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

ОТЧЕТ по лабораторной работе №3

по дисциплине «Объектно-ориентированное программирование»

ТЕМА: КОНТЕЙНЕРЫ.

Студент гр. 7382 Ленковский В.В.

Преподаватель Жангиров Т.М.

Санкт-Петербург

2019

Цель работы:

Изучить реализация контейнеров list и vector в языке программирования C++. Протестировать полученные реализации на практике.

Задача:

Реализовать конструктор, деструктор , операторы присваивания, функцию assign, функцию resize, функцию earse, функцию insert и функцию push_back. Поведение реализованных функций должно быть таки же как и у std::vector. Реализовать список с функциями: вставка элемента в голову, вставка элемента в хвост, получение элемента из головы, получение элемента из хвоста, удаление из головы, из хвоста, очистка списка, проверка размера, деструктор, конструктор копирования, конструктор перемещения, оператор присваивания, insert, earse, а так же итераторы для списка: =, ==, !=, ++(постфиксный и префексный), *, ->. Поведение реализованных функций должно быть таким же, как у класса std::list.

Ход работы:

- **List**.

  В ходе реализации list были созданы следующие функции:

1. Функции вставки элемента в голову и в хвост. Принимает на вход элемент и помещает его в вектор.
2. Функции получение элемента из головы и из хвоста. Возвращает элемент из головы или из хвоста.
3. Функции удаления из головы, удаления из хвоста. Совершает удаление элемента из начала или конца списка.
4. Функции очистки списка, проверки размера.
5. Деструктор, конструктор копирования, конструктор перемещения, оператор присваивания.

6. Операторы для итератора списка: =, ==, !=, ++, *, ->.
7. Функции удаления элемента и вставка элемента в произвольное место. Получает на вход элемент и помещает его в заданное место в массиве. Так же имеет возможно удалить элемент из заданного положения.

- **Vector.**

В ходе реализации vector были созданы следующие функции:

1. Конструкторы и деструктор для вектора.
2. Реализованные конструкторы включают в себя – конструктор копирования, присваивания и перемещения.
3. Оператор присваивания и функция assign.
4. Функции изменения размера и стирания элементов в массиве (resize, erase).
5. Resize – принимает на вход необходимый размер вектора, который будет присвоен текущему. Erase – может принимать как одну переменную – индекс, начиная с которого произойдет очистка вектора, так из пару переменных – интервал в векторе, которой очистится.

Результаты тестирования программы

1) Vector:

Входные данные:

```
stepik::vector<int> Vec = {11,23,44,12,1};
std::cout << "Vector:" << std::endl;
for(int i = 0; i<Vec.size(); i++) std::cout << Vec[i] << " ";
std::cout << std::endl;

std::cout << "Copy:" << std::endl;
stepik::vector<int> newVec (Vec.begin(), Vec.end());
for(int i = 0; i<newVec.size(); i++) std::cout << newVec[i] << " ";
std::cout << std::endl;

std::cout << "Resize:" << std::endl;
newVec.resize(7);
newVec[5]=23; newVec[6]=213;
for(int i = 0; i<newVec.size(); i++) std::cout << newVec[i] << " ";
std::cout << std::endl;

std::cout << "Erase:" << std::endl;
newVec.erase(newVec.begin()+2, newVec.begin()+4);
for(int i = 0; i<newVec.size(); i++) std::cout << newVec[i] << " ";
std::cout << std::endl;

std::cout << "Insert:" << std::endl;
newVec.insert(newVec.begin()+4, 1234);
for(int i = 0; i<newVec.size(); i++) std::cout << newVec[i] << " ";
std::cout << std::endl;
```

```
std::cout << "Push back:" << std::endl;
newVec.push_back(67);
for(int i = 0; i<newVec.size(); i++) std::cout << newVec[i] << " ";
std::cout << std::endl;
```

Выходные данны:

```
Vector:
11 23 44 12 1
Copy:
11 23 44 12 1
Resize:
11 23 44 12 1 23 213
Erase:
11 23 1 23 213
Insert:
11 23 1 23 1234 213
Push back:
11 23 1 23 1234 213 67
```

2) List:

Входные данные:

```
std::cout << "List:" << std::endl;
stepik::list<int> my_list;
my_list.push_back(32);
my_list.push_back(45);
my_list.push_front(62);
my_list.push_front(13);
my_list.print();

std::cout << "Size:" << std::endl;
std::cout << my_list.size() << std::endl;

std::cout << "Pop front:" << std::endl;
my_list.pop_front();
my_list.print();

std::cout << "Pop back:" << std::endl;
my_list.pop_back();
my_list.print();

std::cout << "Insert:" << std::endl;
stepik::list_iterator<int> pos = my_list.begin(); pos++;
my_list.insert(pos, 57);
my_list.print();

std::cout << "Erase:" << std::endl;
stepik::list_iterator<int> pos1 = my_list.begin(); pos1++;
my_list.erase(pos1);
my_list.print();
```

Выходные данные:

```
List:
13 62 32 45
Size:
4
Pop front:
62 32 45
Pop back:
62 32
Insert:
62 57 32
Erase:
62 32
```

Вывод:

Таким образом, в ходе лабораторной работы была подробно изучена реализация контейнеров list и vector. Поведение реализованных функций каждого из классов совпадает с реальным поведением функций из стандартной библиотеки C++. Полученные результаты были протестированы на практике.