ЗВІТ

про виконання лабораторної роботи № 5.4

« Обчислення сум та добутків за допомогою рекурсії »

з дисципліни

«Алгоритмізація та програмування»

студента групи IT-12

М’ягких Владислава Юрійовича

**Варіант 11**

**Умова завдання**

За допомогою рекурсивних функцій обчислити значення суми чи добутку.

В одній програмі слід вивести результати звертання до п’яти функцій, які реалізують наступні способи:

1) обчислення здійснюється ітераційним способом (для контролю правильності рекурсивних способів).

2) обчислення здійснюються на рекурсивному спуску, значення параметра в рекурсивних викликах – спадає;

3) обчислення здійснюються на рекурсивному спуску, значення параметра в рекурсивних викликах – зростає;

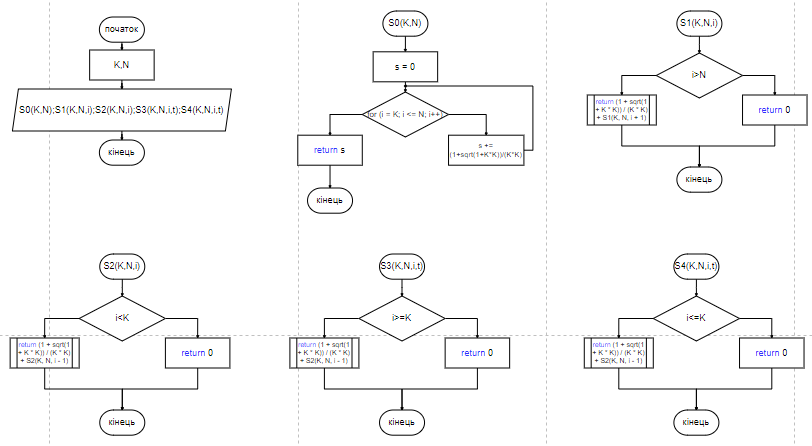
4) обчислення здійснюються на рекурсивному підйомі, значення параметра в рекурсивних викликах – спадає;

5) обчислення здійснюються на рекурсивному підйомі, значення параметра в рекурсивних викликах – зростає;

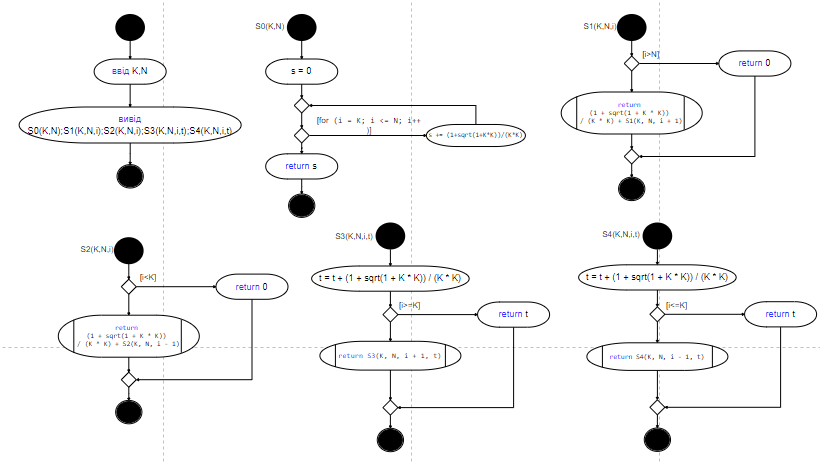
Всі п’ять результатів мають збігатися. Всю необхідну функціям інформацію слід передавати лише за допомогою параметрів.



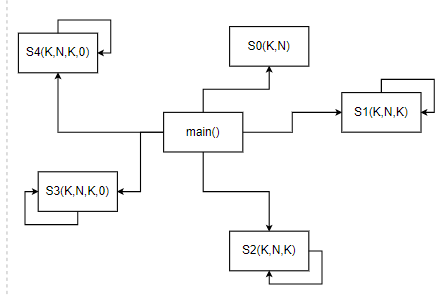
**Блок-схеми алгоритму програми та кожної функції:**



**UML-діаграма дії програми та кожної функції:**



**Структурна схема програми:**



**Текст програми:**

// Лабораторна робота № 5.4

#include <iostream>

#include <cmath>

using namespace std;

double S0(const int K, const int N);

double S1(const int K, const int N, const int i);

double S2(const int K, const int N, const int i);

double S3(const int K, const int N, const int i, double t);

double S4(const int K, const int N, const int i, double t);

int main()

{

int K, N;

cout << "K = "; cin >> K;

cout << "N = "; cin >> N;

cout << "(iter) S0 = " << S0(K, N) << endl;

cout << "(rec up ++) S1 = " << S1(K, N, K) << endl;

cout << "(rec up --) S2 = " << S2(K, N, N) << endl;

cout << "(rec down ++) S3 = " << S3(K, N, K, 0) << endl;

cout << "(rec down --) S4 = " << S4(K, N, N, 0) << endl;

return 0;

}

double S0(const int K, const int N)

{

double s = 0;

for (int i = K; i <= N; i++)

s += (1+sqrt(1+K\*K))/(K\*K);

return s;

}

double S1(const int K, const int N, const int i)

{

if (i > N)

return 0;

else

return (1 + sqrt(1 + K \* K)) / (K \* K) + S1(K, N, i + 1);

}

double S2(const int K, const int N, const int i)

{

if (i < K)

return 0;

else

return (1 + sqrt(1 + K \* K)) / (K \* K) + S2(K, N, i - 1);

}

double S3(const int K, const int N, const int i, double t)

{

t = t + (1 + sqrt(1 + K \* K)) / (K \* K);

if (i >= N)

return t;

else

return S3(K, N, i + 1, t);

}

double S4(const int K, const int N, const int i, double t)

{

t = t + (1 + sqrt(1 + K \* K)) / (K \* K);

if (i <= K)

return t;

else

return S4(K, N, i - 1, t);

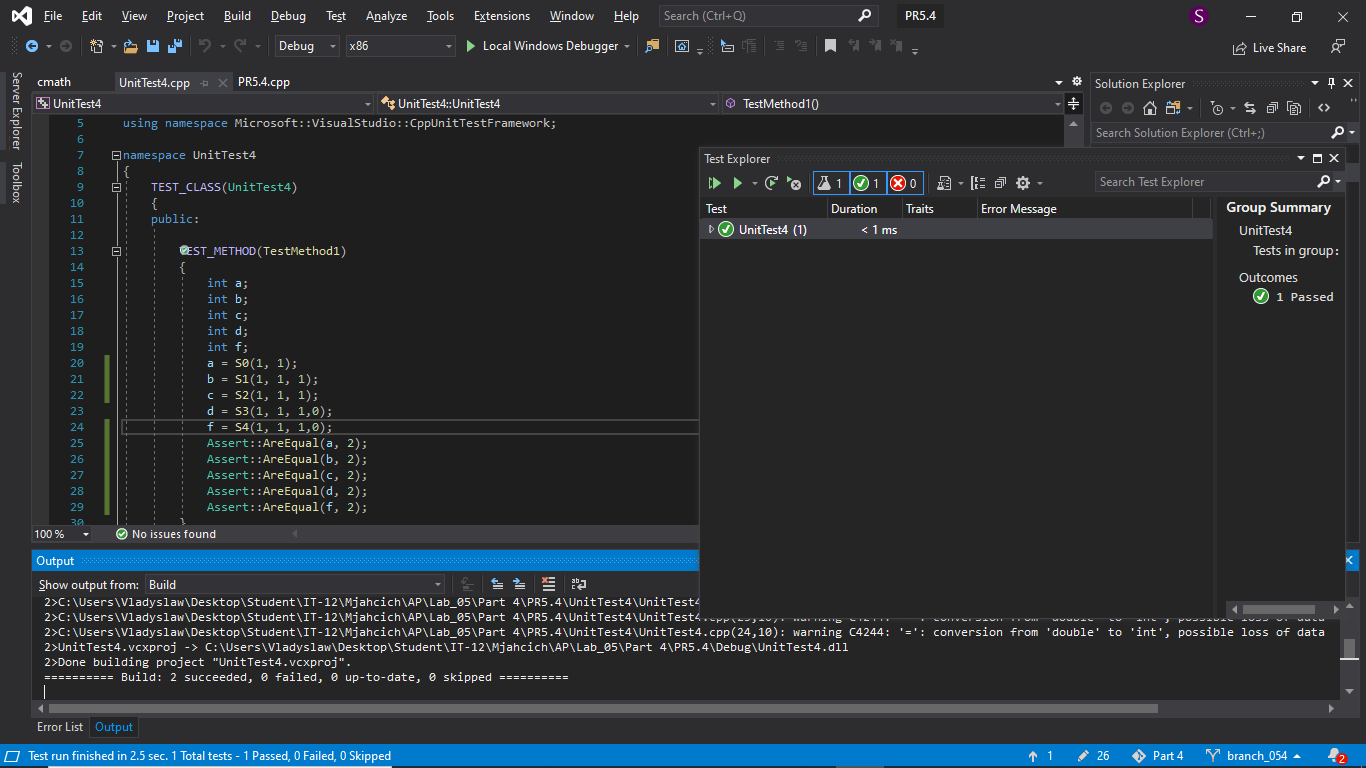
}

**Посилання на git-репрезиторій з проектом**

[**https://github.com/ElitLien/-lab05\_repos.git**](https://github.com/ElitLien/-lab05_repos.git)

**(branch\_054)**

**Результати unit-тесту:**



**Висновки**

На даній лабораторній роботі я більше навчився користуватися рекурсивними функціями