Федеральное государственное автономное образовательное учреждение высшего

образования

**«НАЦИОНАЛЬНЫЙ** **ИССЛЕДОВАТЕЛЬСКИЙ** **УНИВЕРСИТЕТ** **«ВЫСШАЯШКОЛАЭКОНОМИКИ»** Факультет компьютерных наук

**ДОМАШНЯЯ РАБОТА №4**

Вычислить прямое произведение множеств А1, А2, А3, А4. Входные данные: множества чисел А1, А2, А3, А4, мощности множеств могут быть не равны между собой и мощность каждого множества больше или равна 1. Количество потоков является входным параметром.

по направлению подготовки Архитектура вычислительных систем образовательная программа «Программная инженерия»

Выполнил:

Ефимов Даниил Александрович, студент БПИ-195

Преподаватель:

Легалов Александр Иванович, Доктор технических наук, профессор

Москва 2020

**Задание**

Вычислить прямое произведение множеств А1, А2, А3, А4. Входные данные: множества чисел А1, А2, А3, А4, мощности множеств могут быть не равны между собой и мощность каждого множества больше или равна 1. Количество потоков является входным параметром.

Составление​​ программы

При составлении программы было принято использовать итеративный параллелизм в качестве модели построения приложения, так как в основном методе программы есть цикл for и данная модель нам будет удобна.

В данной программе в самом начале просится ввести длины множеств и главное, что длина не может быть меньше 1. Если пользователь правда введет меньше 1 программа выведет сообщение о проблеме и остановится. Эти длины дальше используются в методах генерации множеств. Для 4-ех множеств вызывается 4 метода. А сама генерация происходит посредством метода rand().

Далее программа просит ввести пользователя количество потоков, используемых при работе, число также не может быть меньше 1, так как один поток должен быть всегда. После чего через метод библиотеки opn.h присваиваем omp\_set\_num\_threads количество потоков. А перед циклом for напишем строку #pragma omp for для того чтобы OpenMp сам разграничил цикл по потокам.

Завершением является вывод в консоль полученных элементов, а точнее соединенных элементов из 4 множеств. Также в начале в выводе мы можем увидеть какие итерации были присвоены каким потокам.

Методы:

Method – метод прямого произведения 4-ех множеств.

generateVec – метод генерации множества, а точнее вектора с случайными числами.

Текст программы:

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include <thread>

#include <mutex>

#include <omp.h>

using namespace std;

/// <summary>

/// Метод прямого произведения 4-ех множеств

/// </summary>

/// <param name="left">левая граница</param>

/// <param name="right">правая граница</param>

/// <param name="A1">первое множество</param>

/// <param name="A2">второе множество</param>

/// <param name="A3">третье множество</param>

/// <param name="A4">четвертое множество</param>

/// <param name="decard">вывод</param>

void Method(int numberofThreads, std::vector<int> A1, std::vector<int> A2, std::vector<int> A3, std::vector<int> A4, vector<vector<int>>& decard) {

omp\_set\_num\_threads(numberofThreads);

#pragma omp parallel

{

#pragma omp for

for (int i = 0; i < A1.size(); i++)

{

#pragma omp critical

{

cout << "Thread(" << omp\_get\_thread\_num() << ") Index:" << to\_string(i) << endl;

for (int j = 0; j < A2.size(); j++)

{

for (int k = 0; k < A3.size(); k++)

{

for (int n = 0; n < A4.size(); n++)

{

decard.push\_back(vector<int>{A1[i], A2[j], A3[k], A4[n]});

}

}

}

}

}

}

}

/// <summary>

/// Метод генерации множества

/// </summary>

/// <param name="length">длина множества</param>

/// <returns>сгенерированное множество</returns>

vector<int> generateVec(int length) { //Метод генерации вектора

vector<int> Vector;

srand(time(NULL));

cout << "vector: ";

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

int num = rand() % 10;

Vector.push\_back(num);

cout << Vector[i] << " ";

}

cout << endl;

return Vector;

}

int main()

{

int length1, length2, length3, length4, numberofThreads;

string numstr;

vector<vector<int>> decard; //Переменная для вывода

cout << "Enter length of A1: ";

cin >> numstr; //Ввод длины первого множества

length1 = stoi(numstr); //Парсим в инт

if (length1 < 1) { //Проверка на длину

cout << "Length cannot be less than 1";

return 0;

}

std::vector<int> A1 = generateVec(length1); //генерация множества

cout << "Enter length of A2: ";

cin >> numstr; //Ввод длины второго множества

length2 = stoi(numstr); //Парсим в инт

if (length2 < 1) { //Проверка на длину

cout << "Length cannot be less than 1";

return 0;

}

std::vector<int> A2 = generateVec(length2); //генерация множества

cout << "Enter length of A3: ";

cin >> numstr; //Ввод длины третьего множества

length3 = stoi(numstr); //Парсим в инт

if (length3 < 1) { //Проверка на длину

cout << "Length cannot be less than 1";

return 0;

}

std::vector<int> A3 = generateVec(length3); //генерация множества

cout << "Enter length of A4: ";

cin >> numstr; //Ввод длины четвертого множества

length4 = stoi(numstr); //Парсим в инт

if (length4 < 1) { //Проверка на длину

cout << "Length cannot be less than 1";

return 0;

}

std::vector<int> A4 = generateVec(length4); //генерация множества

cout << "Enter number of threads: ";

cin >> numstr; //Ввод количества потоков

numberofThreads = stoi(numstr); //Парсим в инт

if (numberofThreads < 1) { //Проверка на количество потоков

cout << "There cannot be the number of threads less than 1";

return 0;

}

/\*if (length1 < numberofThreads) {

numberofThreads = length1;

}\*/

Method(numberofThreads, A1, A2, A3, A4, decard); //Главный поток

cout << endl << "output: {"; //Вывод

for (size\_t i = 0; i < decard.size(); i++)

{

cout << "(";

for (size\_t j = 0; j < decard[i].size(); j++)

{

if (j != decard[i].size() - 1) {

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

}

else {

cout << decard[i][j];

}

}

if (i != decard.size() - 1) {

cout << "), ";

}

else {

cout << ")";

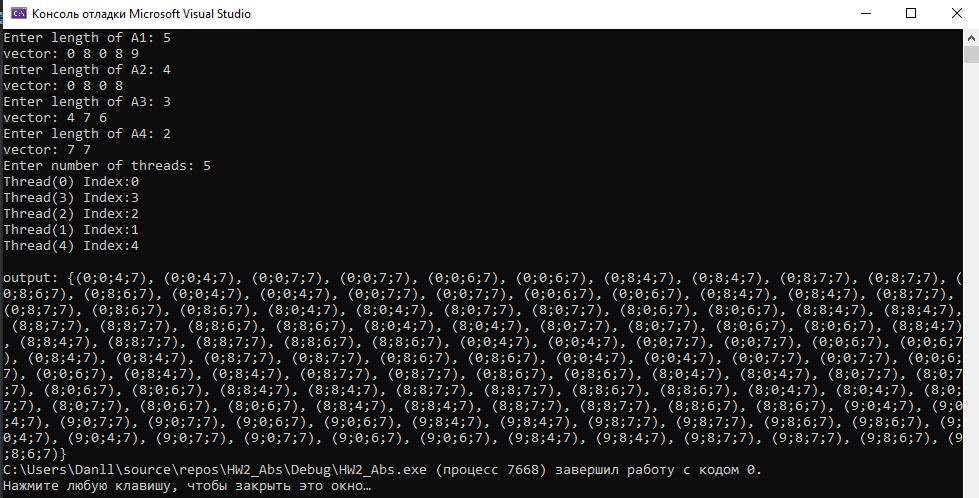
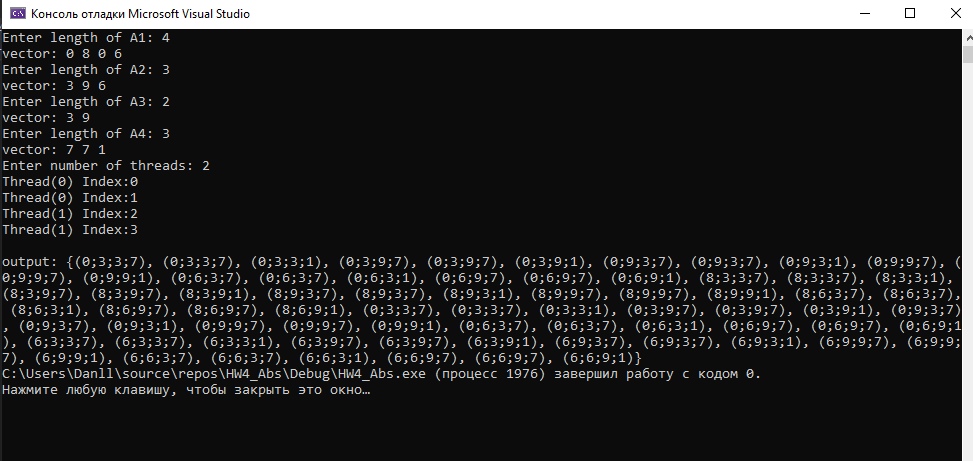
}

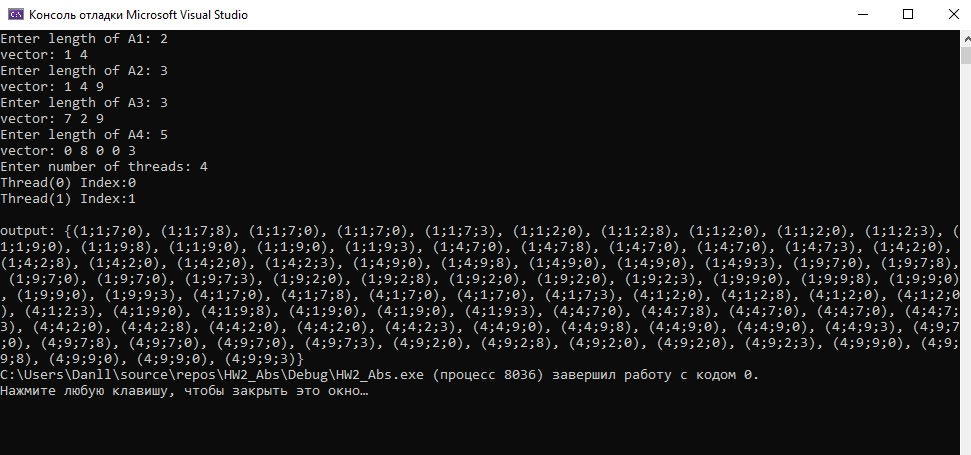
}

cout << "}";

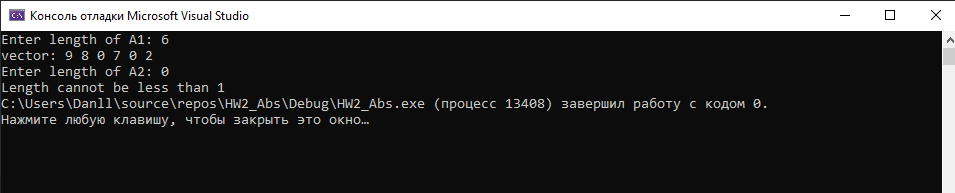
}

Тестирование программы

Проверим программу на нормальных входных данных: 



Теперь введем некорректное количество элементов множества, как например 0:



Как мы видим: вывелось сообщение и программа остановилась. Далее введем некорректное количество потоков:

