**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

**ПРОГРАММА, НАХОДЯЩАЯ ВСЕ ТРОЙКИ**

**КОМПЛАНАРНЫХ ВЕКТОРОВ**

**Пояснительная записка**

Исполнитель

студент группы БПИ193

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /С. М. Курбанова /

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

**Москва 2020**

**СОДЕРЖАНИЕ**

[1. Текст задания 2](#_Toc56543229)

[2. Применяемые расчётные методы 3](#_Toc56543230)

[3. Организация входных и выходных данных 4](#_Toc56543231)

[4. Последовательность работы программы 5](#_Toc56543232)

[5. Тестирование программы 6](#_Toc56543233)

[6. Список использованных источников 8](#_Toc56543234)

[7. Текст программы 9](#_Toc56543235)

# Текст задания

Вариант 10. 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.

# Применяемые расчётные методы

Для нахождения всех троек компланарных векторов [1] был использован основной критерий компланарности трёх векторов – смешанное произведение [2] компланарных векторов равно нулю.

Использованная в расчетах формула нахождения смешанного произведения:

Формула векторного произведения:

Формула скалярного произведения:

Для решения вышеописанной задачи был выбран итерационный параллелизм, так как в решении предусмотрена оптимизация вычисления смешанных произведений, что содержит в себе несколько циклов, перебирающих тройки векторов. Задача равномерно распределена между восемью потоками, которые выполняют одну и ту же функцию.

# Организация входных и выходных данных

Ввод и вывод данных осуществляются с помощью файлов. Для удобства пользователя ответ также выводится в консоль.

# Последовательность работы программы

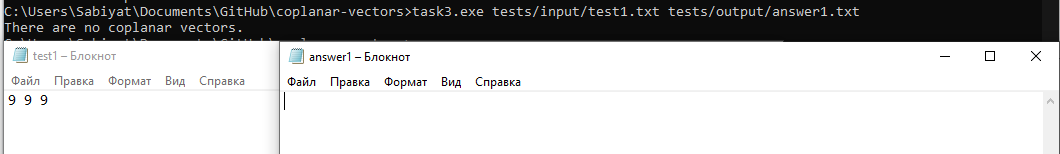
1. Считывание данных из файла;
2. Поиск всех троек компланарных векторов путем перебора векторов вложенными циклами. Здесь каждый поток берет на себя равную часть векторов во внешнем цикле.
3. Вывод получившегося множества троек компланарных векторов в консоль и файл.

# Тестирование программы

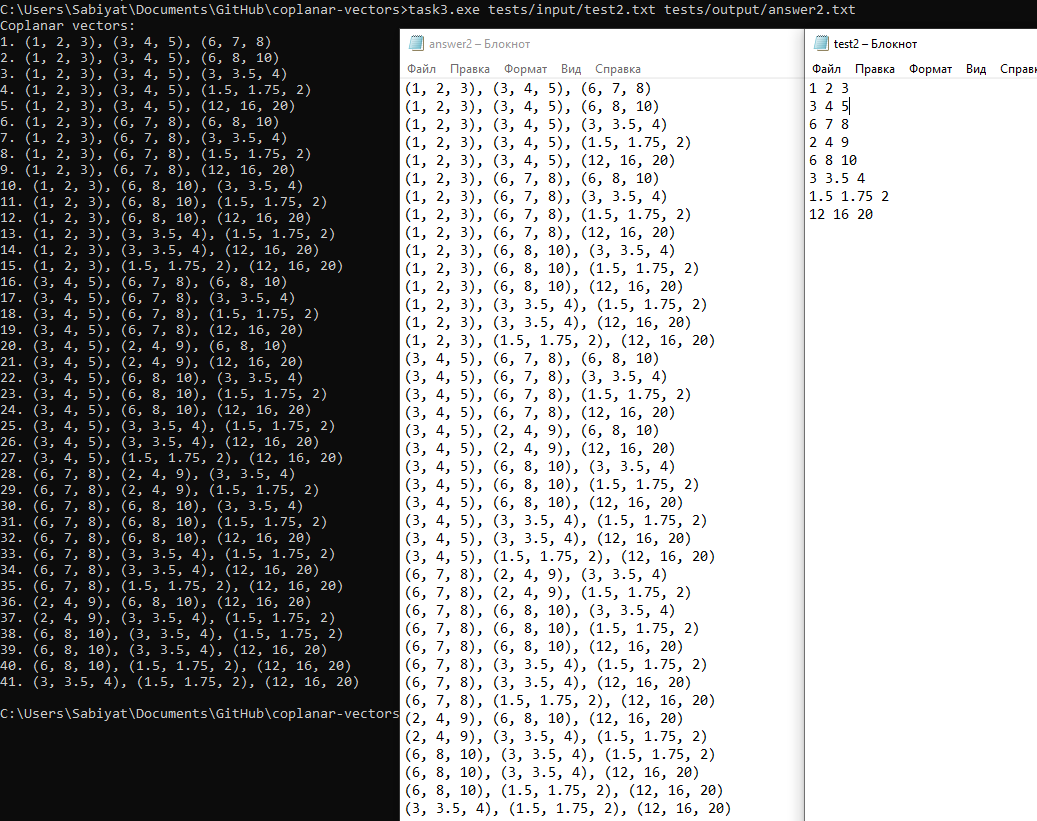
Для запуска программы из командной строки необходимо ввести task3.exe arg1 arg2,

где arg1 – это путь к файлу с тестом, а arg2 – это путь к файлу с ответом.

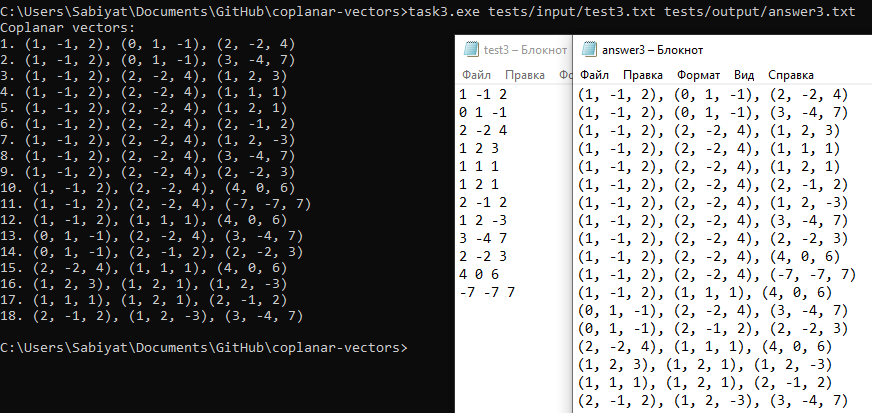
1. Тест 1. Отсутствие компланарных векторов.



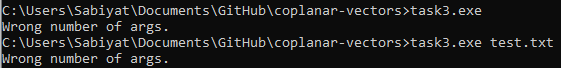
1. Тест 2. Поиск всех троек компланарных векторов.



1. Тест 3. Поиск всех троек компланарных векторов.



1. Тест 4. Ввод неверного количества аргументов.



# Список использованных источников

1. Компланарные векторы // [Электронный ресурс] Режим доступа: <https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BB%D0%B0%D0%BD%D0%B0%D1%80%D0%BD%D0%BE%D1%81%D1%82%D1%8C>, свободный;
2. Смешанное произведение // [Электронный ресурс] Режим доступа: <https://ru.wikipedia.org/wiki/%D0%A1%D0%BC%D0%B5%D1%88%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5>, свободный;
3. Примеры программ // [Электронный ресурс] Режим доступа: <http://softcraft.ru/edu/comparch/practice/thread/01-simple/>, свободный;
4. Оформление задания // [Электронный ресурс] Режим доступа: <http://softcraft.ru/edu/comparch/tasks/t03/>, свободный;
5. Многопоточность // [Электронный ресурс] Режим доступа: <http://softcraft.ru/edu/comparch/lect/07-parthread/multitreading.pdf>, свободный.

# Текст программы

// Курбанова Сабият Магомедовна

// Группа БПИ193

// Вариант 10

#include <iostream>

#include <vector>

#include <fstream>

#include <thread>

constexpr auto THREADS\_COUNT = 8;

struct Vector

{

public:

double X;

double Y;

double Z;

Vector()

{

X = 0;

Y = 0;

Z = 0;

}

Vector(double x, double y, double z)

{

X = x;

Y = y;

Z = z;

}

/\*

\* Checks if this vector and two other are coplanar by

\* checking if scalar triple product is equal to 0

\*/

bool isCoplanar(Vector v2, Vector v3)

{

double p[3] = { v2.Y \* v3.Z - v2.Z \* v3.Y,

v2.Z \* v3.X - v2.X \* v3.Z,

v2.X \* v3.Y - v2.Y \* v3.X };

return X \* p[0] + Y \* p[1] + Z \* p[2] == 0;

}

};

std::ostream& operator<<(std::ostream& strm, const Vector& v) {

return strm << "(" << v.X << ", " << v.Y << ", " << v.Z << ")";

}

/\*

\* Reads vectors from file

\*/

std::vector<Vector> getVectorsFromFile(std::string path)

{

std::vector<Vector> vectors;

std::fstream in(path, std::ios::in);

if (in.is\_open())

{

int vectorsCount = 0;

double x;

double y;

double z;

while (in >> x >> y >> z)

{

Vector currentVector(x, y, z);

vectors.push\_back(currentVector);

}

in.close();

}

return vectors;

}

/\*

\* Finds all triplets of coplanar vectors

\*/

void findTriplets(std::vector<Vector> vectors, std::vector<std::vector<Vector>>& coplanarVectors, int threads, int start)

{

for (int k = start; k < vectors.size(); k += threads)

{

for (int i = k + 1; i < vectors.size(); i++)

{

for (int j = i + 1; j < vectors.size(); j++)

{

if (vectors[k].isCoplanar(vectors[i], vectors[j]))

{

std::vector<Vector> triplet;

triplet.push\_back(vectors[k]);

triplet.push\_back(vectors[i]);

triplet.push\_back(vectors[j]);

coplanarVectors.push\_back(triplet);

}

}

}

}

}

/\*

\* Prints result to console and output file

\*/

void printResult(std::vector<std::vector<Vector>> coplanarVectors, std::string outputPath)

{

std::ofstream out(outputPath, std::ios::out);

if (coplanarVectors.size() == 0)

{

std::cout << "There are no coplanar vectors." << std::endl;;

return;

}

std::cout << "Coplanar vectors: " << std::endl;

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

{

std::cout << i + 1 << ". " << coplanarVectors[i][0] << ", " << coplanarVectors[i][1] << ", " << coplanarVectors[i][2] << std::endl;

if (out.is\_open())

{

out << coplanarVectors[i][0] << ", " << coplanarVectors[i][1] << ", " << coplanarVectors[i][2] << std::endl;

}

}

}

int main(int argc, char\*\* argv)

{

std::vector<Vector> vectors;

std::vector<std::vector<Vector>> coplanarVectors[THREADS\_COUNT];

std::vector<std::vector<Vector>> res;

if (argc < 3)

{

std::cout << "Wrong number of args." << std::endl;

return -1;

}

vectors = getVectorsFromFile(argv[1]);

std::thread\* thr[THREADS\_COUNT];

// Create threads

for (int i = 0; i < THREADS\_COUNT; i++)

{

thr[i] = new std::thread{ findTriplets, vectors, std::ref(coplanarVectors[i]), THREADS\_COUNT, i };

}

for (int i = 0; i < THREADS\_COUNT; i++)

{

thr[i]->join();

res.insert(std::end(res), std::begin(coplanarVectors[i]), std::end(coplanarVectors[i]));

delete thr[i];

}

printResult(res, argv[2]);

}