Практическая работа №7

В контексте выполения этого задания будет продуман и реализован класс Vector2D, представляющий объект двухмерного вектора.

Свойство/метод	Описание
Vector2D()	Конструктор по умолчанию
Vector2D(float x, float y);	Конструктор вектора с координатами x и y
Vector2D& Add(Vector2D& vec);	Сложение векторов
Vector2D& Subtract(Vector2D& vec);	Разность векторов
Vector2D& Multiply(Vector2D& vec);	Произведение векторов
Vector2D& Divide(Vector2D& vec);	Частное векторов
friend Vector2D& operator+(Vector2D& v1, Vector2D& v2);	Перегрузка оператора сложения
friend Vector2D& operator-(Vector2D& v1, Vector2D& v2);	Перегрузка оператора вычитания
friend Vector2D& operator*(Vector2D& v1, Vector2D& v2);	Перегрузка оператора умножения
friend Vector2D& operator/(Vector2D& v1, Vector2D& v2);	Перегрузка оператора деления
Vector2D& operator+=(Vector2D& vec);	Перегрузка операторов сложного присваивания для суммы, разности, произведения и частного.
Vector2D& operator-=(Vector2D& vec);	
Vector2D& operator*=(Vector2D& vec);	
Vector2D& operator/=(Vector2D& vec);	
Vector2D& operator*(& i);	Перегрузка оператора для умножения на число
friend std::ostream& operator<<(std::ostream& stream, Vector2D& vec);	Перегрузка оператора вывода

Возможный вариант реализации:

```
#include "Vector2D.h"
Vector2D::Vector2D() {
	x = 0;
	y = 0;
}
Vector2D::Vector2D(float x, float y) {
	this->x = x;
	this->y = y;
}
Vector2D& Vector2D::Add(Vector2D& vec) {
	this->x += vec.x;
	this->y += vec.y;

	return *this;
}
Vector2D& Vector2D::Subtract(Vector2D& vec) {
	this->x -= vec.x;
	this->y -= vec.y;

	return *this;
}
Vector2D& Vector2D::Multiply(Vector2D& vec) {
	this->x *= vec.x;
	this->y *= vec.y;

	return *this;
}
Vector2D& Vector2D::Divide(Vector2D& vec) {
	this->x /= vec.x;
	this->y /= vec.y;

	return *this;
}

Vector2D& operator+(Vector2D& v1, Vector2D& v2) {
	return v1.Add(v2);
}
Vector2D& operator-(Vector2D& v1, Vector2D& v2) {
	return v1.Subtract(v2);
}
Vector2D& operator*(Vector2D& v1, Vector2D& v2)
{
	return v1.Multiply(v2);
}
Vector2D& operator/(Vector2D& v1, Vector2D& v2) {
	return v1.Divide(v2);
}

Vector2D& Vector2D::operator+=(Vector2D& vec) {
	return this->Add(vec);
}
Vector2D& Vector2D::operator-=(Vector2D& vec) {
	return this->Subtract(vec);
}
Vector2D& Vector2D::operator*=(Vector2D& vec) {
	return this->Multiply(vec);
}
Vector2D& Vector2D::operator/=(Vector2D& vec) {
	return this->Divide(vec);
}

Vector2D& Vector2D::operator*(int& i) {
	this->x *= i;
	this->y *= i;

	return *this;
}

std::ostream& operator<<(std::ostream& stream, Vector2D& vec) {
	stream << "(" << vec.x << ";" << vec.y << ")";
	return stream;
}
```