Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 5

з дисципліни «Паралельне програмування»

на тему «Потоки в бібліотеці OpenMP»

ВИКОНАВ:

студент ІІІ курсу ФІОТ

групи ІО-51

Обозний Д.М.

Залікова – 5112

Варіант – 1.16, 2.30, 3.22

ПЕРЕВІРИВ:

Доцент, к.т.н.

Корочкін О.В.

Київ – 2017

**ЛАБОРАТОРНА РОБОТА №5**

**ПОТОКИ В БІБЛІОТЕЦІ OpenMP**

***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 варіанту.

***Варіант: 1.16 2.30 3.22***

***1.16 –*** d = ((A+B)\*(C\*(MA\*ME)))

***2.30 –*** f = MAX(MG\*MK)-MIN(ML+MH)

***1.16 –*** S = SORT(MS\*MT)\*O-P

Де d, f – змінні типу Integer;

A, B, C, O, P, S – вектори;

MA, ME, MG, MK, ML, MH, MS, MT – матриці;

***Лістинг програми***

***Клас Lab5***

/\*

Laboratory work

on the topic "Threads in OpenMP library"

Variant: 1.16 2.30 3.22

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

T2: f=MAX(MG\*MK)-MIN(ML+MH)

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

Created: 06.10.2017 8:52 PM

Author: Obozniy Dmitriy IO-51

\*/

//The specifications of main program Lab5

#include "stdafx.h"

#include "Data.h"

#include <windows.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;

int\* B;

int\* C;

int\*\* MA;

int\*\* ME;

if (n < 1)

{

A = Vector\_Input("A");

B = Vector\_Input("B");

C = Vector\_Input("C");

MA = Matrix\_Input("MA");

ME = Matrix\_Input("ME");

}

else {

A = Vector\_Input1();

B = Vector\_Input1();

C = Vector\_Input1();

MA = Matrix\_Input1();

ME = Matrix\_Input1();

}

Sleep(100);

int d = Func1(A, B, C, MA, ME);

Sleep(100);

if (n < 10) {

cout << "d = " << d << endl;

}

cout << "TF1 finished" << endl;

}

//Thread function TF2

void TF2(void) {

cout << "TF2 started" << endl;

int\*\* MG;

int\*\* MK;

int\*\* ML;

int\*\* MH;

if (n < 1)

{

MG = Matrix\_Input("MG");

MK = Matrix\_Input("MK");

ML = Matrix\_Input("ML");

MH = Matrix\_Input("MH");

}

else {

MG = Matrix\_Input1();

MK = Matrix\_Input1();

ML = Matrix\_Input1();

MH = Matrix\_Input1();

}

Sleep(100);

int f = Func2(MG, MK, ML, MH);

Sleep(100);

if (n < 10) {

cout << "f = " << f << endl;

}

cout << "TF2 finished" << endl;

}

//Thread function TF3

void TF3(void) {

cout << "TF3 started" << endl;

int\* O;

int\* P;

int\*\* MS;

int\*\* MT;

if (n < 1)

{

O = Vector\_Input("O");

P = Vector\_Input("P");

MS = Matrix\_Input("MS");

MT = Matrix\_Input("MT");

}

else {

O = Vector\_Input1();

P = Vector\_Input1();

MS = Matrix\_Input1();

MT = Matrix\_Input1();

}

Sleep(100);

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

Sleep(100);

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;

}

//Input Vector from keyboard

int\* Vector\_Input(char\* name) {

int\* A = new int[n];

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

cout << name << "[" << i << "] = ";

cin >> A[i];

cout << endl;

}

return A;

}

//Fill Vector with 1

int\* Vector\_Input1() {

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] << " ";

}

}

//Input Matrix from keyboard

int\*\* Matrix\_Input(char\* name)

{

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

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

{

A[i] = new int[n];

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

{

cout << name << "[" << i << "]" << "[" << j << "] = ";

cin >> A[i][j];

cout << endl;

}

}

return A;

}

//Fill Matrix with 1

int\*\* Matrix\_Input1()

{

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 = new int[n];

#pragma omp parallel for

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

{

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

}

return C;

}

//Subtracts the vector B from the vector A

int\* Vector\_Sub(int\* A, int\* B)

{

int\* C = new int[n];

#pragma omp parallel for

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

{

C[i] = A[i] - B[i];

}

return C;

}

//Multiply Vectors

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

{

int C = 0;

#pragma omp parallel for

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

{

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

}

return C;

}

//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;

}

//Search for minimum element in Matrix

int Matrix\_Min(int\*\* A)

{

int B = A[0][0];

#pragma omp parallel for

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

{

#pragma omp parallel for

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

{

if (A[i][j] < B)

{

B = A[i][j];

}

}

}

return B;

}

//Search for maximum element in Matrix

int Matrix\_Max(int\*\* A)

{

int B = A[0][0];

#pragma omp parallel for

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

{

#pragma omp parallel for

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

{

if (A[i][j] > B)

{

B = A[i][j];

}

}

}

return B;

}

//Sorting Matrix. Uses bubble sorting like sorting simple vector.

int\*\* Matrix\_Sort(int\*\* A)

{

bool Finished;

int Temp = 0;

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

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++)

{

B[Temp] = A[i][j];

Temp++;

}

}

#pragma omp parallel for

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

{

Finished = true;

#pragma omp parallel for

for (int j = 1; j < (n \* 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;

}

}

Temp = 0;

#pragma omp parallel for

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

{

#pragma omp parallel for

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

{

C[i][j] = B[Temp];

Temp++;

}

}

return C;

}

/\*

Calculating Func1

d=((A+B)\*(C\*(MA\*ME)))

\*/

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

{

int d = Vector\_Multiply(Vector\_Add(A, B), Matrix\_Vector\_Multiply(Matrix\_Multiply(MA, ME), C));

return d;

}

/\*

Calculating Func2

f=MAX(MG\*MK)-MIN(ML+MH)

\*/

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

{

int f = Matrix\_Max(Matrix\_Multiply(MG, MK)) - Matrix\_Min(Matrix\_Add(ML, MH));

return f;

}

/\*

Calculating Func3

S=SORT(MS\*MT)\*O-P

\*/

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

{

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

return S;

}

***Файл заголовків Data.h***

//Specification of class called data

#pragma once

void Data(int size);

int\* Vector\_Input(char\* name);

int\* Vector\_Input1();

void Vector\_Output(int\* A);

int\*\* Matrix\_Input(char\* name);

int\*\* Matrix\_Input1();

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

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

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

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