**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 1](#_Toc56393918)

[1. ЗАДАЧА 3](#_Toc56393919)

[2. РЕШЕНИЕ 4](#_Toc56393920)

[2.1. bool IsPrime(int num) 4](#_Toc56393921)

[2.2. void GenerateArr(unsigned int\* arr, int elemCount, int seed) 4](#_Toc56393922)

[2.3. void checkPairs(std::vector<int> &resVec, unsigned int\* arrA, unsigned int\* arrB, int startInd, int endInd) 4](#_Toc56393923)

[2.4. void ReadNumber(int &num, int minValue, int maxValue = INT\_MAX) 4](#_Toc56393924)

[2.5. int main() 4](#_Toc56393925)

[Основания функция программы, в ней происходит считывание входных данных, генерация массивов, создание потоков и выполнение задачи. 4](#_Toc56393926)

[3. КОД ПРОГРАММЫ 5](#_Toc56393927)

[4. ТЕСТИРОВАНИЕ 8](#_Toc56393928)

[4.1. Корректные входные данные 8](#_Toc56393929)

[4.2. Некорректные входные данные 9](#_Toc56393930)

[Программа обрабатывает ввод некорректных входных данных и не завершает работу аварийно (см. рис 5). 9](#_Toc56393931)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 11](#_Toc56393932)

# 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 А и B, произвольной длины ≥ 1000. Количество потоков является входным параметром.

# 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 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:

## bool IsPrime(int num)

* num – проверяемое на простоту число;

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

## 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.

## 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.

## 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]

## int main()

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

# КОД ПРОГРАММЫ

#include <iostream>

#include <thread>

#include <mutex>

#include <cstdlib>

#include <ctime>

#include <vector>

#include <algorithm>

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 А и B, произвольной длины ≥ 1000. Количество

\* потоков является входным параметром.

\*/

/\*\*

\* Проверяет является ли переданное число простым

\* @param num проверяемое число

\* @return

\*/

bool IsPrime(int 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<int> &resVec, unsigned int\* arrA, unsigned int\* arrB, int startInd, int endInd) {

for (int i = startInd; i < endInd; ++i) {

if (IsPrime(arrA[i] + arrB[i]) || IsPrime(arrA[i] - arrB[i])) { //проверка на простоту

mtx.lock(); //не даем потокам возможности записывать данные одновременно

resVec.push\_back(i);

mtx.unlock();

}

}

}

/\*\*

\* 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;

std::cout << "Enter size of arrays:";

ReadNumber(size, 1000, 1000000);

std::cout << "Enter count of threads:";

ReadNumber(threadCount, 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<int> result;

int elemsCountForThread = size / threadCount; //вычисляем количество элементов на каждый поток

std::thread threads[threadCount]; //создаем массив потоков

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

int start = i \* elemsCountForThread;

int end = i < threadCount - 1 ? (i + 1) \* elemsCountForThread : size;

threads[i] = std::thread(checkPairs, std::ref(result), arrA, arrB, start, end); //создаем поток

}

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

threads[i].join(); //объединяем все потоки

}

t2 = std::clock(); //останавливаем таймер

std::cout << std::ednl << "time: " << (t2 - t1) / 1000.0 << " sec." << std::endl; //выводим время выполнения операции

std::cout << std::endl << "Result:" << std::endl;

std::sort(result.begin(), result.end()); //сортируем вектор с нужными индексами

for (int i = 0; i < result.size(); ++i) {

std::cout << result[i] << " "; //выводим вектор

}

//удаляем массивы

delete[] arrA;

delete[] arrB;

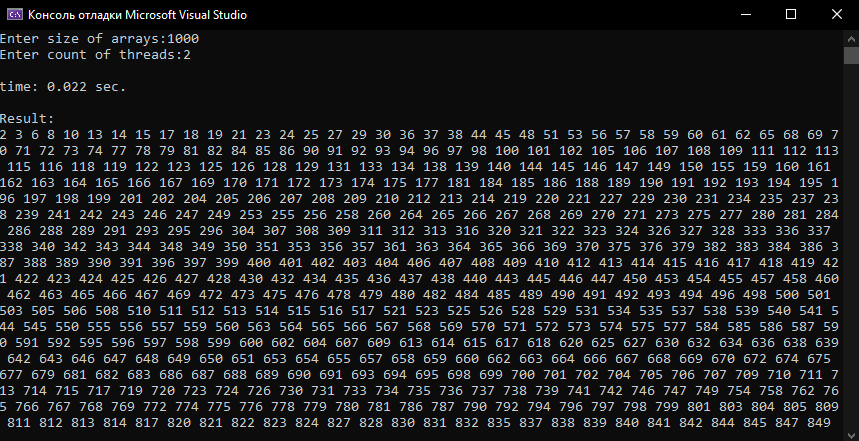
return 0;

}

# ТЕСТИРОВАНИЕ

## 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, двуядерный процессор) (см. рис. 3-4)

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

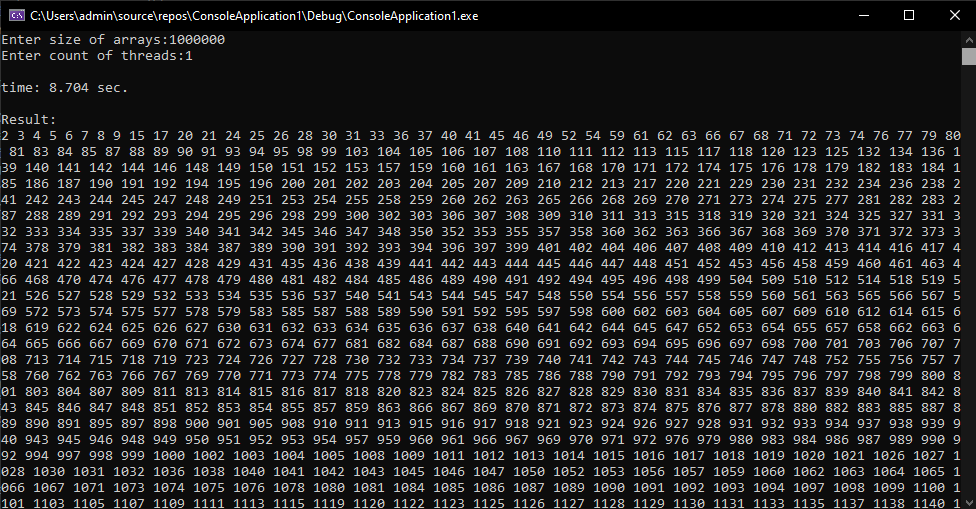


Рисунок 2 – Работа программы при передаче корректных входных данных (1000000, 1)

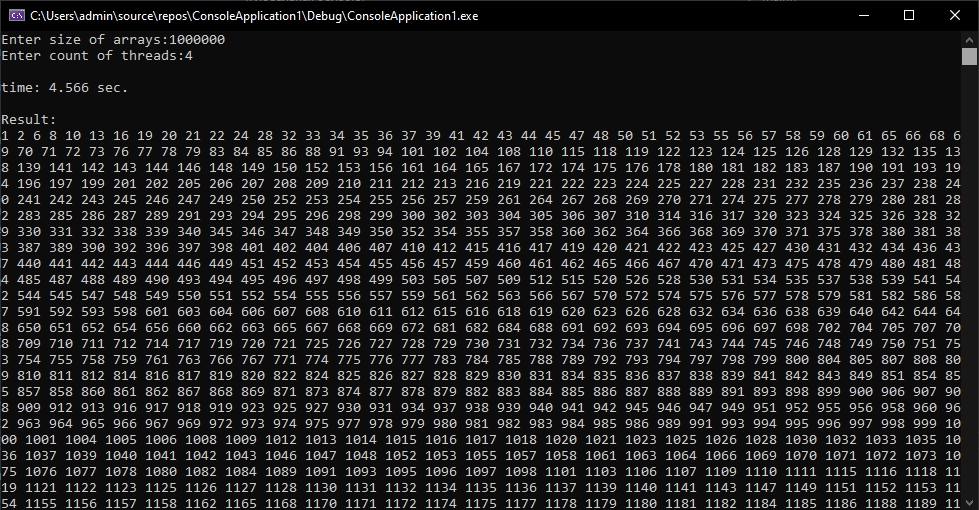


Рисунок 3 – Работа программы при передаче корректных входных данных (1000000, 4)

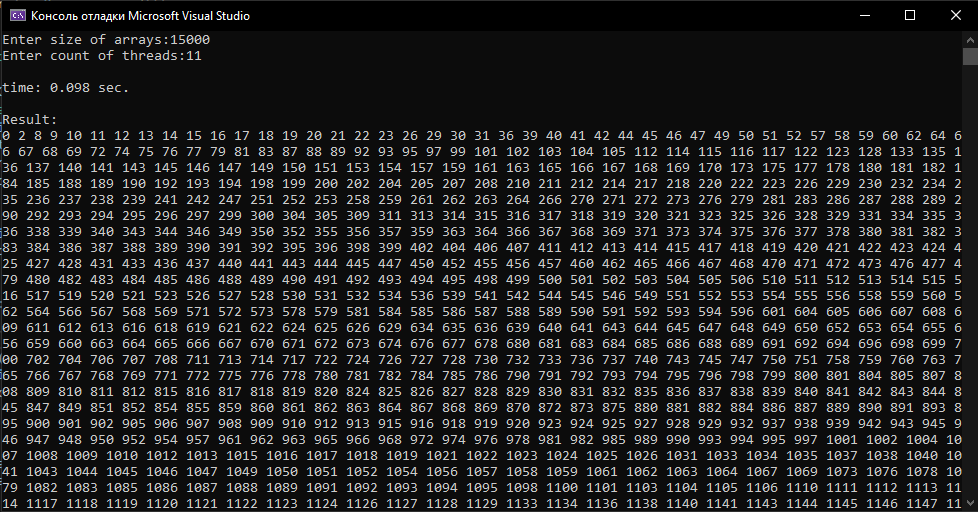


Рисунок 4 – Работа программы при передаче корректных входных данных (15000, 11)

## Некорректные входные данные

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

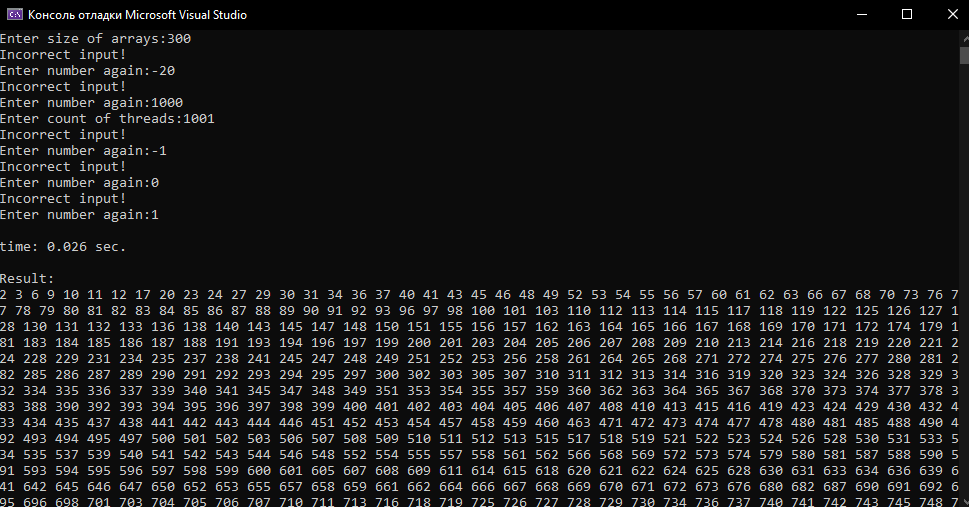


Рисунок 5 – Работа программы при вводе некорректных входных данных

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Блог программиста «Парадигмы параллельного программирования» (<https://pro-prof.com/forums/topic/parallel-programming-paradigms>);
2. YouTube канал «#SimpleCode» плейлист «Многопоточное программирование» (<https://www.youtube.com/watch?v=NawpxG81RRk&list=PLQOaTSbfxUtAc_RpyDiWCHq0YTzLtVSD0>);
3. SoftCraft «Практические приемы построения многопоточных приложений» (<http://www.softcraft.ru/edu/comparch/tasks/t03/>);
4. Wikipedia «Простое число» ([https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%81%D1%82%D0%BE%D0%B5\_%D1%87%D0%B8%D1%81%D0%BB%D0%BE#:~:text=2%2C%203%2C%205%2C%207,%D1%87%D0%B8%D1%81%D0%BB%D0%B0%20%D0%B1%D1%8B%D1%82%D1%8C%20%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%8B%D0%BC%20%D0%BD%D0%B0%D0%B7%D1%8B%D0%B2%D0%B0%D0%B5%D1%82%D1%81%D1%8F%20%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%BE%D1%82%D0%BE%D0%B9.](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%81%D1%82%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE%23:~:text=2%2C%203%2C%205%2C%207,%D1%87%D0%B8%D1%81%D0%BB%D0%B0%20%D0%B1%D1%8B%D1%82%D1%8C%20%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%8B%D0%BC%20%D0%BD%D0%B0%D0%B7%D1%8B%D0%B2%D0%B0%D0%B5%D1%82%D1%81%D1%8F%20%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%BE%D1%82%D0%BE%D0%B9.));
5. CyberForum «Таймер C++» (<https://www.cyberforum.ru/cpp-beginners/thread1149197.html>);