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.

Студентка гр.7303 _____ Дегтярева А.А

Преподаватель _____ Размочаева Н.В

г. Санкт-Петербург

2019 г.

Цель работы:

Необходимо реализовать контейнеры вектор и список. Поведение реализованных функций должно быть таким же, как у классов std::vector и std::list.

Ход работы:

Для выполнения поставленной задачи были реализованы следующие структуры данных:

1. Были дописаны следующие методы для класса vector:
   1.1. explicit vector(size_t count = 0) – создается вектор размера count;
   1.2. vector(InputIterator first, InputIterator last) – создается вектор со значениями от first до last;
   1.3. vector(std::initializer_list <Type> init) – создается вектор с заданным диапазоном значений;
   1.4. vector(const vector& other) – конструктор копирования;
   1.5. vector(vector&& other) – конструктор перемещения;
   1.6. ~vector() – деструктор;
   1.7. vector& operator=(const vector& other) – оператор присваивания копирования;
   1.8. vector& operator=(vector&& other) – оператор присваивания перемещения;
   1.9. void assign(InputIterator first, InputIterator last) – копирование в вектор значений от first до last;
   1.10. void resize(size_t count) – изменение размера вектора;
   1.11. iterator erase(const_iterator pos) – удаление элемента в заданной позиции;
   1.12. iterator erase(const_iterator first, const_iterator last) – удаление элементов от first до last;
   1.13. iterator insert(const_iterator pos, const Type& value) – вставка элемента в заданную позицию;

1.14. iterator insert(const_iterator pos, InputIterator first, InputIterator last) – вставка нескольких элементов в заданную позицию;

1.15. void push_back(const value_type& value) – добавление элемента с заданным значением в конец;

2. Были дописаны следующие методы для класса list:

2.1. void push_back(const value_type& value) – вставка элемента в конец списка;

2.2. void push_front(const value_type& value) – вставка элемента в начало списка;

2.3. reference front() – получение значения первого элемента;

2.4. const_reference front() const – получение постоянного значения первого элемента;

2.5. reference back() – получение значения последнего элемента;

2.6. const_reference back() const – получение постоянного значения последнего элемента;

2.7. void pop_front() – извлечение первого элемента списка;

2.8. void pop_back() – извлечение последнего элемента списка;

2.9. void clear() – отчистка списка;

2.10. bool empty() const – проверка на пустой список;

2.11. size_t size() const – получение размера списка;

2.12. ~list() – деструктор;

2.13. list(const list& other) – конструктор копирования;

2.14. list(list&& other) – конструктор перемещения;

2.15. list& operator= (const list& other) – оператор присваивания;

2.16. iterator insert(iterator pos, const Type& value) - вставка элемента в заданную позицию;

2.17. iterator erase(iterator pos) – удаление элемента из заданной позиции;

3. Были дописаны следующие методы для класса list_iterator:

3.1. list_iterator& operator = (const list_iterator& other) – оператор присваивания;

3.2. bool operator == (const list_iterator& other) const – проверка на равенство двух итераторов;

3.3. bool operator != (const list_iterator& other) const – проверка на неравенство двух итераторов;

3.4. reference operator * () – разыменование;

3.5. pointer operator -> () – получение адреса элемента;

3.6. list_iterator& operator ++ () – постфиксный переход к следующему итератору;

3.7. list_iterator operator ++ (int) – преффиксный переход к следующему итератору.

Заключение

В ходе выполнения данной лабораторной работы была изучена тема контейнеры. Были реализованы методы для контейнеров вектор и список.