

高等教育质量工程信息技术系列示范教材

# 新概念

# C++程序设计 大学教程

张基温 编著

//



清华大学出版社

013023681

TP312C-43  
786

内容简介

本书是“高等教育质量工程信息技术系列示范教材”之一，由清华大学出版社出版。本书共分10章，主要介绍C++语言的基本概念、数据类型、运算符、表达式、语句、函数、数组、指针、字符串、文件、模板、命名空间、异常处理、多线程、STL、C++11新特性等。本书可作为高等院校计算机专业及相关专业的教材，也可供从事计算机工作的工程技术人员参考。

# 新概念 C++程序设计大学教程

张基温 编著



TP312C-43  
786

清华大学出版社  
北京



北航 C1630525

18333281

## 内 容 简 介

本书是作者在多年的教学实践中摸索出的一套全新概念的 C++ 程序设计教学体系。全书分为 3 篇 15 个单元:第 1 篇共 6 个单元,前 5 个单元用于训练面向对象程序设计的基本思维和方法,其中穿插介绍一些最基本的 C++ 语法;第 6 单元介绍面向对象程序设计的几个基本原则及 GoF 设计模式。第 2 篇共 5 个单元,主要介绍 C++ 常量的表示、数组、存储属性、异常处理、动态内存分配等重要语法知识。第 3 篇共 4 个单元,主要介绍 C++ 流类、函数细节、类型变换与运行时鉴别和模板。

本书理念先进、概念清晰、讲解透彻、便于理解,例题经典、习题丰富、覆盖面广;适合作为高等学校各专业学生的程序设计教材,可供培训机构使用,也可供相关领域人员自学。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

### 图书在版编目(CIP)数据

新概念 C++ 程序设计大学教程 / 张基温编著. --北京:清华大学出版社, 2013. 3

高等教育质量工程信息技术系列示范教材

ISBN 978-7-302-31255-0

I. ①新… II. ①张… III. ①C 语言—程序设计—高等学校—教材 IV. ①TP312

中国版本图书馆 CIP 数据核字(2013)第 002395 号

责任编辑:白立军 顾 冰

封面设计:常雪影

责任校对:时翠兰

责任印制:何 芊

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](mailto:c-service@tup.tsinghua.edu.cn)

质 量 反 馈:010-62772015, [zhiliang@tup.tsinghua.edu.cn](mailto:zhiliang@tup.tsinghua.edu.cn)

课 件 下 载: <http://www.tup.com.cn>, 010-62795954

印 刷 者:北京富博印刷有限公司

装 订 者:北京市密云县京文制本装订厂

经 销:全国新华书店

开 本:185mm×260mm

印 张:25

字 数:600 千字

版 次:2013 年 3 月第 1 版

印 次:2013 年 3 月第 1 次印刷

印 数:1~3000

定 价:39.50 元

产品编号:046563-01

# 前 言

程序设计是 IT 类专业工作者的看家本领,也是人们为解决复杂问题所需要的基本思维训练。但是在多年的教学实践和外出讲学调研中,笔者却发现,目前程序设计教学中普遍存在的三个突出问题:学习了程序设计课程,但碰到问题还是不知道如何下手;虽能编写程序,但用 C++ 语言编写出来的程序却是面向过程的;编写出了程序却不知道如何测试。

## 1. 内容体系与写作思想

本书的写作目的就是试图从上述三个方面实现一些突破,改善 C++ 的教学效果。全书分为 3 篇。第 1 篇共 6 个单元,第 1 单元介绍面向对象的基本概念;接着的 4 个单元各用一个实例帮助学习者快速进入面向对象世界,并掌握不同程序的基本测试方法;第 6 单元通过介绍面向对象程序设计的基本原则,使读者知晓如何设计出优美的面向对象的程序。第 2 篇用 5 个单元介绍 C++ 支持面向对象程序设计的重要机制,使读者在学习了第 1 单元后,能在面向对象程序设计上上一个台阶。第 3 篇用 4 个单元使读者能进一步了解 C++ 的一些细节。

采用这样的结构是出于如下几点考虑。

### (1) 逻辑思维训练先行。

目前几乎所有的程序设计类教材都是采用面向语法的体系。这种从语言的语法手册改写而成的教材,尽管有人进行了“浅显易懂”的加工,说到底还是囿于应试,要把学习者的注意力引导到语法细节而不是程序设计的思路,就会出现虽然学过程序设计,但遇到问题不知道如何下手的后果。

反思当前程序设计教材的这种弊病,本书第 1 篇采用了问题体系的写法,目的是把教材为中心的教学体系转移到以问题为中心的体系上来,加强基于算法的逻辑思维训练,通过一些经典的例子,介绍如何整理思路、构造解题算法,提高学习兴趣和解决实际问题的能力。

以问题为中心,加强基于算法的逻辑思维训练,不是不介绍语法,而是本着语法够用就行的思想,把语法和程序测试穿插于逻辑思维训练中。“皮之不存,毛将焉附”,不介绍语法,就不可能写出任何程序,进行逻辑思维训练成了一句空话。但是,语法够用就行,即只要能写出程序就行。

### (2) 面向对象提前。

在经济学中有一个路径依赖(path dependence)理论:一旦人们做了某种选择,就好比走上了一条不归之路,惯性的力量会使这一选择不断自我强化,并让你不能轻易走出去。中国人将之称为“先入为主”。美国经济学家道格拉斯·诺思用这个理论成功地阐释了经济制度的演进规律,从而获得了 1993 年的诺贝尔经济学奖。在教学中,先入为主常常会使得理论

上认为简单、自然的方法更不易被接受,因为它要人们付出一定的转移成本。

现在的 C++ 程序设计教材,尽管许多教材名上冠以“面向对象”,但却几乎都是从面向过程入手,讲词法、讲语法,然后转向面向对象。先入为主的训练,使得学习者无法理解书本中标榜的“面向对象是一种很自然的方法”。针对这种弊病,本书一开始就进入了面向对象的世界,对读者进行定义类—生成对象—操作对象面向对象的三步解题训练,并将面向过程作为面向对象的实现环节,将选择、迭代和穷举三种基本算法融入其中,避免思维模式转换带来的转移成本。

### (3) 用设计模式点化面向对象。

不知道设计模式,就无法真正了解面向对象程序设计的精髓;不掌握设计模式中透射出的原则,就设计不出来优雅的面向对象程序。本书在通过前 5 个单元进行的基本算法和构建面向对象程序的基本过程训练之后,立即转入设计模式的学习。但是,设计模式对于许多人还是一条鸿沟。为了帮助读者越过这条鸿沟,本书采用了讲故事的方式,先引入从设计模式折射出来的几个面向对象程序设计的基本原则,然后对 GoF 设计模式进行概要介绍。

### (4) 将程序测试融入程序设计之中。

程序测试的目的是找出程序逻辑错误,应该说是程序设计最后的重要环节。但是,这个环节却被人们忽视了。在程序设计课程的进行中,几乎所有学习者都是编写出一个程序后,只能靠碰运气地上机试通,等到学习软件工程课程(可惜并非所有学习程序设计的人都有再学习软件工程的安排),这种陋习已经难于纠正。而实际上计算机专业在上软件工程课程时,并没有很多时间用于程序测试的训练,就连软件工程专业开设的软件测试课程,也缺少充分的实践环节。从根本上讲,程序测试应当是程序设计不可或缺的组成部分,把程序测试的训练纳入到程序设计中,应该说是最高效、最合理的安排。基于如此考虑,从 20 世纪 90 年代初,作者就开始尝试将程序测试融入程序设计的教学中。实践证明,这不仅可行,而且很有好处。

### (5) 淡化指针,分散安排。

指针是 C 和 C++ 中最富有特色的机制,它把 C/C++ 灵活、高效的特点表现得淋漓尽致。然而,指针又是程序中最容易出错、难以理解的部分。从 20 世纪 80 年代,就有人把指针与 goto 语句列入应当限制使用的“黑名单”。目前,一些新的 C 族语言,如 Java、C#(读作 c sharp)都向用户隐蔽了指针。对于初学者来说,应当养成少用指针,尽量不用指针的编程习惯。为此没有专门介绍指针的部分,而是将指针内容分散在必须使用的部分。

## 2. 学习环境与方法

由 J. Piaget、O. Kernberg、R. J. Sternberg、D. Katz、Vogotsky 等创建的建构主义(constructivism)学习理论认为,知识不是通过教师传授得到,而是学习者在一定的情境,即社会文化背景下借助其他人(包括教师和学习伙伴)的帮助,利用必要的学习资料,通过意义建构的方式而获得的。在信息时代,人们获得知识的途径发生了根本性的变化,教师不再是单一的“传道、授业、解惑”者,帮助学习者构建一个良好的学习环境成为其一项重要职责。当然,这也是现代教材的责任。本书充分考虑了这些问题。

本书中除了正文外,在每一大节后面都安排了概念辨析、代码分析、开发实践和探索深究 4 种自测和训练实践环节,建立起一个全面的学习环境。

#### 1) 概念辨析

概念辨析主要提供选择和判断两类自测题目,帮助学习者理解本节学习过的有关概念,把当前学习内容所反映的事物尽量和自己已经知道的事物相联系,并认真思考这种联系,通过“自我协商”与“相互协商”,形成新知识的同化与顺应。

#### 2) 代码分析

代码阅读是程序设计者的基本能力之一。代码分析部分的主要题型是通过阅读程序找出错误或给出程序执行结果。

#### 3) 探索思考

建构主义提倡,学习者要用探索法和发现法去建构知识的意义。学习者要在意义建构的过程中主动地搜集和分析有关的信息资料,对碰到的问题提出各种假设并努力加以验证。按照这一理论,本书还提供了一个探索思考栏目,以培养学习者获取知识的能力和不断探索的兴趣。

#### 4) 开发实践

提高程序开发能力是本书的主要目标。本书在绝大多数大节后面都给出了相应的作业题目。但是,完成这些题目并非就是简单地写出其代码,而要将它看作是一个思维+语法+方法的工程训练。因此,要求每道题的作业都要以文档的形式提交。文档的内容包括:

- (1) 问题分析与建模;
- (2) 源代码设计;
- (3) 测试用例设计;
- (4) 程序运行结果分析;
- (5) 编程心得(包括运行中出现的问题与解决方法、对于测试用例的分析、对于运行结果的分析等)。

文档的排版也要遵照统一的格式。

### 3. 感谢与期待

从 20 世纪 80 年代末,本人就开始探索程序设计课程从语法体系到问题驱动的改革;到了 20 世纪 90 年代中期又在此基础上考虑让学生在学程序的同时掌握程序测试技能;2003 年开始考虑如何改变学习了 C++ 而设计出的程序却是面向过程的状况。每个阶段的探索,都反映在自己不同时期的相关作品中。本书则是自我认识又一次深化的表达。

尽管有了近 20 年探索的积累,但笔者却越来越感觉到编写教材的责任和困难。要编写一本好的教材,不仅需要对本课程涉及内容有深刻了解,还要熟悉相关领域的知识,特别是要不断探讨贯穿其中的教学理念和教育思想。所以,越到后来,就越感到自己知识和能力的不足。可是作为一项历史性任务的研究又不愿意将之半途而废,只能硬着头皮写下去。每一次任务的完成,都得益于一些热心者的支持和帮助。在本书的写作过程中,参加了部分工作的有姚威博士、陶利民博士、张展为博士以及张秋菊、文明瑶、史林娟、李博、张有明等。在



# 目 录

## 第 1 篇 面向对象奠基

第 1 单元 对象世界及其建模	3
1.1 程序=模型+表现	3
1.1.1 程序的概念	3
1.1.2 模型	3
1.1.3 模型表现工具	5
1.2 面向对象程序设计的基本概念	8
1.2.1 对象与类	8
1.2.2 类的层次性	9
1.2.3 消息传递	10
1.3 UML 建模	10
1.3.1 用例图	11
1.3.2 序列图	11
1.3.3 状态图	11
1.3.4 类图与类间关系	12
1.3.5 对象图	13
1.3.6 类间联系的 UML 表示	13
习题 1	15
第 2 单元 学生类	17
2.1 类 Student 的声明	17
2.1.1 类静态属性的 C++ 描述	17
2.1.2 类行为的 C++ 描述	19
2.1.3 类成员的访问控制	20
2.1.4 类 Student 声明的完整形式	20
2.2 类 Student 的实现	22
2.2.1 函数定义概述	22
2.2.2 成员函数 setStud() 的定义	22
2.2.3 成员函数 dispStud() 的定义	23
2.3 类的测试与主函数	24
2.3.1 类测试的概念	24
2.3.2 对象的生成及其成员的访问	25
2.3.3 主函数	25



2.4	用构造函数初始化对象	27
2.4.1	构造函数的初始化机制	27
2.4.2	部分初始化构造函数、无参构造函数与构造函数重载	29
2.4.3	默认构造函数	30
2.4.4	析构函数	30
2.5	语法知识扩展 1	31
2.5.1	C++ 程序的组成	31
2.5.2	类与对象	31
2.5.3	C++ 单词	32
2.5.4	数据类型初步	33
2.5.5	变量与常量	35
	习题 2	36
<b>第 3 单元</b>	<b>呼叫器</b>	40
3.1	呼叫器建模与类声明	40
3.1.1	问题与建模	40
3.1.2	呼叫器类声明的 C++ 描述	41
3.2	呼叫器类的实现	41
3.2.1	用 if-else 结构定义 display()	42
3.2.2	用 switch 结构定义 display()	45
3.2.3	if-else 判断结构与 switch 判断结构比较	47
3.3	选择结构的测试	47
3.3.1	逻辑覆盖测试及其策略	47
3.3.2	测试用例的使用	51
3.3.3	呼叫器类测试	51
3.4	用静态成员变量存储类对象的共享常量——呼叫器类的改进	53
3.4.1	自动变量与静态变量: 不同生命期的变量	53
3.4.2	使用静态成员变量的呼叫器类及其测试	54
3.4.3	静态成员变量的特点	56
3.5	王婆卖瓜——静态成员变量作为类对象的共享成员的另一例	56
3.5.1	问题与模型	56
3.5.2	WangPo 类声明与实现	57
3.5.3	WangPo 类的测试	57
3.5.4	WangPo 类的改进	58
	习题 3	60
<b>第 4 单元</b>	<b>累加器</b>	64
4.1	累加器类结构设计与类声明	64
4.1.1	累加器类结构设计	64
4.1.2	累加器类声明	64
4.2	累加器类的实现	65

4.2.1	构造函数的实现	65
4.2.2	成员函数 calc() 的定义与 while 语句	66
4.2.3	使用 do-while 结构的 calc() 函数	67
4.2.4	使用 for 结构的 calc() 函数	68
4.2.5	三种循环流程控制结构的比较	68
4.3	循环结构的测试	68
4.3.1	等价分类法与边值分析法	68
4.3.2	循环结构的测试用例设计	70
4.3.3	累加器类的测试	70
4.3.4	变量(对象)的作用域问题	71
4.4	语法知识扩展 2	71
4.4.1	C++ 语句	71
4.4.2	函数	72
4.4.3	对象的存储	73
	习题 4	74
<b>第 5 单元</b>	<b>简单的公司人员体系</b>	<b>77</b>
5.1	公司人员的类层次结构	77
5.1.1	问题建模	77
5.1.2	类层次结构声明	77
5.1.3	在派生类中重定义基类成员函数	80
5.1.4	类层次结构中成员函数执行规则	82
5.1.5	基于血缘关系的访问控制——protected	85
5.2	指针与引用	85
5.2.1	指针 = 基类型 + 地址	85
5.2.2	指向对象的指针与 this	88
5.2.3	引用	89
5.3	类层次中的赋值兼容规则与里氏代换原则	91
5.3.1	类层次中的赋值兼容规则	91
5.3.2	里氏代换原则	92
5.4	虚函数与抽象类	93
5.4.1	动态绑定与虚函数	93
5.4.2	虚函数表	94
5.4.3	虚函数规则与虚析构函数	95
5.4.4	纯虚函数与抽象类	96
5.4.5	虚函数在面向对象程序设计中的意义	98
5.5	多基派生与虚拟派生	98
5.5.1	多基派生	98
5.5.2	多基派生的歧义性问题	101

5.5.3 虚拟派生	102
习题 5	103
<b>第 6 单元 面向对象程序设计的原则与设计模式</b>	111
6.1 面向对象程序设计的基本原则	111
6.1.1 从可重用说起: 合成/聚合优先原则	113
6.1.2 从可维护性说起: 开-闭原则	115
6.1.3 面向抽象原则	117
6.1.4 单一职责原则	123
6.1.5 接口分离原则	124
6.1.6 不要和陌生人说话	128
6.2 GoF 设计模式	130
6.2.1 创建型设计模式	131
6.2.2 结构型设计模式	133
6.2.3 行为型设计模式	137
习题 6	144

## 第 2 篇 C++ 进阶

<b>第 7 单元 C++ 常量</b>	149
7.1 字面常量	149
7.1.1 整型字面常量的表示和辨识	149
7.1.2 浮点类型字面常量的表示和辨识	150
7.1.3 字符常量	150
7.1.4 bool 类型常量	153
7.2 const 保护	153
7.2.1 用 const 修饰简单变量	153
7.2.2 const 修饰函数	154
7.2.3 const 修饰类成员与对象	157
7.2.4 const 修饰引用	159
7.2.5 const 修饰指针	160
7.3 枚举类型	163
7.3.1 枚举类型与枚举常量	163
7.3.2 枚举变量及其定义	164
7.3.3 对枚举变量和枚举元素的操作	164
7.3.4 用枚举为类提供整型符号常量名称	165
7.4 宏	166
7.4.1 宏定义	166
7.4.2 整型类型的极值宏	168
7.4.3 带参宏定义	169

习题 7 .....	172
<b>第 8 单元 数组——顺序地组织同类型数据</b> .....	178
8.1 数组及其应用 .....	178
8.1.1 数组基础 .....	178
8.1.2 对象数组的定义与初始化 .....	180
8.1.3 数组元素的搜索与排序 .....	182
8.1.4 基于容器的 for 循环 .....	185
8.1.5 const 数组 .....	186
8.1.6 用数组作为类的数据成员 .....	186
8.1.7 数组下标越界问题 .....	189
8.2 二维数组 .....	190
8.2.1 二维数组的定义 .....	190
8.2.2 二维数组的初始化 .....	190
8.2.3 二维数组元素的访问 .....	192
8.3 数组元素的指针形式 .....	192
8.3.1 一维数组元素的指针形式 .....	192
8.3.2 二维数组元素的指针形式 .....	194
8.3.3 多维数组元素的指针形式 .....	195
8.4 栈 .....	196
8.4.1 用数组建栈 .....	196
8.4.2 栈的应用 .....	197
8.5 字符串 .....	198
8.5.1 C 字符串 .....	198
8.5.2 string 字符串 .....	203
习题 8 .....	207
<b>第 9 单元 变量的作用域、生命期、连接性和名字空间</b> .....	212
9.1 基本概念 .....	212
9.1.1 标识符的作用域及其分类 .....	212
9.1.2 变量的生命期与内存分配 .....	215
9.1.3 标识符的连接性 .....	216
9.2 C/C++ 的存储属性关键字 .....	217
9.2.1 用 auto 或 register 定义自动变量 .....	217
9.2.2 用 extern 定义或修饰全局变量 .....	217
9.2.3 static 关键词 .....	220
9.3 名字空间域 .....	227
9.3.1 名字冲突与名字空间 .....	227
9.3.2 名字空间的使用 .....	229
9.3.3 无名名字空间和全局名字空间 .....	231

习题 9 .....	232
<b>第 10 单元 C++ 异常处理 .....</b>	<b>236</b>
10.1 程序异常及其应对 .....	236
10.1.1 程序异常的概念 .....	236
10.1.2 程序异常的一般应对 .....	236
10.1.3 C++ 异常处理机制 .....	239
10.2 C++ 异常类型 .....	241
10.2.1 简单异常类型 .....	241
10.2.2 用类作为异常类 .....	243
10.2.3 C++ 标准异常类 .....	245
10.3 常用异常处理技术 .....	247
10.3.1 捕获任何异常 .....	247
10.3.2 重新抛出异常 .....	248
10.3.3 抛出多个异常 .....	251
习题 10 .....	253
<b>第 11 单元 动态内存分配与链表 .....</b>	<b>257</b>
11.1 C++ 动态存储分配方法 .....	257
11.1.1 存储空间的编译器分配和程序员分配 .....	257
11.1.2 用 new 为单个数据动态分配内存 .....	257
11.1.3 用 delete 回收单个数据的动态存储空间 .....	258
11.1.4 数组的动态存储分配 .....	260
11.1.5 对象的动态存储分配 .....	261
11.1.6 数据成员的动态存储分配 .....	261
11.1.7 对象的浅复制与深复制 .....	263
11.2 C++ 动态存储分配中的异常处理 .....	264
11.2.1 捕获 std::bad_alloc 异常 .....	264
11.2.2 避免使用 std::bad_alloc .....	265
11.3 链表 .....	265
11.3.1 链表及其特点 .....	265
11.3.2 单向链表类设计 .....	267
11.3.3 单链表的操作与成员函数设计 .....	269
11.3.4 链表的测试 .....	273
习题 11 .....	276
<b>第 3 篇 C++ 探幽</b>	
<b>第 12 单元 C++ I/O 流 .....</b>	<b>281</b>
12.1 C++ 流与流类 .....	281
12.1.1 流与缓冲区 .....	281

12.1.2	C++ 流类库	282
12.1.3	ios 类声明	284
12.2	标准流对象与标准 I/O 流操作	285
12.2.1	C++ 标准流对象	285
12.2.2	标准输入输出流操作	285
12.3	流的格式化	286
12.3.1	ios 类的格式化成员函数和格式化标志	286
12.3.2	格式化操作符	287
12.4	文件流	288
12.4.1	文件流的概念及其分类	288
12.4.2	文件操作过程	288
12.5	流的错误状态及其处理	292
12.5.1	流的出错状态	292
12.5.2	测试与设置出错状态位的 ios 类成员函数	292
习题 12		293
<b>第 13 单元</b>	<b>C++ 函数细节</b>	<b>294</b>
13.1	函数的参数	294
13.1.1	值传递：变量/对象参数	294
13.1.2	名字传递：引用参数	296
13.1.3	地址传递：地址/指针参数	299
13.1.4	函数调用时的参数匹配规则	302
13.1.5	关于函数实参的计算顺序	303
13.1.6	形参带有默认值的函数	303
13.1.7	参数数目可变的函数	305
13.2	函数返回	306
13.2.1	函数返回的基本特点	306
13.2.2	返回指针值的函数	306
13.2.3	返回引用的函数	307
13.3	函数名重载	310
13.3.1	函数名重载的基本方式	310
13.3.2	编译器对于函数名重载的匹配规则	310
13.3.3	函数名重载的误区	311
13.3.4	函数名重载中的二义性	311
13.3.5	函数名重载与在派生类中重定义基类成员函数	312
13.4	操作符重载	312
13.4.1	操作符重载及其规则	313
13.4.2	成员函数形式的操作符重载	314

13.4.3	友元函数形式的操作符重载	318
13.4.4	赋值操作符重载	322
习题 13		323
<b>第 14 单元</b>	<b>类型转换与运行时类型鉴别</b>	<b>331</b>
14.1	数据类型转换	331
14.1.1	算术类型的隐式转换规则与校验表	331
14.1.2	显式类型转换	333
14.1.3	对象的向上转换和向下转换	334
14.1.4	转换构造函数	334
14.2	dynamic_cast 操作符	338
14.2.1	dynamic_cast 及其格式	338
14.2.2	上行强制类型转换与下行强制类型转换	338
14.2.3	交叉强制类型转换	342
14.2.4	类指针到 void * 的强制转换	342
14.2.5	dynamic_cast 应用实例	342
14.3	用 typeid 获得对象的类型信息	344
14.3.1	typeid 类	344
14.3.2	typeid 操作符的应用	345
14.3.3	关于 typeid 的进一步说明	346
习题 14		347
<b>第 15 单元</b>	<b>模板</b>	<b>349</b>
15.1	算法抽象模板——函数模板	349
15.1.1	从函数重载到函数模板	349
15.1.2	模板函数重载	350
15.1.3	函数模板的实例化与具体化	352
15.2	数据抽象模板——类模板	355
15.2.1	类模板的定义	355
15.2.2	类模板的实例化与具体化	356
15.2.3	类模板的使用	357
15.2.4	类模板实例化时的异常处理	359
15.3	标准模板库	360
15.3.1	容器	360
15.3.2	算法与函数对象	362
15.3.3	迭代器	367
15.3.4	STL 标准头文件	370
习题 15		371

附录 A C++ 保留字 .....	377
A.1 C++ 关键字 .....	377
A.2 替代标记 .....	377
A.3 C++ 库保留名称 .....	378
A.4 特定字 .....	378
附录 B C++ 运算符的优先级别和结合方向 .....	379
参考文献 .....	381



# 第 1 篇 面向对象奠基

程序设计是一个逻辑思维传达过程,即把人的求解问题的逻辑思维传达到计算机操作的过程。面向对象程序设计作为现代程序设计的主流,其核心是将逻辑思维向前追溯到观察问题的视角,向后延伸到程序的组织结构上。也就是说,面向对象作为一种方法,更作为一种思维模式,要求人们以面向对象的观点去分析问题,还要求人们用面向对象的观念去组织程序。只有做到了这两点,才算奠定了面向对象的基础。

为了帮助初学者奠定这个基础,本篇首先介绍面向对象的一些基本概念;接着用 4 个单元通过实例进行用面向对象的视角观察问题的训练;最后用一个单元介绍按照面向对象的观念组织程序的一些重要原则,将读者引入真正的面向对象世界,为进一步学习奠定良好的基础。