**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

Разработка многопоточных приложений с использованием OpenMP. Вариант 3

**Исполнитель**

Студент группы БПИ191

Архаров Д. П.

Задание

Найти алгебраическое дополнение для каждого элемента матрицы. целое положительное число n, произвольная матрица А размерности n х n. Количество потоков является входным параметром, при этом размерность матриц может быть не кратна количеству потоков.

Решение:

Программа считает алгебраическое дополнение для каждого элемента матрицы, считая определитель минора через перестановки, а после умножая результат на (-1)^i+j, где i и j – позиция элемента в матрице. Таким образом итоговая формула: DOP = (-1)^i+j \* M(i,j), где M(i,j) – определитель минора составленного по элементу. Распараллеливается процесс подсчёта миноров разделяя один на много потоков, работая с глобальными переменными. Работа с общими для разных потоков переменными регулируется с помощью omp\_lock\_t. Перенос программы с std::threads на openMP прошёл без казусов, секция массива с потоками была заменена на секцию parallel.

Текст программы:

// Архаров Дмитрий Павлович

// БПИ191

// 3-ий Вариант.

#include <iostream>

#include <vector>

#include <algorithm>

#include <omp.h>

using namespace std;

//область с глобальными переменными

//

bool STOP = false; // Булевская переменная для того чтобы быть индикатором конца

int n; // Размер входной матрицы

int NumOfTHR; // кол-во потоков

long long DET; // Определитель матрицы считаемый многопоточно

long long DOP; // алгебраическое дополнение эллемента матрицы

omp\_lock\_t writelock; // Lock для контроля работы с потоками

vector<vector<int>> matIn; // Исходная матрица ввода

vector<vector<int>> matDop; // Матрица для поиска определителя

vector<int> PER; // Перестановка

//

int CountInv(int\* now) // Подсчитываем колличество инверсий

{

int invs = 0;

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

{

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

{

if (now[i] > now[j])

{

invs++;

}

}

}

return invs;

}

void THR\_Routine() // Метод используемый потоками

{

while (true) // Остановка по условию будет выполнятся с помощью return

{

// Используем lock чтобы залочить поток в рамках работы с глобальной перестановкой для индикатора конца

omp\_set\_lock(&writelock);

if (STOP)

{

omp\_unset\_lock(&writelock);

return;

}

if (!next\_permutation(PER.begin(), PER.end())) // Если мы просмотрели все перестановки, то

{

STOP = true; //Останавливаем вычисление

}

int\* nowPER = new int[n - 1];

copy(PER.begin(), PER.end(), nowPER);

omp\_unset\_lock(&writelock); // Снимаем Lock

int invs = CountInv(nowPER);

int slog = 1;

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

{

slog \*= matDop[i][nowPER[i]]; // считаем слогаемое для определителя

}

DET += invs % 2 ? -slog : slog; // добавляем слогаемое к определителю со знаком учитывающим кол-во инверсий

delete[] nowPER; // Удаляем массив во избежание утечки памяти

}

}

void FindDop(int x, int y) // Метод для поиска алгебраического дополнения для эллемента (x,y)

{

// Перезадаём определитель и изначальную подстановку для поиска определителя

DET = 0;

DOP = 0;

STOP = false;

PER.resize(n - 1);

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

{

PER[i] = i;

}

// Вырезаем из общей матрицы нужный нам минор

matDop.resize(n - 1);

int i1 = 0, j1 = 0;

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

{

if (i != x)

{

matDop[i1].resize(n - 1);

j1 = 0;

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

{

if (j != y)

{

matDop[i1][j1] = matIn[i][j];

j1++;

}

}

i1++;

}

}

omp\_init\_lock(&writelock); // инициализируем lock

// Считаем определитель для минора

#pragma omp parallel

{

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

{

THR\_Routine();

}

}

DOP = ((long long)pow(-1, x + y)) \* DET; // Алгебраическое дополнение эллемента (i,j) = (-1)^i+j \* M(i,j), где M(i,j) - минор матрицы

cout << "Алгебраическое дополнение эллемента " << "Mat[" << x << "][" << y << "] = " << DOP << '\n';

omp\_destroy\_lock(&writelock);

}

int main()

{

setlocale(LC\_ALL, "Russian");

//считываем нужные данные

while (n <= 0)

{

cout << "Введи размерность матрицы (больше 0) = ";

cin >> n;

}

while (NumOfTHR <= 0)

{

cout << "Введи кол-во потоков для работы (больше 0) = ";

cin >> NumOfTHR;

}

matIn.resize(n);

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

{

matIn[i].resize(n);

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

{

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

cin >> matIn[i][j];

}

}

//Выводим матрицу

cout << '\n' << "Введённая матрица:" << '\n';

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

{

cout << "(";

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

{

cout << (matIn[i][j] < 0 ? "" : " ") << matIn[i][j] << " ";

}

cout << ")" << '\n' << '\n';

}

//Выводим для каждого эллемента матрицы его алгебраическое дополнение

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

{

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

{

FindDop(i, j);

}

}

return 0;

}