面向21世纪课程教材
Textbook Series for 21st Century

C++程序设计实验教程

李师贤 主编

高等教育出版社

面向21世纪课程教材

Textbook Series for 21st Century

C++程序设计实验教程

李师贤 主编

李师贤 蒋爱军 林 瑛 李宏新 编

高等教育出版社

图书在版编目（CIP）数据

C++程序设计实验教程 / 李师贤主编. —北京：高等教育出版社，2006.7
ISBN 7-04-019646-8

Ⅰ.C... Ⅱ.李... Ⅲ.C语言—程序设计—高等学校—教材 Ⅳ.TP312

中国版本图书馆CIP数据核字（2006）第069599号

策划编辑 倪文慧 **责任编辑** 倪文慧 **封面设计** 于文燕 **责任印制** 朱学忠

出版发行	高等教育出版社	**购书热线**	010-58581118
社　　址	北京市西城区德外大街4号	**免费咨询**	800-810-0598
邮政编码	100011	**网　　址**	http://www.hep.edu.cn
总　　机	010-58581000		http://www.hep.com.cn
		网上订购	http://www.landraco.com
经　　销	蓝色畅想图书发行有限公司		http://www.landraco.com.cn
印　　刷	北京新丰印刷厂	**畅想教育**	http://www.widedu.com
开　　本	787×1092　1/16	**版　　次**	2006年7月第1版
印　　张	20.5	**印　　次**	2006年7月第1次印刷
字　　数	410 000	**定　　价**	25.70元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

物料号　19646-00

内 容 提 要

本书是“面向21世纪课程教材”《面向对象程序设计基础》(高等教育出版社出版)一书的配套实验教材。全书通过实验方式介绍C++程序设计,内容共15章:第1~3章涉及程序设计的目标、技术与特点,包括程序设计的目标与准则、程序设计必须遵循的一般性技术原理、程序设计风格;第4~5章涉及程序中的基本控制结构及结构化程序设计的思想与措施;第6~9章涉及程序中的数据结构;第10~12章涉及控制与数据的相互作用;第13章涉及程序测试;第14章结合*Booch*方法介绍面向对象程序设计;第15章为综合实验。

每一章包含若干具有针对性的实验,每个实验分为目的要求、原理、实验内容、思考要点四个部分,突出程序设计的思想方法和技术,强调程序设计语言(C++)的知识要点,提示易犯的错误,引导读者深入思考。书后的四个附录分别介绍了上机编程过程、实验报告的书写、集成开发环境Turbo C++ 3.0和Visual C++ 6.0。

本书内容丰富、概念清晰、实用性强,既可与面向对象程序设计课程结合使用,也可用作独立的实验课程教材,还可供软件开发人员参考。

前　言

在计算机科学与技术领域中，到处呈现一派“无边落木萧萧下，不尽长江滚滚来”的新陈代谢景象。要想应对如此快速发展的现实，就要通过理论联系实际的教学，提高学生的科学素养，培养学生的创新意识、创新能力和团队协作精神，发挥学生自身的潜能，提高学生的专业素质技能。

程序设计是一门实践性很强的基础课程，它有助于培养和发展解决问题和创造性思维的能力，也可以培养运用算法来解决实际问题的能力，这种解决问题的方式是计算机学科所独有的，也只有通过计算机程序设计语言和程序设计方法的学习与实践，才有可能获得这种解决问题的能力。因此，N. Wirth 曾说过，程序对于实干的人来说是个遍布黄金之地。

Christopher Strachey 曾这样说过：“……我一直在做程序设计语言方面的工作，因为依我看来，如果一个人不能理解程序设计语言，他就不能真正地理解计算机。理解程序设计语言不仅仅意味着会使用它们，许多人能使用它们但是并不真正理解”。这也是我们组织本教程材料的依据之一。

程序设计实验教学是用实验的方法去学习与研究程序设计方法与技术，以及理解程序设计语言的各种成分与机制。程序设计实验教学的一个显著特点是它的实践性。这里所指的实践性有三层意思：一是动手能力的培养和锻炼，单凭读书是学不会程序设计的；二是思维和判断能力的培养和锻炼；三是良好编程习惯的培养和锻炼。我们必须实际动手编写程序代码，使用与操作各种软件工具，在计算机上调试与运行程序，分析和纠正所出现的各种错误，并对所获得的结果做出分析与判断。本教程特别强调在实验中对问题做探索性研究，以促进将科学研究渗入到本科教学中去。要知道，如果不去仔细琢磨程序设计过程中的各个步骤，不去仔细比较衡量程序代码中各个构成元素，不去做探索性研究，就不可能了解、体会到程序设计的深奥与微妙之处，也就不可能领悟与掌握程序设计的真谛。

程序设计实验课可以在学完第一门高级程序设计语言（C++）理论课开设，也可与初学高级语言（C++）程序设计课同步开设。本教程既可以作为单独设立的实验课教材，也可以作为与理论课相结合使用的教材，但我们更主张前者。

我们尽力整合（例如，数据结构、良好的程序设计方法等内容）和优化了实验内容，压缩了部分（例如，关于语言微观方面的表达式、运算符等以及程序的顺序结构、程序框图之类的内容）验证性实验，提高了综合性、设计型和研究创新性实验的比重。我们希望读者通过本

教程的学习，能从程序设计技术的角度详细地考察（高级）程序设计语言（C++）的机制，并对程序设计的思想方法和技术有较为全面、正确的理解。本实验教程在突出程序设计的思想方法和技术以及程序设计语言知识要点的同时，强调良好的编程习惯，警示易犯的错误。

本实验教程共15章和4个附录。

第1、2、3章较为抽象地讨论程序设计的目标、技术与特点。第1章强调程序设计的目标与准则，如正确性、可靠性、可理解性、有效性、可维护性和可重用性等；第2章介绍为支持第1章所主张目标与准则所必须遵循的一般性技术原理、手段与措施；第3章介绍能用以支持问题解或者说为了显式地呈现前二章内容所希望的程序设计风格。之所以这样安排，是因为众多程序设计人员都是从痛苦的经验中得到顿悟："最好在起步的时候就正确地编写程序"（Seegmuller. G）。但对于初学者而言，若刚开始就要深入阅读理解这些较为抽象的材料，并上机实验，确实会有一定的难度。尽管如此，我们还是建议先大致了解一下这3章的内容。随着实验教学的不断深入，能及时多次地回过头来再进行这3章中有关要求的实验。

第4、5章涉及程序中的基本控制机制，包括有关选择结构、循环结构以及结构化程序设计思想与技术措施的实验。

第6、7、8、9章涉及程序中的数据结构。第6章介绍程序设计中的基本数据及其应用，了解C++语言中不同数制的表示；第7章讨论抽象数据类型与类的定义和应用，数组的抽象数据类型；第8章研究栈、队列的顺序和链式表示方法及其应用；第9章是关于树和图的表示方法与应用。数据信息的表示是计算机科学的基础，大多数计算机程序的主要目标与其说是完成运算，不如说是存储信息和尽快地寻找指定的信息。因此，研究数据结构和算法就成了计算机科学以及程序设计的核心问题。这4章的目的就是要帮助读者理解如何组织信息数据，了解程序设计所使用的基本数据结构，并训练读者设计支持高效数据处理的程序的能力。

第10、11、12章涉及控制与数据的相互作用。第10章的实验内容是使用C++函数进行模块化程序设计，学会如何处理模块/函数间的数据交换问题，以及圆满地解决客户方函数与服务器方函数的职责分工问题；第11章是关于动态数据类型和动态数据结构的实验，不仅可以提高内存的使用效率，还可以提高程序的适应能力。第12章讨论封装，包括两方面的内容：用类来实现封装（及信息隐藏），通过封装与继承和多态的结合使用来应对需求的变更。

第13章主要介绍几种简单的测试实验方法，使初涉程序设计领域的人对程序测试过程有一个较全面的认识，并理解程序测试在程序设计过程中的重要作用。

第14章通过几个简单的实验，展示如何应用Booch面向对象方法进行程序设计与开发。

第15章为综合实验，共包括4个实验，希望读者能进一步认识到在计算机上求解问题时必须考虑的3个前提，即必须将问题形式化；必须有一个算法；算法必须有合理的复杂度。

本实验教程所给出的完整源代码均在 Microsoft Visual C++6.0 中编译通过。为了在与 C++ 新标准相符的同时兼顾旧版本的使用风格，部分代码中使用了 C++ 标准库中定义的名字空间 std 及新的头文件名，部分代码仍使用旧的头文件名。

本书由李师贤、蒋爱军、林瑛和李宏新编写，周晓聪老师详细地审阅了全部书稿，并提出了很有价值的修改意见，在此谨向他致以衷心的感谢！

梅晓勇、蔡树彬、孙恒、古思山、李思恒、曹响等老师和同学在本书的编写过程中给予了许多具体的帮助，在此特向他们表示真诚的谢意。

本教程得到了中山大学实验教学研究改革项目基金的资助和中山大学软件学院的资助与支持，特此表示衷心的感谢！

受作者水平所限，书中难免存在缺点错误，恳请广大读者不吝批评指正。

作者

2006 年 4 月 30 日于广州

郑重声明

目　录

第1章　程序设计的基本目标与准则

程序是对计算机所处理的任务的描述。本教程中用实现级语言(C++)编写的代码即为程序。程序设计是指设计、编写和调试程序的方法与过程。程序是计算机工作的依据,计算机的工作性能通过程序的质量来体现。因此,程序设计的工作在信息化领域中有着非常重要的地位。

程序设计是一种目标明确的工程活动,其智力性非常强。因此很难定量地制定程序的质量标准。常用的定性标准包括:

- 正确性(Correctness)
- 健壮性(Robustness)
- 可靠性(Reliability)
- 可理解性(Understandability)
- 有效性(Efficiency)
- 可维护性(Maintainability)
- 可重用性(Reusability)

1.1　实验:程序正确性

【目的要求】

1. 掌握程序正确性的概念。
2. 学习程序规范及其应用。

【原理】

事实上,正确性不只是程序质量的标准,也是我们处理任何事情的准则。现在的问题是:何谓程序正确性?如何保证其正确性?

所谓程序正确性是指一个程序是否正确地实现了预定的目标。可见,这个“预定目标”将作为我们衡量程序正确与否的标准。显然,要弄清这个预定目标的第一步就是必须明确“做什么”。所谓“做什么”是指对任务的描述。关于“做什么”的描述通常称为程序的规格说明,形式化的程序规格说明称为程序规范。通常,一个程序规范由两部分组成,一部分是要完成这个任务必须满足的前提,另一部分是完成此项任务的结果必须具备的性质。前者称为初始断言(条件)或前置断言(条件),用谓词 φ 表示;后者称为结果断言(条件)或后置

断言(条件),用谓词 ψ 表示。因此一个程序规范可以记为(φ,ψ)。

例 1-1-1 取整数 x 的绝对值的程序规范为:

$$\varphi: \text{true}$$
$$\psi: y \geqslant 0 \wedge (y = x \vee y = -x)$$

其中,y 为结果变量。

例 1-1-2 找出包含 100 个元素的整数数组 a 中的最大元素值,并把结果放在变量 max 中,其程序规范为:

φ: true

ψ: $(\exists j \in \{0,1,\cdots,99\})(\max = a[j]) \wedge (\forall k \in \{0,1,\cdots,99\})(\max \geqslant a[k])$

例 1-1-3 找出整数 a、b、c 中的最大值,并把结果放在变量 max 中,其程序规范为:

φ: true

ψ: $(\max = a \vee \max = b \vee \max = c) \wedge \max \geqslant a \wedge \max \geqslant b \wedge \max \geqslant c$

例 1-1-4 求两个正数 x 和 y 的商和余数的程序规范为:

$$\varphi: x \geqslant 0 \wedge y > 0$$
$$\psi: q = x \text{ div } y \wedge r = x \bmod y$$

其中,q 为商,r 为余数。

程序断言是指对程序性质的陈述。常见的一种程序断言形式如下:

$$\{\varphi\}S\{\psi\}$$

其中,S 是一个程序(语句),φ 是程序(语句)的前置断言,ψ 是程序(语句)的后置断言。

断言 $\{\varphi\}S\{\psi\}$ 称为 S 关于程序规范(φ,ψ)的正确性断言。其含义为:

"若 S 开始执行时 φ 为真,则 S 的执行必终止且终止于 ψ 为真"。

因此,若断言 $\{\varphi\}S\{\psi\}$ 为真,则意味着 S 是满足规范(φ,ψ)的一个程序。或者说,程序 S 是规范(φ,ψ)的一个实现。从这一点来看,程序设计的任务便是:对给定的规范(φ,ψ),构造一个程序 S,使得断言 $\{\varphi\}S\{\psi\}$ 为真。

此外,从上述描述也不难看出,所谓的程序正确性证明,就是证明在程序的任何一种可能的处理过程中所设置的断言都是成立的。

对这一问题,早在 1947 年美籍匈牙利科学家 J. von Neumann 发表的论文中就提到过程序正确性证明。1967 年美国科学家 R. W. Floyd 系统地提出了验证程序正确性证明的归纳断言方法。1969 年英国科学家 C. A. R. Hoare 将归纳断言方法形式化,提出了程序验证公理系统。虽然众多科学家对这一问题进行了大量的研究,但至今尚未找到一种切实可行又实用的途径。

【实验内容】

1. 试给出求两个整数中较大值的程序规范。
2. 试给出寻找含 100 个元素的数组 b 中第一个最大值元素位置的程序规范。

3. 试给出寻找第 n 个 Fibonacci 数 f_n 的程序规范。Fibonacci 数列的定义是：$f_0 = 0$，$f_1 = 1$，对于 $n > 1$，$f_n = f_{n-1} + f_{n-2}$。

4. 试给出求两个各含 50 个元素的数组 a 与 b 内积的程序规范。

【思考要点】

1. 从前面的讨论中不难看出，一个程序规范的两个谓词分别确定了程序或语句 S 执行前后状态所必需满足的条件。也可以说，它们刻画了程序输入与输出的关系，或者说刻画了程序语言结构的语义特征。由此，可以考虑一种基于后置条件的向后推导的程序构造技术。有兴趣的读者可以尝试，关于这方面的参考资料有：

- Dijkstra E W. A discipline of programming. Prentice-Hall, 1976.
- Grices D. The science of programming. Spring-Verlag, 1981.
- 陈火旺等. 程序设计方法学基础. 长沙：湖南科学技术出版社，1987.

2. 证明程序的正确性，就是要证明程序是否能达到预定目标。无疑，这个预定目标便是这里所讨论的程序规范或者说程序规格说明。关于程序正确性的形式化证明的讨论请参见实验 2.4。

1.2 实验：程序健壮性

【目的要求】

1. 理解程序健壮性的概念。
2. 初步掌握提高程序健壮性的设计方法。

【原理】

程序的正确性只能保证程序在正常输入下能正常工作，程序在异常输入下能否正常工作则取决于程序的健壮性。IEEE 软件工程标准术语词汇表把健壮性定义为“系统或者组件在接受不合法的输入或在异常环境下正常运转的程度”。这里的正常运转，可理解为程序实体不会被挂起、崩溃或者破坏系统。

程序健壮性是衡量程序在异常输入和应力环境条件下保持正常工作能力的一种度量。健壮性与正确性的区别是，正确性用来描述程序在正常外界环境下的行为特征，而健壮性则用来描述程序在异常外界环境下的行为特征。程序健壮性体现了程序的容错能力和故障恢复能力。引起程序出错的原因既可以来自程序内部，也可以来自程序外部。一个健壮的程序应该既可以容忍输入的错误，也可以容忍内部构件的故障。

随着计算机技术的发展，程序规模越来越大，程序所处的环境也越来越复杂，对程序健壮性的要求也越来越重要。

程序健壮性取决于程序本身的设计。程序设计中可通过一些方法提高程序的健壮性，

如检查输入数据的合法性、模块(包括函数)相互调用时检查参数的合法性、利用信息隐藏把模块内的实现细节与外界隔离、降低模块间的耦合度等。

【实验内容】

1. 阅读理解表1-2-1和表1-2-2所示程序,给出其程序规范,并指出哪一个程序的健壮性较高,为什么? 可以上机进行试验。

表1-2-1 程序

```
#include <iostream.h>

void main()
{
    int x;
    int y;
    int result;
    cout <<"Enter two number \n";
    cin >> x >> y;
    result = x /y;
    cout <<"The quotient of the two number: " <<result <<"\n";
}
```

表1-2-2 程序

```
#include <iostream.h>

void main()
{
    int x;
    int y;
    int result;
    cout <<"Enter two number \n";
    cin >> x >> y;
    while (y == 0) {
        cout <<"The divisor cannot be 0 \n";
        cout <<"Enter two number \n";
        cin >> x >> y;
    }
    result = x /y;
    cout <<"The quotient of the two number: " <<result <<"\n";
}
```

2. 就程序健壮性而言，表 1-2-3 所示程序是否有值得改进的地方？如果有，应该如何改进？

表 1-2-3 程序

```
//功能:给定正整数 n,求其平方值不超过 n 的最大正整数

#include <iostream.h>

void main()
{
    int n;              //用户给定的正整数
    int max;            //所求的最大整数
    //用户给定一个自然数
    cout << "Enter a number: ";
    cin >> n;
    //利用循环求出平方值大于 n 的最小整数
    max = 1;
    while (max * max <= n) max = max + 1;
    //输出结果
    cout << " The maximum integer is: " <<max - 1 << "\n";
}
```

【思考要点】

1. 如何理解程序正确性和健壮性之间的关系？
2. 提高程序健壮性的设计方法有哪些？

1.3 实验：程序可靠性

【目的要求】

1. 理解程序可靠性的概念。
2. 了解程序设计可见性技术。
3. 初步了解 UML 建模方法。

【原理】

21 世纪初，某银行曾发生营业员错把客户身份证号当作存款金额录入，导致数额巨大的储蓄存款错误事件。羊城晚报曾报道，广西某地用一张银行取款卡在 ATM 上取款，越取账户上余款越多。总的来看，目前我们通常使用的程序，大多数存在这样那样的错误或问题。

程序可靠性是指在给定的时段内，在一个给定的环境下，一个给定的程序无故障运行的

概率。一般来说,一个程序的故障越少,其可靠性越高。程序可靠性的重要性与日俱增,因为它可以定量地衡量程序的失效性,已被公认为是影响系统可依赖性的关键因素。从程序可靠性的定义中可以看到它的3个要素:失效(指未达到所期望的目标)、时间、运行环境。从现象上看,谈论程序可靠性度量中涉及的仅是评价与测试。但是,决定一个程序可靠性的主要因素是程序设计过程。可靠性本身就应当是一个设计目标,但它不是通过对设计进行静态分析就能评价的设计特性。相反,程序可靠性依赖于设计中所显示的其他特征,如正确性、可理解性、更小的复杂性以及更强的易修改性等。就程序的可靠性而言,最重要的特征是设计可见性和设计单元间的耦合度。

设计可见性是对设计的实现反映程序规格说明的程度所进行的评估。这就意味着,对规格说明的任何部分,都可以标识出实现它的地方。这些实现必须以这样一种形式来表示,即易于理解,其结构要反映出规格说明的结构。至今,还没有出现有关程序的设计可见性的任何度量技术。

使用一种一致的方法来实现规格说明是取得设计可见性的最好办法。最适宜的是采用图形设计表示法,结合形式化和非形式化的符号进行文字和表格的描述。图形符号特别重要,它将纯文字符号表示的一维结构扩展为视野大为开阔的二维结构,更易于人们理解。这正是今天UML建模表示法大行其道的原由。

设计的耦合度是对设计单元的依赖性评估。这里的设计单元可以视为程序模块、程序段、甚至是程序语句。低耦合度就意味着,这种设计由互相独立、几乎没有涉及共享数据的结构构成。

【实验内容】

1. 图1-3-1是简单的UML活动图,其对应程序代码如表1-3-1所示,试根据该活动图解释程序的功能,并仔细体会图形表示和程序代码表示各自的优缺点。

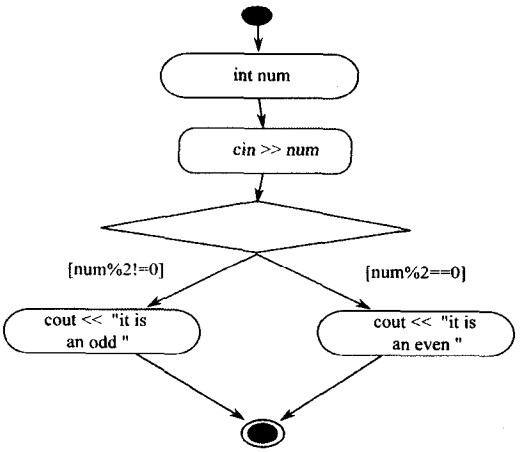

图1-3-1 UML活动图

表 1-3-1 程序

```
#include <iostream.h>

void main()
{
    int num;
    cin >> num;
    if (num % 2 != 0) cout << "It is an odd \n";
    else cout << "It is an even \n";
}
```

2. 解释表 1-3-2 所示程序的功能，并参考上题 UML 活动图的表示方式，画出表 1-3-2所示程序对应的 UML 活动图。

表 1-3-2 程序

```
#include <iostream.h>

void main()
{
    float i;
    float j;
    float k;
    float max;

    cout << "Enter 3 float number please: \n";
    cin >> i >> j >> k;
    max = i;
    if (j > max) max = j;
    if (k > max) max = k;

    cout << "Maximum is " << max << "\n";
}
```

3. 考虑程序的健壮性，分别修改表 1-3-1 和表 1-3-2 所示程序，并修改相应的 UML 活动图。

【思考要点】

1. UML 图形表示法在程序设计中能起到什么作用？对程序的可靠性又有什么影响？

2. 设计单元间的耦合度会如何影响程序的可靠性？在程序设计中有什么方法可用来降低设计单元间的耦合度？