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 института перспективной инженерии

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №3
дисциплины «Программирование на Python»
«Условные операторы и циклы в языке Python»
Вариант 12

Выполнила:
Коробка В.А.,
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.

Цель: 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.

Порядок выполнения работы

Адрес репозитория: https://github.com/13r4lera/Python3.git

Был создан общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python, выполнено клонирование данного репозитория. Файл .gitignore был дополнен правилами, необходимыми для работы с PyCharm (раскомментирована строка с .idea/).

Репозиторий был организован в соответствии с моделью ветвления git-flow. Была создана ветка develop от основной ветки main (git checkout -b develop). Затем ветка develop была добавлена на удаленном репозитории GitHub.

```
nout1@MSI MINGW64 ~/Documents/university/PYTHON/3/Python3 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

nout1@MSI MINGW64 ~/Documents/university/PYTHON/3/Python3 (develop)
$ git push origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:      https://github.com/13r4lera/Python3/pull/new/develop
remote:
To https://github.com/13r4lera/Python3.git
 * [new branch]      develop -> develop

nout1@MSI MINGW64 ~/Documents/university/PYTHON/3/Python3 (develop)
$ ^C

nout1@MSI MINGW64 ~/Documents/university/PYTHON/3/Python3 (develop)
$
```

Рисунок 1. Создание ветки develop

Был создан проект PyCharm в папке репозитория. В нем были созданы модули для каждого примера, изменения зафиксированы в репозитории. В первом примере составлена программа с использованием конструкции if-elif-else, вычисляющая значение функции в зависимости от ввода пользователем значения x. Для вывода значения y используются f-строки.

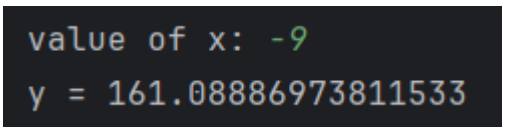

Рисунок 2. Результат выполнения первого примера при отрицательном x

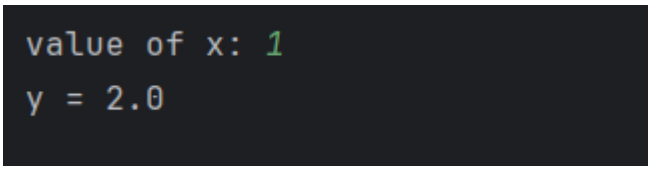

Рисунок 3. Результат выполнения примера при значении x меньше 5

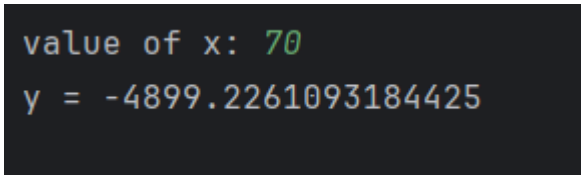

Рисунок 4. Результат выполнения примера при значении x больше 5

Полный код примера:

```
#!/usr/bin/env python3
# -*- 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}")
```

Во втором примере составлена программа, в которой пользователь должен с клавиатуры ввести номер месяца от 1 до 12. Затем программа выводит наименование времени года для месяца с этим номером. Вывод ошибок происходит с помощью print("Ошибка!", file=sys.stderr). После этого выполняется вызов exit(1), что приводит к немедленному завершению программы и операционной системе передается 1 в качестве кода возврата.

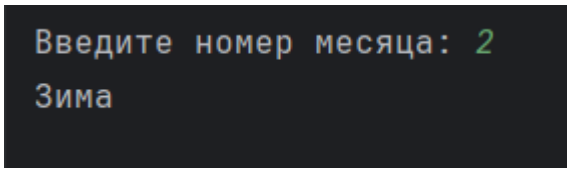

Рисунок 5. Результат работы примера при корректном вводе месяца

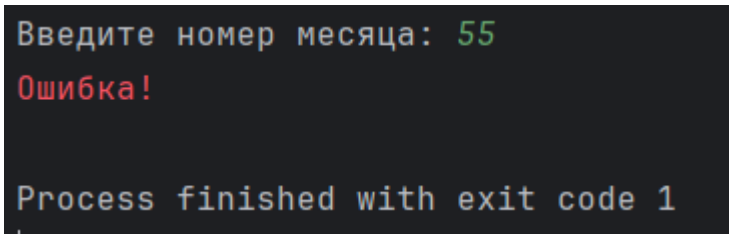

Рисунок 6. Результат работы примера при ошибочном вводе месяца

Полный код примера:

```
#!/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)
```

В примере 3 написана программа, позволяющая вычислить конечную сумму:

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

где n и k вводятся с клавиатуры. В цикле for перебираются значения от 1 до n, вычисляется член ряда и вычисленное значение прибавляется к сумме. После завершения цикла программа выводит итоговое значение суммы.

```
Value of n: 5
Value of x: 7
s = 3.294431456662813

Process finished with exit code 0
```

Рисунок 7. Результат работы примера 3

Полный код примера:

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

В примере 4 вычисляется значение квадратного корня положительного числа, вводимого с клавиатуры с некоторой заданной точностью с помощью рекуррентного соотношения:

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

```
Value of a: 36
x = 6.0
X = 6.0

Process finished with exit code 0
```

Рисунок 8. Результат работы примера 4 при вводе квадрата числа

```
Value of a: 15
x = 3.872983346207417
X = 3.872983346207417

Process finished with exit code 0
```

Рисунок 9. Результат работы примера 4 при вводе положительного числа

```
Value of a: -5
Value of a must be positive

Process finished with exit code 1
```

Рисунок 10. Результат работы примера при вводе отрицательного числа

Полный код примера:

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

```
import math
import sys


if __name__ == '__main__':
    a = float(input("Value of a: "))
    if a < 0:
        print("Value of a must be positive", 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)}")
```

В примере 5 вычисляется значение специальной (интегральной показательной) функции

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

где γ = 0,5772156649… - постоянная Эйлера, по ее разложению в ряд с точностью $\varepsilon = 10^{-10}$, аргумент x вводится с клавиатуры.

```
Value of x:-5
Ei(-5.0) = -0.0011482955950130602

Process finished with exit code 0
```

Рисунок 11. Корректный ввод значения x

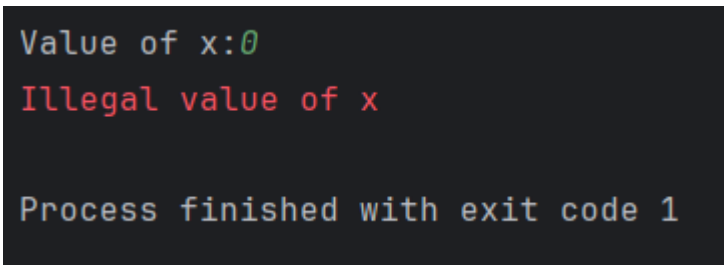

Рисунок 12. Вывод ошибки при вводе нулевого значения

Полный код примера:

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

import math
import sys

#Постоянная Эйлера.
EULER = 0.5772156649015328696
#Тончость 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}")
```

Для примеров 4 и 5 были построены UML-диаграммы деятельности.

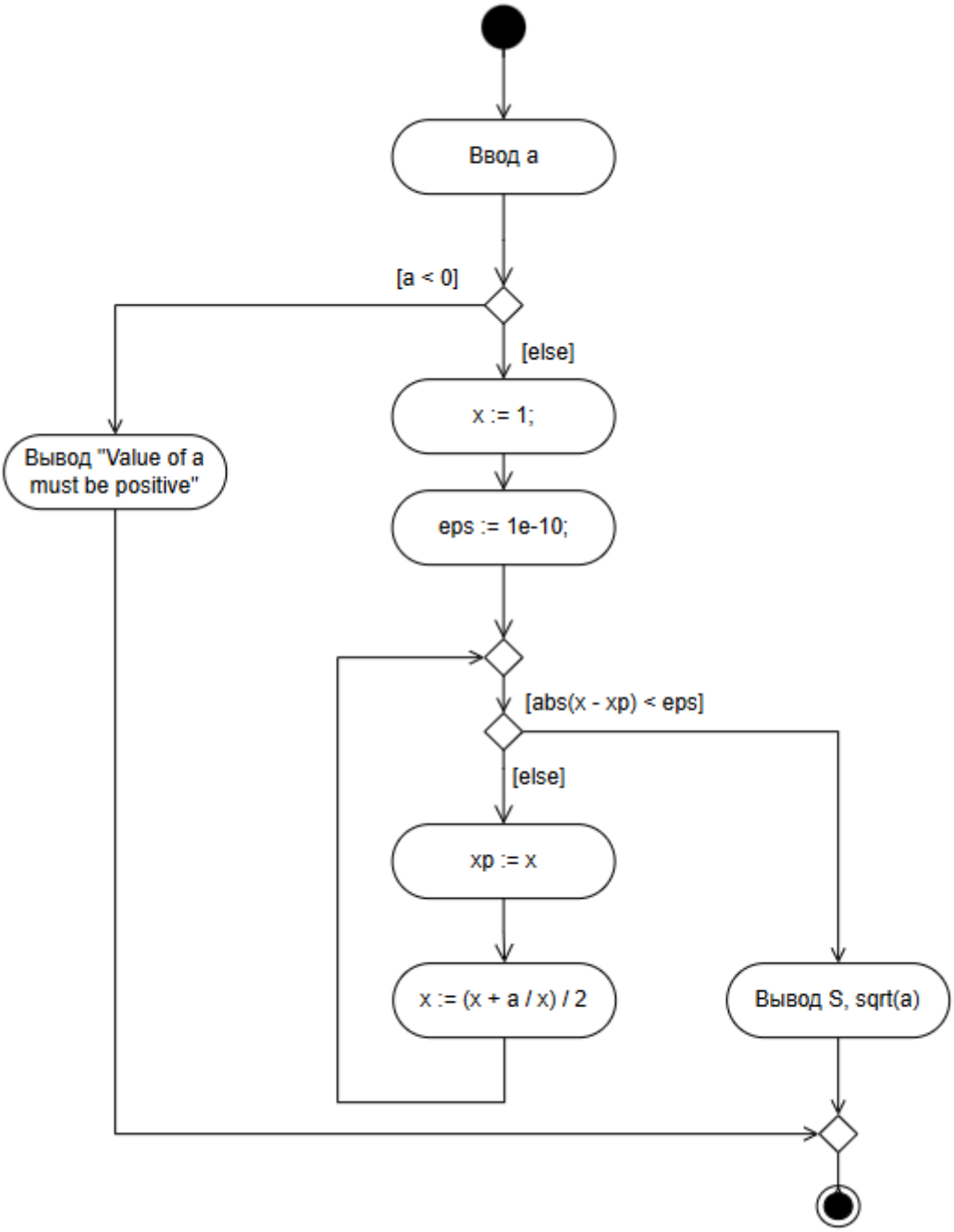

Рисунок 13. UML-диаграмма деятельности для примера 4

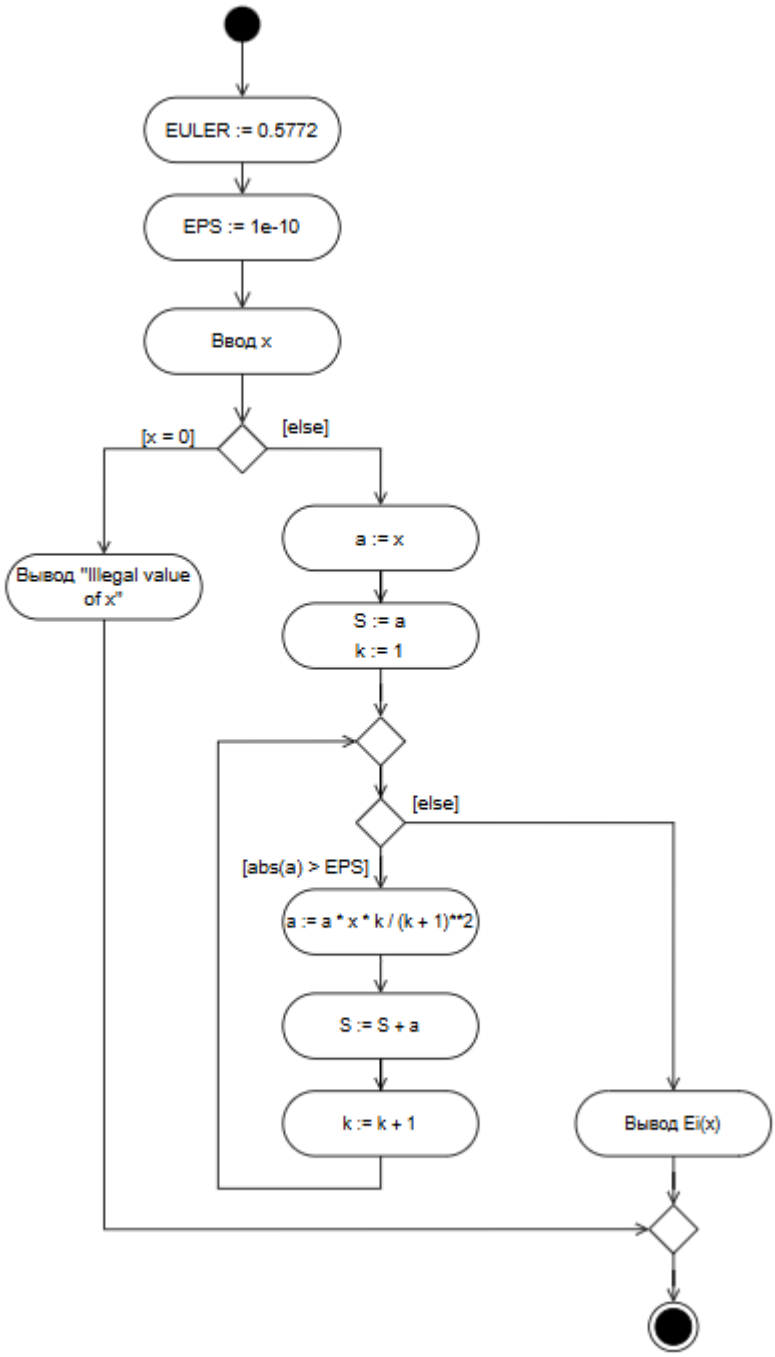

Рисунок 14. UML-диаграмма деятельности для примера 5

Условие индивидуального задания по варианту 12: При покупке товара на сумму от 200 до 500 руб предоставляется скидка 3%, при покупке товара на сумму от 500 до 800 - скидка 5%, при покупке товара на сумму от 800 до 1000 руб - скидка 7%, свыше 1000 руб - скидка 9%. Покупатель приобрел 8 рулонов обоев по цене X_1 и две банки краски по цене X_2. Сколько он заплатил? Была составлена UML-диаграмма деятельности по задаче.

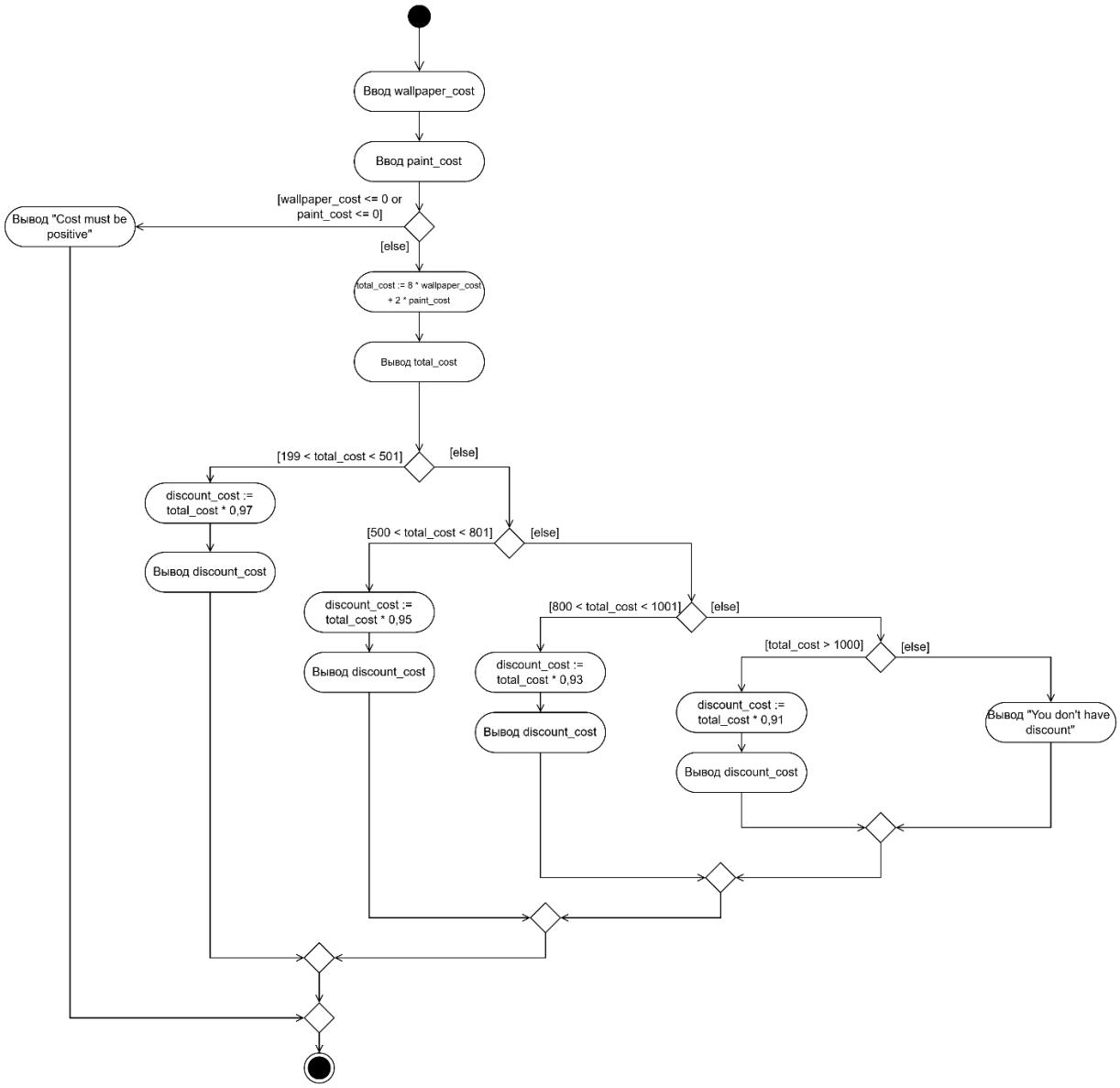

Рисунок 15. UML-диаграмма для индивидуального задания

Для решения задачи была написана программа, в которой пользователь вводит стоимость одного рулона обоев и одной банки краски. Если хотя бы одно из введенных чисел неположительное, то программа выводит сообщение об ошибке и завершает работу. Затем рассчитывается итоговая стоимость материалов и в зависимости от полученной суммы применяется скидка. Выводится итоговая стоимость без скидки, а затем стоимость со скидкой или информация о том, что скидки нет.

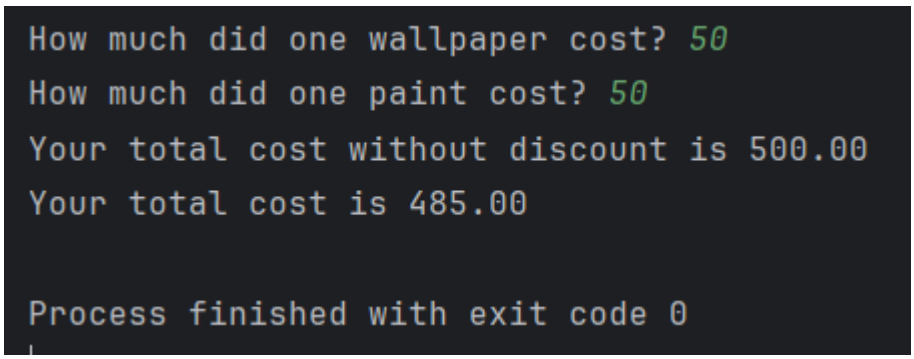

Рисунок 16. Пример работы программы

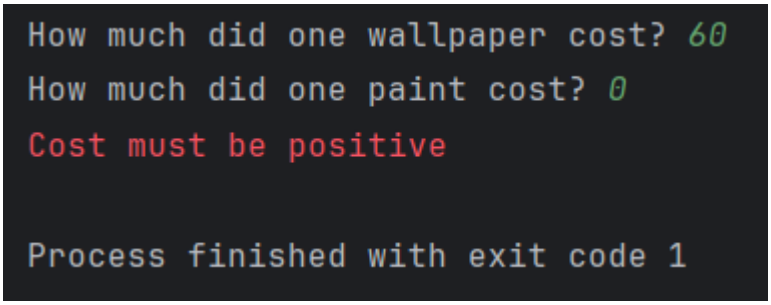

Рисунок 17. Вывод ошибки о неправильном вводе

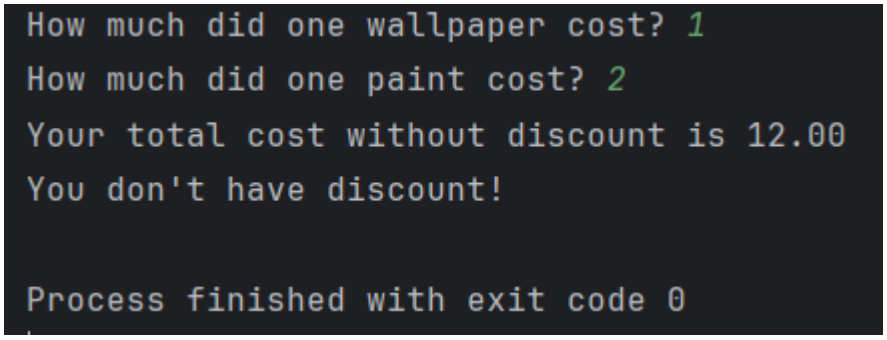

Рисунок 18. Вывод сообщения об отсутствии скидки

Полный код программы:

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

import sys


wallpaper_cost = float(input("How much did one wallpaper cost? "))
paint_cost = float(input("How much did one paint cost? "))

if wallpaper_cost <= 0 or paint_cost <= 0:
```

```
    print("Cost must be positive", file=sys.stderr)
    exit(1)

total_cost = 8 * wallpaper_cost + 2 * paint_cost
print(f"Your total cost without discount is {total_cost:.2f}")

if 199 < total_cost < 501:
    discount_cost = total_cost * 0.97
    print(f"Your total cost is {discount_cost:.2f}")
elif 500 < total_cost < 801:
    discount_cost = total_cost * 0.95
    print(f"Your total cost is {discount_cost:.2f}")
elif 800 < total_cost < 1001:
    discount_cost = total_cost * 0.93
    print(f"Your total cost is {discount_cost:.2f}")
elif total_cost > 1000:
    discount_cost = total_cost * 0.91
    print(f"Your total cost is {discount_cost:.2f}")
else:
    print("You don't have discount!")
```

Условие индивидуального задания 2 по варианту 12: 12. Две окружности заданы координатами центра и радиусами. Сколько точек пересечения имеют эти окружности? Для решения задачи была написана программа, которая сначала считывает с клавиатуры 6 значений (координаты центров окружностей и радиусы). Если были введены отрицательные значения радиусов, то выводится сообщение об ошибке и работа программы завершается. По формуле вычисляется расстояние между центрами, и затем это расстояние сравнивается с суммой и разностью радиусов, на основании чего и выводится результат.

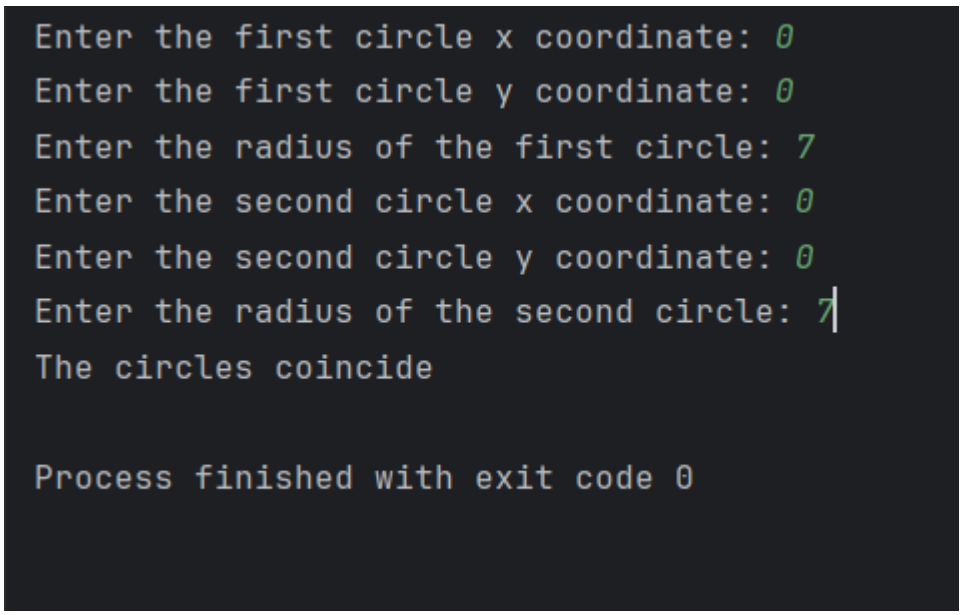

Рисунок 19. Результат работы программы при совпадающих окружностях

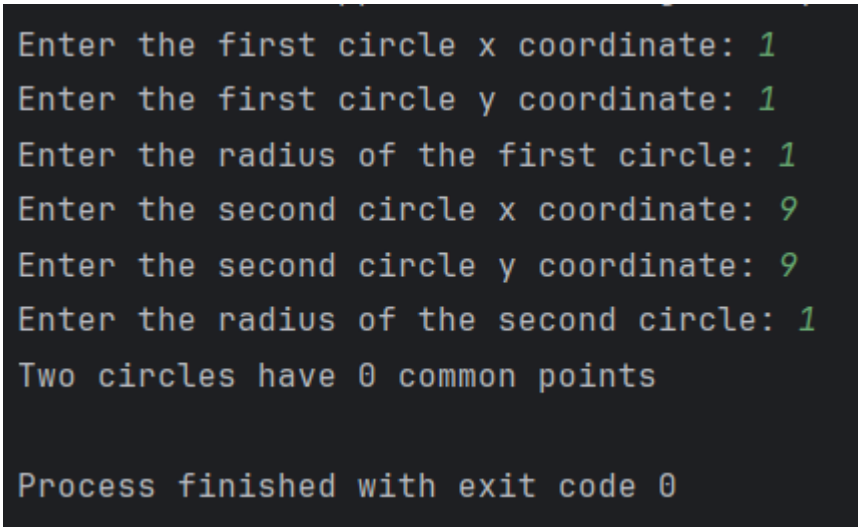

Рисунок 20. Результат работы программы, если окружности не пересекаются

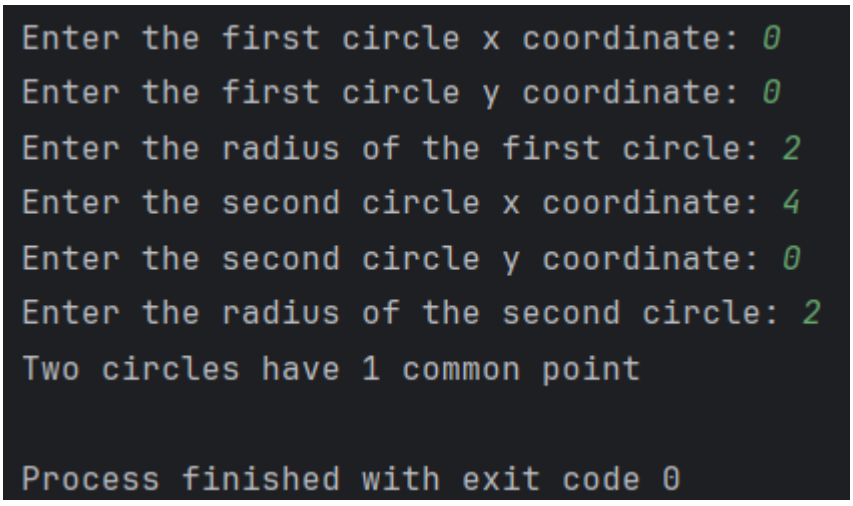

Рисунок 21. Результат работы программы, если окружности касаются

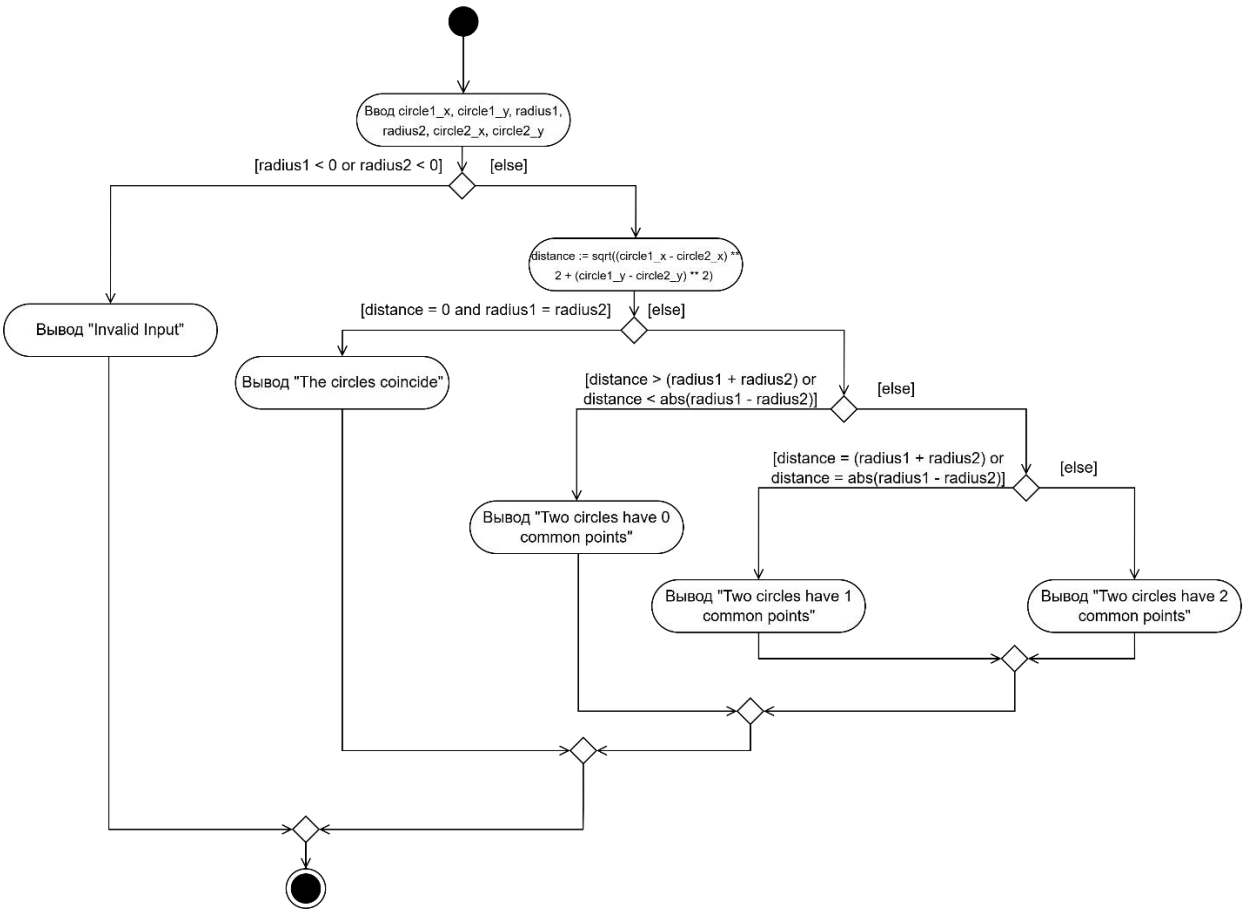

Рисунок 22. UML-диаграмма деятельности для задания 2

Полный код программы:

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

import math
import sys



if __name__ == '__main__':
    circle1_x = float(input("Enter the first circle x coordinate: "))
    circle1_y = float(input("Enter the first circle y coordinate: "))
    radius1 = float(input("Enter the radius of the first circle: "))

    circle2_x = float(input("Enter the second circle x coordinate: "))
```

```
circle2_y = float(input("Enter the second circle y coordinate: "))
radius2 = float(input("Enter the radius of the second circle: "))

if radius1 < 0 or radius2 < 0:
    print("Invalid input", file=sys.stderr)
    sys.exit(1)

distance = math.sqrt((circle1_x - circle2_x) ** 2 +
                     (circle1_y - circle2_y) ** 2)

if distance == 0 and radius1 == radius2:
    print("The circles coincide")
elif distance > (radius1 + radius2) or distance < abs(radius1 - radius2):
    print("Two circles have 0 common points")
elif distance == (radius1 + radius2) or distance == abs(radius1 - radius2):
    print("Two circles have 1 common point")
else:
    print("Two circles have 2 common points")
```

Условие индивидуального задания 3: Покупатель должен заплатить в кассу S р. У него имеются 1, 2, 5, 10, 100, 500 р. Сколько купюр разного достоинства отдаст покупатель, если он начинает платить с самых крупных?

Для решения была написана программа на Python. Сначала в программе запрашивает у пользователя сумму, а затем поочередно проверяется каждый номинал купюр. Для каждого номинала вычисляется необходимое количество купюр, это значение прибавляется к общему числу купюр. Программа выводит число купюр каждого номинала и общее число купюр.

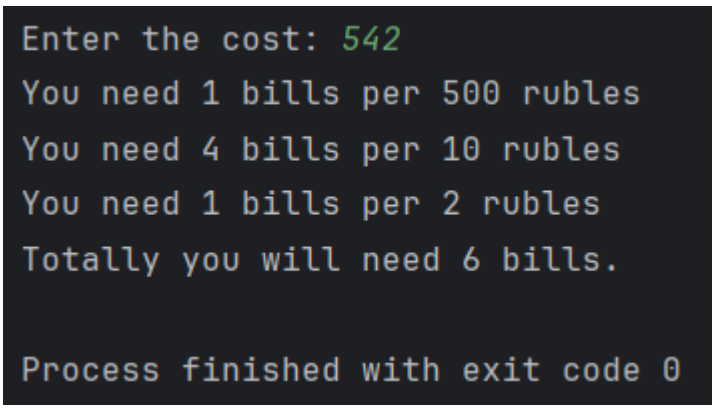

Рисунок 23. Результат работы программы

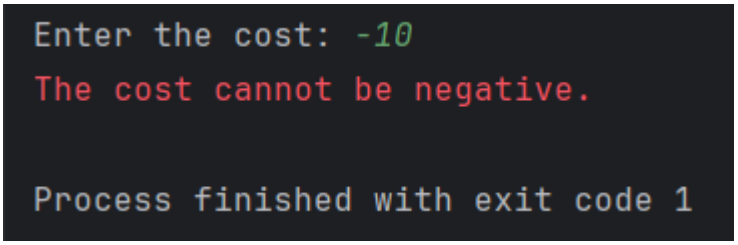

Рисунок 24. Вывод ошибки

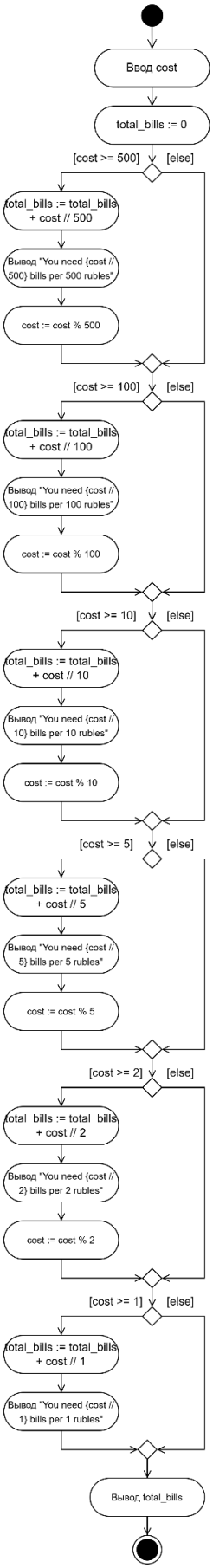

Рисунок 25. UML-диаграмма для индивидуального задания 3

Полный код программы:

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

import sys



if __name__ == "__main__":
    cost = int(input("Enter the cost: "))

    if cost < 0:
        print("The cost cannot be negative.", file=sys.stderr)
        exit(1)

    total_bills = 0
    if cost >= 500:
        total_bills += cost // 500
        print(f"You need {cost // 500} bills per 500 rubles")
        cost = cost % 500
    if cost >= 100:
        total_bills += cost // 100
        print(f"You need {cost // 100} bills per 100 rubles")
        cost = cost % 100
    if cost >= 10:
        total_bills += cost // 10
        print(f"You need {cost // 10} bills per 10 rubles")
        cost = cost % 10
    if cost >= 5:
        total_bills += cost // 5
        print(f"You need {cost // 5} bills per 5 rubles")
```

```
    cost = cost % 5
if cost >= 2:
    total_bills += cost // 2
    print(f"You need {cost // 2} bills per 2 rubles")
    cost = cost % 2
if cost >= 1:
    total_bills += cost // 1
    print(f"You need {cost // 1} bills per 1 ruble")


print(f"Totally you will need {total_bills} bills.")
```

Задание повышенной сложности по варианту 12: Интегральный гиперболический синус

$$Shi(x) = \int_0^x \frac{sh\,x}{t}dt = \sum_{n=1}^{\infty} \frac{x^{2n+1}}{(2n+1)(2n+1)!}$$

Для выполнения задания было найдено рекуррентное соотношение, позволяющее определить следующий член ряда исходя из значения текущего:

$$a_{n+1} = a_n \frac{x^2 \cdot (2n+1)}{(2n+3)^2(2n+2)}$$

Был найден первый член ряда при n = 0 a_0 = x.

```
x = 5
Shi(5.0) = 20.093211825696482

Process finished with exit code 0
```

Рисунок 26. Результат работы программы

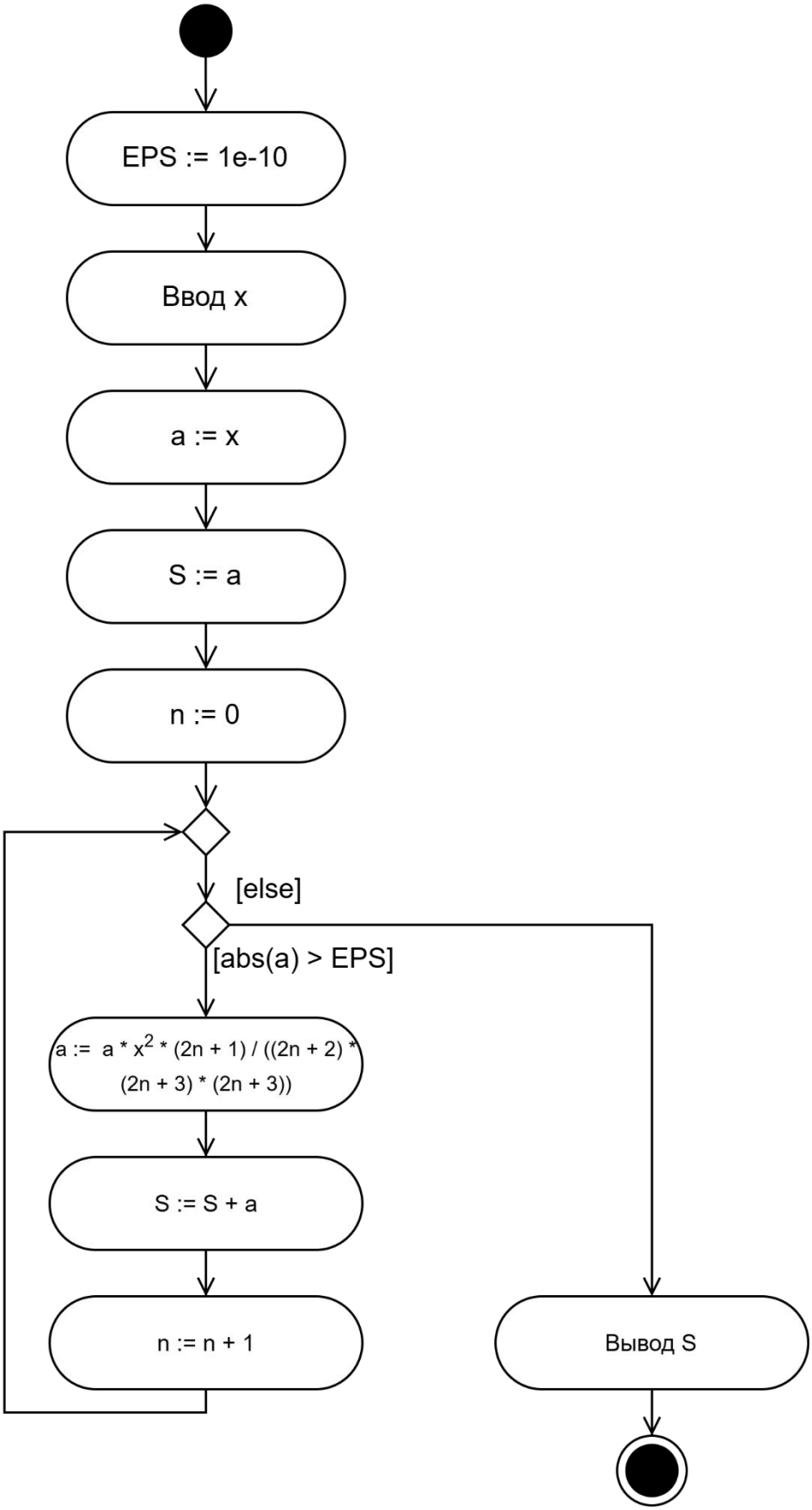

Рисунок 27. UML-диаграмма деятельности для задания повышенной сложности

Полный код программы:

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

import math

#Точность вычислений
EPS = 1e-10

if __name__ == "__main__":
    x = float(input("x = "))

    # Первый член ряда: a0 = x
    a = x
    S = a
    n = 0

    #Найти сумму членов ряда
    #Рекуррентная формула: a_n+1 = a_n * x^2 * (2n + 1)/ (2n+2)(2n+3)^2
    while math.fabs(a) > EPS:
        a *= x * x * (2 * n + 1) / ((2 * n + 2) * (2 * n + 3) * (2 * n + 3))
        S += a
        n += 1

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

Ответы на контрольные вопросы

1. Для чего нужны диаграммы деятельности UML?

Диаграммы деятельности языка UML предназначены для графического представления динамики поведения системы: бизнес-процессов, потоков работ, алгоритмов, последовательности выполнения действий и

взаимодействия объектов. Они используются для моделирования логики выполнения сценариев использования, методов классов, а также сложных бизнес-правил, включая условные переходы, параллельное выполнение и циклы.

2. Что такое состояние действия и состояние деятельности?

В потоке управления, моделируемом диаграммой деятельности происходят различные события (вычисление выражения, выполнение операции над объектом, послание сигнала). Выполняемые атомарные вычисления называются состояниями действия, поскольку каждое из них есть состояние системы, представляющее собой выполнение некоторого действия. 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.

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. 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.

5. Чем отличается разветвляющийся алгоритм от линейного?

В программе разветвляющейся структуры имеется один или несколько условных операторов. Для программной реализации условия используется логическое выражение.

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

Оператор ветвления if позволяет выполнить определенный набор инструкций в зависимости от некоторого условия. Если выражение из условия истинно, то выполняются инструкции, определяемые данным оператором. Бывают случаи, когда необходимо при истинном условии выполнить один набор инструкций, а при ложном - другой. Условие такого вида можно записать как тернарное выражение. Для этого используется конструкция if - else. Для реализации выбора из нескольких альтернатив можно использовать конструкцию if - elif - else.

7. Какие операторы сравнения используются в Python?

Операторы сравнения в Python: > (больше), < (меньше), >= (больше или равно), <= (меньше или равно), == (равно), != (не равно).

8. Что называется простым условием? Приведите примеры.

Логические выражения являются простыми, если в них выполняется только одна логическая операция. Например, kByte >= 1023.

9. Что такое составное условие? Приведите примеры.

Может понадобиться получить ответ "Да" или "Нет" в зависимости от результата выполнения двух простых выражений. В таких случаях широко используются специальные операторы, объединяющие два и более простых логических выражения. Например, y < 15 and x > 8.

10. Какие логические операторы допускаются при составлении сложных условий?

Широко используются два оператора - так называемые логические И (and) и ИЛИ (or). Чтобы получить True при использовании оператора and, необходимо, чтобы результат обоих простых выражений, которые связывает данный оператор, были истинными. Чтобы получить True при использовании оператора or, необходимо, чтобы результат хотя бы одного простого выражения, входящего в состав сложного, был истинным.

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

Да, операторы ветвления могут быть вложенными.

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.

Оператор цикла while выполняет указанный набор инструкций до тех пор, пока условие цикла истинно. Истинность условия определяется также как и в операторе if. Оператор for выполняет указанный набор инструкций заданное количество раз, которое определяется количеством элементов в наборе.

14. Назовите назначение и способы применения функции range.

Функция range возвращает неизменяемую последовательность чисел в виде объекта range. С помощью range можно генерировать возрастающие и убывающие последовательности чисел.

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

range(15, 0, -2). Принимает параметры start (с какого числа наичнается последовательность), stop (до какого числа продолжается последовательность чисел), step (с каким шагом растут числа).

16. Могут ли быть циклы вложенными?

Да, циклы в Python могут быть вложенными.

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

Бесконечный цикл образуется, когда условие завершения никогда не становится ложным. Можно прервать с помощью оператора 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 используется при работе с циклами. Он 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 стандартные потоки 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?

Для того, чтобы использовать поток stderr необходимо передать его в параметре file функции print.

22. Каково назначение функции exit?

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

Вывод: 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.