Лабораторная работа номер 8

Основные требования:

- каждая функция должна иметь docstring - множ-й комментарий (по типу что делает данная функция)
- написание функций должно быть компактным (не в 100 строчек кода)
- именования функций должны быть нормальными и четко отражать смысл самой функции
- именование файлов разбиения дожно соответствовать задаче (не приветствуется например такое: ufhiqh.cpp, ggg.h)
- код должен соответствовать стандарту языка C++.

Задание на лабораторную работу "Перегрузка операций с использованием шаблонных классов":

Необходимо написать **шаблонный класс "Vector"**, который реализует работу с векторами. Класс должен иметь следующие методы:

1. **Конструктор без параметров** - создает пустой вектор.
2. **Конструктор с параметром** - создает вектор заданной размерности.
3. **Конструктор копирования** - создает копию вектора.
4. **Деструктор** - освобождает память, занятую вектором.
5. Метод **size**() - возвращает размер вектора.
6. Метод **push_back**() - добавляет элемент в конец вектора.
7. Метод **pop_back**() - удаляет последний элемент вектора.
8. Метод **clear**() - очищает вектор.
9. Метод **contains**() - проверка на сущ-е элемента вектора.
10. Метод **operator**[] - перегруженный оператор доступа к элементу вектора.
11. Метод **operator+** - перегруженный оператор сложения двух векторов.

12. Метод **operator-** - перегруженный оператор вычитания двух векторов.
13. Метод **operator*** - перегруженный оператор умножения вектора на скаляр.
14. Метод **operator/** - перегруженный оператор деления вектора на скаляр.
15. Метод **operator==** - перегруженный оператор сравнения двух векторов на равенство.
16. Метод **operator!=** - перегруженный оператор сравнения двух векторов на неравенство.
17. Метод operator<< - перегруженный оператор вывода вектора в поток. После написания класса необходимо протестировать его работу на различных типах данных. (int, double, char* (или std::string), bool)

Пример создания класса Vector

```
template<typename T>
class Vector{
private:
  T* vector;
  size_t _size; // размер вектора
public:
  Vector(size_t size = 0) : _size(size) {}
...
};
```