МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО
ОБРАЗОВАНИЯРОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №5 по дисциплине «Основы программной инженерии»

Выполнил студент
2 курса, группы ПИЖ-б-о-20-1
Тотубалина С.С.
Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2021 г

Ход работы

```python
#!/usr/bin/env python
# _*_ 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}")
```

Рис. 1 – код программы lab.5_ex.1.py

```
Value of x? -2
y = 7.583853163452858

Process finished with exit code 0
```

Рис. 2 – пример работы программы lab.5_ex.1.py при x< =0

```
Value of x? 3
y = 4.0

Process finished with exit code 0
```

Рис. 3 – пример работы программы lab.5_ex.1.py при x>0 и x<5

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

Process finished with exit code 0
```

Рис. 4 – пример работы программы lab.5_ex.1.py при x>=5

```python
#!/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)
```

Рис. 5 – код программы lab.5_ex.2.py

```
Введите номер месяца: 1
Зима

Process finished with exit code 0
```

Рис. 6 – пример работы программы lab.5_ex.2.py при n=1

```
Введите номер месяца: 3
Весна

Process finished with exit code 0
```

Рис. 7 – пример работы программы lab.5_ex.2.py при n=3

```
Введите номер месяца: 6
Лето

Process finished with exit code 0
```

Рис. 8 – пример работы программы lab.5_ex.2.py при n=6

```
Введите номер месяца: 9
Осень

Process finished with exit code 0
```

Рис. 9 – пример работы программы lab.5_ex.2.py при n=9

```
Введите номер месяца: 13
Ошибка!

Process finished with exit code 1
```

Рис. 10 – пример работы программы lab.5_ex.2.py при n=13

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

Рис. 11 – код программы lab.5_ex.3.py

```
Value of n? 4
Value of x? 5
S = 2.6732119195688706

Process finished with exit code 0
```

Рис. 12 – пример работы программы lab.5_ex.3.py при n=4, x=5

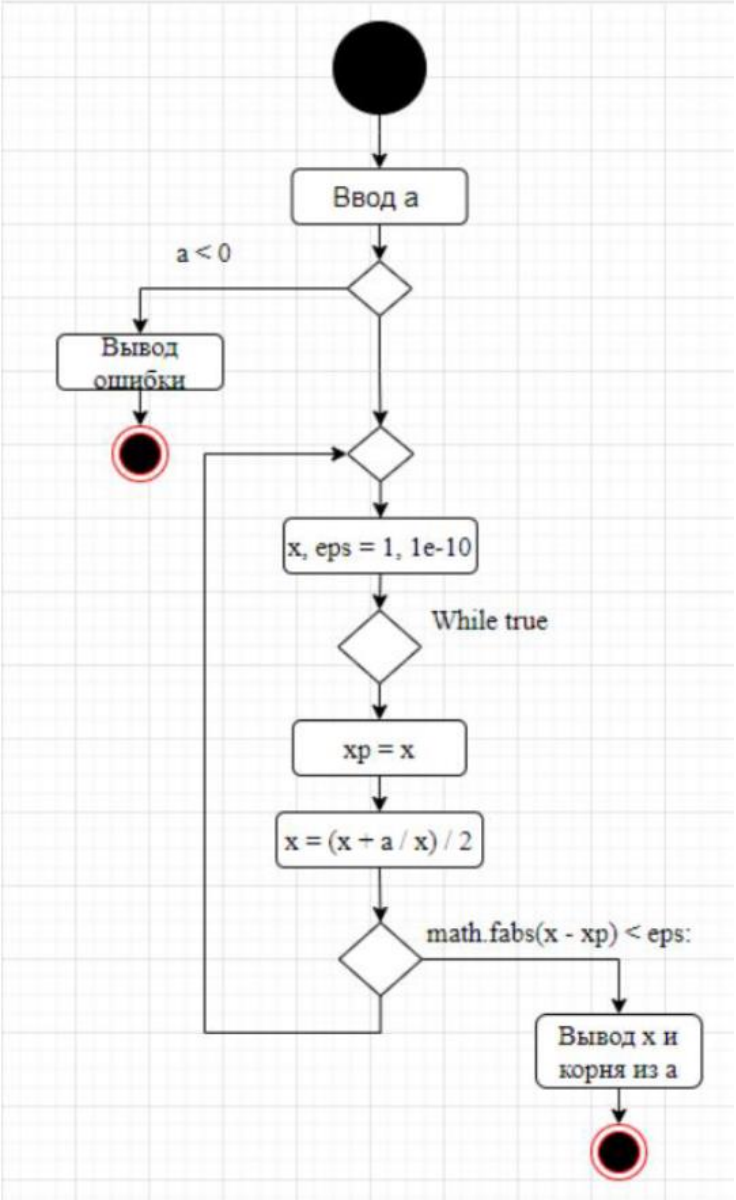

Рис. 13 – UML-диаграмма алгоритма

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

Рис. 14 – код программы

```
Value of a? -2
Illegal value of a

Process finished with exit code 1
```

Рис. 15 – пример работы программы lab.5_ex.4.py при a<0

```
Value of a? 4
x = 2.0
X = 2.0
```

Рис. 16 – пример работы программы lab.5_ex.4.py при a=4

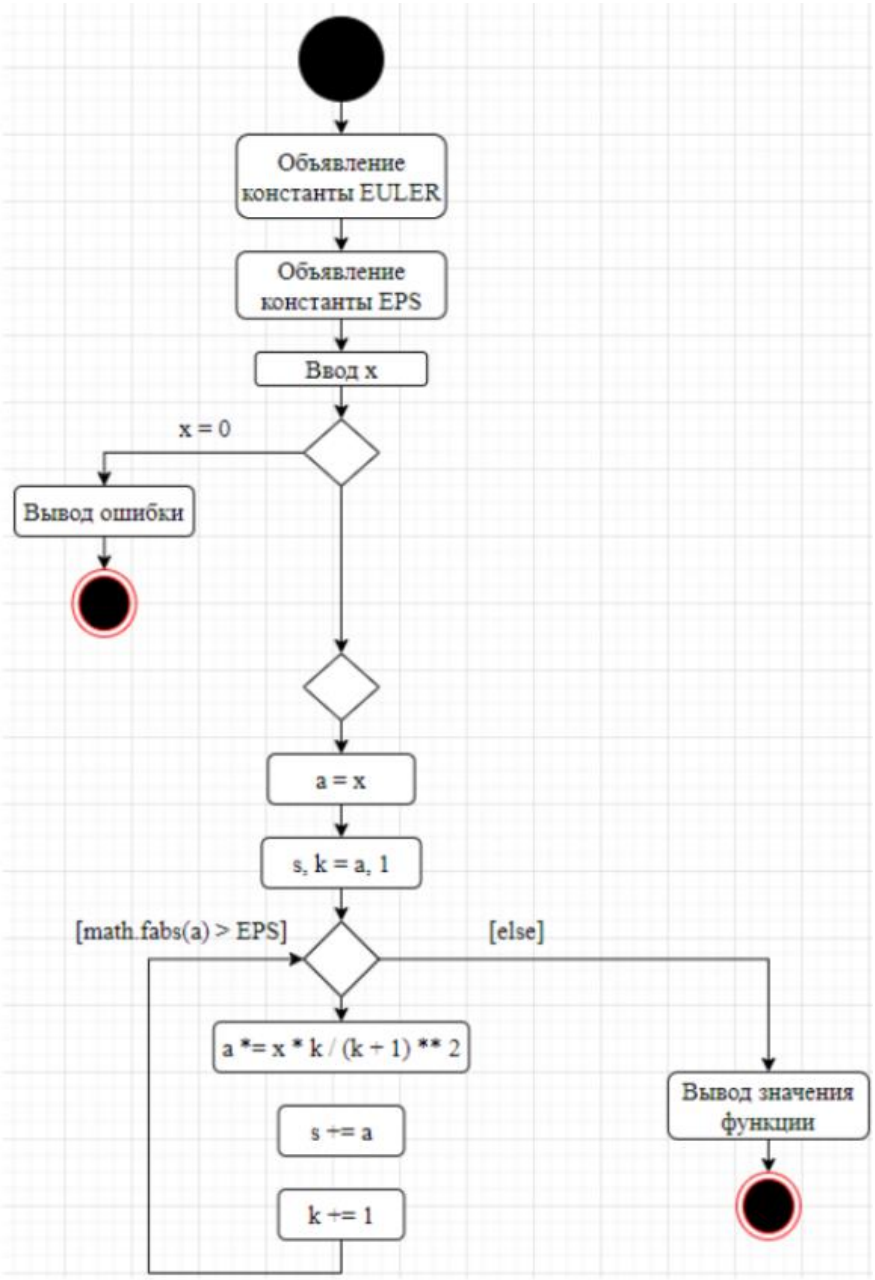

Рис. 17 – UML-диаграмма алгоритма

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

Рис. 18 – код программы

```
Value of x? 0
Illegal value of x

Process finished with exit code 1
```

Рис. 19 – пример работы программы lab.5_ex.5.py при x=0

```
Value of x? 4
Ei(4.0) = 19.63087447005282

Process finished with exit code 0
```

Рис. 20 – пример работы программы lab.5_ex.5.py при x =4

```
ex = int(input("Введите число экзаменов: "))
print(f"Мы успешно сдали {ex} экзаменов!")
```

Рис. 21 – индивидуальное задание №1

```
Введите число экзаменов: 25
Мы успешно сдали 25 экзаменов!

Process finished with exit code 0
```

Рис. 22 – пример работы индивидуального задания №1

```
a = float(input('Введите положительное число для \"a\": '))
b = float(input('Введите положительное число для \"b\": '))
M = input('Введите для точки \"M\" координаты X,Y: ').split(',')
X = float(M[0])
Y = float(M[1])
tan = b/a
if a>0 and b>0 and (Y/X < b/a):
    print('True')
else:
    print('False')
```

Рис. 23 – индивидуальное задание №2

```
Введите положительное число для "a": 9.6
Введите положительное число для "b": 7.5
Введите для точки "M" координаты X,Y: 7,1
True

Process finished with exit code 0
```

Рис. 24 – пример работы индивидуального задания №2

```
from datetime import date, timedelta

print((date(int(input('Year: ')), 1, 1) + timedelta(int(input('Day: ')) - 1)).strftime('%d %B'))
```

Рис. 23 – индивидуальное задание №3

```
Year: 2000
Day: 68
08 March

Process finished with exit code 0
```

Рис. 24 – пример работы индивидуального задания №3

Ответы на вопросы:

1. Для чего нужны диаграммы деятельности UML?

Диаграммы деятельности - это один из пяти видов диаграмм, применяемых в UML для моделирования динамических аспектов поведения системы. Диаграмма деятельности - это, по существу, блок-схема, которая показывает, как поток управления переходит от одной деятельности к другой, однако, по сравнению с последней, у ней есть явные преимущества: поддержка многопоточности и объектно-ориентированного проектирования. Диаграмма деятельности (Activity diagram) показывает поток переходов от одной деятельности к другой.

2. Что такое состояние действия и состояние деятельности?

Вы можете вычислить выражение, в результате чего изменяется значение некоторого атрибута или возвращается некоторое значение. Также, например, можно выполнить операцию над объектом, послать ему сигнал или даже создать его или уничтожить. Все эти выполняемые атомарные вычисления называются состояниями действия, поскольку каждое из них есть состояние системы, представляющее собой выполнение некоторого действия. Состояния действия не могут быть подвергнуты декомпозиции. Кроме того, они атомарны. Это значит, что внутри них могут происходить различные события, но выполняемая в состоянии действия работа не может быть прервана. Обычно предполагается, что длительность одного состояния действия занимает неощутимо малое время.

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. Предполагается, что для их завершения требуется заметное время. Можно считать, что состояние действия - это частный вид состояния деятельности, а конкретнее – такое состояние, которое не может быть подвергнуто дальнейшей декомпозиции. А состояние деятельности можно представлять себе как составное состояние, поток управления которого включает только другие состояния деятельности и действий.

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

В UML переход представляется простой линией со стрелкой. Точка ветвления представляется ромбом. В точку ветвления может входить ровно один переход, а выходить – два или более.

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Алгоритм 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?

Линейный алгоритм выполняется последовательно независимо от чего-либо, а алгоритм ветвления выполняется определенные действия в зависимости от выполнения условия или условий.

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

Оператор ветвления «if» позволяет выполнить определенный набор инструкций в зависимости от некоторого условия. Возможны следующие варианты использования:

- Конструкция «if»
- Конструкция «if» - «else»
- Конструкция «if» - «elif» - «else»

7. Какие операторы сравнения используются в Python?

<, >, <=, >=, ==, !=.

8. Что называется простым условием? Приведите примеры.

Логические выражения являются простыми, если в них выполняется только одна логическая операция. «x > 15» или «a != b»

9. Что такое составное условие? Приведите примеры.

В составных условиях используется 2 и более логические операции. «x > 8 and y <= 3»

10. Какие логические операторы допускаются при составлении сложных условий?

and и or

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

Да, может.

12. Какой алгоритм является алгоритмом циклической структуры?

Алгоритм 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.

Цикл «while» и цикл «for».

14. Назовите назначение и способы применения функции range.

Функция range возвращает неизменяемую последовательность чисел в виде объекта range. Параметры функции:

• start - с какого числа начинается последовательность. По умолчанию - 0

• stop - до какого числа продолжается последовательность чисел. Указанное число не включается в диапазон.

• step - с каким шагом растут числа. По умолчанию 1

Функция range хранит только информацию о значениях start, stop и step и вычисляет значения по мере необходимости. Это значит, что независимо от размера диапазона, который описывает функция range, она всегда будет занимать фиксированный объем памяти.

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

range(15, 0, -2).

16. Могул ли быть циклы вложенными?

Да, могут.

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

a = 0 while a >= 0:

if a == 7:

break a += 1

print("A")

Оператор «break» предназначен для досрочного прерывания работы цикла «while».

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» 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. По умолчанию функция print использует поток stdout. Хорошим стилем программирования является наличие вывода ошибок в стандартный поток stderr поскольку вывод в потоки stdout и stderr может обрабатываться как операционной системой, так и сценариями пользователя по-разному.

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. В данном примере выполняется вызов exit(1) , что приводит к немедленному завершению программы и операционной системе передается 1 в качестве кода возврата, что говорит о том, что в процессе выполнения программы произошли ошибки.