**Міністерство освіти і науки України  
Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського»  
Факультет інформатики та обчислювальної техніки  
Кафедра обчислювальної техніки**

**Лабораторна робота №1**

з дисципліни  
«Алгоритми і структури даних»

Виконав: Перевірив:

студент групи ІМ-42 Сергієнко А. М.  
Федоренко Іван Русланович  
номер у списку групи: 29

Київ 2025

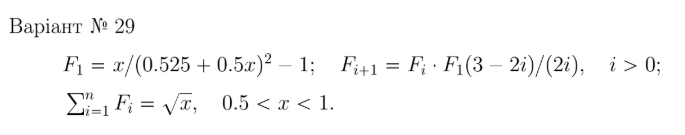
**Завдання**

Дане натуральне число n. Знайти суму перших n членiв ряду чисел, заданого рекурентною формулою. Розв’язати задачу трьома способами:

1) у програмі використати рекурсивну функцію, яка виконує обчислення i членiв ряду, i суми на рекурсивному спуску;

2) у програмі використати рекурсивну функцію, яка виконує обчислення i членiв ряду, i суми на рекурсивному поверненні;

3) у програмі використати рекурсивну функцію, яка виконує обчислення членiв ряду на рекурсивному спуску, а обчислення суми на рекурсивному поверненні



На жаль, формула не працює коректно для апроксимації функції , тому для обчислень я використав ряд Тейлора:

**Обчислення**  за допомогою **біноміального ряду** навколо точки x=1.

**Код програми:**

#include <stdio.h>

#include <math.h>

double sqrt\_series\_descent(double x, int n, int i, double term, double sum) {

sum += term;

printf("f\_%d = %.10lf, sum = %.10lf\n", i, term, sum);

if (i == n - 1) {

return sum;

}

double next\_term = term \* ((3.0 - 2.0 \* (i + 1)) / (2.0 \* (i + 1))) \* (x - 1);

return sqrt\_series\_descent(x, n, i + 1, next\_term, sum);

}

typedef struct {

double term;

double sum;

} Result;

Result sqrt\_series\_return(double x, int n, int i) {

if (i == 0) {

printf("f\_0 = %.10lf\n", 1.0);

return (Result){1.0, 1.0};

}

Result prev = sqrt\_series\_return(x, n, i - 1);

double t = x - 1;

double term = prev.term \* ((3.0 - 2.0 \* i) / (2.0 \* i)) \* t;

double sum = prev.sum + term;

printf("f\_%d = %.10lf, sum = %.10lf\n", i, term, sum);

return (Result){term, sum};

}

double sqrt\_series\_hybrid(double x, int n, int i, double prev\_term) {

double term = (i == 0) ? 1.0 :

prev\_term \* ((3.0 - 2.0 \* i) / (2.0 \* i)) \* (x - 1);

if (i == n - 1) {

printf("f\_%d = %.10lf, sum = %.10lf\n", i, term, term);

return term;

}

double rest\_sum = sqrt\_series\_hybrid(x, n, i + 1, term);

double total = term + rest\_sum;

printf("f\_%d = %.10lf, sum = %.10lf\n", i, term, total);

return total;

}

double sqrt\_series\_loop(int n, double x) {

double sum = 1.0; // f\_0 = 1

double term = 1.0;

for (int i = 1; i < n; i++) {

term \*= ((3.0 - 2.0 \* i) / (2.0 \* i)) \* (x - 1);

sum += term;

printf("f\_%d = %.10lf, sum = %.10lf\n", i, term, sum);

}

return sum;

}

int main() {

int n = 5;

double x = 0.79;

if(x > 0.5 && x < 1) {

printf("\nN = %d, X = %.3lf.", n, x);

printf("\n--- [1] Method1 ---\n");

double res1 = sqrt\_series\_descent(x, n, 0, 1.0, 0.0);

printf("Final sum = %.10lf\n", res1);

printf("\n--- [2] Method2 ---\n");

Result res2 = sqrt\_series\_return(x, n, n - 1);

printf("Final sum = %.10lf\n", res2.sum);

printf("\n--- [3] Method3 ---\n");

double res3 = sqrt\_series\_hybrid(x, n, 0, 0.0);

printf("Final sum = %.10lf\n", res3);

printf("\n--- [4] Method4 ---\n");

double res4 = sqrt\_series\_loop(n, x);

printf("Final sum = %.10lf\n", res4);

double actual = sqrt(x);

printf("\nActual sqrt(%.2lf) = %.10lf\n", x, actual);

printf("\nError: %.10lf", fabs(res1 - actual));

} else {

printf("\nmIvalid data: %d, %.3lf", n, x);

}

return 0;

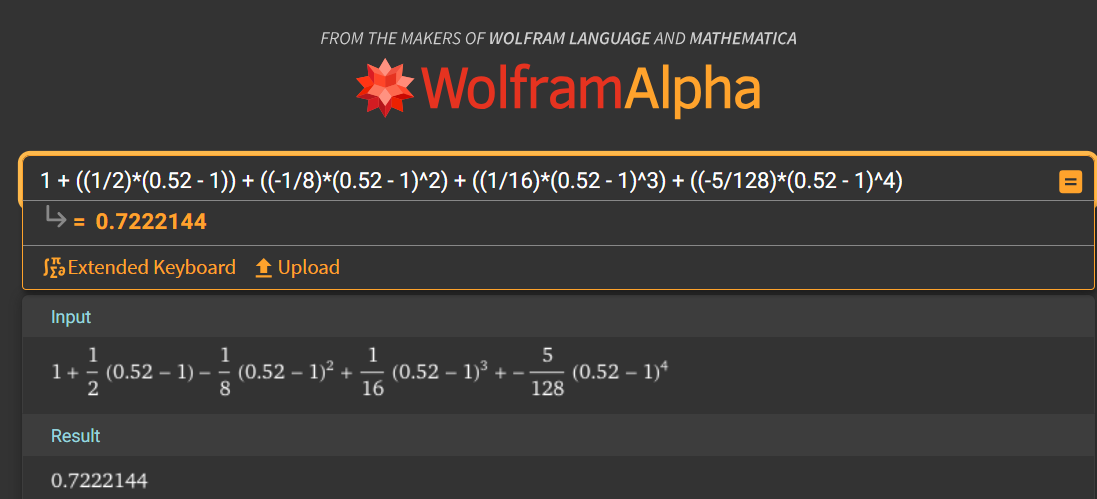
}

**Результати тестування:**

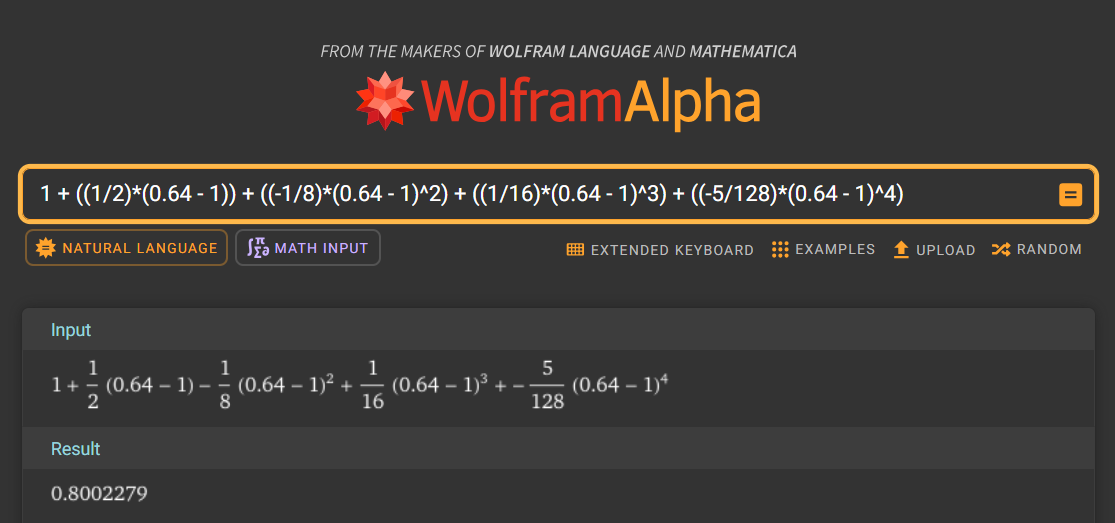
|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

**Перевірка результатів обчислень через WolframAlpha:**

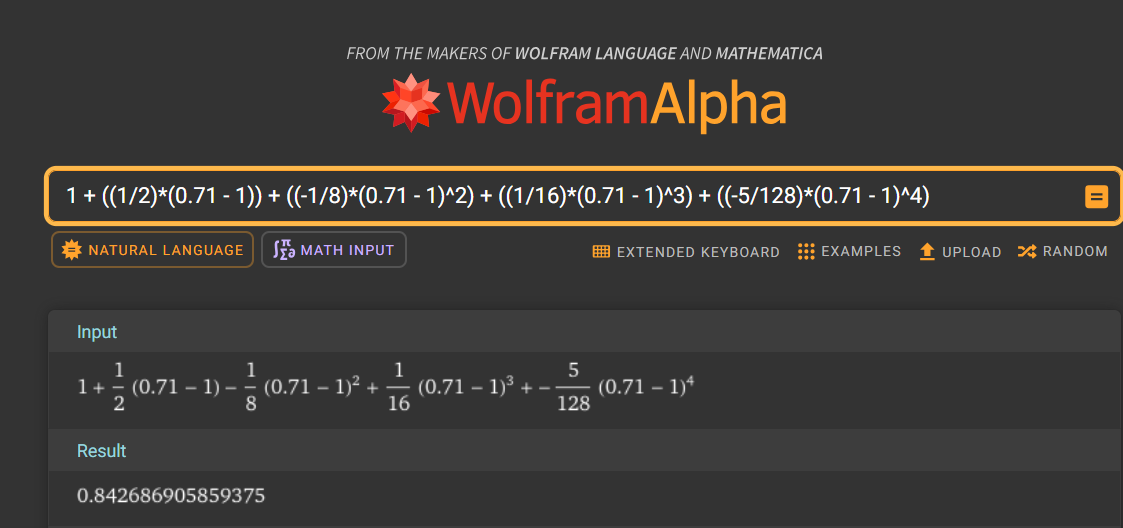
**X=0.52**

****

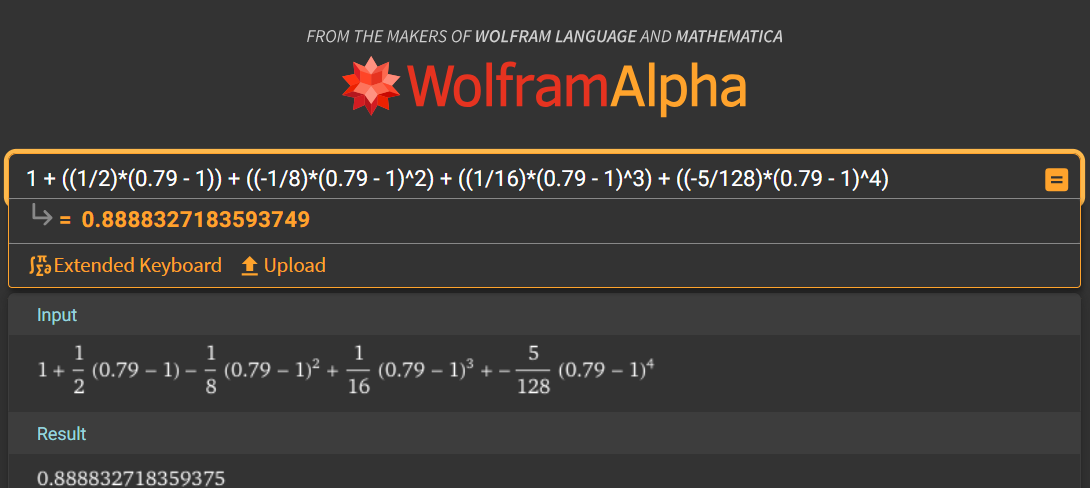
**X = 0.64**

****

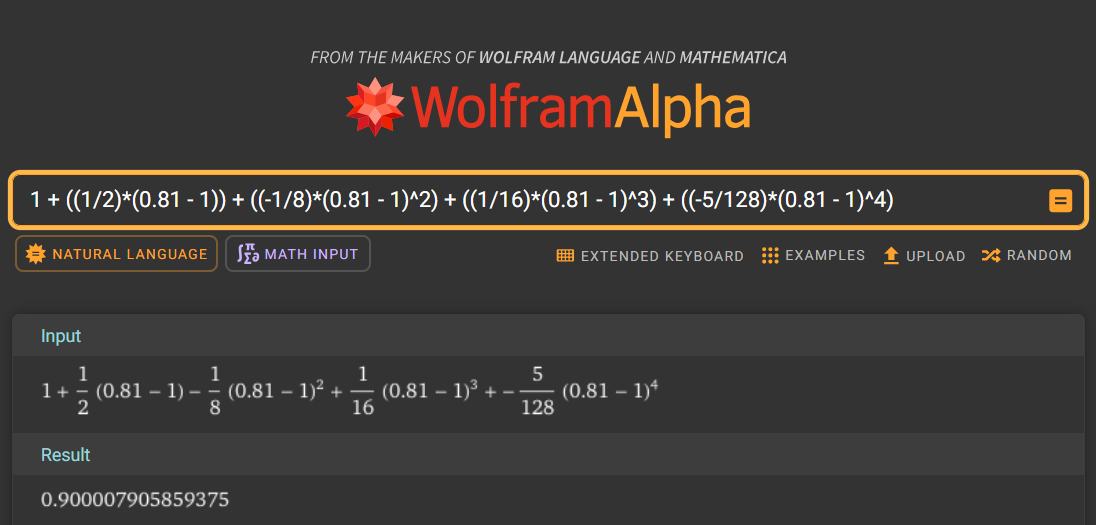
**Х=0.71**

****

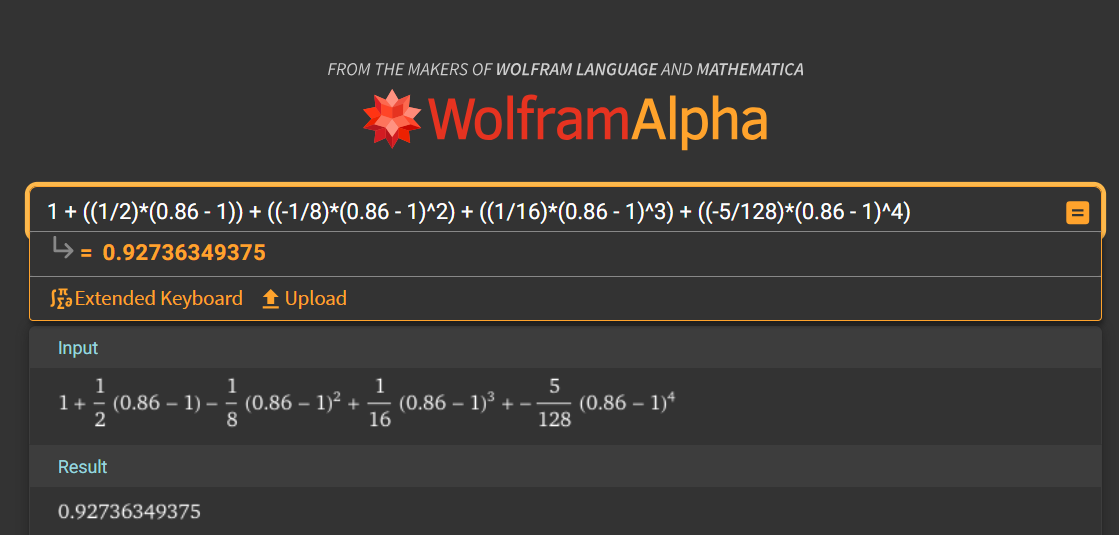
**X = 0.79**

****

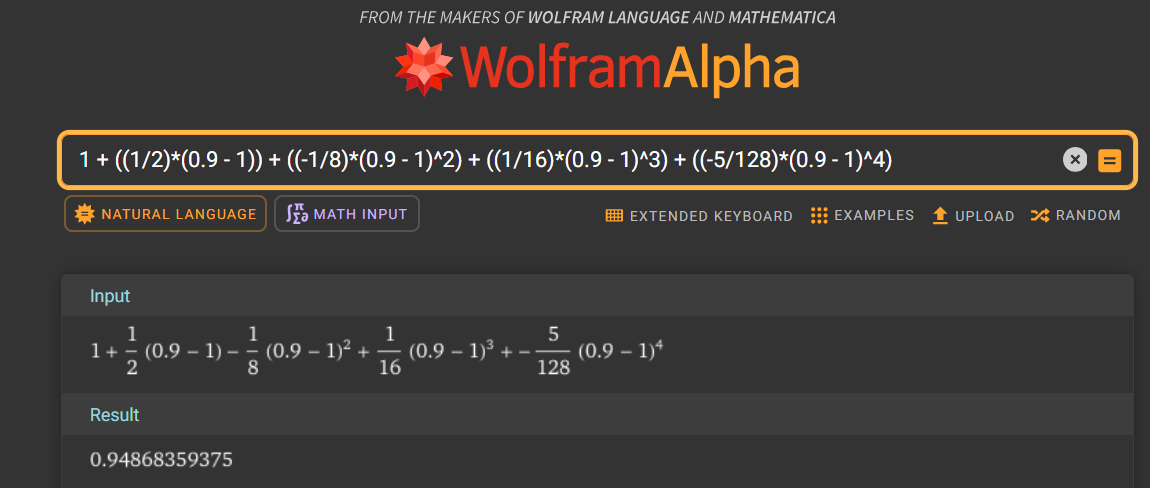
**X=0.81**

****

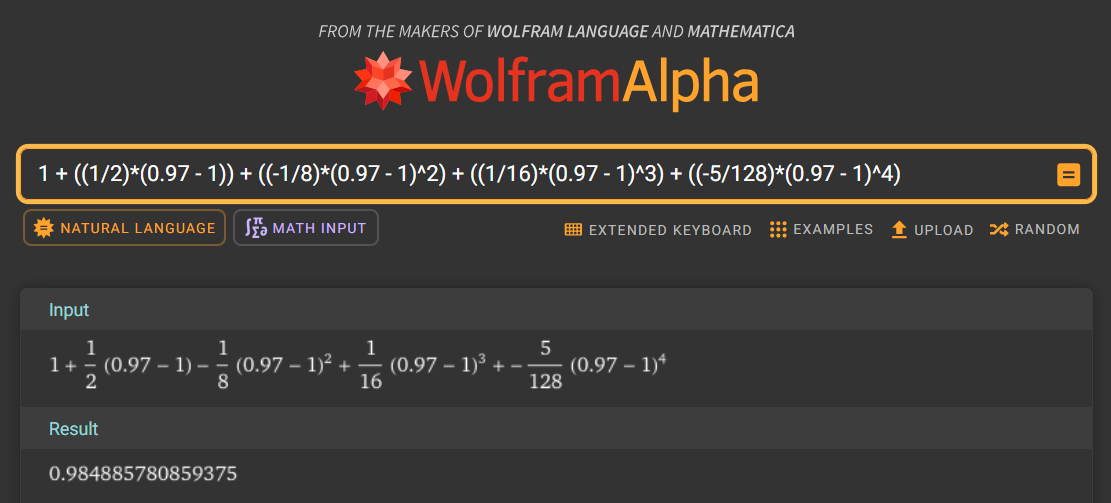
**Х=0.86**

****

**X=0.90**

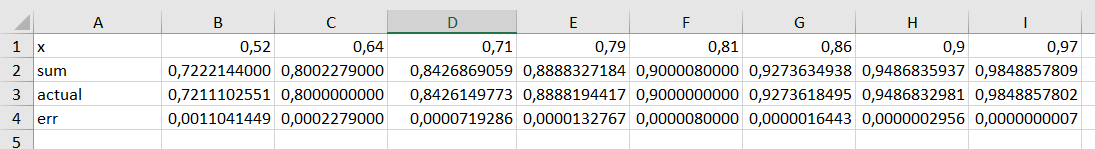
****

**X=0.97**

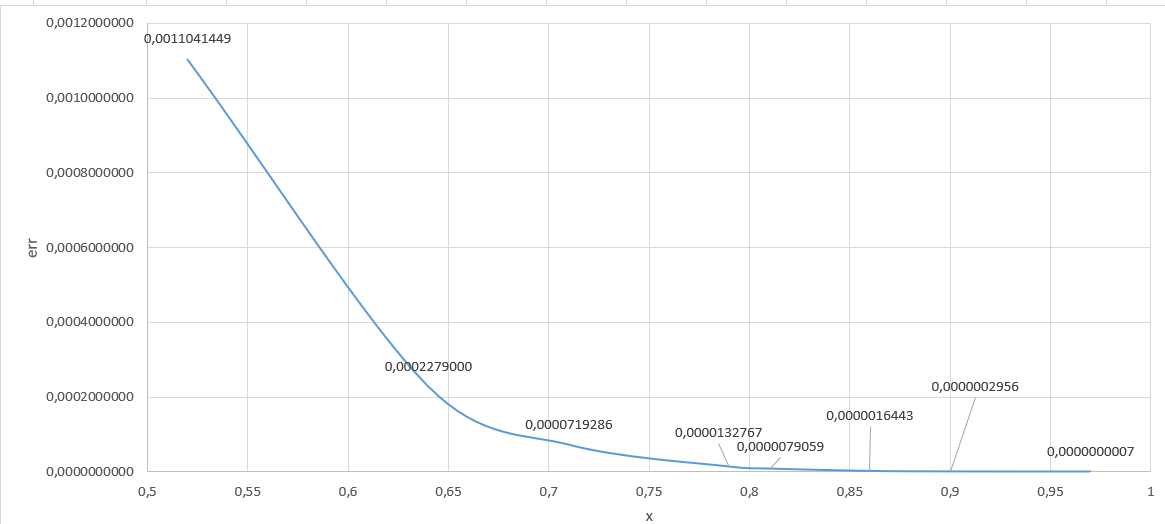
****

**Графік похибки:**

**Табличка зі значеннями:**

****

**Графік:**

****

**Висновки:**

У ході виконання лабораторної роботи я реалізував три рекурсивні функції, які обчислюють наближення квадратного кореня з числа 𝑥 за допомогою ряду, розкладеного за узагальненою біноміальною формулою.

Для тестування використав циклічний (ітераційний) варіант обчислення функції, який повторює логіку обчислення ряду без використання рекурсії.