ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ**

**ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

(НИУ «БелГУ»)

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

**Отчет по лабораторной работе 3**

**по дисциплине: «Модели параллельного программирования»**

**Тема работы «Решение практических задач с применением технологии OpenMP»**

студента очного отделения

2 курса 12001801 группы

Капустина Виктора Сергеевича

Проверил(а):

Петров Денис Васильевич

Белгород 2020

**Цель:** Изучить опции директив OpenMP. Получить практический навык использования технологии OpenMP при решении прикладных задач.

**Индивидуальное задание. 17 Вариант:** Дана последовательность арифметических выражений, операндами которых являются однозначные числа, а число операций не больше двух. Найти значения всех выражений.

**Теоретическая часть.**

В OpenMP предусмотрены следующие конструкции синхронизации:

· critical – критическая секция

· atomic – атомарность операции

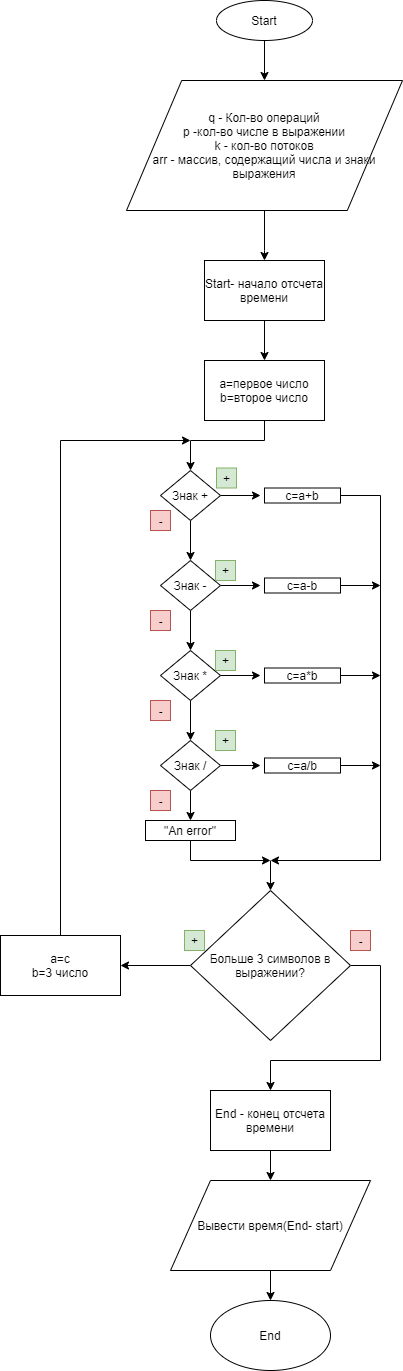
· barrier – точка синхронизации

· master – блок, который будет выполнен только основным потоком. Все остальные потоки пропустят этот блок. В конце блока неявной синхронизации нет.

· ordered – выполнять блок в заданной последовательности

· flush – немедленный сброс значений разделяемых переменных в память.

**Блок-схема**

**  
Ход работы**

**Программа на MS VS.**

**Листинг 1. Код программы на MS VS в тестовом режиме**

#include<iostream>

#include<stdio.h>

#include<ctime>

#include<omp.h>

using namespace std;

int main()

{

int i;

setlocale(LC\_ALL, "Rus");

int q,p;

cout << "Введите кол-во арифметических операций " ;

cin >> q;

cout << endl;

int \*\*arr = new int\*[q];

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

{

arr[i] = new int[7];

}

//заполнение

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

{

cout << "Введите кол-во чисел в выражении ( от 2 до 3) ";

cin >> p;

cout << endl;

arr[i][0] = p \* 2 - 1;

for (int j = 1; j <= arr[i][0]; j++)

{

switch (j%2)

{

case 1:

cout << "Введите " << (j + 1) / 2 << " число от 0 до 9 " ;

cin >> arr[i][j];

cout << endl;

break;

case 0:

cout << "Введите номер знака" << endl;

cout << " + -- 1 " << endl;

cout << " - -- 2 " << endl;

cout << " \* -- 3 " << endl;

cout << " / -- 4 " << endl;

cin >> arr[i][j];

break;

default:

cout << "Не выполнены условия";

break;

}

}

}

//расчет

cout << "Введите кол-во потоков ";

int k;

cin >> k;

cout << endl;

omp\_set\_num\_threads(k);

double start = omp\_get\_wtime(); //начало отсчёта

#pragma omp parallel shared(arr,q) private(i)

{

#pragma omp for schedule(dynamic, 2) nowait

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

{

int a = arr[i][1];

int b = arr[i][3];

int c;

switch (arr[i][2])

{

case 1: c = a + b; break;

case 2: c = a - b; break;

case 3: c = a \* b; break;

case 4: c = a / b; break;

default:

cout << "An Error" << endl;

break;

}

if (arr[i][0] > 3)

{

a = c;

b = arr[i][5];

switch (arr[i][4])

{

case 1: c = a + b; break;

case 2: c = a - b; break;

case 3: c = a \* b; break;

case 4: c = a / b; break;

default:

cout << "An Error" << endl;

break;

}

}

arr[i][6] = c;

}}

double end = omp\_get\_wtime();

cout << "Время: " << (end - start) \* 1000 << endl;

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

{

char xc;

switch (arr[i][2])

{

case 1: xc = '+'; break;

case 2: xc = '-'; break;

case 3: xc = '\*'; break;

case 4: xc = ':'; break;

}

cout << arr[i][1] << xc << arr[i][3];

if (arr[i][0] > 3) {

char sc;

switch (arr[i][4])

{

case 1:sc = '+'; break;

case 2:sc = '-'; break;

case 3:sc = '\*'; break;

case 4:sc = ':'; break;

}

cout << sc << arr[i][5];

} cout << "=" << arr[i][6] << endl;

}

//cout << " Время работы " << time;

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

delete[] arr[i];

}

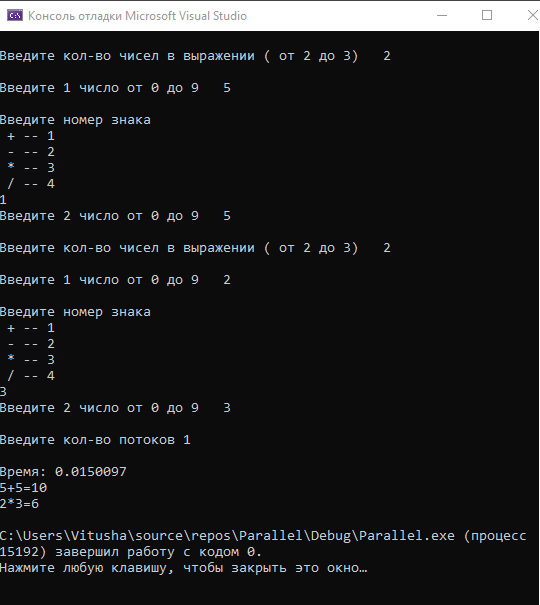
delete[] arr;

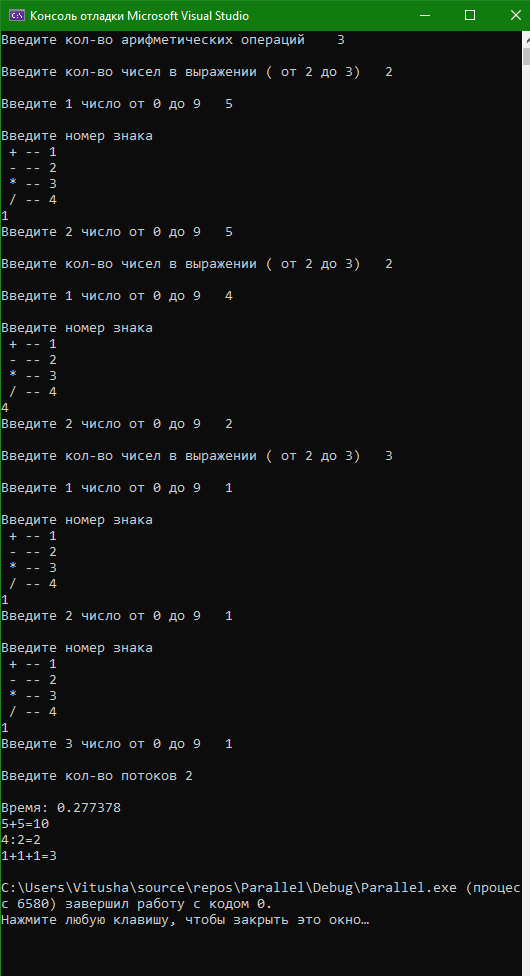
return 0;

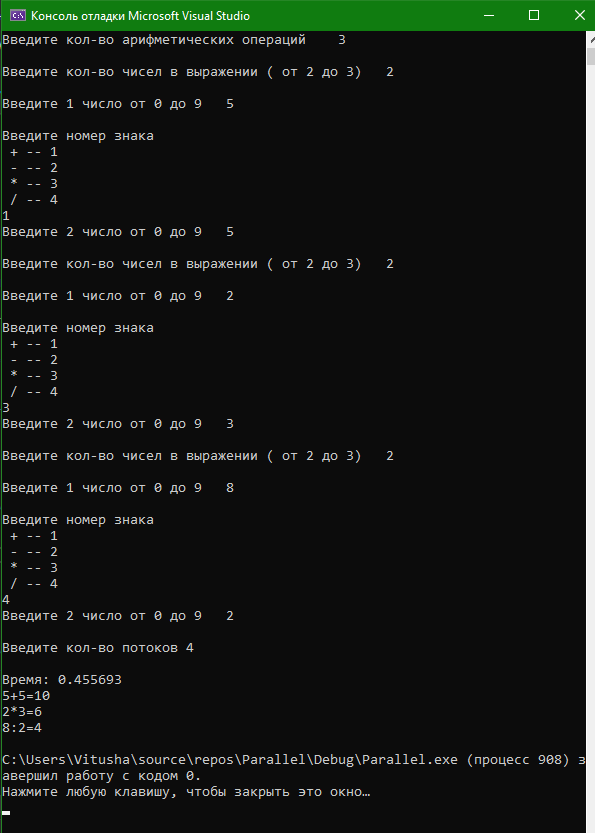
}

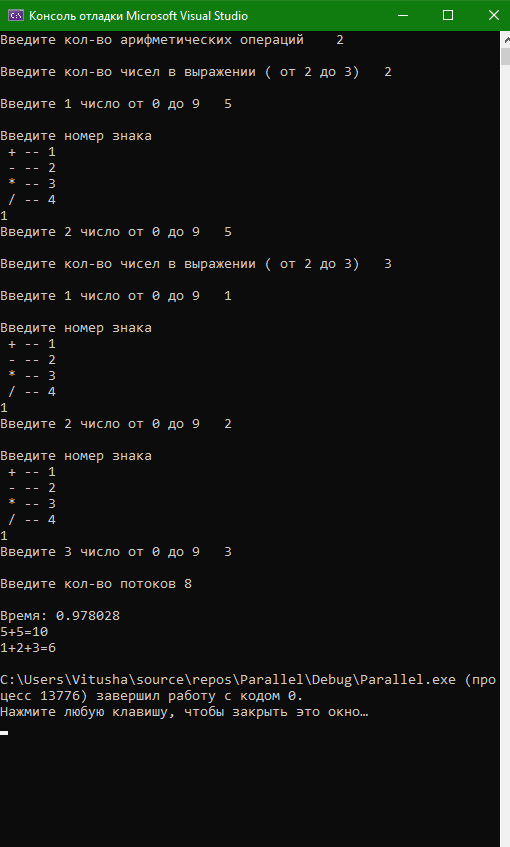
**Конец листинга 1**

**Скриншоты работы программы в тестовом режиме.**

  
**Скриншот 1**

  
**Скриншот 2**

  
**Скриншот 3**

**  
Скриншот 4**

**Листинг 2. Код программы на MS VS в расчетном режиме**#include<iostream>

#include<stdio.h>

#include<ctime>

#include<omp.h>

using namespace std;

int main()

{

srand(time(0));

int i;

setlocale(LC\_ALL, "Rus");

int q,p;

cout << "Введите кол-во арифметических операций " ;

cin >> q;

cout << endl;

int \*\*arr = new int\*[q];

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

{

arr[i] = new int[7];

}

cout << "Введите кол-во потоков ";

int k;

cin >> k;

cout << endl;

omp\_set\_num\_threads(k);

double start = omp\_get\_wtime(); //начало отсчёта

#pragma omp parallel shared(arr,q) private(i)

{

#pragma omp for schedule(dynamic, 2) nowait

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

{

p = 2 + rand() % 2;

arr[i][0] = p \* 2 - 1;

for (int j = 1; j <= arr[i][0]; j++)

{

switch (j % 2)

{

case 1:

arr[i][j] = 1 + rand() % 8;

break;

case 0:

arr[i][j] = 1 + rand() % 4;

break;

default:

cout << "Не выполнены условия";

break;

}

}

int a = arr[i][1];

int b = arr[i][3];

int c;

switch (arr[i][2])

{

case 1: c = a + b; break;

case 2: c = a - b; break;

case 3: c = a \* b; break;

case 4: c = a / b; break;

default:

cout << "An Error" << endl;

break;

}

if (arr[i][0] > 3)

{

a = c;

b = arr[i][5];

switch (arr[i][4])

{

case 1: c = a + b; break;

case 2: c = a - b; break;

case 3: c = a \* b; break;

case 4: c = a / b; break;

default:

cout << "An Error" << endl;

break;

}

}

arr[i][6] = c;

}}

double end = omp\_get\_wtime();

cout << "Время: " << (end - start) \* 1000 << endl;

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

delete[] arr[i];

}

delete[] arr;

return 0;

}

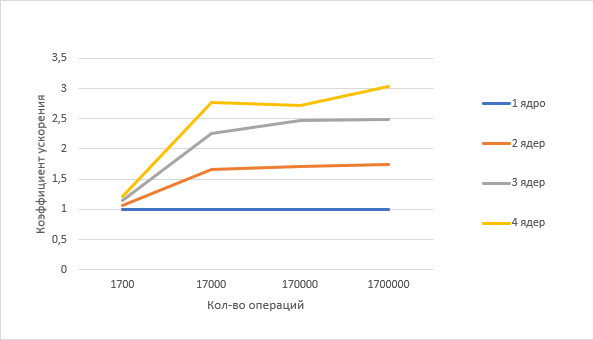
**Конец листинга 2**

**Таблица 1. Время подсчета элементов на MS VS**

| № | Кол-во операций | Время расчета на 1 ядре, ms | Время расчета на 2 ядрах, ms | Время расчета на 3 ядрах, ms | Время расчета на 4 ядрах, ms |
| --- | --- | --- | --- | --- | --- |
|
| 1 | 1700 | 0.6851 | 0.6432 | 0.5987 | 0.5658 |
| 2 | 17000 | 6.999 | 4.1985 | 3.0973 | 2.527 |
| 3 | 170000 | 68.1717 | 39.9037 | 27.5586 | 25.1021 |
| 4 | 1700000 | 675.89 | 387.26 | 271.024 | 221.766 |

**Таблица 2. Коэффициент ускорения на MS VS**

| № | Кол-во операций | Коэффициент ускорения на 1 ядре | Коэффициент ускорения на 2 ядрах | Коэффициент ускорения на 3 ядрах | Коэффициент ускорения на 4 ядрах |
| --- | --- | --- | --- | --- | --- |
|
| 1 | 1700 | 1 | 1,065 | 1,144 | 1,210 |
| 2 | 17000 | 1 | 1,667 | 2,259 | 2,769 |
| 3 | 170000 | 1 | 1,708 | 2,473 | 2,715 |
| 4 | 1700000 | 1 | 1,745 | 2,493 | 3,04 |



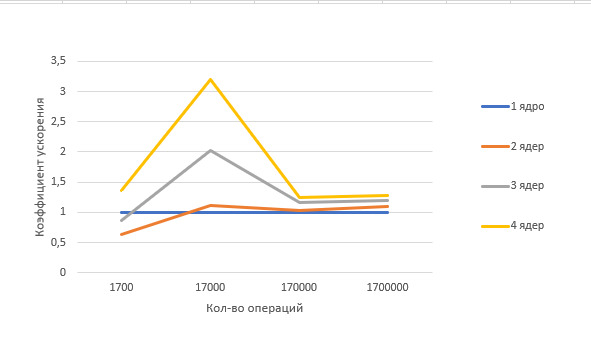
**График 1. График зависимости коэффициента ускорения от кол-ва операций на разном кол-ве ядер в MS VS**

**Таблица 3. Время подсчета элементов на Кластере**

| № | Кол-во операций | Время расчета на 1 ядре, ms | Время расчета на 2 ядрах, ms | Время расчета на 3 ядрах, ms | Время расчета на 4 ядрах, ms |
| --- | --- | --- | --- | --- | --- |
|
| 1 | 1700 | 1.5039 | 2.3957 | 1.7526 | 1,1114 |
| 2 | 17000 | 46,3493 | 41,8648 | 22,9761 | 14,5240 |
| 3 | 170000 | 356,93 | 349,177 | 305,947 | 245,059 |
| 4 | 1700000 | 2937,03 | 2695,36 | 2468,63 | 2301,06 |

**Таблица 2. Коэффициент ускорения на Кластере**

| № | Кол-во операций | Коэффициент ускорения на 1 ядре | Коэффициент ускорения на 2 ядрах | Коэффициент ускорения на 3 ядрах | Коэффициент ускорения на 4 ядрах |
| --- | --- | --- | --- | --- | --- |
|
| 1 | 1700 | 1 | 0,627 | 0,858 | 1,353 |
| 2 | 17000 | 1 | 1,107 | 2,0172 | 3,191 |
| 3 | 170000 | 1 | 1,022 | 1,166 | 1,249 |
| 4 | 1700000 | 1 | 1,089 | 1,189 | 1,276 |

  
**График 2. График зависимости коэффициента ускорения от кол-ва операций на разном кол-ве ядер на кластере**

**Вывод:** В данной лабораторной работе лучше изучил открытый стандарт для распараллеливания программ OpenMP. Использовал полученные знания для реализации расчета арифметических выражений.