**Міністерство освіти і науки України**

**Національний університет "Львівська політехніка"**

**Кафедра ЕОМ**

****

**Звіт з лабораторної роботи №1**

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

**Варіант 17**

**Виконав: студент .гр. КІ-33**

**Скрипник Д.О.**

**Прийняв: асистент**

**Козак Н.Б.**

**Львів 2020 р.**

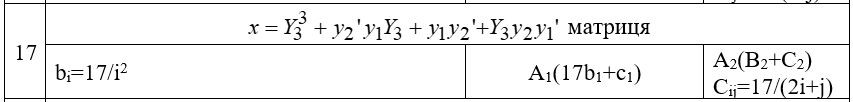
**Мета:** Вивчити методи декомпозицій задач. Набути навиків розв’язування задач з використанням функціональної декомпозиції.

**Завдання:**

Використовуючи метод функціональної декомпозиції, розробити алгоритм обчислення запропонованого матрично-векторного виразу, який би враховував можливість паралельного виконання і був оптимальним з точки зору часових затрат.

На основі створеного алгоритму написати програму яка дозволяє обчислити вираз та ілюструє проведену декомпозицію.

**Індивідуальне завдання:**



**Код програми:**

#include <iostream>

#include <vector>

#include <ctime>

#include <iomanip>

#include <cmath>

#define initV(A, N) A.resize(N, 0);

#define initM(A, N) A.resize(N); for(int i = 0;i<N;++i) initV(A[i], N)

#define zeroV(A) for(size\_t i = 0;i<A.size();++i) A[i] = 0.0;

#define zeroM(A) for(size\_t j = 0;j<A.size();++j) zeroV(A[j])

#define outputV(A) for(size\_t i = 0;i<A.size();++i) cout << A[i] << "\t";

#define outputM(A) for(size\_t j = 0;j<A.size();++j) { outputV(A[j]) cout << endl;}

using namespace std;

class Container

{

private:

int N, i, j, k;

public:

vector<vector<double>> result, x;

vector<double> b, c1, b1, y1, y2;

vector<vector<double>> A, A1, A2, B2, C2, Y3;

//declare temp values start

double tD;

vector<double> tempV, tempV\_1;

vector<vector<double>> tempM, tempM\_1, tempM\_Y3, tempM\_2, tempM\_3;

Container(int N)

{

this->N = N;

i = j = k = 0;

tD = 0;

}

void Initialize()

{

//

initM(x, N);

initM(result, N);

initV(b, N);

initV(b1, N);

initV(c1, N);

initV(y1, N);

initV(y2, N);

initM(A, N);

initM(A1, N);

initM(A2, N);

initM(B2, N);

initM(C2, N);

initM(Y3, N);

//

initV(tempV, N);

initV(tempV\_1, N);

initM(tempM, N);

initM(tempM\_1, N);

initM(tempM\_2, N);

initM(tempM\_3, N);

initM(tempM\_Y3, N);

ConstInit();

}

void ConstInit()

{

for (i = 0; i < N; i++) {

b[i] = 17.0 / pow(i + 1, 2);

}

for (i = 0; i < N; i++) {

for (j = 0; j < N; j++) {

C2[i][j] = 17.0 / (2 \* (i + 1.0) + (j + 1.0));

}

}

}

void ManualInputArray()

{

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];

}

}

}

void RandomizeArray()

{

const int R = 20;

for (i = 0; i < N; i++) {

b1[i] = rand() % R;

}

for (i = 0; i < N; i++) {

c1[i] = rand() % R;

}

for (i = 0; i < N; i++) {

for (j = 0; j < N; j++) {

A[i][j] = rand() % R;

}

}

for (i = 0; i < N; i++) {

for (j = 0; j < N; j++) {

A1[i][j] = rand() % R;

}

}

for (i = 0; i < N; i++) {

for (j = 0; j < N; j++) {

A2[i][j] = rand() % R;

}

}

for (i = 0; i < N; i++) {

for (j = 0; j < N; j++) {

B2[i][j] = rand() % R;

}

}

}

void OutputArrays()

{

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<double> MMV(vector<vector<double>> a, vector<double> b) {

vector<double> res;

initV(res, N);

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

for (int j = 0; j < N; j++) {

res[i] += (a[i][j] \* b[j]);

}

}

return res;

}

vector<vector<double>> MMM(vector<vector<double>> a, vector<vector<double>> b) {

vector<vector<double>> res;

initM(res, N)

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

for (int j = 0; j < N; j++) {

for (int k = 0; k < N; k++) {

res[i][j] += a[i][k] \* b[k][j];

}

}

}

return res;

}

vector<vector<double>> DMM(double a, vector<vector<double>> b) {

vector<vector<double>> res;

initM(res, N);

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

for (int j = 0; j < N; j++) {

res[i][j] = a \* b[i][j];

}

}

return res;

}

vector<vector<double>> MAM(vector<vector<double>> a, vector<vector<double>> b) {

vector<vector<double>> res;

initM(res, N);

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

for (int j = 0; j < N; j++) {

res[i][j] = double(a[i][j]) + double(b[i][j]);

}

}

return res;

}

vector<double> DMV(double a, vector<double> b) {

vector<double> res;

initV(res, N);

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

res[i] = a \* b[i];

}

return res;

}

vector<double> VAV(vector<double> a, vector<double> b) {

vector<double> res;

initV(res, N);

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

res[i] = a[i] + b[i];

}

return res;

}

vector<vector<double>> VcolMVstr(vector<double> a, vector<double> b) {

vector<vector<double>> res;

initM(res, N);

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

for (int j = 0; j < N; j++) {

res[i][j] = b[j] \* a[i];

}

}

return res;

}

double VstrMVcol(vector<double> a, vector<double> b) {

double res = 0;

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

res += a[i] \* b[i];

}

return res;

}

//

void Check()

{

#pragma region y1=A\*b

zeroV(y1);

for (i = 0; i < N; i++) {

for (j = 0; j < N; j++) {

y1[i] += (A[i][j] \* b[j]);

}

}

#pragma endregion

#pragma region y2=A1(17\*b1+c1)

zeroV(tempV);

zeroV(y2);

for (i = 0; i < N; i++) {

tempV[i] = 17 \* b1[i] + c1[i];

}

for (i = 0; i < N; i++) {

for (j = 0; j < N; j++) {

y2[i] += (A1[i][j] \* tempV[j]);

}

}

#pragma endregion

#pragma region y3=A2(B2+C2)

zeroM(tempM\_1);

zeroM(Y3);

for (i = 0; i < N; i++) {

for (j = 0; j < N; j++) {

tempM\_1[i][j] = (B2[i][j] + C2[i][j]);

}

}

for (i = 0; i < N; i++) {

for (j = 0; j < N; j++) {

for (k = 0; k < N; k++) {

Y3[i][j] += A2[i][k] \* tempM\_1[k][j];

}

}

}

#pragma endregion

#pragma region tempM\_1 = y1 \* y2\_ => matrix

zeroM(tempM\_1);

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

for (int j = 0; j < N; j++) {

tempM\_1[i][j] = y2[j] \* y1[i];

}

}

//tempM\_1 = VcolMVstr(y1, y2);

#pragma endregion

#pragma region temp\_2 = Y3\*y2\*y1\_ => matrix

zeroV(tempV\_1);

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

for (int j = 0; j < N; j++) {

tempV\_1[i] += (Y3[i][j] \* y2[j]);

}

}

//tempV\_1 = MMV(Y3, y2);

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

for (int j = 0; j < N; j++) {

tempM\_2[i][j] = y1[j] \* tempV\_1[i];

}

}

//tempM\_2 = VcolMVstr(tempV\_1, y1);

#pragma endregion

#pragma region temp\_3 = y2\_\*y1\*Y3 => matrix

tD = 0;

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

tD += y2[i] \* y1[i];

}

//tD = VstrMVcol(y2, y1);

zeroM(tempM\_3);

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

for (int j = 0; j < N; j++) {

tempM\_3[i][j] = tD \* Y3[i][j];

}

}

//tempM\_3 = DMM(tD, Y3);

#pragma endregion

#pragma region temp\_Y3 = Y3 ^ 3 => matrix

tempM\_Y3 = MMM(MMM(Y3,Y3),Y3);

#pragma endregion

#pragma region x = Y3^3 + y2\_\*y1\*Y3 + y1\*y2\_ + Y3\*y2\*y1\_

//x = MAM(MAM(MAM(tempM\_Y3, tempM\_3), tempM\_1), tempM\_2);

zeroM(x);

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

for (int j = 0; j < N; j++)

{

x[i][j] = tempM\_Y3[i][j] + tempM\_3[i][j] + tempM\_1[i][j] + tempM\_2[i][j];

}

}

#pragma endregion

}

void PsevdoParallel()

{

tempM = MAM(B2, C2); //B2+C2

tempV = DMV(17, b1); //17\*b1

Y3 = MMM(A2, tempM); //y3 = A2\*(B2+C2)

tempV\_1 = VAV(tempV, c1); //17\*b1+c1

y1 = MMV(A, b); //y1=A\*b

tempM\_Y3 = MMM(Y3, Y3); //y3^2

y2 = MMV(A1, tempV\_1); //y2 = A1\*(17\*b1+c1)

tempM\_Y3 = MMM(tempM\_Y3, Y3);//y3^3

tempV\_1 = MMV(Y3, y2);//Y3 \* y2

tempM\_1 = VcolMVstr(y1, y2); //y1 \* y2'

tD = VstrMVcol(y2, y1); //y2' \* y1

tempM\_2 = VcolMVstr(tempV\_1, y1); //Y3\*y2\*y1'

tempM\_3 = DMM(tD, Y3); //y2'\*y1\*Y3

result = MAM(tempM\_Y3, tempM\_3); //x = Y3^3 + y2'\*y1\*Y3

result = MAM(result, tempM\_1); //x = Y3^3 + y2'\*y1\*Y3 + y1\*y2'

result = MAM(result, tempM\_2); //x = Y3^3 + y2'\*y1\*Y3 + y1\*y2' + Y3\*y2\*y1'

}

void OutputResult()

{

cout << "Result(result): " << endl;

for (i = 0; i < N; i++) {

for (j = 0; j < N;++j) {

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

}

cout << endl;

}

cout << endl << endl;

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

for (i = 0; i < N; i++) {

for (j = 0; j < N; ++j) {

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

}

cout << endl;

}

cout << endl;

}

};

int main()

{

srand(time(NULL));

int n, choice = 0;

cout << "x = Y3^3 + y2'\*y1\*Y3 + y1\*y2' + Y3\*y2\*y1'" << endl << endl;

m:

cout << "Enter size(n):";

cin >> n;

if (n < 3)

{

cout << "n must be >= 3" << endl;

goto m;

}

//---------------------------------------------array declare start

Container container = Container(n);

container.Initialize();

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

k:

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

cin >> choice;

if (choice == 1) {

container.ManualInputArray();

}

else if (choice == 2) {

container.RandomizeArray();

}

else

{

cout << "Choose correct value" << endl;

goto k;

}

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

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

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

container.OutputArrays();

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

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

container.PsevdoParallel();

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

//------------------------------------------check start

container.Check();

//-----------------------------------------check end

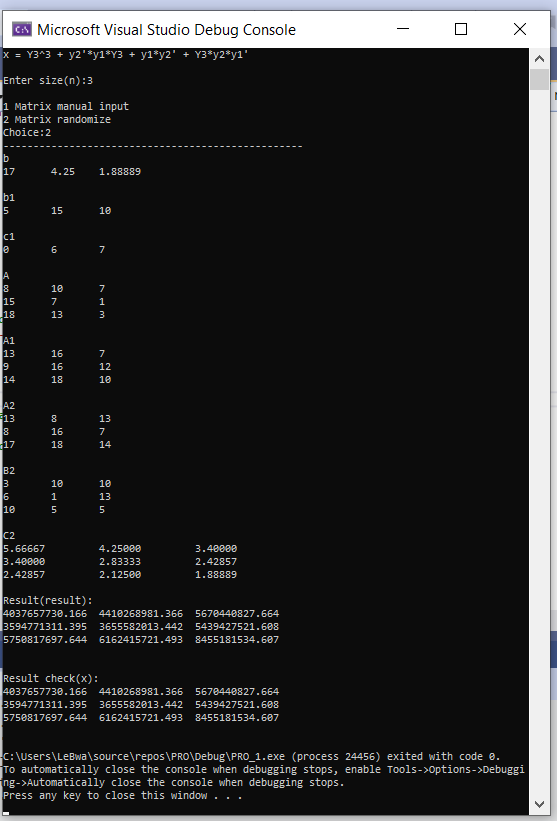
//result out start

container.OutputResult();

//result out end

}

**Результат виконання програми:**



**Висновок:**

На цій лабораторній роботі я вивчив методи декомпозицій задач та набув навички розв’язування задач з використанням функціональної декомпозиції.