Part III. Containers

Containers are one of the most useful data structures in C++. The standard library provides many containers, and the Boost libraries provide even more.

- Boost.MultiIndex goes one step further: the containers from this library can support multiple interfaces from other containers at the same time. Containers from Boost.MultiIndex are like merged containers and provide the advantages of all of the containers they have been merged with.
- Boost.Bimap is based on Boost.MultiIndex. It provides a container similar to `std::unordered_map`, except the elements can be looked up from both sides. Thus, depending on how the container is accessed, either side can be the key. When one side is the key, the other side is the value.
- Boost.Array and Boost.Unordered define the classes `boost::array`, `boost::unordered_set`, and `boost::unordered_map`, which were added to the standard library with C++11.
- Boost.CircularBuffer provides a container whose most important property is that it will overwrite the first element in the buffer when a value is added to a full circular buffer.
- Boost.Heap provides variants of priority queues – classes that resemble `std::priority_queue`.
- Boost.Intrusive lets you create containers that, unlike the containers from the standard library, neither copy nor move objects. However, to add an object to an intrusive list, the object’s type must meet certain requirements.
- Boost.MultiArray tries to simplify the use of multidimensional arrays. For example, it’s possible to treat part of a multidimensional array as a separate array.
- Boost.Container is a library that defines the same containers as the standard library. Using Boost.Container can make sense if, for example, you need to support a program on multiple platforms and you want to avoid problems caused by implementation-specific differences in the standard library.

Table of Contents

20. Boost.Container