Отчет по лабораторной работе №3

«Программирование для многопоточных приложений. OpenMP»

по дисциплине  
«ЭВМ и периферические устройства»

|  |  |
| --- | --- |
| Выполнили:  студенты гр. 3530904/80004 | Захарова А. А.  Федоров Д. В.  Селезнев В. А. |
| Преподаватель | А. В. Петров |
|  |  |

«\_\_\_»\_\_\_\_\_\_\_\_\_\_2020 г.

**Задание на выполнение работы**

1. Загрузите тестовую программу. Не забудьте активизировать директивы OpenMP на страницах свойств проекта, выбрав Configuration Properties, C/C++, Language и изменив значение свойства OpenMP Support.

2. Экспериментально подтвердите параллельную работу **двух потоков** при использовании директив *for, sections*. Подтвердите, используя директиву *OMP\_GET\_NUM\_*THREADS. Объясните и покажите использование директивы *barrier*.

3. Напишите многопотоковую программу с применением двух директив (single, critical, master, atomic, ordered… ) синхронизации потоков (директивы синхронизации у студентов должны отличаться). Программа может выводить на экран номер потока или заполнять массив из двух потоков с использованием директив синхронизации.

4. Напишите многопотоковую программу с применением синхронизации на базе замков *OMP\_INIT\_LOCK(var), OMP\_SET\_LOCK, OMP\_UNSET\_LOCK, OMP\_TEST\_LOCK.*

**Код программы**

#include <omp.h>

#include <iostream>

#include <windows.h>

#include <stdio.h>

#include <conio.h>

using namespace std;

void func()

{

for (int i = 0; i < 500000; i++)

{

rand();

}

}

int main()

{

omp\_set\_num\_threads(4);

const int size = 100;

int array[size];

#pragma omp parallel num\_threads(2)

{

#pragma omp critical

{

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

{

array[i] = i + 1;

cout << array[i] << " ";

}

}

}

cout << "\n";

#pragma omp parallel

{

#pragma omp for

for (int i = 0; i < 100; i++)

{

cout << 0;

func();

}

#pragma omp single

cout << "\n" << omp\_get\_num\_threads() << " threads\n";

}

#pragma omp parallel

{

#pragma omp sections nowait

{

#pragma omp section

for (int i = 0; i < 10; i++)

{

cout << 1; func();

}

#pragma omp section

for (int i = 0; i < 20; i++)

{

cout << 2; func();

}

}

#pragma omp barrier

for (int i = 0; i < 10; i++)

{

cout << 3;

func();

}

}

cout << "\n\n\n";

omp\_lock\_t lock;

omp\_init\_lock(&lock);

int a[10][10];

#pragma omp parallel num\_threads(2)

{

for (int i = 0; i < 10; i++)

{

omp\_set\_lock(&lock);

for (int j = 0; j < 10; j++)

{

a[i][j] = i;

cout << a[i][j] << ' ';

}

cout << "\n";

omp\_unset\_lock(&lock);

}

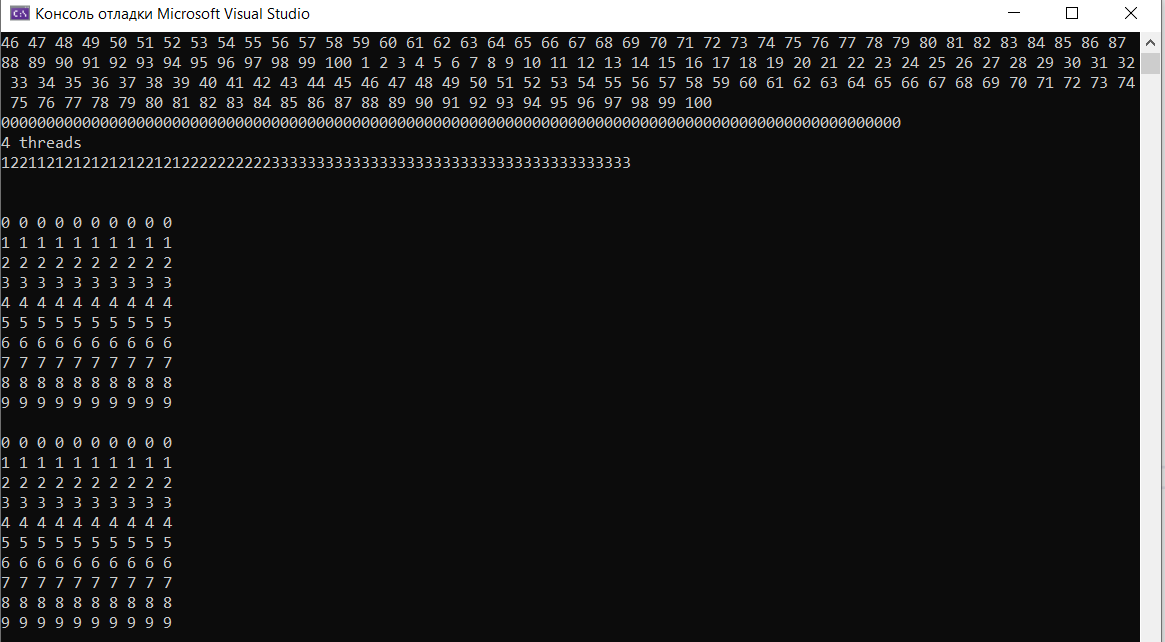
cout << "\n";

}

omp\_destroy\_lock(&lock);

}

**Результат работы программы**

****