МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение   
высшего профессионального образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №14

РАБОТА ЗАЩИЩЕНА С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| старший преподаватель |  |  |  | А. Ю. Сыщиков |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1  OpenMP, Распараллеливание операций над векторами |
|  |
| по дисциплине: [Системы с параллельной обработкой информации](https://pro.guap.ru/inside_s#subjects/2436975) |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 1742 |  |  |  |  |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

1. **Цель работы**

Реализовать и распараллелить с помощью технологии OpenMP различные операции над векторами размерности N.

|  |  |  |  |
| --- | --- | --- | --- |
| № варианта | N | Тип элемента вектора | Тип операции |
| 2 | 7 | Знаковый короткий целый | Сложение векторов (a[i]=b[i]+c[i]) |

1. **Текст программы**

/\*

omp\_set\_nested(1); разрешает вложенную параллельность

omp\_set\_nested(0); запрещает вложенную параллельность (по умолчанию)

omp\_set\_num\_threads(2); разрешает использовать только 2 процесса в последуюзих областях

omp\_set\_dynamic(1); разрешает динамическое распределение (автоматическое), запуститься максимум ветвей (16)

omp\_set\_dynamic(0); запрещает динамическое распределение (автоматическое), будет запускаться количество ветвей оределенных пользователей, по умолчанию 1

omp\_get\_thread\_num(); возвращает номер нити

omp\_get\_num\_threads() возвращает Всего нитей в области

omp\_get\_max\_threads() возвращает максимально допустимое число нитей для использования в следующей параллельной области

omp\_get\_num\_procs возвращает количество процессоров, доступных для использования программе пользователя на момент вызова(изменчиво)

omp\_in\_parallel() возвращает 1 если она была вызвана из активной параллельной области программы, если 0 - то последовательная область

setlocale(0, "");

omp\_get\_wtime(); вернет значение времени в секундах

omp\_get\_wtick(); вернет точность таймера в секундах

#pragma omp parallel -- распаралллить Параллельная область

#pragma omp parallel reduction (+: count) -- локальная count для каждой ветки. Оператор + означате, что по завершению всей области будет соверешнно суммирование всех локальных count, поэтому по умолчанию count =0

reduction(оператор:список) – задаёт оператор и список общих пе-

ременных; для каждой переменной создаются локальные копии в каж-

дой нити; локальные копии инициализируются соответственно типу

оператора (для аддитивных операций – 0 или его аналоги, для мульти-

пликативных операций – 1 или её аналоги); над локальными копиями

переменных после выполнения всех операторов параллельной области

выполняется заданный оператор; оператор это: для языка Си – +, \*, -,

&, |, ^, &&, ||, для языка Фортран – +, \*, -, .and., .or., .eqv., .neqv.,

max, min, iand, ior, ieor; порядок выполнения операторов не опреде-

лён, поэтому результат может отличаться от запуска к запуску.

#pragma omp parallel num\_threads(3) разрешает в этой области использовать 3 процесса

#pragma omp master позволяет обеспечить печать только процессом-мастером

#pragma omp single Одна нить будет выполнять данный фрагмент, а все остальные нити будут ожидать завершения её работы

#pragma omp single nowait Одна нить будет выполнять данный фрагмент, а все остальные следующие действия, не дожидаясь ее завершения

#сложнаpragma omp single copyprivate(n) - после выпонения нити, новое значение n будет использоваться во всех нитях до и после в нашей области.

#pragma omp master выделяют участок кода, который будет выполнен только нитью-мастером. Синхронизации нет.

shared - общие переменные

private - локальные данные

shared - данные перед входом в область

private - порожденные в области

Исключение составляют

переменные, являющиеся счетчиками итераций в цикле, по очевидным при-

чинам

firstprivate - создается лок переменная, инициализация автоматическая - предыдущее значение, из общего потока

private - создается лок переменная, инициализация не происходит.

#pragma omp for

Итеративная переменная неявно

делается локальной при входе в цикл. После завершения цикла значение ите-

ративной переменной цикла не определено, если она не указана в опции

lastprivate.

\*/

#include <omp.h>

#include <time.h>

#include <locale.h>

#include <iostream>

using namespace std;

constexpr auto N = 7;

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

{

setlocale(0, "");

short int A[N], B[N], C[N];

int i, n;

/\* Заполним исходные массивы/ Массив А пустой, так как в него записывать будем сумма массивов B and C\*/

for (i = 0; i < N; i++) { A[i] = 0; B[i] = rand() % 100; C[i] = rand() % 100; }

cout << "A:";

for (i = 0; i < N; i++) { cout << " " << A[i]; }

cout << endl;

cout << "B:";

for (i = 0; i < N; i++) { cout << " " << B[i]; }

cout << endl;

cout << "C:";

for (i = 0; i < N; i++) { cout << " " << C[i]; }

cout << endl;

//паралельная область, общие данные A B C индивидуальные данные i and n

#pragma omp parallel shared(A, B, C) private(i, n)

{

/\* Получим номер текущей нити \*/

n = omp\_get\_thread\_num();// у каждой нити он свой

#pragma omp for //этот фор автоматически раздели все иттерации между всем нитями

for (i = 0; i < N; i++)

{

A[i] = B[i] + C[i];

printf("Нить %d сложила элемент B[%d] с элементом С[%d] и записала результат %u в элемент A[%d]\n", n, i, i, A[i], i);

}

}

cout << "A:";

for (i = 0; i < N; i++) { cout << " " << A[i]; }

}

1. **Результат работы программы**

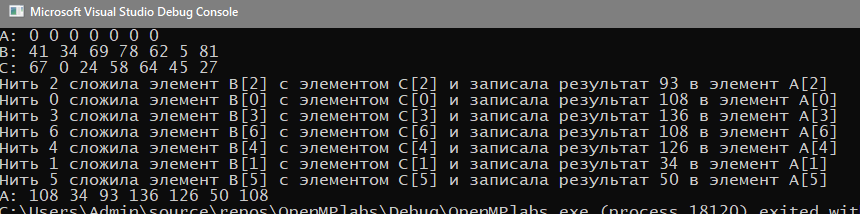


Рисунок 1. Результат работы программы.