уМИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**

**(национальный исследовательский университет)**

Институт № 7 «Интеллектуальные и робототехнические системы»

Кафедра 703

**Курсовая работа по курсу**

**«Технология разработки программного обеспечения»**

Работу выполнил

Cтудент группы М7О-406С-20

Кокурин Константин Алексеевич\_\_\_\_\_\_\_\_\_\_\_\_

Работу принял

Старший преподаватель кафедры 703

Барчев Николай Борисович\_\_\_\_\_\_\_\_\_\_\_\_

Дата сдачи отчета: \_\_\_\_\_\_\_\_\_\_\_\_\_

**Москва**

**2024**

Оглавление

[1.Задание 3](#_Toc153909957)

[2. Псевдокод алгоритмов решения поставленных задач 4](#_Toc153909958)

[3. Сведения о программной реализации 4](#_Toc153909959)

[4. Инструкция пользователя 8](#_Toc153909960)

[5. Тестирование программы 8](#_Toc153909961)

[6. Листинг 13](#_Toc153909962)

# Задание

Составить описание класса для объектов-векторов, задаваемых координатами концов в трехмерном пространстве. Обеспечить операции сложения и вычитания векторов с получением нового вектора (суммы или разности), вычисления скалярного произведения двух векторов, длины вектора, косинуса угла между векторами.

# 2. Псевдокод алгоритмов решения поставленных задач

**Функция записи координат вектора**

НАЧАЛО РАБОТЫ ФУНКЦИИ

Пока флаг равно ИСТИНА

Записать координату

Если оператор cin выдает ошибку

Вывести сообщение об ошибке

Очистить поток

Продолжить ввод следующей координаты

КОНЕЦ РАБОТЫ ФУНКЦИИ

**ФУНКЦИЯ ВЫВОДА КООРДИНАТ ВЕКТОРА**

НАЧАЛО РАБОТЫ ФУНКЦИИ

Вывести в консоль открытую скобку

Для каждой координаты вектора

Вывести координату

Поставить запятую

Вывести в консоль закрытую скобку

КОНЕЦ РАБОТЫ ФУНКЦИИ

**ФУНКЦИЯ СЛОЖЕНИЯ ДВУХ ВЕКТОРОВ**

НАЧАЛО РАБОТЫ ФУНКЦИИ

Для каждой координаты вектора

Сложить координату первого вектора с координатой второго вектора

КОНЕЦ РАБОТЫ ФУНКЦИИ

**ФУНКЦИЯ ВЫЧИТАНИЯ ДВУХ ВЕКТОРОВ**

НАЧАЛО РАБОТЫ ФУНКЦИИ

Для каждой координаты вектора

Вычесть координату первого вектора с координатой второго вектора

КОНЕЦ РАБОТЫ ФУНКЦИИ

**ФУНКЦИЯ ПЕРЕМНОЖЕНИЯ ДВУХ ВЕКТОРОВ**

НАЧАЛО РАБОТЫ ФУНКЦИИ

Для каждой координаты вектора

Умножить координату первого вектора с координатой второго вектора

КОНЕЦ РАБОТЫ ФУНКЦИИ

**ФУНКЦИЯ ВЫЧИСЛЕНИЯ СКАЛЯРНОГО ПРОИЗВЕДЕНИЯ ДВУХ ВЕКТОРОВ**

НАЧАЛО РАБОТЫ ФУНКЦИИ

Для каждой координаты вектора

Добавить в параметр, отвечающий за значение скалярно произведения двух векторов произведение координаты первого и второго векторов

КОНЕЦ РАБОТЫ ФУНКЦИИ

**ФУНКЦИЯ ВЫЧИСЛЕНИЯ ДЛИНЫ ВЕКТОРА**

НАЧАЛО РАБОТЫ ФУНКЦИИ

Для всех координат в векторе координат

Перемножить координаты вектора друг на друга

Вычислить квадратный корень из получившегося числа

КОНЕЦ РАБОТЫ ФУНКЦИИ

**ФУНКЦИЯ ВЫЧИСЛЕНИЯ УГЛА МЕЖДУ ВЕКТОРАМИ**

НАЧАЛО РАБОТЫ ФУНКЦИИ

Определить косинус угла по формуле косинуса угла между векторами

Определить аркосинус от получившегося значения

Перевести значение из радиан в градусы

КОНЕЦ РАБОТЫ ФУНКЦИИ

# 3.Сведения о программной реализации

При написании программы использовался язык программирования С++. Стандарт языка ISO C++ 14 Standard. Среда разработки – Microsoft Visual Studio 2020, версия 17.7.5. Минимальные системные требования к программе: OC Windows 10, процессор Intel Core i3-10105F c частотой 3.7 ГГц, оперативная память 4 Гб, 100 Мб свободного места на носителе информации.

В качестке входных данных выступают координаты двух векторов, введенные пользователем через консоль.

В качестве выходных данных выступают результаты математических операций, выведенные в консоль.

Краткое описание спроектированных подпрограмм:

Табл. 1 программные единицы файла “ kursovaya\_vectori.cpp”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название функции | Входные параметры | Выходные параметры | Возвращаемые значения | Краткое описание функции |
| void setCoordinates | - | - | - | Данная функция реализует запись координат вектора из консоли |
| void print | - | - | - | Данная функция реализует вывод координат векторов в консоль |
| Vector3D add | Vector3D other – второй вектор | - | Vector3D result – результат суммирования | Данная функция вычисляет сумму двух векторов |
| Vector3D subtract | Vector3D other – второй вектор | - | Vector3D result – результат разности | Данная функция вычисляет разность первого и второго векторов |
| Vector3D multiply | Vector3D other – второй вектор | - | Vector3D result – результат перемножения | Данная функция вычисляет произведение первого и второго векторов |
| double ScalarProduct | Vector3D other – второй вектор | - | Vector3D result – результат перемножения | Данная функция вычисляет скалярное произведение двух векторов |
| double magnitude | - | - | sqrt(sumOfSquares) – квадратный корень из числа | Данная функция вычисляет длину вектора |
| Double angle | Vector3D other – второй вектор | - | angleDeg – угол в градусах | Данная функция вычисляет угол между двумя векторами |

# 

# 4.Инструкция пользователя

Данное руководство пользователю кратко описывает базовые принципы работы программы. Программа предназначена для расчета суммы, разности, произведения, скалярного произведения векторов, а также вычисления угла между ними.

При запуске программы выводится информация о создателе программы и ее функционале

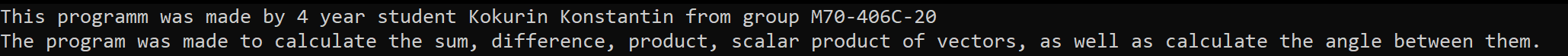


Рис.1 – информация о создателе программы и ее функционале

Далее выводится сообщение с просьбой ввести координаты первого вектора. После ввода координат первого вектора, выводится сообщение с просьбой ввести координаты второго вектора

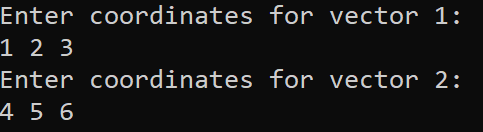


Рис.2 – ввод координат двух векторов

После ввода координат векторов выводится все результаты вычислений

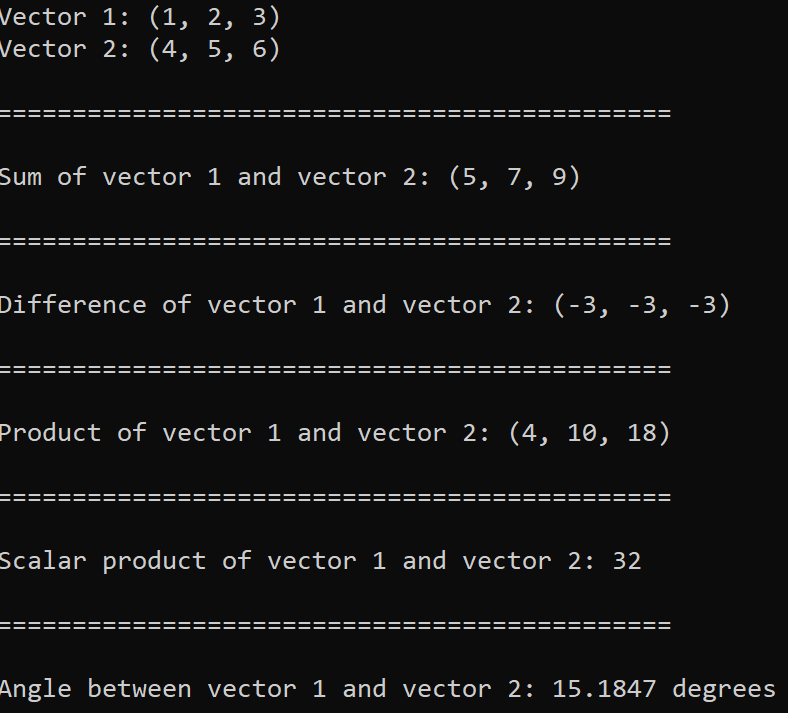


Рис.3 – результат работы программы

# 5.Тестирование программы

# 1)Проверки работы программы при вводе символов в поле ввода координат вектора

# 1.1)Проверка работы программы при корректном вводе координат вектора:

# Введем “1 2 3”

# Ожидаемое поведение: Программа отработает корректно и попросит ввести координаты второго вектора

# 

# Рис.4 – Корректный ввод координат вектора

# Ожидаемое и фактическое поведения совпадают

# 1.2) Проверка работы программы при некорректном вводе координат вектора:

# Введем “qwe 5 4”

# Ожидаемое поведение: Программа отработает корректно и выведет сообщение об ошибке в консоль

# 

# Рис.5 – Некорректный ввод координат вектора

# Ожидаемое и фактическое поведения совпадают

# 1.3) Проверка работы программы при некорректном вводе координат вектора

# Введем “1a 2 3”

# Ожидаемое поведение: Программа отработает корректно и, пропустив букву, после ввода второго вектора выведет

# 

# Рис.6 – Некорректный ввод командного сигнала

# Ожидаемое и фактическое поведения совпадают

# 1.4) Проверка работы программы при некорректном вводе координат вектора

# Введем “ 5

# 7

# 9”

# Ожидаемое поведение: Программа отработает корректно и попросит ввести координаты второго вектора и, после ввода второго вектора, выведет первый вектор как

# Vector 1: (5, 7, 9)

# 

# Рис.7 – некорректный ввод координат вектора

# 

# Рис. 8 - некорректный ввод координат вектора

# Ожидаемое и фактическое поведения совпадают

# 1.5) Проверка работы программы корректном вводе координат векторов

# Для координат первого вектора введем “1,1,1”

# Для координат второго вектора введем “2/3/4”

# Ожидаемое поведение: Программа отработает корректно и выведет результаты вычислений

# 

# Рис.9 – корректный ввод координат вектора

# Ожидаемое и фактическое поведения совпадают

# 1.6) Проверка работы программы вводе специальных символов

# Введем “$”

# Ожидаемое поведение: Программа отработает корректно и выведет сообщение об ошибке в консоль

# 

# Рис.10 – некорректный ввод координат вектора

# Ожидаемое и фактическое поведения совпадают

# 6. Листинг

**Kursovaya\_vectori.cpp**

#include <iostream>

#include <vector>

#include <cmath>

#include <limits.h> //для игноров

class Vector3D {

private:

std::vector<double> coordinates;

public:

Vector3D() {}

// Метод для установки координат вектора

void setCoordinates() {

double coord;

bool flag = true;

while (flag) {

coordinates.clear();

std::cin >> coord;

if (std::cin.fail()) {

std::cout << "one of your coordinates is wrong. Pleasr try again\n";

std::cin.clear(); //сбрасываем failbit

std::cin.ignore(static\_cast<std::streamsize>(std::numeric\_limits<int>::max()), '\n'); //очищает поток

continue;

}

std::cin.ignore(1);

coordinates.push\_back(coord);

std::cin >> coord;

if (std::cin.fail()) {

std::cout << "one of your coordinates is wrong. Pleasr try again\n";

std::cin.clear();

std::cin.ignore(static\_cast<std::streamsize>(std::numeric\_limits<int>::max()), '\n');

continue;

}

std::cin.ignore(1);

coordinates.push\_back(coord);

std::cin >> coord;

if (std::cin.fail()) {

std::cout << "one of your coordinates is wrong. Pleasr try again\n";

std::cin.clear();

std::cin.ignore(static\_cast<std::streamsize>(std::numeric\_limits<int>::max()), '\n');

continue;

}

coordinates.push\_back(coord);

flag = false;

}

std::cin.ignore(static\_cast<std::streamsize>(std::numeric\_limits<int>::max()), '\n');

}

// Метод для вывода координат вектора

void print() {

std::cout << "(";

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

std::cout << coordinates[i];

if (i != coordinates.size() - 1) {

std::cout << ", ";

}

}

std::cout << ")" << std::endl;

}

// Метод для сложения двух векторов

Vector3D add(Vector3D other) {

Vector3D result;

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

result.coordinates.push\_back(coordinates[i] + other.coordinates[i]);

}

return result;

}

// Метод для вычитания одного вектора из другого

Vector3D subtract(Vector3D other) {

Vector3D result;

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

result.coordinates.push\_back(coordinates[i] - other.coordinates[i]);

}

return result;

}

// Метод для умножения двух векторов

Vector3D multiply(Vector3D other) {

Vector3D result;

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

result.coordinates.push\_back(coordinates[i] \* other.coordinates[i]);

}

return result;

}

// Метод для вычисления скалярного произведения двух векторов

double ScalarProduct(Vector3D other) {

double product = 0.0;

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

product += coordinates[i] \* other.coordinates[i];

}

return product;

}

// Метод для вычисления длины вектора

double magnitude() {

double sumOfSquares = 0.0;

for (double coordinate : coordinates) {

sumOfSquares += coordinate \* coordinate \* coordinate;

}

return std::sqrt(sumOfSquares);

}

// Метод для вычисления угла между двумя векторами в градусах

double angle(Vector3D other) {

double dotProd = ScalarProduct(other);

double mag1 = magnitude();

double mag2 = other.magnitude();

double angleRad = std::cos(dotProd / (mag1 \* mag2));

angleRad = std::acos(angleRad);

double angleDeg = angleRad \* 180.0 / 3.1415; // Преобразование радиан в градусы

return angleDeg;

}

void cleaner() {

coordinates.clear();

}

};

int main() {

Vector3D vec1, vec2;

std::cout << "This programm was made by 4 year student Kokurin Konstantin from group M70-406C-20\n This proggram was made for finding angle between two vectors,multyplying two vector and etc\n";

char choice = 'y';

while (choice == 'y' || choice == 'Y') {

choice = 0;

std::cout << "Enter coordinates for vector 1:" << std::endl;

vec1.setCoordinates();

std::cout << "Enter coordinates for vector 2:" << std::endl;

vec2.setCoordinates();

std::cout << "Vector 1: ";

vec1.print();

std::cout << "Vector 2: ";

vec2.print();

std::cout << "\n=============================================\n\n";

Vector3D sum = vec1.add(vec2);

std::cout << "Sum of vector 1 and vector 2: ";

sum.print();

std::cout << "\n=============================================\n\n";

Vector3D difference = vec1.subtract(vec2);

std::cout << "Difference of vector 1 and vector 2: ";

difference.print();

std::cout << "\n=============================================\n\n";

Vector3D product = vec1.multiply(vec2);

std::cout << "Product of vector 1 and vector 2: ";

product.print();

std::cout << "\n=============================================\n\n";

double dotProduct = vec1.ScalarProduct(vec2);

std::cout << "Scalar product of vector 1 and vector 2: " << dotProduct <<"\n";

std::cout << "\n=============================================\n\n";

double angle = vec1.angle(vec2);

std::cout << "Angle between vector 1 and vector 2: " << angle << " degrees\n\n";

vec1.cleaner();

vec2.cleaner();

std::cout << "Do you want to enter new coordinates? (y/n): ";

std::cin >> choice;

}

std::cout << "Thanks for using. See ya!";

return 0;

}