Міністерство освіти і науки, молоді та спорту України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



**Звіт**

з лабораторної роботи № 2

з дисципліни: “Паралельні та розподілені обчислення”

на тему: “Паралельне представлення алгоритмів”

Виконав: студент гр.. КІ-34

Юсько В.А.

Прийняв:

Козак Н. Б.

Львів

2020

**Мета роботи:** Вивчити можливості паралельного представлення алгоритмів. Набути навиків такого представлення

**Завдання:**

Запропонувати та реалізувати локально-рекурсивний алгоритм обчислення виразу:

 ,

де А та В матриці з елементами  та , відповідно(), тобто:

 () .

Тип вхідних послідовностей визначається згідно варіанту.

**Варіант завдання – 16**

Матриця А задається однозначно і залежить лише від розмірності даних.

Для матриці В: заштрихована область – довільні цілі числа, відмінні від нуля, а не заштрихована область – нулі.

|  |  |  |
| --- | --- | --- |
| **варіант**  **№** | **Тип матриці А** | **Тип матриці В** |
| **16** | n 0 .0 ... 0  n-1 n 0 . ..0  n-2 n-1 n ....  ...  1 2 3 … n |  |

**Текст програми з одноразовим присвоюванням**

// PRO\_lab2.cpp : This file contains the 'main' function. Program execution begins and ends there.

//

#include "pch.h"

#include <iostream>

#include <ctime>

using namespace std;

void main()

{

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

int N;

cout << " Введiть порядок n: ";

cin >> N;

int\*\* A = new int\*[N]; // а - указатель на указатели, создается массив указателей

cout << " \t \============\ \tЗаповнення матрицi A.. \t\t \============\ \n";

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

{

A[i] = new int[N];

for (int j = i + 1; j < N; j++)

{

A[i][j] = 0;

}

}

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

{

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

{

if (j == 0)

{

A[i][j] = N - i;

}

else if (j > 0)

{

A[i][j] = A[i - 1][j - 1];

}

}

}

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

{

cout << "\t";

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

{

cout << "\t" << A[i][j];

}

cout << endl;

}

int\*\* B = new int\*[N];

cout << " \t \============\ \tЗаповнення матрицi B.. \t \============\ \n";

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

{

B[i] = new int[N];

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

{

B[i][j] = 0;

}

}

for (int i = N / 2 - 1; i >= 0; i--)

{

for (int j = N - i - 1; j < N; j++)

{

B[i][j] = rand() % 50 + 1;

}

}

for (int i = N / 2; i < N; i++)

{

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

{

B[i][j] = rand() % 50 + 1;

}

}

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

{

cout << "\t";

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

{

cout << "\t" << B[i][j];

}

cout << endl;

}

cout << endl;

int \*\*\*Y;

Y = new int \*\*[N];

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

Y[i] = new int \*[N];

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

Y[i][j] = new int[N];

}

}

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

{

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

{

Y[i][j][0] = 0;

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

Y[i][j][k + 1] = Y[i][j][k] + (A[i][k] \* B[k][j]);

}

}

}

cout << " \t \============\ \tМатриця Y.. \t \============\ \n";

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

{

cout << "\t";

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

{

cout << "\t" << Y[i][j][N];

}

cout << endl;

}

cout << endl;

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

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

delete[] Y[i][j];

}

delete[] Y[i];

}

delete[] Y;

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

delete[] A[i];

delete[] B[i];

}

delete[] A;

delete[] B;

}

// Run program: Ctrl + F5 or Debug > Start Without Debugging menu

// Debug program: F5 or Debug > Start Debugging menu

// Tips for Getting Started:

// 1. Use the Solution Explorer window to add/manage files

// 2. Use the Team Explorer window to connect to source control

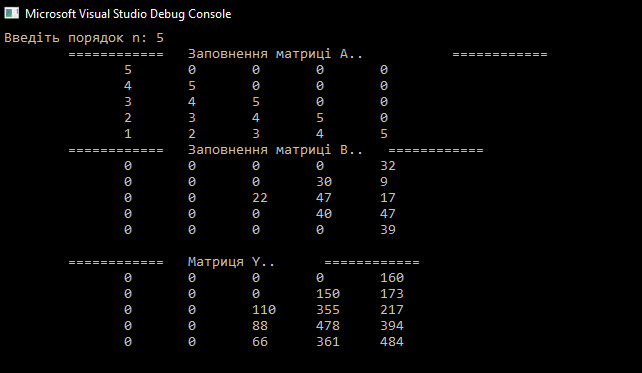
// 3. Use the Output window to see build output and other messages

// 4. Use the Error List window to view errors

// 5. Go to Project > Add New Item to create new code files, or Project > Add Existing Item to add existing code files to the project

// 6. In the future, to open this project again, go to File > Open > Project and select the .sln file

**Результат роботи**



**Рекурсивні рівняння**

,

де :

* – індекс рекурсії

**Локалізований граф залежностей**

*i*

*j*

*k*

*A(1)*

*A(1)*

*A(2)*

*A(2)*

*B(2)*

*B(2)*

*B(1)*

*B(1)*

*Y(1)*

*Y(3)*

*Y(2)*

*Y(4)*

*1*

*1*

*1*

*2*

*2*

*2*

**Оптимізований локалізований граф залежностей**

*i*

*j*

*k*

*A(1)*

*A(1)*

*A(2)*

*A(2)*

*Y(1)*

*Y(3)*

*Y(2)*

*Y(4)*

*1*

*2*

*1*

*1*

*2*

*2*

**Аналітичні оцінки кількості арифметичних операцій та їх порівняння**

Для неоптимізованого графу для 1 ітерації виконується 4 операції. Всього 60 ітерацій, а значить – 240 операцій. Для оптимізованого графу для 1 ітерації виконується 3 операції. Всього 60 ітерацій, а значить – 180 операцій. Кількість операцій скоротилась на 60 штук

**Текст програми, що реалізовує оптимізований локально-рекурсивний алгоритм**

// PRO\_lab2.cpp : This file contains the 'main' function. Program execution begins and ends there.

//

#include "pch.h"

#include <iostream>

#include <ctime>

using namespace std;

void main()

{

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

/\*

int \*\*\*Y;

Y = new int \*\*[N];

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

Y[i] = new int \*[N];

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

Y[i][j] = new int[N];

}

}\*/

int N;

cout << " Введiть порядок n: ";

cin >> N;

int\*\* A = new int\*[N]; // а - указатель на указатели, создается массив указателей

cout << " \t \============\ \tЗаповнення матрицi A.. \t\t \============\ \n";

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

{

A[i] = new int[N];

for (int j = i + 1; j < N; j++)

{

A[i][j] = 0;

}

}

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

{

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

{

if (j == 0)

{

A[i][j] = N - i;

}

else if (j > 0)

{

A[i][j] = A[i - 1][j - 1];

}

}

}

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

{

cout << "\t";

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

{

cout << "\t" << A[i][j];

}

cout << endl;

}

int \*\*\*A\_2;

A\_2 = new int \*\*[N];

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

A\_2[i] = new int \*[N];

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

A\_2[i][j] = new int[N];

A\_2[i][j][0] = A[i][j];

}

}

int\*\* B = new int\*[N];

cout << " \t \============\ \tЗаповнення матрицi B.. \t \============\ \n";

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

{

B[i] = new int[N];

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

{

B[i][j] = 0;

}

}

for (int i = N / 2 - 1; i >= 0; i--)

{

for (int j = N - i - 1; j < N; j++)

{

B[i][j] = rand() % 50 + 1;

}

}

for (int i = N / 2; i < N; i++)

{

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

{

B[i][j] = rand() % 50 + 1;

}

}

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

{

cout << "\t";

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

{

cout << "\t" << B[i][j];

}

cout << endl;

}

cout << endl;

/\*int \*\*\*B\_2;

B\_2 = new int \*\*[N];

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

B\_2[i] = new int \*[N];

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

B\_2[i][j] = new int[N];

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

{

B\_2[i][j][k] = B[i][j];

}

}

}

cout << endl;

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

{

cout << "\t";

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

{

cout << "\t" << B\_2[i][j][0];

}

cout << endl;

}

cout << endl;\*/

int \*\*\*Y;

Y = new int \*\*[N];

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

Y[i] = new int \*[N];

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

Y[i][j] = new int[N];

}

}

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

{

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

{

Y[i][j][0] = 0;

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

Y[i][j][k + 1] = Y[i][j][k] + A[i][k] \* B[k][j];

}

}

}

cout << " \t \============\ \tМатриця Y (одноразове присвоєння).. \t \============\ \n";

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

{

cout << "\t";

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

{

cout << "\t" << Y[i][j][N];

}

cout << endl;

}

cout << endl;

// Неоптимізований алгоритм

/\*for (int i = 0; i < N; i++)

{

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

{

Y[i][j][0] = 0;

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

A\_2[i][j][k + 1] = A\_2[i][j][k];

B\_2[i][j][k + 1] = B\_2[i][j][k];

Y[i][j][k + 1] = Y[i][j][k] + A\_2[i][k][j] \* B\_2[k][j][i];

}

}

}\*/

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

{

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

{

Y[i][j][0] = 0;

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

A\_2[i][j][k + 1] = A\_2[i][j][k];

//B\_2[i][j][k + 1] = B\_2[i][j][k];

Y[i][j][k + 1] = Y[i][j][k] + A\_2[i][k][j] \* B[k][j];

}

}

}

cout << " \t \============\ \tМатриця Y (локалiзований алгоритм).. \t \============\ \n";

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

{

cout << "\t";

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

{

cout << "\t" << Y[i][j][N];

}

cout << endl;

}

cout << endl;

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

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

delete[] Y[i][j];

delete[] A\_2[i][j];

//delete[] B\_2[i][j];

}

delete[] Y[i];

delete[] A\_2[i];

//delete[] B\_2[i];

}

delete[] Y;

delete[] A\_2;

//delete[] B\_2;

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

delete[] A[i];

delete[] B[i];

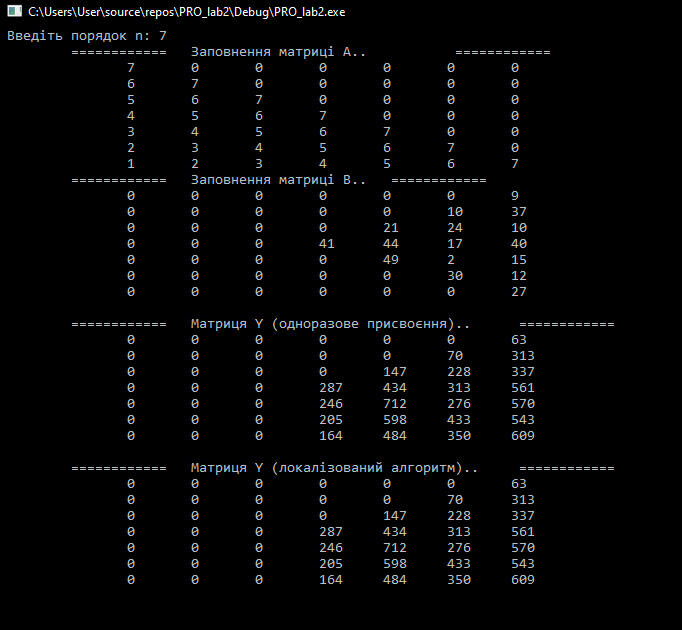
}

delete[] A;

delete[] B;

}

**Результат роботи**



**Висновок:**

Під час виконання лабораторної роботи я вивчив можливості паралельного представлення алгоритмів та набув навиків такого представлення