Preface

What you will learn

This book is an introduction to the Boost C++ Libraries. The Boost C++ Libraries complement the standard library. Because the Boost C++ Libraries are based on the standard, they are implemented using state-of-the-art C++. They are platform independent and are supported on many operating systems, including Windows and Linux, by a large developer community.

The Boost C++ Libraries enable you to boost your productivity as a C++ developer. For example, you can benefit from smart pointers that help you to write more reliable code or use one of the many libraries to develop platform-independent network applications. Since the Boost libraries partly anticipate developments in the standard, you can benefit earlier from tools without having to wait for them to become available in the standard library.

What you should know

Since the Boost libraries are based on, and extend, the standard, you should know the standard well. You should understand and be able to use containers, iterators, and algorithms, and ideally you should have heard of concepts such as RAII, function objects, and predicates. The better you know the standard, the more you will benefit from the Boost libraries.

In general, you don’t need any knowledge of template meta programming to use the libraries introduced in this book. The main focus is on libraries that can be learned quickly and easily and that can be immediately of great benefit in your work as a C++ developer.

Many examples use features that were added to the standard with C++11. For example, the keyword `auto` is used to avoid specifying types explicitly. Constructors are called through uniform initialization: variables are initialized, if possible, with a pair of curly brackets instead of parentheses. Many examples use lambda functions to make code shorter and more compact. While you can understand many examples without detailed knowledge of C++11, this book is based on the current standard.

Typographical Conventions

The following text styles are used in this book:

`Monospace font`

> A monospace font is used for class names, function names, and keywords – basically for any C++ code. It is also used for code examples, command line options, and program output. For example: `int i = 0;`

`Monospace bold font`

> A monospace bold font is used for variable names, objects, and user input. For example: The variable **`i`** is initialized with 0.

Bold

Commands are marked in bold. For example: The Boost libraries are compiled with a program called **bjam**.

Italic

An italic font is used when a new concept is introduced and mentioned for the first time. For example: *RAII* is the abbreviation for Resource Acquisition Is Initialization – a concept smart pointers are based on.

Examples

This book contains more than 430 examples. Every example is complete and can be compiled and executed. You can download all examples from https://theboostcpplibraries.com/examples for a quick start.

All examples have been tested with the following compilers: Microsoft Visual Studio Professional 2013 Update 1 (64-bit Windows 7 Professional with Service Pack 1), GCC 4.8.3 (64-bit Cygwin 1.7.30), GCC 4.6.3 (32-bit Ubuntu 12.04.4), and Clang 3.3 (32-bit Ubuntu 12.04.4).

All of the examples in this book are based on the C++11 standard. During testing, all of the compilers were configured to enable support for C++11. Most examples will work on Windows, Linux, and OS X, but a few are platform dependent. The exceptions are noted in the example descriptions.