

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 ПРАКТИЧЕСКОЙ РАБОТЕ №3
дисциплины «**Программирование на Python**»
Вариант №11

Выполнила:

Ковжого Елизавета Андреевна

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)

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

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2025 г.

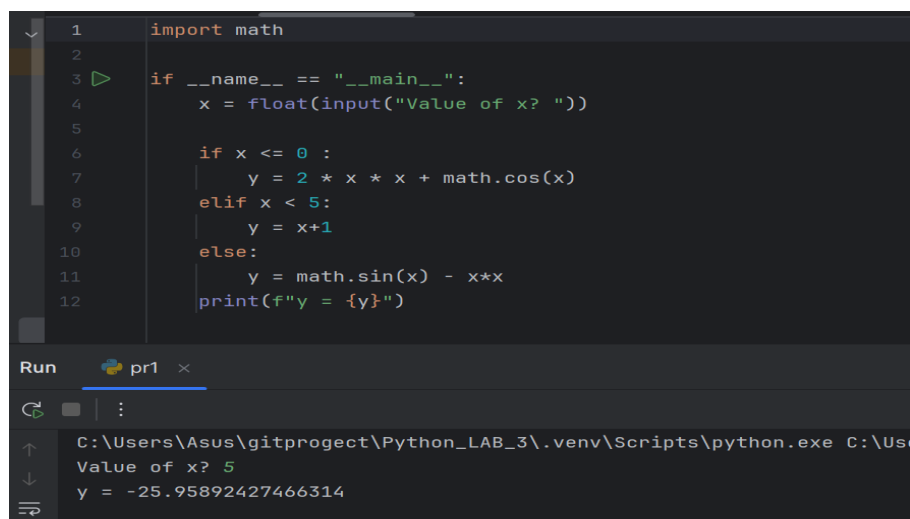
Тема: условные операторы и циклы в языке Python.

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

Порядок выполнения работы:

Адрес репозитория: https://github.com/LissKovzogo/Python_LAB_33.git

1. Создали, настроили и клонировали репозиторий Python_LAB_3.
2. Создали проект PyCharm в папке репозитория.
3. Проработали все примеры лабораторной работы и создали для каждого отдельный модуль Python.



```
1 import math
2
3 if __name__ == "__main__":
4     x = float(input("Value of x? "))
5
6     if x <= 0 :
7         y = 2 * x * x + math.cos(x)
8     elif x < 5:
9         y = x+1
10    else:
11        y = math.sin(x) - x*x
12    print(f"y = {y}")
```

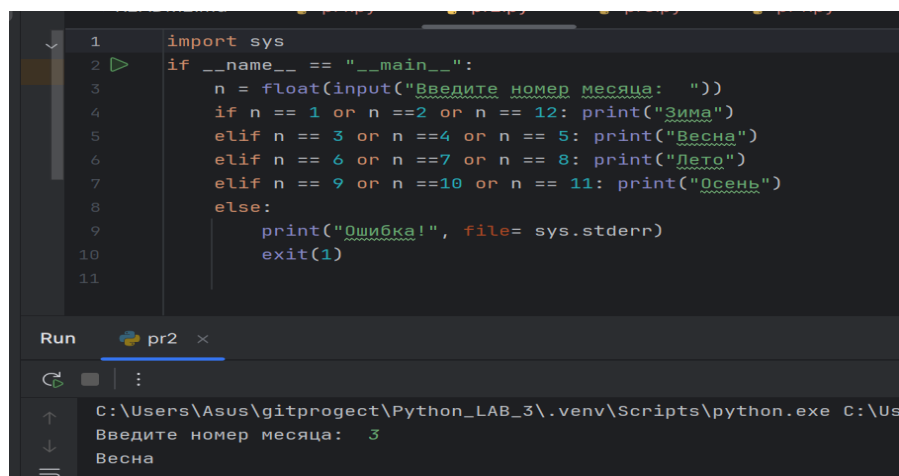
Run pr1

C:\Users\Asus\gitproject\Python_LAB_3\.venv\Scripts\python.exe C:\Users\Asus\gitproject\Python_LAB_3\pr1.py

Value of x? 5

y = -25.95892427466314

Рис. 1 — Работа с примером № 1



```
1 import sys
2 if __name__ == "__main__":
3     n = float(input("Введите номер месяца: "))
4     if n == 1 or n == 2 or n == 12: print("Зима")
5     elif n == 3 or n == 4 or n == 5: print("Весна")
6     elif n == 6 or n == 7 or n == 8: print("Лето")
7     elif n == 9 or n == 10 or n == 11: print("Осень")
8     else:
9         print("Ошибка!", file= sys.stderr)
10        exit(1)
11
```

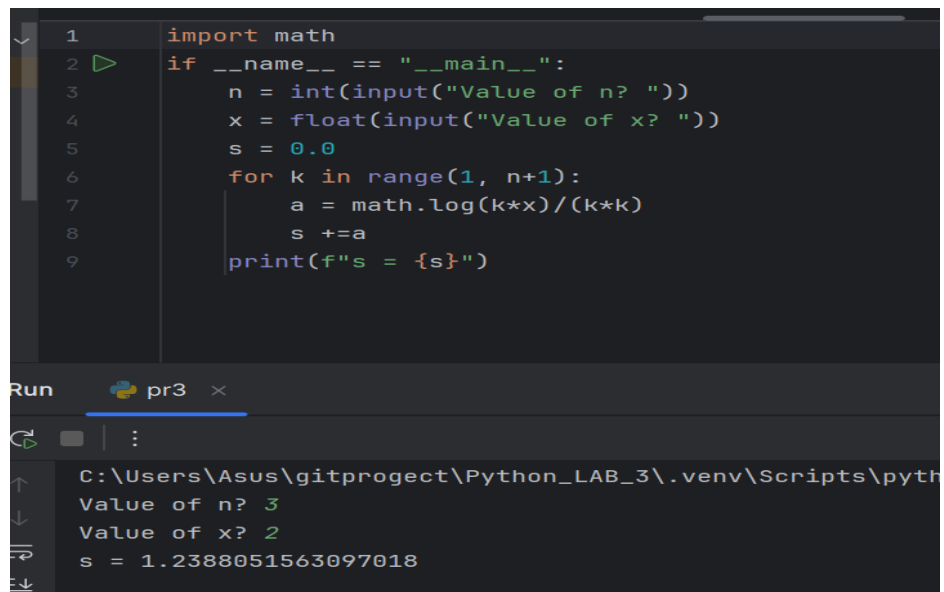
Run pr2

C:\Users\Asus\gitproject\Python_LAB_3\.venv\Scripts\python.exe C:\Users\Asus\gitproject\Python_LAB_3\pr2.py

Введите номер месяца: 3

Весна

Рис. 2 — Работа с примером № 2



```
1 import math
2 if __name__ == "__main__":
3     n = int(input("Value of n? "))
4     x = float(input("Value of x? "))
5     s = 0.0
6     for k in range(1, n+1):
7         a = math.log(k*x)/(k*k)
8         s += a
9     print(f"s = {s}")
```

Run pr3

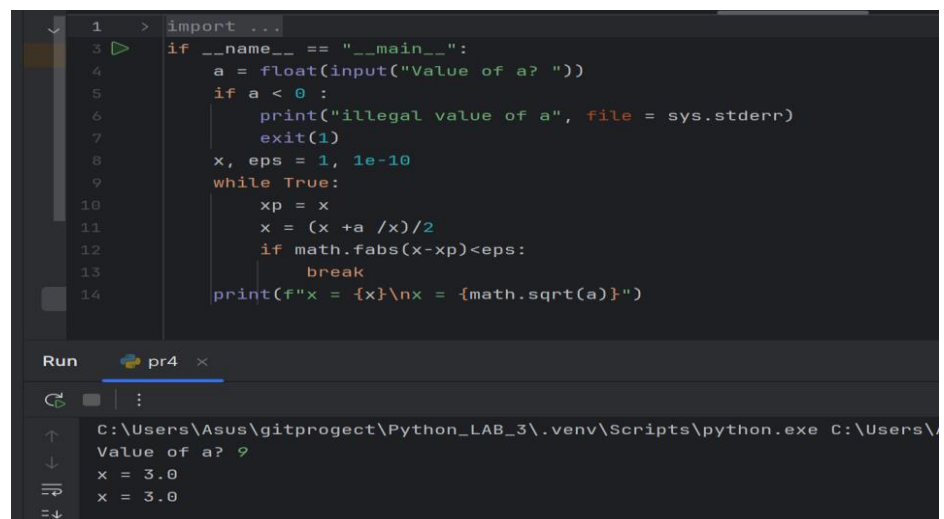
C:\Users\Asus\gitproject\Python_LAB_3\.venv\Scripts\python.exe C:\Users\Asus\gitproject\Python_LAB_3\pr3.py

Value of n? 3

Value of x? 2

s = 1.2388051563097018

Рис. 3 — Работа с примером № 3



```
1 > import ...
3 if __name__ == "__main__":
4     a = float(input("Value of a? "))
5     if a < 0:
6         print("illegal value of a", file = sys.stderr)
7         exit(1)
8     x, eps = 1, 1e-10
9     while True:
10         xp = x
11         x = (x + a / x) / 2
12         if math.fabs(x - xp) < eps:
13             break
14     print(f"x = {x}\nx = {math.sqrt(a)}")
```

Run pr4

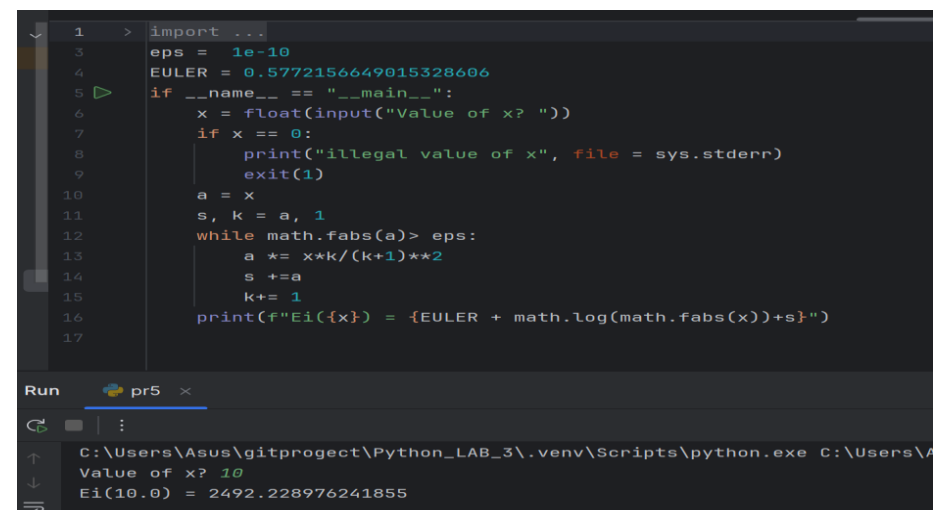
C:\Users\Asus\gitproject\Python_LAB_3\.venv\Scripts\python.exe C:\Users\Asus\gitproject\Python_LAB_3\pr4.py

Value of a? 9

x = 3.0

x = 3.0

Рис. 4 — Работа с примером № 4



```
1 > import ...
3 eps = 1e-10
4 EULER = 0.5772156649015328606
5 if __name__ == "__main__":
6     x = float(input("Value of x? "))
7     if x == 0:
8         print("illegal value of x", file = sys.stderr)
9         exit(1)
10     a = x
11     s, k = a, 1
12     while math.fabs(a) > eps:
13         a *= x*k/(k+1)**2
14         s += a
15         k += 1
16     print(f"Ei({x}) = {EULER + math.log(math.fabs(x))+s}")
17
```

Run pr5

C:\Users\Asus\gitproject\Python_LAB_3\.venv\Scripts\python.exe C:\Users\Asus\gitproject\Python_LAB_3\pr5.py

Value of x? 10

Ei(10.0) = 2492.228976241855

Рис. 5 — Работа с примером № 5

4. Построили UML-диаграммы деятельности для примеров №4 и №5.

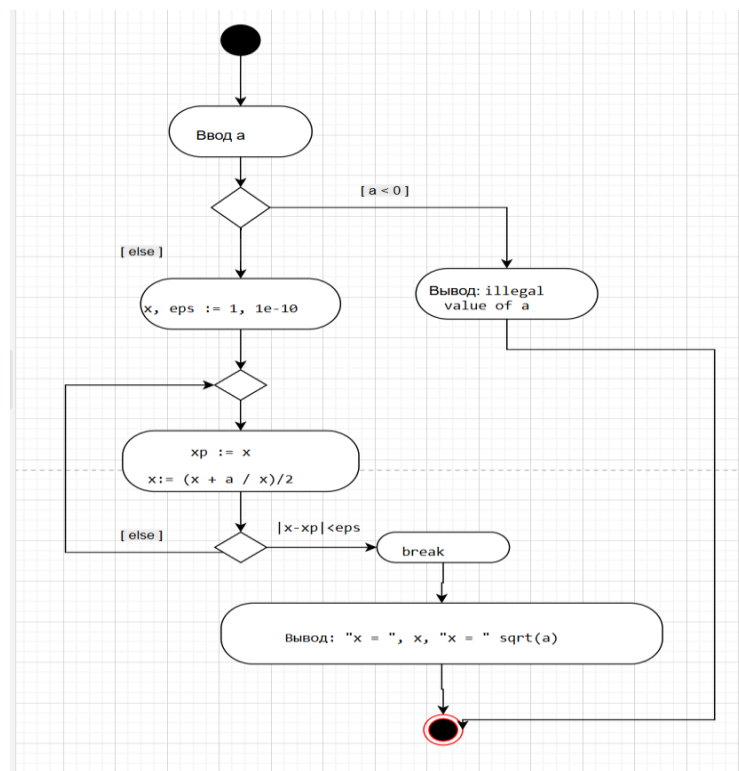


Рис. 6 — UML-диаграммы деятельности для примеров №4

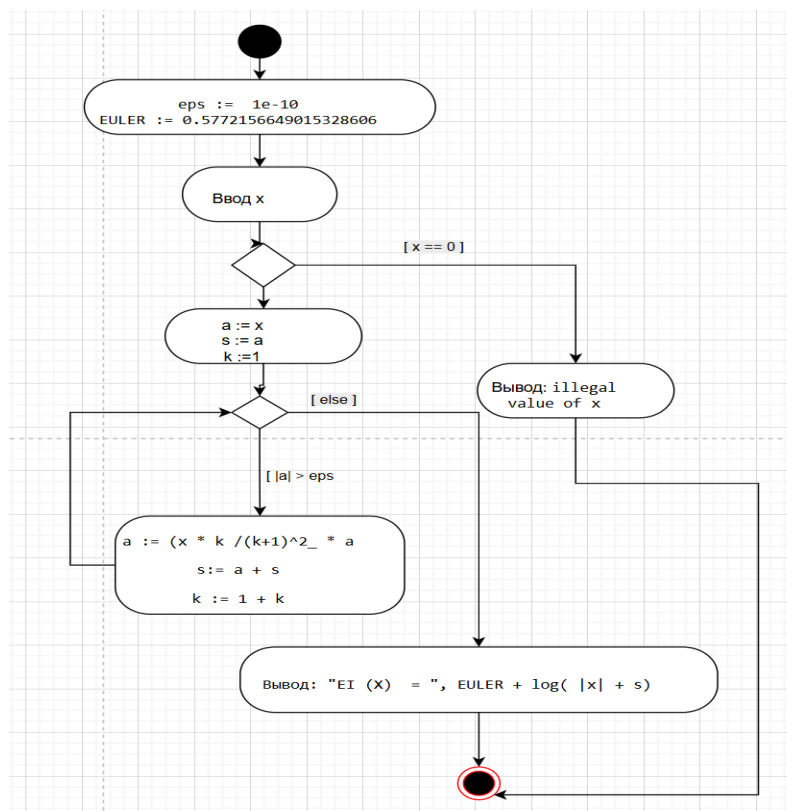


Рис. 7 — UML-диаграммы деятельности для примеров №5

5. Выполнили индивидуальные задания согласно варианту и построили UML-диаграммы деятельности.

```
1  x = int(input("Value of x? "))
2  zp = 0
3  if x <= 50:
4      zp = 30 * x
5  elif 50 < x <= 75:
6      zp = 50 * x
7  elif 75 < x <= 90:
8      zp = 65 * x
9  else: zp = 70 * x + 2000
10
11 print(f"Зарплата в копейках = {zp}, "
12       f"Зарплата в рублях = {zp/100}")
```

in pr1 x in_1 x

C:\Users\Asus\gitproject\Python_LAB_3\.venv\Scripts\python.exe

Value of x? 80

Зарплата в копейках = 5200, Зарплата в рублях = 52.0

Рис. 8 — Индивидуальное задание №1

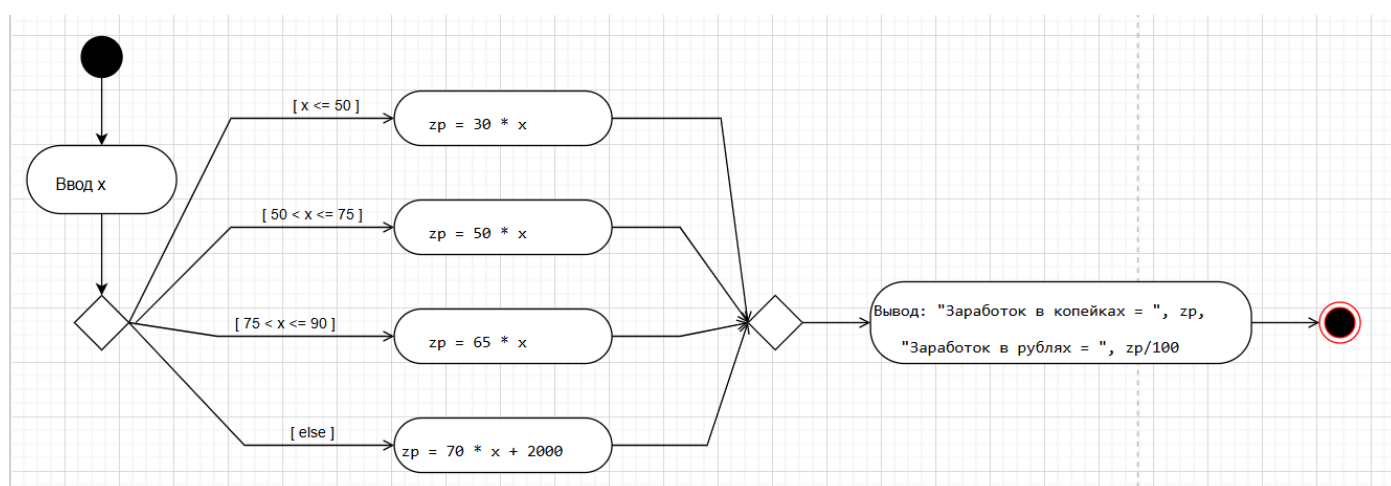


Рис. 9 — UML-диаграммы деятельности для индивидуальное задания №1

```
1 x = int(input("Value of x? "))
2 y = int(input("Value of y? "))
3
4 if x % y == 0:
5     print("x делится на y без остатка")
6 else:
7     print("x не делится на y без остатка")
```

Run pr1 in_2

C:\Users\Asus\gitprogect\Python_LAB_3\.venv\Scr
Value of x? 10
Value of y? 3
x не делится на y без остатка

Рис. 10 — Индивидуальное задание №2

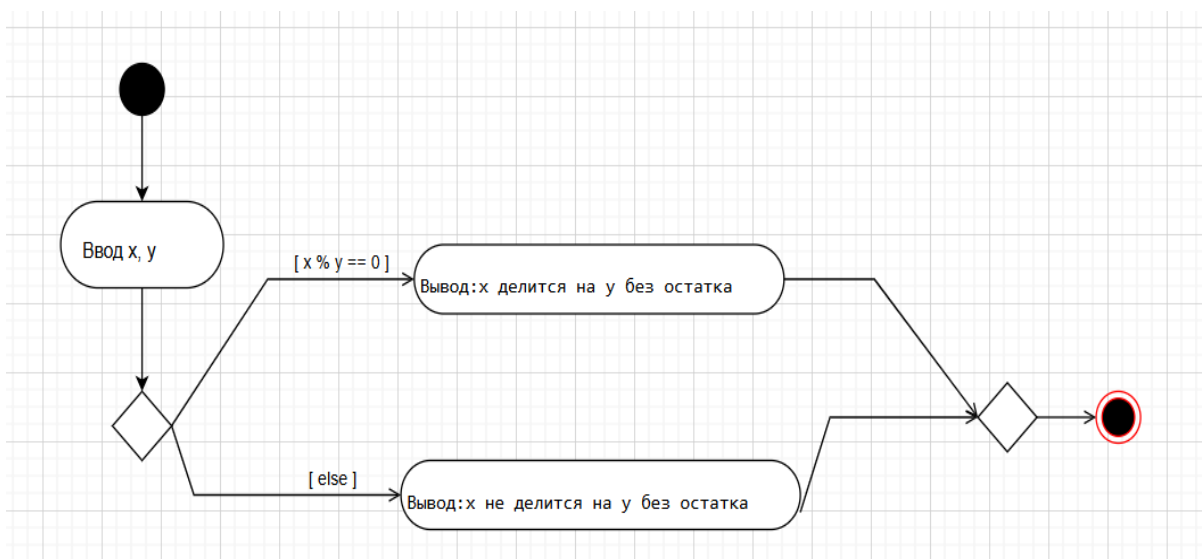


Рис. 11 — UML-диаграммы деятельности для индивидуальное задания №2

```

1  import math
2
3  n = int(input("Value of n? "))
4  divisors = []
5  for i in range(1, int(math.sqrt(n) + 1)):
6      if n % i == 0:
7          divisors.append(n//i)
8          divisors.append(i)
9
10 print(f"Делители числа {n}: {divisors}")

```

pr1 x in_3 x

C:\Users\Asus\gitproject\Python_LAB_3\.venv\Scripts
Value of n? 50
Делители числа 50: [50, 1, 25, 2, 10, 5]

Рис. 12 — Индивидуальное задание №3

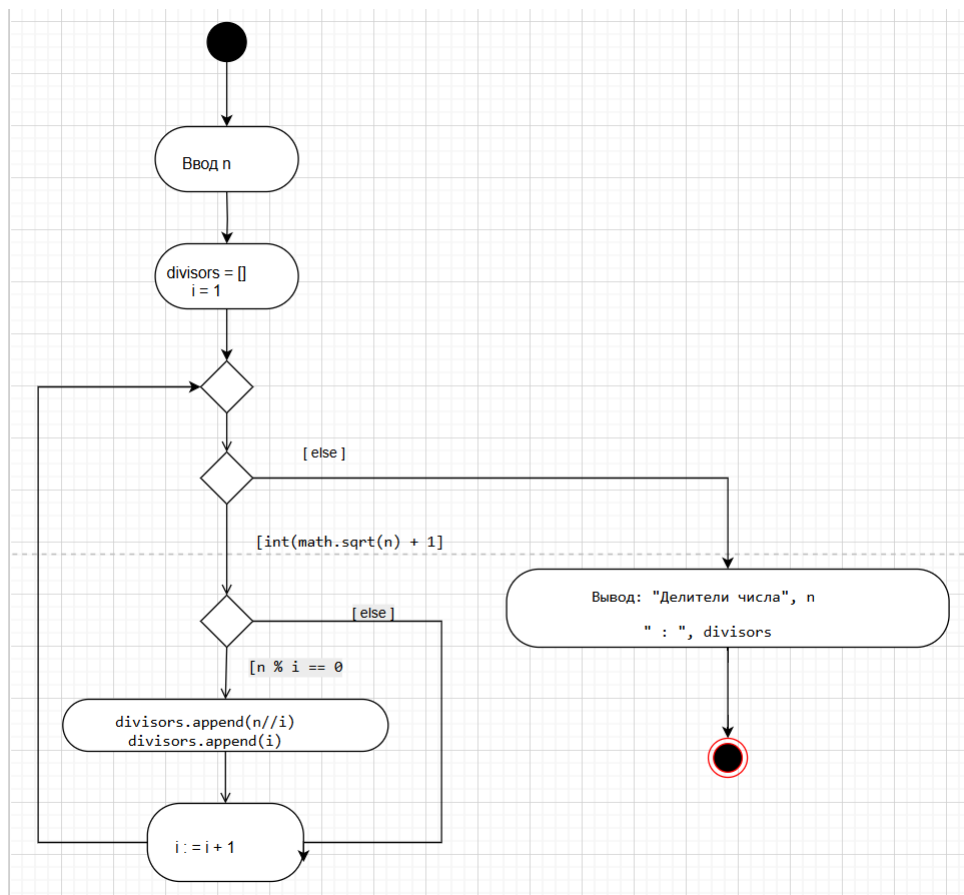


Рис. 13 — UML-диаграммы деятельности для индивидуальное задание №3

6. Выполнили индивидуальное задание повышенной сложности и составили UML-диаграмму деятельности.

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

if __name__ == "__main__":

    x = float(input("Введите x: "))
    epsilon = 1e-10
    gamma = 0.57721566490153286060

    result = gamma + math.log(x)
    n = 1
    term = 1

    while abs(term) > epsilon:
        term = ((-1) ** n) * (x ** (2 * n)) / (2 * n * math.factorial(2 * n))
        result += term
        n += 1

    print(f"Ci({x}) = {result}")
```

Рис. 14 — Индивидуальное задание №3

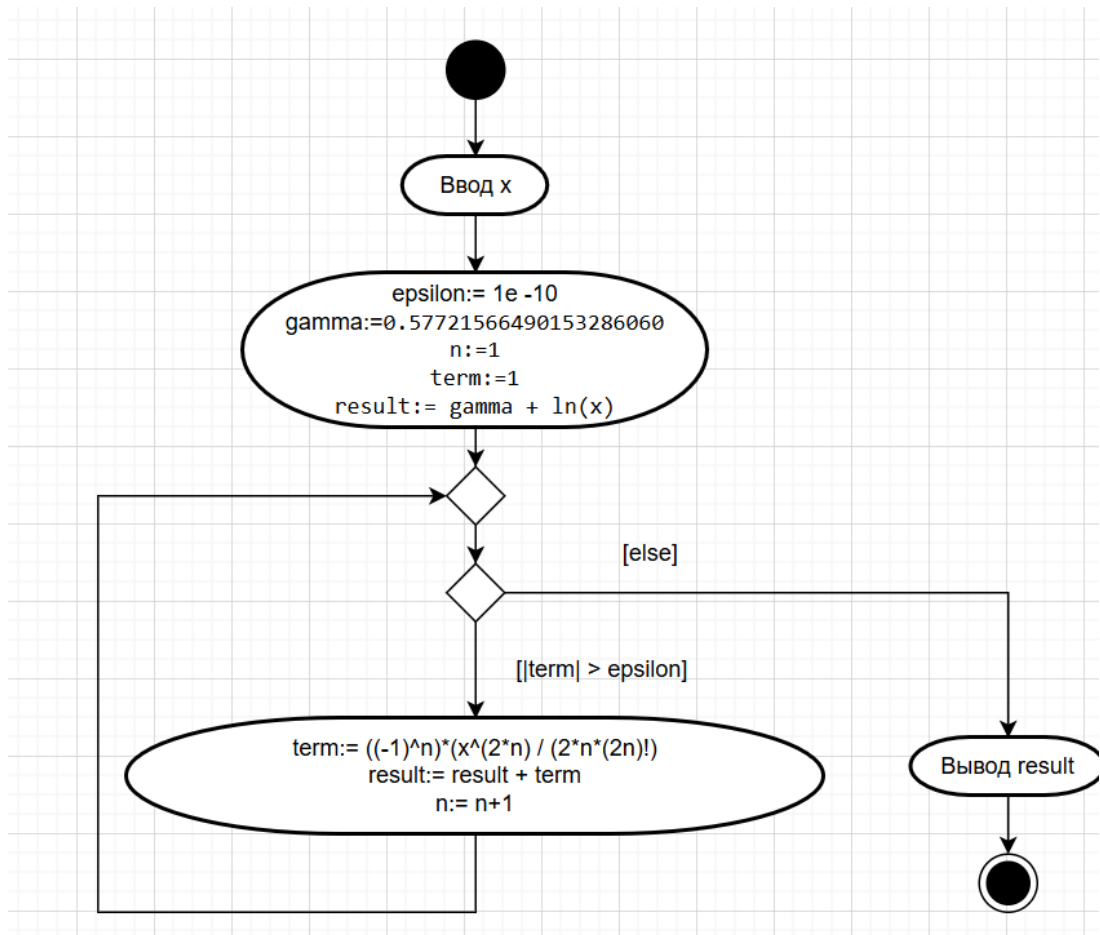


Рис. 15 — UML-диаграммы деятельности для индивидуальное задания №3

7. Зафиксировали все изменения в репозиторий.

```

Asus@DESKTOP-KEJGOCU MINGW64 ~/gitprogect/Python_LAB_3 (main)
$ git commit -m"Без документов"
[main 318da95] Без документов
17 files changed, 129 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/Python_LAB_3.iml
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 PyCharm/in_1.py
create mode 100644 PyCharm/in_2.py
create mode 100644 PyCharm/in_3.py
create mode 100644 PyCharm/in_4.py
create mode 100644 PyCharm/pr1.py
create mode 100644 PyCharm/pr2.py
create mode 100644 PyCharm/pr3.py
create mode 100644 PyCharm/pr4.py
create mode 100644 PyCharm/pr5.py
create mode 100644 "Python_\320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\2
create mode 100644 "~$thon_\320\277\321\200\320\276\320\263\321\200\320\260\320\274\320\2
Asus@DESKTOP-KEJGOCU MINGW64 ~/gitprogect/Python_LAB_3 (main)
$
  
```

Рис. 16 — Фиксирование изменений

Контрольные вопросы:

1. Для чего нужны диаграммы деятельности UML?

Для визуализации, специфицирования, конструирования и документации артефактов программных систем.

2. Что такое состояние действия и состояние деятельности?

Состояние действия - атомарная операция, состояние системы, представляющее собой выполнение некоторого действия.

Состояние деятельности - составная операция, содержащая поддеятельности, и которую можно прервать.

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

Для переходов используются стрелки, показывающие путь из одного состояния в другое.

Ромб является точкой ветвления, из которой алгоритм разветвляется.

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Это алгоритм, в котором вычислительный процесс осуществляется либо по одной, либо по другой ветви, в зависимости от условия.

5. Чем отличается разветвляющийся алгоритм от линейного? Наличием условий и альтернативных ветвей выполнения.

6. Что такое условный оператор? Какие существуют его формы? Оператор ветвления: if, if-else, if-elif-else.

7. Какие операторы сравнения используются в Python? ==, !=, <, >, <=, >=.

8. Что называется простым условием? Приведите примеры. Условие с одной логической операцией: `x > 5`; `name == "John"`.

9. Что такое составное условие? Приведите примеры. Условие с логическими операторами: `x > 5 and y < 10`, `age >= 18 or parent_consent`.

10. Какие логические операторы допускаются при составлении сложных условий?

Логические операторы: and, or, not.

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

Да, это называется вложенные условия.

12. Какой алгоритм является алгоритмом циклической структуры?

Алгоритм с многократно повторяющимися участками программы.

13. Типы циклов в языке Python.

for, while, do while.

14. Назовите назначение и способы применения функции range.

Генерация последовательностей чисел: 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): ...
```

16. Могут ли быть циклы вложенными?

Да, цикл внутри другого цикла.

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

```
while True: ....
```

```
break ...
```

18. Для чего нужен оператор break?

Для досрочного выхода из цикла.

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

В циклах, для перехода к следующей итерации.

20. Для чего нужны стандартные потоки stdout и stderr?

stdout - стандартный вывод данных и информационных сообщений,
stderr – вывод сообщений об ошибках.

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

```
import sys;
```

....

```
print("Error!", file=sys.stderr)
```

22. Каково назначение функции `exit`?

Завершение программы с кодом возврата: `exit(1)`

Вывод: приобрели навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Освоили операторы языка Python версии 3.x `if`, `while`, `for`, `break` и `continue`, позволяющих реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.