

**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
2.	Применяемые расчётные методы	3
3.	Организация входных и выходных данных.....	4
4.	Последовательность работы программы	5
5.	Тестирование программы	6
6.	Список использованных источников	8
7.	Текст программы.....	9

1. 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.

2. 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:

$$(a, b, c) = a \cdot (b \times c)$$

Формула векторного произведения:

$$(b \times c) = (b_y \cdot c_z - b_z \cdot c_y) + (b_z \cdot c_x - b_x \cdot c_z) + (b_x \cdot c_y - b_y \cdot c_x)$$

Формула скалярного произведения:

$$a \cdot b = a_x \cdot b_x + a_y \cdot b_y + a_z \cdot b_z$$

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.

3. Организация входных и выходных данных

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

4. 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.

5. 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.

```
C:\Users\Sabiyat\Documents\GitHub\coplanar-vectors>task3.exe tests/input/test1.txt tests/output/answer1.txt
There are no coplanar vectors.
```

test1 – Блокнот

Файл Правка Формат Вид Справка

9 9 9

answer1 – Блокнот

Файл Правка Формат Вид Справка

2. Тест 2. Поиск всех троек компланарных векторов.

```
C:\Users\Sabiyat\Documents\GitHub\coplanar-vectors>task3.exe tests/input/test2.txt tests/output/answer2.txt
Coplanar vectors:
1. (1, 2, 3), (3, 4, 5), (6, 7, 8)
2. (1, 2, 3), (3, 4, 5), (6, 8, 10)
3. (1, 2, 3), (3, 4, 5), (3, 3.5, 4)
4. (1, 2, 3), (3, 4, 5), (1.5, 1.75, 2)
5. (1, 2, 3), (3, 4, 5), (12, 16, 20)
6. (1, 2, 3), (6, 7, 8), (6, 8, 10)
7. (1, 2, 3), (6, 7, 8), (3, 3.5, 4)
8. (1, 2, 3), (6, 7, 8), (1.5, 1.75, 2)
9. (1, 2, 3), (6, 7, 8), (12, 16, 20)
10. (1, 2, 3), (6, 8, 10), (3, 3.5, 4)
11. (1, 2, 3), (6, 8, 10), (1.5, 1.75, 2)
12. (1, 2, 3), (6, 8, 10), (12, 16, 20)
13. (1, 2, 3), (3, 3.5, 4), (1.5, 1.75, 2)
14. (1, 2, 3), (3, 3.5, 4), (12, 16, 20)
15. (1, 2, 3), (1.5, 1.75, 2), (12, 16, 20)
16. (3, 4, 5), (6, 7, 8), (6, 8, 10)
17. (3, 4, 5), (6, 7, 8), (3, 3.5, 4)
18. (3, 4, 5), (6, 7, 8), (1.5, 1.75, 2)
19. (3, 4, 5), (6, 7, 8), (12, 16, 20)
20. (3, 4, 5), (2, 4, 9), (6, 8, 10)
21. (3, 4, 5), (2, 4, 9), (12, 16, 20)
22. (3, 4, 5), (6, 8, 10), (3, 3.5, 4)
23. (3, 4, 5), (6, 8, 10), (1.5, 1.75, 2)
24. (3, 4, 5), (6, 8, 10), (12, 16, 20)
25. (3, 4, 5), (3, 3.5, 4), (1.5, 1.75, 2)
26. (3, 4, 5), (3, 3.5, 4), (12, 16, 20)
27. (3, 4, 5), (1.5, 1.75, 2), (12, 16, 20)
28. (6, 7, 8), (2, 4, 9), (3, 3.5, 4)
29. (6, 7, 8), (2, 4, 9), (1.5, 1.75, 2)
30. (6, 7, 8), (6, 8, 10), (3, 3.5, 4)
31. (6, 7, 8), (6, 8, 10), (1.5, 1.75, 2)
32. (6, 7, 8), (6, 8, 10), (12, 16, 20)
33. (6, 7, 8), (3, 3.5, 4), (1.5, 1.75, 2)
34. (6, 7, 8), (3, 3.5, 4), (12, 16, 20)
35. (6, 7, 8), (1.5, 1.75, 2), (12, 16, 20)
36. (2, 4, 9), (6, 8, 10), (12, 16, 20)
37. (2, 4, 9), (3, 3.5, 4), (1.5, 1.75, 2)
38. (6, 8, 10), (3, 3.5, 4), (1.5, 1.75, 2)
39. (6, 8, 10), (3, 3.5, 4), (12, 16, 20)
40. (6, 8, 10), (1.5, 1.75, 2), (12, 16, 20)
41. (3, 3.5, 4), (1.5, 1.75, 2), (12, 16, 20)
```

answer2 – Блокнот

Файл Правка Формат Вид Справка

(1, 2, 3), (3, 4, 5), (6, 7, 8)
(1, 2, 3), (3, 4, 5), (6, 8, 10)
(1, 2, 3), (3, 4, 5), (3, 3.5, 4)
(1, 2, 3), (3, 4, 5), (1.5, 1.75, 2)
(1, 2, 3), (3, 4, 5), (12, 16, 20)
(1, 2, 3), (6, 7, 8), (6, 8, 10)
(1, 2, 3), (6, 7, 8), (3, 3.5, 4)
(1, 2, 3), (6, 7, 8), (1.5, 1.75, 2)
(1, 2, 3), (6, 7, 8), (12, 16, 20)
(1, 2, 3), (6, 8, 10), (3, 3.5, 4)
(1, 2, 3), (6, 8, 10), (1.5, 1.75, 2)
(1, 2, 3), (6, 8, 10), (12, 16, 20)
(1, 2, 3), (3, 3.5, 4), (1.5, 1.75, 2)
(1, 2, 3), (3, 3.5, 4), (12, 16, 20)
(1, 2, 3), (1.5, 1.75, 2), (12, 16, 20)
(3, 4, 5), (6, 7, 8), (6, 8, 10)
(3, 4, 5), (6, 7, 8), (3, 3.5, 4)
(3, 4, 5), (6, 7, 8), (1.5, 1.75, 2)
(3, 4, 5), (6, 7, 8), (12, 16, 20)
(3, 4, 5), (2, 4, 9), (6, 8, 10)
(3, 4, 5), (2, 4, 9), (12, 16, 20)
(3, 4, 5), (6, 8, 10), (3, 3.5, 4)
(3, 4, 5), (6, 8, 10), (1.5, 1.75, 2)
(3, 4, 5), (6, 8, 10), (12, 16, 20)
(3, 4, 5), (3, 3.5, 4), (1.5, 1.75, 2)
(3, 4, 5), (3, 3.5, 4), (12, 16, 20)
(3, 4, 5), (1.5, 1.75, 2), (12, 16, 20)
(6, 7, 8), (2, 4, 9), (3, 3.5, 4)
(6, 7, 8), (2, 4, 9), (1.5, 1.75, 2)
(6, 7, 8), (6, 8, 10), (3, 3.5, 4)
(6, 7, 8), (6, 8, 10), (1.5, 1.75, 2)
(6, 7, 8), (6, 8, 10), (12, 16, 20)
(6, 7, 8), (3, 3.5, 4), (1.5, 1.75, 2)
(6, 7, 8), (3, 3.5, 4), (12, 16, 20)
(6, 7, 8), (1.5, 1.75, 2), (12, 16, 20)
(2, 4, 9), (6, 8, 10), (12, 16, 20)
(2, 4, 9), (3, 3.5, 4), (1.5, 1.75, 2)
(6, 8, 10), (3, 3.5, 4), (1.5, 1.75, 2)
(6, 8, 10), (3, 3.5, 4), (12, 16, 20)
(6, 8, 10), (1.5, 1.75, 2), (12, 16, 20)
(3, 3.5, 4), (1.5, 1.75, 2), (12, 16, 20)

test2 – Блокнот

Файл Правка Формат Вид Справка

1 2 3
3 4 5
6 7 8
2 4 9
6 8 10
3 3.5 4
1.5 1.75 2
12 16 20

3. Тест 3. Поиск всех троек компланарных векторов.

```
C:\Users\Sabiyat\Documents\GitHub\coplanar-vectors>task3.exe tests/input/test3.txt tests/output/answer3.txt
Coplanar vectors:
1. (1, -1, 2), (0, 1, -1), (2, -2, 4)
2. (1, -1, 2), (0, 1, -1), (3, -4, 7)
3. (1, -1, 2), (2, -2, 4), (1, 2, 3)
4. (1, -1, 2), (2, -2, 4), (1, 1, 1)
5. (1, -1, 2), (2, -2, 4), (1, 2, 1)
6. (1, -1, 2), (2, -2, 4), (2, -1, 2)
7. (1, -1, 2), (2, -2, 4), (1, 2, -3)
8. (1, -1, 2), (2, -2, 4), (3, -4, 7)
9. (1, -1, 2), (2, -2, 4), (2, -2, 3)
10. (1, -1, 2), (2, -2, 4), (4, 0, 6)
11. (1, -1, 2), (2, -2, 4), (-7, -7, 7)
12. (1, -1, 2), (1, 1, 1), (4, 0, 6)
13. (0, 1, -1), (2, -2, 4), (3, -4, 7)
14. (0, 1, -1), (2, -1, 2), (2, -2, 3)
15. (2, -2, 4), (1, 1, 1), (4, 0, 6)
16. (1, 2, 3), (1, 2, 1), (1, 2, -3)
17. (1, 1, 1), (1, 2, 1), (2, -1, 2)
18. (2, -1, 2), (1, 2, -3), (3, -4, 7)

C:\Users\Sabiyat\Documents\GitHub\coplanar-vectors>
```

test3 – Блокнот	answer3 – Блокнот			
Файл	Правка	Формат	Вид	Справка
1 -1 2	(1, -1, 2), (0, 1, -1), (2, -2, 4)			
0 1 -1	(1, -1, 2), (0, 1, -1), (3, -4, 7)			
2 -2 4	(1, -1, 2), (2, -2, 4), (1, 2, 3)			
1 2 3	(1, -1, 2), (2, -2, 4), (1, 1, 1)			
1 1 1	(1, -1, 2), (2, -2, 4), (1, 2, 1)			
1 2 1	(1, -1, 2), (2, -2, 4), (2, -1, 2)			
2 -1 2	(1, -1, 2), (2, -2, 4), (1, 2, -3)			
1 2 -3	(1, -1, 2), (2, -2, 4), (3, -4, 7)			
3 -4 7	(1, -1, 2), (2, -2, 4), (2, -2, 3)			
2 -2 3	(1, -1, 2), (2, -2, 4), (4, 0, 6)			
4 0 6	(1, -1, 2), (2, -2, 4), (-7, -7, 7)			
-7 -7 7	(1, -1, 2), (1, 1, 1), (4, 0, 6)			
	(0, 1, -1), (2, -2, 4), (3, -4, 7)			
	(0, 1, -1), (2, -1, 2), (2, -2, 3)			
	(2, -2, 4), (1, 1, 1), (4, 0, 6)			
	(1, 2, 3), (1, 2, 1), (1, 2, -3)			
	(1, 1, 1), (1, 2, 1), (2, -1, 2)			
	(2, -1, 2), (1, 2, -3), (3, -4, 7)			

4. Тест 4. Ввод неверного количества аргументов.

```
C:\Users\Sabiyat\Documents\GitHub\coplanar-vectors>task3.exe
Wrong number of args.
C:\Users\Sabiyat\Documents\GitHub\coplanar-vectors>task3.exe test.txt
Wrong number of args.
```


6. 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.

7. 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]);
}

```