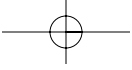


Ulla Kirch-Prinz, Peter Prinz

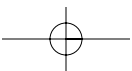
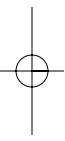
A Complete Guide to
C++

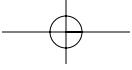
Translated by
Ian Travis

Dedicated to our children
Vivi and Jeany

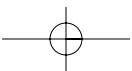
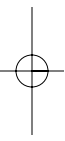


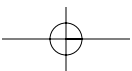
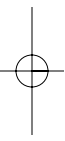
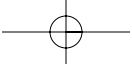
CIP page





Dedication







preface

This book was written for readers interested in learning the C++ programming language from scratch, and for both novice and advanced C++ programmers wishing to enhance their knowledge of C++. It was our goal from the beginning to design this text with the capabilities of serving dual markets, as a textbook for students and as a holistic reference manual for professionals.

The C++ language definition is based on the American National Standards Institute *ANSI Standard X3J16*. This standard also complies with ISO norm 14882, which was ratified by the International Standardization Organization in 1998. The C++ programming language is thus platform independent in the main with a majority of C++ compilers providing ANSI support. New elements of the C++ language, such as exception handling and templates are supported by most of the major compilers. Recommended compilers available through Jones and Bartlett Publishers:

Borland C++Builder 4 Foundation
Metrowerks CodeWarrior Academic Learning Edition
Microsoft Visual C++ 4.0

The **chapters** in this book are organized to guide the reader from elementary language concepts to professional software development, with in depth coverage of all the C++ language elements en route. The order in which these elements are discussed reflects our goal of helping the reader to create useful programs at every step of the way.

Each *double page* spread in the book is organized to provide a description of the language elements on the right page while illustrating them by means of graphics and sample programs on the left page. This type of visual representation offered by a folio will provide students and professionals with an unmatched guide throughout the text. The sample programs were chosen to illustrate a typical application for each language element. In addition, filter programs and case studies introduce the reader to a wide range of application scenarios.

To gain command over a programming language you need a lot of experience in developing programs. Thus, each chapter comprises *exercises* followed by *sample solutions*, allowing the reader to test and enhance his or her performance and understanding of C++.

The *appendix* provides further useful information, such as binary number representation, pre-processor directives, and operator precedence table making this book a well structured and intelligible reference guide for experienced C++ programmers.

In order to test and expand you acquired knowledge you can *download* sample programs and solutions to the exercises at:

<http://completecpp.jpup.com>

Content Organization

Chapter 1 gives a thorough description of the fundamental characteristics of the object-oriented C++ programming language. In addition, students are introduced to the steps necessary for creating a fully functional C++ program. Many examples are provided to help enforce these steps and to demonstrate the basic structure of a C++ program.

Chapter 2 provides a complete introduction to the basic types and objects used by C++ programs. Integral types and constants, fundamental types, and Boolean constants are just a few of the topics discussed.

Chapter 3 describes how to declare and call standard functions. This chapter also teaches students to use standard classes, including standard header files. In addition, students work with string variables for the first time in this chapter.

Chapter 4 explains the use of streams for input and output, with a focus on formatting techniques. Formatting flags and manipulators are discussed, as are field width, fill characters, and alignment.

Chapter 5 introduces operators needed for calculations and selections. Binary, unary, relational, and logical operators are all examined in detail.

Chapter 6 describes the statements needed to control the flow of a program. These include loops with while, do-while, and for; selections with if-else, switch, and the conditional operator; and jumps with goto, continue, and break.

Chapter 7 provides a thorough introduction to the definition of symbolic constants and macros, illustrating their significance and use. Furthermore, a comprehensive examination of standard macros for character handling is included.

Chapter 8 introduces implicit type conversions, which are performed in C++ whenever different arithmetic types occur in expressions. Additionally, the chapter explores an operator for explicit type conversion.

Chapter 9 takes an in-depth look at the standard class string, which is used to represent strings. In addition to defining strings, the chapter looks at the various methods of string manipulation. These include inserting and erasing, searching and replacing, comparing, and concatenating strings.

Chapter 10 describes how to write functions of your own. The basic rules are covered, as are passing arguments, the definition of inline functions, overloading functions and default arguments, and the principle of recursion.

Chapter 11 gives a thorough explanation of storage classes for objects and functions. Object lifetime and scope are discussed, along with global, static, and auto objects. Namespaces and external and static functions are also included in the discussion.

Chapter 12 explains how to define references and pointers and how to use them as parameters and/or return values of functions. In this context, passing by reference and read-only access to arguments are introduced.

Chapter 13 provides a complete description of how classes are defined and how instances of classes, that is, objects, are used. In addition, structs and unions are introduced as examples of special classes.

Chapter 14 describes how constructors and destructors are defined to create and destroy objects. Also discussed is how inline methods, access methods, and read-only methods can be used. Furthermore, the chapter explains the pointer `this`, which is available for all methods, and what you need to pay attention to when passing objects as arguments or returning objects.

Chapter 15 gives a complete explanation of member objects and how they are initialized, and of data members that are created once only for all the objects in a class. In addition, this chapter describes constant members and enumerated types.

Chapter 16 takes an in-depth look at how to define and use arrays. Of particular interest are one-dimensional and multidimensional arrays, C strings, and class arrays.

Chapter 17 describes the relationship between pointers and arrays. This includes pointer arithmetic, pointer versions of functions, pointers as return values and read-only pointers, and pointer arrays. Students learn that operations that use C strings illustrate how to use pointers for efficient programming, and that string access via the command line of an application program is used to illustrate pointer arrays.

Chapter 18 explains sequential file access using file streams. Students will develop an understanding of how file streams provide simple and portable file handling techniques.

Chapter 19 provides a complete description of the various uses of overloaded operators. Arithmetic operators, comparisons, the subscript operator, and the shift operators for input and output are overloaded to illustrate the appropriate techniques. In addition, the concept of friend functions, which is introduced in this context, is particularly important for overloading operators. Students learn how overloading operators allows you to apply existing operators to objects of class type.

Chapter 20 discusses how implicit type conversion occurs in C++ when an expression cannot be compiled directly but can be compiled after applying a conversion rule. The programmer can stipulate how the compiler will perform implicit type conversion for classes by defining conversion constructors and functions. Finally, the chapter discusses ambiguity occurring due to type conversion and how to avoid it.

Chapter 21 describes how a program can allocate and release memory dynamically in line with current memory requirements. Dynamic memory allocation is an important factor in many C++ programs, and the following chapters contain several additional case studies to help students review the subject.

Chapter 22 explains how to implement classes containing pointers to dynamically allocated memory. These include your own copy constructor definition and overloading the assignment operator. A class designed to represent arrays of any given length is used as a sample application.

Chapter 23 provides a thorough description of how derived classes can be constructed from existing classes by inheritance. In addition to defining derived classes, this chapter discusses how members are redefined, how objects are constructed and destroyed, and how access control to base classes can be realized.

Chapter 24 discusses implicit type conversion within class hierarchies, which occurs in the context of assignments and function calls. Explicit type casting in class hierarchies is also described, paying particular attention to upcasting and downcasting.

Chapter 25 gives a complete explanation of how to develop and manage polymorphic classes. In addition to defining virtual functions, dynamic downcasting in polymorphic class hierarchies is introduced.

Chapter 26 describes how defining pure virtual methods can create abstract classes and how you can use abstract classes at a polymorphic interface for derived classes. To illustrate this, an inhomogeneous list, that is, a linked list whose elements can be of various class types, is implemented.

Chapter 27 describes how new classes are created by multiple inheritance and explains their uses. Besides introducing students to the creation and destruction of objects in multiply derived classes, virtual base classes are depicted to avoid ambiguity in multiple inheritance.

Chapter 28 explains how a C++ program uses error-handling techniques to resolve error conditions. In addition to throwing and catching exceptions, the chapter also examines how exception specifications are declared and exception classes are defined. In addition, the use of standard exception classes is discussed.

Chapter 29 examines random access to files based on file streams, and options for querying file state. Exception handling for files is discussed as well. In addition, the chapter illustrates how to make objects in polymorphic classes persistent, that is, how to save them in files. The applications introduced in this chapter include simple index files and hash tables.

Chapter 30 provides a thorough explanation of the advanced uses of pointers. These include pointers to pointers, functions with a variable number of arguments, and pointers to functions. In addition, an application that defines a class used to represent dynamic matrices is introduced.

Chapter 31 describes bitwise operators and how to use bit masks. The applications included demonstrate calculations with parity bits, conversion of lowercase and capital letters, and converting binary numbers. Finally, the definition of bit-fields is introduced.

Chapter 32 discusses how to define and use function and class templates. In addition, special options, such as default arguments, specialization, and explicit instantiation are

discussed. Students learn that templates allow the construction of functions and classes based on types that have not yet been stated. Thus, templates are a powerful tool for automating program code generation.

Chapter 33 explains standard class templates used to represent containers for more efficient management of object collections. These include sequences, such as lists and double ended queues; container adapters, such as stacks, queues, and priority queues; associative containers, such as sets and maps; and bitsets. In addition to discussing how to manage containers, the chapter also looks at sample applications, such as bitmaps for raster images, and routing techniques.

Additional Features

Chapter Goals A concise chapter introduction, which contains a description of the chapter's contents, is presented at the beginning of each chapter. These summaries also provide students with an idea of the key points to look for throughout the chapter.

Chapter Exercises Each chapter contains exercises, including programming problems, designed to test students' knowledge and understanding of the main ideas. The exercises also provide reinforcement for key chapter concepts. Solutions are included to allow students to check their work immediately and correct any possible mistakes.

Case Studies Every chapter contains a number of case studies that were designed to introduce the reader to a wide range of application scenarios.

Notes This feature provides students with helpful tips and information useful to learning C++. Important concepts and rules are highlighted for additional emphasis and easy access.

Hints These are informative suggestions for easier programming. Also included are common mistakes and how to avoid making them.

Acknowledgements

Our thanks go out to everyone who helped produce this book, particularly to

Ian Travis, who translated the book into English applying both linguistic empathy and his solid computing background

Alexa Doehring, who reviewed all samples and program listings, and gave many valuable hints from the American perspective

Michael Stranz and **Amy Rose** at Jones and Bartlett Publishers, who managed the publishing agreement and the production process so smoothly

Our children, **Vivi** and **Jeany**, who left us in peace long enough to get things finished!

And now all that remains is to wish you, Dear Reader, **lots of fun with C++!**

Ulla Kirch-Prinz
Peter Prinz

