国家精品课程教材

清華大學 计算机系列教材

C++ Programming

C++语言程序设计（英文版）

在线教学版
（中文版）

教学资源 · 练习与测试
互动教学 · 智能学习

郑莉 杨芳 董渊 编著

智学苑
www.izhixue.cn
本教材配套教学网站

扫一扫
登录在线教学平台

清华大学出版社

清華大學 计算机系列教材

C++ Programming

C++语言程序设计（英文版）

郑莉 杨芳 董渊 编著

清华大学出版社
北京

图书在版编目(CIP)数据

C++语言程序设计：英文版/郑莉，杨芳，董渊编著. —北京：清华大学出版社，2015
清华大学计算机系列教材
ISBN 978-7-302-37484-8

Ⅰ. ①C… Ⅱ. ①郑… ②杨… ③董… Ⅲ. ①C语言－程序设计－高等学校－教材－英文 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2014)第170853号

责任编辑：谢　琛　薛　阳
封面设计：常雪影
责任校对：白　蕾
责任印制：沈　露

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 装 者：北京嘉实印刷有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　印　张：28.25　　字　数：686千字
版　　次：2015年6月第1版　　印　次：2015年6月第1次印刷
印　　数：1～2000
定　　价：59.00元

产品编号：059734-01

Preface

The Chinese edition of this book is firstly published in 1999, the second edition is published in 2001, the third edition is published in 2003, and the fourth edition is published in 2010. This English edition is based on the former Chinese editions, which the author took advice widely from readers and colleagues, consulted the latest material and based on his teaching experience to complete.

1. Background of Writing This Book

C++ is an object-oriented programming language, which is evolved from C. C++ has two main characteristics: one is its full compatibility with C and the other is that it supports object-oriented methods.

The object-oriented program design encapsulates data and related operations together to form an interdependent and indivisible whole-object. By abstracting common features of objects of the same category, we can get **class**. Most data in a class can only be processed by the methods encapsulated in the class. A class communicates with the outside world through a simple external interface, and objects communicate with each other through messages. In this way, we can have simple relationships among program modules, and module independency and data security can be ensured. Meanwhile, through inheritance and polymorphism, codes can be well reused, which facilitates both development and maintenance of software.

Because of the outstanding qualities of object-oriented methods, they have now become the major ways to develop large-scale software, and C++ is one of the most widely used object-oriented programming languages.

C++ has long been considered hard to use, and is seldom used as an introduction language to teach. Are C++ and object-oriented program design indeed hard to learn? The answer is no. In fact, when C was firstly created, it was only used by a few professional developers. However, along with the development of computer science, computer technologies have permeated to researches and applications of different subjects. Now C has been widely used by various engineers and technicians, and it has also been used as the introduction programming language in many schools. C++ is fully compatible with C, while it provides stricter and more secure grammars. In this sense, C++ firstly is a better C.

C++ is an Object-Oriented Programming (OOP) language. OOP has once been considered as a comparatively advanced technology. This is because that before the theories of Object-Oriented Analysis (OOA) and Object-Oriented Design (OOD) came up, to write a good object-oriented program, programmers should firstly learn to use object-oriented methods to understand and describe problems. Now, since the works of

understanding problem domains and designing system components have been done during the phases of system analysis and system design, the work of OOP becomes much easier—it is just to write every component of an OOD model by an object-oriented programming language.

The emergence of object-oriented methods is in fact a process that the program design gets back to the roots. Essentially, software development is to correctly understand problems that the software needs to handle and to accurately describe the understandings. The fundamental principle that object-oriented methods emphasize is to develop software directly facing the objective existence, and to apply the ways of human thinking and human expressions to software development. Thus, software development can return back to the real world from the past methods, rules and skills that are extravagantly specialized.

Then, do we need to learn C before learning C++? No. Although C++ is evolved from C, C++ itself is an integral programming language, and it has a completely different design philosophy with C. Our learning course needs not to exactly follow the development course of science technology. Only by mastering the latest theories and technologies quickly can we stand on giant's shoulders.

Thus, we think that C++ can be used as an introduction programming language to learn.

2. Features of This Book

This book is comprehensive, tries to explain problems in simple terms, and has abundant complementary materials.

This book is for programmer beginners. Since the publication of the first editionis in 1999, the book has been used by different majors in many universities including Tsinghua University, and has got good effects.

Using C++ as the introduction programming language for college students, this book not only details the language itself, but also introduces data structures, algorithms, object-oriented design ideas, programming, and the Unified Modeling Language (UML). In each chapter of this book, we firstly introduce the related object-oriented programming ideas and methods, and then expound the necessary grammar through practical examples, explaining its meaning and usage primarily from the aspect of the programming methodology. The purpose of this book is to make readers be able not only to master the C++ language itself, but also to use computer languages to describe simple practical problems and their solutions. However, to describe complex problems, readers still have to learn other object-oriented courses such as object-oriented software engineering.

As a book for programmer beginners, this book aims at explaining complicated subjects in simple terms.

3. Content Abstract

Chapter 1 Introduction

From a development perspective, this chapter firstly introduces the history and the

characteristics of object-oriented programming language, as well as the origin and the primary basic concepts of object-oriented methods. Then it makes a brief introduction on object-oriented software engineering. Finally, the chapter takes a look at how information is represented and stored in computers and the development procedure of programs.

Chapter 2 Elementary C++ Programming

This chapter focuses on the basic knowledge of C++ programming. It firstly introduces the history and the characteristics of the C++ language, then it discusses the basic elements that construct a C++ statement—character sets, keywords, identifiers, and operators etc. The chapter also introduces basic data types and user-defined data types in C++, and three main control structures in algorithms: sequential, case and loop structures.

Chapter 3 Functions

This chapter focuses on the functions in C++. In object-oriented programming, function is the basic unit of module division, the basic abstract unit of problem-solving processes, and also the abstract of functionalities. Using functions offers support for code reuse. From an application perspective, this chapter mainly introduces the definitions and usages of various functions, especially the usages of system functions.

Chapter 4 Class and Object

This chapter firstly introduces the basic idea of object-oriented program design and its main characteristics: abstraction, encapsulation, inheritance and polymorphism. Then, revolving around encapsulation, the chapter focuses on the core concept of object-oriented methods—class, including the definition and the implementation of class, and how to use class to solve practical problems. Finally, it briefly introduces how to use Unified Modeling Language (UML) to describe the characteristics of class. Later chapters will always use UML to describe the relationships between class and object.

Chapter 5 Data Sharing and Protecting

This chapter introduces the scope and the visibility of identifiers, and the lifetime of variables and objects. We can see how to use local variables, global variables, data members of classes, static members of classes, and friends to achieve data sharing and protection of shared data. Finally, the chapter introduces how to use multifile structures to organize and write programs to solve complex problems.

Chapter 6 Arrays, Pointers and Strings

This chapter focuses on arrays, pointers and strings. Array and pointer are the most commonly used compound (structure) data types. They are the primary means by which we organize and represent data and objects, and are the useful tools to manipulate math operations. This chapter firstly introduces the basic concepts of arrays and pointers, and discusses the dynamic memory allocation. Then, revolving around the organization issues of data and objects, the chapter focuses on how to use arrays and pointers to link and coordinate data, functions and objects. Finally, the chapter introduces the concept of

string and two methods to process strings: using character arrays and using the class *string*.

Chapter 7 Inheritance and Derivation

This chapter focuses on the inheritance characteristic of class. Revolving around the derivation process, the chapter primarily discusses the access control issues of base class members under different inheritance modes, as well as how to add constructor and destructor in a derived class. Then, the chapter discusses the issues of unique identification and access of class members in comparatively complex inheritance relations. Finally, the chapter gives two instances of class inheritance—"Use Complete Gaussian Pivoting Elimination Method to Solve Linear Equations" and "Personnel Information Management Program for a Small Company".

Chapter 8 Polymorphism

This chapter introduces another important characteristic of class—polymorphism. Polymorphism refers to that a same message can result in different actions when received by different kinds of objects. Polymorphism is a re-abstract of specific function members of a class. C++ supports many forms of polymorphism, and the main forms include overloading (include function overloading and operator overloading) and virtual functions, which are also the learning focus. Finally, the chapter gives two instances of class polymorphism—"Variable-Step Trapezoid Integral Algorithm" and "Improvement of Personnel Information Management Program for a Small Company".

Chapter 9 Collections and the Organization of Collection Data

A collection refers to a set of data elements. Collections can be divided into two main categories: linear collections and non-linear collections. This chapter mainly introduces some commonly used collection class templates.

The organization issues of collection data refers to the sorting and the searching methods of the data elements in a collection. Sorting is also called classification or reorganization. It is a process of making an unordered array ordered. Searching is the process of finding specific data elements in an array by some specific method.

Chapter 10 Generic Programming and Standard Template Library

Generic Programming is to write programs as general as possible without loss of efficiency. This chapter briefly introduces some concepts and terms that C++ Standard Template Library (STL) involves, as well as the structure of STL and the usage of its primary components. We focus on the basic applications of containers, iterators, algorithms and function objects, in order to give readers a conceptual understanding of STL and generic programming.

Chapter 11 I/O Stream Library and Input/Output

This chapter introduces the concept of stream, as well as the structure and usage of the stream library. Like C, there is no Input/Output statement in C++. However, the compiler of C++ has an object-oriented I/O software packet which is the I/O stream

library.

Chapter 12 Exception Handling

This chapter focuses on the exception handling. Exception is a kind of program-defined errors. In C++, exception handling refers to a set of implementation mechanisms that handles predicted errors in the run-time of programs. *Try*, *throw* and *catch* statements are the mechanisms in C++ to implement exception handling. With exception handling of C++, programs can deliver unexpected events to execution contexts of higher levels, and thus to better recover from these exceptions.

4. User's Guide and Related Resources

The author assigns 32 class hours for teaching with this book, 32 class hours for experiments, and 32 class hours for computer practices outside class. Thus, there are 96 class hours in and out of class, and each class hour has 45 minutes. We recommend distributing the teaching hours as follows:

Chapter 1: 2 class hours; Chapter 2: 4 class hours; Chapter 3: 2 class hours; Chapter 4: 4 class hours; Chapter 5: 2 class hours; Chapter 6: 4 class hours; Chapter 7: 4 class hours; Chapter 8: 2 class hours; Chapter 9: 4 class hours; Chapter 10: 2 class hours; Chapter 11: 1 class hour; Chapter 12: 1 class hour.

The readers can download the learning resources from the Tsinghua University Press Web site.

5. Acknowledgement

In the Chinese version of this book, Chapters 1～3, 9, 11 and 12 are written by Zheng Li; Chapters 4～8 are written by Dong Yuan, Zheng Li and Zhang Ruifeng; Chapter 10 is written by Zhang Ruifeng and Zheng Li. Yang Fang rewrite the book in English based on the Chinese version. Besides, Zhou Zhiwei, Dai Nike, Wang Jing, Shan Liang, Mai Haohui, Liu Yintao, Xu Chen, Fu Shixing, Tian Rongpai, Meng Hongli, Meng Wei, Zhang Wenju, Yang Xingpeng and Wang Xuan participated in parts of the writing work.

Thanks readers for using this book, any criticisms and suggestions are warmly welcomed. With your reply please specify your e-mail address. The email addresses of the author is: zhengli@tsinghua.edu.cn.

Contents

Chapter 1 Introduction

This chapter briefly introduces the history and characteristics of object-oriented programming languages, the origin of object-oriented method and its basic concepts, and the definition of Object-Oriented Software-Engineering. Besides, we will introduce how information is represented and stored in a computer and the development process of a program.

1.1 The Development of Computer Programming Language

Language is a system with a set of grammatical and morphological rules. Language is a tool for thinking and the thoughts are expressed by languages. **Computer programming language is a language that can be recognized by computers. It describes solutions to problems, which can be read and executed by computers.**

1.1.1 Machine Language and Assembly Language

Since the birth of the first digital computer in the world—ENIAC, in February 1946, computer science has developed rapidly in the past 60+ years. Computer and its applications have penetrated into various areas of the society, effectively promoting the development of the whole information society, wherein computer has become an essential tool.

Computer System consists of software and hardware. It is not only the strong hardware but also the software system that makes computer system so powerful. **The software system consists of all the programs a computer needs for running and relevant documents.** The work of computer is controlled by programs, and computer can do nothing without a program. A program is a set of instructions. Software engineers translate their solutions to problems and their procedures into a series of instructions, which make up of programs, and input these programs into the computer storage system. The computer executes the instruction sequence to complete the scheduled task.

The so-called instructions are commands that can be recognized by computers. We know that every ethnic group has rich languages for expression, communication and recordation, while these languages are difficult for computer to recognize. The only instruction type that computer can recognize is simple combinations of 0s and 1s. **The set of instructions that can be recognized by the hardware system of a computer is called the instruction system.**

All binary instructions that can be recognized by the hardware system constitute the machine language. Undoubtedly, though machine language is easy for computer to recognize, it is too obscure for human beings to understand, let alone to remember. However, in the early years of computers software engineers can only use machine language to write pro-