

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«кИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. ІГОРЯ СІКОРСЬКОГО»

Кафедра обчислювальної техніки

Лабораторна робота №5

на тему : **«**Потоки в бібліотеці OpenMP**»**

Виконала

студентка 3 курсу

гр. ІО-51

ФІОТ

Поштацька К. В.

5115

Варіант 1.15, 2.29, 3.21

Перевірив:

Доцент, к.т.н.

Корочкін О.В

.

Київ 2017

**Мета роботи** вивчення засобів бібліотеки OpenMP для роботи з процесами (потоками).

**Завдання**

Розробити програму, що містить паралельні потоки (задачі), кожна з яких реалізує функцію F1, F2, F3 згідно з варіантом з додатку Б.

**Варіант завдання**

**1.15** MD = MAX(A + B + C) \*(MA\*ME)

**2.29** MF = (MG + MH)\*(MK \* ML)\*(MG + ML)

**3.21** S = SORT(O\*MO)\*(MS \*MT)

**Лістинг програми**

**Клас Lab5**

/\*

Laboratory work

on the topic "Threads in OpenMP library"

Variant: 1.15 2.29 3.21

T1: MD=MAX(A+B+C)\*(MA\*ME)

T2: MF=(MG+MH)\*(MK\*ML)\*(MG+ML)

T3: S=SORT(O\*MO)\*(MS\*MT)

Created: 26.09.2017 8:52 AM

Author: Poshtatska Katerina IO-51

\*/

//The specifications of main program Lab5

#include "stdafx.h"

#include "Data.h"

#include <iostream>

#include <cstdlib>

#include <omp.h>

using namespace std;

void TF1(void);

void TF2(void);

void TF3(void);

int n;

//Main method of Lab5

int main(void)

{

cout << "Lab5 started" << endl;

/\*

Variable n using for rhe size of all matrix and vectors in program.

data is an exeplar of class Data with parametr n.

\*/

n = 1000;

Data(n);

omp\_set\_num\_threads(3);

#pragma omp parallel

{

#pragma omp sections

{

#pragma omp section

{int i = omp\_get\_num\_threads();

cout << "Number of threads " << i << endl;

TF1();}

#pragma omp section

{TF2();}

#pragma omp section

{TF3();}

}

}

cout << "Lab5 finished" << endl;

system("pause");

}

//Thread function TF1

void TF1(void) {

cout << "TF1 started" << endl;

int\* A = Vector\_Input();

int\* B = Vector\_Input();

int\* C = Vector\_Input();

int\*\* MA = Matrix\_Input();

int\*\* ME = Matrix\_Input();

int\*\* MD = Func1(A, B, C, MA, ME);

if (n < 10) {

cout << "MD =" << endl;

Matrix\_Output(MD);

}

cout << "TF1 finished" << endl;

}

//Thread function TF2

void TF2(void) {

cout << "TF2 started" << endl;

int\*\* MG = Matrix\_Input();

int\*\* MH = Matrix\_Input();

int\*\* ML = Matrix\_Input();

int\*\* MK = Matrix\_Input();

int\*\* MF = Func2(MG, MK, ML, MH);

if (n < 10) {

cout << "MF =" << endl;

Matrix\_Output(MF);

}

cout << "TF2 finished" << endl;

}

//Thread function TF3

void TF3(void) {

cout << "TF3 started" << endl;

int\* O = Vector\_Input();

int\*\* MO = Matrix\_Input();

int\*\* MS = Matrix\_Input();

int\*\* MT = Matrix\_Input();

int\* S = Func3(O, MO, MS, MT);

if (n < 10) {

cout << "S = ";

Vector\_Output(S);

}

cout << "TF3 finished" << endl;

}

**Клас Data**

//Implementation of class called Data

#include "stdafx.h"

#include "Data.h"

#include <string>

#include <iostream>

using namespace std;

extern int n;

void Data(int size) {

n = size;

}

//Fill Vector with 1

int\* Vector\_Input() {

int\* A = new int[n];

#pragma omp parallel for

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

A[i] = 1;

}

return A;

}

//Print Vector on screen

void Vector\_Output(int\* A)

{

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

{

cout << A[i] << " ";

}

}

//Fill Matrix with 1

int\*\* Matrix\_Input()

{

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

#pragma omp parallel for

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

{

A[i] = new int[n];

#pragma omp parallel for

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

{

A[i][j] = 1;

}

}

return A;

}

//Print Matrix on screen

void Matrix\_Output(int\*\* A)

{

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

{

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

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

}

cout << endl;

}

}

//Adds the vector A to the vector B

int\* Vector\_Add(int\* A, int\* B, int\* C)

{

int\* D = new int[n];

#pragma omp parallel for

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

{

D[i] = A[i] + B[i] + C[i];

}

return D;

}

//Adds the matrix A to the matrix B

int\*\* Matrix\_Add(int\*\* A, int\*\* B)

{

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

#pragma omp parallel for

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

{

C[i] = new int[n];

#pragma omp parallel for

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

{

C[i][j] = A[i][j] + B[i][j];

}

}

return C;

}

//Subtracts the matrix B from the matrix A

int\*\* Matrix\_Multiply(int\*\* A, int\*\* B)

{

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

#pragma omp parallel for

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

{

C[i] = new int[n];

#pragma omp parallel for

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

{

C[i][j] = 0;

#pragma omp parallel for

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

{

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

}

}

}

return C;

}

//Multiply Vector on Matrix

int\* Matrix\_Vector\_Multiply(int\*\* A, int\* B)

{

int\* C = new int[n];

#pragma omp parallel for

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

{

C[i] = 0;

#pragma omp parallel for

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

{

C[i] += A[j][i] \* B[i];

}

}

return C;

}

//Multiply Value on Matrix

int\*\* Matrix\_Value\_Multiply(int\*\* A, int B)

{

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

#pragma omp parallel for

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

{

C[i] = new int[n];

#pragma omp parallel for

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

{

C[i][j] = A[j][i] \* B;

}

}

return C;

}

//Search for maximum element in Vector

int Vector\_Max(int\* A)

{

int B = A[0];

#pragma omp parallel for

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

{

if (A[i] > B)

{

B = A[i];

}

}

return B;

}

//Sorting Vector. Uses bubble sorting.

int\* Vector\_Sort(int\* A)

{

bool Finished;

int Temp = 0;

int\* B = new int[n];

#pragma omp parallel for

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

{

B[i] = A[i];

}

#pragma omp parallel for

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

Finished = true;

#pragma omp parallel for

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

if (B[j - 1] > B[j]) {

Temp = B[j - 1];

B[j - 1] = B[j];

B[j] = Temp;

Finished = false;

}

}

if (Finished) {

break;

}

}

return B;

}

/\*

Calculating Func1

MD=MAX(A+B+C)\*(MA\*ME)

\*/

int\*\* Func1(int\* A, int\* B, int\* C, int\*\* MA, int\*\* ME)

{

int\*\* MD = Matrix\_Value\_Multiply(Matrix\_Multiply(MA, ME), Vector\_Max(Vector\_Add(A, B, C)));

return MD;

}

/\*

Calculating Func2

MF=(MG+MH)\*(MK\*ML)\*(MG+ML)

\*/

int\*\* Func2(int\*\* MG, int\*\* MK, int\*\* ML, int\*\* MH)

{

int\*\* MF = Matrix\_Multiply(Matrix\_Multiply(Matrix\_Add(MG, MH), Matrix\_Multiply(MK, ML)), Matrix\_Add(MG, ML));

return MF;

}

/\*

Calculating Func3

S=SORT(O\*MO)\*(MS\*MT)

\*/

int\* Func3(int\* O, int\*\* MO, int\*\* MS, int\*\* MT)

{

int\* S = Matrix\_Vector\_Multiply(Matrix\_Multiply(MS, MT), Vector\_Sort(Matrix\_Vector\_Multiply(MO, O)));

return S;

}**Клас Data.h**

//Specification of class called data

#pragma once

void Data(int size);

int\* Vector\_Input();

void Vector\_Output(int\* A);

int\*\* Matrix\_Input();

void Matrix\_Output(int\*\* A);

int\*\* Func1(int\* A, int\* B, int\* C, int\*\* MA, int\*\* ME);

int\*\* Func2(int\*\* MG, int\*\* MH, int\*\* MK, int\*\* ML);

int\* Func3(int\* O, int\*\* MO, int\*\* MS, int\*\* MT);