Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Лабораторна робота №7

з дисципліни «Паралельне програмування -1.

Основи паралельного програмування»

Тема: **ПОТОКИ В БІБЛІОТЕЦІ OpenMP**

Виконав

студент групи ІП-31

Кобилинський Дмитро

Київ 2015

**Цель работы:** изучение средств библиотеки OpenMP для работы с задачами.

**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 библио- теке ОpenMP можно найти в [1, 2, 4, 64].

// Lab7\_PP.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include <iostream>

#include <algorithm>

#include <Windows.h>

#include <omp.h>

using namespace std;

/\*\*

\* Parallel programming

\* Kobylynskiy Dmytro

\* IP-31

\* F1 = MIN(A + B) \* (B + C) \*(MA\*MD)

\* F2 = ML + MK \*MO

\* F3 = SORT(S + T)\*TRANS(MS\*MR)

\* 24.11.2015

\*/

const int n = 1000;

void inputVector(int[]);

void inputMatrix(int[][n]);

void outputVector(int[]);

void outputMatrix(int[][n]);

void sortVector(int v[]);

void transpose(int m1[][n], int result[][n]);

void sumVector(int add1[], int add2[], int result[]);

void multiplyMatrixOnVector(int m1[][n], int v1[], int res[n]);

void sumVector(int add1[], int add2[], int result[]);

void sumMatrix(int a[][n], int b[][n], int result[][n]);

void minVector(int mas[], int& p);

void multiplyVectorOnDigit(int v1[], int digit, int result[]);

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

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

void taskF1(void);

void taskF2(void);

void taskF3(void);

int main(void)

{

cout << "Lab 7 started" << endl;

#pragma comment(linker, "/STACK:40000000000000")

#pragma omp parallel sections num\_threads(3)

{

#pragma omp section

{

taskF1();

}

#pragma omp section

{

taskF2();

}

#pragma omp section

{

taskF3();

}

}

cout << "Lab 7 ended" << endl;

system("pause");

return 0;

}

void taskF1(){

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

int c;

int A[n], B[n], C[n], D[n], R[n], R1[n], R2[n];

int MA[n][n], MD[n][n], RES[n][n];

inputVector(A);

inputVector(B);

inputVector(C);

inputMatrix(MA);

inputMatrix(MD);

multiplyMatrix(MA, MD, RES);

sumVector(B, C, R);

multiplyMatrixOnVector(RES, R, R1);

sumVector(A, B, A);

minVector(A, c);

multiplyVectorOnDigit(R1, c, R2);

Sleep(400);

if (n <= 8){

cout << "F1 : " << endl;

outputVector(R2);

}

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

}

void taskF2(){

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

int mk[n][n], ml[n][n], mm[n][n], mo[n][n], mf[n][n], res[n][n];

int s = 0;

inputMatrix(mk);

inputMatrix(ml);

inputMatrix(mo);

multiplyMatrix(mk, mo, mf);

sumMatrix(mf, mk, res);

Sleep(200);

if (n <= 8){

cout << "F2:" << endl;

outputMatrix(res);

}

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

}

void taskF3(){

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

int MS[n][n], MR[n][n], RM1[n][n], mf[n][n], ml[n][n];

int S[n], T[n], R1[n], RES[n];

int s = 0;

inputMatrix(MS);

inputMatrix(MR);

inputVector(S);

inputVector(T);

sumVector(S, T, R1);

sortVector(R1);

multiplyMatrix(MS, MR, RM1);

transpose(RM1, mf);

multiplyMatrixOnVector(mf, R1, RES);

Sleep(600);

if (n <= 8){

cout << "F3:" << endl;

outputVector(RES);

}

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

}

void inputVector(int v[n]){

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

v[i] = 1;

}

}

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

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

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

m[i][j] = 1;

}

}

}

void outputVector(int v[n]){

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

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

}

cout << endl;

}

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

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

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

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

}

cout << endl;

}

}

void multiplyMatrixOnVector(int m1[][n], int v1[], int res[n]){

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

int s = 0;

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

s += m1[i][j] \* v1[j];

}

res[i] = s;

}

}

void multiplyMatrix(int left[][n], int right[][n], int res[][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 minVector(int mas[], int& p){

p = mas[0];

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

if (mas[i]<p){

p = mas[i];

}

}

}

void multiplyMatrixOnDigit(int m1[][n], int digit, int m2[][n]){

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

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

m2[i][j] = digit\*m1[i][j];

}

}

}

void sortVector(int v[]) {

int k = 0;

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

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

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

k = v[j];

v[j] = v[j + 1];

v[j + 1] = k;

}

}

}

}

void transpose(int m1[][n], int result[][n]){

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

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

result[i][j] = m1[j][i];

}

}

}

void sumMatrix(int a[][n], int b[][n], int result[][n]) {

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

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

result[i][j] = b[i][j] + a[i][j];

}

}

}

void sumVector(int add1[], int add2[], int result[]){

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

result[i] = add1[i] + add2[i];

}

}

void multiplyVectorOnDigit(int v1[], int digit, int result[]){

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

result[i] = digit\*v1[i];

}

}