**Міністерство освіти і науки України**

**Національний університет "Львівська політехніка"**

**Кафедра ЕОМ**



**Звіт з лабораторної роботи №2**

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

**на тему: ” ПАРАЛЕЛЬНЕ ПРЕДСТАВЛЕННЯ АЛГОРИТМІВ ”**

**Виконав: ст. гр. КІ-33**

**Красноштан О.О.**

**Прийняв: викладач**

**Козак Н.Б.**

**Львів 2020 р.**

**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 – 5

|  |  |  |
| --- | --- | --- |
| **варіант**  **№** | **Тип матриці А** | **Тип матриці В** |
| **5** | 100…001  110…011  …………  110…011  100…001 |  |

**Хід роботи:**

1. Пишу програму з одноразовим присвоюванням.
2. Знаходжу рекурсивні рівняння:

Yi(j+1) = Yi(j) + Ai(k) \* Bk(j)

Ai(k) = A[i][k]

Bk(j) = B[k][j]

J - індекс рекурсії

1. Будую граф залежностей та виходячи з нього локалізований граф залежностей
2. Пишу програму що реалізує локально-рекурсивний алгоритм.

Код програми наведений в додатку А

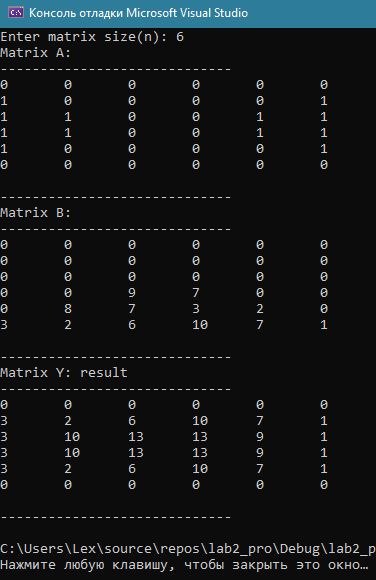


Рис 1. Результат виконання програми

**Висновок**: вивчив можливості паралельного представлення алгоритмів. Набув навиків такого представлення.

**Додаток А**

#include <iostream>

#include <vector>

#include <ctime>

using namespace std;

void DisplayMatrix(int n, vector<vector<int>>& matrix) {

int i, j;

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

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

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

}

cout << endl;

}

cout << endl;

}

vector<vector<int>> MatrixMulMatrix(vector<vector<int>> a, vector<vector<int>> b, int size) {

vector<vector<int>> 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<vector<int>> InitMatrixA(int size) {

vector<vector<int>> res;

res.resize(size);

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

res[i].resize(size);

}

int startIndex = 0, lastIndex = size - 1;

for (int i = 0; i < size / 2; i++) {

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

if (j <= lastIndex && j >= startIndex) {

res[i][j] = 0;

}

else res[i][j] = 1;

}

startIndex++;

lastIndex--;

}

if (size % 2 == 0) {

startIndex--;

lastIndex++;

}

for (int i = size / 2; i < size; i++) {

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

if (j <= lastIndex && j >= startIndex) {

res[i][j] = 0;

}

else res[i][j] = 1;

}

startIndex--;

lastIndex++;

}

return res;

return res;

}

vector<vector<int>> InitMatrixB(int size) {

vector<vector<int>> res;

res.resize(size);

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

res[i].resize(size);

}

int startIndex = 0, lastIndex = size - 1;

for (int i = 0; i < size / 2; i++) {

startIndex++;

lastIndex--;

}

if (size % 2 == 0) {

startIndex--;

lastIndex++;

}

for (int i = size / 2; i < size; i++) {

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

if (j <= lastIndex && j >= startIndex) {

res[i][j] = rand() % 10 + 1;

}

else res[i][j] = 0;

}

startIndex--;

lastIndex++;

}

return res;

}

int main() {

srand(time(NULL));

int i, j, n, k;

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

cin >> n;

vector<vector<int>>

A(n, vector<int>(n, 0)),

B(n, vector<int>(n, 0)),

Y(n, vector<int>(n, 0));

cout << "Matrix A:" << endl;

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

A = InitMatrixA(n);

DisplayMatrix(n, A);

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

cout << "Matrix B:" << endl;

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

B = InitMatrixB(n);

DisplayMatrix(n, B);

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

Y = MatrixMulMatrix(A, B, n);

cout << "Matrix Y: result" << endl;

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

DisplayMatrix(n, Y);

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

return 0;

}