

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:
Ламская Ксения Вячеславовна
1 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и
сопровождение программного
обеспечения», очная форма обучения

(подпись)

Доцент кафедры инфокоммуникаций
Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

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.


Порядок выполнения работы

1. Создала репозиторий GitHub.

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 *	Repository name *
 ksenia-lamskaya ▾	/ 5laba
✔ 5laba is available.	

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

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 ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  `main` as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создание репозитория GitHub

2. Проработала примеры из лабораторной работы.

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

Рисунок 2.1 – Код из примера 1

```
PS C:\Ksen\51laba> & C:/Us
Value of x? -3
y = 17.010007503399553
PS C:\Ksen\51laba> & C:/Us
Value of x? 4
y = 5.0
PS C:\Ksen\51laba> & C:/Us
Value of x? 10
y = -100.54402111088937
PS C:\Ksen\51laba> █
```

Рисунок 2.2 – Вывод программы из примера 1

```
#!/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 2.3 – Код из примера 2

```
Введите номер месяца: 5
Весна
PS C:\Ksen\51aba> & C:/Us
Введите номер месяца: 14
Ошибка!
```

Рисунок 2.4 – Вывод программы из примера 2

```
#!/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
    for k in range(1, n + 1):
        a = math.log(k * x) / (k * k)
        S += a

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

Рисунок 2.5 – Код и примера 3

```
Value of n? 4
Value of x? 4
S = 2.355542280545169
```

Рисунок 2.6 – Вывод программы из примера 3

```
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 2.7 – Код из примера 4

```
Value of a? 5
x = 2.23606797749979
X = 2.23606797749979
PS C:\Ksen\51aba> & C
Value of a? -3
Illegal value of a
```

Рисунок 2.8 – Вывод программы из примера 4

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

EULER = 0.5772156649015329
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

    while math.fabs(a) > EPS:
        a *= x * k / (k + 1) ** 2
        S += a
        k += 1

    print(f"Ei({x}) = {EULER + math.log(math.fabs(x)) + S}")
```

Рисунок 2.9 – Код из примера 5

```
Value of x? 2.4
Ei(2.4) = 6.600670276342365
PS C:\Ksen\51aba> & C:/Users/
Value of x? 0
Illegal value of x
```

Рисунок 2.10 – Вывод программы из примера 5

3. UML-диаграммы для программ из примеров 4 и 5.

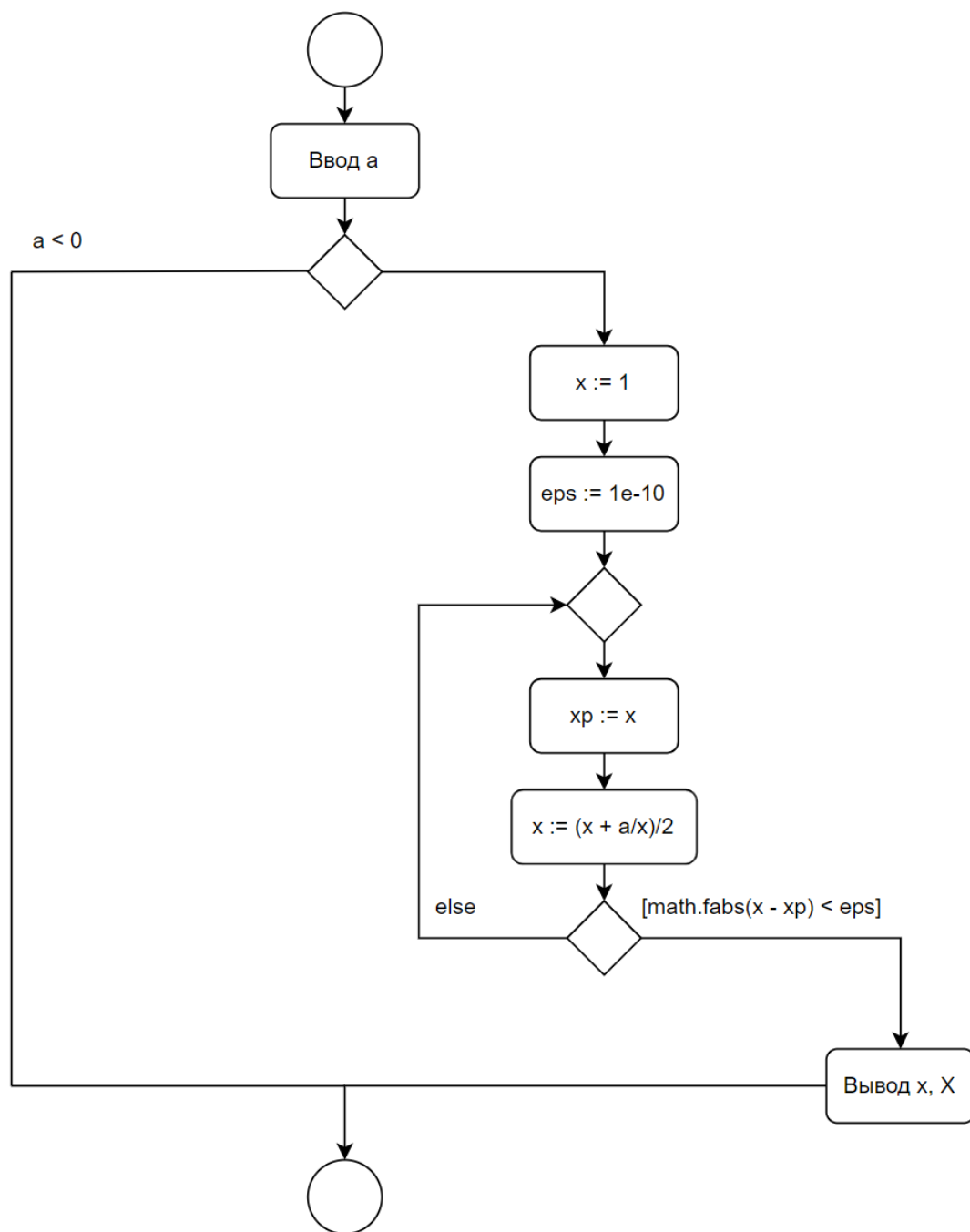


Рисунок 3.1 – UML-диаграмма для примера 4

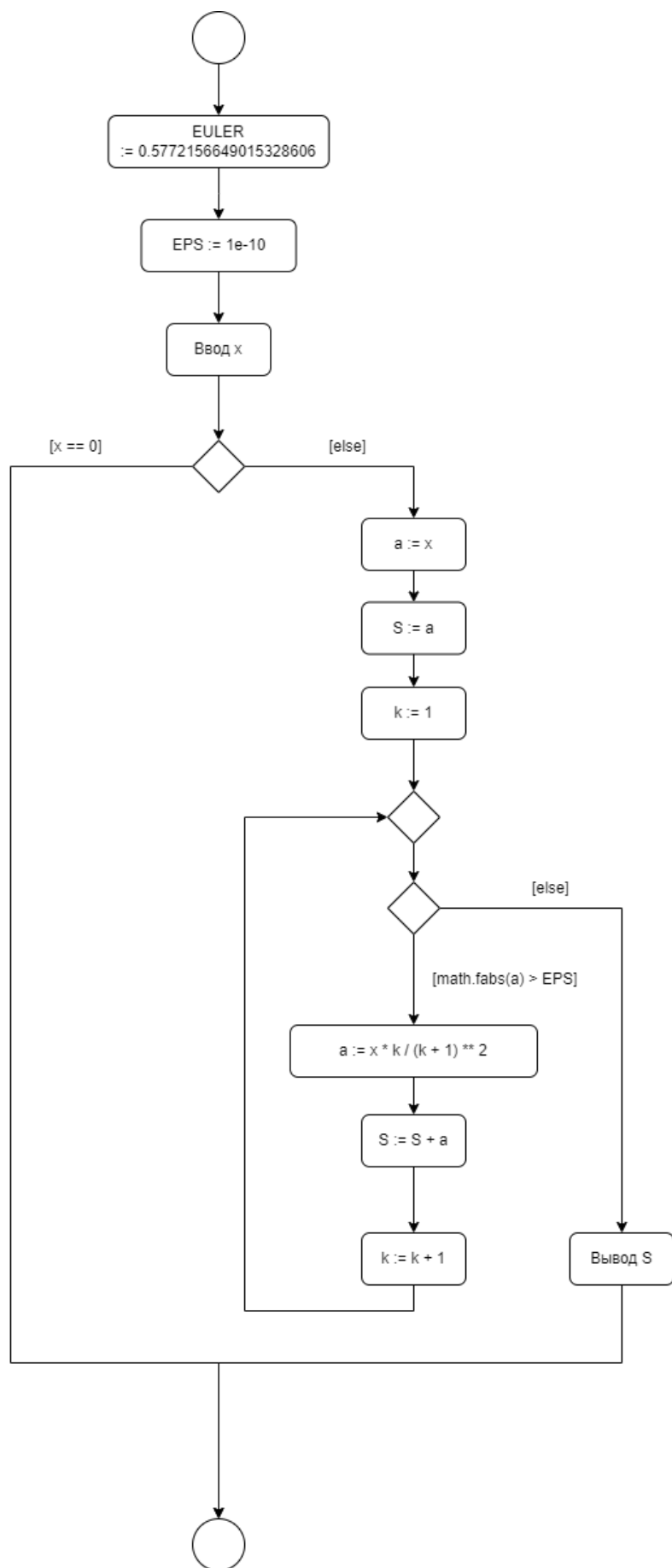


Рисунок 3.2 – UML-диаграмма для примера 5

9. Вводится число экзаменов $N \leq 20$. Напечатать фразу «Мы успешно сдали N экзаменов», согласовав слово экзамен с числом N .

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

if __name__ == '__main__':
    n = int(input("Введите количество экзаменов: "))

    if n == 1:
        print(f"Мы успешно сдали {n} экзамен")
    elif n == 2 or n == 3 or n == 4:
        print(f"Мы успешно сдали {n} экзамена")
    elif n >= 5 and n <= 20:
        print(f"Мы успешно сдали {n} экзаменов")
    else:
        print("Ошибка! Введите число меньше 21", file=sys.stderr)
        exit(1)
```

Рисунок 4.1 – Код программы

```
Введите количество экзаменов: 1
Мы успешно сдали 1 экзамен
PS C:\Ksen\51aba> & C:/Users/irbis/
Введите количество экзаменов: 4
Мы успешно сдали 4 экзамена
PS C:\Ksen\51aba> & C:/Users/irbis/
Введите количество экзаменов: 11
Мы успешно сдали 11 экзаменов
```

Рисунок 4.2 – Вывод программы

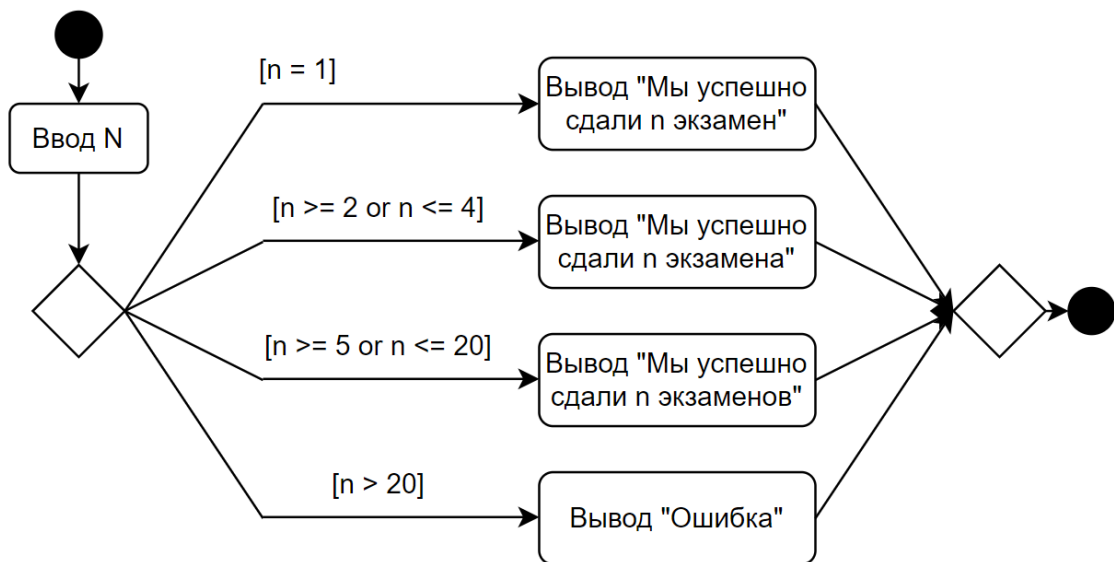


Рисунок 4.3 – UML-диаграмма

9. Найти координаты точек пересечения прямых, заданных уравнениями $a_1x + b_1y + c_1 = 0$ и $a_2x + b_2y + c_2 = 0$, либо сообщить совпадают, параллельны или не существуют .

```

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

if __name__ == '__main__':
    a1 = float(input("Введите a1: "))
    b1 = float(input("Введите b1: "))
    c1 = float(input("Введите c1: "))

    a2 = float(input("Введите a2: "))
    b2 = float(input("Введите b2: "))
    c2 = float(input("Введите c2: "))

    det = a1*b2 - a2*b1
    if det == 0:
        if a1*c2 == a2*c1 and b1*c2 == b2*c1:
            print("Прямые совпадают")
        else:
            print("Прямые параллельны")
    else:
        x = (b2*c1 - b1*c2)/det
        y = (a1*c2 - a2*c1)/det
        print(f"Точка пересечения прямых: ({x}, {y})")
  
```

Рисунок 5.1 – Код программы

```
Введите a1: 2
Введите b1: 3
Введите c1: 4
Введите a2: 3
Введите b2: 5
Введите c2: 6
Точка пересечения прямых: (2.0, 0.0)
```

Рисунок 5.2 – Вывод программы

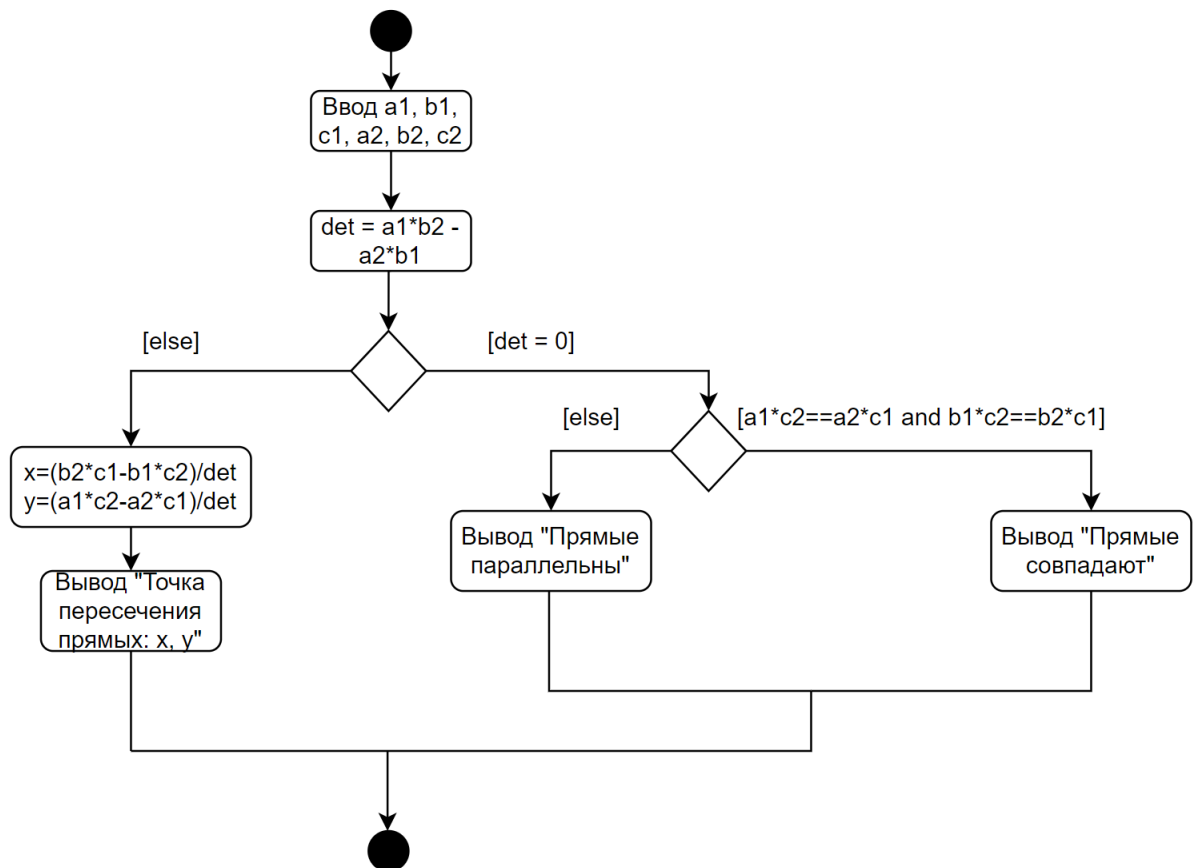


Рисунок 5.3 – UML-диаграмма

9. Если к сумме цифр двузначного числа прибавить квадрат этой суммы, то снова получится это двузначное число. Найти все эти числа.

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

for i in range(10, 100):
    tens = i // 10
    ones = i % 10
    if (tens + ones) + (tens + ones) ** 2 == i:
        print(i)
```

Рисунок 6.1 – Код программы

```
12
42
90
```

Рисунок 6.2 – Вывод программы

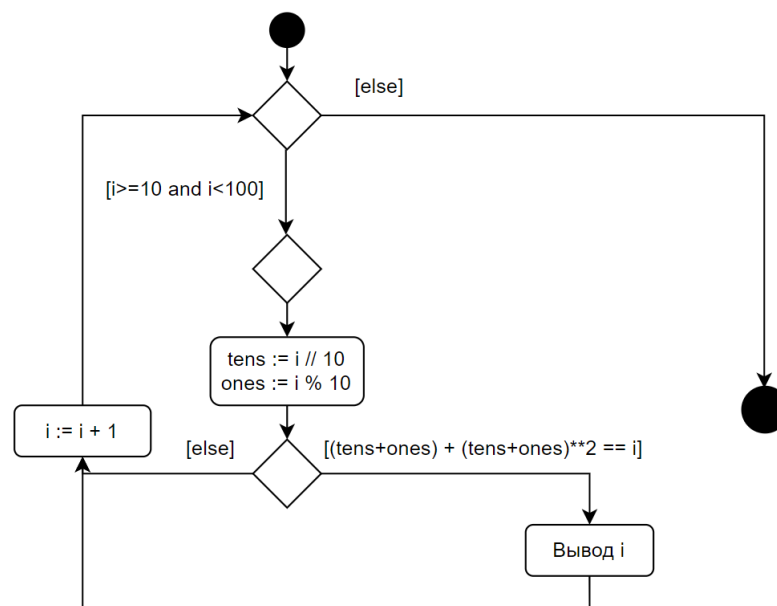


Рисунок 6.3 – UML-диаграмма

10. Задание повышенной сложности Вариант 9. Вычислить данную функцию по разложению в ряд.

$$f(x) = - \int_1^x \frac{\ln t}{t-1} dt = \sum_{k=1}^{\infty} \frac{(-1)^k (x-1)^k}{k^2}, \quad 0 \leq x \leq 2.$$

Рисунок 7.1 – Исходная функция

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

EPS = 1e-10

if __name__=="__main__":
    x = float(input("Value of x?: "))
    if x < 0 and x > 2:
        print("Illegalvalue of x")
        exit(1)

    a = x
    S, k = a, 1

    while math.fabs(a) > EPS:
        a = (((-1)**k * (x - 1)**k)/k**2)
        S += a
        k += 1

    print(f'f(x) = {S}')
```

Рисунок 7.2 – Код программы

```
Value of x?: 1.2
f(x) = 1.0091998622265996
```

Рисунок 7.3 – Вывод программы

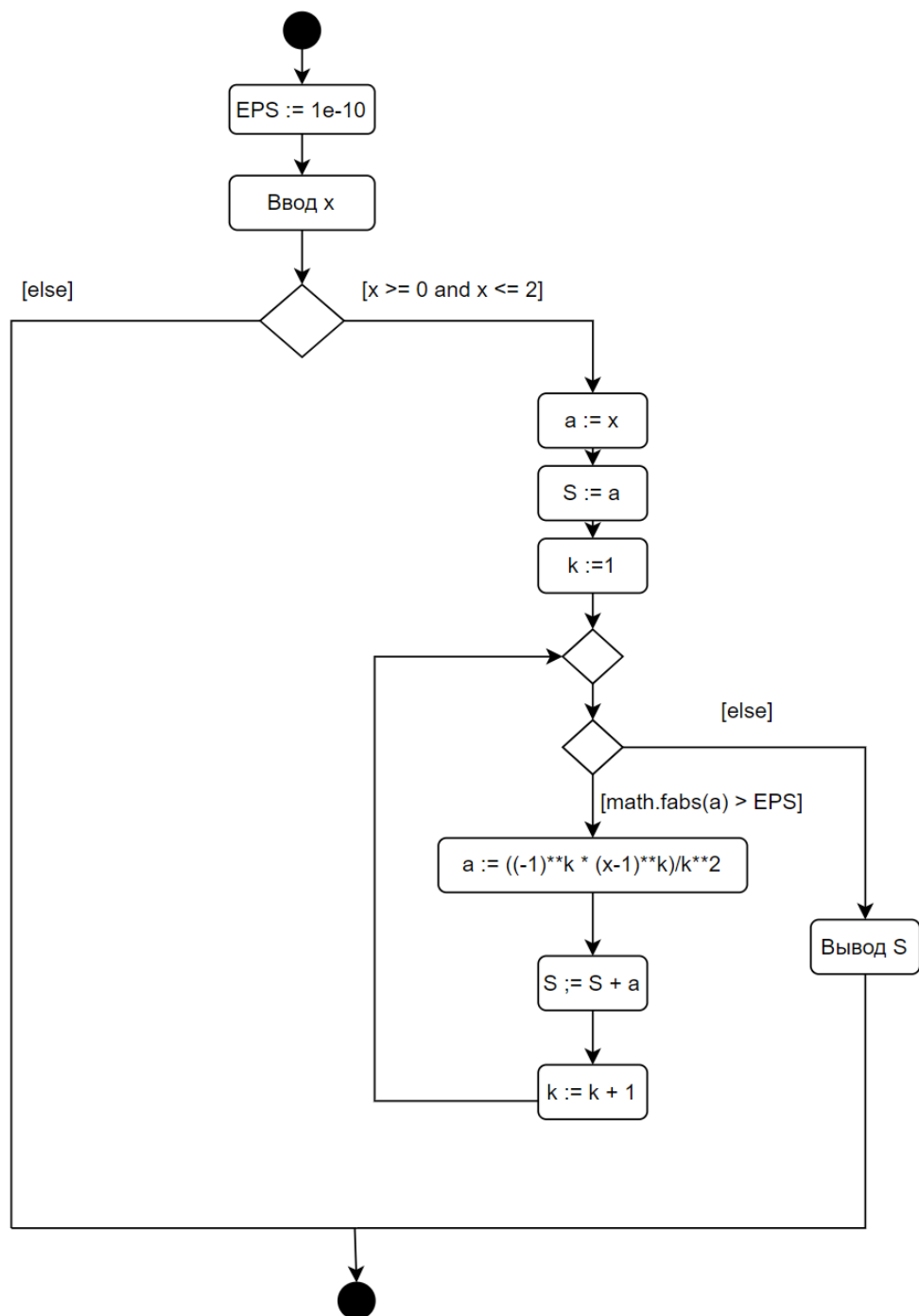


Рисунок 7.4 – UML-диаграмма

Ответы на контрольные вопросы

1. Для чего нужны Диаграммы деятельности в UML? Диаграммы деятельности в UML (Unified Modeling Language) используются для моделирования и визуализации процесса выполнения действий или задач в системе. Они представляют собой графическое представление последовательности действий, включая условия, переходы и ветвления, и

помогают разработчикам и заинтересованным сторонам лучше понять и описать работу системы.

2. Что такое состояние действия и состояние деятельности?

Состояние действия (action state) в диаграммах деятельности описывает мгновенное выполнение операции или действия. Это простое действие, которое выполняется без переходов в другие состояния.

Состояние деятельности (activity state) представляет собой группу действий, которые выполняются вместе. В диаграммах деятельности оно может быть использовано для описания более сложных действий, которые состоят из нескольких этапов.

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

- Переходы (Transitions) обычно обозначаются стрелками и могут иметь условное выражение или событие, при котором происходит переход.
- Ветвления (Decisions) обычно обозначаются ромбовидными формами и представляют точки принятия решений. Они могут иметь один или несколько исходящих переходов, каждый из которых имеет условное выражение.
- Слияния (Merge) обычно обозначаются стрелками, сходящимися к одной точке, и представляют объединение потоков выполнения в один.

4. Какой алгоритм является алгоритмом разветвляющейся структуры? Алгоритм разветвляющейся структуры (conditional structure) представляет собой алгоритм, в котором происходит разделение потока выполнения на два или более возможных направления в зависимости от

условия. Примером такого алгоритма может быть условный оператор if-else в языке программирования.

5. Чем отличается разветвляющийся алгоритм от линейного?

Разветвляющийся алгоритм отличается от линейного алгоритма тем, что он позволяет выбирать одну из нескольких возможных ветвей выполнения в зависимости от условия. Такой алгоритм может иметь различные пути выполнения, в то время как линейный алгоритм имеет только один последовательный путь выполнения без разветвлений.

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

Условный оператор (conditional statement) — это конструкция в программировании, которая позволяет выполнить определенные действия в зависимости от условия, которое может быть истинным или ложным. В Python условный оператор обычно представлен конструкцией if-elif-else. Есть также форма условного оператора без else, называемая простым условным оператором.

7. Какие операторы сравнения используются в Python? В Python используются следующие операторы сравнения:

- == (равно)
- != (не равно)
- > (больше)
- < (меньше)
- >= (больше или равно)
- <= (меньше или равно)

8. Что называется простым условием? Приведите примеры. Простое условие (simple condition) в условном операторе представляет собой одно условие, которое может быть истинным или ложным.

9. Что такое составное условие? Приведите примеры. Составное условие (compound condition) в условном операторе представляет собой комбинацию нескольких простых условий с помощью логических операторов.

10. Какие логические операторы допускаются при составлении сложных условий? При составлении сложных условий в условном операторе в Python допускаются следующие логические операторы:

- and (логическое И)
- or (логическое ИЛИ)
- not (логическое НЕ)

11. Может ли оператор ветвления содержать внутри себя другие ветвления? Да, оператор ветвления в Python может содержать внутри себя другие ветвления. Это позволяет создавать более сложные иерархии условий и выполнять различные действия в зависимости от набора условий.

12. Какой алгоритм является алгоритмом циклической структуры? Алгоритм циклической структуры (loop structure) представляет собой алгоритм, в котором определенный блок кода выполняется несколько раз в зависимости от условия. Примером такого алгоритма может быть цикл while или for в языке программирования.

13. Типы циклов в языке Python. В языке Python существуют следующие типы циклов:

- Цикл for используется для итерации по последовательности или коллекции элементов.
- Цикл while выполняется, пока условие истинно.

14. Назовите назначение и способы применения функции range. Функция range в Python используется для создания последовательности чисел.

Она возвращает итерируемый объект, который может быть использован в циклах for. Функция range может принимать от одного до трех аргументов: range(stop), range(start, stop), range(start, stop, step).

15. Как с помощью функции range организовать перебор значений от 15 до 0 с шагом 2? Для организации перебора значений от 15 до 0 с шагом 2 с помощью функции range можно использовать следующий код:

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

16. Могут ли быть циклы вложенными? Да, в языке программирования Python циклы могут быть вложенными, то есть один цикл может находиться внутри другого цикла. Это позволяет создавать более сложные алгоритмы и выполнять итерации в нескольких уровнях.

17. Как образуется бесконечный цикл и как выйти из него? Бесконечный цикл (infinite loop) образуется, когда условие цикла всегда истинно, или когда цикл не имеет условия выхода. Для выхода из бесконечного цикла можно использовать операторы break или return, которые прерывают выполнение цикла и переносят управление за его пределы.

18. Для чего нужен оператор break? Оператор break используется в циклах для немедленного прекращения выполнения цикла и выхода из него. Когда break достигается, программа продолжает выполнение со следующего после цикла оператора. Оператор break особенно полезен для прерывания циклов в зависимости от определенных условий.

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

Оператор `continue` используется в циклах для пропуска текущей итерации и перехода к следующей итерации цикла. Когда `continue` достигается, оставшаяся часть текущего цикла не выполняется, и управление передается обратно к началу цикла для следующей итерации. Оператор `continue` позволяет пропустить определенные шаги цикла в зависимости от условий.

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

Стандартные потоки `stdout` (стандартный поток вывода) и `stderr` (стандартный поток ошибок) используются для обработки вывода программы.

`stdout` используется для вывода обычной информации, результатов работы программы и других сообщений.

`stderr` используется для вывода сообщений об ошибках и предупреждений.

Обычно `stdout` отображается в консоли или сохраняется в файл, в то время как `stderr` также может быть перенаправлен в отдельный файл или игнорироваться.

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

Чтобы организовать вывод в стандартный поток `stderr` в Python, можно использовать модуль `sys` и его атрибут `stderr`:

22. Каково назначение функции `exit`?

Функция `exit` в Python используется для немедленного выхода из программы. Когда вызывается эта функция, выполнение программы прекращается, и управление возвращается операционной системе или среде выполнения. Функция `exit` принимает необязательный аргумент, который может использоваться для указания кода завершения программы.