Part IV. Data Structures

Data structures are similar to containers since they can store one or multiple elements. However, they differ from containers because they don't support operations containers usually support. For example, it isn't possible, with the data structures introduced in this part, to access all elements in a single iteration.

- Boost.Optional makes it easy to mark optional return values. Objects created with Boost.Optional are either empty or contain a single element. With Boost.Optional, you don't need to use special values like a null pointer or -1 to indicate that a function might not have a return value.
- Boost.Tuple provides `boost::tuple`, a class that has been part of the standard library since C++11.
- Boost.Any and Boost.Variant let you create variables that can store values of different types. Boost.Any supports any arbitrary type, and Boost.Variant lets you pass the types that need to be supported as template parameters.
- Boost.PropertyTree provides a tree-like data structure. This library is typically used to help manage configuration data. The data can also be written to and loaded from a file in formats such as JSON.
- Boost.DynamicBitset provides a class that resembles `std::bitset` but is configured at runtime.
- Boost.Tribool provides a data type similar to `bool` that supports three states.
- Boost.CompressedPair defines the class `boost::compressed_pair`, which can replace `std::pair`. This class supports the so-called empty base class optimization.

Table of Contents