**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІМЕНІ ТАРАСА ШЕВЧЕНКА**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**КАФЕДРА ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ**

Індивідуальне завдання № 2  
з дисципліни «Алгоритмізація та програмування»

Тема роботи: «Наближені обчислення та

рекурентні співвідношення.»

**Варіант** **№ 8**

Виконав(-ла) студент(-ка)

групи АнД-11

Яковкін Микола Андрійович

Перевірив(-ла):

Київ 2021

**Головна програма**

* 1. **Математична постановка задачі (МПЗ).**

*Вхідні дані:*

initial\_border, final\_border – границі інтервалу, дійсні числа.

amount\_of\_steps – кількість точко табулювання, ціле число.

*Вихідні дані:*

iteratedFunction(x) – значення епсилон, вирахуване ітеративною функцією, дійсне число

checkingFunction(x) – значення епсилон, вирахуване бібліотечною функцією, дійсне число.

RecursedFunction(x) – значення епсилон, вирахуване рекурсивною функцією, дійсне число.

*Математична постановка здачі:*

initial\_border = |initial\_border|, final\_border = |final\_border|,

amount\_of\_steps = |amount\_of\_steps|;

difference = [final\_border – initial\_border] / amount\_of\_steps;

Доки i = 1, amount\_of\_steps:

Виведення значень: iteratedFunction(initial\_border), checkingFunction(initial\_border), recursedFunction(initial\_border);

initial\_border = initial\_border + difference;

Подання математичної постановки задачі у вигляді таблиці:

Таблиця 1.1

|  |  |  |
| --- | --- | --- |
| **Вхідні дані** | **Дії** | **Вихідні дані** |
| initial\_border, final\_border - дійсного типу.  amount\_of\_steps - дійсного типу. | initial\_border = |initial\_border|, final\_border = |final\_border|,  amount\_of\_steps=|amount\_of\_steps|;  difference = [final\_border-initial\_border] / amount\_of\_steps;    Доки i = 1, amount\_of\_steps:  Виведення значень: iteratedFunction(initial\_border), checkingFunction(initial\_border), recursedFunction(initial\_border);  initial\_border = initial\_border + difference; | iteratedFunction(x) –дійсного типу.  checkingFunction(x) – дійсного типу.  RecursedFunction(x) – дійсного типу. |

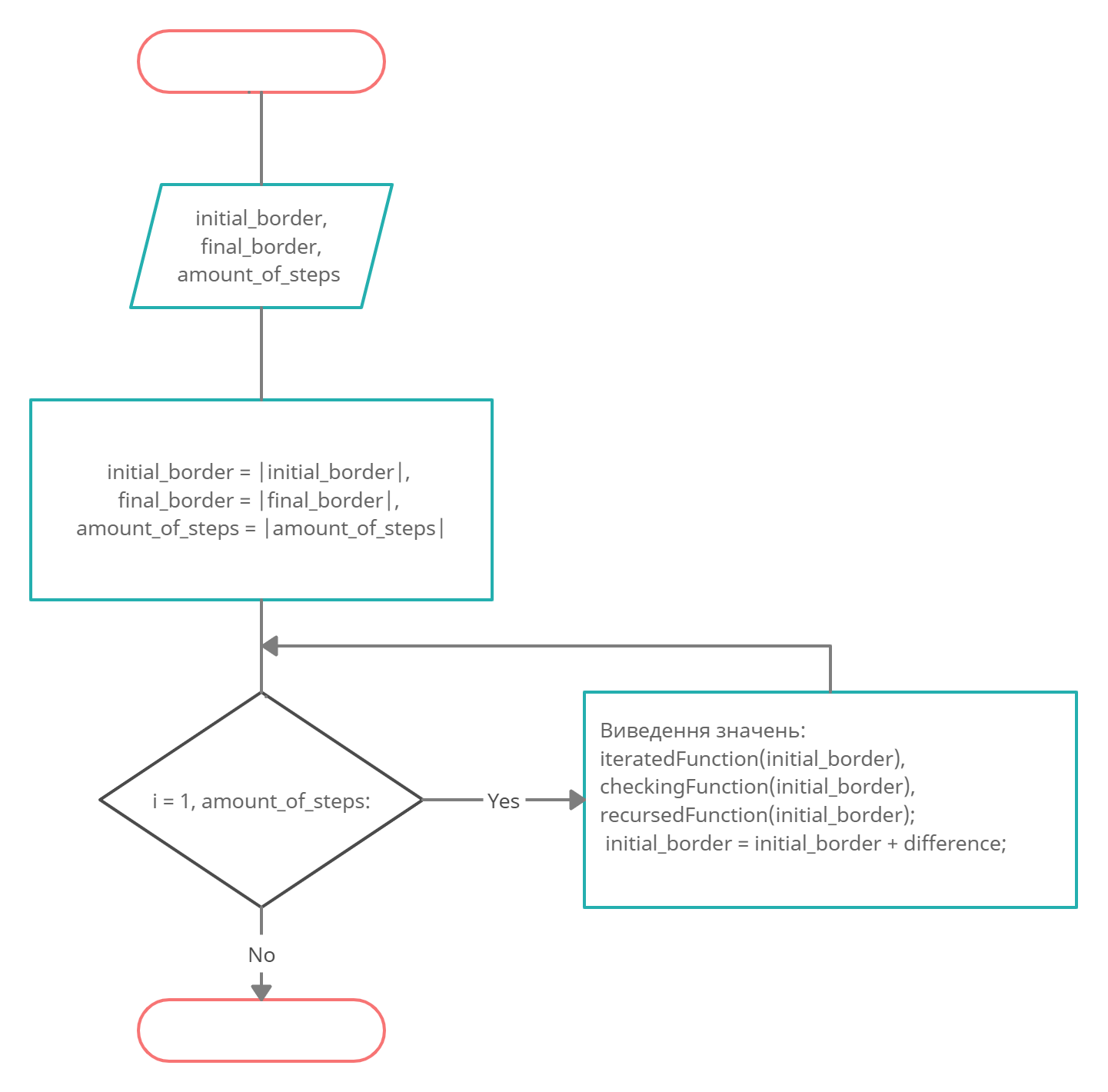


Рисунок 1.1 – Схема алгоритму основної програми.

* 1. **Тестування програми**
     1. *Цикл виконується декілька ітерацій:* initial\_border = 2, final\_border = 3,

amount\_of\_steps = 4;

1 ітерація. i=1; iteratedFunction(2) = 7.3906, checkingFunction(2) = 7.3906, recursedFunction(2) = 7.3906.

2 ітерація. i=2; iteratedFunction(2.25) = 9.48774 , checkingFunction(2.25) = 9.48774, recursedFunction(2.25) = 9.48774.

3 ітерація. i=3; iteratedFunction(2.5) = 12.1825, checkingFunction(2.5) =12.1825, recursedFunction(2.5) = 12.1825.

4 ітерація. i=4; iteratedFunction(2.75) = 15.6426, checkingFunction(2.75) = 15.6426­­, recursedFunction(2.75) = 15.6426.

5 ітерація. i=5; iteratedFunction(3) = 20.0855, checkingFunction(3) 20.0855, recursedFunction(3) = 20.0855.

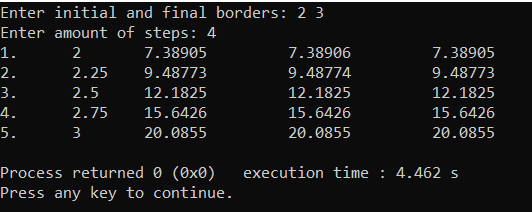


Рисунок 1.2 – Тестування програми.

* + 1. *Цикл виконується один раз:* initial\_border = 1, final\_border = 1,

amount\_of\_steps = 0;

1 ітерація. i=1; iteratedFunction(1) = 7.3906, checkingFunction(1) = 7.3906, recursedFunction(1) = 7.3906.

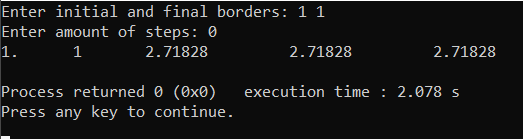


Рисунок 1.3 – Тестування програми.

* + 1. *Цикл не виконується – не існує.*

*Висновок:* результат роботи програми підтверджено тестовими розрахунками.

* 1. **Текст програмною мовою С++.**

#**include** <iostream>

#**include** <cmath>

#**include** "function.h"

using namespace **std**;

**double** checkingFunction(**double** x)

{

**return** exp(x);

}

**int** main(){

**int** amount\_of\_steps;

**double** initial\_border, final\_border;

**double** difference;

cout << "Enter initial and final borders: ";

cin >> initial\_border >> final\_border;

cout << "Enter amount of steps: ";

cin >> amount\_of\_steps;

difference = (final\_border - initial\_border) / amount\_of\_steps;

**for**(**int** i = 1; i <= amount\_of\_steps + 1; i++)

{

cout << i << "." << '\t' << initial\_border << '\t' << iteratedFunction(initial\_border) << '\t';

cout << '\t' << checkingFunction(initial\_border) << '\t' << '\t' << recursedFunction(initial\_border) << endl;

initial\_border += difference;

}

**return** 0;

}

**Рекурсивна функція факторіала**

**2.1 Математична постановка задачі (МПЗ).**

*Вхідні дані:* i – ціле число.

*Вихідні дані:* factorial(i) – ціле число.

*Математична постановка задачі:*

Якщо і = 0: повертати 1;

Інакше: повертати i \* factorial(i - 1);

Подання математичної постановки задачі у вигляді таблиці:

Таблиця 2.1

|  |  |  |
| --- | --- | --- |
| **Вхідні дані** | **Дії** | **Вихідні дані** |
| і – цілого типу. | Якщо і = 0: повертати 1;  Інакше: повертати i \* factorial(i - 1); | factorial(i) – цілого типу. |

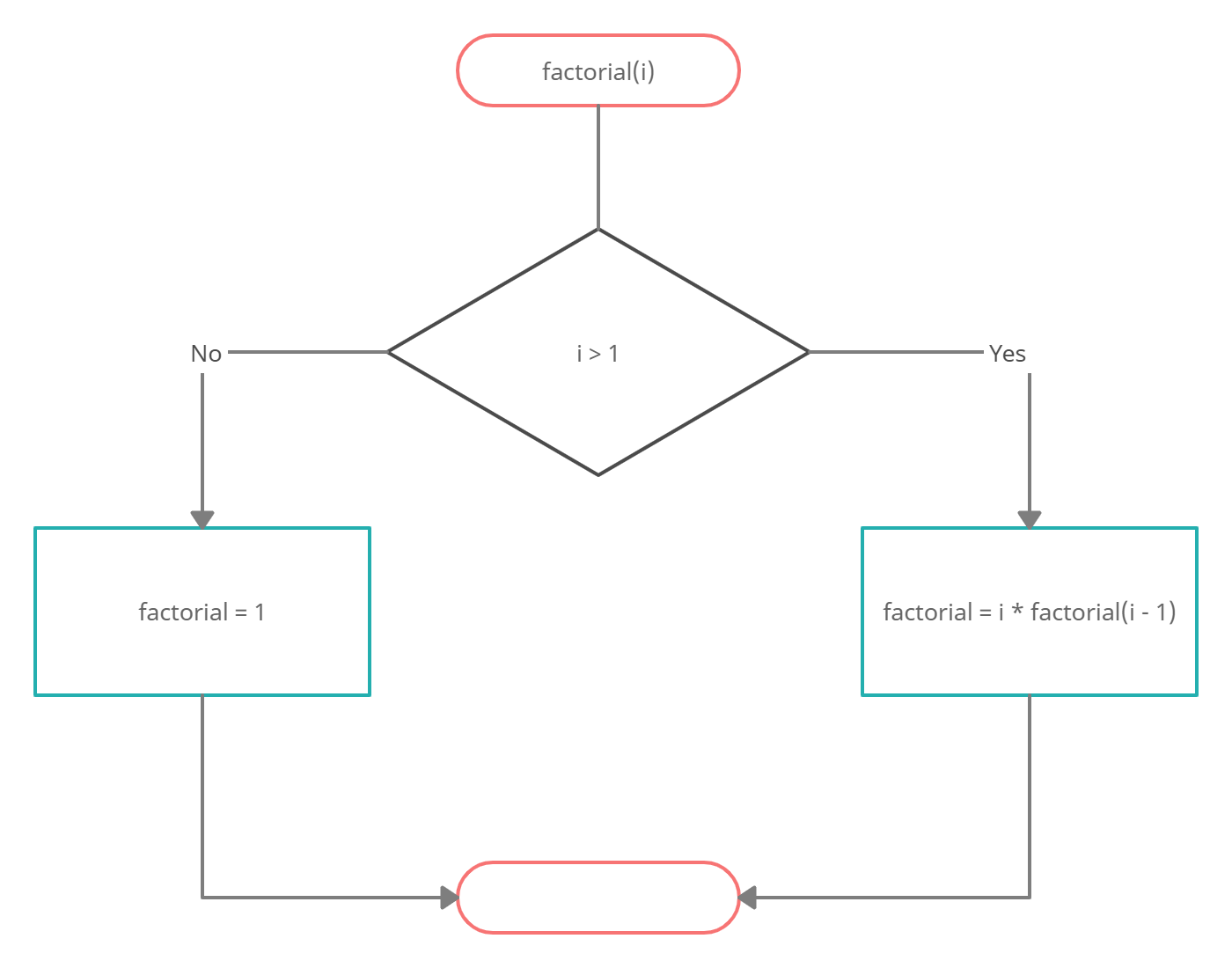


Рисунок 2.1 – Схема алгоритму роботи рекурсивної функції факторіала.

**2.2 Тестування програми.**

1. i = 3. factorial(i) = 1 \* 2 \* 3 = 6;

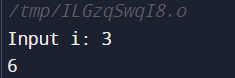


Рисунок 2.2 – Тестування програми.

1. i = 6. factorial(i) = 1 \* 2 \* 3 \* 4 \* 5 \* 6= 720;

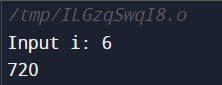


Рисунок 2.3 – Тестування програми.

1. i = 0. factorial(i) = 0 = 1;

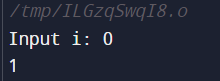


Рисунок 2.4 – Тестування програми.

*Висновок:* результат роботи програми підтверджено тестовими розрахунками.

**2.3 Текст програмною мовою С++.**

**float** factorial(**int** i)

{

**if** (i==0) **return** 1;

**else** **return** i\*factorial(i-1);

}

**Ітеративна функція обчислення експоненти**

**3.1 Математична постановка задачі**

*Вхідні дані:* x – число з плаваючою точкою.

*Вихідні дані:* sum – число з плаваючою точкою.

*Математична постановка задачі:*

x = |x|, sum = 0, step = 0;

Доки [xstep / factorial] >= min\_value:

Якщо step = 0: sum = sum + 1;

Інакше: sum = sum + [xstep / factorial(step)];

step = step + 1;

Подання математичної постановки задачі у вигляді таблиці:

Таблиця 3.1

|  |  |  |
| --- | --- | --- |
| **Вхідні дані** | **Дії** | **Вихідні дані** |
| x – число з плаваючою точкою. | x = |x|, sum = 0, step = 0;  Доки [xstep / factorial] >= min\_value:  Якщо step = 0: sum = sum + 1;  Інакше: sum = sum + [xstep / factorial(step)];  step = step + 1; | sum – число з плаваючою точкою. |

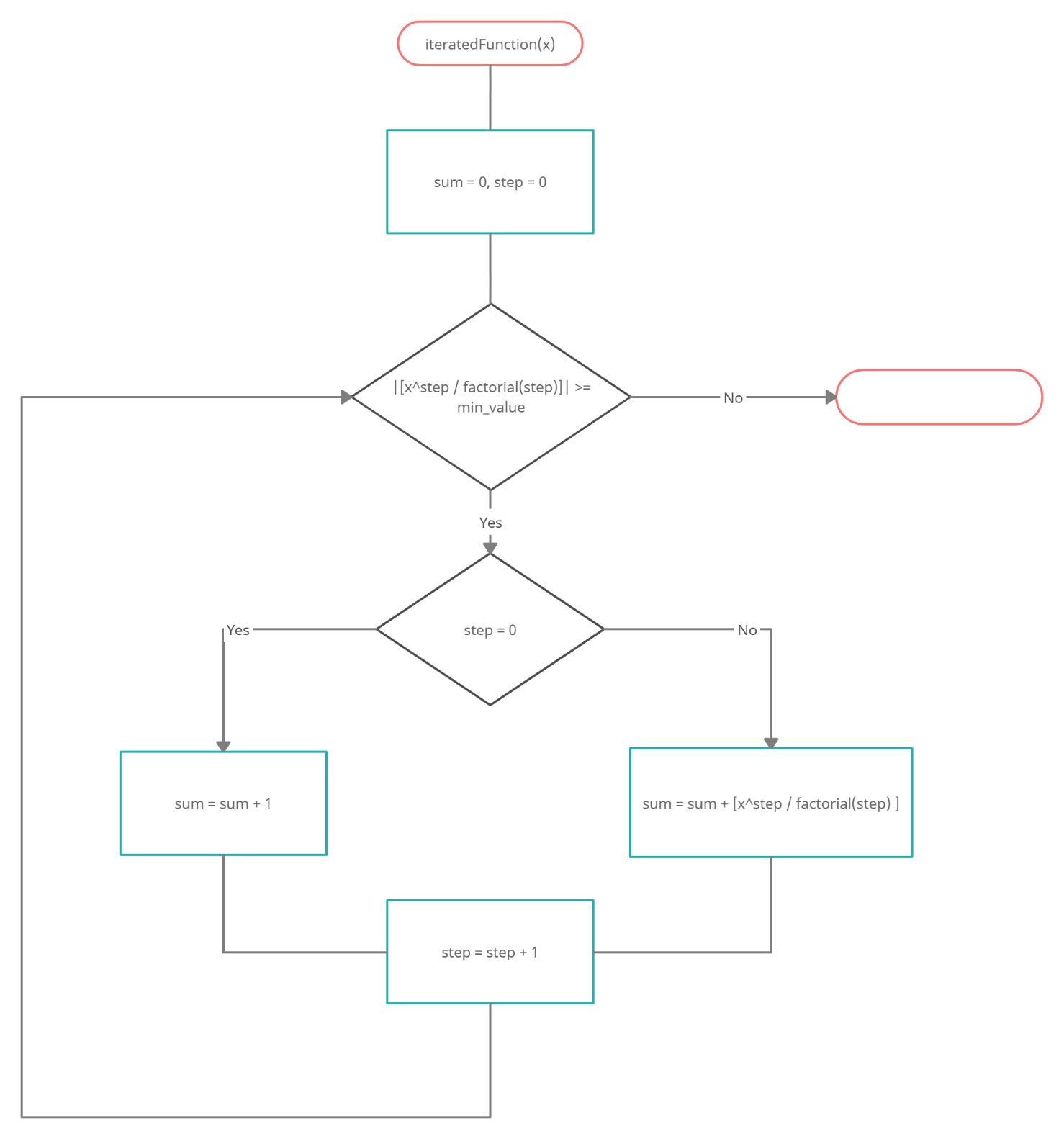


Рисунок 3.1 – Схема алгоритму роботи ітеративної функції по знаходженню експоненти.

* 1. **Тестування програми.**

1. *Цикл виконується декілька разів:* x = 3.

1 ітерація. sum = sum + [x0 / factorial(0)] = 0 + 1 = 1;

2 ітерація. sum = sum + [x1 / factorial(1)] = 1 + 3 = 4;

3 ітерація. sum = sum + [x2 / factorial(2)] = 4 + 4.5 = 8.5;

4 ітерація. sum = sum + [x3 / factorial(3)] = 8.5 + 4.5 = 13;

5 ітерація. sum = sum + [x4 / factorial(4)] = 13 + 3.375 = 16.375;

6 ітерація. sum = sum + [x5 / factorial(5)] = 16.375 + 2.025 = 18.4;

7 ітерація. sum = sum + [x6 / factorial(6)] = 18.4 + 1.0125 = 19.4125;

8 ітерація. sum = sum + [x7 / factorial(7)] = 19.4125 + 0.4339 = 19.8464;

9 ітерація. sum = sum + [x8 / factorial(8)] = 19.8464 + 0.1627 = 20.0091;

10 ітерація. sum = sum + [x9 / factorial(9)] = 20.0091 + 0.0542 = 20.0633

11 ітерація. sum = sum + [x10 / factorial(10)] = 20.0633 + 0.0162 = 20.0795;

12 ітерація. sum = sum + [x11 / factorial(11)] = 20.0795 + 0.0044 = 20.0839;

13 ітерація. sum = sum + [x12 / factorial(12)] = 20.0839 + 0.0011 = 20.085;

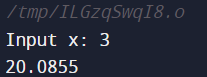


Рисунок 3.2 – Тестування програми.

1. *Цикл виконується 1 один раз*: х = 0

1 ітерація. sum = sum + [x0 / factorial(0)] = 0 + 1 = 1;

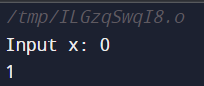
**

Рисунок 3.3 – Тестування програми.

1. *Цикл не виконується – не існує*.

*Висновок:* результат роботи програми підтверджено тестовими розрахунками.

* 1. **Текст програмною мовою С++.**

**double** iteratedFunction(**double** x)

{

**double** sum = 0;

**unsigned long** step = 0;

**while**(**fabs**(pow(x, step) / factorial(step)) >= min\_value)

{

**if**(step == 0) sum += 1;

**else** sum += (pow(x, step) / factorial(step));

step++;

}

**return** sum;

**Рекурсивна функція обчислення експоненти**

**4.1 Математична постановка задачі (МПЗ).**

*Вхідні дані:* x – число з плаваючою точкою, k – ціле число.

*Вихідні дані:* result – число з плаваючою точкою.

*Математична постановка задачі:*

x = |x|, Якщо k не надане значення: k = 1, інакше k = |k|,

result = [xk / factorial(k)];

Якщо result < min\_value:

result = 1;

Інакше:

result = result + recursedFunction(x, k + 1);

Подання математичної постановки здачі у вигляді таблиці:

Таблиця 4.1

|  |  |  |
| --- | --- | --- |
| **Вхідні дані** | **Дії** | **Вхідні дані** |
| x – число з плаваючою точкою  k – цілого типу. | x = |x|,  Якщо k не надане значення: k = 1,  інакше k = |k|,  result = [xk / factorial(k)];  Якщо result < min\_value:  result = 1;  Інакше:  result = result + recursedFunction(x, k + 1); | result – число з плаваючою точкою. |

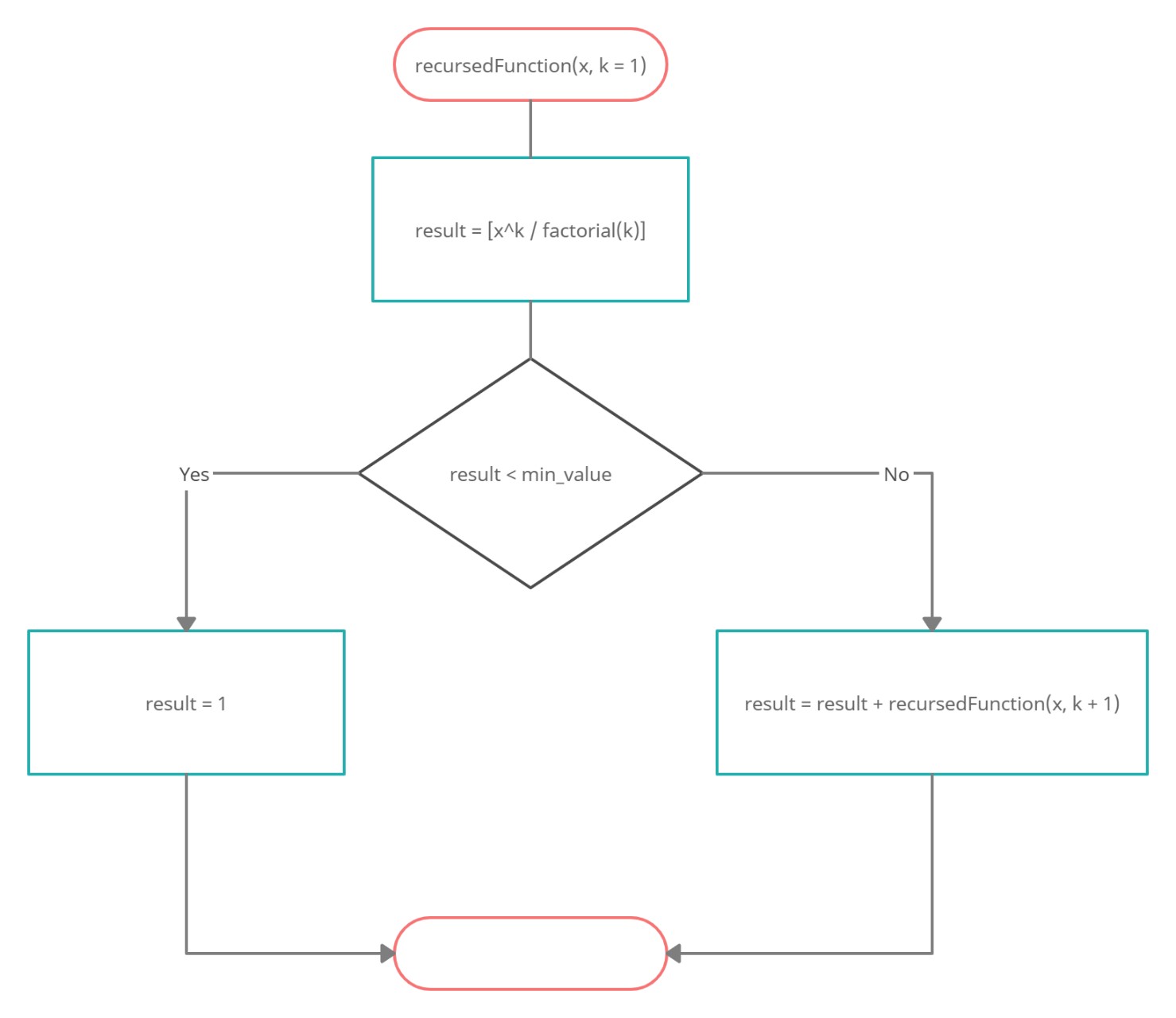


Рисунок 4.1– Схема алгоритму роботи рекурсивної функції по знаходженню експоненти.

**4.2 Тестування програми.**

1. x = 1; result = 1 + 1 + 0.5 + 0.16 + 0.0416 + 0.0083 = 2.718

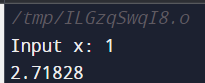


Рисунок 4.2 – Тестування програми.

1. x = 2; result = 1 + 2 + 2 + 1.3 + 0.6 + 0.26 + 0.08 + 0.0253 + 0.0063 + 0.0014 = 7.389

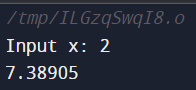


Рисунок 4.3 – Тестування програми.

1. x = 3; result = 1 + 3 + 4.5 + 4.5 + 3.375 + 2.025 + 1.0125 + 0.4339 + 0.1627 + + 0.0542 + 0.0162 + 0.0044 + 0.0011 = 20.085.

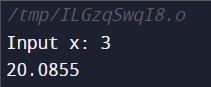


Рисунок 4.4 – Тестування програми.

**4.3 Текст програмною мовою С++.**

**double** recursedFunction(**double** x, **int** k = 1)

{

**double** result = (pow(x, k) / factorial(k));

**if**(result < min\_value)

{

result = 1;

}

else

{

result += recursedFunction(x, k + 1);

}

**return** result;

}