Міністерство освіти і науки, молоді та спорту України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №7**

З дисципліни «ПРО»

Виконав: Перевірив:

Студент групи ІО-21 доц. Корочкін О.В.

Коноз А.О.

Дата здачі\_\_\_\_\_\_\_\_\_\_\_\_\_

Захищено з балом\_\_\_\_\_

Київ 2014

**Завдання**

1.23 2.13 3.17

1. F1: E:=A+B+C+D(MA\*MZ).

2. F2: MZ:=Min(MA)\*MB+Max(MT)\*(MX\*MZ).

3. F3: d:=Min(A\*Trans(MB\*MM)+B\*Sort(MC).

**Лістинг коду**

#include<windows.h>

#include<stdio.h>

#include<omp.h>

const int N = 3000;

/\*\*

\* Виконав: Коноз Андрій

\* Група: ІО-21

\* Дата: 14.10.14

\* F1: E:=A+B+C+D(MA\*MZ)

\* F2: MZ:=Min(MA)\*MB+Max(MT)\*(MX\*MZ)

\* F3: d:=Min(A\*Trans(MB\*MM)+B\*Sort(MC))

\*/

/\*\*

\* Головний метод

\*/

int main()

{

omp\_set\_num\_threads(3);

#pragma omp parallel

{

#pragma omp sections

{

#pragma omp section

{

TF1();

}

#pragma omp section

{

TF2();

}

#pragma omp section

{

TF3();

}

}

}

getchar();

return 0;

}

/\*\*

\* Додає два вектори

\* @param vectA вектор

\* @param vectB вектор

\* @return вектор суми

\*/

int\* addVect(int\* vectA, int\* vectB){

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

vectA[i] += vectB[i];

}

return vectA;

}

/\*\*

\* Перемножає дві матриці

\* @param matrA матриця

\* @param matrB матриця

\* @return повертає добуток двох матриць

\*/

int\*\* multMatr(int\*\* matrA, int\*\* matrB){

int temp = 0;

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

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

resultMatr[i] = new int[N];

}

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

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

temp = 0;

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

temp += matrA[i][k] \* matrB[k][j];

}

resultMatr[i][j] = temp;

}

}

return resultMatr;

}

/\*\*

\* Транспонує матрицю

\* @param matrA матриця для транспонування

\* @return транспоновану матрицю

\*/

int\*\* transpMatr(int\*\* matrA){

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

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

resultMatr[i] = new int[N];

}

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

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

resultMatr[i][j] = matrA[j][i];

}

}

return resultMatr;

}

/\*\*

\* Знаходить максимальний елемент в матриці

\* @param matrA матриця для пошуку

\* @return максимальний елемент матриці

\*/

int findMaxMatr(int\*\* matrA){

int max = matrA[0][0];

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

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

if(max < matrA[i][j]){

max = matrA[i][j];

}

}

}

return max;

}

/\*\*

\* Знаходить мінімальний елемент матриці

\* @param matrA матриця для пошуку

\* @return мінімальний елемент

\*/

int findMinMatr(int\*\* matrA){

int min = matrA[0][0];

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

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

if(min > matrA[i][j]){

min = matrA[i][j];

}

}

}

return min;

}

/\*\*

\* Знаходить мінімальний елемент в векторі

\* @param vectA вектор для пошуку

\* @return мінімальний елемент

\*/

int findMinVect(int\* vectA){

int min = vectA[0];

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

if(min > vectA[i]){

min = vectA[i];

}

}

return min;

}

/\*\*

\* Додає дві матриці

\* @param matrA матриця

\* @param matrB матриця

\* @return суму матриць

\*/

int\*\* addMatr(int\*\* matrA, int\*\* matrB){

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

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

resultMatr[i] = new int[N];

}

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

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

resultMatr[i][j] = matrA[i][j] + matrB[i][j];

}

}

return resultMatr;

}

/\*\*

\* Перемножає вектор на матрицю

\* @param vectA вектор

\* @param matrB матрицю

\* @return добуток вектора і матриці

\*/

int\* multVectMatr(int\* vectA, int\*\* matrB){

int temp = 0;

int\* resultVect = new int[N];

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

temp = 0;

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

temp += vectA[j]\*matrB[j][i];

}

resultVect[i] = temp;

}

return resultVect;

}

/\*\*

\* Сортує вектор за зростанням

\* @param vectA вектор для сортування

\* @return відсортований вектор

\*/

int\* sortVect(int\* vectA){

int temp = 0;

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

for (int j = 0; j < (i-1); j++) {

if(vectA[j] > vectA[j+1]){

temp = vectA[j];

vectA[j] = vectA[j+1];

vectA[j+1] = temp;

}

}

}

return vectA;

}

/\*\*

\* Сортує матрицю за спаданням

\* @param matrA матриця для сортування

\* @return відсортовану матрицю

\*/

int\*\* sortMatr(int\*\* matrA){

int temp = 0;

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

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

for (int j = 0; j < (i-1); j++) {

if(matrA[k][j] < matrA[k][j+1]){

temp = matrA[k][j];

matrA[k][j] = matrA[k][j+1];

matrA[k][j+1] = temp;

}

}

}

}

return matrA;

}

/\*\*

\* Множить матрицю на число

\* @param a число

\* @param matrB матриця

\* @return результат множення

\*/

int\*\* multNumbMatr(int a, int\*\* matrB){

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

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

matrB[i][j] = matrB[i][j]\*a;

}

}

return matrB;

}

/\*\*

\* Зчитує матрицю

\* @param n кількість елементів

\* @param message повідомлення для відображення

\* @return зчитану матрицю

\*/

int\*\* readMatr( int n, char\* message){

int\*\* matr = new int\*[n];

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

matr[i] = new int[n];

}

printf(message);

printf("\n");

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

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

matr[i][j] = 1;

}

}

return matr;

}

/\*\*

\* Відображає матрицю

\* @param matr матриця для відображення

\* @param message повідомлення для відображення

\*/

void showMatr(int\*\* matr, char\* message){

// printf(message);

// printf("\n");

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

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

printf("%d ", matr[i][j]);

}

printf("\n");

}

printf("\n");

}

/\*\*

\* Зчитує вектор

\* @param n кількість елементів вектора

\* @param message повідомлення для відображення

\* @return зчитаний вектор

\*/

int\* readVect(int n, char\* message){

int\* vect = new int [n];

printf(message);

printf("\n");

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

vect[i] = 1;

}

return vect;

}

/\*\*

\* Відображає вектор

\* @param vect вектор для відображення

\* @param message повідомлення для відображення

\*/

void showVect(int\* vect, char\* message){

printf(message);

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

printf("%d ", vect[i]);

}

printf("\n");

}

/\*

\* F1: E:=A+B+C+D(MA\*MZ)

\*/

void TF1(){

int\* vectA;

int\* vectB;

int\* vectC;

int\* vectD;

int\*\* matrA;

int\*\* matrZ;

int n = N;

int\* vectRes;

printf( "TF1 started");

vectA = readVect(n, "Enter vector A for F1");

vectB = readVect(n, "Enter vector B for F1");

vectC = readVect(n, "Enter vector C for F1");

vectD = readVect(n, "Enter vector D for F1");

matrA = readMatr(n, "Enter matrix A for F1");

matrZ = readMatr(n, "Enter matrix Z for F1");

Sleep(500);

vectRes = addVect(addVect(vectA,vectB),addVect(vectC,multVectMatr(vectD,multMatr(matrA,matrZ))));

showVect(vectRes, "Result of F1");

printf("TF1 finished\n");

}

/\*\*

\* F2: MZ:=Min(MA)\*MB+Max(MT)\*(MX\*MZ)

\*/

void TF2(){

int\*\* matrA;

int\*\* matrB;

int\*\* matrM;

int\*\* matrX;

int\*\* matrZ;

int n = N;

int\*\* matrRes;

printf("TF2 started\n");

matrA = readMatr(n, "Enter matrix A for F2");

matrB = readMatr(n, "Enter matrix B for F2");

matrM = readMatr(n, "Enter matrix M for F2");

matrX = readMatr(n, "Enter matrix X for F2");

matrZ = readMatr(n, "Enter matrix Z for F2");

Sleep(300);

matrRes = addMatr(multNumbMatr(findMinMatr(matrA),matrB),multNumbMatr(findMaxMatr(matrM),multMatr(matrX,matrZ)));

showMatr(matrRes, "Result of F2");

printf("TF2 finished\n");

}

/\*\*

\* F3: d:=Min(A\*Trans(MB\*MM)+B\*Sort(MC))

\*/

void TF3(){

int\* vectA;

int\* vectB;

int\*\* matrB;

int\*\* matrM;

int\*\* matrC;

int n = N;

int result;

printf("TF3 started\n");

vectA = readVect(n, "Enter vector A for F3");

vectB = readVect(n, "Enter vector B for F3");

matrB = readMatr(n, "Enter matrix B for F3");

matrM = readMatr(n, "Enter matrix M for F3");

matrC = readMatr(n, "Enter matrix C for F3");

Sleep(100);

result = findMinVect(addVect(multVectMatr(vectA,transpMatr(multMatr(matrB,matrM))),multVectMatr(vectB,sortMatr(matrC))));

printf("Result of F3: ");

printf("%d ", result);

printf("\n");

printf("TF3 finished\n");

}