

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Департамент программной инженерии

**КОНСОЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ НАХОЖДЕНИЯ ВСЕХ
ВОЗМОЖНЫХ ТРОЕК КОМПЛАНАРНЫХ ВЕКТОРОВ ИЗ
МНОЖЕСТВА НЕ РАВНЫХ МЕЖДУ СОБОЙ ВЕКТОРОВ (X, Y, Z).
Пояснительная записка**

Исполнитель:
студент группы БПИ199
_____ Т. С. Кочарян
« _____ » _____ 2020 г.

Москва 2020

Оглавление:

Текст задания.	3
Применяемые расчетные методы.	4
Список используемых источников.	5
Описание работы программы.	6
Приложение №1.	11

1. Текст задания.

Найти все возможные тройки компланарных векторов. Входные данные: множество неравных между собой векторов (x, y, z) , где x, y, z – числа. Оптимальное количество потоков выбрать самостоятельно.

2. Применяемые расчетные методы.

Для проверки компланарности трех векторов используется формула смешанного произведения (см. [5]). Если смешанное произведение равно 0, следовательно тройка векторов компланарна. Для трех векторов a, b, c с координатами (x, y, z) формула имеет вид:

$$(a.x * b.y * c.z) + (a.y * b.z * c.x) + (a.z * b.x * c.y) - (a.z * b.y * c.x) - (a.x * b.z * c.y) - (b.x * a.y * c.z)$$

Также в работе программы используется итеративный параллелизм (см. [2]).

Для считывания данных используется поток ввода данных из файл `std::ifstream` библиотеки `fstream`.

Для хранения векторов и потоков используется контейнер `std::vector`.

3. Список используемых источников.

[1] Инструкция по составлению пояснительной записки [Электронный ресурс]. //URL: <http://softcraft.ru/edu/comparch/tasks/mp01/> (Дата обращения: 15.11.2020, режим доступа: свободный)

[2] Статья “Парадигмы параллельного программирования” [Электронный ресурс]. //URL: http://bit.do/iterator_paradigma (Дата обращения: 15.11.2020, режим доступа: свободный)

[3] Практические приемы построения многопоточных приложений [Электронный ресурс]. //URL: <http://softcraft.ru/edu/comparch/tasks/t03/> (Дата обращения: 15.11.2020, режим доступа: свободный)

[4] Multithreading in C++ [Электронный ресурс]. //URL: <https://www.geeksforgeeks.org/multithreading-in-cpp/> (Дата обращения: 15.11.2020, режим доступа: свободный)

[5] Компланарные векторы и условие компланарности [Электронный ресурс]. //URL: <http://bit.do/coplanar> (Дата обращения: 15.11.2020, режим доступа: свободный)

4. Описание работы программы.

Программа через консоль принимает на вход строку, которая является названием файла, который находится в одной директорией с исполняемым файлом или абсолютным путем к этому файлу. Считывание данных из файла обосновано тем, что таким образом достаточно удобно структурировать и считать количество векторов и сами вектора. В файле сначала должно находиться целочисленное значение общего количества векторов. Затем для каждого вектора будет по очереди введены целые числа - значения координат x , y , z . При считывании данных из файла происходят следующие проверки на валидность данных:

- Если в консоль было передано неверное количество аргументов, то будет выведено соответствующее сообщение об ошибке и программа завершит свою работу.
- Если при попытке потока ввода данных открыть файл произошла ошибка, то будет выведено соответствующее сообщение об ошибке и программа завершит свою работу. Проверка осуществляется с помощью флага `good()`
- Если в самом файле вместо целочисленного значения будет храниться некорректный вид данных (например, вместо числа будет храниться строка, которая не может быть переведена в число), то будет выведено соответствующее сообщение об ошибке и программа завершит свою работу. Проверка осуществляется с помощью флага `fail()`

Файл должен иметь вид:

- На первой строке записано количество векторов $\langle N \rangle$. ($N > 0$)
- Затем идут N строк по 3 числа. Каждое число целое и обозначает соответствующую координату вектора $\langle x \ y \ z \rangle$

Если все проверки были пройдены успешно, то по мере считывания векторов и их координат, программа заполняет одномерный массив проиндексированных элементов. Затем программа пронумеровано выводит в консоль все считанные вектора.

Для решения задачи фундаментальными являются два метода:

- Метод проверки на компланарность `isCoplanar` принимает на вход 3 параметра - вектора a, b, c и по формуле смешанного произведения проверяет, равен ли результат смешанного произведения 0. Если да, то тройка компланарна, в случае нет - некомпланарна.
- Метод `threadFunction` принимает на вход текущую позицию вектора i и проходясь от $i+1$ и $i+2$ до общего количества векторов, проверяет, компланарны ли векторы i , $i+1$, $i+2$. Если да, то в консоль выводится сообщение с информацией о компланарных векторах.

Для ускоренной проверки на компланарность трех векторов программа использует разделение работы на потоки. В данной задаче было принято решение использовать итеративный параллелизм, так как процессы выполняют циклические вычисления, решая одну задачу. Для реализации распараллеливания создаются потоки, количество которых равно общему количеству векторов. В каждый поток будет отправлена функция с

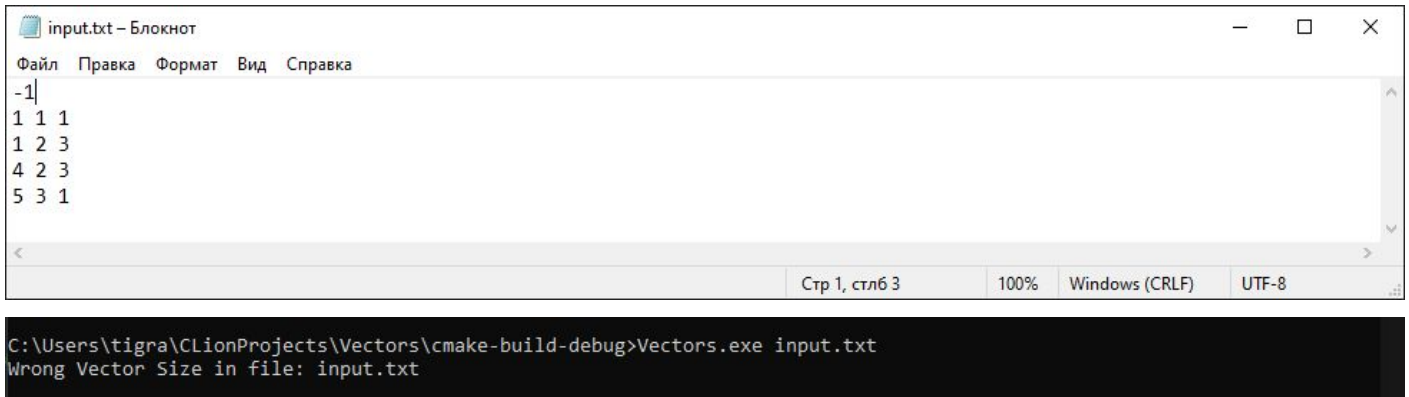
параметрами. Затем поток добавляется в одномерный массив проиндексированных элементов и соединяется с основным потоком.

В конце работы всех потоков программа завершает свое выполнение.

5. Тестирование программы.

Крайними случаями в данной программе могут быть некорректные данные в файле, некорректно заданные параметры в консоли. Для таких случаев предусмотрены проверки.

Пример 1: Некорректное значение количества векторов в файле. Вывод в консоль сообщения об ошибке и завершение работы.

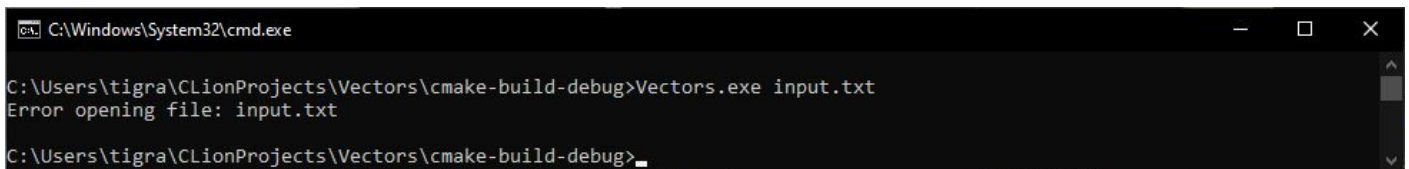


The image shows two windows. The top window is a Notepad editor titled 'input.txt - Блокнот'. It contains the following text:

```
-1|
1 1 1
1 2 3
4 2 3
5 3 1
```

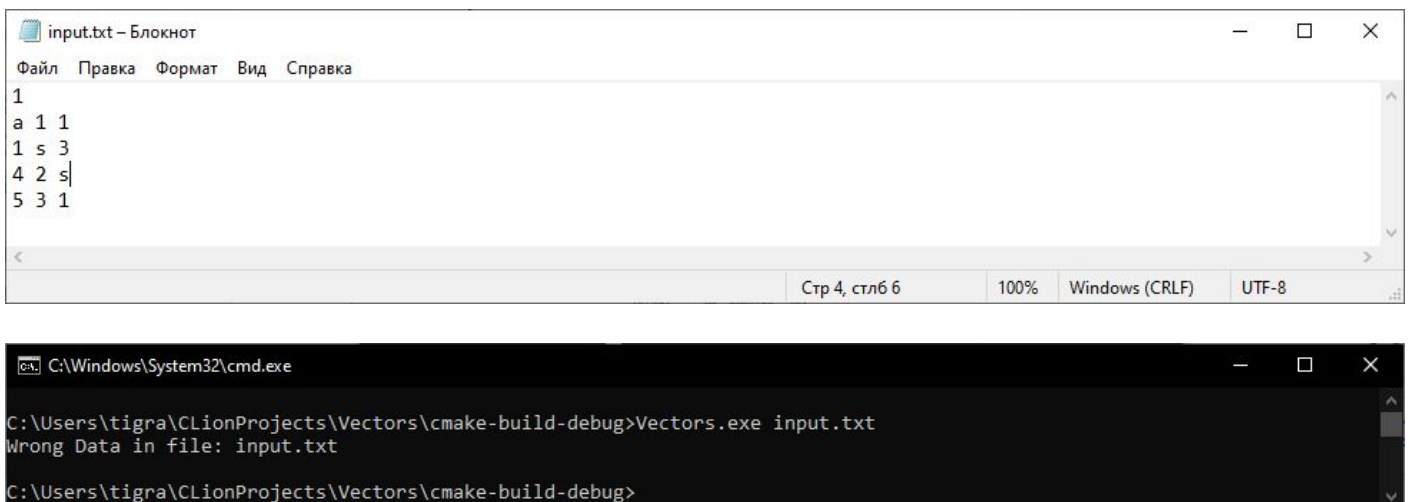
The bottom window is a command prompt titled 'C:\Windows\System32\cmd.exe'. It shows the command 'C:\Users\tigra\CLionProjects\Vectors\cmake-build-debug>Vectors.exe input.txt' and the output 'Wrong Vector Size in file: input.txt'.

Пример 2: Отсутствие файла, указанного в параметрах командной строки. Вывод сообщения о соответствующей ошибке в консоль и завершение работы приложения.



The image shows a command prompt window titled 'C:\Windows\System32\cmd.exe'. It shows the command 'C:\Users\tigra\CLionProjects\Vectors\cmake-build-debug>Vectors.exe input.txt' and the output 'Error opening file: input.txt'.

Пример 3. Некорректный тип данных количества векторов или координат вектора (например, строка или символ). Вывод сообщения о соответствующей ошибке в консоль и завершение работы приложения.



The image shows two windows. The top window is a Notepad editor titled 'input.txt - Блокнот'. It contains the following text:

```
1
a 1 1
1 s 3
4 2 s|
5 3 1
```

The bottom window is a command prompt titled 'C:\Windows\System32\cmd.exe'. It shows the command 'C:\Users\tigra\CLionProjects\Vectors\cmake-build-debug>Vectors.exe input.txt' and the output 'Wrong Data in file: input.txt'.

Пример 4. Некорректные введенные параметры в консоль при запуске. Вывод сообщения о соответствующей ошибке в консоль и завершение работы приложения.

```
C:\Windows\System32\cmd.exe
C:\Users\tigra\CLionProjects\Vectors\cmake-build-debug>Vectors.exe
Wrong Console Data
C:\Users\tigra\CLionProjects\Vectors\cmake-build-debug>_
```

Программа полноценно функционирует с разбиением на различные потоки с помощью итеративного параллелизма. Примеры будут описаны и показаны ниже:

Пример 5. Файл с входными данными содержит корректные данные о 6 векторах. Вывод каждым потоком отдельно всех найденных им компланарных векторов и завершение работы.

```
C:\Windows\System32\cmd.exe
C:\Users\tigra\CLionProjects\Vectors\cmake-build-debug>Vectors.exe input.txt
The input vectors are:
Vector #1 (1 1 1)
Vector #2 (1 2 3)
Vector #3 (4 2 3)
Vector #4 (5 3 1)
Vector #5 (4 6 3)
Vector #6 (7 8 2)

Coplanar vectors: (1 1 1) (1 2 3) (5 3 1)
C:\Users\tigra\CLionProjects\Vectors\cmake-build-debug>_
```

Пример 6. Файл с входными данными содержит корректные данные о 3 векторах. Вывод каждым потоком отдельно всех найденных им компланарных векторов и завершение работы.

```
C:\Windows\System32\cmd.exe
C:\Users\tigra\CLionProjects\Vectors\cmake-build-debug>Vectors.exe input.txt
The input vectors are:
Vector #1 (1 1 1)
Vector #2 (2 2 2)
Vector #3 (3 3 3)

Coplanar vectors: (1 1 1) (2 2 2) (3 3 3)
C:\Users\tigra\CLionProjects\Vectors\cmake-build-debug>
```

Пример 7. Файл с входными данными содержит корректные данные о 9 векторах. Вывод каждым потоком отдельно всех найденных им компланарных векторов и завершение работы.

```
C:\Windows\System32\cmd.exe
C:\Users\tigra\CLionProjects\Vectors\cmake-build-debug>Vectors.exe input.txt
The input vectors are:
Vector #1 (1 2 3)
Vector #2 (3 1 2)
Vector #3 (2 3 1)
Vector #4 (1 3 5)
Vector #5 (3 5 7)
Vector #6 (5 7 9)
Vector #7 (1 4 3)
Vector #8 (2 5 6)
Vector #9 (5 7 5)

Coplanar vectors: (1 2 3) (1 3 5) (3 5 7)
Coplanar vectors: (1 3 5) (3 5 7) (5 7 9)
Coplanar vectors: (1 2 3) (1 3 5) (5 7 9)
Coplanar vectors: (1 2 3) (3 5 7) (5 7 9)
Coplanar vectors: (1 2 3) (1 4 3) (2 5 6)
C:\Users\tigra\CLionProjects\Vectors\cmake-build-debug>
```

Приложение №1.

```
/**
 * Вариант задания 10.
 * Условие задания: Найти все возможные тройки
 * компланарных векторов. Входные
 * данные: множество не равных между собой векторов
 * (x, y, z), где x, y, z –
 * числа. Оптимальное количество потоков выбрать
 * самостоятельно.
 * Выполнил студент БПИ199 Кочарян Тигран Самвелович.
 */

#include <iostream>
#include <thread>
#include <fstream>
#include <sstream>
#include <vector>

/**
 * Класс для хранения координат вектора с
 * возможностью вывода в виде строки.
 */
class Vector {
public:
    int x;
    int y;
    int z;

    /**
     * Конструктор, принимающий координаты.
     * @param cordX
     * @param cordY
     * @param cordZ
     */
    Vector(int cordX, int cordY, int cordZ) :
```

```

        x(cordX), y(cordY), z(cordZ) {}

    /**
     * Пустой конструктор для инициализации
     * коллекций.
     */
    Vector() {}

    /**
     * Перевод координаты вектора в виде строки для
     * вывода.
     * @return
     */
    std::string toString() {
        std::ostringstream os;
        os << "(" << x << " " << y << " " << z << ") ";
        return os.str();
    }
};

/**
 * Проверка трех входных параметров-векторов на
 * компланарность.
 * Смешанное произведение == 0 для компланарных
 * векторов.
 * @param a
 * @param b
 * @param c
 * @return
 */
bool isCoplanar(Vector a, Vector b, Vector c) {
    int value = (a.x * b.y * c.z) + (a.y * b.z * c.x) + (a.z * b.x * c.y) -
                (a.z * b.y * c.x) - (a.x * b.z * c.y) - (b.x * a.y * c.z);
    return value == 0;
}

```



```

/**
 * Проверка текущего потока считывания файла на
 * корректные данные. В случае, если данные
 * некорректные,
 * программа завершается с соответствующей ошибкой.
 * @param in
 * @param filename
 */
void isIncorrect(std::ifstream &in, const std::string &filename) {
    if (in.fail()) {
        std::cout << "Wrong Data in file: " << filename << std::endl;
        std::exit(EXIT_FAILURE);
    }
}

/**
 * Функция для запуска потоков.
 * Проверяет на компланарность i+1 и i+2 векторов между
 * собой.
 * @param i
 * @param vectorCount
 * @param vectors_
 */
void threadFunction(int i, int vectorCount, std::vector<Vector> *vectors_) {
    std::vector<Vector> &vectors = *vectors_;
    for (int j = i + 1; j < vectorCount; ++j) {
        for (int k = j + 1; k < vectorCount; ++k) {
            if (isCoplanar(vectors[i], vectors[j], vectors[k])) {
                std::ostringstream os;
                os << "\nCoplanar vectors: " << vectors[i].toString() <<
                    vectors[j].toString() << vectors[k].toString();
                std::cout << os.str();
            }
        }
    }
}

```

```

}

/**
 * Точка входа в программу.ы
 * @param argsNumber
 * @param args
 * @return
 */
int main(int argsNumber, char **args) {
    /**
     * Если количество аргументов не равно 2, то у нас
     * отсутствует название файла
     * с входными данными => сообщение об ошибке.
     */
    if (argsNumber != 2) {
        std::cout << "Wrong Console Data" << std::endl;
        std::exit(EXIT_FAILURE);
    }

    std::ifstream in{args[1]};
    int vectorCount, x, y, z;

    /**
     * Если файл существует, то продолжаем работу.
     * В противном случае, закрываем программу с
     * сообщением об ошибке.
     */
    if (!in.good()) {
        std::cout << "Error opening file: " << args[1] << std::endl;
        std::exit(EXIT_FAILURE);
    } else {
        in >> vectorCount;
        if(vectorCount < 1) {
            std::cout << "Wrong Vector Size in file: " <<args[1] << std::endl;
            std::exit(EXIT_FAILURE);
        }
    }
}

```

```

}

std::vector<Vector> vectors;

/**
 * Построчное считывание векторов из файла с
 * проверкой на корректность.
 */
for (int i = 0; i < vectorCount; ++i) {
    in >> x;
    in >> y;
    in >> z;
    isIncorrect(in, args[1]);
    vectors.emplace_back(Vector(x, y, z));
}

/**
 * Вывод всех векторов в консоль для удобного
 * взаимодействия и проверки.
 */
std::cout << "The input vectors are:" << std::endl;
for (int i = 0; i < vectorCount; ++i) {
    std::cout << "Vector #" << i + 1 << " " << vectors[i].toString() <<
std::endl;
}

/**
 * Заполнение вектора потоками и последующий
 * join() с основным потоком.
 */
std::vector<std::thread> threads;
for (int i = 0; i < vectorCount; ++i) {
    threads.emplace_back(threadFunction, i, vectorCount, &vectors);
}
for (int i = 0; i < vectorCount; ++i) {
    threads[i].join();
}

```


}

}

}