Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 – «Программная инженерия»

**Лабораторная работа №11.**

**«Последовательные контейнеры библиотеки STL»**

Выполнил студент гр. РИС-24-2б

Молочко Артём Анатольевич

Проверил:

Доц. Каф. ИТАС

Ольга Андреевна Полякова

(оценка) (подпись)

(дата)

г. Пермь, 2025

**Постановка задачи**

**Используемые контейнеры:**

1. **Последовательные контейнеры:**
   * std::list (двусвязный список).
2. **Адаптеры контейнеров:**
   * std::priority\_queue (очередь с приоритетами).

**Типы данных:**

1. Стандартные:

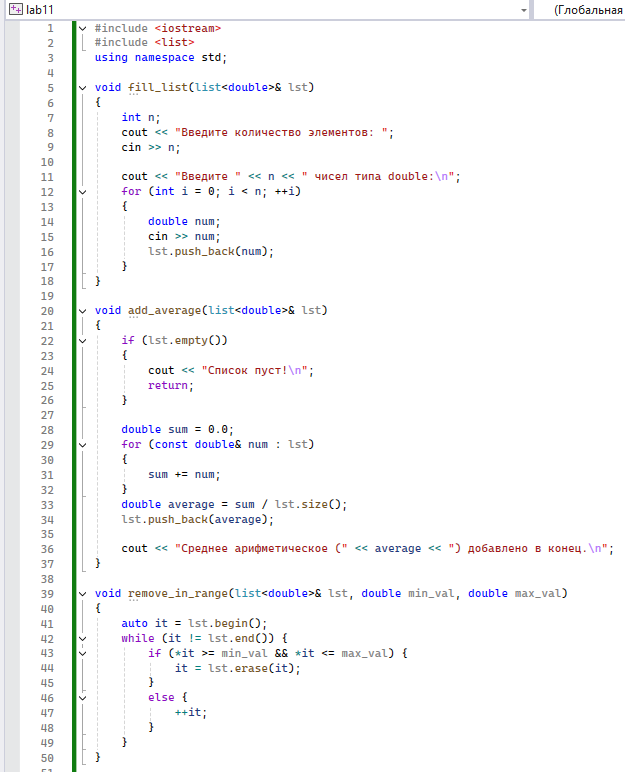
double.

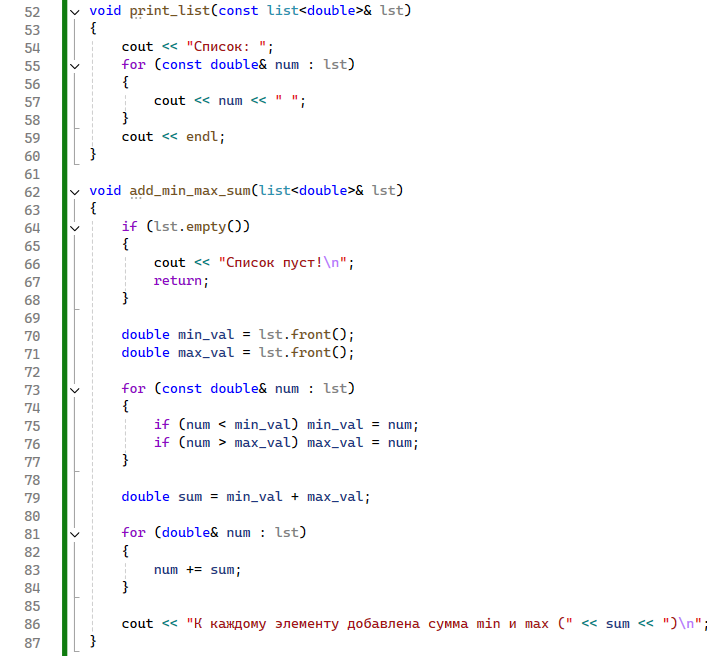
1. Пользовательские:
   * Pair (из лабораторной работы №3).

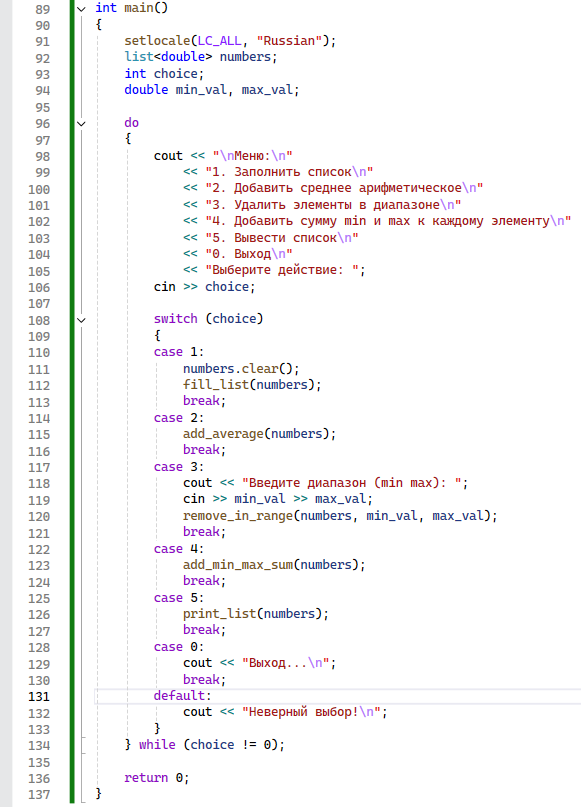
**Задачи:**

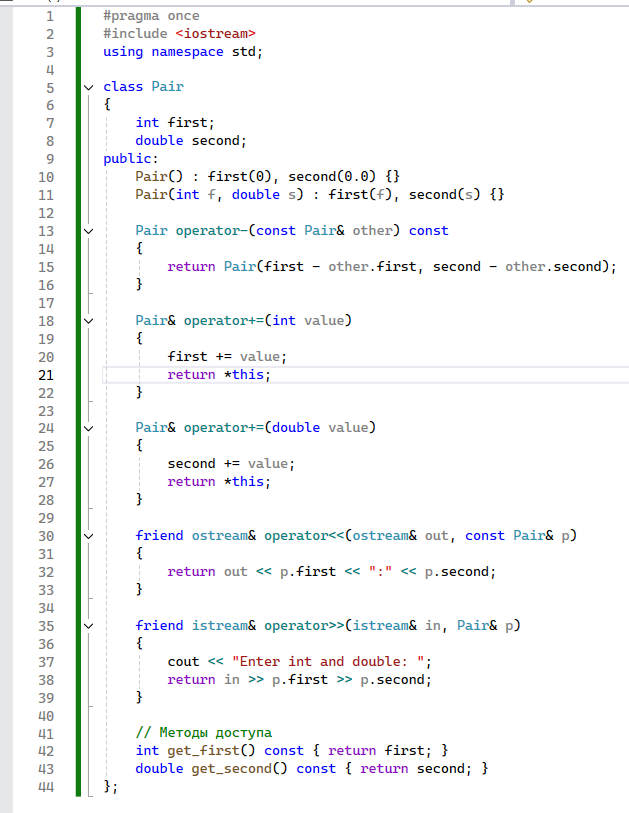
1. **Задача 1:**
   * Создать список (std::list) элементов типа double.
   * Заполнить, добавить и удалить элементы в соответствии с вариантом.
   * Реализовать операции в виде глобальных функций.
2. **Задача 2:**
   * Создать список (std::list) элементов пользовательского типа Pair.
   * Перегрузить необходимые операции для типа Pair.
   * Заполнить, добавить и удалить элементы.
   * Реализовать операции в виде глобальных функций.
3. **Задача 3:**
   * Создать параметризированный класс "Список" (на основе std::list).
   * Заполнить его элементами (тип зависит от варианта).
   * Найти среднее арифметическое и добавить его в конец контейнера.
   * Реализовать операции в виде методов класса.
4. **Задача 4:**
   * Создать очередь с приоритетами (std::priority\_queue) для пользовательского типа Pair.
   * Перегрузить операции сравнения для Pair.
   * Найти элементы с ключами из заданного диапазона и удалить их.
   * Реализовать операции в виде глобальных функций.
5. **Задача 5:**
   * Создать параметризированный класс "Список" (на основе std::list).
   * Использовать адаптер std::priority\_queue для работы с элементами.
   * К каждому элементу добавить сумму минимального и максимального элементов.
   * Реализовать операции в виде методов класса

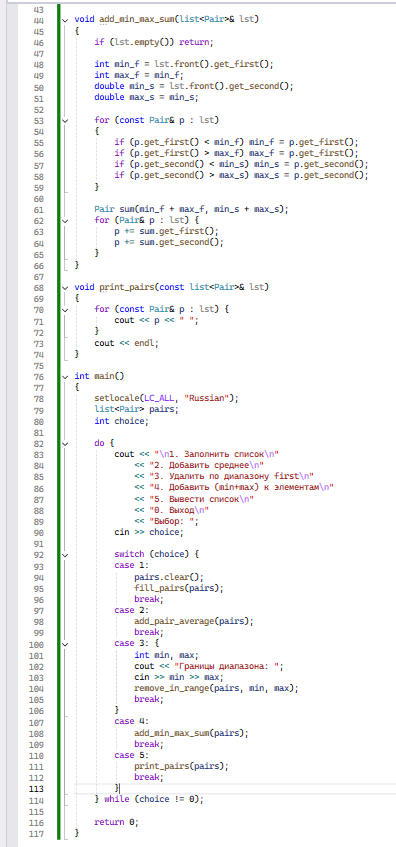
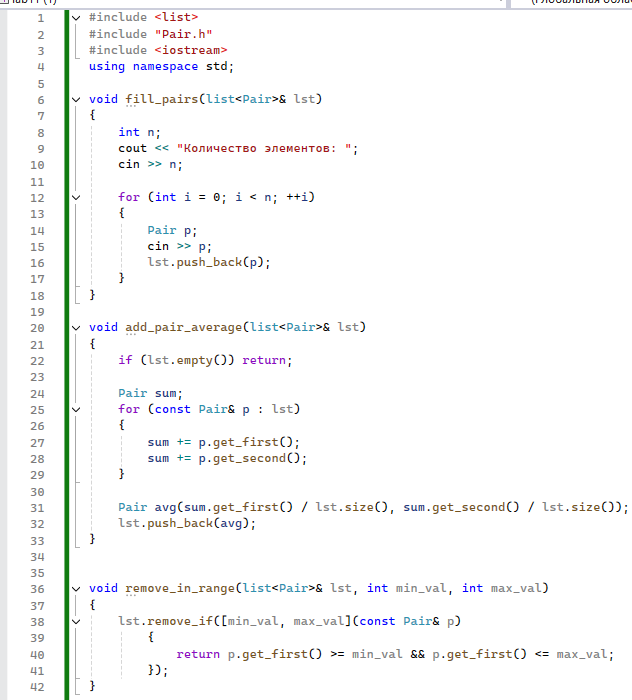
**Код программы**

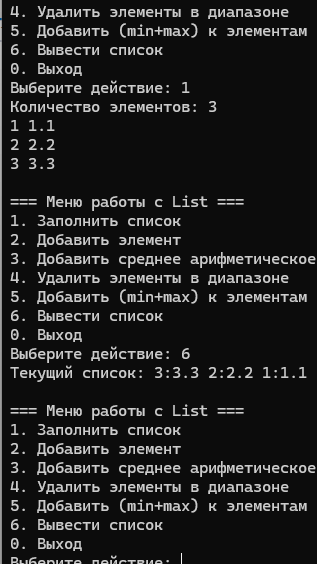
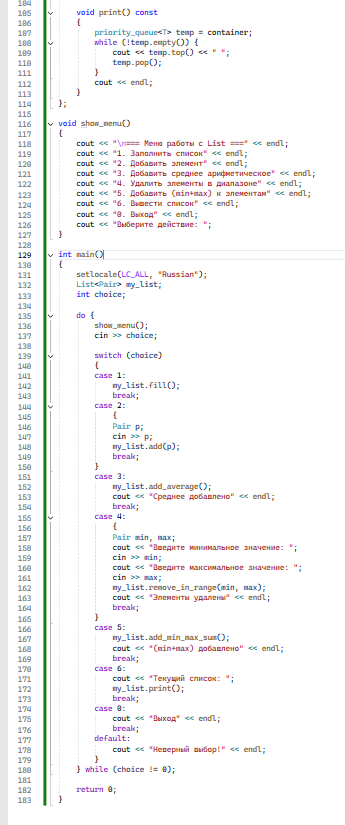
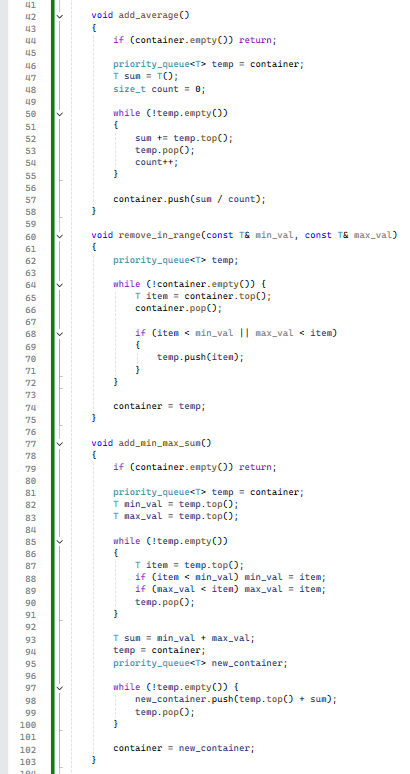
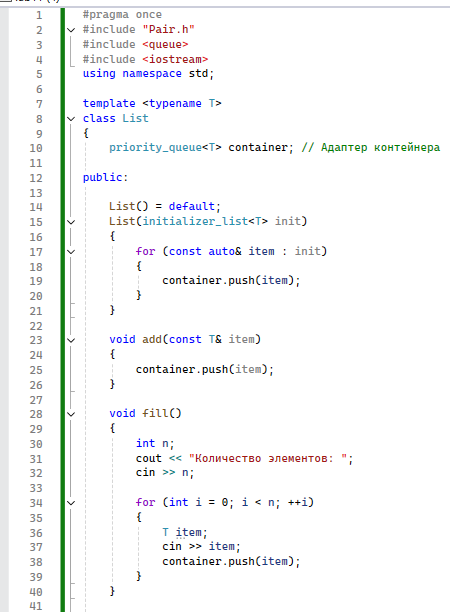
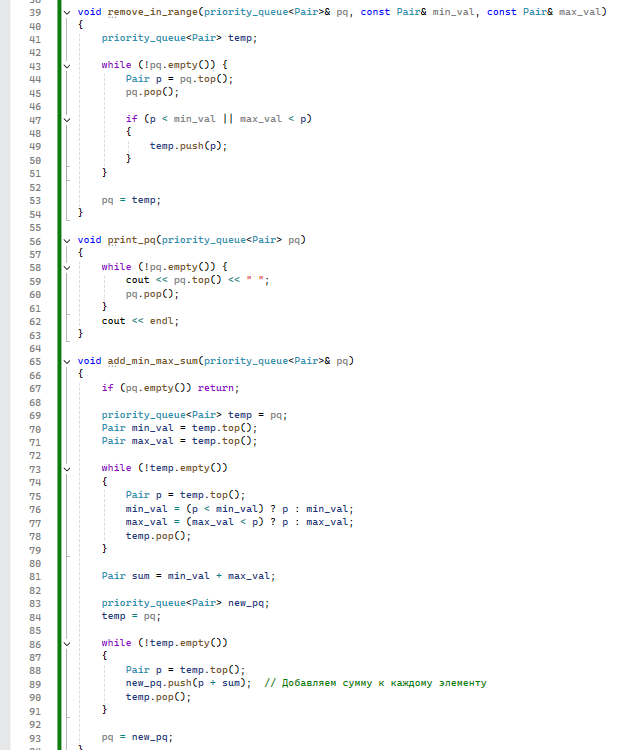
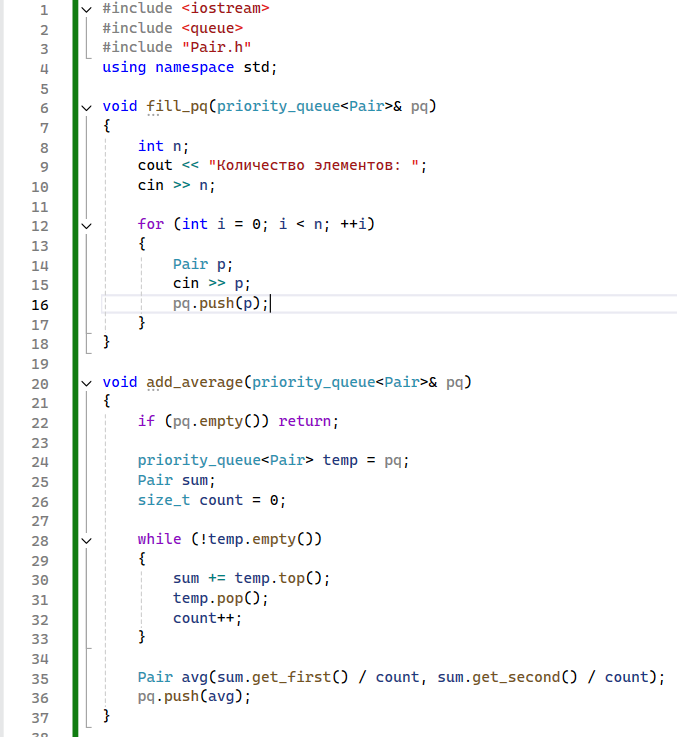
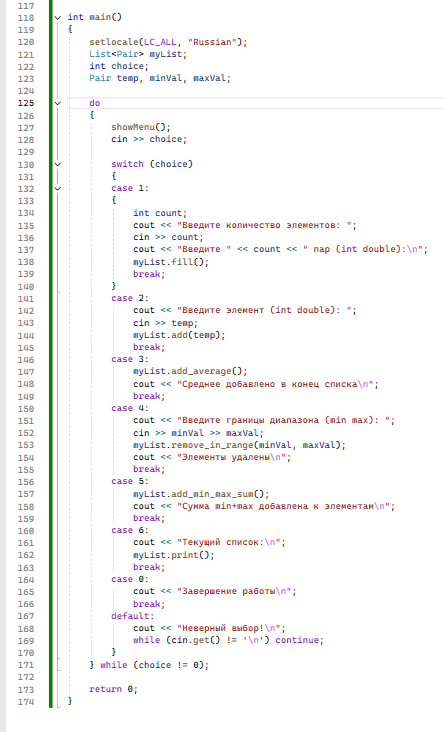
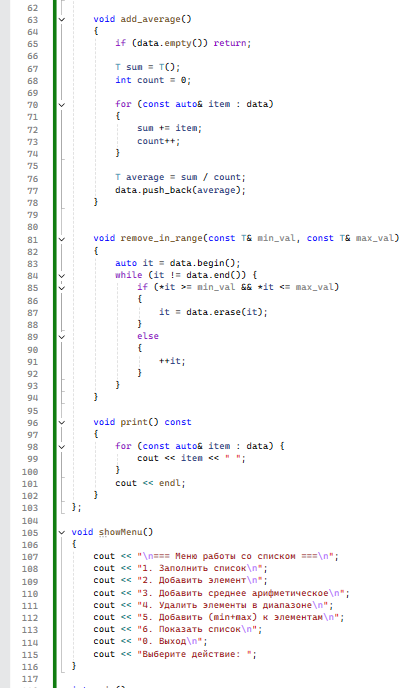
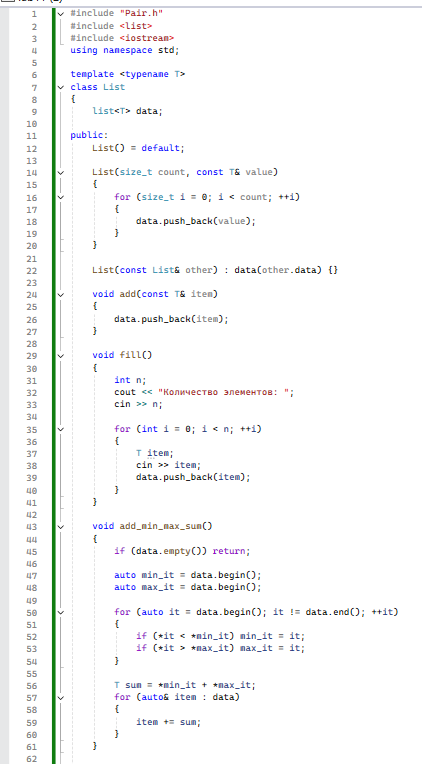


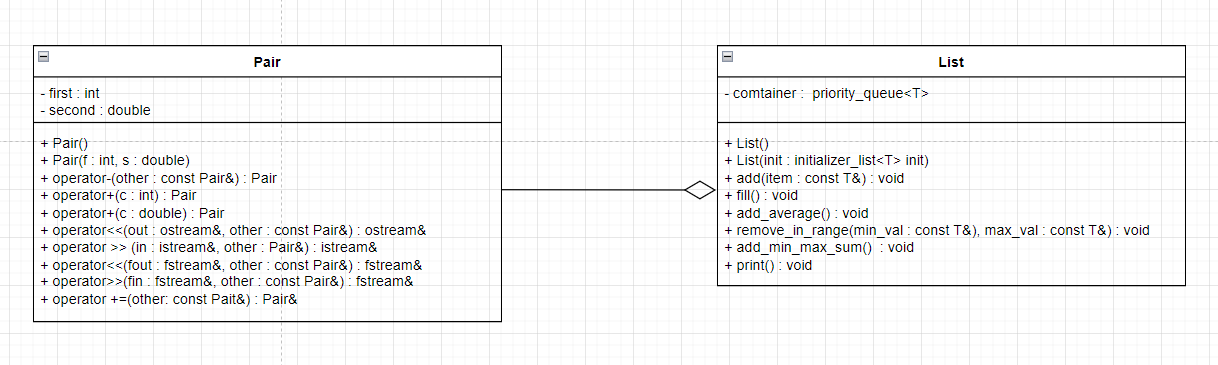










**Контрольные вопросы**

1. Из каких частей состоит библиотека STL?  
   Ответ: STL состоит из контейнеров, алгоритмов, итераторов, функторов (функциональных объектов) и адаптеров.
2. Какие типы контейнеров существуют в STL?  
   Ответ: В STL есть последовательные (vector, list, deque, array), ассоциативные (set, map, multiset, multimap), а также адаптеры (stack, queue, priority\_queue).
3. Что нужно сделать для использования контейнера STL в своей программе?  
   Ответ: Необходимо подключить соответствующий заголовочный файл (например, <vector>, <list>) и использовать пространство имен std.
4. Что представляет собой итератор?  
   Ответ: Итератор — это объект, который предоставляет доступ к элементам контейнера и позволяет перебирать их.
5. Какие операции можно выполнять над итераторами?  
   Ответ: Инкремент (++), декремент (--), разыменование (\*), сравнение (==, !=, <, >), арифметические операции (для случайного доступа, например, +, -).
6. Каким образом можно организовать цикл для перебора контейнера с использованием итератора?

for (auto it = container.begin(); it != container.end(); ++it) {

// работа с \*it

}

1. Какие типы итераторов существуют?  
   Ответ: Входные (input), выходные (output), однонаправленные (forward), двунаправленные (bidirectional), итераторы произвольного доступа (random access)
2. Перечислить операции и методы общие для всех контейнеров.  
   Ответ: begin(), end(), size(), empty(), clear(), swap().
3. Какие операции являются эффективными для контейнера vector? Почему?  
   Ответ: Доступ по индексу ([], at()), добавление в конец (push\_back()), удаление с конца (pop\_back()), так как vector использует непрерывный блок памяти.
4. Какие операции являются эффективными для контейнера list? Почему?  
   Ответ: Вставка и удаление в любом месте (insert(), erase()), так как list реализован как двусвязный список.
5. Какие операции являются эффективными для контейнера deque? Почему?  
   Ответ: Добавление и удаление с обоих концов (push\_front(), push\_back(), pop\_front(), pop\_back()), так как deque — это двусторонняя очередь.
6. Перечислить методы, которые поддерживает последовательный контейнер vector.  
   Ответ: push\_back(), pop\_back(), insert(), erase(), resize(), reserve(), operator[], at().
7. Перечислить методы, которые поддерживает последовательный контейнер list.  
   Ответ: push\_back(), push\_front(), pop\_back(), pop\_front(), insert(), erase(), remove(), sort(), merge(), splice().
8. Перечислить методы, которые поддерживает последовательный контейнер deque.  
   Ответ: push\_back(), push\_front(), pop\_back(), pop\_front(), insert(), erase(), operator[], at().
9. Задан контейнер vector. Как удалить из него элементы со 2 по 5?

vec.erase(vec.begin() + 1, vec.begin() + 5);

1. Задан контейнер vector. Как удалить из него последний элемент?  
   Ответ: vec.pop\_back();
2. Задан контейнер list. Как удалить из него элементы со 2 по 5?  
   Ответ:

auto it1 = std::next(lst.begin(), 1);

auto it2 = std::next(lst.begin(), 5);

lst.erase(it1, it2);

1. Задан контейнер list. Как удалить из него последний элемент?

lst.pop\_back();

1. Задан контейнер deque. Как удалить из него элементы со 2 по 5?  
   Ответ: deq.erase(deq.begin() + 1, deq.begin() + 5);
2. Задан контейнер deque. Как удалить из него последний элемент? deq.pop\_back();
3. Написать функцию для печати последовательного контейнера с использованием итератора.  
   Ответ:

template<typename Container>

void print(const Container& c) {

for (auto it = c.begin(); it != c.end(); ++it) {

cout << \*it << " ";

}

scout << endl;

}

1. Что представляют собой адаптеры контейнеров?  
   Ответ: Адаптеры (stack, queue, priority\_queue) — это обертки над другими контейнерами, предоставляющие специфичный интерфейс.
2. Чем отличаются друг от друга объявления stack<int> s и stack<int, list<int>> s?  
   Ответ: Первое использует deque по умолчанию, второе явно задает list как базовый контейнер.
3. Перечислить методы, которые поддерживает контейнер stack.  
   Ответ: push(), pop(), top(), empty(), size().
4. Перечислить методы, которые поддерживает контейнер queue.  
   Ответ: push(), pop(), front(), back(), empty(), size().
5. Чем отличаются друг от друга контейнеры queue и priority\_queue?  
   Ответ: queue — FIFO, priority\_queue — элементы извлекаются по приоритету (по умолчанию максимум первый).
6. Задан контейнер stack. Как удалить из него элемент с заданным номером?  
   Ответ: Нельзя напрямую, нужно временно извлекать элементы в другой стек.
7. Задан контейнер queue. Как удалить из него элемент с заданным номером?  
   Ответ: Нельзя напрямую, нужно временно извлекать элементы в другую очередь.
8. Написать функцию для печати контейнера stack с использованием итератора.  
   Ответ: stack не поддерживает итераторы
9. Написать функцию для печати контейнера queue с использованием итератора.  
   Ответ: queue не поддерживает итераторы