Preface

This book draws on the author's extensive teaching experience on C++ and takes its current form after incorporating valuable advices from colleagues and students.

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 both data and related operations to form an interdependent and indivisible whole – an object. By abstracting common features of objects of the same category, we can get a **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 the 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 for teaching. Are C++ and object-oriented program design indeed hard to learn? The answer is no. In fact, when C was first created, it was only used by a few professional developers. However, along with the development of computer science, computer technologies have permeated research 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, although it provides a stricter and more secure grammar. In this sense, C++ is primarily a better C.

C++ is an Object-Oriented Programming (OOP) language. OOP has once been considered a comparatively advanced technology. This is because before the theories of Object-Oriented Analysis (OOA) and Object-Oriented Design (OOD) were developed, in order to write a good object-oriented program, programmers would first learn to use object-oriented methods to understand and describe problems. Now, since the work of understanding problem domains and designing system components are 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 with an object-oriented programming language.

https://doi.org/10.1515/9783110471977-201

The emergence of object-oriented methods is in fact a process where the program design gets back to its roots. Essentially, software development aims 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 past methods, rules and skills that are extravagantly specialized.

Thus, 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 from C. Our learning course does not need to exactly follow the development course of science and technology. Only by mastering the latest theories and technologies quickly can we stand on the shoulders of giants.

Thus, we think that C++ can be taught as an introduction programming language.

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 edition in 1999, the book has been used by different majors in many universities including Tsinghua University, and has achieved 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 and programming, and the Unified Modeling Language (UML). In each chapter of this book, we first 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 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 programming beginners, this book aims at explaining complicated subjects in simple terms.

3. Content Abstract

Chapter 1: Introduction
From a development perspective, this chapter first 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 first 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, 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 first 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 using 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 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 the protection of shared data. Finally, the chapter introduces using 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 for manipulating math operations. This chapter first introduces the basic concepts of arrays and pointers, and discusses 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 strings 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 the constructor and destructor in a derived class. Then, the chapter discusses the issues of unique identification and the 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 how 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 nonlinear collections. This chapter mainly introduces some commonly used collection class templates.

The organization issues of collection data refer to the sorting and 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 writing programs as general as possible without loss of efficiency. This chapter briefly introduces some concepts and terms that are involved in the C++ Standard Template Library (STL), 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 error. In C++, exception handling refers to a set of implementation mechanisms that handles predicted errors in the runtime of programs. *Try*, *throw* and *catch* statements are the mechanisms in C++ to implement exception handling. With the exception handling of C++, programs can deliver unexpected events to execution contexts at higher levels, and thus 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 practice 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 hours; Chapter 12: 1 class hour.

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

5. Acknowledgement

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; and Chapter 10 are written by Zhang Ruifeng and Zheng Li. Yang Fang took great efforts to rewrite this book in fluent English prose. Additionally, 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.

Thank you to the readers for using this book; any criticisms or suggestions are warmly welcomed. In your note, please specify your email address. The email address of the author is: zhengli@tsinghua.edu.cn