НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №6

з дісципліни **«**Паралельні та розподілені обчислення**»**

Виконав:

студент 3 курсу

ФІОТ гр. ІО-21

Кузьменко Володимир

Перевірив:

Корочкін О. В.

Київ – 2014 р.

ПОТОКИ В БИБЛИОТЕКЕ OpenMP

Цель работы: изучение средств библиотеки OpenMP для работы с задачами (процессами).

Выполнение работы: Разработать программу, содержащую п а р а л л е л ь н ы е потоки, каждый из которых реализует функцию F1, F2, F3 из лабораторной работы номер 1или 2. Требования к созданию потоков и необходимые исследования программы описаны в лабораторной работе 2.

Необходимые теоретические сведения: библиотека OpenMP обеспечивает возможность программирования параллельных процессов с помощью специальных директив. Теоретические сведения по программированию потоков и управлению ими в библиотеке ОpenMP можно найти в [1, 2, 4, 64].

Лістинг програми

#include "stdafx.h"

#include <iostream>

#include <algorithm>

#include <Windows.h>

#include <omp.h>

using namespace std;

/\*\*

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\* \*

\* Laboratory work #6. Threads with OpenMP \*

\* \*

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

\* F2: MC = SORT (MA+MB\*MM) \*

\* F3: p = MAX(SORT(MS)+MA\*MM \*

\* \*

\* author Kuzmenko Vladimir \*

\* group IO-21 \*

\* date 22.10.14 \*

\* \*

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\*/

const int n = 4000;

void inputVector(int[]);

void inputMatrix(int[][n]);

void outputVector(int[]);

void outputMatrix(int[][n]);

void sortMatrix(int[][n]);

void add(int[], int[], int[n]);

void add(int[][n], int[][n], int[][n]);

void mul(int[][n], int[][n], int[][n]);

void mul(int[], int[][n], int[]);

void maxMatr(int mas[][n], int p[]);

void f1(int a[], int b[], int c[], int ma[][n], int mz[][n], int e[]);

void f2(int ma[][n], int mb[][n], int mm[][n], int mc[][n]);

void f3(int ma[][n], int mm[][n], int ms[][n], int p[]);

void taskF1(void);

void taskF2(void);

void taskF3(void);

int main(void)

{

//поток Майстер

cout << "Lab6 started" << endl;

#pragma comment(linker, "/STACK:400000000")

#pragma omp parallel sections num\_threads(3)

{

#pragma omp section

{

taskF1();

}

#pragma omp section

{

taskF2();

}

#pragma omp section

{

taskF3();

}

}

cout << "Lab6 ended" << endl;

system("pause");

return 0;

}

void taskF1(){

cout << "Task F1 started" << endl;

int a[n], b[n], c[n], e[n];

int ma[n][n], mz[n][n];

inputVector(a);

inputVector(b);

inputVector(c);

inputMatrix(ma);

inputMatrix(mz);

f1(a, b, c, ma, mz, e);

cout << "F1=" << endl;

outputVector(e);

cout << "Task F1 finished" << endl;

}

void taskF2(){

cout << "Task F2 started" << endl;

int ma[n][n], mb[n][n], mm[n][n], mc[n][n];

inputMatrix(ma);

inputMatrix(mb);

inputMatrix(mm);

f2(ma, mb, mm, mc);

cout << "F2= " << endl;

outputMatrix(mc);

cout << "Task F2 finished" << endl;

}

void taskF3(){

cout << "Task F3 started" << endl;

int mm[n][n], ma[n][n], ms[n][n];

int p[1];

inputMatrix(ma);

inputMatrix(mm);

inputMatrix(ms);

f3(ma, mm, ms, p);

cout << "F3 = " << endl;

cout << p[0] << endl;

cout << "Task F3 finished" << endl;

}

void f1(int a[], int b[], int c[], int ma[][n], int mz[][n], int e[]){

int mamz[n][n];

int matrC[n];

int bc[n];

mul(ma, mz, mamz);

mul(c, mamz, matrC);

add(b, matrC, bc);

add(bc, a, e);

}

void f2(int ma[][n], int mb[][n], int mm[][n], int mc[][n]){

int mbmm[n][n];

mul(mb, mm, mbmm);

add(ma, mbmm, mc);

sortMatrix(mc);

}

void f3(int ma[][n], int mm[][n], int ms[][n], int p[]){

int mamm[n][n];

int mammms[n][n];

mul(ma, mm, mamm);

sortMatrix(ms);

add(ms, mamm, mammms);

maxMatr(mammms, p);

}

void inputVector(int v[n])

{

#pragma omp parallel for

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

{

v[i] = 1;

}

}

void inputMatrix(int m[n][n])

{

#pragma omp parallel for

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

{

#pragma omp parallel for

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

{

m[i][j] = 1;

}

}

}

void outputVector(int v[n])

{

if (n <= 10)

{

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

{

cout << v[i] << " ";

}

cout << endl;

}

}

void outputMatrix(int m[n][n])

{

if (n <= 10)

{

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

{

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

{

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

}

cout << endl;

}

}

}

void sortMatrix(int v[][n])

{

#pragma omp parallel for

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

sort(v[i], v[i] + n);

}

}

void add(int left[], int right[], int res[])

{

res[n];

{

#pragma omp parallel for

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

{

res[i] = left[i] + right[i];

}

}

}

void add(int left[][n], int right[][n], int res[][n]){

res[n][n];

{

#pragma omp parallel for

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

{

#pragma omp parallel for

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

res[i][j] = left[i][j] + right[i][j];

}

}

}

}

void mul(int left[][n], int right[][n], int res[][n])

{

res[n][n];

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

{

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

{

res[i][j] = 0;

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

{

res[i][j] = res[i][j] + left[i][k] \* right[k][j];

}

}

}

}

void mul(int left[], int right[][n], int res[])

{

res[n];

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

{

res[j] = 0;

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

{

res[j] = res[j] + left[k] \* right[k][j];

}

}

}

void maxMatr(int mas[][n], int p[]){

int m = mas[0][0];

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

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

if (mas[i][j]>m){

m = mas[i][j];

}

}

}

p[0] = m;

}