Министерство образования и науки Российской Федерации

Федеральное государственное образовательное автономное учреждение высшего образования

"Пермский национальный исследовательский политехнический университет"

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №11

Дисциплина: Основы алгоритмизации и программирования

Тема: Последовательные контейнеры библиотеки STL.

Вариант 15

|  |  |
| --- | --- |
| Выполнил работу | |
| Студент группы РИС-22-1б | |
| Мерзляков В.Н. | |
|  | |
| Проверил работу | |
| Доцент 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.

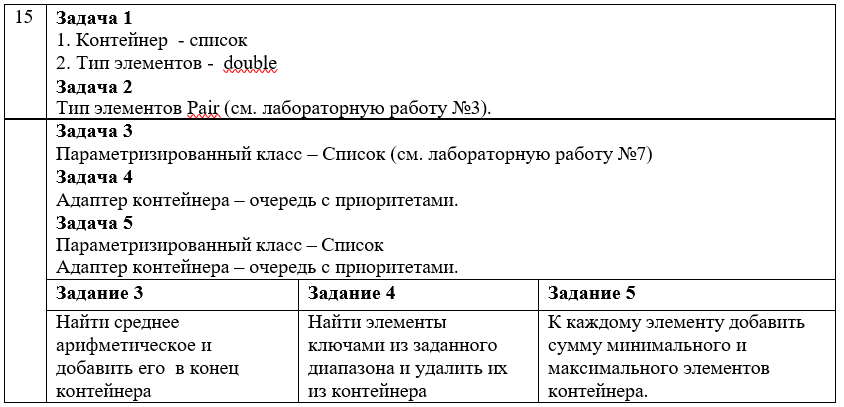
Задача 4

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.

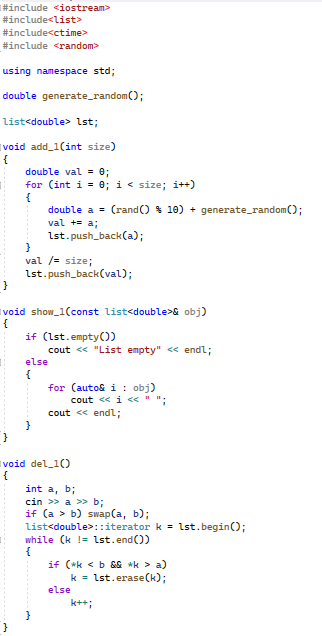
Задача 5

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.

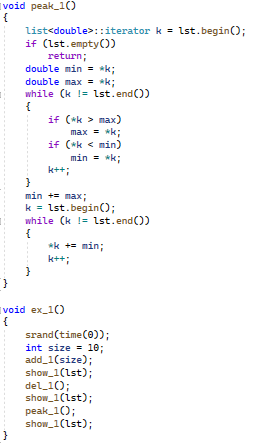
Вариант 15:



# Задание 1

****

*Приложение 1 – Задание 1*

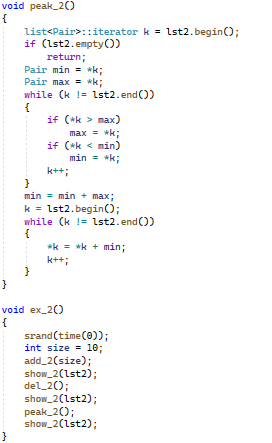
****

*Приложение 2 – Задание 1*

# Задание 2

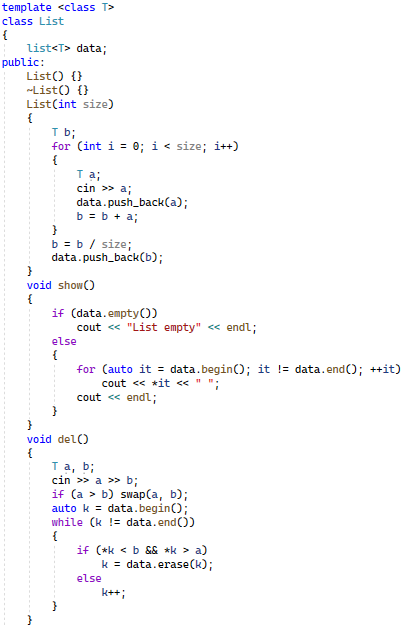
# 

*Приложение 3 – Задание 2*

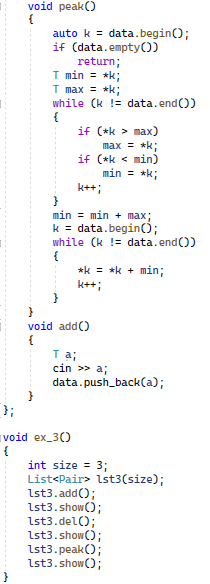


*Приложение 4 – Задание 2*

# Задание 3



*Приложение 5 – Задание 3*

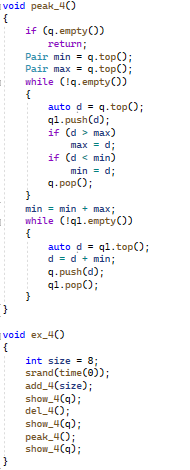
**

*Приложение 6 – Задание 3*

# Задание 4

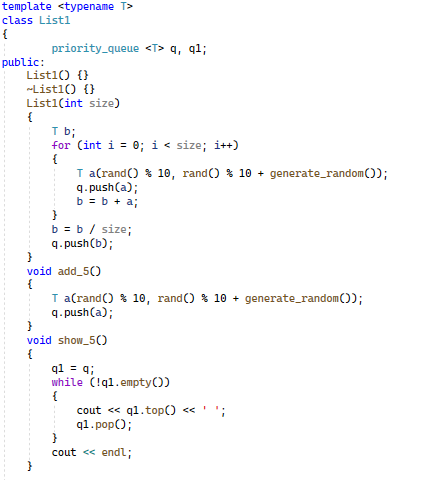
# 

*Приложение 7 – Задание 4*

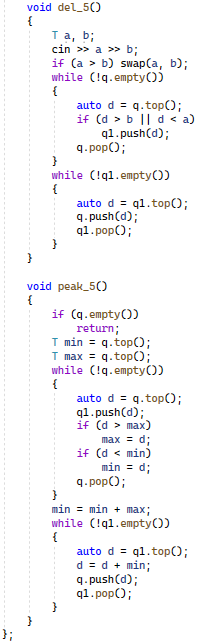


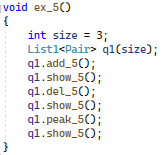
*Приложение 8 – Задание 4*

# Задание 5

****

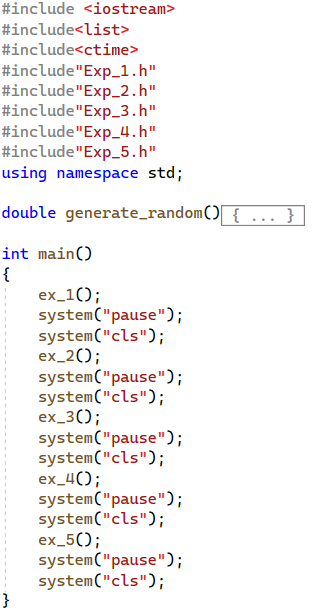
*Приложение 9 – Задание 5*

****

**

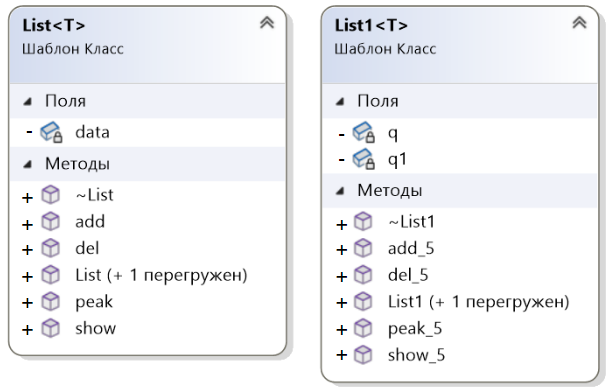
*Приложение 10 – Задание 5*

**Функция main()**

****

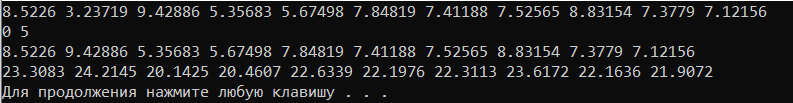
*Приложение 11 – Главная функция*

**UML-Диаграмма**

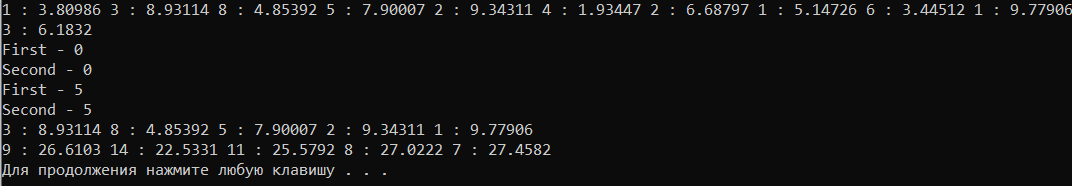
****

*Приложение 12 – UML-Диаграмма*

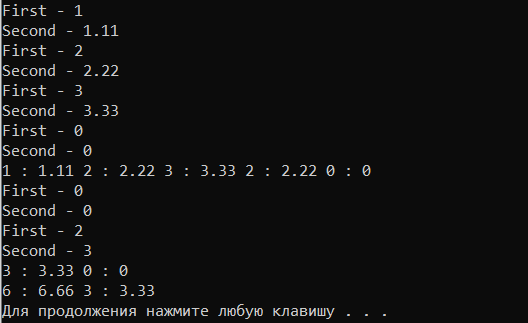
**Работа программы**



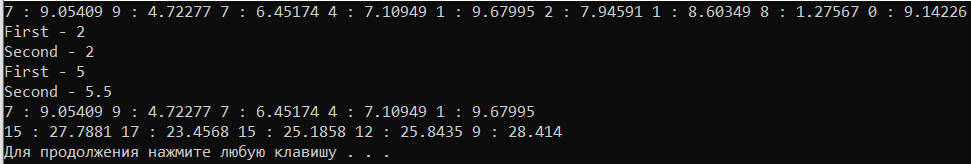
*Приложение 13 – Работа программы - задание 1*

**

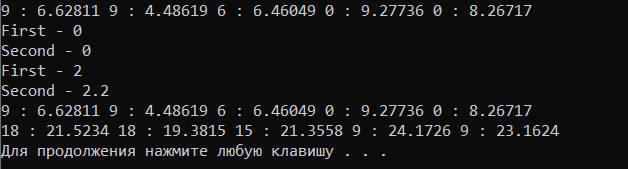
*Приложение 14 – Работа программы – задание 2*

**

*Приложение 15 – Работа программы – задание 3*

**

*Приложение 16 – Работа программы – задание 4*

**

*Приложение 17 – Работа программы – задание 5*

**Ответы на контрольные вопросы**

1.Из каких частей состоит библиотека STL?

Библиотека STL (Standard Template Library) состоит из трех частей: контейнеры, алгоритмы и итераторы.

2.Какие типы контейнеров существуют в STL?

В STL существует 3 типа контейнеров: последовательные (vector, list, deque), ассоциативные (set, map) и адаптеры (stack, queue).

3.Что нужно сделать для использования контейнера STL в своей программе?

Для использования контейнера STL в своей программе нужно подключить заголовочный файл, соответствующий нужному контейнеру (например, для vector - #include <vector>), а также объявить объект нужного типа.

4.Что представляет собой итератор?

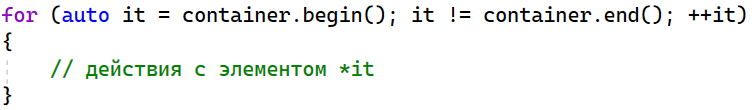
Итератор - это объект, который позволяет перебирать элементы контейнера. Он работает как указатель на элемент и может быть инкрементирован или декрементирован для перемещения по контейнеру.

5.Какие операции можно выполнять над итераторами?

С помощью итераторов можно выполнять операции: разыменование (\*), инкремент (++), декремент (--), сравнение (==, !=), доступ к элементу (->).

6.Каким образом можно организовать цикл для перебора контейнера с использованием итератора?

Цикл для перебора контейнера с использованием итератора можно организовать следующим образом:



7.Какие типы итераторов существуют?

Существуют 5 типов итераторов: input iterator, output iterator, forward iterator, bidirectional iterator и random access iterator.

8.Перечислить операции и методы общие для всех контейнеров.

Общие операции и методы для всех контейнеров: size(), empty(), clear(), begin(), end(), rbegin(), rend(), insert(), erase().

9.Какие операции являются эффективными для контейнера vector? Почему?

Эффективными операциями для контейнера vector являются доступ к элементу по индексу (оператор []) и добавление элемента в конец (метод push\_back()), так как вектор хранит элементы в непрерывной области памяти.

10.Какие операции являются эффективными для контейнера list? Почему?

Эффективными операциями для контейнера list являются добавление/удаление элемента в произвольном месте (методы insert(), erase()), так как список хранит элементы в виде связного списка.

11.Какие операции являются эффективными для контейнера deque? Почему?

Эффективными операциями для контейнера deque являются добавление/удаление элемента в начале/конце (методы push\_front(), pop\_front(), push\_back(), pop\_back()), так как дек хранит элементы в виде двух буферов, каждый из которых может быть расширен по мере необходимости.

12.Перечислить методы, которые поддерживает последовательный контейнер vector.

Методы, которые поддерживает последовательный контейнер vector: push\_back(), pop\_back(), front(), back(), size(), max\_size(), resize(), reserve(), clear(), empty().

13.Перечислить методы, которые поддерживает последовательный контейнер list.

Методы, которые поддерживает последовательный контейнер list: push\_front(), push\_back(), pop\_front(), pop\_back(), front(), back(), size(), max\_size(), resize(), clear(), empty().

14.Перечислить методы, которые поддерживает последовательный контейнер deque.

Методы, которые поддерживает последовательный контейнер deque: push\_front(), push\_back(), pop\_front(), pop\_back(), front(), back(), size(), max\_size(), resize(), clear(), empty().

15.Задан контейнер vector. Как удалить из него элементы со 2 по 5?

Чтобы удалить элементы со 2 по 5 из контейнера vector, можно использовать метод erase() с передачей ему итераторов на начало и конец диапазона, который нужно удалить:



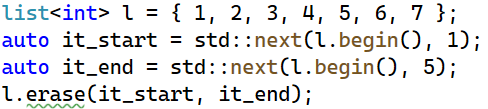
16.Задан контейнер vector. Как удалить из него последний элемент?

Чтобы удалить последний элемент из контейнера vector, можно использовать метод pop\_back():



17.Задан контейнер list. Как удалить из него элементы со 2 по 5?

Чтобы удалить элементы со 2 по 5 из контейнера list, можно использовать методы begin() и erase() с передачей им итераторов на начало и конец диапазона, который нужно удалить:



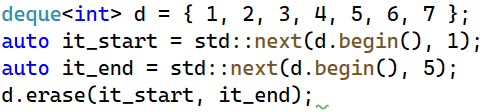
18.Задан контейнер list. Как удалить из него последний элемент?

Чтобы удалить последний элемент из контейнера list, можно использовать метод pop\_back():



19.Задан контейнер deque. Как удалить из него элементы со 2 по 5?

Чтобы удалить элементы со 2 по 5 из контейнера deque, можно использовать методы begin() и erase() с передачей им итераторов на начало и конец диапазона, который нужно удалить:



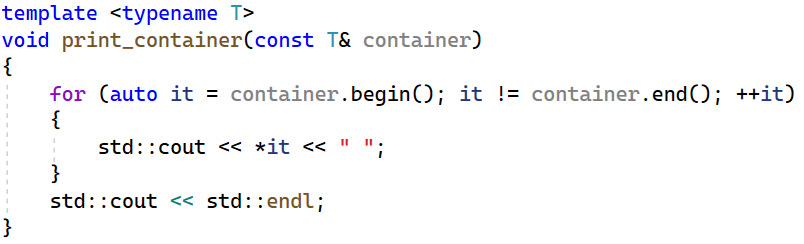
20.Задан контейнер deque. Как удалить из него последний элемент?

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



21.Написать функцию для печати последовательного контейнера с использованием итератора.

Функция для печати последовательного контейнера с использованием итератора может выглядеть так:



Эта функция принимает любой последовательный контейнер (vector, list, deque) и выводит его элементы через пробел.

22.Что представляют собой адаптеры контейнеров?

Адаптеры контейнеров - это классы-обертки над стандартными контейнерами, которые предоставляют новый интерфейс для работы с контейнером. Например, stack и queue являются адаптерами контейнеров.

23.Чем отличаются друг от друга объявления stack<int> s и stack<int, list<int> > s?

Объявление stack<int> s использует по умолчанию контейнер deque для хранения элементов стека. А объявление stack<int, list<int>> s указывает явно, что для хранения элементов стека будет использоваться контейнер list.

24.Перечислить методы, которые поддерживает контейнер stack.

Методы контейнера stack: push(), pop(), top(), empty(), size().

25.Перечислить методы, которые поддерживает контейнер queue.

Методы контейнера queue: push(), pop(), front(), back(), empty(), size().

26.Чем отличаются друг от друга контейнеры queue и priority\_queue?

Контейнер queue - это очередь, в которой элементы добавляются в конец и удаляются из начала. Контейнер priority\_queue - это очередь с приоритетами, в которой элементы добавляются в произвольное место и удаляются в порядке убывания приоритета.

27.Задан контейнер stack. Как удалить из него элемент с заданным номером?

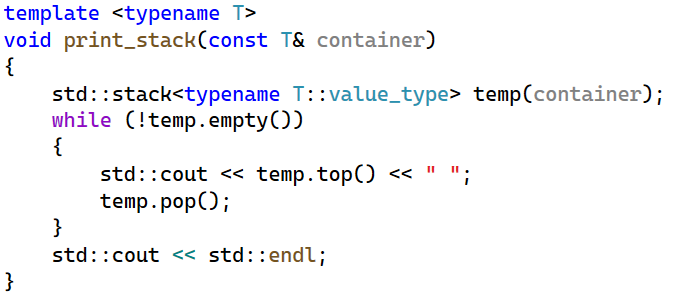
В контейнере stack нельзя удалить элемент с заданным номером, так как доступ к элементам осуществляется только через вершину стека.

28.Задан контейнер queue. Как удалить из него элемент с заданным номером?

В контейнере queue нельзя удалить элемент с заданным номером, так как доступ к элементам осуществляется только к первому элементу в очереди.

29.Написать функцию для печати контейнера stack с использованием итератора.

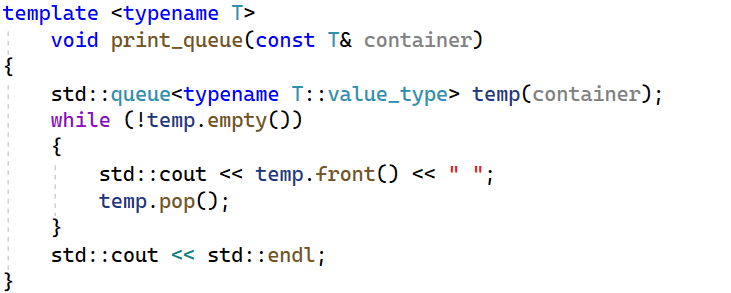
Функция для печати контейнера stack с использованием итератора может выглядеть так:



Эта функция создает временный стек, копирует в него элементы из переданного контейнера и выводит их на экран.

30.Написать функцию для печати контейнера queue с использованием итератора.

Функция для печати контейнера queue с использованием итератора может выглядеть так:



Эта функция создает временную очередь, копирует в нее элементы из переданного контейнера и выводит их на экран.