

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ
по лабораторной работе №3
по дисциплине «Объектно-ориентированное программирование»
Тема: Контейнеры

Студент гр. 7303

Швец А.А.

Преподаватель

Размочаева Н.В.

Санкт-Петербург

2019

Цель работы.

Реализовать базовые методы для контейнеров вектор и список, по поведению аналогичные методам для контейнеров `std::vector` и `std::list`.

Задание.

Реализовать конструкторы, деструктор, операторы присваивания, функцию `assign`, функцию `resize`, функцию `erase`, функцию `insert` и функцию `push_back` для контейнера вектор. Поведение реализованных функций должно быть таким же, как у класса `std::vector`.

Реализовать список со следующими функциями: `push_front`, `push_back`, `front`, `back`, `pop_front`, `pop_back`, `clear`, `empty`, деструктор, конструктор копирования, конструктор перемещения, оператор присваивания, `insert`, `erase`, а также итератор для списка с операторами: `=`, `==`, `!=`, `++` (постфиксный и префиксный), `*`, `->`. Поведение реализованных функций должно быть таким же, как у класса `std::list`.

Ход работы.

Реализация вектора.

1. Были реализованы: конструктор от размера, конструктор от итераторов на начало и конец, конструктор от списка инициализации, конструктор копирования, конструктор перемещения, и деструктор, очищающий выделенную под массив данных память.

2. Были реализованы операторы присваивания и метод `assign`.

3. Были реализованы методы: `resize`, принимающий новый размер вектора, и изменяющий его содержимое аналогично библиотечной функции, и два метода `erase`, первый принимающий позицию элемента для удаления, второй принимающий позицию первого элемента для удаления, и позицию последнего элемента для удаления. В первом случае метод удаляет элемент и возвращает следующий за ним, во втором метод удаляет все элемента в диапазоне `[first; last)`, и возвращает позицию элемента за последним удаленным.

4. Были реализованы методы `push_back` и `insert`. Первый - вставляет переданное значение в конец вектора. Второй метод имеет две реализации. В первом случае метод вставляет значение перед переданной позицией, во втором вставляет диапазон значений перед позицией. Метод в первом случае возвращает позицию вставленного элемента, во втором – позицию первого вставленного элемента.

Реализация списка.

1. Были реализованы методы `push_back` (добавление в конец), `push_front` (добавление в начало), `front` (получение значения головы списка), `back` (получение значения конца списка), `pop_back` (удаление элемента из конца списка), `pop_front` (удаление элемента из начала списка), `clear` (очистка списка), `empty` (проверка размера).

2. Были реализованы: деструктор, конструктор копирования, конструктор перемещения, оператор присваивания.

3. Был реализован однонаправленный итератор для списка, включающий в себя операторы `=`, `==`, `!=`, `++` (постфиксный и префиксный), `*`, `->`. Поведение реализованных функций соответствует `std::list`.

4. Были реализованы методы `insert` и `erase` с использованием итераторов. `Insert` вставляет значение перед переданной позицией, и возвращает итератор, указывающий на вставленное значение. `Erase` удаляет элемент в переданной позиции и возвращает итератор, следующий за последним удаленным элементом.

Вывод.

При выполнении данной лабораторной работы были реализованы основные функции для работы контейнерами вектор и список, такие как: как вставка в произвольное место, удаление произвольного элемента, изменение размера, необходимые конструкторы, деструкторы, итераторы для работы с этими контейнерами.