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. 7304 Дементьев Е.В.

Преподаватель Размочаева Н.В

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:
  - Функции вставки элемента в голову и в хвост.
    Принимает на вход элемент и помещает его в вектор.
  - Функции получение элемента из головы и из хвоста.
    Возвращает элемент из головы или из хвоста.
  - Функции удаления из головы, удаления из хвоста.
    Совершает удаление элемента из начала или конца списка.
  - Функции очистки списка, проверки размера.
  - Деструктор, конструктор копирования, конструктор перемещения, оператор присваивания.
  - Операторы для итератора списка: =, ==, !=, ++, *, ->.
  - 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.

Поведение функций такое же, как у класса std::list.

- **Vector.**

  В ходе реализации vector были созданы следующие функции:

  - Конструкторы и деструктор для вектора.
    Реализованные конструкторы включают в себя – конструктор копирования, присваивания и перемещения.
  - Оператор присваивания и функция assign.
  - Функции изменения размера и стирания элементов в массиве (resize, erase).
    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.

Поведение функций такое же, как у класса std::vector.

Результаты тестирования программы

1) Vector:

- *Входные данные:*

```
int main()
{
    vector<int> a(10);
    std::cout << "\nCreate Vector - ";
    for (auto& t : a)
    {
        t = rand() % 10;
        std::cout << t << " ";
    }

    std::cout << std::endl <<
        "------------------------------
        << std::endl;
    //==================================
    std::cout << "\nConstructor -";
    vector<int> b(a);
    for (const auto& t : b)
        std::cout << t << " ";;

    std::cout << std::endl <<
        "------------------------------
        << std::endl;
    //==================================
    vector<int> c(5);
    c = a;
    std::cout << "\nCopy -";
    for (const auto& t : c)
        std::cout << t << " ";

    std::cout << std::endl <<
        "------------------------------
        << std::endl;
    //==================================
    std::cout << "\nResize -";
    a.resize(6);
    for (const auto& t : a)
        std::cout << t << " ";

    std::cout << std::endl <<
        "------------------------------
        << std::endl;
```

```
//=========================================
std::cout << "\nErase -";
a.erase(a.end()-1);
for (const auto& t : a)
    std::cout << t << " ";

std::cout << std::endl <<
    "-----------------------------------------------"
    << std::endl;
//=========================================
std::cout << "\nErase -";
a.erase(a.begin() + 1, a.end() - 2);
for (const auto& t : a)
    std::cout << t << " ";

std::cout << std::endl <<
    "-----------------------------------------------"
    << std::endl;
//=========================================
std::cout << "\nInsert -";
a.insert(a.begin() + 1, 6);
for (const auto& t : a)
    std::cout << t << " ";

std::cout << std::endl <<
    "-----------------------------------------------"
    << std::endl;
//=========================================
std::cout << "\nInsert -";
a.insert(a.begin() + 2, b.begin() + 1, b.end() - 1);
for (const auto& t : a)
    std::cout << t << " ";

std::cout << std::endl <<
    "-----------------------------------------------"
    << std::endl;
//=========================================
std::cout << "\nPush back -";
a.push_back(13);
for (const auto& t : a)
    std::cout << t << " ";
std::cout << std::endl;
return 0;
```

- Выходные данные:

```
Create Vector - 7 9 3 8 0 2 4 8 3 9
-----------------------------------

Constructor -7 9 3 8 0 2 4 8 3 9
-----------------------------------

Copy -7 9 3 8 0 2 4 8 3 9
-----------------------------------

Resize -7 9 3 8 0 2
-----------------------------------

Resize -7 9 3 8 0 2 0 0 0 0 0 0
-----------------------------------

Erase -7 9 3 8 0 2 0 0 0 0 0
-----------------------------------

Erase -7 0 0
-----------------------------------

Insert -7 6 0 0
-----------------------------------

Insert -7 6 9 3 8 0 2 4 8 3 0 0
-----------------------------------

Push back -7 6 9 3 8 0 2 4 8 3 0 0 13
```

2) List:

- *Входные данные:*

```
int main()
{

    std::cout << "Create list:\n";
    list<int> lst;

    std::cout << "List: ";

    for (int i = 0; i < 30; i++)
        lst.push_back(rand()%20);
    lst.print();

    std::cout << std::endl <<
        "------------------------------
        << std::endl;


    std::cout << "\nPush front 222\n";
    std::cout << "List: ";
    lst.push_front(222);
    lst.print();

    std::cout << std::endl <<
        "------------------------------
        << std::endl;

    std::cout << "\nPrint size\n";
    std::cout << "List: ";
    std::cout << lst.size();

    std::cout << std::endl <<
        "------------------------------
        << std::endl;
```

```
std::cout << "\nPop back\n";
std::cout << "List: ";
lst.pop_back();
lst.print();

std::cout << std::endl <<
    "-------------------------------------
    << std::endl;

std::cout << "\nPop front\n";
std::cout << "List: ";
lst.pop_front();
lst.print();

std::cout << std::endl <<
    "-------------------------------------
    << std::endl;

std::cout << "\nClear\n";
std::cout << "List: ";
lst.clear();
lst.print();

std::cout << std::endl <<
    "-------------------------------------
    << std::endl;

std::cout << "\nPush back 7304 7 times\n";
std::cout << "List: ";
for (int i = 0; i < 7; i++)
    lst.push_back(7304);
lst.print();
```

```
    std::cout << std::endl <<
        "-------------------------------
        << std::endl;

    std::cout << "\nCopy construct\n";
    std::cout << "List: ";
    list<int> lst_2(lst);
    lst_2.print();

    std::cout << std::endl <<
        "-------------------------------
        << std::endl;

    return 0;
}
```

- *Выходные:*

```
Create list:
List: [ 7 9 13 18 10 12 4 18 3 9 0 5 12 2 7 3 7 9 0 12 3 9 9 17 0 13 19 18 16 15 ]

--------------------------------------------------------------------------------

Push front 222
List: [ 222 7 9 13 18 10 12 4 18 3 9 0 5 12 2 7 3 7 9 0 12 3 9 9 17 0 13 19 18 16 15 ]

--------------------------------------------------------------------------------

Print size
List: 31
--------------------------------------------------------------------------------

Pop back
List: [ 222 7 9 13 18 10 12 4 18 3 9 0 5 12 2 7 3 7 9 0 12 3 9 9 17 0 13 19 18 16 ]

--------------------------------------------------------------------------------

Pop front
List: [ 7 9 13 18 10 12 4 18 3 9 0 5 12 2 7 3 7 9 0 12 3 9 9 17 0 13 19 18 16 ]

--------------------------------------------------------------------------------

Clear
List:
--------------------------------------------------------------------------------

Push back 7304 7 times
List: [ 7304 7304 7304 7304 7304 7304 7304 ]

--------------------------------------------------------------------------------

Copy construct
List: [ 7304 7304 7304 7304 7304 7304 7304 ]

--------------------------------------------------------------------------------
```

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.