Лабораторная работа №7

ВНИМАНИЕ!!!!! Эта лабораторная работа будет тестироваться автоматически. Сдавать нужно только .h файл с *полной* Вашей реализацией класса Vector. Названия всех методов(кроме Ваших вспомогательных) в Вашем коде должны ПОЛНОСТЬЮ совпадать с названием этих же методов в лабораторной. За несоблюдение этих требований будет снижаться оценка(при каждой новой попытке сдать лабу).

При сдаче лабораторной работы файлы, которые будут заливаться на портал, прошу называть следующим образом: **ivanov_7lab.cpp, ivanov_7lab.zip** и т.п. Также **при наличии нескольких файлов** убедительная просьба сначала **создать папку для этих файлов, а затем её архивировать**.

В рамках данной лабораторной работы вам предстоит написать собственный **класс Vector**, который будет являться упрощённым аналогом шаблонного класса **std::vector**. В этой лабораторной запрещено использовать стандартные контейнеры и умные указатели, поэтому аккуратно работайте с памятью.

В классе **Vector** будем хранить элементы типа **int**, а также в нём есть:

- Конструктор по умолчанию.
- Конструктор, принимающий размер вектора и заполняющий его нулями.
- Конструктор, принимающий список инициализации **std::intitializer_list<int>** — это позволит создать вектор следующим образом:

```
Vector v{1, 2, 3, 4, 5};
```

Листинг 1.1: Создание элемента класса Vector с помощью std::initializer_list

- Конструкторы копирования и перемещения.
- Оператор присваивания: копирующий и перемещающий.
- Деструктор.
- Метод ***Swap()***, который принимает другой вектор по ссылке и меняет содержимое текущего вектора с ним.

- Операторы индексирования: константный и нет. Последний должен позволять *менять содержимое контейнера по индексу*.
- Метод ***Size()***, возвращающий число элементов в контейнере.
- Метод ***Capacity()***, возвращающий текущее число выделенных ячеек памяти под вектор.
- Метод ***PushBack()***, который добавляет элемент в конец вектора. Если при этом память, которая выделена для вектора, заполнена, то выполните *реаллокацию*: выделите массив вдвое большего размера, скопируйте (или переместите) туда элементы, после чего удалите старый массив. В этом случае **capacity** также должна увеличится вдвое.
- Метод ***PopBack()***, который удаляет последний элемент вектора. Сужать вектор при этом не нужно, должен измениться только **size**.
- Метод ***Clear()***, который делает контейнер пустым, *очищая его*. Аналогично, сужать вектор при этом не нужно, **size** должен стать нулевым.
- Метод ***Reserve()***, принимающий новое значение **capacity**, что позволяет зарезервировать место в векторе. Если текущая **capacity** не мешьне заданного, то метод ничего не должен делать. В ином случае выполните реаллокацию в массив размера **capacity**.
- Оператор ввода >>. При попытке ввода в непустой контейнер должно выбрасываться исключение ***std::length_ error***.
- Оператор вывода <<. Оператор вывода должен отображать вектор следующим образом:

```
Vector v{1, 2, 3, 4, 5};
std::cout << v; // вывод на консоль [1, 2, 3, 4, 5]

Vector v1;
std::cout << v; // вывод на консоль []
```

Листинг 1.2: Пример вывода вектора на консоль

Обратите внимание, что эти операторы *не являются членами класса*, а определяются вне класса как обычные функции.