

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 «Основы программной инженерии»

Выполнил:
Магомедов Имран Борисович
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., кандидат технических
наук, доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

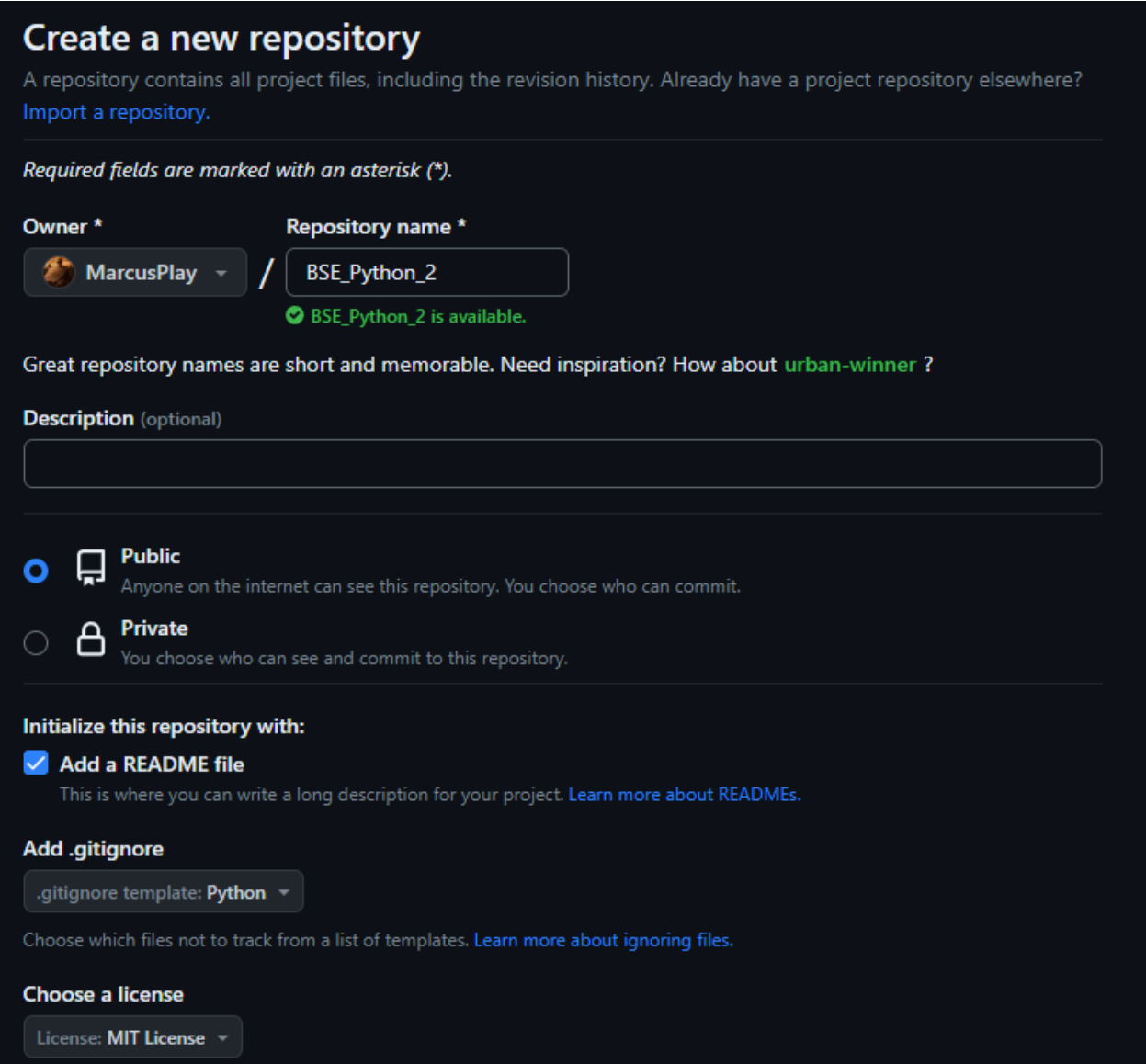
Ставрополь, 2023 г.

Тема: Условные операторы и циклы в языке Python

Цель работа: 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. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия 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 * MarcusPlay / **Repository name *** BSE_Python_2

✔ BSE_Python_2 is available.

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

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

3. Выполните клонирование созданного репозитория.

```
Imran@HOME-PC MINGW64 /g/Другие компьютеры/Компьютер/СКФУ/Основное  
торная 5  
$ git clone https://github.com/MarcusPlay/BSE_Python_2.git  
Cloning into 'BSE_Python_2'...  
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.
```

4. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

5. Самостоятельно изучите рекомендации к оформлению исходного кода на языке Python PEP-8 (<https://pep8.org/>). Выполните оформление исходного примеров лабораторной работы и индивидуальных созданий в соответствии с PEP-8.

6. Проработайте примеры лабораторной работы. Создайте для каждого примера отдельный модуль языка Python. Зафиксируйте изменения в репозитории.



7. Приведите в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных, вводимых с клавиатуры.


Пример 1.

Examples >  example_1.py > ...

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

```
Value of x? 5
y = -25.95892427466314
```


Пример 2.

Examples >  example_2.py > ...

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  if __name__ == '__main__':
8      n = int(input("Введите номер месяца: "))
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)
```

```
Введите номер месяца: 12
Зима
```


Пример 3.

Examples >  example_3.py > ...

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

```
Value of n? 5
Value of x? 6
S = 3.068814808882306
```

Пример 4.

Examples >  example_4.py > ...

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

```
Value of a? 3
x = 1.7320508075688772
X = 1.7320508075688772
```

Пример 5.

```

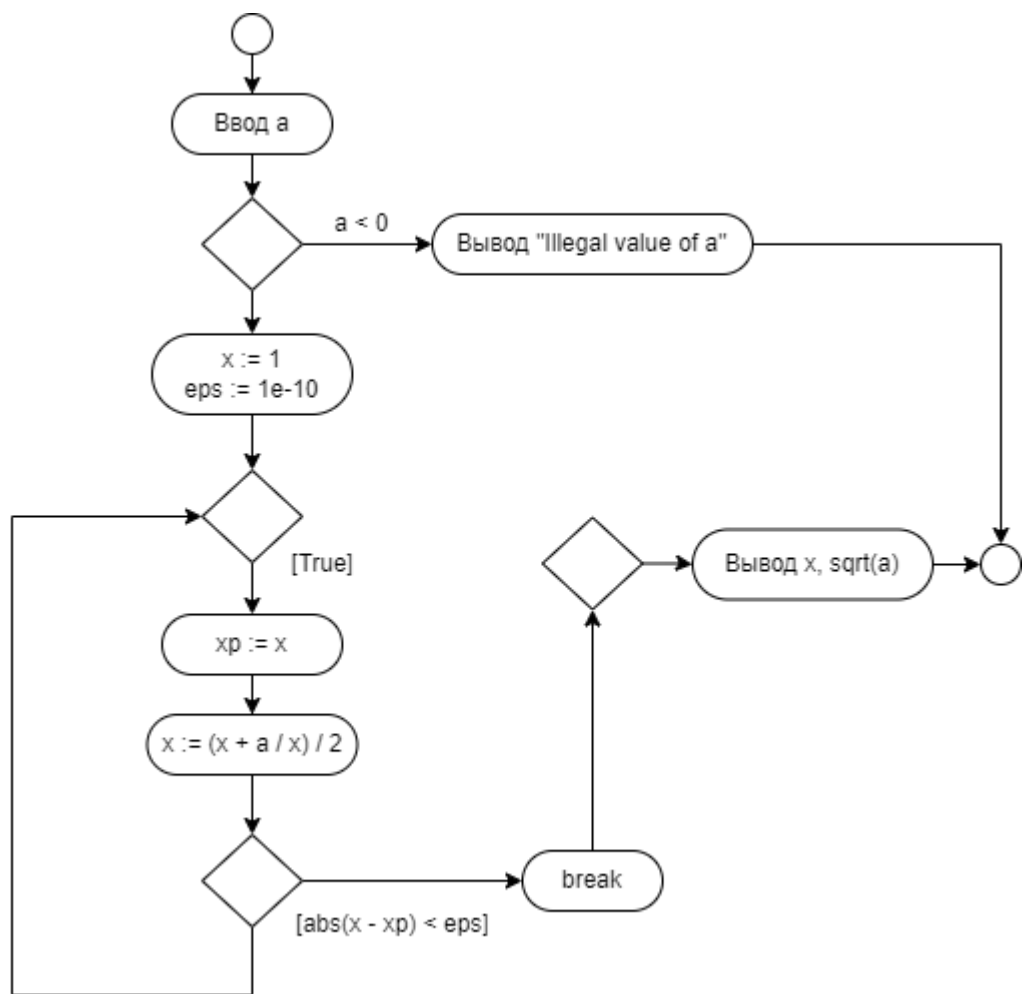
Examples > 🐍 example_5.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5  import sys
6
7  # Постоянная Эйлера.
8  EULER = 0.5772156649015328606
9  # Точность вычислений.
10 EPS = 1e-10
11
12
13 if __name__ == '__main__':
14     (variable) x: float
15     x = float(input(f x? "))
16     if x == 0:
17         print("Illegal value of x", file=sys.stderr)
18         exit(1)
19
20     a = x
21     S, k = a, 1
22
23     # Найти сумму членов ряда.
24     while math.fabs(a) > EPS:
25         a *= x * k / (k + 1) ** 2
26         S += a
27         k += 1
28
29     # Вывести значение функции.
30     print(f"Ei({x}) = {EULER + math.log(math.fabs(x)) + S}")

```

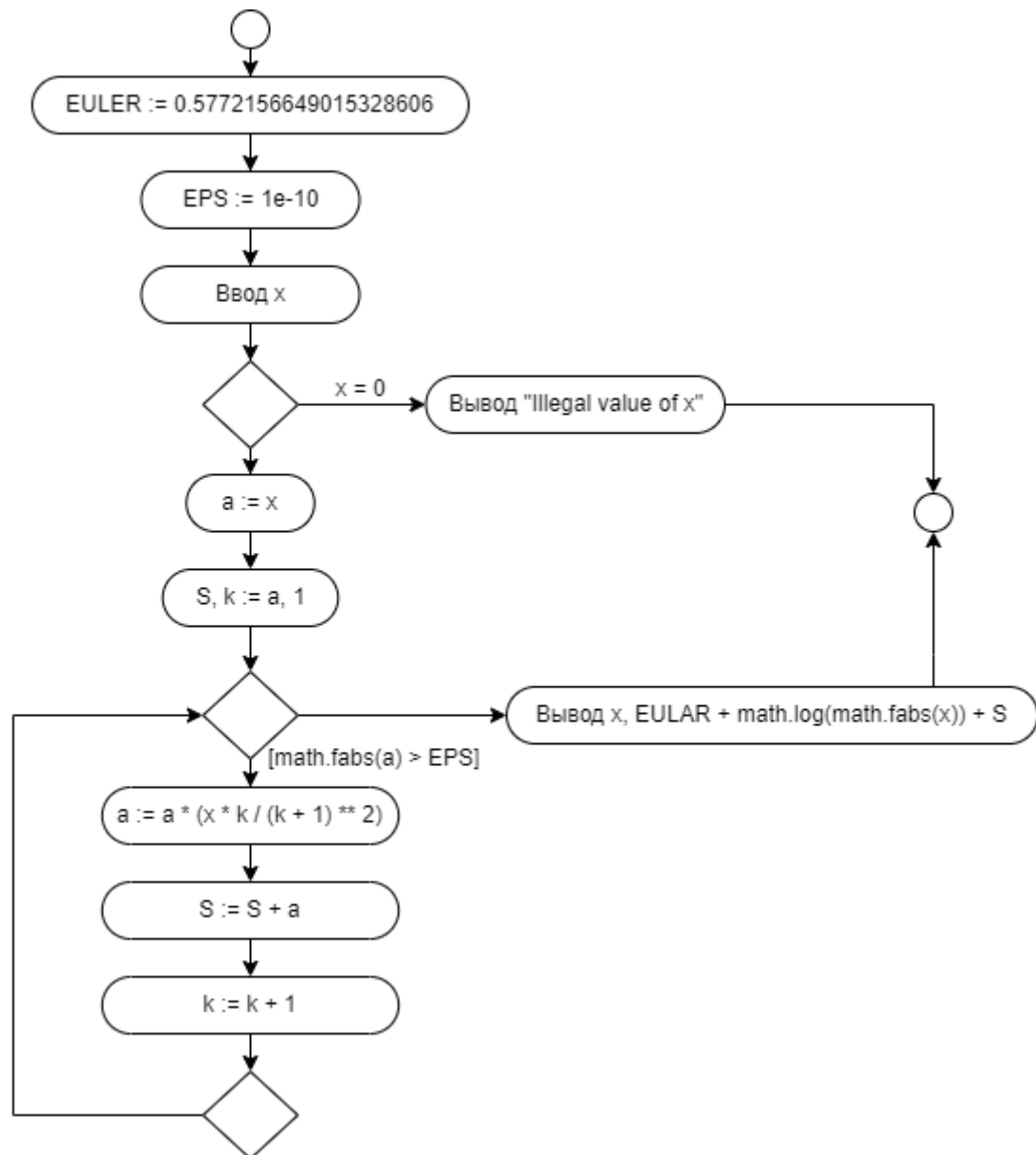
Value of x? 5
Ei(5.0) = 40.18527535579794

8. Для примеров 4 и 5 постройте UML-диаграмму деятельности. Для построения диаграмм деятельности использовать веб-сервис Google <https://www.diagrams.net/>.

Пример 4.



Пример 5.



9. Выполните индивидуальные задания, согласно своему варианту. Для заданий повышенной сложности номер варианта должен быть получен у преподавателя.

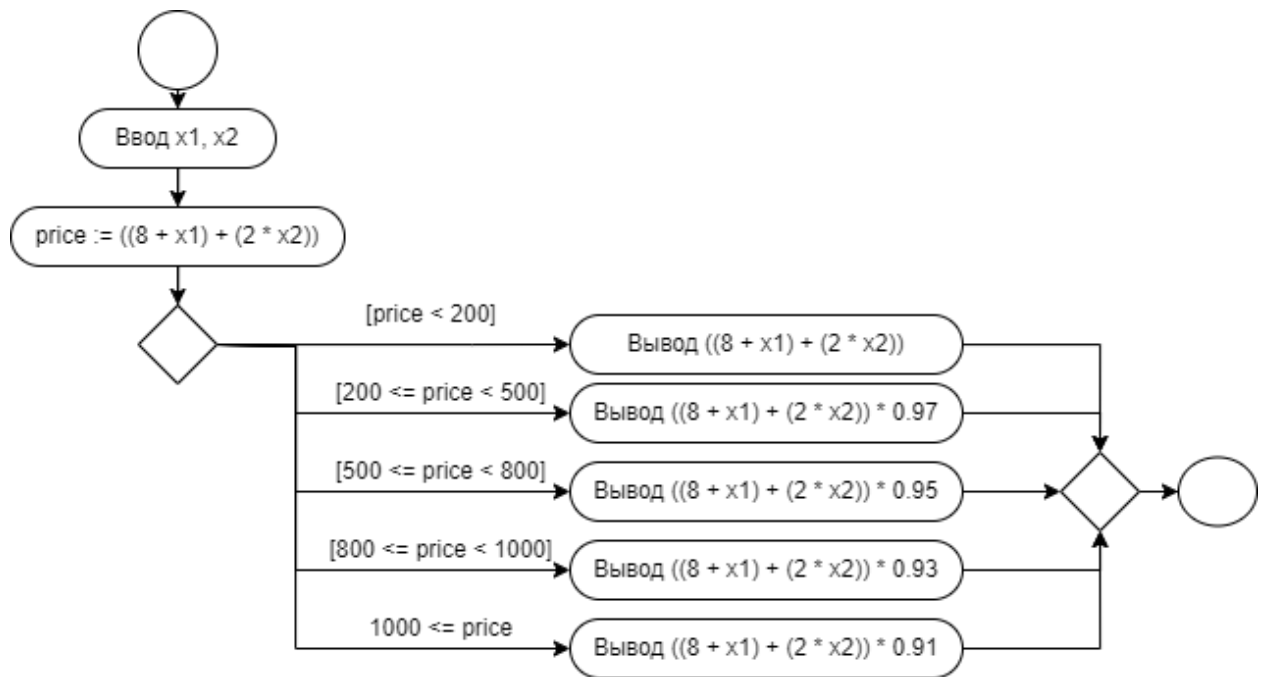
Задание 1 (вариант – 12)

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 и две банки краски по цене X. Сколько он заплатил?

Код решения:

```
individual_task_1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def total_price(price: float):
5      if (price < 200):
6          return price
7      elif (200 <= price < 500):
8          return price * 0.97
9      elif (500 <= price < 800):
10         return price * 0.95
11     elif (800 <= price < 1000):
12         return price * 0.93
13     elif (1000 <= price):
14         return price * 0.91
15
16
17 if __name__ == "__main__":
18     x1 = int(input())
19     x2 = int(input())
20
21     print(total_price((8 * x1) + (2 * x2)))
```

UML-диаграмма для решения:



Задание 2 (вариант – 12)

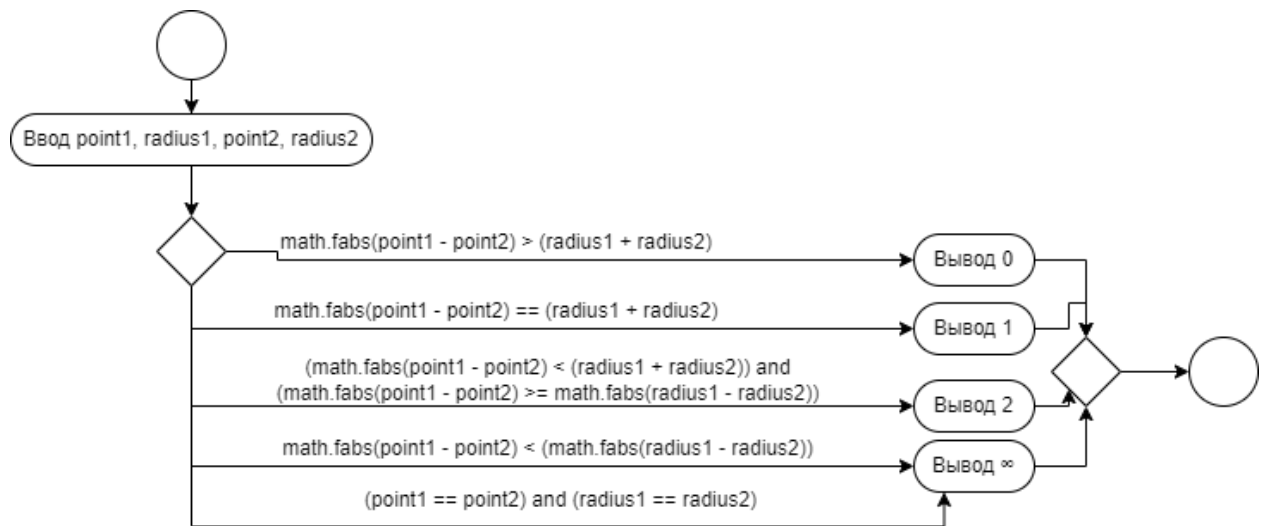
Две окружности заданы координатами центра и радиусами. Сколько точек пересечения имеют эти окружности?

Код решения:

```

individual_task_2.py U
individual_task_2.py > ...
1  # -*- coding: utf-8 -*-
2
3
4  import math
5
6
7  def find_point(point1, radius1, point2, radius2):
8      if (math.fabs(point1 - point2) > (radius1 + radius2)):
9          return 0
10     elif (math.fabs(point1 - point2) == (radius1 + radius2)):
11         return 1
12     elif ((math.fabs(point1 - point2) < (radius1 + radius2)) and (math.fabs(point1 - point2) >= math.fabs(radius1 - radius2))):
13         return 2
14     elif (math.fabs(point1 - point2) < (math.fabs(radius1 - radius2))):
15         return "∞"
16     elif ((point1 == point2) and (radius1 == radius2)):
17         return "∞"
18
19
20  if __name__ == "__main__":
21      point1 = float(input())
22      radius1 = float(input())
23      point2 = float(input())
24      radius2 = float(input())
25
26      points = find_point(point1, radius1, point2, radius2)
27      print(f"Эти две окружности имеют {points} точек пересечения")
28      input()
  
```

UML-диаграмма для решения:



Задание 3 (вариант - 12)

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.

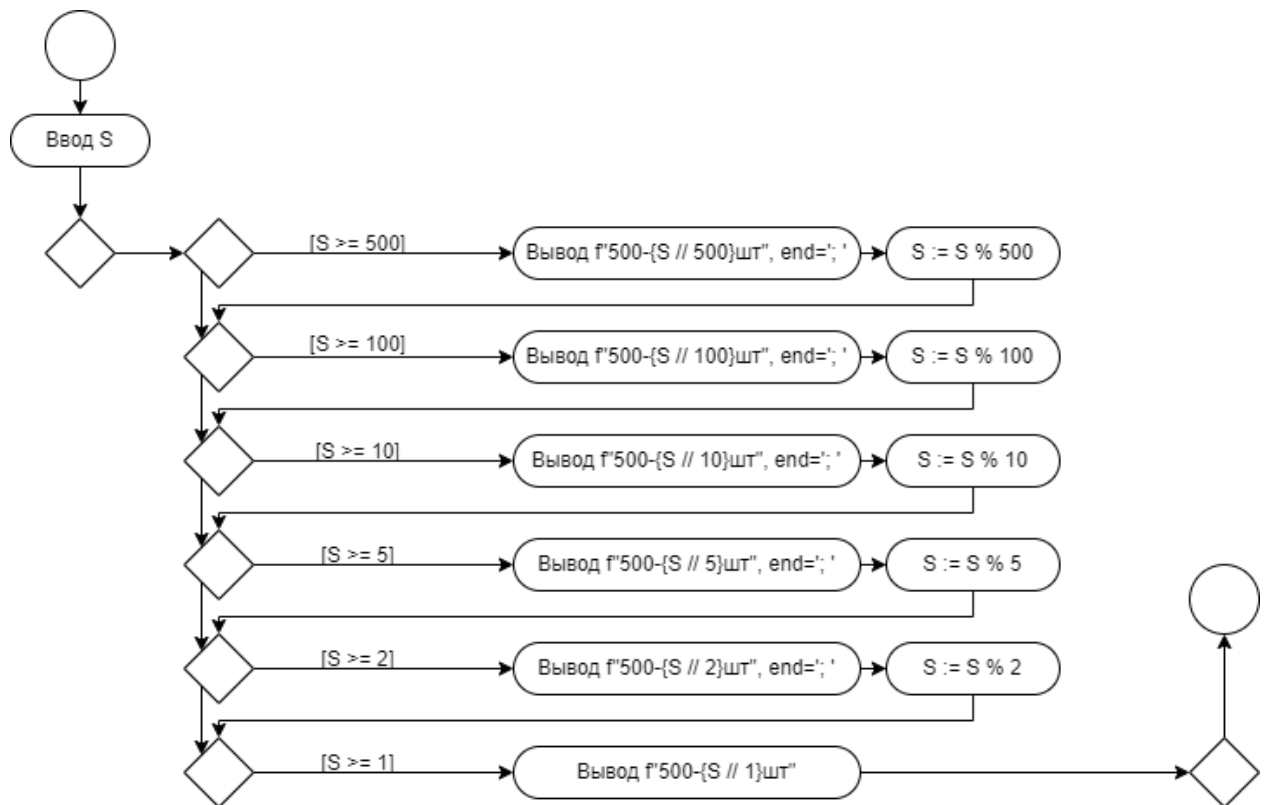
Код решения:

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def how_much(S):
5      ....# 1, 2, 5, 10, 100, 500
6      ....if S >= 500:
7          ....print(f"500-{S // 500}шт", end='; ')
8          ....S = S % 500
9      ....if S >= 100:
10         ....print(f"100-{S // 100}шт", end='; ')
11         ....S = S % 100
12     ....if S >= 10:
13         ....print(f"10-{S // 10}шт", end='; ')
14         ....S = S % 10
15     ....if S >= 5:
16         ....print(f"5-{S // 5}шт", end='; ')
17         ....S = S % 5
18     ....if S >= 2:
19         ....print(f"2-{S // 2}шт", end='; ')
20         ....S = S % 2
21     ....if S >= 1:
22         ....print(f"1-{S // 1}шт")
23         ....S = S % 1
24
25
26 if __name__ == "__main__":
27     ....S = int(input())
28     ....how_much(S)

```

UML-диаграмма для решения:



10. Задача повышенной сложности.

Составить UML-диаграмму деятельности, программу и произвести вычисления вычисление значения специальной функции по ее разложению в ряд с точностью $\varepsilon = 10^{-10}$, аргумент функции вводится с клавиатуры. Номер варианта необходимо получить у преподавателя. Интегральный гиперболический синус: $Shi(x) = \int_0^x \frac{\text{sh } t}{t} dt = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{(2n+1)(2n+1)!}$

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5
6
7  def shi(x, tolerance=1e-10):
8      ....result = 0.0
9      ....term = x
10     ....n = 0
11
12     ....while abs(term) > tolerance:
13         ....result += term
14         ....n += 1
15         ....term *= (x * x) / ((2 * n + 1) * (2 * n))
16
17     ....return result
18
19
20 if __name__ == "__main__":
21     ....x = int(input())
22     ....print(math.sinh(x))
23     ....print(shi(x))

```

11. Зафиксируйте сделанные изменения в репозитории.
12. Выполните слияние ветки для разработки с веткой main / master.
13. Отправьте сделанные изменения на сервер GitHub.

Контрольные вопросы

1. Для чего нужны диаграммы деятельности UML?

Диаграммы деятельности UML используются для визуализации и моделирования процессов и действий в системе, позволяя легче понимать, анализировать и проектировать бизнес-процессы.

2. Что такое состояние действия и состояние деятельности?

Состояние действия (action state) в диаграммах деятельности UML представляет мгновенное выполнение действия, а состояние деятельности (activity state) представляет набор действий, которые выполняются в течение некоторого времени.

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

Для обозначения переходов используют стрелки, а для ветвлений - ромбы.

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Алгоритм разветвляющейся структуры — это алгоритм с условным оператором (if-else), который позволяет выполнять различные действия в зависимости от условия.

5. Чем отличается разветвляющийся алгоритм от линейного?

Разветвляющийся алгоритм имеет условный оператор и может выполнять различные действия в зависимости от условия, в то время как линейный алгоритм выполняет действия последовательно, без разветвлений.

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

Условный оператор — это конструкция, позволяющая выполнять определенные действия в зависимости от выполнения условия. В Python, основные формы условного оператора: if, if-else, и if-elif-else.

7. Какие операторы сравнения используются в Python?

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

8. Что называется простым условием? Приведите примеры.

Простое условие — это условие, которое проверяет одну конкретную величину. Пример: `if x > 5:`.

9. Что такое составное условие? Приведите примеры.

Составное условие — это условие, состоящее из нескольких простых условий, объединенных логическими операторами. Пример: `if x > 5 and y < 10:`.

10. Какие логические операторы допускаются при составлении сложных условий?

Для составления сложных условий в Python используются логические операторы: `and` (логическое И), `or` (логическое ИЛИ), и `not` (логическое НЕ).

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

Да, оператор ветвления (например, `if`) может содержать внутри себя другие операторы ветвления, создавая вложенные структуры.

12. Какой алгоритм является алгоритмом циклической структуры?

Алгоритм циклической структуры — это алгоритм, в котором определенные действия выполняются многократно в цикле. Примеры: циклы `for` и `while` в Python.

13. Типы циклов в языке Python.

В Python существуют два основных типа циклов: цикл `for` (перебор элементов в итерируемом объекте) и цикл `while` (повторение действий до выполнения условия).

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

Функция `range` используется для создания последовательности чисел в определенном диапазоне. Она может использоваться в циклах для управления повторением действий.

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

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

16. Могут ли быть циклы вложенными?

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

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

Бесконечный цикл образуется, когда условие цикла всегда истинно. Для выхода из него используется оператор `break`, который позволяет прервать выполнение цикла.

18. Для чего нужен оператор `break`?

Оператор `break` используется для прерывания выполнения цикла и выхода из него, даже если условие цикла остается истинным.

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

Оператор `continue` используется внутри циклов для пропуска текущей итерации и перехода к следующей итерации без выполнения оставшихся действий в текущей итерации.

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

Стандартные потоки `stdout` (стандартный вывод) и `stderr` (стандартный вывод ошибок) используются для направления вывода информации и ошибок программы. `stdout` используется для нормального вывода, а `stderr` для вывода ошибок.

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

Для вывода в стандартный поток ошибок (stderr) можно воспользоваться методом `sys.stderr.write("Текст ошибки")`` после импорта модуля `sys`.

22. Каково назначение функции exit?

Функция `exit` используется для завершения выполнения программы. Она позволяет передать код завершения и завершить выполнение программы с указанным кодом.