Preface

C++ Boost Libraries

Boost is a collection of C++ libraries, each one focusing on a particular issue of concern to software developers. There are approximately one hundred libraries that can be classified into one or more categories. The functionality corresponds to text and string processing, input/output, multi-threading and higher-order functions, to name a few. Most of the libraries are licensed under the Boost Software License and this allows Boost code to be used with both free and proprietary software projects. The libraries can be used by a wide range of C++ users and in many kinds of application domains. Furthermore, several Boost libraries have been incorporated into the new C++ standard. Boost makes extensive use of templates in order to ensure efficiency and flexibility.

The approach in this book is based on the assumption that readers wish to use the Boost libraries as soon as possible in their applications. In particular, we adopt the following strategies; first, we give numerous examples that contain all code to help you become acquainted with a given library as soon as possible. Second, we discuss more advanced features of the Boost libraries and we create code to show how to apply these features. Finally, we show how to use the Boost libraries as building blocks in applications that use design patterns. In fact, a number of these libraries are implementations of some well-known patterns. However, you can skip the discussion of design patterns in Boost without loss of continuity.

We do feel that it is important to actually use, modify and run the code examples that accompany the book if you wish to get a good understanding of how to use the libraries, in particular the mundane but time-consuming issues such as resolving compiler errors and getting used to the syntax of each library. Furthermore, the code should help those developers who wish to improve their knowledge of C++ templates and generic programming.

The Boost site is www.boost.org where the reader can find all relevant information, including software downloads, installation, Boost mailing-lists, documentation and other technical information.

Goals of this Book

This book describes approximately 25 C++ libraries in Boost. We focus mainly on categories for higher-order functions, data types and data structures, text and string processing and a number of miscellaneous libraries for smart pointers, serialization, multi-threading, flyweight objects and random number generation. We have deliberately included these libraries (and excluded others) because we wish to appeal to a wide range of C++ developers and we hope that the 18 chapters in this book will be useful to them in their daily work.

The main goals of this book are: first, to describe the Boost libraries in such a way that readers can understand and learn how to use them. In each chapter we introduce essential syntax in combination with simple examples and we then progress to more extended examples. The second goal is to introduce a number of modern software techniques and

methods that can be directly implemented in Boost, for example reference-counted memory pointers, multi-threading concepts and their implementation, multi-dimensional array structures and regular expressions. Finally, we discuss how several popular design patterns are directly supported in Boost.

Structure of the book
There are 18 chapters in the book that we group into several major categories:

- Higher-order functions
- Data types and data structures
- Text and string processing
- System libraries (for example, smart pointers)
- Miscellaneous libraries

Since there are almost 100 libraries in Boost we have made an attempt to discuss the 25 libraries that we think will be of benefit to developers most of the time. It is for this reason that we decided not to include the more mathematically-oriented libraries and libraries for meta-programming in this book. On the other hand, we include a chapter on multi-threading because of the increasing interest in parallel programming models for multi-core and multi-processor CPUs.

In order to retain the relationship with software design and application development we have included a chapter and three appendices on migrating legacy code to code that uses Boost libraries for serialisation, smart pointers in combination with object-oriented and generic design patterns.

For whom is this book?
This book is for C++ developers and programmers who have several years experience with small, medium and large applications. We assume that the reader knows how the object-oriented model is supported in C++ (encapsulation, inheritance, polymorphic functions) and knows STL containers, algorithms and iterators. Working experience of these techniques is assumed. Advanced generics – such as traits and policy based design – will be reviewed in appendices A and B for those readers for whom these topics are new. Appendix C contains exercises and projects to test your hands-on knowledge of the Boost libraries.

Using the Boost Libraries
How can we use these libraries? In general, it is better to use them rather than implementing the same functionality yourself in our opinion. For example, instead of writing code to do string manipulation we prefer to use the String Algo library. Likewise, we use Boost.MultiArray instead of creating our own n-dimensional data structures. Furthermore, the Pareto rule (80/20) rule will probably be applicable; a number of libraries will be important in a given software project and for each of these libraries you will see that some classes and interfaces are more often used that other ones. Finally, we can use the Boost libraries directly in new projects and a second scenario is to replace code by similar Boost code. For example, the authors use the Boost mathematical libraries rather than creating their own or using other proprietary libraries.

The chapters in this book should hopefully appeal to a wide audience. We have resisted discussing more advanced and domain-specific libraries, for example networking, graph and mathematical libraries as well as libraries for advanced data structures, interoperability and advanced parsers. We discuss these libraries in Volume II and their applications to

engineering, optimization, mixed language development (Python, C#, C++/CLI), computer graphics and computational finance.

Full source code for all the examples that we discuss in the book are downloadable from the Datasim site www.datasimfinancial.com if you have an official key (which you receive when you purchase this book).

We wish to thank Ilona Hooft Graafland of Datasim Education who produced this book and made it production-ready.