**Міністерство освіти і науки України**

**Національний університет "Львівська політехніка"**

**Кафедра ЕОМ**

****

**Звіт з лабораторної роботи №2**

**з дисципліни “Паралельні та розподілені обчислення ”**

**Варіант 1**

**Виконав: студент .гр. КІ-33**

**Барсієнко В.О.**

**Прийняв: асистент**

**Козак Н.Б.**

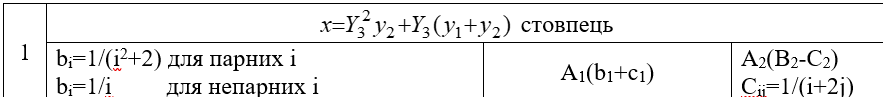
**Львів 2020 р.**

**Мета:** Набути навиків програмної реалізації паралельних обчислень.

**Завдання:**

Використовуючи засоби OpenMP, розробити програму для обчислення запропонованого матрично-векторного виразу, який би виконувався паралельно.

**Індивідуальне завдання:**



**Код програми:**

#include <iostream>

#include <vector>

#include <ctime>

#include <iomanip>

#include <omp.h>

#include <thread>

#include <chrono>

using namespace std;

void resOut(int n, vector<float> &result, vector<float> &x) {

int i;

cout << endl;

cout << "Result(result): " << endl;

for (i = 0; i < n; i++) {

cout << fixed << setprecision(3) << result[i] << "\t";

}

cout << endl << endl;

cout << "Result parallel(x): " << endl;

for (i = 0; i < n; i++) {

cout << fixed << setprecision(3) << x[i] << "\t";

}

cout << endl;

}

void printOut(int n, vector<float> &b, vector<float>& b1, vector<float>& c1, vector<vector<float>> &A, vector<vector<float>> &A1, vector<vector<float>> &A2, vector<vector<float>> &B2, vector<vector<float>> &C2) {

int i, j;

cout << "b" << endl;

for (i = 0; i < n; i++) {

cout << b[i] << "\t";

}

cout << endl << endl;

cout << "b1" << endl;

for (i = 0; i < n; i++) {

cout << b1[i] << "\t";

}

cout << endl << endl;

cout << "c1" << endl;

for (i = 0; i < n; i++) {

cout << c1[i] << "\t";

}

cout << endl << endl;

cout << "A" << endl;

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

cout << A[i][j] << "\t";

}

cout << endl;

}

cout << endl;

cout << "A1" << endl;

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

cout << A1[i][j] << "\t";

}

cout << endl;

}

cout << endl;

cout << "A2" << endl;

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

cout << A2[i][j] << "\t";

}

cout << endl;

}

cout << endl;

cout << "B2" << endl;

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

cout << B2[i][j] << "\t";

}

cout << endl;

}

cout << endl;

cout << "C2" << endl;

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

cout << fixed << setprecision(5) << C2[i][j] << setw(3) << "\t";

}

cout << endl;

}

cout << endl;

}

//func naming :

//3 letters ABC => A and C - matrix(M) or vector(V);

//B - multiply(M) or add(A) or substract(S)

//func where we need matrix^2 will be done through multiplication thought M2 => MMM

//myfuncs : MSM, MMM, MAM, VAV, MMV

vector<vector<float>> MAM(vector<vector<float>> a, vector<vector<float>> b, int size) {

vector<vector<float>> res;

res.resize(size);

for (int i = 0; i < size; ++i) {

res[i].resize(size);

}

for (int i = 0; i < size; i++) {

for (int j = 0; j < size; j++) {

res[i][j] = (a[i][j] + b[i][j]);

}

}

return res;

}

vector<vector<float>> MSM(vector<vector<float>> a, vector<vector<float>> b, int size) {

vector<vector<float>> res;

res.resize(size);

for (int i = 0; i < size; ++i) {

res[i].resize(size);

}

for (int i = 0; i < size; i++) {

for (int j = 0; j < size; j++) {

res[i][j] = (a[i][j] - b[i][j]);

}

}

return res;

}

vector<vector<float>> MMM(vector<vector<float>> a, vector<vector<float>> b, int size) {

vector<vector<float>> res;

res.resize(size);

for (int i = 0; i < size; ++i) {

res[i].resize(size);

}

for (int i = 0; i < size; i++) {

for (int j = 0; j < size; j++) {

res[i][j] = 0;

for (int k = 0; k < size; k++) {

res[i][j] += a[i][k] \* b[k][j];

}

}

}

return res;

}

vector<float> VAV(vector<float> a, vector<float> b, int size) {

vector<float> res;

res.resize(size);

for (int i = 0; i < size; i++) {

res[i] = a[i] + b[i];

}

return res;

}

vector<float> MMV(vector<vector<float>> a, vector<float> b, int size) {

vector<float> res;

res.resize(size);

for (int i = 0; i < size; i++) {

res[i] = 0;

for (int j = 0; j < size; j++) {

res[i] += (a[i][j] \* b[j]);

}

}

return res;

}

int main() {

omp\_set\_num\_threads(4);

srand(time(NULL));

int n, i, j, k, choice = 0;

cout << "x = Y3^2 \* y2 + Y3 \* (y1 + y2)" << endl << endl;

bigger\_than\_3:

cout << "Enter size(n):";

cin >> n;

if (n < 3) { cout << "n must be >= 3" << endl; goto bigger\_than\_3; }

vector<float> result(n);

vector<float> b(n), c1(n), b1(n), x(n), y1(n), y2(n);

vector<vector<float>> A(n, vector<float>(n)), A1(n, vector<float>(n)), A2(n, vector<float>(n)), B2(n, vector<float>(n)), C2(n, vector<float>(n)), Y3(n, vector<float>(n));

//declare temp values start

vector<float> temp(n), temp\_y1(n), temp\_y21(n), temp\_y22(n), temp\_y3\_y2(n), temp\_y3\_y12(n), temp\_y1\_y2(n);

vector<vector<float>> temp\_1(n, vector<float>(n)), temp\_b2\_c2(n, vector<float>(n)), temp\_y3(n, vector<float>(n)), temp\_y32(n, vector<float>(n));

//declare temp values end

//---------------------------------------------array initialize start

for (i = 0; i < n; i++) {

if (i % 2 == 0) { b[i] = 1.0 / (((i + 1.0) \* (i + 1.0)) + 2); }

else { b[i] = 1.0 / (i + 1.0); }

}

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

C2[i][j] = 1.0 / ((i + 1.0) + (2 \* (j + 1.0)));

}

}

//-------------------------------------------array initialize end

//-----------------------------------------array in start

cout << endl << "1 Matrix manual input" << endl << "2 Matrix randomize" << endl << "Choice:";

cin >> choice;

if (choice == 1) {

for (i = 0; i < n; i++) {

cout << "b1[" << i + 1 << "]= ";

cin >> b1[i];

}

for (i = 0; i < n; i++) {

cout << "c1[" << i + 1 << "]= ";

cin >> c1[i];

}

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

cout << "A[" << i + 1 << "][" << j + 1 << "]= ";

cin >> A[i][j];

}

}

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

cout << "A1[" << i + 1 << "][" << j + 1 << "]= ";

cin >> A1[i][j];

}

}

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

cout << "A2[" << i + 1 << "][" << j + 1 << "]= ";

cin >> A2[i][j];

}

}

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

cout << "B2[" << i + 1 << "][" << j + 1 << "]= ";

cin >> B2[i][j];

}

}

}

else if (choice == 2) {

for (i = 0; i < n; i++) {

b1[i] = rand() % 100;

}

for (i = 0; i < n; i++) {

c1[i] = rand() % 100;

}

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

A[i][j] = rand() % 100;

}

}

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

A1[i][j] = rand() % 100;

}

}

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

A2[i][j] = rand() % 100;

}

}

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

B2[i][j] = rand() % 100;

}

}

}

//--------------------------------------------array in end

cout << "--------------------------------------------------" << endl;

//--------------------------------------array out start

printOut(n, b, b1,c1, A, A1, A2, B2, C2);

//------------------------------------------array out end

//-----------------------------------------calculating start

temp\_y1 = MMV(A, b, n);//y1

temp\_y21 = VAV(b1, c1, n);//y2

temp\_y22 = MMV(A1, temp\_y21, n);//y2

temp\_b2\_c2 = MSM(B2, C2, n);//y3

temp\_y3 = MMM(A2, temp\_b2\_c2, n);//y3

temp\_y32 = MMM(temp\_y3, temp\_y3, n);//y3^2

temp\_y3\_y2 = MMV(temp\_y32, temp\_y22, n);//y3^2 \* y2

temp\_y1\_y2 = VAV(temp\_y1, temp\_y22, n);//y1 + y2

temp\_y3\_y12 = MMV(temp\_y3, temp\_y1\_y2, n);//y3 \* (y1+y2)

result = VAV(temp\_y3\_y2, temp\_y3\_y12, n);//x = temp\_y3\_y2 + temp\_y3\_y12 = Y3^2 \* y2 + Y3 \* (y1 + y2)

//-----------------------------------------calculating end

//------------------------------------------parallel start

//y1=A\*b

#pragma omp parallel for

for (int i = 0; i < n; i++) {

y1[i] = 0;

#pragma omp parallel for

for (int j = 0; j < n; j++) {

y1[i] += (A[i][j] \* b[j]);

}

}

//y2=A1(b1+c1)

#pragma omp parallel for

for (i = 0; i < n; i++) {

temp[i] = b1[i] + c1[i];

}

#pragma omp parallel for

for (int i = 0; i < n; i++) {

y2[i] = 0;

#pragma omp parallel for

for (int j = 0; j < n; j++) {

y2[i] += (A1[i][j] \* temp[j]);

}

}

//y3=A2(B2-C2)

#pragma omp parallel for

for (int i = 0; i < n; i++) {

#pragma omp parallel for

for (int j = 0; j < n; j++) {

B2[i][j] = (B2[i][j] - C2[i][j]);

}

}

#pragma omp parallel for

for (i = 0; i < n; i++) {

#pragma omp parallel for

for (j = 0; j < n; j++) {

Y3[i][j] = 0;

#pragma omp parallel for

for (k = 0; k < n; k++) {

Y3[i][j] += A2[i][k] \* B2[k][j];

}

}

}

#pragma omp parallel for

for (i = 0; i < n; i++) {

temp[i] = 0;

}

//temp = Y3^2 \* y2

#pragma omp parallel for

for (i = 0; i < n; i++) {

#pragma omp parallel for

for (j = 0; j < n; j++) {

temp\_1[i][j] = 0;

#pragma omp parallel for

for (k = 0; k < n; k++) {

temp\_1[i][j] += Y3[i][k] \* Y3[k][j];

}

}

}

#pragma omp parallel for

for (int i = 0; i < n; i++) {

temp[i] = 0;

#pragma omp parallel for

for (int j = 0; j < n; j++) {

temp[i] += (temp\_1[i][j] \* y2[j]);

}

}

//y1 = y1 + y2

#pragma omp parallel for

for (i = 0; i < n; i++) {

y1[i] += y2[i];

}

#pragma omp parallel for

for (i = 0; i < n; i++) {

y2[i] = 0;

}

//y2 = Y3 \* (y1 + y2)

#pragma omp parallel for

for (int i = 0; i < n; i++) {

y2[i] = 0;

#pragma omp parallel for

for (int j = 0; j < n; j++) {

y2[i] += (Y3[i][j] \* y1[j]);

}

}

//x = temp + y2 = Y3^2 \* y2 + Y3 \* (y1 + y2)

#pragma omp parallel for

for (i = 0; i < n; i++) {

x[i] = temp[i] + y2[i];

this\_thread::sleep\_for(chrono::milliseconds(5));

cout << "Thread num: " << omp\_get\_thread\_num() << endl;

}

//-----------------------------------------parallel end

//result out start

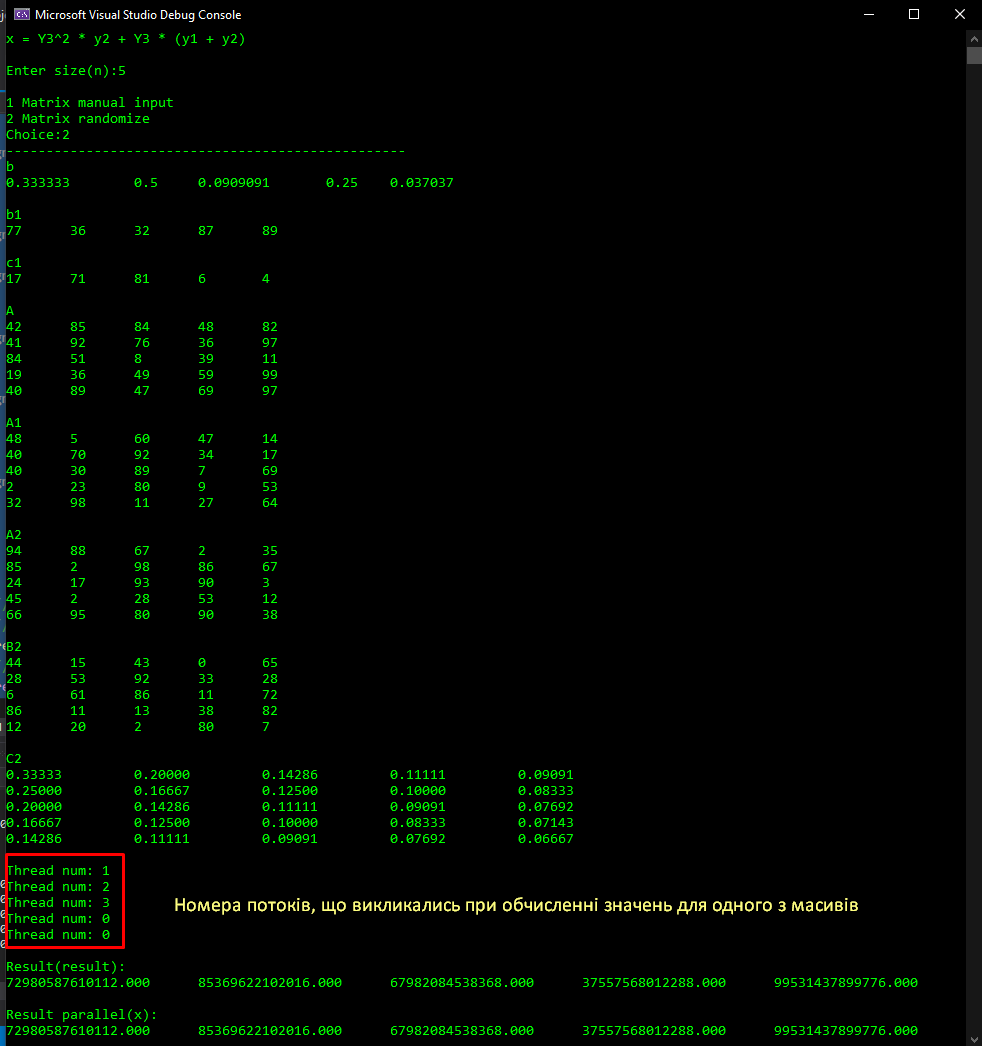
resOut(n, result, x);

//result out end

return 0;

}

**Результат виконання програми:**



**Висновок:**

На цій лабораторній роботі я вивчив методи програмної реалізації паралельних обчислень та набув навички розв’язування задач з використанням алгоритмів паралельного обчислення.