диаграмма для программы повышенной сложности

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

**Ответы на контрольные вопросы**

1. Диаграммы деятельности UML используются для визуализации процессов и действий в системе, чтобы лучше понять их последовательность и взаимодействие.

2. Состояние действия - это состояние, в котором объект находится в процессе выполнения определенной деятельности. Состояние деятельности - это состояние, в котором объект находится, когда он выполняет определенную деятельность.

3. Для обозначения переходов и ветвлений в диаграммах деятельности используются стрелки и различные символы, такие как ромбы для ветвлений и пересечения линий для переходов.

4. Алгоритм разветвляющейся структуры - это алгоритм, в котором возможно выполнение различных последовательностей действий в зависимости от условий.

5. Разветвляющийся алгоритм позволяет выполнить различные последовательности действий в зависимости от условий, в то время как линейный алгоритм выполняет действия последовательно, без разветвлений.

6. Условный оператор - это конструкция в программировании, которая позволяет выполнить определенные действия в зависимости от условия. Его формы включают if-else, if-elif-else и switch-case.

7. В Python используются операторы сравнения: == (равно), != (не равно), < (меньше), > (больше), <= (меньше или равно), >= (больше или равно).

8. Простое условие - это условие, содержащее только одно логическое выражение. Например, if x > 5: print("x больше 5").

9. Составное условие - это условие, содержащее несколько логических выражений, объединенных логическими операторами. Например, if x > 5 and y < 10: print("x больше 5 и y меньше 10").

10. При составлении сложных условий в Python можно использовать логические операторы: and (и), or (или), not (не).

11. Да, оператор ветвления может содержать внутри себя другие ветвления, создавая так называемые вложенные условия.

12. Алгоритм циклической структуры - это алгоритм, выполняющий определенную последовательность действий несколько раз.

13. В языке Python существуют циклы while и for.

14. Функция range используется для создания последовательности чисел. Она может применяться для организации циклов и перебора значений.

15. Для организации перебора значений от 15 до 0 с шагом 2 с помощью функции range можно использовать следующий код: for i in range(15, 0, -2): print(i).

16. Да, циклы могут быть вложенными, то есть один цикл может находиться внутри другого.

17. Бесконечный цикл образуется, когда условие цикла всегда остается истинным. Чтобы выйти из бесконечного цикла, можно использовать оператор break или изменить условие цикла.

18. Оператор break используется для выхода из цикла досрочно, без выполнения оставшихся итераций.

19. Оператор continue используется для пропуска текущей итерации цикла и перехода к следующей итерации.

20. Стандартные потоки stdout (стандартный вывод) и stderr (стандартный вывод ошибок) используются для вывода информации и ошибок соответственно.

21. В Python вывод в стандартный поток stderr можно организовать с помощью модуля sys: import sys sys.stderr.write("Ошибка\n").

22. Функция exit используется для завершения выполнения программы с заданным кодом завершения.