Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Основи паралельного програмування

Лабораторна робота №6

**«Потоки в бібліотеці MPI»**

Виконала:

студентка групи ІВ-71

Молчанова В.С.

Перевірив:

Долголенко О.М.

Київ

2019 р.

**Завдання:**

Func1: D = (SORT(A + B) + C) \* (MA \* ME)

Func2: ML = SORT(MF + MG \* MH)

Func3: T = (O + P) \* (ML \* MS)

**Лістинг програми:**

**Lab6.cpp**

// Lab6.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include <iostream>

#include <algorithm>

#include <functional>

#include "mpi.h"

using namespace std;

// Lab 6: Molchanova Varvara, IV-71

// Func1: D = (SORT(A + B) + C) \* (MA \* ME)

// Func2: ML = SORT(MF + MG \* MH)

// Func3: T = (O + P) \* (ML \* MS)

const int n = 3;

int res1[n];

int res2[n][n];

int res3[n];

void thread\_function1();

void thread\_function2();

void thread\_function3();

void f1(int a[], int b[], int c[], int ma[][n], int me[][n], int res[]);

void f2(int mf[][n], int mg[][n], int mh[][n], int res[][n]);

void f3(int o[], int p[], int ml[][n], int ms[][n], int res[]);

void vector\_fill(int v[n], int value);

void matrix\_fill(int matrix[n][n], int value);

void vector\_output(int vector[n]);

void matrix\_output(int matrix[n][n]);

void add(int a[], int b[], int c[]);

void add(int ma[][n], int mb[][n], int c[][n]);

void mul(int a[], int ma[][n], int c[]);

void mul(int ma[][n], int mb[][n], int c[][n]);

void vector\_sort(int v[]);

void matrix\_sort(int matrix[][n]);

int main(int argc, char \*argv[])

{

int rank;

int size;

MPI\_Init(&argc, &argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

switch (rank) {

case 0:

thread\_function1();

break;

case 1:

thread\_function2();

break;

case 2:

thread\_function3();

break;

default:

break;

}

MPI\_Finalize();

system("pause");

return 0;

}

void thread\_function1()

{

cout << "Thread 1 started" << endl;

int a[n];

vector\_fill(a, 1);

int b[n];

vector\_fill(b, 1);

int c[n];

vector\_fill(c, 1);

int ma[n][n];

matrix\_fill(ma, 1);

int me[n][n];

matrix\_fill(me, 1);

f1(a, b, c, ma, me, res1);

cout << endl << "F1 = " << endl;

vector\_output(res1);

cout << "Thread 1 finished" << endl;

}

void thread\_function2()

{

cout << "Thread 2 started" << endl;

int mf[n][n];

matrix\_fill(mf, 1);

int mg[n][n];

matrix\_fill(mg, 1);

int mh[n][n];

matrix\_fill(mh, 1);

f2(mf, mg, mh, res2);

cout << endl << "F2 = " << endl;

matrix\_output(res2);

cout << "Thread 2 finished" << endl;

}

void thread\_function3()

{

cout << "Thread 3 started" << endl;

int o[n];

vector\_fill(o, 1);

int p[n];

vector\_fill(p, 1);

int ml[n][n];

matrix\_fill(ml, 1);

int ms[n][n];

matrix\_fill(ms, 1);

f3(o, p, ml, ms, res3);

cout << endl << "F3 = " << endl;

vector\_output(res3);

cout << "Thread 3 finished" << endl;

}

void f1(int a[], int b[], int c[], int ma[][n], int me[][n], int res[])

{

// Func1: D = (SORT(A + B) + C) \* (MA \* ME)

int m\_temp[n][n];

int v\_temp1[n];

int v\_temp2[n];

add(a, b, v\_temp1);

vector\_sort(v\_temp1);

add(v\_temp1, c, v\_temp2);

mul(ma, me, m\_temp);

mul(v\_temp2, m\_temp, res);

}

void f2(int mf[][n], int mg[][n], int mh[][n], int res[][n])

{

// Func2 : ML = SORT(MF + MG \* MH)

int m\_temp[n][n];

mul(mg, mh, m\_temp);

add(mf, m\_temp, res);

matrix\_sort(res);

}

void f3(int o[], int p[], int ml[][n], int ms[][n], int res[])

{

// Func3 : T = (O + P) \* (ML \* MS)

int m\_temp[n][n];

int v\_temp[n];

add(o, p, v\_temp);

mul(ml, ms, m\_temp);

mul(v\_temp, m\_temp, res);

}

void vector\_fill(int v[n], const int value)

{

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

{

v[i] = value;

}

}

void matrix\_fill(int matrix[n][n], const int value)

{

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

{

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

matrix[i][j] = value;

}

}

}

void vector\_output(int vector[n])

{

if (n <= 10)

{

cout << "|| ";

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

{

cout << vector[i] << " ";

}

cout << "||" << endl;

}

}

void matrix\_output(int matrix[n][n])

{

if (n <= 10)

{

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

{

cout << "|| ";

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

{

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

}

cout << "||" << endl;

}

}

}

void add(int ma[][n], int mb[][n], int c[][n])

{

c[n][n];

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

{

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

{

c[i][j] = ma[i][j] + mb[i][j];

}

}

}

void add(int a[], int b[], int c[])

{

c[n];

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

{

c[i] = a[i] + b[i];

}

}

void mul(int a[], int ma[][n], int c[])

{

c[n];

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

{

c[j] = 0;

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

{

c[j] += a[k] \* ma[k][j];

}

}

}

void mul(int ma[][n], int mb[][n], int c[][n])

{

c[n][n];

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

{

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

{

c[i][j] = 0;

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

{

c[i][j] += ma[i][k] \* mb[k][j];

}

}

}

}

void vector\_sort(int v[])

{

sort(v, v + n);

}

void matrix\_sort(int matrix[][n])

{

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

sort(matrix[i], matrix[i] + n, greater<int>());

}

}

**Результат роботи:**

Thread 3 started

F3 =

Thread 2 started

|| 18 18 18 ||

Thread 3 finished

F2 =

|| 4 4 4 ||

|| 4 4 4 ||

|| 4 4 4 ||

Thread 2 finished

Thread 1 started

F1 =

|| 27 27 27 ||

Thread 1 finished