国外计算机科学教材系列

国家教育部
双语教学示范课程
使用教材

双语版
C++程序设计
（第2版）

[爱尔兰] Paul Kelly
苏小红 著

Learn C++
through English and Chinese, Second Edition

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

国外计算机科学教材系

双语版C++ 程序设计

（第2版）

Learn C++ through English and Chinese
Second Edition

［爱尔兰］ Paul Kelly 著
苏小红

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内容简介

本书由在计算机程序设计方面有着丰富教学和实践经验的中外作者合作编写。本书内容共分14章，由浅入深、全面介绍C++程序设计方法。本书通俗易懂，例子贴近生活，尤其强调读者的亲自参与意识。所有实例经过精心挑选。每章都为初学者提供了常见错误分析，每章结尾有很多有趣的习题，可以提高读者上机编程的兴趣。

本书是国内首次出版的中英文对照混排式双语版C++程序设计教材的更新版，既方便初学者熟悉相关概念和内容，也便于英文非母语的读者熟悉英文专业词汇。

本书可作为高等学校计算机、软件工程和其他理工类专业的C++程序设计双语教材，也可供程序员和编程爱好者参考使用。

图书在版编目（CIP）数据

双语版C++程序设计：汉英对照/（爱尔兰）凯利（Kelly, P.），苏小红著. —2版.
北京：电子工业出版社，2016.7
国外计算机科学教材系列
ISBN 978-7-121-29358-0

Ⅰ. ①双… Ⅱ. ①凯… ②苏… Ⅲ. ①C语言—程序设计—高等学校—教材—汉、英
Ⅳ. ①TP312

中国版本图书馆CIP数据核字（2016）第157959号

策划编辑：马 岚
责任编辑：冯小贝
印 刷：三河市华成印务有限公司
装 订：三河市华成印务有限公司
出版发行：电子工业出版社
北京市海淀区万寿路173信箱 邮编：100036
开 本：787×1092 1/16 印张：22 字数：716千字
版 次：2010年6月第1版
2016年7月第2版
印 次：2016年7月第1次印刷
定 价：55.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。
质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。
本书咨询联系方式：classic-series-info@phei.com.cn。

Preface

This textbook teaches the fundamentals of programming using C++, a programming language which supports the development of software using the object-oriented paradigm.

Although the book is primarily intended as a textbook for a programming module in a computer science course, it is equally suited to an individual familiar with another programming language and who now wants to learn how to program in C++.

This book focuses on core concepts and features of C++ while keeping the explanations as simple as possible. Drawing on their professional experience, the authors teach C++ programming largely by way of examples that are organised for easy step-by-step learning.

Like so many other programming languages, C++ contains numerous English technical terms that are difficult for all students, including the native English speaking student.

Learn C++ through English and Chinese explains C++ concepts and terminology in English with additional explanatory annotations in Chinese. This bi-lingual approach will be appreciated by Chinese students and will help them focus on the C++ language without being over-burdened by English technical terminology. Despite C++ being available on a wide variety of platforms, this book is not specific to any particular machine, compiler, or operating system. All programs are designed to be portable with little or no modification to a wide variety of platforms.

Learn C++ through English and Chinese

- Is a comprehensive introduction to programming in C++.
- Uses practical examples to explain difficult theoretical examples.
- Uses a step-by-step approach with detailed explanation of programming examples.
- Uses explanatory annotations written in Chinese.
- Provides end-of-chapter 'programming pitfalls' commonly experienced by learners.
- Provides a 'quick syntax reference' at the end of each chapter that summarises the C++ syntax covered in the chapter. This is a useful resource for experienced programmers as well as for learners.
- Provides end-of-chapter exercises, allowing the learner to test and re-enforce their understanding of C++.
- Is suitable for students new to programming and those familiar with some other language, such as C or Basic, and who now wish to learn C++.
- Is accompanied by a web site containing the example programs, solutions to selected exercises, frequently asked questions and links to other useful resources.

Typographic Conventions

The line numbers to the left of the program examples are for reference purposes only and are not part of the C++ language.

When a new term is introduced it is in *italic* type.

```
C++ statements, keywords, program variables and values are in this font.
```

This font is used in examples to show values that should be typed at the keyboard by the user.

Paul Kelly
Dublin Institute of Technology

前　言

针对国内大学逐渐与国际接轨的发展趋势，英语教学和双语教学逐渐被人们重视起来。一方面，在教育部的大力支持下，许多课程建设成了教育部双语教学示范课程，但是这些课程大多采用英文原版教材，而面向双语教学的双语版教材在国内实属罕见。另一方面，以“国际化、工业化”为办学理念，注重国际化、工业化人才培养的国家示范性软件学院的部分课程还邀请了一些外籍教师进行全英语授课，但是由于目前国内学生的英语水平参差不齐，导致全英语授课的教学效果不是非常理想。本书正是在此背景和需求下应运而生的。

本书的第一作者是爱尔兰都柏林工业大学（DIT）的高级讲师Paul Kelly。Kelly 老师长期从事程序设计类课程的教学工作，在程序设计类课程教学方面教学实践经验丰富，在国外已先后出版多本程序设计语言类书籍。自哈尔滨工业大学软件示范学院成立以来，多次作为外聘教师在哈尔滨工业大学国家示范性软件学院从事程序设计方面的教学工作，对中国学生比较了解，针对其在教学中发现的问题，即初学者面临着既不熟悉专业术语和基本概念、又不熟悉英文专业词汇的双重困难，提出了出版英汉对照混排式双语版教材的思路，帮助学生在克服语言障碍的同时，能够更快更好地熟悉和掌握计算机程序设计方面的基础知识，为国内的双语教学提供了一种最佳的解决方案。

本书内容共分14 章，由浅入深、全面介绍C++程序设计方法，既适合于以C++作为入门语言的读者，也适合于学习过其他程序设计语言后想再学习C++ 的读者。本书的特点如下：

1. 使用非常实用和贴近生活的例子以及图示来通俗易懂地讲解难于理解的概念，在介绍面向对象程序设计方法的同时，尤其强调读者的亲自参与意识。
2. 采用案例驱动和循序渐进方式，从一个应用实例出发，先利用现有知识编写出一个较为简单的程序，然后在此基础上，不断扩充，在扩充的过程中引入一个新的概念和知识点，逐渐编写出一个较大的程序，每个例程都有详细的讲解。有的章以一个例子为中心贯穿始终讲解，后面章节的例子还会重用已有的部分程序代码，前后章节之间既有内容上的联系，也有例程上的联系。
3. 重点内容和段落给出了中文注解。
4. 每章后面都有一节介绍初学者编程时易犯的错误，以帮助初学者在程序设计中避免这些错误。
5. 每章后面都有快速语法索引，总结本章内容，便于读者快速查询相关内容。
6. 每章后面都有精心设计的、有趣的习题，便于读者测试和强化对相关内容的理解。
7. 有相关的教学网站（华信教育资源网，网址http://www.hxedu.com.cn），方便读者下载示例的源代码、部分习题解答及教学课件等资料。

本书由在计算机程序设计方面有着丰富教学和实践经验的中外作者合作编写。本书是国内首次出版的中英文对照混排式双语版教材的更新版，适合低年级的学生对照阅读，既方便初学者熟悉相关概念和内容，也便于母语不是英语的读者熟悉英文的专业词汇，尤其适合作为双语教学示范课程的教材。

Paul Kelly是一位治学非常严谨的教师，本书的第二作者苏小红在与他合著过程中，经常为一个细节内容的编写进行反复交流与讨论，书稿完成后又进行了多次校对工作。本着对所有读者负责的精神，我们真诚地欢迎读者对教材提出宝贵意见，可以通过发送电子邮件或在网站（http://book.sunner.cn）上留言等多种方式与我们交流讨论。

作者E-mail地址为Paul.Kelly@comp.dit.ie，sxh@hit.edu.cn。

苏小红

哈尔滨工业大学计算机科学与技术学院

目　录

Chapter One　Typographic Conventions（绪论） …… 1
1.1 What is a computer program?（什么是计算机程序?）…… 1
1.2 Developing a computer program（开发计算机程序）…… 2
1.2.1 Program development cycle …… 2
1.3 Learning C++（学习 C++）…… 4
1.4 Web site for this book（本书的网站）…… 4
1.5 Brief history of C++（C++简史）…… 4
1.6 ANSI/ISO C++ standard（ANSI/ISO C++标准） …… 5

Chapter Two　Beginning to Program in C++（C++编程入门）…… 6
2.1 Constants（常量）…… 6
2.2 Variables（变量）…… 6
2.3 Simple output to the screen（简单的屏幕输出）…… 7
2.4 Comments（注释） …… 9
2.5 Data types（数据类型）…… 10
2.5.1 Short integer data types …… 10
2.5.2 Long integer data types …… 10
2.5.3 Boolean data types …… 11
2.5.4 Double floating-point data types …… 11
2.5.5 Unsigned integer data types …… 11
2.6 Data type sizes（数据类型的大小）…… 11
2.7 Operators（运算符） …… 12
2.7.1 The assignment operator …… 12
2.7.2 Arithmetic operators …… 12
2.7.3 Increment and decrement operators …… 13
2.7.4 Combined assignment operators …… 15
2.8 Operator precedence（运算符的优先级） …… 16
2.9 Data type conversions and casts（类型转换和强转） …… 18
Programming pitfalls …… 20
Quick syntax reference …… 21
Exercises …… 22

Chapter Three　Keyboard Input and Screen Output（键盘输入和屏幕输出） …… 26
3.1 Simple keyboard input（简单的键盘输入）…… 26
3.2 Manipulators（流操纵符） …… 28
3.3 Single-character input and output（单个字符的输入和输出） …… 30
Programming pitfalls …… 32
Quick syntax reference …… 32
Exercises …… 32

Chapter Four Selection and Iteration（选择与循环） ····· 34
4.1 Selection（选择） ····· 34
4.1.1 The `if` statement ····· 34
4.1.2 The `if-else` statement ····· 35
4.1.3 Compound statements ····· 35
4.1.4 Logical operators ····· 37
4.1.5 Nested `if` statements ····· 37
4.1.6 The `switch` statement ····· 37
4.1.7 The conditional operator `?:` ····· 39
4.2 Iteration（循环） ····· 40
4.2.1 The `while` statement ····· 40
4.2.2 The `do-while` loop ····· 42
4.2.3 The `for` statement ····· 43
4.2.4 Nested loops ····· 45
Programming pitfalls ····· 47
Quick syntax reference ····· 49
Exercises ····· 50

Chapter Five Arrays and Structures（数组和结构体） ····· 53
5.1 Arrays（数组） ····· 53
5.1.1 Introduction ····· 53
5.1.2 Initialising an array ····· 56
5.1.3 Two-dimensional arrays ····· 57
5.1.4 Initialising a two-dimensional array ····· 59
5.1.5 Multi-dimensional arrays ····· 60
5.2 Structures（结构体） ····· 60
5.2.1 Introduction ····· 60
5.2.2 Declaring a structure ····· 61
5.2.3 Initialising a structure variable ····· 63
5.2.4 Nested structures ····· 64
5.3 The `typedef` statement（typedef 语句） ····· 65
5.4 Arrays of structures（结构体数组） ····· 66
5.5 Enumerated data types（枚举数据类型） ····· 66
Programming pitfalls ····· 68
Quick syntax reference ····· 68
Exercises ····· 69

Chapter Six Strings（字符串） ····· 72
6.1 C-strings（C 风格字符串） ····· 72
6.2 C-string input and output（C 风格字符串的输入和输出） ····· 73
6.3 Accessing individual characters of a C-string（访问C 风格字符串中的单个字符） ····· 77
6.4 C-string functions（C 风格字符串函数） ····· 77
6.4.1 Finding the length of a C-string ····· 78

6.4.2 Copying a C-string ····· 78
6.4.3 C-string concatenation ····· 79
6.4.4 Comparing C-strings ····· 79
6.4.5 Other C-string functions ····· 79
6.4.6 Converting numeric C-strings to numbers ····· 80
6.5 C++ strings（C++ 字符串） ····· 80
6.5.1 `string` initialisation and assignment ····· 82
6.5.2 `string` concatenation ····· 84
6.5.3 `string length`, `string` indexing and sub-strings ····· 85
6.5.4 `string replace`, `erase`, insert and empty strings ····· 86
6.5.5 `string` searching ····· 88
6.5.6 `string` comparisons ····· 89
6.5.7 `string` input ····· 91
6.5.8 `string` conversions ····· 92
6.6 Arrays of strings（string 类型的数组） ····· 93
6.7 Character classification（字符分类） ····· 94
Programming pitfalls ····· 96
Quick syntax reference ····· 96
Exercises ····· 97

Chapter Seven Functions（函数） ····· 100
7.1 Introduction（引言） ····· 100
7.2 Function arguments（函数实参） ····· 102
7.3 Default parameter values（默认的形参值） ····· 105
7.4 Returning a value from a function（从函数返回一个值） ····· 106
7.5 Inline functions（内联函数） ····· 107
7.6 Passing arguments by value（按值传递实参） ····· 108
7.7 Passing arguments by reference（按引用传递实参） ····· 109
7.8 Passing a one-dimensional array to a function（向函数传递一维数组） ····· 112
7.9 Passing a multi-dimensional array to a function（向函数传递多维数组） ····· 115
7.10 Passing a structure variable to a function（向函数传递结构体变量） ····· 116
7.11 Passing a string to function（向函数传递字符串） ····· 118
7.11.1 Passing a C++ string to a function ····· 118
7.11.2 Passing a C-string to a function ····· 119
7.12 Recursion（递归） ····· 120
7.13 Function overloading（函数重载） ····· 122
7.14 Storage classes `auto` and `static`（auto 和static 存储类型） ····· 123
7.14.1 `auto` ····· 123
7.14.2 `static` ····· 124
7.15 The scope of a variable（变量的作用域） ····· 125
7.15.1 Block scope ····· 125
7.15.2 Global scope ····· 126
7.15.3 Reusing a variable name ····· 127

7.16 Mathematical functions（数学函数） …… 129
7.16.1 Some trigonometric functions …… 129
7.16.2 Pseudo-random number functions …… 130
Programming pitfalls …… 132
Quick syntax reference …… 132
Exercises …… 133

Chapter Eight Objects and Classes（对象和类） …… 137
8.1 What is an object?（什么是对象?） …… 137
8.2 What is a class?（什么是类?） …… 137
8.3 Further examples of classes and objects（类和对象的更进一步的示例） …… 140
8.3.1 A student class …… 140
8.3.2 A bank account class …… 140
8.4 Abstraction（抽象） …… 141
8.5 Constructing a class in C++（用C++构造一个类） …… 142
8.6 Using a class: defining and using objects（使用类：定义和使用对象） …… 144
8.7 Abstract data types（抽象数据类型） …… 145
8.8 Constructors（构造函数） …… 146
8.9 Default class constructor（默认的类构造函数） …… 148
8.10 Overloading class constructors（重载类构造函数） …… 149
8.11 Constructor initialisation lists（构造函数初始化列表） …… 151
8.12 Default argument values in a constructor（构造函数中的默认实参值） …… 152
8.13 `static` class data members（静态类数据成员） …… 154
8.14 Using `return` in a member function（在成员函数中使用return） …… 157
8.15 Inline class member functions（内联成员函数） …… 159
8.16 Class interface and class implementation（类的接口和类的实现） …… 160
8.16.1 Separation of class interface and class implementation …… 162
8.16.2 Use of namespaces in header files …… 164
Programming pitfalls …… 167
Quick syntax reference …… 167
Exercises …… 167

Chapter Nine Pointers and Dynamic Memory（指针和动态内存分配） …… 171
9.1 Variable addresses（变量的地址） …… 171
9.2 Pointer variables（指针变量） …… 172
9.3 The dereference operator *（解引用运算符*） …… 173
9.4 Using `const` with pointers（使用const 修饰指针变量） …… 174
9.5 Pointers and one-dimensional arrays（指针和一维数组） …… 175
9.6 Pointers and multi-dimensional arrays（指针和多维数组） …… 177
9.7 Pointers to structures（指向结构体的指针） …… 178
9.8 Pointers to class objects（指向类对象的指针） …… 179
9.9 Pointers as function arguments（指针变量作为函数实参） …… 180
9.10 Dynamic memory allocation（动态内存分配） …… 182

9.10.1 Allocating memory dynamically for an array ····· 183
9.10.2 Initialisation with `new` ····· 184
9.10.3 Allocating memory for multi-dimensional arrays ····· 186
9.10.4 Out of memory error ····· 187
Programming pitfalls ····· 189
Quick syntax reference ····· 190
Exercises ····· 190

Chapter Ten Operator Overloading（运算符重载） ····· 193
10.1 The need for operator overloading（运算符重载的必要性） ····· 193
10.2 Overloading the addition operator +（重载加法运算符+） ····· 193
10.3 Rules of operator overloading（运算符重载的规则） ····· 200
10.4 Overloading ++（重载运算符++） ····· 200
10.4.1 Overloading prefix and postfix forms of ++ ····· 203
10.4.2 Improving the prefix ++ operator member function ····· 206
10.5 Overloading relational operators（重载关系运算符） ····· 206
10.6 Overloading << and >>（重载运算符<<和>>） ····· 209
10.7 Conversion operators（转换运算符） ····· 214
10.8 Use of `friend` functions（使用友元函数） ····· 217
10.9 Overloading the assignment operator =（重载赋值运算符=） ····· 218
10.9.1 A class with a pointer data member ····· 218
10.9.2 Assigning one object to another ····· 220
10.10 The copy constructor（拷贝构造函数） ····· 229
10.11 Overloading the index operator `[]`（重载下标运算符[]） ····· 233
Programming pitfalls ····· 236
Quick syntax reference ····· 237
Exercises ····· 237

Chapter Eleven Inheritance（继承） ····· 240
11.1 What is inheritance?（什么是继承?） ····· 240
11.2 Inheritance syntax（继承语法） ····· 241
11.3 Passing arguments to a base class constructor（向基类的构造函数传递实参） ····· 249
11.4 Protected class members（受保护的类成员） ····· 253
11.5 Types of inheritance: `public`, `protected` and `private`（继承的类型：public、protected和private） ····· 256
11.6 Composition（组合） ····· 258
11.7 Multiple inheritance（多重继承） ····· 259
11.8 Virtual base classes（虚基类） ····· 262
Programming pitfalls ····· 265
Quick syntax reference ····· 265
Exercises ····· 266

Chapter Twelve Polymorphism（多态） ····· 271
12.1 What is polymorphism?（什么是多态?） ····· 271

12.2 Virtual functions（虚函数）…… 274
12.2.1 When to use virtual functions …… 279
12.2.2 Overriding and overloading …… 279
12.3 Abstract base classes（抽象基类）…… 280
Programming pitfalls …… 284
Quick syntax reference …… 284
Exercises …… 285

Chapter Thirteen Templates（模板）…… 288
13.1 Introduction（引言）…… 288
13.2 Function templates（函数模板）…… 288
13.3 Class templates（类模板）…… 292
Programming pitfalls …… 297
Quick syntax reference …… 297
Exercises …… 297

Chapter Fourteen Files and Streams（文件和流）…… 300
14.1 The C++ input/output class hierarchy（C++ 输入/ 输出类的层次结构）…… 300
14.2 Opening a file（打开文件）…… 301
14.3 File error checking（文件出错检查）…… 303
14.4 Single character I/O and detecting the end of a file（单字符的I/O 和文件末尾的检测）…… 304
14.5 Appending data to the end of a file（向文件末尾添加数据）…… 308
14.6 Reading lines from a file（从文件中读取行）…… 309
14.7 Random access（随机存取）…… 310
14.8 Object I/O（对象I/O）…… 313
14.9 Binary I/O（二进制I/O）…… 314
14.9.1 Serial writing of objects to a binary file …… 315
14.9.2 Serial reading of objects from a binary file …… 319
14.9.3 Binary I/O as class member functions …… 320
14.9.4 Binary file random access …… 321
Programming pitfalls …… 325
Quick syntax reference …… 325
Exercises …… 326

Appendix A List of C++ Keywords …… 329

Appendix B Precedence and Associativity of C++ Operators …… 330

Appendix C ASCII Character Codes …… 332

Appendix D Fundamental C++ Built-in Data Types …… 334

Appendix E Common iomanip Manipulators …… 335

Appendix F Escape Sequences …… 336

Appendix G The C++ Preprocessor …… 337

Chapter One
Introduction
第1章 绪 论

1.1 What is a computer program?
（什么是计算机程序？）

Computers are involved in a wide variety of tasks that we do in our everyday lives. Some of these tasks such as using a word processor or checking e-mail obviously use a computer. Less direct examples occur when we use an ATM at a bank, pay at a supermarket checkout or use a phone.

A computer performs all of these tasks by following a predefined set of instructions. This set of instructions is a called a *computer program*. A computer program to a computer is like a recipe to a chef; it specifies the steps needed to perform a certain task. But unfortunately, unlike a recipe, you can't give your instructions to a computer in a language such as English or Chinese. For instructions to be 'intelligible' to a computer, they need to be expressed in a language 'understood' by the computer. The only language 'understood' by a computer is its own machine language, which consists of a series of binary ones and zeroes.

Machine language is very difficult to use directly and so instructions to a computer are given in a special language called a *programming language*. The programming language is neither English nor machine language, but is somewhere in between. In fact, as you will see, it is more like English than machine language.

Machine languages are known as *low-level language*s and programming languages are known as *high-level languages*.

Writing instructions in a high level language is much easier than writing them in low-level machine language, but is still not as easy as writing them in English or Chinese.

For the computer to carry out the program instructions written in a high level language, they have to be translated from the high level language to the machine language of the computer. A *compiler* does this translation.

现在我们日常生活中的很多工作都要用到计算机。

计算机完成所有这些工作都是通过执行预定义的指令序列来实现的。这个指令序列就称为“计算机程序”。计算机程序和计算机之间，就像食谱和厨师之间的关系一样；计算机程序指定了完成某一任务需要的步骤。

但是遗憾的是，不同于菜谱，您不能用英文或者中文这样的自然语言向计算机发送指令。对于计算机来说，指令必须是“易理解”的，必须表示成一种计算机能“理解”的语言。计算机能“理解”的唯一语言就是机器语言，机器语言由一系列二进制的0和1组成。

由于机器语言很难直接使用，所以计算机指令被表示成一种特殊的语言，这种特殊的语言称为“程序设计语言”。程序设计语言既不是英语，也不是机器语言，而是一种介于它们两者之间的语言。实际上，正如下面即将看到的那样，较之机器语言，程序设计语言更像英语。

机器语言被认为是一种低级语言，而程序设计语言则被认为是一种高级语言。

为了让计算机执行由高级语言编写的程序指令，必须把这些指令从高级语言形式翻译成计算机的机器语言形式。编译器用于完成这种翻译。

1.2 Developing a computer program
（开发计算机程序）

The first step in developing a computer program is to define and understand the problem to be solved. If you cannot understand the problem then you will certainly not be able to tell a computer how to solve the problem. This is called the analysis phase and basically answers the question "what is to be done?", ignoring for the time being the question of how the problem is to be solved.

Once it is known what has to be done, the question "how is it to be done?" arises. This is called the design phase and it is in this phase that a solution to the problem is developed. The design phase may reveal problems not previously considered in the analysis phase. So rather than being independent phases, the design and analysis phases are closely related and interact with each other.

The analysis and design phases are important to developing a successful solution to a problem. Neglect at either of these phases will result in a 'solution' that will not solve the original problem and may even contribute to making it worse.

Analysis and design are subjects in their own right and are not covered in this book. This book concentrates on the next phase: writing, compiling and testing C++ programs.

There are many different compilers on the market, with updated versions being released regularly. Up to date instructions for writing, compiling and running C++ programs with some of the popular compilers can be found on the web site for this book at http://www.hxedu.com.cn.

开发计算机程序的第一步就是定义和理解要解决的问题。

如果不理解要解决的问题是什么，那么就无法告诉计算机如何去解决这个问题。这个阶段称为分析阶段。分析阶段主要是回答"做什么"的问题，而不考虑如何去做的问题。

一旦知道了要做什么，那么"如何去做"的问题又产生了。这称为设计阶段，在这个阶段要找到一种解决问题的方法。在设计阶段也可能遇到一些在分析阶段没有考虑到的问题。所以，分析和设计不是两个独立的阶段，而是两个密切相关、相互影响的阶段。分析和设计阶段对找到一种成功的解决问题的方案来说是十分重要的。忽视这两个阶段中的任何一个都会导致最终的解决方案无法解决最初的问题，甚至还可能产生更坏的结果。

分析和设计属于不同范畴的问题，不在本书的讨论范围之内。本书重点介绍其后面的阶段：C++程序的编写、编译和测试。

尽管编译器之间存在差异，但可以给出开发一个C++程序的一般步骤。

1.2.1 Program development cycle

Despite the differences between compilers, the following is a general description of the steps involved in the development of a C++ program.

Step 1: Design the program.

A computer system consists of one or more programs. Each program has to be individually designed to implement the overall solution developed at the analysis and design phases.

After each program is designed, it is important to check its logic before starting to write the program.

第一步：设计程序。

计算机系统由一个或者多个程序组成。每个程序必须单独设计，以实现在分析阶段和设计阶段提出的整体解决方案。

在程序设计好之后、编写程序之前，检查一下程序的逻辑是很重要的。

Step 2: Write the program.

Firstly, the C++ program instructions are typed into a file using a *text*

第二步：编写程序。

使用文本编辑器把 C++程序指

editor. The file containing the C++ statements is called the *source file* and is usually stored on disk. The program instructions are also called the *source code* or the *program code*.

令输入到一个文件中。这个包含C++语句的文件称为源文件，通常存储在磁盘上。程序指令也称为源代码或程序代码。

Step 3: Compile the program.

Next the C++ program is passed through a compiler, which translates the C++ program instructions to machine instructions. The compiler reads the source file, translates the C++ statements into machine or *object code*, and stores the object code in an *object file*.

Some errors are likely to occur in this step. An error in the source code is indicated by the compiler and is referred to as a *compile-time error*. The simplest kind of compile-time error is a *syntax error*. This type of error is relatively easy to correct, as the compiler will indicate where in the source code the error has occurred. Typical syntax errors involve missing punctuation and misspellings. All compile-time errors must be corrected before the compilation process can be completed.

The compiler may sometimes issue a warning message during compilation. A warning message is not as serious as a syntax error and will not prevent the program from being compiled. However, warnings are the compiler's way of drawing attention to what it 'thinks' may be a problem and should be investigated.

第三步：编译程序。

编译器把C++程序指令转换成机器指令。编译器读入源文件，把C++语句翻译成机器代码或目标代码，然后把目标代码存入目标文件中。

在编译阶段很可能出现一些错误。编译器指出的源代码中的错误属于编译时错误。最简单的编译时错误是语法错误。这类错误相对容易纠正，因为编译器会指出这类错误在源代码中的位置。典型的语法错误有遗漏标点和拼写错误。在编译过程完成之前，必须纠正所有的编译错误。

编译器也可能在编译阶段发出一些警告信息。警告信息不像语法错误那样严重，不会妨碍程序的编译。不过，它表示编译器认为可能有问题，希望引起您的注意并检查一下程序。

Step 4: Link the program.

The final step before running the program is to *link* the program using the *linker*. Linking involves combining the object file of the program with other object files from the C++ run-time library to form an *executable file*.

第四步：链接程序。

在运行程序之前的最后一步就是使用链接器来链接程序。

所谓链接就是把程序的目标文件和C++运行时库文件结合起来形成一个可执行文件。

Step 5: Test the program.

When the program is run you may find that it is not working as expected. The fact that a program does not have any compile-time errors does not guarantee that the program will work as required. For example, the programmer may mistakenly have given an instruction to divide a number by zero. This type of error is called a *run-time error* and causes the program to stop before it has completed its task.

The program may complete its task but produce incorrect results or display them at the wrong position on the screen.

These kinds of errors are known as *logic errors* or more commonly as *bugs*. Bugs are much harder to find and fix than compile-time errors.

第五步：测试程序。

程序在运行时有可能没有像预期的那样执行。一个程序没有编译时错误并不能保证这个程序就能按照要求运行。举个例子，编程人员可能错误地在一条指令中把0作为除数。这种错误称为运行时错误，它会导致程序在完成它的任务之前中止。

虽然程序执行完毕，但是结果却是错误的，或者结果显示的屏幕位置是错误的。这类错误称为

Some bugs appear only under certain conditions, for example when a program is run with a particular set of data.
The process of locating and correcting program errors is called *debugging*.

逻辑错误或者俗称为“bugs（缺陷）”。缺陷比编译时错误更难发现和修正。某些 bug仅在特定的条件下才出现，例如仅在特定的数据集上运行程序时才出现。
定位和修正程序错误的过程，称为调试。

Step 6: Debug the program.
Once a bug has been identified, the next step is to find where in the source code the problem lies. Many compilers have tools that can be used to help locate bugs.
Correcting bugs involves going back to step 2, but hopefully not any further. Going back to step 1 or even back to the analysis and design phases would be like asking an architect to redesign parts of a house while it is being built! In general, try to catch errors as early as possible.

第六步：调试程序。
bug一旦被确认，紧接着就要定位其在源代码中的位置。很多编译器都提供了可用于帮助定位缺陷的工具。
改正一个缺陷后要返回第二步，但是希望不要返回到更前面去。如果返回到第一步甚至是分析和设计阶段，就好像要求建筑师重新设计一座正在建造中的房子一样。一般情况下，应该尽可能早地发现错误。

1.3 Learning C++（学习 C++）

You'll learn more from designing, writing, running, and correcting programs than you ever will by simply reading a book. A successful approach to learning to program in C++ depends on large amounts of practice.
To help you practise, there are exercises at the end of each chapter. Do as many of the exercises as possible and get some feedback on your solutions from people who know C++. There are solutions to selected exercises at the web site for this book.

1.4 Web site for this book（本书的网站）

The web site for this book is at http://www.hxedu.com.cn. The source code for all the example programs used in this book as well as answers to selected exercises are available here. In addition, The source code is also available at https://db.tt/e6Uwrli5.

1.5 Brief history of C++（C++ 简史）

C++ is a direct descendent of the C programming language, which was originally developed in 1972 at Bell Laboratories, New Jersey, USA. C evolved from a language called B, which in turn evolved from a language called BCPL (Basic Combined Programming Language). C++ was developed by Bjarne Strostrup at AT&T Bell Laboratories from 1979 to 1983. The initial version of the language was called “C with Classes” and was used internally in AT&T in 1983. Later that year, the name was changed to “C++”. The

C++语言是由C语言直接发展而来的，1972年诞生于美国新泽西州的贝尔实验室。C语言是从一种称为B的语言演化而来的，这种B语言反过来又是从 BCPL语言演化而来的。
C++是Bjarne Strostrup于1979年至 1983年在 AT&T贝尔实验室