**Лабораторная работа №3**

**Использование редукции**

**Цель:** изучить основные особенности использования редукции в OpenMP на примере использования в рамках языка С++.

**Лабораторные задания** (№ варианта = № компьютера%5)

**Задание 1.** В соответствии с вариантом задания, написать на C++ программу, реализующую многопоточность на основе технологии OpenMP: она должна вычислять значение функции с помощью редукции. Измерять время работы программ для различных значений параметра N. **Результаты занести в отчёт.**

|  |  |
| --- | --- |
| **Вариант** | **Задание: написать программы, демонстрирующие работу следующей функции:** |
| **3** | На основе трёх равно размерных массивов A, B и C (длины N) функция возвращает произведение ненулевых значений, полученных таким образом: если  четно: , иначе |

**Задание 2.** Модифицировать программу, составленную по Заданию 1, убрав возможность параллельного использования вычислительных ресурсов. Измерять время работы программы для тех же значений параметров, что были использованы при выполнении Задания 1. **Результаты сравнить и занести в отчёт.**

|  |  |  |
| --- | --- | --- |
| Размер массива n | Время вычисления без OpenMP, сек | Время вычисления с OpenMP, сек |
| 1 | 5.00004e-07 | 0.0119175 |
| 5 | 5.00004e-07 | 3.78999e-05 |
| 10 | 6.00005e-07 | 4.3e-05 |
| 100 | 2.10002e-06 | 4.08e-05 |
| 1000 | 1.52e-05 | 4.32001e-05 |
| 10000 | 7.71999e-05 | 7.61e-05 |
| 50000 | 0.000388 | 0.0002163 |
| 100000 | 0.0017256 | 0.0002687 |
| 1000000 | 0.0098707 | 0.0026271 |

/\*

На основе трёх равно размерных массивов A, B и C (длины N) функция возвращает произведение ненулевых значений,

полученных таким образом: если Ai четно: Bi/Ci , иначе Ai+Bi

\*/

#include <omp.h>

#include <iostream>

#include <ctime>

#include <iomanip>

// Возвращает псевдослучайное число из диапазона [min, max)

int Random(int min, int max) {

return min + rand() % (max - min);

}

//Создание рандомного массива

int\* generateArr(int n) {

//Автоматическая рандомизация

srand((unsigned int)time(NULL));

int\* arr = new int[n];

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

arr[i] = Random(1, 100);

}

return arr;

}

int sub(int n, int\* A, int\* B, int\* C) {

int result = 1;

int i, j, k;

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

if (A[i] % 2 == 0) {

result \*= B[i] / C[i];

}

else {

result \*= (A[i] + B[i]);

}

}

return result;

}

int sub\_omp(int n, int\* A, int\* B, int\* C) {

int result = 1;

int i, j, k;

#pragma omp parallel for shared(A,B,C) private(i) reduction(\*:result)

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

if (A[i] % 2 == 0) {

result \*= B[i] / C[i];

}

else {

result \*= (A[i] + B[i]);

}

}

return result;

}

void test(int n, int\* A, int\* B, int\* C) {

double t1, t2, result;

//Расчет произведения по данным условиям

t1 = omp\_get\_wtime();

result = sub(n, A, B, C);

t2 = omp\_get\_wtime();

std::cout << "Тест без использования OpenMP" << std::endl;

std::cout << "Произведение: " << result << std::endl;

std::cout << "Время исполнения основного вычислительного блока: " << t2 - t1 << std::endl;

}

void test\_omp(int n, int\* A, int\* B, int\* C) {

double t1, t2, result;

//Расчет произведения по данным условиям

t1 = omp\_get\_wtime();

result = sub\_omp(n, A, B, C);

t2 = omp\_get\_wtime();

std::cout << "Тест c использованием OpenMP" << std::endl;

std::cout << "Произведение: " << result << std::endl;

std::cout << "Время исполнения основного вычислительного блока: " << t2 - t1 << std::endl;

}

int main() {

system("chcp 1251");

int n;

while (true) {

//Ввод количества строк и столбцов массива

std::cout << "Введите размер массива n:" << std::endl;

std::cin >> n;

int\* A = generateArr(n);

int\* B = generateArr(n);

int\* C = generateArr(n);

test(n, A, B, C);

test\_omp(n, A, B, C);

}

return 0;

}