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 |
| 12 | 7 | Знаковый короткий целый | Двоичное искл. ИЛИ |

1. **Текст программы**

#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 A[N], B[N];

int i, n;

/\*Рандом\*/

cout << "A:";

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

A[i] = rand() % 10;

cout << " " << A[i];

}

cout << endl;

/\*вывод в двоичной СС\*/

for (int j = 0; j < N; j++) {

for (i = sizeof(A[j]) \* 8 - 1; i >= 0; --i)

{

cout << (int)((A[j] >> i) & 1);

}

cout << " "<< endl;

}

cout << endl;

/\*Рандом + смещение\*/

cout << "B:";

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

B[i] = rand() % 10 +(i%10);

cout << " " << B[i];

}

cout << endl;

/\*вывод в двоичной СС\*/

for (int j = 0; j < N; j++) {

for (i = sizeof(B[j]) \* 8 - 1; i >= 0; --i)

{

cout << (int)((B[j] >> i) & 1);

}

cout << " " << endl;

}

cout << endl;

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

#pragma omp parallel shared(A) 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++)

{

/\*По заданию не сказано кол-во массивов. Случай если 1. Ответ всегда 0, так как XOR самого себя - это обуление\*/

//A[i] ^= A[i];

//A[i] = ~(A[i] & A[i]) & (A[i] | A[i]);

//printf("Нить %d использовала элемент А[%d]\n", n, i);

/\*По заданию не сказано кол-во массивов. Случай если 2.\*/

A[i] = A[i]^B[i];

//A[i] = ~(A[i] & B[i]) & (A[i] | B[i]);

printf("Нить %d использовала элемент А[%d] и элемент B[%d]\n", n, i, i);

}

}

cout << "A:";

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

}

1. **Результат работы программы**

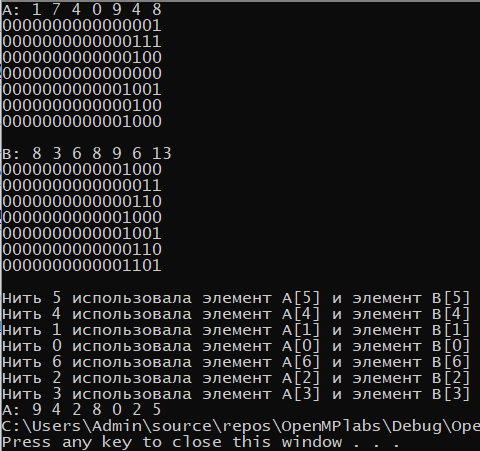


Рисунок 1. Результат работы программы.