**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](#_Toc57560692)

[2. Применяемые расчётные методы 3](#_Toc57560693)

[3. Описание работы программы 4](#_Toc57560694)

[4. Организация входных и выходных данных 5](#_Toc57560695)

[5. Тестирование программы 6](#_Toc57560696)

[6. Список использованных источников 8](#_Toc57560697)

[7. Текст программы 9](#_Toc57560698)

# 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. Использовать OpenMP.

# 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:

# Описание работы программы

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 распределена между максимальным количеством потоков, которые выполняют одну и ту же функцию: каждый поток берёт вектор и перебирает его с двумя следующими, постепенно проходя по всем парам без перестановок.

# Организация входных и выходных данных

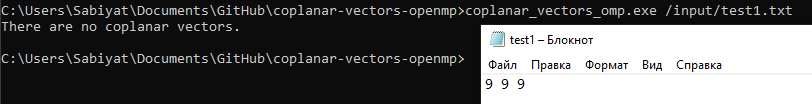
Ввод данных осуществляются с помощью файла, в котором координаты векторов записаны через пробел, каждый вектор находится на отдельной строке.

Ответ выводится в консоль.

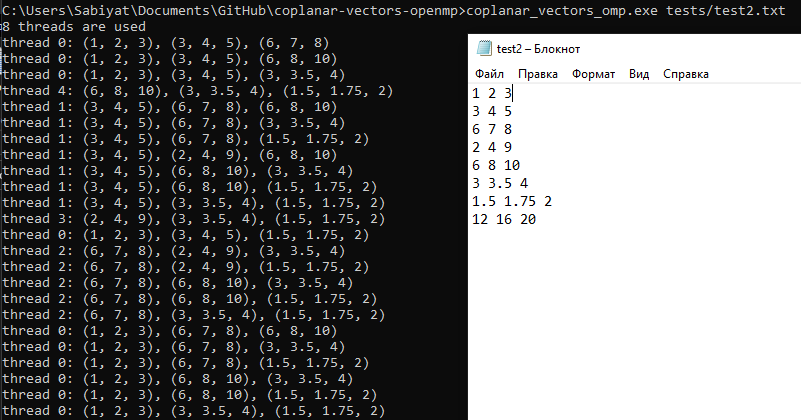
# Тестирование программы

Для запуска программы из командной строки необходимо ввести coplavar\_vectors\_ omp.exe arg1, где arg1 – это путь к файлу с тестом.

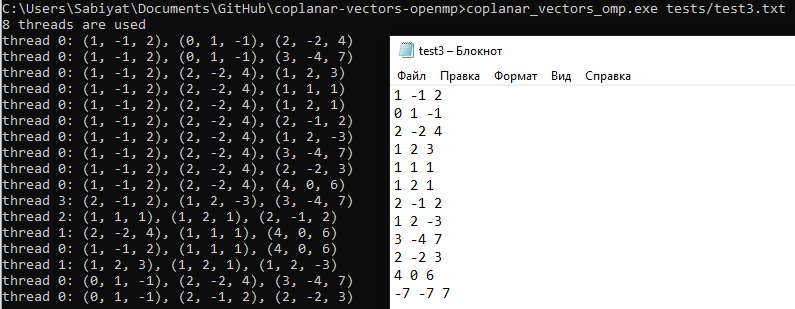
1. Тест 1. Отсутствие компланарных векторов.



1. Тест 2. Поиск всех троек компланарных векторов.



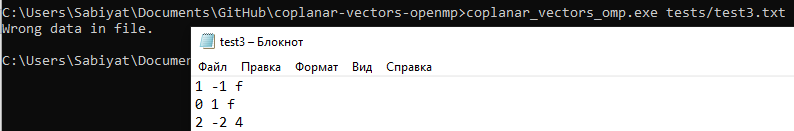
1. Тест 3. Поиск всех троек компланарных векторов.



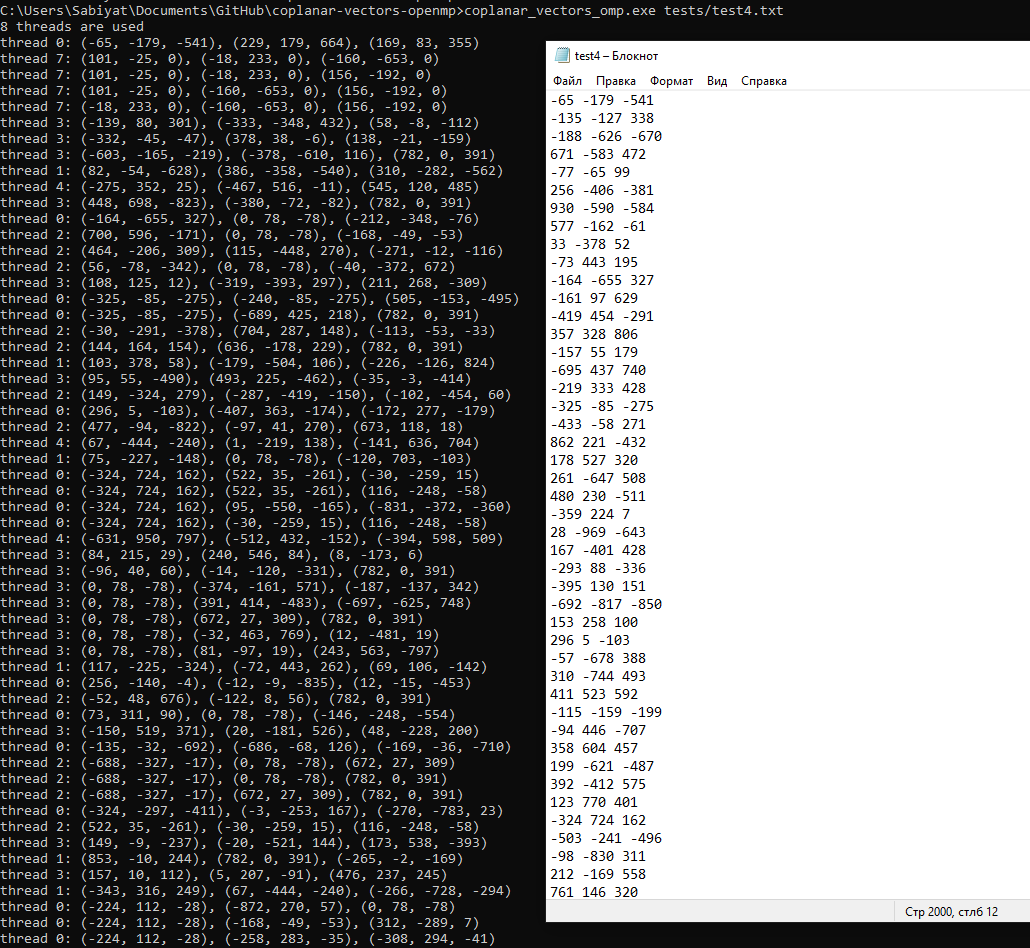
1. Тест 4. Ввод неверного количества аргументов.



1. Тест 5. Неверные значения в файле.



1. Тест 6. Поиск всех троек компланарных векторов.



# 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/03-openmp/>, свободный;
4. Оформление задания // [Электронный ресурс] Режим доступа: <http://softcraft.ru/edu/comparch/tasks/t04/>, свободный;
5. Директивы OpenMP // [Электронный ресурс] Режим доступа: <https://docs.microsoft.com/ru-ru/cpp/parallel/openmp/reference/openmp-directives?view=msvc-160#for-openmp>, свободный;
6. Параллельное программирование на OpenMP // [Электронный ресурс] Режим доступа: <http://ccfit.nsu.ru/arom/data/openmp.pdf>, свободный.

# Текст программы

#include <iostream>

#include <vector>

#include <fstream>

#include "omp.h"

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;

}

/// <summary>

/// Checks if this vector and two other are coplanar by

/// checking if scalar triple product is equal to 0

/// </summary>

/// <param name="v2">second vector</param>

/// <param name="v3">third vector</param>

/// <returns>if vectors are coplanar</returns>

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;

}

};

/// <summary>

/// Reads vectors from file

/// </summary>

/// <param name="path">path to input file</param>

/// <param name="vectors">vector of Vectors</param>

/// <returns>if reading was successful</returns>

bool getVectorsFromFile(std::string path, std::vector<Vector>& vectors)

{

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

double x;

double y;

double z;

if (in.is\_open())

{

in >> x >> y >> z;

while (!in.eof())

{

if (in.fail() && !in.eof())

return false;

Vector currentVector(x, y, z);

vectors.push\_back(currentVector);

in >> x >> y >> z;

}

in.close();

return true;

}

return false;

}

/// <summary>

/// Finds all triplets of coplanar vectors (thread function)

/// </summary>

/// <param name="vectors">input vectors</param>

/// <param name="k">vector's index</param>

void findTriplets(std::vector<Vector> vectors, int k)

{

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]))

{

printf("thread %d: (%g, %g, %g), (%g, %g, %g), (%g, %g, %g)\n",

omp\_get\_thread\_num(),

vectors[k].X, vectors[k].Y, vectors[k].Z,

vectors[i].X, vectors[i].Y, vectors[i].Z,

vectors[j].X, vectors[j].Y, vectors[j].Z);

}

}

}

}

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

{

std::vector<Vector> vectors;

if (argc != 2)

{

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

return -1;

}

if (!getVectorsFromFile(argv[1], vectors))

{

std::cout << "Wrong data in file." << std::endl;

return -1;

}

if (vectors.size() < 3)

{

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

return 0;

}

std::cout << omp\_get\_max\_threads() << " threads are used" << std::endl;

omp\_set\_num\_threads(omp\_get\_max\_threads());

#pragma omp parallel for

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

findTriplets(vectors, i);

}