Министерство высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

по лабораторной работе №11

Тема: «Последовательные контейнеры

Библиотеки STL.»

Выполнил

Студент группы РИС-22-2б

Хохряков А.С.

Проверил доц. Кафедры ИТАС

Полякова О. А.

Пермь 2023

# Постановка задачи

**Задача 1.**

1. Создать последовательный контейнер.

2. Заполнить его элементами стандартного типа (тип указан в варианте).

3. Добавить элементы в соответствии с заданием

4. Удалить элементы в соответствии с заданием.

5. Выполнить задание варианта для полученного контейнера.

6. Выполнение всех заданий оформить в виде глобальных функций.

**Задача 2.**

1. Создать последовательный контейнер.

2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для

пользовательского типа перегрузить необходимые операции.

3. Добавить элементы в соответствии с заданием

4. Удалить элементы в соответствии с заданием.

5. Выполнить задание варианта для полученного контейнера.

6. Выполнение всех заданий оформить в виде глобальных функций.

**Задача 3**

1. Создать параметризированный класс, используя в качестве контейнера

последовательный контейнер.

2. Заполнить его элементами.

3. Добавить элементы в соответствии с заданием

4. Удалить элементы в соответствии с заданием.

5. Выполнить задание варианта для полученного контейнера.

6. Выполнение всех заданий оформить в виде методов параметризированного

класса.

**Задача 4**

1. Создать адаптер контейнера.

2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для

пользовательского типа перегрузить необходимые операции.

3. Добавить элементы в соответствии с заданием

4. Удалить элементы в соответствии с заданием.

5. Выполнить задание варианта для полученного контейнера.

6. Выполнение всех заданий оформить в виде глобальных функций.

**Задача 5**

1. Создать параметризированный класс, используя в качестве контейнера адаптер

контейнера.

2. Заполнить его элементами.

3. Добавить элементы в соответствии с заданием

4. Удалить элементы в соответствии с заданием.

5. Выполнить задание варианта для полученного контейнера.

6. Выполнение всех заданий оформить в виде методов параметризированного

класса.

Вариант 3:  
Задача 1

1. Контейнер - двунаправленная очередь

2. Тип элементов - double

Задача 2

Тип элементов Time (см. лабораторную работу №3).

Задача 3

Параметризированный класс - Вектор (см. лабораторную работу №7)

Задача 4

Адаптер контейнера - очередь с приоритетами.

Задача 5

Параметризированный класс - Вектор

Адаптер контейнера - очередь с приоритетами.

**Контрольные вопросы**

1. Библиотека STL (Standard Template Library) состоит из нескольких частей: контейнеров, алгоритмов, итераторов, функциональных объектов и адаптеров.

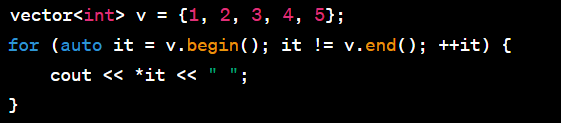
2. В STL существуют следующие типы контейнеров: последовательные контейнеры (vector, list, deque), ассоциативные контейнеры (set, map), ассоциативные контейнеры с хеш-таблицами (unordered\_set, unordered\_map) и контейнеры адаптеры (stack, queue, priority\_queue).

3. Для использования контейнера STL в своей программе необходимо подключить заголовочный файл, соответствующий данному контейнеру. Например, для использования контейнера vector нужно подключить заголовочный файл <vector>.

4. Итератор - это объект, который позволяет обходить элементы контейнера. Итераторы являются абстракцией, скрывающей способ, которым контейнер хранит свои элементы.

5. С помощью итераторов можно выполнять следующие операции: перемещаться по контейнеру вперед и назад, получать доступ к элементам контейнера, изменять значения элементов.

6. Для перебора контейнера с использованием итератора можно использовать цикл for или while. Например, для перебора контейнера vector с использованием итератора можно написать следующий код:



7. Существуют следующие типы итераторов: итераторы произвольного доступа (random access iterators), итераторы двунаправленного доступа (bidirectional iterators), итераторы прямого доступа (forward iterators) и итераторы ввода/вывода (input/output iterators).

8. Общие для всех контейнеров операции и методы: добавление и удаление элементов, доступ к элементам контейнера, проверка наличия элементов, получение размера контейнера.

9. Эффективными операциями для контейнера vector являются: доступ к элементу по индексу, добавление и удаление элементов в конец контейнера, получение размера контейнера.

10. Эффективными операциями для контейнера list являются: добавление и удаление элементов в любом месте контейнера, получение размера контейнера.

11. Операции, которые являются эффективными для контейнера deque, включают вставку и удаление элементов в начало и конец контейнера, а также доступ к первому и последнему элементам. Это связано с тем, что deque представляет собой двухстороннюю очередь, где элементы могут вставляться и удаляться как с начала, так и с конца контейнера.

12. Некоторые методы, которые поддерживает последовательный контейнер vector, включают в себя: push\_back(), pop\_back(), size(), empty(), operator[], front(), back() и т.д.

13. Некоторые методы, которые поддерживает последовательный контейнер list, включают в себя: push\_back(), push\_front(), pop\_back(), pop\_front(), size(), empty(), insert(), erase() и т.д.

14. Некоторые методы, которые поддерживает последовательный контейнер deque, включают в себя: push\_back(), push\_front(), pop\_back(), pop\_front(), size(), empty(), operator[], at(), front(), back() и т.д.

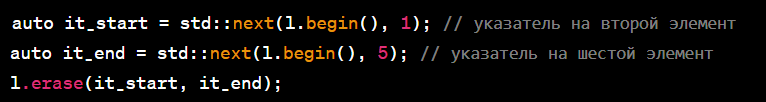
15. Чтобы удалить элементы с 2 по 5 из контейнера vector, можно использовать метод erase() с указателями на начало и конец диапазона элементов, которые нужно удалить. Например, вектор v:



16. Чтобы удалить последний элемент из контейнера vector, можно использовать метод pop\_back(). Например, вектор v:



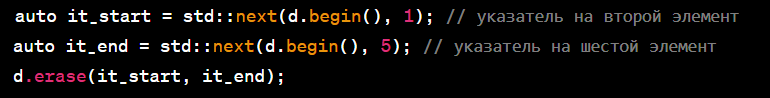
17. Чтобы удалить элементы с 2 по 5 из контейнера list, можно использовать методы erase() и begin() / end(). Например, список l:



18. Чтобы удалить последний элемент из контейнера list, можно использовать метод pop\_back(). Например, список l:



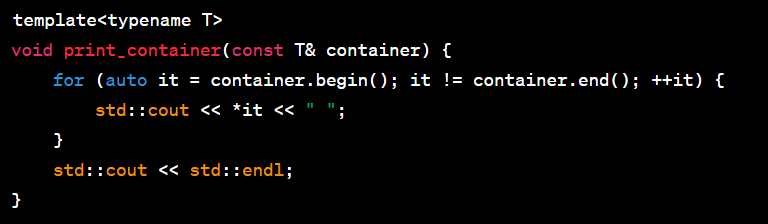
19. Чтобы удалить элементы с 2 по 5 из контейнера deque, можно использовать методы erase() и begin() / end(). Например, deque d:



20. Чтобы удалить последний элемент из контейнера deque, можно использовать метод pop\_back(). Например, deque d:



21. Пример функции для печати последовательного контейнера с использованием итератора:



22. Адаптеры контейнеров представляют собой специальные классы-обертки, которые позволяют изменять интерфейс контейнера или его поведение. Например, адаптер stack предоставляет интерфейс стека, а адаптер queue - интерфейс очереди.

23. Оба объявления создают объект типа stack<int>. Однако, первый вариант использует дефолтный адаптер контейнера deque для реализации стека, а второй - адаптер контейнера list.

24. Контейнер stack поддерживает следующие методы:

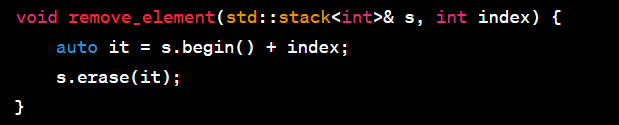
* push: добавляет элемент на вершину стека
* pop: удаляет элемент с вершины стека
* top: возвращает ссылку на элемент на вершине стека
* size: возвращает количество элементов в стеке
* empty: возвращает true, если стек пуст, и false в противном случае

25. Контейнер queue поддерживает следующие методы:

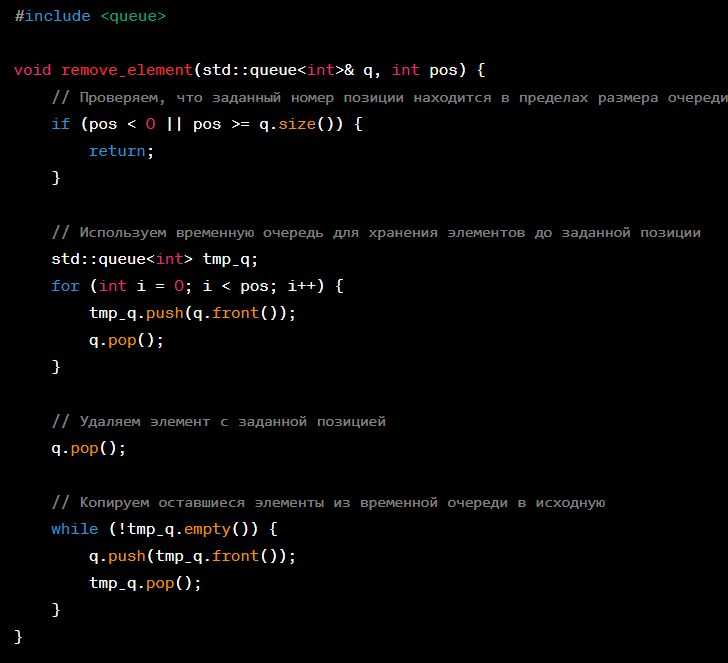
* push: добавляет элемент в конец очереди
* pop: удаляет элемент из начала очереди
* front: возвращает ссылку на элемент в начале очереди
* back: возвращает ссылку на элемент в конце очереди
* size: возвращает количество элементов в очереди
* empty: возвращает true, если очередь пуста, и false в противном случае

26. Основное отличие между контейнерами queue и priority\_queue заключается в том, что queue реализует обычную очередь, в которой элементы добавляются в конец и удаляются из начала, а priority\_queue реализует очередь с приоритетом, в которой элементы добавляются в произвольном порядке и удаляются в порядке убывания приоритета.

27. Чтобы удалить элемент с заданным номером из контейнера stack, необходимо сначала переместиться на этот элемент с помощью итератора, а затем вызвать метод erase у итератора:

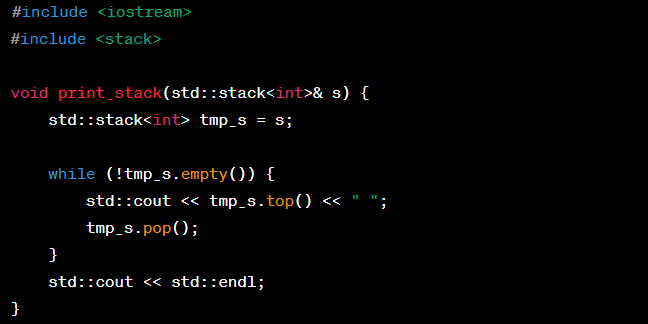


28. Для удаления элемента с заданным номером из контейнера queue в языке C++ можно использовать следующий код:



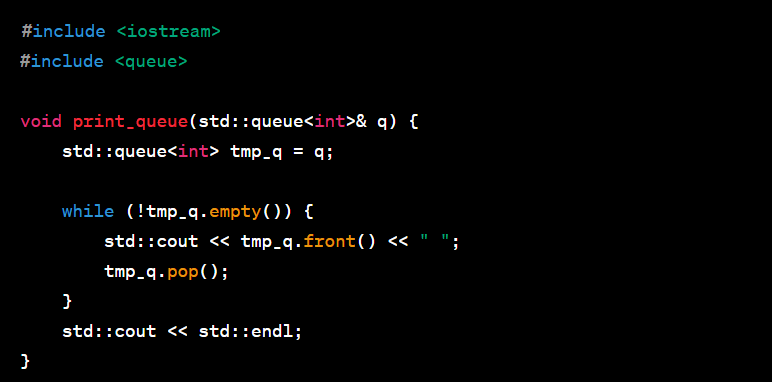
Эта функция принимает ссылку на контейнер queue и номер позиции элемента, который нужно удалить. Сначала мы проверяем, что заданный номер находится в пределах размера очереди. Затем мы используем временную очередь для хранения элементов до заданной позиции. После удаления элемента с заданной позицией мы копируем оставшиеся элементы из временной очереди в исходную.

29. Для печати контейнера stack с использованием итератора в языке C++ можно использовать следующий код:



Эта функция принимает ссылку на контейнер stack и использует временный стек для хранения копии исходного стека. Затем мы итерируемся по временному стеку, печатая элементы в обратном порядке, и удаляем их.

30. Для печати контейнера queue с использованием итератора в языке C++ можно использовать следующий код:



Эта функция принимает ссылку на контейнер queue и использует временную очередь для хранения копии исходной очереди. Затем мы итерируемся по временной очереди, печатая элементы в порядке их поступления, и удаляем их.