Introduction

This chapter deals with the different features that are present in all types of C++ containers.

WE'LL COVER THE FOLLOWING

Although the sequential and associative containers of the Standard Template library are two very different classes of containers, they have a lot in common. For example, the operations used to `create` or `delete` a container, to determine its `size`, to `access` its elements, to `assign` or `swap`, are all independent of the type of elements in a container. It is common for the containers to be defined with an arbitrary size. The reason is that each container has an allocator, hence the size of a container can be adjusted at runtime. The allocator works in the background most of the time. This can be seen for an `std::vector`. The call `std::vector<int>` results in a call `std::vector<int, std::allocator<int>>`. Because of the `std::allocator` we can adjust the size of all containers dynamically, except for `std::array`. However, they have even more in common. We can access the elements of a container quite easily with an iterator.

Having so much in common, the containers differ in their details. The chapters Sequential Containers and Associative Containers in General provide these details.

C++ covers the sequential containers std::array, std::vector, std::deque, std::list, and std::forward_list, in detail.

The same holds true for associative containers, which can be classified in the ordered and unordered associative container.

Further information

- Sequential Containers

- Associative Containers in General
- std::array
- std::vector
- std::deque
- std::list
- std::forward_list

In the next lesson, we'll discuss how to create and delete containers with particular parameters.