

**Правительство Российской Федерации**

**Федеральное государственное автономное образовательное**

**учреждение высшего образования «Национальный**

**исследовательский университет «Высшая школа экономики»**

**Факультет компьютерных наук**

**Департамент программной инженерии**

**Отчет к домашнему заданию По дисциплине**

**«Архитектура вычислительных систем»**

**Работу выполнил:**

**Студент группы БПИ-194 Назмутдинов Р.Р.**

**Москва 2020**

## Содержание

Отчет к домашнему заданию По дисциплине.....	0
1. ЗАДАЧА.....	2
2. РЕШЕНИЕ.....	3
2.1. bool IsPrime(int num) .....	3
2.2. void GenerateArr(unsigned int* arr, int elemCount, int seed) .....	3
2.3. void checkPairs(std::vector<int> &resVec, unsigned int* arrA, unsigned int* arrB, int startInd, int endInd).....	3
2.4. void ReadNumber(int &num, int minValue, int maxValue = INT_MAX) .....	3
2.5. int main() .....	3
Основания функция программы, в ней происходит считывание входных данных, генерация массивов, создание потоков и выполнение задачи. ....	3
3. КОД ПРОГРАММЫ .....	4
4. ТЕСТИРОВАНИЕ.....	7
4.1. Корректные входные данные .....	7
4.2. Некорректные входные данные.....	8
Программа обрабатывает ввод некорректных входных данных и не завершает работу аварийно (см. рис 5). ....	8
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ.....	10

## 1. ЗАДАЧА

Определить множество индексов  $i$ , для которых  $(A[i] - B[i])$  или  $(A[i] + B[i])$  являются простыми числами. Входные данные: массивы целых положительных чисел  $A$  и  $B$ , произвольной длины  $\geq 1000$ . Количество потоков является входным параметром.

## 2. РЕШЕНИЕ

Простое число – это число делителем которого является 1 и оно само. Опишем алгоритм решения данной задачи. Считывается количество элементов в массивах  $N$  и количество потоков  $M$ , использующихся в решении. Далее создаются массивы размера  $N$ , после чего созданные массивы заполняются случайными числами и создается  $M$  потоков каждый из которых отвечает за  $N/M$  элементов массивов. Потоки проходятся по своим кускам массивов и сохраняют все индексы при которых  $(A[i] + B[i])$  или  $(A[i] - B[i])$  являются простыми числами. При завершении работы все потоки закрываются и основной поток выводит все индексы при которых выполнялось описанное в задаче условие.

Таким образом в задаче используется итеративный параллелизм при котором используется несколько потоков, каждый из которых содержит в себе циклы.

При написании кода программы задача была разбита на несколько функций:

### 2.1. bool IsPrime(int num)

- num – проверяемое на простоту число;

Проверяет является ли переданное число простым. Если является, то возвращает True иначе False.

### 2.2. void GenerateArr(unsigned int\* arr, int elemCount, int seed)

- arr – ссылка на заполняемый массив;
- elemCount – количество элементов в массиве;
- seed – ключ генерации;

Заполняет массив, переданный по ссылке, с количеством элементов elemCount случайными положительными числами.

### 2.3. void checkPairs(std::vector<int> &resVec, unsigned int\* arrA, unsigned int\* arrB, int startInd, int endInd)

- resVec – ссылка на вектор хранящий в себе подходящие индексы;
- arrA – ссылка на массив A;
- arrB – ссылка на массив B;
- startInd – начальный индекс проверки;
- endIndex – конечный индекс проверки;

Проверяет является ли  $arrA[i] + arrB[i]$  или  $arrA[i] - arrB[i]$  простым числом и если является, то записывает индекс в вектор result.

### 2.4. void ReadNumber(int &num, int minValue, int maxValue = INT\_MAX)

- num – ссылка по которой будет записано значение;
  - minValue – минимальное значение, которое можно считать;
  - maxValue – максимальное значение, которое можно считать;
- Считывает число в отрезке  $[minValue, maxValue]$

### 2.5. int main()

Основания функция программы, в ней происходит считывание входных данных, генерация массивов, создание потоков и выполнение задачи.

### 3. КОД ПРОГРАММЫ

```

#include <iostream>
#include <mutex>
#include <cstdlib>
#include <ctime>
#include <vector>
#include <algorithm>
#include <omp.h>

std::mutex mtx;

/*
 * Студент: Назмутдинов Роман Ренатович
 * Группа: БПИ-194
 * Вариант 13
 * Определить множество индексов i, для которых (A[i] - B[i]) или
 * (A[i] + B[i]) являются простыми числами. Входные данные: массивы целых
 * положительных чисел A и B, произвольной длины  $\geq 1000$ . Количество
 * потоков является входным параметром.
 */

/**
 * Проверяет является ли переданное число простым
 * @param num проверяемое число
 * @return
 */
bool IsPrime(int num) {
    num = abs(num);
    for (int i = 2; i <= num / 2; ++i)
        if (num % i == 0)
            return false;
    return true;
}

/**
 * Заполняет массив случайными числами
 * @param arr ссылка на заполняемый массив
 * @param elemCount количество элементов в массиве
 * @param seed ключ генерации
 */
void GenerateArr(unsigned int* arr, int elemCount, int seed) {
    srand(seed);
    for (int i = 0; i < elemCount; ++i) {
        arr[i] = abs(rand());
    }
}

/**
 * Проверяет является ли arrA[i] + arrB[i] или arrA[i] - arrB[i]
 * простым числом и если является, то записывает индекс в вектор
 * result
 * @param resVec вектор с индексами
 * @param arrA
 * @param arrB
 * @param startInd начальный индекс проверки
 * @param endInd конечный индекс проверки
 */
void checkPairs(std::vector<std::pair<int, int>>* resVec, unsigned int* arrA, unsigned
int* arrB, int startInd, int endInd) {
    for (int i = startInd; i < endInd; ++i) {
        bool sum = false;
        if ((sum = IsPrime( arrA[i] + arrB[i])) || IsPrime(arrA[i] - arrB[i])) {
            //проверка на простоту

```

```

        int prime;
        if (sum)
            prime = arrA[i] + arrB[i];
        else
            prime = arrA[i] - arrB[i];
        #pragma omp critical
        resVec->push_back(std::pair<int, int>(i, prime));
    }
}

/**
 * Считывает число в отрезке [minValue, maxValue]
 * @param num ссылка по которой будет записано значение
 * @param minValue минимальное значение
 * @param maxValue максимальное значение
 */
void ReadNumber(int &num, int minValue, int maxValue = INT_MAX) {
    std::cin >> num;
    while (num < minValue || num > maxValue) {
        std::cout << "Incorrect input!" << std::endl;
        std::cout << "Enter number again:";
        std::cin >> num;
    }
}

int main() {
    srand(static_cast<int>(time(0))); //для генерации случайных чисел

    //считываем входные данные
    int size, threadCount, countOfLines;
    std::cout << "Enter size of arrays:";
    ReadNumber(size, 1000, 1000000);
    std::cout << "Enter count of threads:";
    ReadNumber(threadCount, 1, size);
    std::cout << "Enter the max number of lines to output:";
    ReadNumber(countOfLines, 1, size);

    //для таймера
    std::clock_t t1, t2;
    t1 = std::clock();

    //Создаем массивы
    unsigned int* arrA = new unsigned int[size];
    GenerateArr(arrA, size, rand());
    unsigned int* arrB = new unsigned int[size];
    GenerateArr(arrB, size, rand());

    //Создаем вектор для записи результата
    std::vector<std::pair<int, int>> result;
    int elemsCountForThread = size / threadCount; //вычисляем количество элементов на
каждый поток

    #pragma omp parallel num_threads(threadCount)
    {
        int i = omp_get_thread_num();
        int start = i * elemsCountForThread;
        int end = i < threadCount - 1 ? (i + 1) * elemsCountForThread : size;
        checkPairs(&result, arrA, arrB, start, end);
    }

    std::cout << "\nResult:" << std::endl;
    t2 = std::clock(); //останавливаем таймер

```

```

        std::cout << "time: " << (t2 - t1) / 1000.0 << " sec." << std::endl; //выводим время
        выполнения операции
        std::sort(result.begin(), result.end()); //сортируем вектор с нужными индексами
        int countOutputLines = countOfLines > result.size() ? result.size() : countOfLines;
        for (int i = result.size() - countOutputLines; i < result.size(); ++i) {
            std::printf("[%d] %u +/- %u = %d\n", result[i].first, arrA[result[i].first],
                arrB[result[i].first], result[i].second);
        }

        //удаляем массивы
        delete[] arrA;
        delete[] arrB;
        return 0;
    }

```

## 4. ТЕСТИРОВАНИЕ

### 4.1. Корректные входные данные

При вводе корректных входных данных программа выполняет поставленную задачу без каких-либо проблем (см. рис. 1-3). Также можно заметить, что при большем количестве потоков, задача выполняется быстрее (несмотря на то, что количество потоков увеличилось в 4 раза, время уменьшилось в 2 раза так как у компьютера, на котором проводились тесты, двоядерный процессор) (в каждом тесте, для удобства, выводятся последние 10 индексов) (см. рис. 3-4)

```

Консоль отладки Microsoft Visual Studio
Enter size of arrays:1000
Enter count of threads:2
Enter the max number of lines to output:10

Result:
time: 0.008 sec.
[944] 31396 +/- 30639 = 757
[945] 16658 +/- 17253 = 33911
[948] 21404 +/- 29789 = 51193
[950] 8142 +/- 611 = 8753
[964] 22050 +/- 32393 = 54443
[966] 14145 +/- 12268 = 1877
[978] 24753 +/- 13496 = 11257
[983] 31234 +/- 4359 = 35593
[986] 29788 +/- 11645 = 18143
[991] 19050 +/- 11119 = 30169

C:\Users\admin\source\repos\ConsoleApplication1\Debug\ConsoleApplication1.exe (процесс 11328) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно.
  
```

Рисунок 1 – Работа программы при передаче корректных входных данных (1000, 2)

```

Консоль отладки Microsoft Visual Studio
Enter size of arrays:1000000
Enter count of threads:1
Enter the max number of lines to output:10

Result:
time: 9.776 sec.
[999969] 8419 +/- 2382 = 6037
[999978] 10922 +/- 11971 = -1049
[999979] 26684 +/- 5307 = 31991
[999980] 5460 +/- 16139 = 21599
[999982] 32600 +/- 15023 = 47623
[999983] 29152 +/- 28351 = 57503
[999985] 21460 +/- 14509 = 35969
[999989] 19150 +/- 21653 = -2503
[999994] 9358 +/- 15687 = -6329
[999999] 3685 +/- 3688 = -3

C:\Users\admin\source\repos\ConsoleApplication1\Debug\ConsoleApplication1.exe (процесс 20148) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно.
  
```

Рисунок 2 – Работа программы при передаче корректных входных данных (1000000, 1)



```

Консоль отладки Microsoft Visual Studio
Enter size of arrays:1000000
Enter count of threads:4
Enter the max number of lines to output:10

Result:
time: 5.169 sec.
[999969] 8419 +/- 2382 = 6037
[999978] 10922 +/- 11971 = -1049
[999979] 26684 +/- 5307 = 31991
[999980] 5460 +/- 16139 = 21599
[999982] 32600 +/- 15023 = 47623
[999983] 29152 +/- 28351 = 57503
[999985] 21460 +/- 14500 = 35969
[999989] 19150 +/- 21653 = -2503
[999994] 9358 +/- 15687 = -6329
[999999] 3685 +/- 3688 = -3

C:\Users\admin\source\repos\ConsoleApplication1\Debug\ConsoleApplication1.exe (процесс 21728) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...

```

Рисунок 3 – Работа программы при передаче корректных входных данных (1000000, 4)

```

Консоль отладки Microsoft Visual Studio
Enter size of arrays:15000
Enter count of threads:11
Enter the max number of lines to output:10

Result:
time: 0.09 sec.
[14958] 16363 +/- 25704 = -9341
[14961] 21539 +/- 18688 = 2851
[14966] 3571 +/- 25878 = -22307
[14975] 5758 +/- 19547 = -13789
[14977] 29905 +/- 12114 = 42019
[14980] 21575 +/- 5702 = 27277
[14981] 3098 +/- 10219 = -7121
[14993] 23047 +/- 19390 = 42437
[14996] 24886 +/- 18185 = 6701
[14999] 10750 +/- 10691 = 59

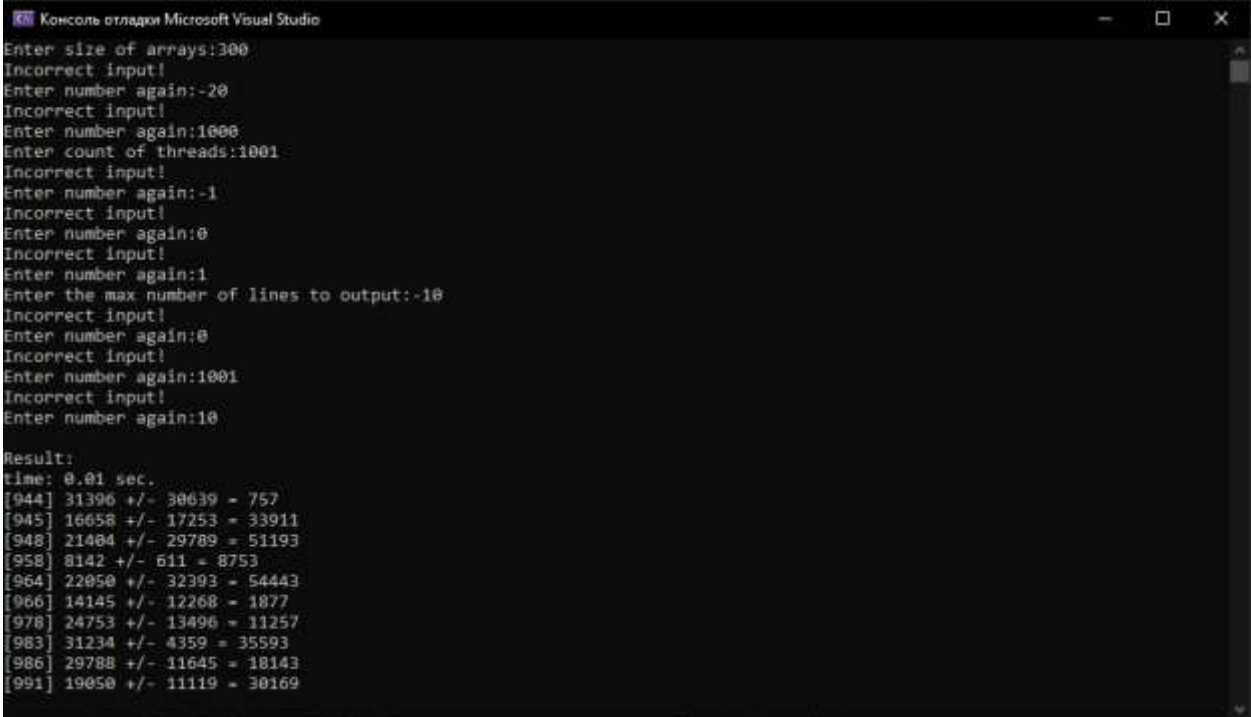
C:\Users\admin\source\repos\ConsoleApplication1\Debug\ConsoleApplication1.exe (процесс 8384) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...

```

Рисунок 4 – Работа программы при передаче корректных входных данных (15000, 11)

## 4.2. Некорректные входные данные

Программа обрабатывает ввод некорректных входных данных и не завершает работу аварийно (см. рис 5).



```

Консоль отладки Microsoft Visual Studio
Enter size of arrays:300
Incorrect input!
Enter number again:-20
Incorrect input!
Enter number again:1000
Enter count of threads:1001
Incorrect input!
Enter number again:-1
Incorrect input!
Enter number again:0
Incorrect input!
Enter number again:1
Enter the max number of lines to output:-10
Incorrect input!
Enter number again:0
Incorrect input!
Enter number again:1001
Incorrect input!
Enter number again:10

Result:
time: 0.01 sec.
[944] 31396 +/- 30039 = 757
[945] 16658 +/- 17253 = 33911
[948] 21404 +/- 29789 = 51193
[958] 8142 +/- 611 = 8753
[964] 22050 +/- 32393 = 54443
[966] 14145 +/- 12268 = 1877
[978] 24753 +/- 13496 = 11257
[983] 31234 +/- 4359 = 35593
[986] 29788 +/- 11645 = 18143
[991] 19050 +/- 11119 = 30169

```

Рисунок 5 – Работа программы при вводе некорректных входных данных

## СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. SoftCraft «Многопоточное программирование. OpenMP» (<http://www.softcraft.ru/edu/comparch/practice/thread/03-openmp/>)
2. Coursera «Технология OpenMP, особенность и ее компоненты» (<https://ru.coursera.org/lecture/parallelnoye-programmirovaniye/2-2-tiekhnologhiia-openmp-osobiennosti-i-ieie-komponenty-vhScx>)
3. Coursera «Задание параллельной области и опции, влияющие на ее выполнение» (<https://ru.coursera.org/lecture/parallelnoye-programmirovaniye/2-3-zadaniie-paralliel-noi-oblasti-i-optsii-vliiaiushchiie-na-ieie-vypolnieniie-Ywk5H>)
4. Coursera «Модель памяти. Классы переменных в OpenMP» (<https://ru.coursera.org/lecture/parallelnoye-programmirovaniye/2-4-modiel-pamiati-klassy-pieriemiennykh-v-openmp-OYrCJ>)
5. Microsoft «OpenMP Directives» (<https://docs.microsoft.com/ru-ru/cpp/parallel/openmp/reference/openmp-directives?view=msvc-160#parallel>)