ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»  
(РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

ОТЧЁТ  
О ЛАБОРАТОРНОЙ РАБОТЕ №1

По дисциплине «Языки программирования»

ВАРИАНТ 17

Выполнил: ст. гр. ТКИ - 141

Пылаева Светлана Алексеевна

Проверил: к.т.н., доц. Васильева М. А.

(Проверил: к.т.н, доц. Балакина Е. П.)

Москва 2024

1. Формулировка задания

Разработать библиотеку классов.

Вектор в трехмерном пространстве. Предусмотреть задание вектора двумя точками класса Точка (класс Точка задается тройкой чисел (абсцисса, ордината, аппликата)). Переопределить операторы сдвига влево, равенства и

неравенства векторов, сложения векторов, вычитание векторов, умножение на скаляр.

Для класса Точка определить все необходимые для дальнейшего

использования операторы (операторы сравнения, сдвига влево и вправо,

сложение и вычитание Точки и вещественного числа).

1. Диаграмма UML

Диаграмма UML для классов Point и Vector представлена ниже (Рисунок 1).

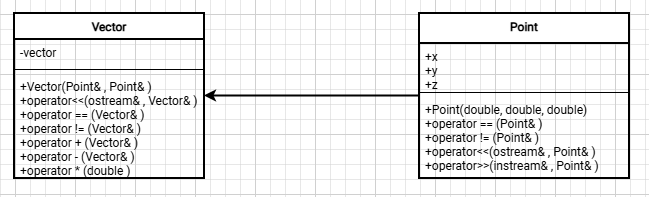


Рисунок 1 – Диаграмма UML

1. Текст программы на языке C++

main.cpp:

#include <iostream>

#include "..\Domain\Point.h"

#include "..\Domain\Vector.h"

int main()

{

Point::Point A(1, 1, 1), B(2, 4, 1);

std::cout << A << std::endl << B << std::endl;

Vector::Vector AB(A, B);

std::cout << AB << std::endl;

Vector::Vector BA(B, A);

std::cout << BA << std::endl;

std::cout << AB + BA << std::endl << AB - BA << std::endl << AB \* 3;

return 0;

}

Point.h:

#pragma once

#include <iostream>

namespace Point

{

struct Point

{

/\*\*

\*@brief Абсцисса точки.

\*/

double x = 0;

/\*\*

\*@brief Ордината точки.

\*/

double y = 0;

/\*\*

\*@brief Аппликата точки.

\*/

double z = 0;

/\*\*

\*@brief Создает экземпляр класса Point.

\*@param Абсцисса точки.

\*@param Ордината точки.

\*@param Аппликата точки.

\*/

Point(const double abscissa = 0, const double ordinate = 0, const double applicate = 0);

/\*

\*@brief оператор "==" для класса Point.

\*/

bool operator == (const Point& point)const;

/\*

\*@brief оператор "!=" для класса Point.

\*/

bool operator != (const Point& point)const;

/\*

\*@brief оператор "<<" для класса Point.

\*/

friend std::ostream& operator<<(std::ostream& output, const Point& point);

/\*

\*@brief оператор ">>" для класса Point.

\*/

friend std::istream& operator>>(std::istream& input, Point& point);

};

}

Point.cpp:

#include <iostream>

#include "Point.h"

#include "cmath"

Point::Point::Point(const double abscissa, const double ordinate, const double applicate) : x{ abscissa }, y{ ordinate }, z{ applicate }

{

x = abscissa;

y = ordinate;

z = applicate;

}

bool Point::Point::operator==(const Point& point) const

{

return(std::abs(this->x - point.x)<= std::numeric\_limits<double>::epsilon() && std::abs(this->y - point.y) <= std::numeric\_limits<double>::epsilon()&& std::abs(this->z - point.z) <= std::numeric\_limits<double>::epsilon());

}

bool Point::Point::operator!=(const Point& point) const

{

return !(\*this == point);

}

std::ostream& Point::operator<<(std::ostream& output, const Point& point)

{

output << point.x << point.y << point.z ;

return output;

}

std::istream& Point::operator>>(std::istream& input, Point& point)

{

input >> point.x >> point.y >> point.z;

return input;

}

Vector.h:

#pragma once

#include "Point.h"

namespace Vector

{

class Vector

{

private:

/\*

\*@brief Вектор

\*/

Point::Point vector;

public:

/\*\*

\*@brief Конструктор для класса вектор по двум точкам.

\*@param start-точка начала вектора.

\*@param end-точка конца вектора.

\*/

Vector(const Point::Point& start, const Point::Point& end);

/\*

\*@brief оператор "<<" для класса Vector.

\*/

friend std::ostream& operator<<(std::ostream& output, const Vector& vector);

/\*

\*@brief оператор "==" для класса Vector.

\*/

bool operator == (const Vector& vector)const;

/\*

\*@brief оператор "!=" для класса Vector.

\*/

bool operator != (const Vector& vector)const;

/\*

\*@brief оператор "+" для класса Vector.

\*@return Возвращает вектор, координаты которого равны сумме других векторов.

\*/

Vector operator + (const Vector& other\_vector);

/\*

\*@brief оператор "-" для класса Vector.

\*@return Возвращает вектор, координаты которого равны разнице других векторов.

\*/

Vector operator - (const Vector& other\_vector);

/\*

\*@brief оператор "\*" для класса Vector.

\*@return Возвращает вектор, координаты которого умножены на скаляр.

\*/

Vector operator \* (const double value);

};

}

Vector.cpp:

#include "Vector.h"

#include "cmath"

Vector::Vector::Vector(const Point::Point& start, const Point::Point& end)

{

Point::Point temporary(end.x - start.x, end.y - start.y, end.z - start.z);

this->vector = temporary;

}

std::ostream& Vector::operator<<(std::ostream& output, const Vector& vector)

{

output << vector.vector;

return output;

}

bool Vector::Vector::operator==(const Vector& vector) const

{

return(this->vector == vector.vector);

}

bool Vector::Vector::operator!=(const Vector& vector) const

{

return !(\*this == vector);

}

Vector::Vector Vector::Vector::operator+(const Vector& other\_vector)

{

Vector result{ \*this };

result.vector.x += other\_vector.vector.x;

result.vector.y += other\_vector.vector.y;

result.vector.z += other\_vector.vector.z;

return result;

}

Vector::Vector Vector::Vector::operator-(const Vector& other\_vector)

{

Vector result{ \*this };

result.vector.x -= other\_vector.vector.x;

result.vector.y -= other\_vector.vector.y;

result.vector.z -= other\_vector.vector.z;

return result;

}

Vector::Vector Vector::Vector::operator\*(const double value)

{

Vector result{ \*this };

result.vector.x \*= value;

result.vector.y \*= value;

result.vector.z \*= value;

return result;

}

1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 2).

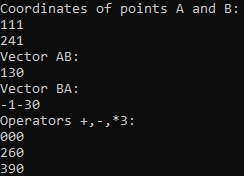


Рисунок 2 – Результат выполнения программы

1. Тесты классов

UnitTest1.cpp:

#include "pch.h"

#include "CppUnitTest.h"

#include "..\Domain\Point.h"

#include "cmath"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace Test

{

TEST\_CLASS(PointTest)

{

public:

TEST\_METHOD(OperatorEqual\_EqualData\_True)

{

Point::Point a(0, 0, 0), b(0, 0, 0);

Assert::IsTrue(a == b);

}

TEST\_METHOD(OperatorEqual\_NotEqualData\_False)

{

Point::Point a(0, 0, 0), b(1, 3, 5);

Assert::IsFalse(a == b);

}

TEST\_METHOD(OperatorNotEqual\_NotEqualData\_True)

{

Point::Point a(0, 0, 0), b(23, 40, -21);

Assert::IsTrue(a != b);

}

TEST\_METHOD(OperatorNotEqual\_EqualData\_False)

{

Point::Point a(0, 0, 0), b(0, 0, 0);

Assert::IsFalse(a != b);

}

};

}

UnitTest2.cpp:

#include "pch.h"

#include "CppUnitTest.h"

#include "..\Domain\Point.h"

#include "..\Domain\Vector.h"

#include "cmath"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace Test

{

TEST\_CLASS(VectorTest)

{

TEST\_METHOD(OperatorEqual\_EqualData\_True)

{

Point::Point a(0, 0, 0), b(0, 0, 0);

Vector::Vector first(a, b), second(a, b);

Assert::IsTrue(first == second);

}

TEST\_METHOD(OperatorEqual\_NotEqualData\_False)

{

Point::Point a(0, 0, 0), b(0, 0, 0),

c(4, 3, 5), d(3, 5, 8);

Vector::Vector first(a, b), second(c, d);

Assert::IsFalse(first == second);

}

TEST\_METHOD(OperatorNotEqual\_NotEqualData\_True)

{

Point::Point a(0, 0, 0), b(2, 3, -14),

c(4, 3, 5), d(3, 5, 8);

Vector::Vector first(a, b), second(c, d);

Assert::IsTrue(first != second);

}

TEST\_METHOD(OperatorNotEqual\_EqualData\_False)

{

Point::Point a(0, 0, 0), b(0, 0, 0);

Vector::Vector first(a, b), second(a, b);

Assert::IsFalse(first != second);

}

TEST\_METHOD(OperatorSum\_CorrectData\_True)

{

Point::Point a(0, 0, 0), b(2, 3, -14),

c(4, 3, 5), d(3, 5, 8),

exp\_s(0, 0, 0), exp\_e(1, 5, -11);

Vector::Vector first(a, b), second(c, d), expected\_vector(exp\_s, exp\_e);

Assert::IsTrue(first + second == expected\_vector);

}

TEST\_METHOD(OperatorSum\_NotCorrectData\_False)

{

Point::Point a(0, 0, 0), b(2, 3, -14),

c(4, 3, 5), d(3, 5, 8),

exp\_s(0, 0, 0), exp\_e(7, 10, 3);

Vector::Vector first(a, b), second(c, d), expected\_vector(exp\_s, exp\_e);

Assert::IsFalse(first + second == expected\_vector);

}

TEST\_METHOD(OperatorSubtraction\_CorrectData\_True)

{

Point::Point a(0, 0, 0), b(2, 3, -14),

c(4, 3, 5), d(3, 5, 8),

exp\_s(0, 0, 0), exp\_e(3, 1, -17);

Vector::Vector first(a, b), second(c, d), expected\_vector(exp\_s, exp\_e);

Assert::IsTrue(first - second == expected\_vector);

}

TEST\_METHOD(OperatorSubtraction\_NotCorrectData\_False)

{

Point::Point a(0, 0, 0), b(2, 3, -14),

c(4, 3, 5), d(3, 5, 8),

exp\_s(0, 0, 0), exp\_e(7, 10, 3);

Vector::Vector first(a, b), second(c, d), expected\_vector(exp\_s, exp\_e);

Assert::IsFalse(first - second == expected\_vector);

}

TEST\_METHOD(OperatorMultipliedToScalar\_Data\_True)

{

double v = -1.24;

Point::Point a(0, 0, 0), b(2, 3, -14),

exp\_s(0, 0, 0), exp\_e(2 \* v, 3 \* v, -14 \* v);

Vector::Vector first(a, b), expected\_vector(exp\_s, exp\_e);

Assert::IsTrue(first \* v == expected\_vector);

}

TEST\_METHOD(OperatorMultipliedToScalar\_NotCorrectData\_False)

{

double v = -1.24;

Point::Point a(0, 0, 0), b(2, 3, -14),

exp\_s(0, 0, 0), exp\_e(7 \* v, 10 \* v, 3 \* v);

Vector::Vector first(a, b), expected\_vector(exp\_s, exp\_e);

Assert::IsFalse(first \* v == expected\_vector);

}

};

}

Результаты выполнения тестов классов представлены ниже

(Рисунок 3).

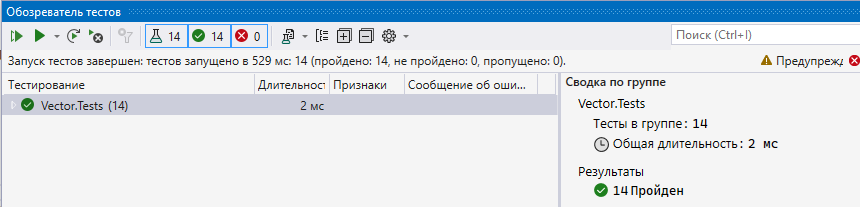


Рисунок 3 – Результат выполнения тестов

1. Отметка о выполнении задания в веб-хостинге системы контроля версий

Ниже представлено доказательство того, что задание было принято (Рисунок 4).

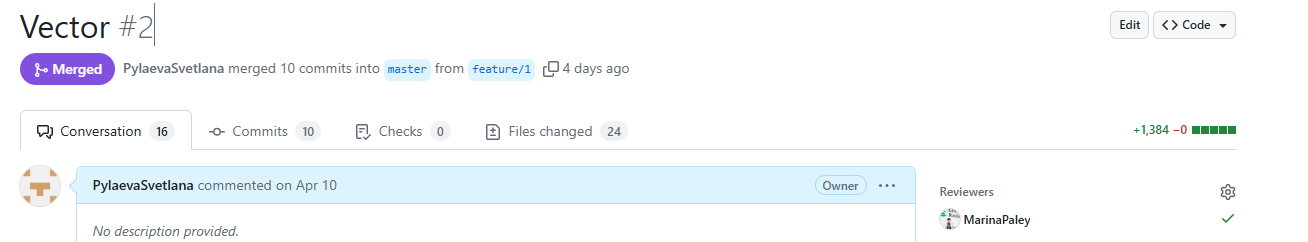


Рисунок 4 – Approve