

**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ  
 НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
імені ІГОРЯ СІКОРСЬКОГО»**

**Навчально-науковий інститут прикладного системного аналізу  
Кафедра системного проектування**

**Звіт**

**про виконання практичної роботи №1  
з дисципліни «Паралельні обчислення»**

**На тему: Дослідження базових операцій з потоками виконання**

Виконав:  
студент III курсу, групи ДА-22  
Байдюк Антон Олексійович

Прийняв:

асистент Яременко В. С.

Київ – 2025

**ПОСТАНОВКА ЗАДАЧІ**

1. Визначити основні характеристики ПК, котрі на думку студента впливають на ефективність виконання паралельних обчислень. Зафіксувати значення даних характеристик для ПК студента, та занести їх до протоколу роботи.
2. Створити або використати наявній механізм для заміру часу виконання програми, або інших параметрів, котрі студент вважає релевантними. Занести опис механізму до роботи.
3. Вирішити обрану за варіантом задачу, не використовуючи паралелізацію. Заміряти час вирішення задачі, або інші параметри, котрі студент вважає релевантними.
4. Вирішити оборану за варіантом задачу, використовуючи паралелізацію. Заміряти час вирішення задачі, або інші параметри, котрі студент вважає релевантними. Обґрунтувати вибір алгоритму паралелізації (розподілення даних між потоками), надати опис та обґрунтування в протоколі роботи.
5. Перевірити алгоритм на фіксованих кількостях потоків: 2-рази меншій, ніж кількість фізичних ядер, на кількості рівній фізичним ядрам, на кількості рівній логічних ядрам, на кількості більшій в 2, 4, 8, 16 разів ніж кількість логічних ядер.
6. Повторити пункт 5 з використанням різної розмірності даних.
7. Зробити таблиці та графіки залежності часу виконання завдання від кількості потоків для різної розмірності даних. Надати опис графіків, з причинами виникнення отриманих результатів в протоколі роботи.
8. Надати висновок, що повинен містити аналіз отриманих результатів.

Варіант 1:

Створити вектор з n елементами випадкових чисел. Знайти норму вектору.

**ХІД РОБОТИ**

1. Скористаємося програмою Cupid для візуалізації характеристик системи:

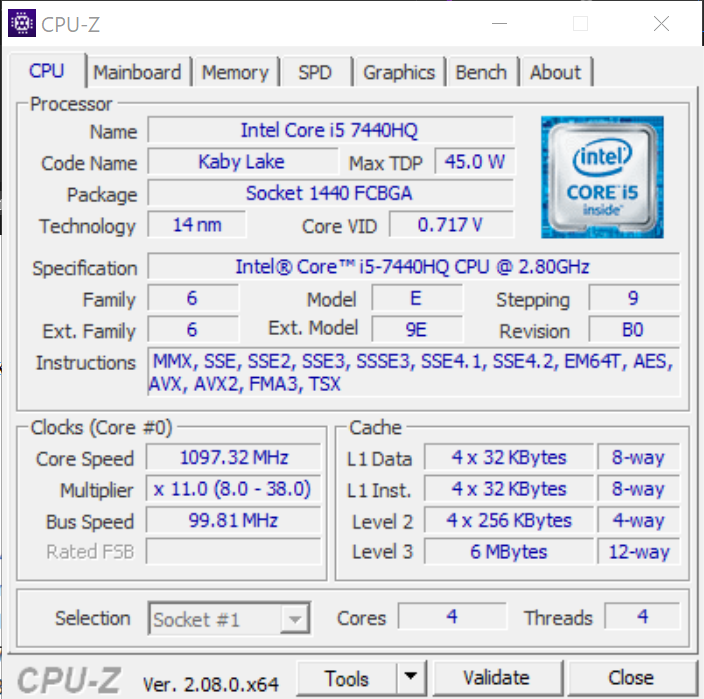


Рис. 1 - Властивості процесора

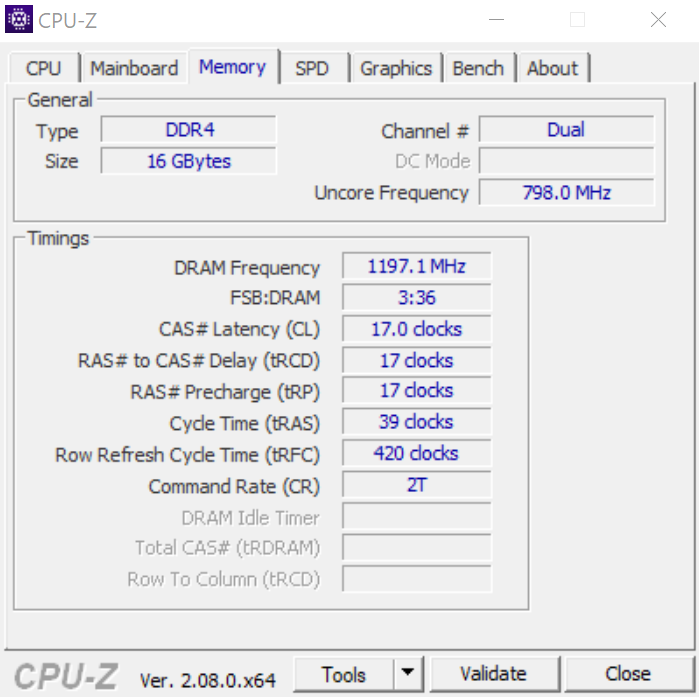


Рис. 2 - Властивості оперативної пам’яті

1. Код програми

#include <iostream>

#include <iomanip>

#include <cstdlib>

#include <ctime>

#include <cmath>

#include <chrono>

#include <thread>

using namespace std;

using namespace std::chrono;

void worker(int\* arr, int size, int start, int step, double& local\_sum) {

local\_sum = 0;

for (int i = start; i < size; i += step) {

local\_sum += arr[i] \* arr[i];

}

}

double norm\_dynamic\_threads(int\* arr, int size, int num\_threads, double& sum, long long& duration) {

double\* local\_sum = new double[num\_threads] {0};

thread\* workers = new thread[num\_threads];

auto start = high\_resolution\_clock::now();

for (int i = 0; i < num\_threads; ++i) {

workers[i] = thread(worker, arr, size, i, num\_threads, ref(local\_sum[i]));

}

for (int i = 0; i < num\_threads; ++i) {

workers[i].join();

}

sum = 0;

for (int i = 0; i < num\_threads; ++i) {

sum += local\_sum[i];

}

double norm = sqrt(sum);

auto end = high\_resolution\_clock::now();

duration = duration\_cast<nanoseconds>(end - start).count();

delete[] local\_sum;

delete[] workers;

return norm;

}

double norm\_slow(int\* arr, int size, double& sum, long long& duration) {

auto start = high\_resolution\_clock::now();

sum = 0;

for (int i = 0; i < size; i++) {

sum += arr[i] \* arr[i];

}

double norm = sqrt(sum);

auto end = high\_resolution\_clock::now();

duration = duration\_cast<nanoseconds>(end - start).count();

return norm;

}

void print\_results(int\* arr, int size, double (\*norm\_func)(int\*, int, int, double&, long long&), int num\_threads, const string& label) {

double sum = 0;

long long duration = 0;

double norm;

if (num\_threads == 0) {

norm = norm\_slow(arr, size, sum, duration);

}

else {

norm = norm\_func(arr, size, num\_threads, sum, duration);

}

cout << left << setw(10) << size

<< setw(15) << sum

<< setw(20) << (size \* 2)

<< setw(15) << norm

<< setw(20) << duration

<< setw(25) << label

<< endl;

}

int main() {

srand(static\_cast<unsigned int>(time(0)));

const int sizes[] = { 10, 100, 200, 500, 1000, 5000, 10000, 50000, 100000, 500000, 1000000, 5000000, 10000000, 50000000, 100000000 };

const int numSizes = sizeof(sizes) / sizeof(sizes[0]);

int\* arrays[numSizes];

for (int i = 0; i < numSizes; i++) {

arrays[i] = new int[sizes[i]];

for (int j = 0; j < sizes[i]; j++) {

arrays[i][j] = rand() % 1000 + 1;

}

}

cout << left

<< setw(10) << "Size"

<< setw(15) << "Sum"

<< setw(20) << "Total Operations"

<< setw(15) << "Norm"

<< setw(20) << "Execution Time (ns)"

<< setw(20) << "Calculation Method" << endl;

cout << string(100, '-') << endl;

for (int i = 0; i < numSizes; i++) {

print\_results(arrays[i], sizes[i], norm\_dynamic\_threads, 0, "Default");

print\_results(arrays[i], sizes[i], norm\_dynamic\_threads, 1, "1 Thread");

print\_results(arrays[i], sizes[i], norm\_dynamic\_threads, 2, "2 Threads");

print\_results(arrays[i], sizes[i], norm\_dynamic\_threads, 5, "5 Threads");

print\_results(arrays[i], sizes[i], norm\_dynamic\_threads, 10, "10 Threads");

print\_results(arrays[i], sizes[i], norm\_dynamic\_threads, 25, "25 Threads");

print\_results(arrays[i], sizes[i], norm\_dynamic\_threads, 50, "50 Threads");

print\_results(arrays[i], sizes[i], norm\_dynamic\_threads, 100, "100 Threads");

cout << string(100, '-') << endl;

}

for (int i = 0; i < numSizes; i++) {

delete[] arrays[i];

}

return 0;

}

1. Результат виконання

Зпиишемо у таблиці час виконання обчислень у наносекундах:

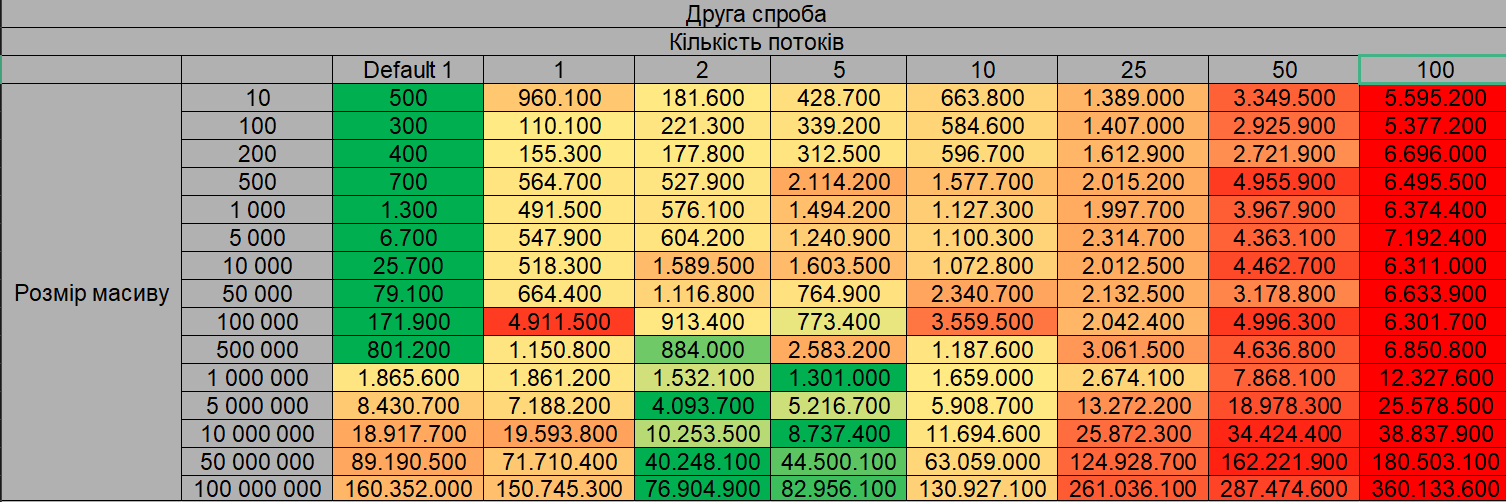
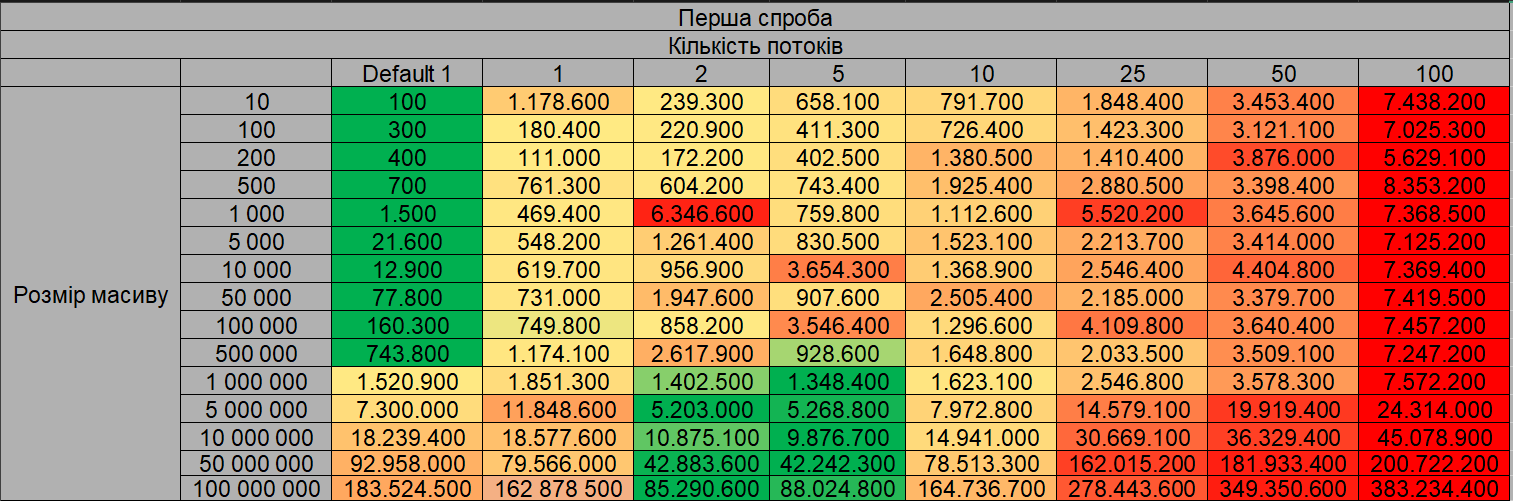


Рис. 3 та 4 - Результат виконання програми

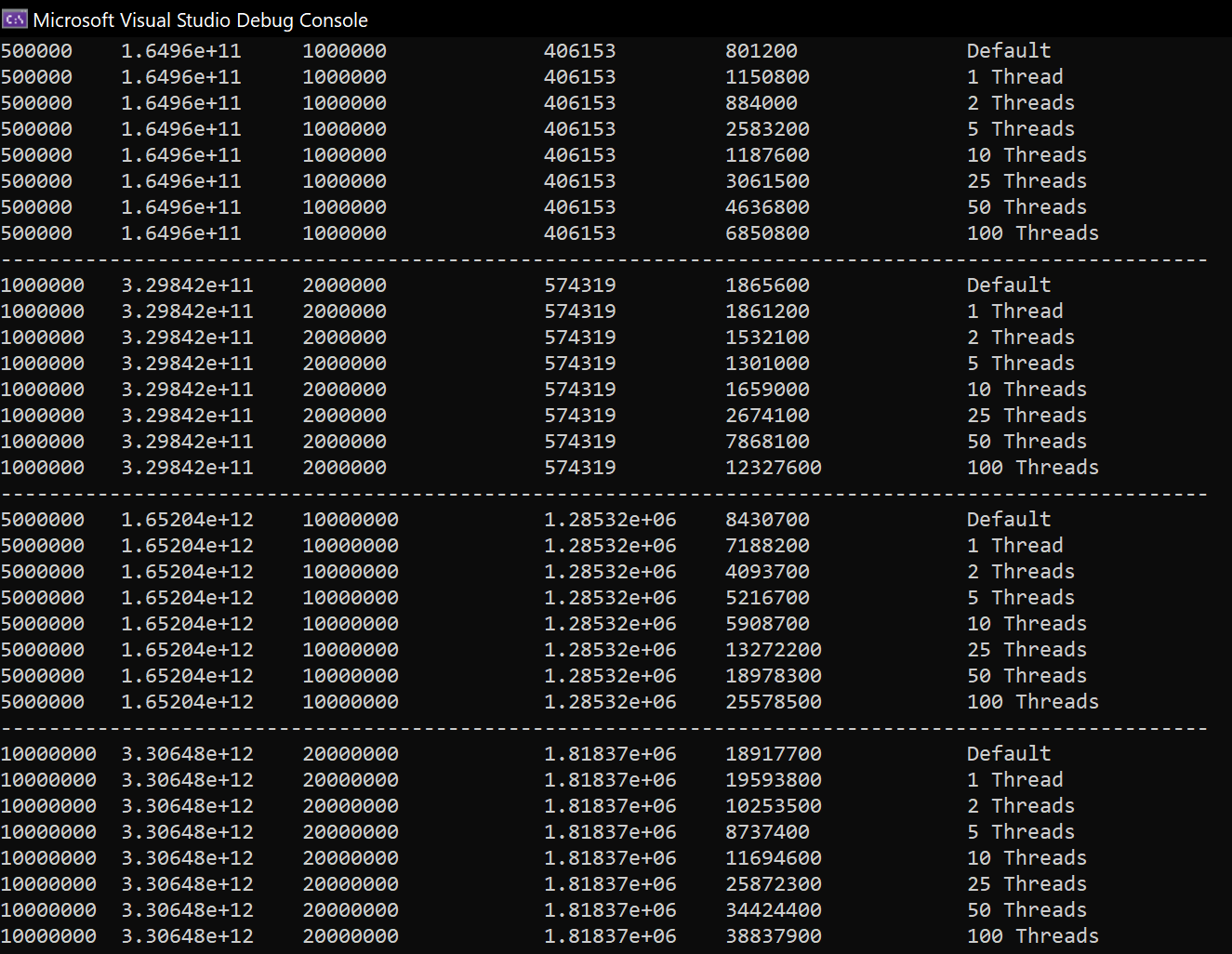
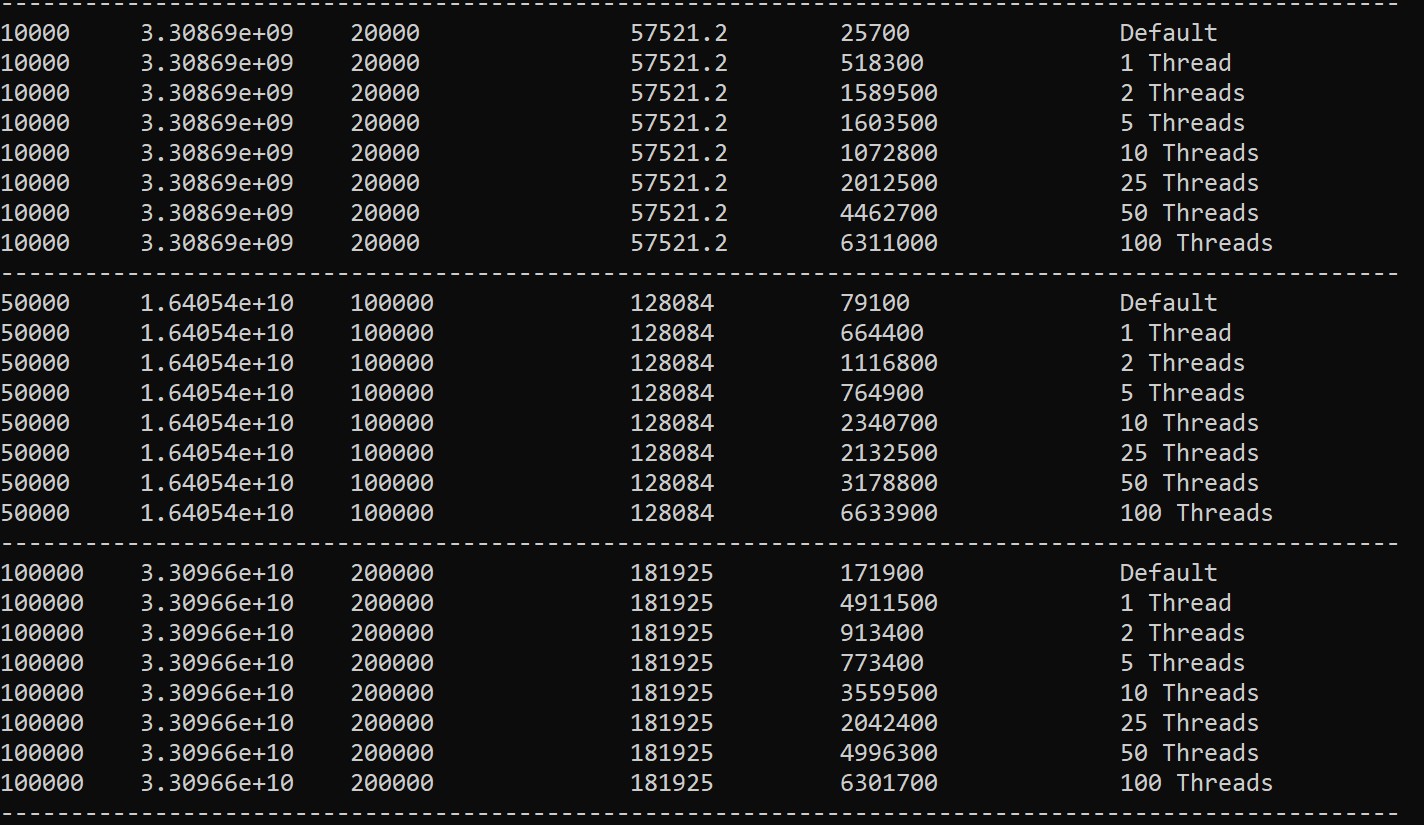
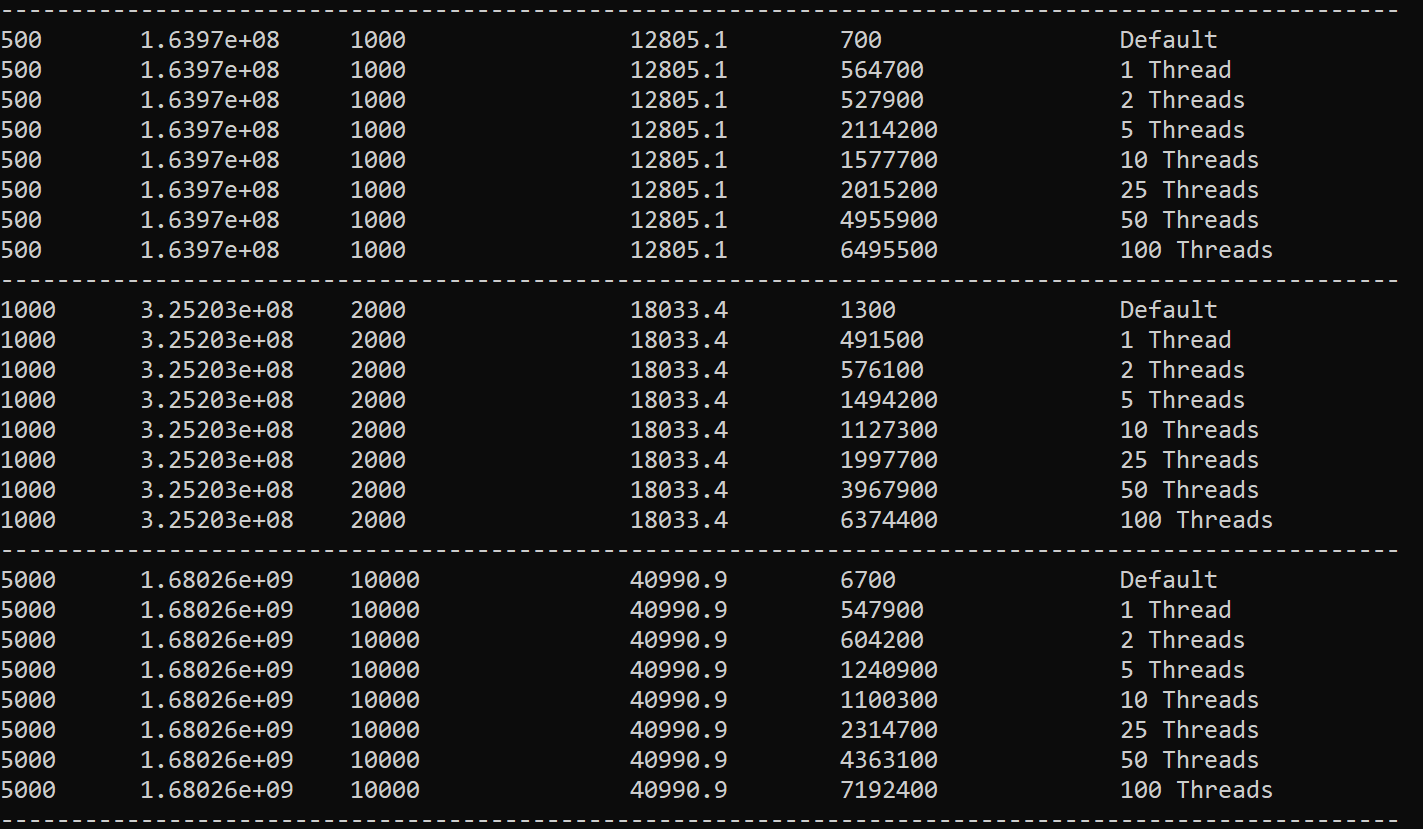
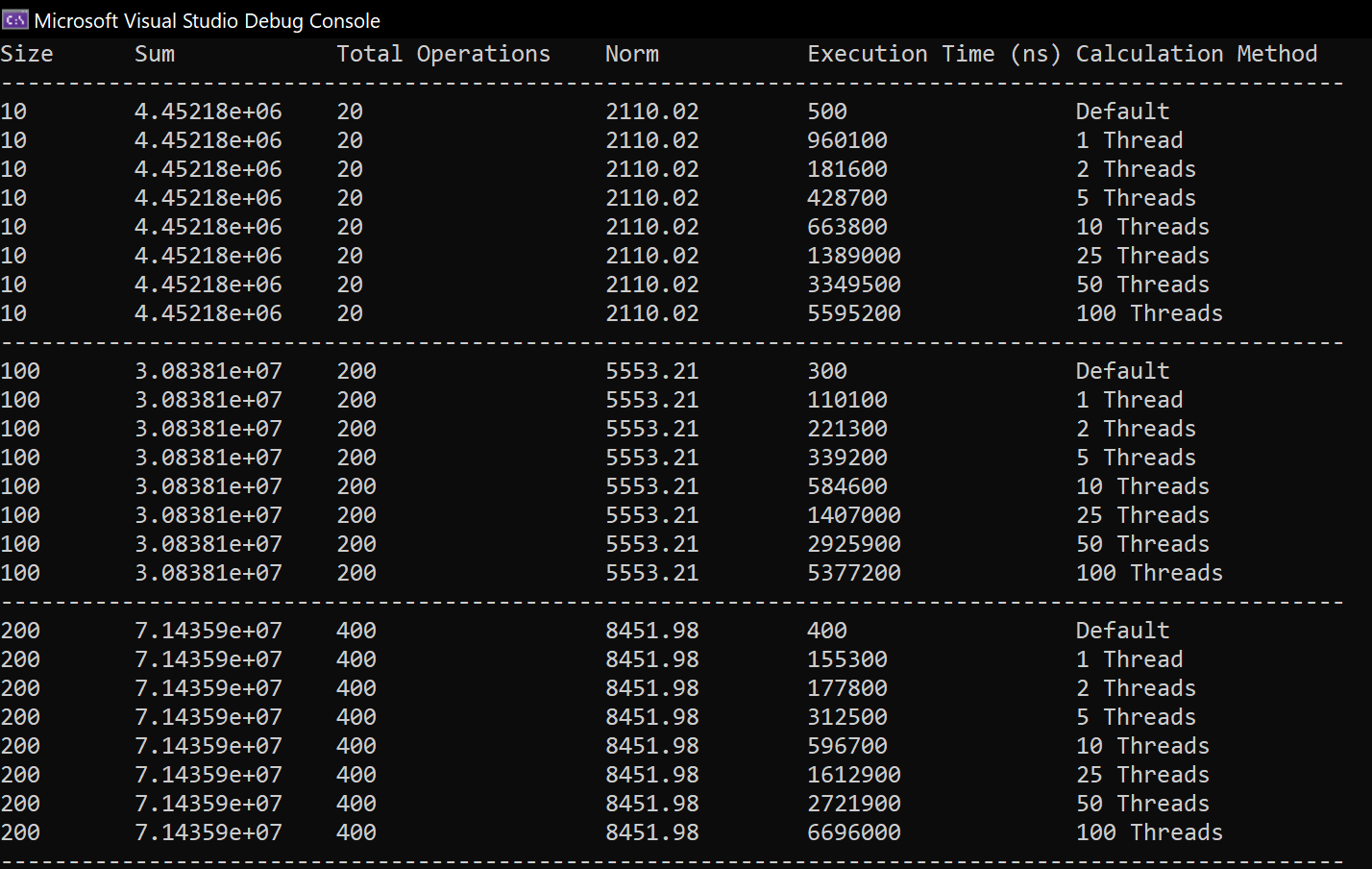


Рисунок 5 - 8 - Результат другого запуску програми у консолі

**ВИСНОВКИ**

В ході виконання роботи було реалізовано обчислення норми вектора заповненого випадковими значеннями від 1 до 1000. Протестовано послідовне (default 1 thread) та паралельне (створенням 1 - 100 потоків) вирішення завдання. Для кожного типу обчислення було заміряно:

* Size - розмір вектора
* Sum - сума усіх елементів вектора
* Total operations - загальна кількість дій при обчисленні
* Norm - результат номри вектора
* Execution time - час виконання обчислення норми вектора
* Calculation method - метод обчислення

Було протестовано такі розмірності вектора: 10, 100, 200, 500, 1к, 5к, 10к, 50к, 100к, 500к, 1m, 5m, 10m, 50m та 100m.

Показники “sum”, “total operations” та “norm” забезбечили звірення правильності обчислень.

У “calculation method” розглянути послідовне обчислення, створення 1 потоку (щоб наочно побачити ресурсомісткість операцій над потоками в порівнянні із одним потоком при паралельному обчисленні за замовчуванням).

“Calculation time” показав наступні результати:

Операції із потоками є ресурснозатратними, що видно із порівняння між першими двома колонками порівняльної таблиці. При розмірі вектору до мільйону чисел, створення потоку явно демонструє свою “ціну”, саме тому на даних до 500к чисел найефективніше працює послідовне обчислення.

Створення великої кількості потоків недоцільне через вище згадану особливість, тому 10 та більше потоків потребують надлишок ресурсів, що значно зповільнює роботу. Виграшу у продуктивності вони не забезпечать, хіба що зрівняються із 2-5 потоками при гігантській кількості даних, де дії над потоками складатимуть незначну частку ресурсів.

Для кількості даних понад 500к чудовий результат показують 2-5 потоків.

Саме така кількість найчастіше використовується для оптимізації відносно необ’ємних завдань при паралельному обчисленні.

Під час паралельних обчислень іноді трапляються “аномалії” через які час обчислень значно підвищується, як показано на рисунках 3 та 4. Таке може траплятися через особливості операційної системи, поточними процесами та характеристики персонального комп’ютера. У моєму випадку обчислення були виконані у “release” режимі visual studio 2022 із вимкненими сторонніми програмами та із рівнем використованої операційної пам’яті 30%

Посилання на репозиторій із матеріалами - [GitHub](https://github.com/anton265463/Lab1_Parellel_Computing)