

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 №2.2
дисциплины «Программирование на Python»
Вариант ____

Выполнил:
Иващенко Олег Андреевич
2 курс, группа ИВТ-б-о-22-1,
09.03.02 «Информационные и
вычислительные машины»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем»

(подпись)

Руководитель практики:
Воронкин Роман Александрович,
доцент кафедры инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 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.

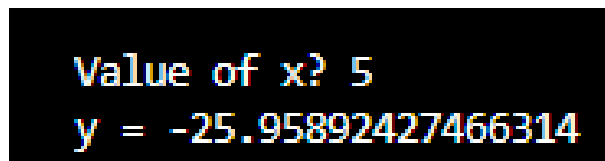
Порядок выполнения работы

Таблица 1 – Код программы example_1.py

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



The image shows a terminal window with a black background. The text 'Value of x? 5' is displayed in a yellow, monospaced font. Below it, the text 'y = -25.95892427466314' is displayed in the same font. The text is slightly blurred, suggesting it was captured from a video or a fast-moving image.

Рисунок 1 – Вывод программы example_1.py

Таблица 2 – Код программы example_2.py

```
#!/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)
```



Введите номер месяца: 12
Зима

Рисунок 2 – Вывод программы example_2.py

Таблица 3 – Код программы example_3.py

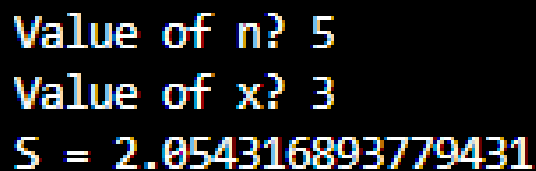
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

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}")
```



Value of n? 5
Value of x? 3
S = 2.054316893779431

Рисунок 3 – Вывод программы example_3.py

Таблица 4 – Код программы example_4.py

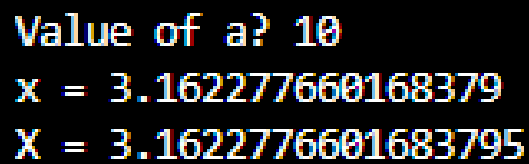
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

```
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)}")
```



```
Value of a? 10
x = 3.162277660168379
X = 3.1622776601683795
```

Рисунок 4.1 – Вывод программы example_4.py

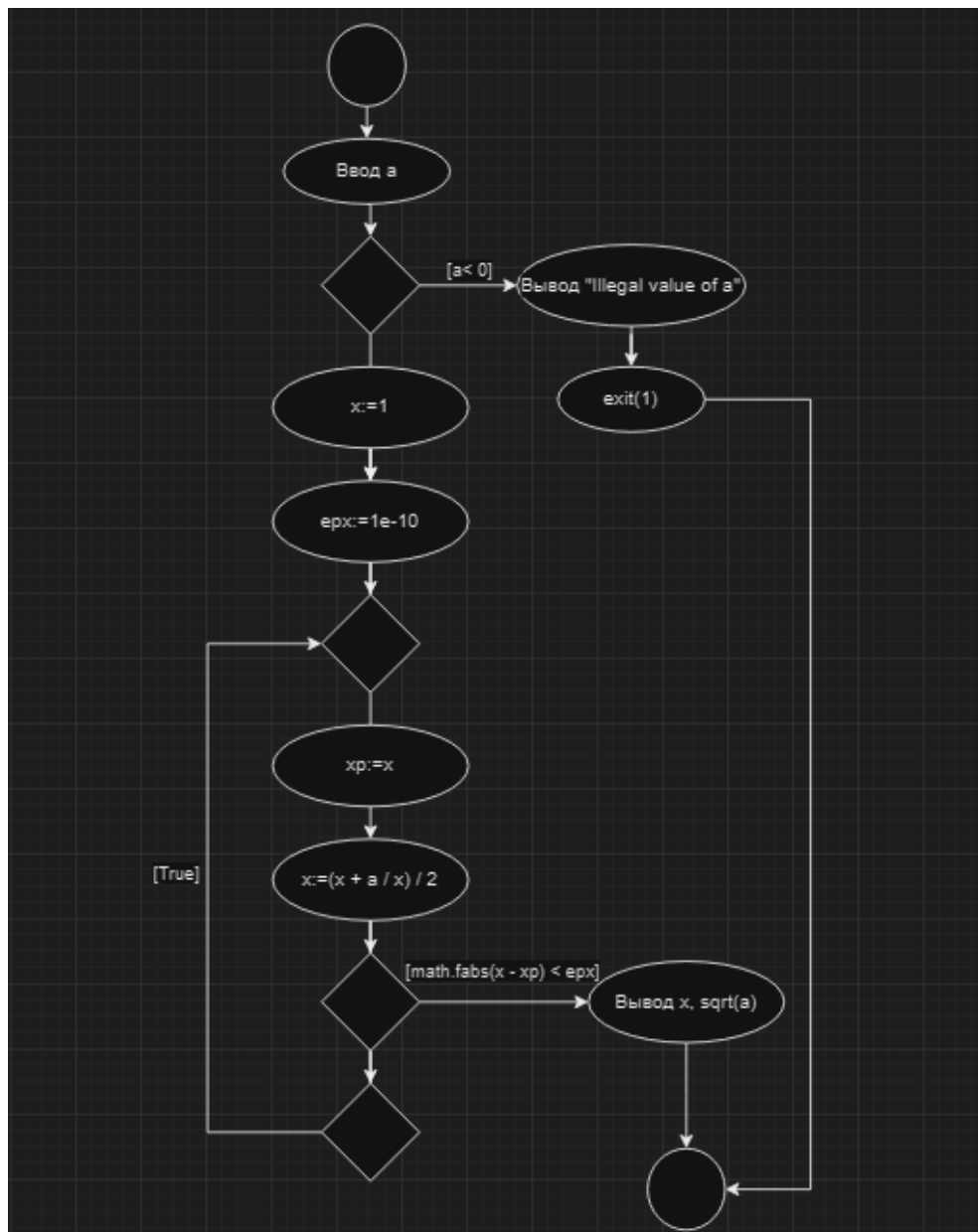


Рисунок 4.2 – Блок-схема программы example_4.py

Таблица 5 – Код программы example_5.py

```

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

import math
import sys

# Постоянная Эйлера.
EULER = 0.5772156649015328606
# Точность вычислений.
EPS = 1e-10

if __name__ == '__main__':

```

```

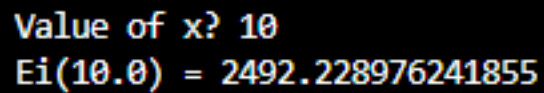
x = float(input("Value of x? "))
if x == 0:
    print("Illegal value of x", file=sys.stderr)
    exit(1)

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}")

```



```

Value of x? 10
Ei(10.0) = 2492.228976241855

```

Рисунок 5.1 – Вывод программы example_5.py

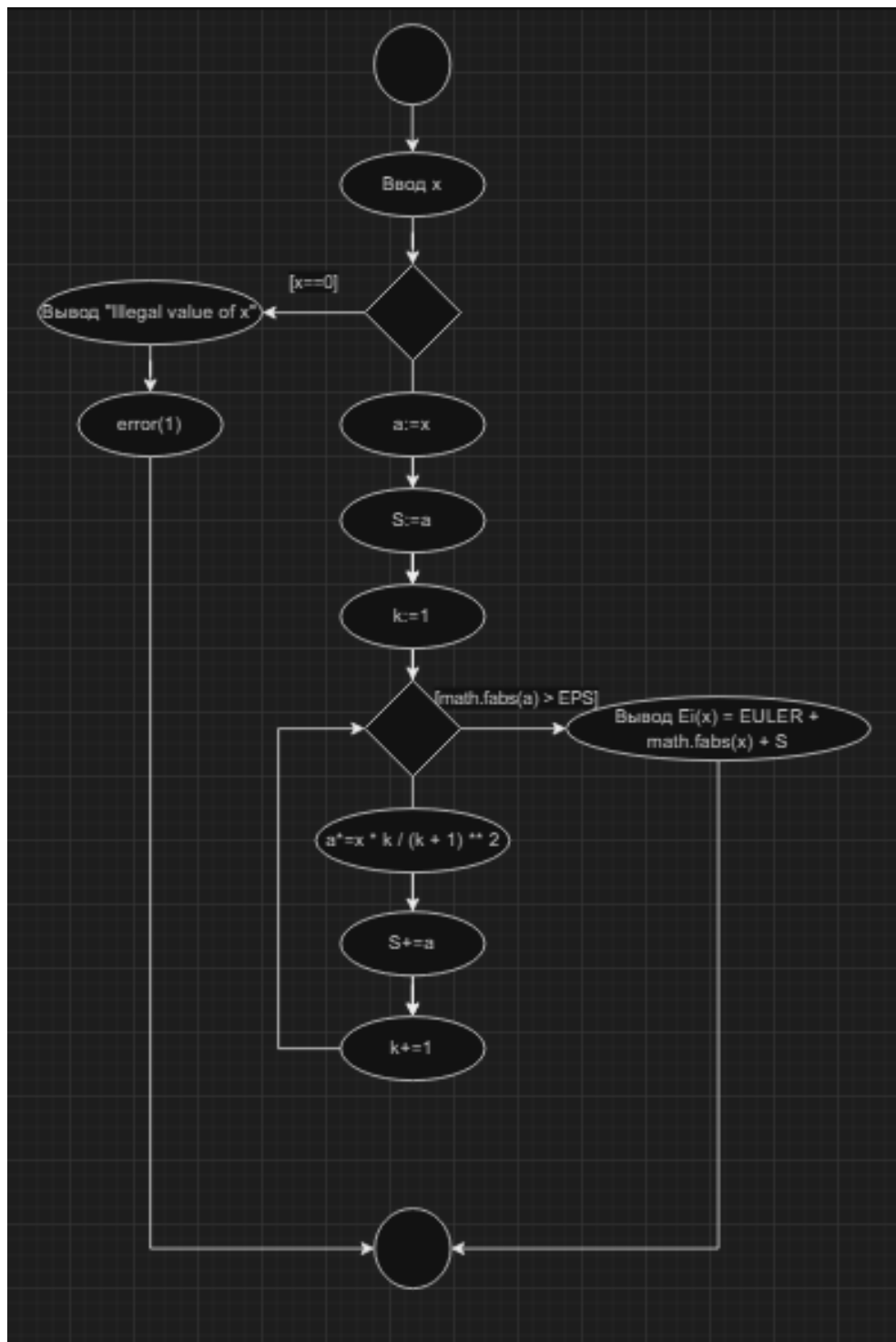
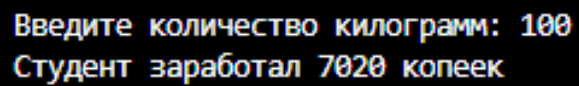


Рисунок 5.2 – Блок-схема программы example_5.py

Индивидуальное задание 1. Студенты убирают урожай помидоров. При сборе до 50 кг в день работа оплачивается из расчёта 30 коп. за 1 кг; при сборе от 50 до 75 кг в день - 50 коп. за 1 кг; при сборе от 75 до 90 кг в день - 65 коп. за 1 кг; при сборе свыше 90 кг в день - 70 коп. за 1 кг плюс 20 руб. премия. Студент собрал X кг за день. Определить его заработок.

Таблица 6 – Код программы individual_1.py

```
if __name__ == "__main__":  
    kg = int(input("Введите количество килограмм: "))  
    pay = 0  
  
    if kg < 0:  
        print("Введено неправильное число")  
        exit(1)  
  
    if kg < 50:  
        pay = kg * 30  
    elif kg >= 50 and kg < 75:  
        pay = kg * 50  
    elif kg >= 75 and kg <= 90:  
        pay = kg * 65  
    elif kg > 90:  
        pay = kg * 70 + 20  
  
    print(f"Студент заработал {pay} копеек")
```

A screenshot of a terminal window with a black background and yellow text. It shows the program's output for an input of 100 kilograms, resulting in a payment of 7020 kopecks.

Введите количество килограмм: 100
Студент заработал 7020 копеек

Рисунок 6.1 – Вывод программы individual_1.py

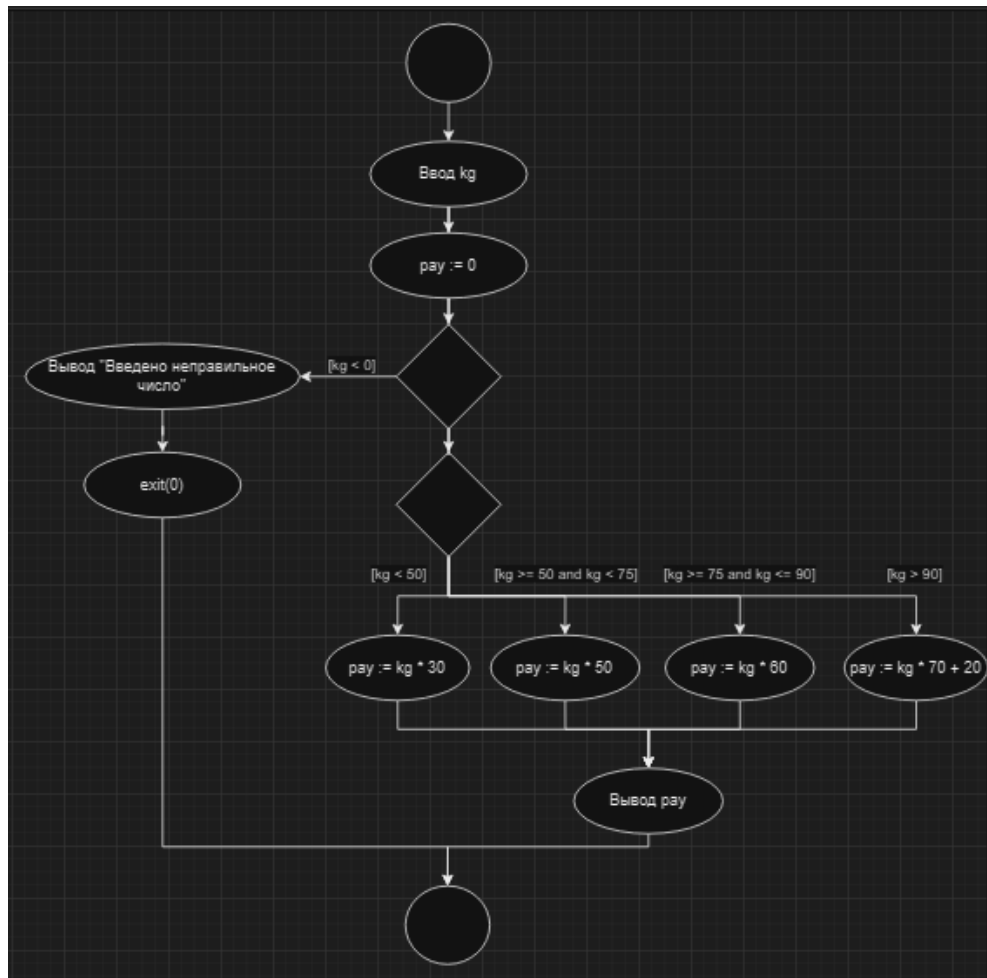


Рисунок 6.2 – Блок-схема программы individual_1.py

Индивидуальное задание 2. Составить программу, выясняющую, делится ли натуральное число x нацело на натуральное число y .

Таблица 7 – Код программы individual_2.py

```

if __name__ == "__main__":
    x = int(input("Введите значение x: "))
    y = int(input("Введите значение y: "))

    if x / y == x // y:
        print(f"{x} делится на {y} целочисленно. {x} / {y} = {x / y}")
    else:
        print(f"{x} не делится на {y} целочисленно. {x} / {y} = {x / y}")
  
```

```
Введите значение x: 10
Введите значение y: 2
10 делится на 2 целочисленно. 10 / 2 = 5.0
```

Рисунок 7.1 – Вывод программы individual_2.py в случае безостаточного деления

```
Введите значение x: 15
Введите значение y: 10
15 не делится на 10 целочисленно. 15 / 10 = 1.5
```

Рисунок 7.2 – Вывод программы individual_2.py в случае деления с остатком

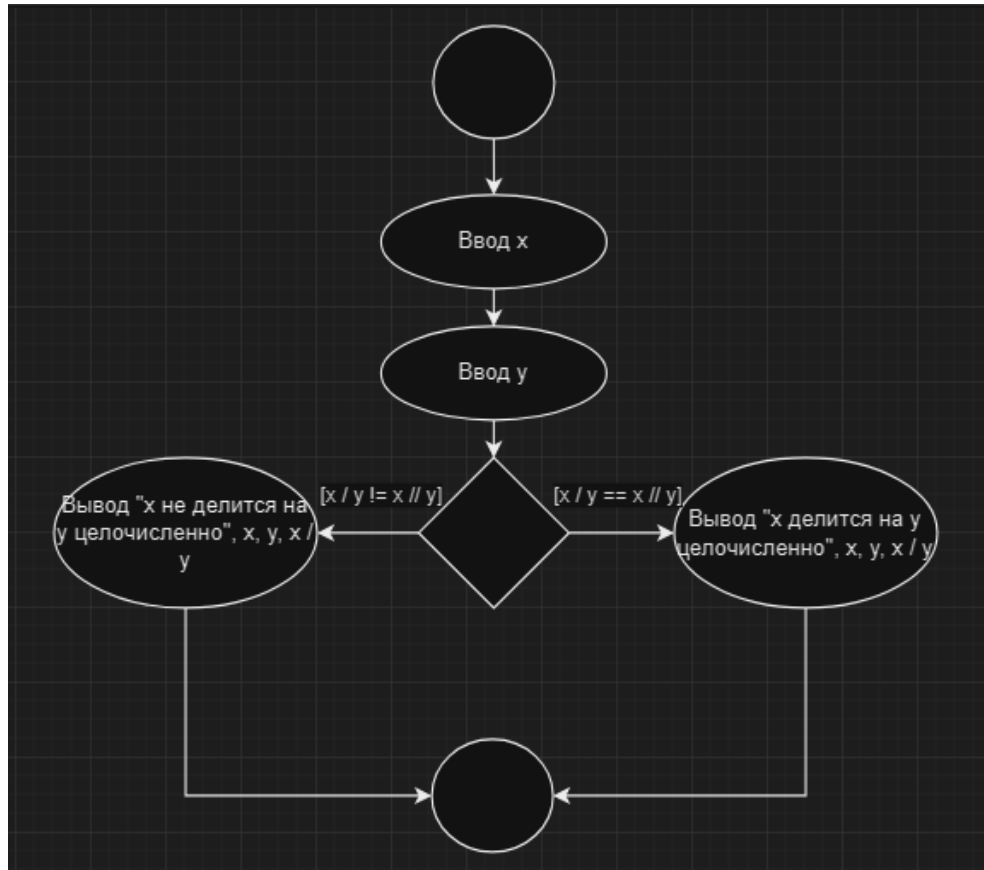


Рисунок 7.3 – Блок-схема программы individual_2.py

Индивидуальное задание 3. Дано натуральное число n . Определить все его натуральные делители.

Таблица 8 – Код программы individual_3.py

```
if __name__ == "__main__":
    n = int(input("Введите значение N: "))

    print(f"Натуральные делители {n}:")
```

```
for k in range(1, n):  
    if n / k == n // k:  
        print(f"{n} / {k} = {n // k}")  
        k += 1
```

```
Введите значение N: 15  
Натуральные делители 15  
15 / 1 = 15  
15 / 3 = 5  
15 / 5 = 3
```

Рисунок 8.1 – Вывод программы individual_3.py

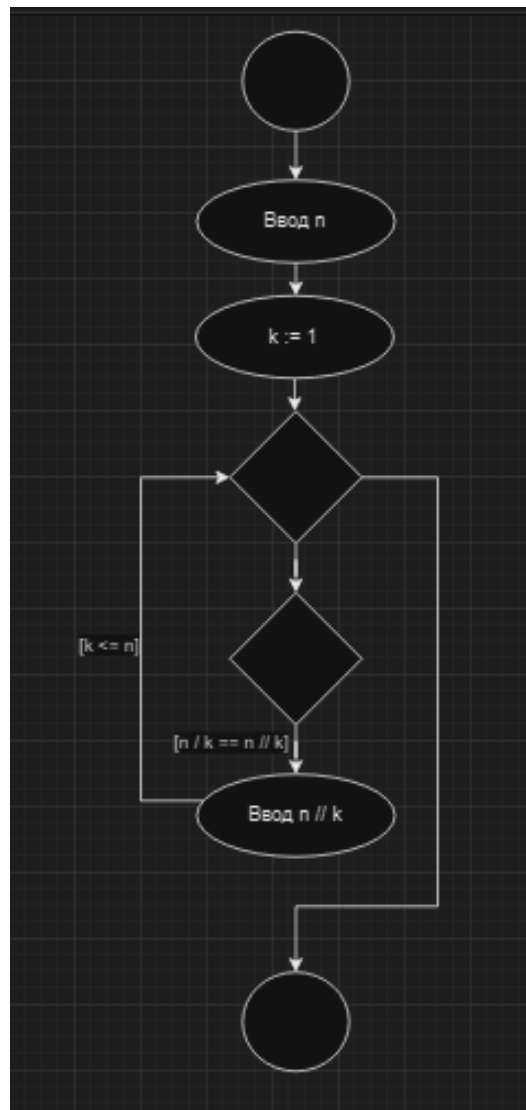


Рисунок 8.2 – Блок-схема программы individual_3.py

Задание повышенной сложности. Составить UML-диаграмму деятельности, программу и произвести вычисления значения специальной функции по ее разложению в ряд с точностью , аргумент функции вводится с клавиатуры.

$$\text{Chi}(x) = \gamma + \ln x + \int_0^x \frac{e^t - 1}{t} dt = \gamma + \ln x + \sum_{n=1}^{\infty} \frac{x^{2n}}{(2n)(2n)!}.$$

Рисунок 9.1 – Формула интегрального гиперболического косинуса

Таблица 9 – Код программы inc_def.py

```
import math

if __name__ == "__main__":
    x = int(input("Введите значение x: "))
    EULER = 0.5772156649015328606
    Chix, n = 0, 1

    while True:
        temp = ((-1) ** n * x ** (2 * n)) / ((2 * n) * math.factorial(2 * n))
        if abs(Chix) < 1e-10:
            break

        Chix += temp
        n += 1

    print(f"Chi(x) = {EULER + math.log10(x) + Chix}")
```

Введите значение x: 10
Chi(x) = 1.5772156649015328

Рисунок 9.2 – Вывод программы inc_def.py

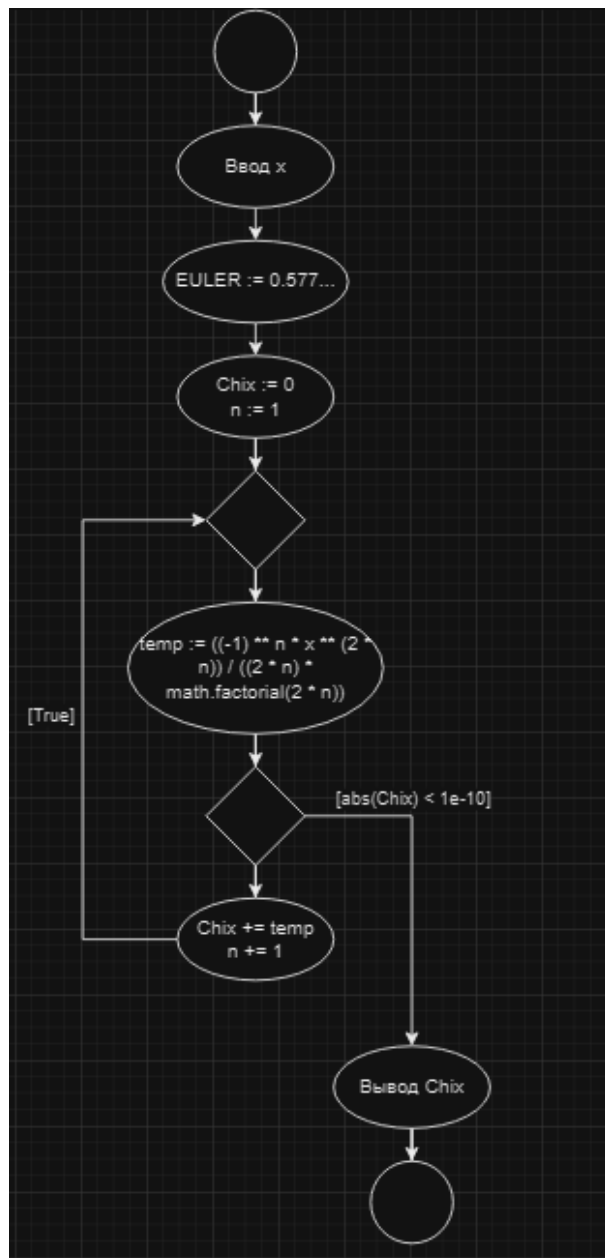


Рисунок 9.3 – Блок-схема программы inc_def.py

Контрольные вопросы

1. Для чего нужны диаграммы деятельности UML?

Диаграмма деятельности показывает поток переходов от одной деятельности к другой. Деятельность – это продолжающийся во времени неатомарный шаг вычислений в автомате. Деятельность в конечном итоге приводит к выполнению некоего действия, составленного из выполняемых атомарных вычислений, каждое из которых либо изменяет состояние системы, либо возвращает какое-то значение.

2. Что такое состояние действия и состояние деятельности?

Состояние действия представляет собой точку в процессе, где выполняется какое-то действие или операция. Используется для обозначения конкретного действия или операции, которая выполняется в данном этапе процесса.

Состояние деятельности представляет собой более общую концепцию и обозначает фазу, в которой система находится в процессе выполнения какой-то деятельности. Используется для обозначения более широких блоков выполнения, включающих в себя несколько действий. Состояние деятельности может охватывать последовательность действий или вложенных подсостояния.

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

В диаграммах деятельности для обозначения переходов и ветвлений применяются следующие нотации:

- Стрелка, указывающая направление перехода между состояниями или действиями;
- Ветвления – линия, разделяющаяся на две или более линии, представляющие параллельные пути выполнения;
- Слияния – линии, сходящиеся от двух или более линий, представляющих собой объединение параллельных путей выполнения.

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 без разветвлений, тогда как разветвляющийся алгоритм содержит условия или ветвления, позволяющие изменять ход выполнения в зависимости от условий или входных данных.

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

Условный оператор в языке программирования Python используется для выполнения блока кода в зависимости от истинности или ложности некоторого условия. Основной формой считается «if-else», однако можно использовать также формы «if», «if-elif» и «if-elif-else».

7. Какие операторы сравнения используются в Python?

В языке Python используются следующие операторы сравнения:

- «==» (равно) - проверяет, равны ли два значения;
- «!=» (не равно) - проверяет, не равны ли два значения;
- «<» (меньше) – проверяет, меньше ли левое значение правого;
- «<=» (меньше или равно) – проверяет, меньше или равно ли левое значение правого;
- «>» (больше) – проверяет, больше ли левое значение правого;
- «>=» (больше или равно) – проверяет, больше или равно ли левое значение правого;

8. Что называется простым условием? Приведите примеры.

Простое условие – это условие, которое проверяет истинность или ложность одного выражения. Пример:

`x = 5`

`if x == 5:`

9. Что такое составное условие? Приведите примеры.

Составное условие – это комбинация нескольких простых условий, объединённых логическими операторами. Пример:

age = 25

if age >= 18 and age <= 30:

10. Какие логические операторы допускаются при составлении сложных условий?

При составлении сложных условий допускаются следующие операторы:

- «and» (логическое «И») – возвращает true, если условия истинны, или false, если одно из условий ложно;
- «or» (логическое «ИЛИ») – возвращает true, если хотя бы одно из условий истинно;
- «not» (логическое «НЕ») – инвертирует значение условия.

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

Оператор ветвления может содержать внутри себя другие ветвления, создавая вложенные условные конструкции.

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.

В языке программирования Python существует несколько типов циклов:

- Цикл «for» - перебирает элементы в итерируемом объекте (списке, строке). Пример использования – *for i in range(5):*
- Цикл «while» - выполняет блок кода, пока условие остаётся истинным. Пример использования – *while i < 5:*

- Цикл «for..in» - перебирает элементы в итерируемом объекте, таком как список или кортеж. Пример использования:

```
fruits = ["яблоко", "груша", "апельсин"]
```

```
for fruit in fruits:
```

14. Назовите назначение и способы применения функции range.

Функция range() в языке программирования Python используется для создания последовательности чисел в указанном диапазоне. Она облегчает создание итерируемых объектов, таких как списки, используемых в циклах.

Способы применения:

- Создание последовательности чисел от начального (включительно) до конечного (исключая) значения с определённым шагом – `numbers = list(range(1, 10, 2))`
- Использование в цикле `for` – `for i in range(5):`
- Генерация последовательности для создания списка – `numbers = list(range(5))`

15. Как с помощью функции range организовывать перебор значения от 15 до 0 с шагом 2?

```
range(15, -1, -2)
```

16. Могут ли быть циклы вложенными?

Циклы могут быть вложенными в языке программирования Python.

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

Бесконечный цикл образуется, когда условие цикла всего истинно. Это может произойти, например, при использовании цикла «while» без явного изменения переменной условия внутри блока цикла.

Чтобы выйти из бесконечного цикла, нужно использовать оператор break внутри блока.

18. Для чего нужен оператор break?

Оператор «break» позволяет прервать выполнения цикла и перейти к следующему за блоком цикла оператору.

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

Оператор «continue» используется в циклах в языке программирования Python и служит для пропуска текущей итерации цикла и перехода к следующей.

Встречая оператор «continue» внутри цикла, выполнения последующего кода в блоке цикла прерывается и начинается заново с следующей итерацией цикла.

20. Для чего нужны стандартные потоки stdout и stderr?

Стандартный вывод (stdout) используется для вывода обычных данных, результатов работы программы, сообщений и т.д. По умолчанию, вывод «stdout» направляется на консоль.

Стандартный поток ошибок (stderr) используется для вывода сообщений об ошибках и предупреждений. Этот поток также направляется на консоль, но может быть перенаправлен в другие места для логирования и обработок ошибок.

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

В Python для организации вывода в стандартный поток ошибок (stderr) можно воспользоваться библиотекой «sys». Пример использования:

```
sys.stderr.write("Сообщение, выводимое в stderr")
```

22. Каково назначение функции exit?

Функция exit() в языке программирования Python используется для немедленного завершения выполнения программы. Код возврата 0 указывает

на успешное завершение программы, а другие значения могут использоваться для указания на различные ошибочные ситуации.

Выводы: В процессе выполнения лабораторной работы были приобретены навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры, освоены операторы языка Python версии 3.x if, while, for, break и continue, написаны 9 программ (5 примеров, 3 индивидуальных задания и 1 задание повышенной сложности), а так же построены 6 ULM-диаграмм деятельности.