C++语言培训教材

（上册）基础篇

● 刘峻桐　刘　豫　著

URL:http://www.phei.co.cn

PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
電子工業出版社

C++语言培训教材

（上册） 基础篇

刘竣桐 刘 豫 编著
高永泉 贾宝金 审校

電子工業出版社

内容简介

全书分上、中、下三册，上册为基础篇，讲述了C++语言的特点及优越性和C++程序设计的风格，C++的操作符、语句、函数、数组、变量作用域、模块化程序设计及指针等。读者通过上册的学习，基本上可以掌握C++语言的基础。

中册为提高篇，进一步对指针和数组进行深入的研究，进而提出类、对象以及继承性，并围绕类和对象进一步阐述了C++语言面向对象的特点。随后又对构造函数、析构函数、虚函数以及函数重载等作了详细的讲述，培养读者用类解决实际问题能力，掌握面向对象的程序设计方法。

下册为应用篇，重点讲述C++语言的输入输出(I/O)及介绍4个应用例程，通过解剖分析述4个C++实用程序，培养读者运用所学的C++语言编写程序时解决实际问题的能力。

本书适用于初中级教材，也可作为有关的训练和自学教材。

C++语言培训教材
(上册) 基础篇
刘峻明　刘　俊　编著
高水泉　曹千金　审校
责任编辑　毛北金
特约编辑　张成全

电子工业出版社出版
北京市海淀区万寿路173信箱 (100036)
电子工业出版社发行　各地新华书店经销
北京科普印刷厂印刷

开本 787×1092毫米 1/16 印张 10.25 字数 240 千字
1996年12月第一版　1996年12月第一次印刷
印数 5000册 定价 12.50元
ISBN 7-5053-3717-0/TP·1567

前　言

C++代表着程序设计语言领域的一个重要进步。目前，学习C++程序设计语言的人越来越多，许多以前用C语言编程的人员现在开始转向用C++。C++代表了C语言的发展趋势。C++采用了许多重要技术，这使它替代C语言只是一个时间问题了。

C++是C语言的增强型版本，在C语言的基础上，C++做了功能扩充以支持面向对象的程序设计。这些扩充极大地增强了C语言的能力。C++语言的强大功能及其通用性，决定了它必将成为继C之后的主要程序设计语言之一。

事实上，C语言在应用上所形成的广泛基础已经注定了C++语言在未来程序设计语言中的主导地位。从本质上讲，C++与C有着不同的编程风格。学习C++语言，不仅要学习语言要素本身，而更重要的是学会如何用C++的思维方式进行程序设计，为了使读者能够尽早地养成用C++编程的思维方式来考虑问题的习惯，本书力求一开始即给读者展现C++的编程风格，而不是传统的C语言风格。

在语言学习中，单调的语法讲解不仅无助于有经验的程序设计人员提高编程技巧和能力，而且往往会使初学者感到迷惑。因此，本书在讲解C++语言要素的同时，结合大量精短的程序，以帮助读者理解和掌握C++语言。在阅读本书时读者会发现，用这种方法讲解语言，不仅可以使你非常容易地掌握C++语言要素，与此同时还能学到许多编程技巧。

本书分上、中、下三册。上册为基础篇，基础篇共十七章，首先讲述了由C语言演变为C++语言的发展过程，C++语言的特点以及与C语言的主要区别，阐述了C++语言的优越性。进而讲述了C++程序设计的风格及C++下的操作符、语句、函数、数组、变量作用域、模块化程序设计及指针等。因此可以说上册是学习C++语言的基础。读者通过上册的学习，基本上可以掌握C++语言的基础。

中册为提高篇，共九章（第十八章至第二十六章），是在上册的基础上进一步对指针和数组进行了深入的研究讨论，进而提出类、对象以及其继承性，并围绕类和对象进一步阐述了C++语言面向对象的特点。随后又对构造函数、析构函数、虚函数以及函数重载等作了详细的讲述，以培养读者用类模拟解决实际问题的能力，进而掌握面向对象的程序设计方法。

下册为应用篇，由本书的第二十七章至第三十一章以及两个附录组成。重点讲述C++语言的输入输出(I/O)及介绍4个应用例程，并通过举例解剖分析4个C++实用程序，来培养读者运用所学的C++语言解决实际编程问题的能力。

本书在编写过程中，高水泉、贾宝金、王世兰、李桂萍等给予了大力支持，并做了大量工作，在此表示衷心的感谢。但因作者水平有限，虽然从事编程多年，亦恐书中仍有错误不妥之处，欢迎读者批评指正。

目　录

第一章　C++概述

C++是众多程序设计语言中的一个新成员。尽管这种语言八十年代才开始提出，但在近几年中越来越普及。提供C++编译器的公司越来越多，C++编译器设计人员正在提高它的效率，进行语言标准化。这样，C++语言很快将会适用于世界上绝大多数计算机。目前，在PC机界，两个PC机软件巨头——Borland和Microsoft公司都在不断地推出各自最新版本的C++编译器。

本章主要介绍以下内容：

- 从C到C++
- 什么是面向对象的程序设计
- 类——C++用来模拟现实世界的方法
- 封装
- 继承
- 多态性
- C++和C的另外一些区别
- C++程序的一般结构

本章旨在综合阐述C++所具有的特点，对于没有学过C语言的初学者来说，并不一定要马上弄懂本章中讲的所有概念。等阅读完本书全部内容之后，再回过头来看本章的内容，将会进一步加深你对C++语言的理解和认识。

1.1　从C到C++

C++是C语言的增强版本。在学习C++语言之前，了解一下C++程序设计语言的演变过程是非常有益的。C++首先是由工作在美国新泽西州玛瑞辉尔的贝尔试验室的Bjourn Stroustrup先生于1980年提出的，他在原有C语言的基础上增加了一些特性以弥补C存在的一些缺陷，并通过增加面向对象特征改变了程序员思考问题的方法。最初他把这种新的语言叫做"含类的C"，到1983年才改名为C++。

尽管C语言是世界上最受欢迎和应用最广的专业程序设计语言之一，但C++的发明仍然是必需的，这主要是由于现代程序的大规模性和复杂性所决定的。用C设计出的程序，当程序代码达到2,500至100,000行时，它就会变得非常复杂，全面掌握就很困难。而C++正是为了解决这个问题而设计的。C++的本质就是能够让程序员更加容易地理解和管理更大、更复杂的程序。

Stroustrup先生对C作了许多补充以支持面向对象的程序设计(OOP)，Stroustrup意识到，C++程序员需要一种具有灵活性和真正面向对象(OOP)的程序设计语言提供模块化的程序设计方法。关于什么是面向对象的程序设计，我们将在下面进行详细的论述。

Stroustrup 先生称 C++ 的某些面向对象的特点受到了另一种称为 Simula67 的面向对象语言的启发。所以，C++ 代表着两种具有强大功能的程序设计语言的组合。

在 C++ 的发明中，Stroustrup 先生保留了 C 的所有的精髓，如高效率、灵活性，同时增加了对面向对象程序设计的支持。C++ 给程序员提供了对 C 的自由控制以及管理对象的能力。C++ 的面向对象的特点，使程序的结构更加清晰、易于扩展、易于维护且不失其效率。

尽管C++ 最初的设计本意是帮助管理大型程序，但用途不仅限于此，事实上 C++ 的许多特点使其当之无愧地成为最优秀的程序设计语言之一。它可被有效地用于各种实际的编程工作。C++ 常常被用于设计编辑器、数据库系统及通信程序等等。由于 C++ 共享 C 的所有优点，所以用 C++ 可以构成许多高性能的系统软件。

1.2 什么是面向对象的程序设计

所谓"面向对象的程序设计"一般是指在进行程序设计时，把问题分为由相关部分组成的组，每一部分考虑和分组相关的代码和数据，同时把这些分组按层次关系组织起来，最后把这些分组转换成为叫做对象的独立的单元。

具体地讲，面向对象的程序设计是一种试图模仿我们建立现实世界模型方法的程序设计方法。为应付现实生活的复杂性，人们已逐渐形成了很好地概括、分类和抽象的能力，在我们人类词汇中的几乎每一个名词都表示一类对象，每类对象都具有一组属性或行为特征。如从一个千差万别充满个性的狗的世界中，人们抽象出称为狗的抽象类，这就允许我们发展和总结有关狗的概念而无须将注意力分散到有关任何一个特殊的狗的细节中去。在 C++ 中，面向对象的程序设计的发展有利于我们对事物分类和抽象这样一种自然的倾向。

面向对象的程序设计方法，与传统的结构化设计有很大的不同。

传统的软件系统开发时，首先对客观世界的需求进行功能需求分析，通常我们把它称作外功能，即系统对外部用户所表现出的功能，这是系统设计所要实现的目标，然后，通过计算机语言逐步求精地定义软件每一模块。

在传统的软件工程中，通常用二种方法构造模块，即按功能模块构成和按数据流为依据构造。

长期以来，大量的软件开发实践证明，无论是软件的开发还是软件的维护过程中，系统的功能是最不稳定的因素。传统的程序设计方法的着眼点是功能，软件最终的结构是依据功能而定的，而功能实际上是最难确定、最不稳定的因素，软件开发者和用户往往在最初确定软件需求以后，根据情况和变化要求改动功能或者提出新的要求。那么，这时的软件改动要涉及的面会很宽，难度很大，并且容易顾此失彼，因此，可能导致软件质量下降，维护困难。

尽管系统中数据是相对稳定的因素，但是，按数据分析为基础的软件构造方式，过早地确定操作的时序，操作时序与软件需求是有相当密切的关系的，而且这种关系是相对某特定的要求而言的。除此之外，时序约束通常是有层次的，低层次上的时序关系要高于高层次上的时序约束。高层次上的时序约束很松散，有时甚至无约束关系。因此，软件设计中的数据流分析方式过早地确定时序约束，同样影响了软件的构造和维护的灵活性。

长期以来，软件的易维护性和可复用性一直是软件开发人员追求的目标。面向对象的

程序设计与传统方法的最大不同是，在面向对象语言的程序设计中，把功能和时序约束分别地组织到各类层次的对象中，学习本书的后续内容时读者会逐渐体会到，用这种方法构造的各模块和用这些模块组织成的软件具有很大的稳定性和可复用性，同时，在很大程度上提高了软件的易维护性。

C++最大的特点是：类、封装、抽象、继承和多态性。

1.3 类——C++用来模拟现实世界的方法

C++程序设计的核心是构造出用以模拟所要解决的实际问题的类体系。传统的程序设计方法是从事务表面着手的，它首先确定的是软件要做什么，然后是怎样做。这种设计思想是直观的。而面向对象的程序设计则是从事物的本质着手，它首先要做的是怎样抽象事物，然后再着手解决事物的联系问题。因此我们说这种方法是抽象的。

面向对象方法的核心思想是把客观事物都看成由大大小小各种对象按层次组合而成的，而每个对象都具有它自身的独立含义，这些含义用它的属性特征来描述表达。程序中主要的语言和成分是对象。

对象有两部分组成，一部分是数据结构定义，数据结构描述了一个对象的形状和状态；另一部分是操作，在一个对象里定义的操作是施加于另一个对象的数据结构上的，用来改变数据结构的状态，这种操作通常是用函数来描述的。

实际上，对象是一种变量，它的类型由声明它的类的类型决定。也许一开始我们把对象作为一种把代码和数据联系起来的变量来理解似乎有些困难，但这确实是面向对象的程序设计语言的一个重要的特征，在后面一些章节中我们将更详细地讨论这个问题。

对象是通过发送消息而被操作的。每个对象用自己合适的动作响应发来的消息。所有能够发送给指定对象的消息均可在说明它的类中指明，每个可能消息的响应在类描述中用相应的成员函数给出。

在C++中，方法是用函数原型来定义的。消息发送到对象这一个过程是用类似于函数调用的机制来实现的，例如，object是一个对象，而func是某个具有单一整型参数的方法，则消息func(6)发送到对象object可写成：

```
object func(6);
```

必须注意的是，在C++中，原则上并不用去描述和定义每一个对象，而所要描述和定义的是对象的抽象表示。这种对象的抽象表示被称作类(class)。类定义了一组具有相同行为属性和数据结构类型的对象。

在面向对象的程序设计中，类定义是静态和抽象表示的，它并不给数据结构分配具体存储空间，只有用类声明该类类型的对象时，才分配给对象具体的存储空间(即对象是实体，它对应着一定的内存单元)。因此，我们说对象是动态模块，而类则是用于描述软件的抽象模块。

面向对象语言所构造的对象模块，无论在语法上还是在语义上都是独立的单位。这种模块在客观上具有独立的含义，对它的理解以及它的创立、使用和维护都可单独进行，独立地存在。这种独立性和完整性使这种模块具有很好的结合性，即这种模块很容易与其它模块组

或包括该语言单位在内的更大的语言单位。

在C++中，要创建一个对象，首先必须用关键字 class 定义它的一般形式。类和结构(struct)在语法上很相似。下面的类定义了一个用于创建栈的类型 stack。

```
#define SIZE 100
class stack
{
        int s[SIZE];
        int tos;
    public:
        void init();
        void push(int i);
        int pop(void);
};
```

通常情况下，类中包括的变量和函数分别称作类的成员变量和成员函数，或者统称为类成员，如上面定义的类中变量 tos 为类 stack 的成员变量(有时也称数据成员)，函数 pop 等为 stack 的成员函数。

一个类可能包含专有(或私有)成员，也可能包含公有成员。缺省时，类中定义的所有成员都是专有的。这意味着任何不是该类成员的函数都不能存取这些变量。这是获得封装性的一种方法，即把变量专有化，从而控制数据项的存取权。同样也可以定义只能被该类的成员调用的专有函数。

要把类的某些成员定义为公有(可被程序其它部分存取)，就须用关键字 public 说明，关键字 public 后面说明的所有变量和函数可以被程序中的所有其它函数存取。从根本上讲，程序的其它部分总是通过公有函数访问对象的成员的。

这里必须指出，尽管用户可以在类中定义公有变量，但原则上应尽量少用或不用。应该使所有数据都成为私有的，通过公有函数控制对这些数据的存取。

这里我们应该注意的是，对象是数据和代码的联合体，类的专有部分只能被该类的成员函数存取。类实际上是一种由用户自己定义的新的类型，与C的其它数据类型所不同的是，它定义和说明的是连接代码和数据的新类型。

一旦定义了一个类，就可以用类名来创建这种类型的对象。实际上，类名变成了新数据类型声明符。如，下述语句声明了一个类型 stack 的对象 mystack：

```
stack mystack;
```

实际上，在C++中，用户定义的每个类都创建了一种可以用来创建这种类型的对象的新的数据类型。所以，一个对象就是所属类的一个实例，如同某个整型变量是整型的一个实例一样。换句话说，类是逻辑抽象，而对象才是实体，它存在于对应的计算机内存中。

类的一般声明格式为：

```
class class_name
{
    // private data and functions
  public data and functions
} object_name_list;
```

其中，对象名表(object name list)可以为空。

1.4 封装

在C中，数据和代码是相互分开的，程序所拥有的数据通常和程序本身并没有必然的联系。在C++中，则强调数据和代码的一致性。

C++语言所构造的对象模块采用封装机制，外部对模块的使用，仅能通过模块中定义的对外接口来实现。模块中不必要让外部使用的模块属性被封在内部，对外部来讲是屏蔽的，模块设计者也可以把那些不希望被外部使用的属性封装在模块内部，这被称为信息隐藏。由于减少并优化了对外接口，从而保证了模块的安全性。

模块封装机制还把模块中属性的具体实现细节隐藏起来。这样使模块的使用者仅通过接口来实现使用，并不知道实现细节，因此，在模块的修改和维护时，只要不改变接口的使用方式，仅修改和维护其实现细节决不会影响到程序中原来对模块的使用。

数据隐藏原则是当今程序设计的一个重要的原则。在C++中，类的成员的属性有三种：private，protected 和 public，private 成员是类的专(私)有成员，对于类的私有成员来讲，除了该类的成员函数或友元函数外，其它函数(代码)不能访问它们，protected 成员是受保护成员，它除了可以被该类的成员函数和友元函数访问外，还可以被其派生类的成员函数访问，而 public 成员则是公开的，可以被任何函数访问。

1.5 继承

面向对象程序设计的最重要的特征是具有继承机制。在C++语言中，类继承的基本含义是：C++提供了在已定义的类的基础上定义继承类的机制，这种继承类具有其它类一般所具有的属性。通过继承类中的定义，在继承类中可以继承它所继承的类的所有属性，也可以对它所继承的类的某些属性进行修改、删除，并且可以进一步新定义一些它所继承的类中所没有的属性。而且，一个继承类还可以同时继承多个类。

继承机制是提供对象模块可复用和可扩充的重要技术手段。

在模块提供给外部使用时，对象模块中是以对外分开的属性接口方式提供的。对象模块具有独立性，这使它可被用于不同系统中，然而，不同系统对于模块的要求往往会有所不同。要解决这个问题，通常的办法是在旧的模块基础上重新建立一个新的模块，这时旧模块中的保留部分如何与新模块中的增加部分合在一起，这就要涉及到所谓的模块开放性原则。

C++提供的继承类定义机制，就是实现模块开放原则的一种技术措施。继承机制保证在定义继承时，虽然被继承类中那些要保留的属性不出现在继承类中，但继承机制本身保证了继承类能够使用这些属性对外提供服务。

在C++中，类的继承是大型程序设计所需要考虑一大问题，继承机制使软件开发者能够通过建立具有继承关系的类体系，从而更系统化地模拟现实问题。这种方法更符合人们认识事物的一般过程。事实上人们在认识事物的过程中，总是首先根据某一类事物所具有的共同点，抽象地概括出它们的共性，然后再在研究共性的基础上，具体地研究每一特定分类所具有的个性，用类继承的概念构造某一类类型体系，刚好符合人们创建现实世界模型的思维方式。

1.6 多态性

多态性是指可以给不同的行为取一个名字或符号，它从上到下共享一个类等级。例如，在一个类中可以定义多个相同名字的函数，只要这几个函数的函数原型不同就行，下面的函数声明在C++中是允许的：

```
class myclass
{
        public:
          void func (int para1);
          void func(long para2);
};
```

在C++中虚函数和操作符重载也是程序获得多态性的重要途径。多态性使C++在编程使用上更加方便、灵活。

1.7 C++语言和C的其它一些区别

1.7.1 C++特有的输入/输出

首先，C++继承了C语言输入输出的优点，同样没有输入输出语句。这是C++能够在各种不同的机器上使用的重要原因。除此之外，C++通过重载移位运算符"<<"和">>"扩充了它们的功能，使其用于C++的输入输出。请看下面的C++程序：

```
#include "iostream.h"
main()
{
        int k;
        cout << "this is output. \n";  //Output a string.
        cout << "enter a number.";
        cin >> k;                      //Input a number.
        cout << k <<"squared is" << k * k << "\n";
        return 0;
}
```

上述程序看起来似乎和一般的C程序有很大的不同，而程序开始时嵌入的头文件iostream.h是由C++定义的，用来支持C++风格的I/O操作。下面这个语句反映出了C++程序的第一个不同等风格：

```
cout <<"this is output. \n";       //Output a comment.
```

该语句在屏幕上显示this is output.，然后回车换行。在C++中，虽然<<仍然是左移位运算符，但当用在上述情况下时，它又是输出运算符。cout是和屏幕相连的标识符，用

cout 和<<可以输出包括字符串在内的任何内部数据类型和变量。

当然，C++程序也可以用 printf()函数和任何其它的 C 语言输入输出函数，但多数程序员认为用 cout <<能体现 C++风格。更进一步讲，在这种情况下，尽管用 printf()函数其输出结果等价于 cout <<，但 C++的输入输出系统可以扩展到用户定义的对象，而这一点 printf()函数是无法办到的。

上述程序中接下来是下述语句：

```
cin >> k;
```

这个语句从键盘上读入一个值。在 C++中，运算符>>除了仍具有右移的含义之外，它又是 C++运算符。这个结构把从键盘读取的值送给 k。标识符 cin 表示键盘。通常可以用 cin <<输入一个包括字符串在内的任何基本数据类型的数据。

紧接着，下面的语句是：

```
cout << k <<"squared is" << k * k <<"\n";
```

很显然，如果该程序运行时我们从键盘上输入数字 100 给变量，那么这行程序将显示 100 squared is 10000，接着是回车换行。该行说明一次可执行几个<<输出运算。

由上述可知，C++扩展了运算符<<和>>的功能，把它们用于输入和输出，能够用于处理 C++的任何内部数据类型。

1.7.2 C++允许在程序块中任何位置声明局部变量

与 C 语言不同，C++允许在程序中的任何位置声明局部变量。在 C 中，程序块中所有的局部变量都必须在程序块的开始处声明，而 C++编译器却允许在程序块的任何位置声明所需要的局部变量。下面程序段 C 语言认为是非法的，但 C++却完全可以接受：

```
stock()
{
    for (int i = 0; i < 10; i++)

        int j;
        j = i * 2;
        int k;
        k = j + 2;
        }
}
```

无用 C++编程时，在块的开头处声明所有局部变量或是在第一次使用的地方声明，取决于编程人员的爱好。

在越靠近所使用的变量的位置声明变量有助于避免产生意外的副作用。在函数很大时，这样做显得很有利。在较短的函数中，把局部变量集中在函数开始处声明是合理的。

然而，有人认为如果把变量声明散布在整个程序块中，那么就很难找到程序中所有变量的声明就很困难了。这对于程序的维护也会造成困难。因此，有些程序员不主张利用

这一特点，读者可以根据自己的爱好，选择在C++程序块中的什么位置声明变量。

C++程序设计风格除了具有上述不同的特点之外，它继承了所有C语言不同于其它程序设计语言的特色。我们在这里就不一一列举了。

1.7.3 作用域限定运算符

操作符::用来处理名字冲突或限定变量作用域，请看下面的程序：

```
#include "iostream.h"
class   myclass
{
        int a;
      public:
        void show() { cout << a << "\n"; }
        myclass(int x){a=x;}
};

int a;
void main()
{
        myclass ob(5);
        int a;
        a = 10;
        ::a = 30;
        cout << "local a is " << a << "\n";
        cout << "global a is " << ::a << "\n";
        cout << "object ob's a is ";
        ob.show();
}
```

程序运行结果为：

```
local a is 10
global a is 30
object ob's a is 5
```

1.7.4 无名 union

C++允许在结构内定义没有名字的union，使两个或更多的实体共享结构内空间，从而节省内存。

1.7.5 const 说明符

const 说明符在实体作用域范围内可冻结一个实体值。它能冻结用一指针变量指向的数据、指针地址的值，或者指针地址和指向的数据。