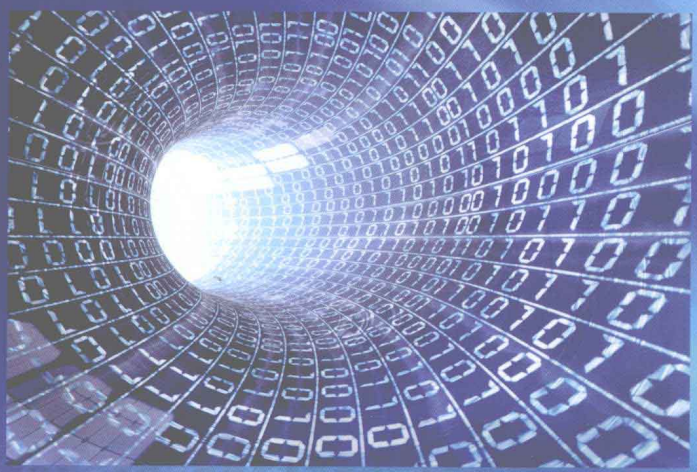
C++面向对象程序设计实用教程

主　编　沈学东



上海交通大學出版社
SHANGHAI JIAO TONG UNIVERSITY PRESS

上海市重点课程教材

C++面向对象程序设计实用教程

主　编　沈学东
编　委　王淮亭　吉顺如　陈献忠　陶　询
　　胡　静　王中华　陈群贤

上海交通大学出版社

内 容 提 要

C++是由C语言发展而来的一种面向对象的程序设计语言。本教材着重介绍C++中类与对象、类的封装、继承和多态等面向对象的特性，共计四个部分。其中第一部分(第1章)，主要介绍C++的基本语法概念和程序结构；第二部分(第2章至第4章)，主要介绍类、类的封装、继承和多态等；第三部分(第5章至第6章)，主要介绍I/O流和文件、模板和异常处理；第四部分(第7章至第8章)，主要介绍C++自带的字符串类和STL泛型编程。每个章节后给出一个连贯的综合案例，另外均附有课后习题，便于读者强化该章节所学知识，并引导读者将相关章节的知识紧密结合起来，融会贯通、灵活运用。

本书适合作为应用型本科计算机科学与技术、软件工程、网络工程等信息类专业的程序设计相关课程的教材，也可作为计算机等级考核等各类培训的参考用书。

图书在版编目(CIP)数据

C++面向对象程序设计实用教程/沈学东主编. —上海：上海交通大学出版社，2012

ISBN 978-7-313-08118-6

Ⅰ. C... Ⅱ. 沈... Ⅲ. C语言—程序设计—教材 Ⅳ. TP312

中国版本图书馆CIP数据核字(2012)第087796号

C++面向对象程序设计实用教程

沈学东 主编

上海交通大学出版社出版发行

(上海市番禺路951号 邮政编码200030)

电话：64071208 出版人：韩建民

上海交大印务有限公司 印刷 全国新华书店经销

开本：787mm×1092mm 1/16 印张：23.5 字数：535千字

2012年8月第1版 2012年8月第1次印刷

印数：1～3 030

ISBN 978-7-313-08118-6/TP 定价：28.00元

前　言

C++语言是从C语言发展演变而来的一种高效实用的程序设计语言，在实际中有着广泛的应用。它是一种非纯面向对象的编程语言，具备了面向对象程序分析和设计的基本原理和技术，又和C语言有着承接的关系。C++语言强调对高级抽象的支持，实现了类的封装、继承和多态，大大地提高了代码的可维护性和重用性。随着C++成为ANSI标准，这种面向对象的编程语言迅速普及开来，几乎在所有计算机软件开发领域都能看到C++的身影。

编写本教材的目的是改变程序设计课程的教学方法。本教材采用基于案例项目的教学方法，给出大量实际的案例情境，使学生在内容丰富的案例和教学方法中掌握C++语言，培养学生面向对象程序设计的思想，并为以后的专业学习和工作打下坚实的基础。

《C++面向对象程序设计实用教程》着重介绍了类与对象、类的封装、继承和多态等面向对象的特性。由于C++语言比较抽象而且知识点众多，初学者往往难以理解，更难以融会贯通、灵活运用。有鉴于此，教材在介绍语法概念的时候以许多丰富的实例进行说明，深入浅出地进行阐述，大大降低了该课程的学习难度。由于本课程的理论性和实践性都很强，所以理论课学时和实验课学时所占比例为2∶1。

本教材具有以下特色和特点。

1. 知识点全面，语言严谨、精炼

C++中概念众多且比较抽象，本书对这些概念进行了清楚准确的解释，并结合案例说明，让读者能全面掌握每一个知识点。

2. 实际案例开发与理论教学紧密结合，适合案例教学方法教学

为了使学生能快速地掌握C++相关知识的使用方法，本教材在各个章节的重要知识点后都附有典型的实训，本书最后一章更结合每章中的实训内容给出一个完整的项目。通过这些，学生不仅能够加深对知识点的理解，还能

将所学内容用到实际开发中去，达到教学的最终目的。

3. 结构合理、有效

本教材按照由浅入深的顺序，循序渐进地介绍了C++面向对象程序设计的相关知识。各个章节在编写的时候都是层层展开、环环相扣的。每章均附有小结和习题，便于读者强化本章节所学知识；同时每部分后面也附有实训题目，便于读者将相关章节的知识紧密结合起来，融会贯通、灵活运用。

4. 内容充实、先进、实用、有深度

本教材既使用了小型案例用于传统的教学，讲解基本知识点和概念，又引入了大型软件项目的各个模块分插在各个章节中进行教学，学完本书后可独立设计大型软件项目和开发。本书中各章的实例都是实用性很强的程序或程序模块，并且每章实例都是作为一个大型软件项目的一部分出现。实例代码可维护性强、重用率高，真正体现了面向对象程序开发的思想，引导学生在学习C++的过程中逐步递进，最终完成一个较大的软件项目，使学生在学习C++语言的同时更为今后参与实际项目开发做好准备。

同时，本教材在传统的C++概念的基础上，新增加了STL泛型编程，使C++编写的控制台程序应用更加丰富多彩。

5. 提供全部源文件、电子教案及课程网站资源

为方便读者使用本教材，书中全部实例的源文件以及电子教案均免费赠送给读者。

本书知识点全面，语言精炼，实例丰富、实用，最后的综合项目实训更是全书的特色所在，使读者通过本教材的学习具有相当的应用软件开发能力。

为方便教师进行教学，将同时出版相关的实验和习题指导书以及相关课程设计案例指导书，其中配备了大量丰富的案例和项目内容，以供教师进行参考。所有案例和项目均提供源码和技术文档，方便教师备课和指导。

本教材共8章，其中陈献忠和陶询老师负责编写第1章，王淮亭老师编写第2章，吉顺如和陈群贤老师联合编写了第3章，第5章、第6章分别由胡静和王中华老师编写，沈学东任主编、统稿并编写了第4、7、8章，还负责编写了各章中的主要综合案例。

由于编者时间仓促及水平有限，书中存在的不妥之处，敬请谅解，并欢迎提出宝贵意见和建议。

作者于上海浦东临港新城

2011年11月

目　录

第1章 面向对象程序设计基础

面向对象程序设计的思想是将数据和对该数据进行合法操作的函数封装在一起作为一个类的定义，同时提供一种对数据访问进行严格控制的机制。C++语言是20世纪80年代初期由美国贝尔实验室的科技人员提出的，它是一种继承了C语言的面向对象的程序设计语言。

本章通过C++程序的实例，介绍C++语言的基本结构、常用数据类型和函数的概念，为后续学习基于类的面向对象程序设计方法打下坚实的基础。

1.1 输入输出方式

1.1.1 简单C++程序

通过列举两个C++语言的程序，说明C++程序的结构特点和基本语法规则。

【程序1-1】 编写程序，输入两个整数x和y，用自定义函数max()求它们中的最大值。

```
#include <iostream>          //预处理命令
using namespace std;         //使用命名空间 std;
int max(int,int);            //声明自定义函数
void main()                  //主函数
{
  int x,y,z;
  cout<<"input two number:"<<endl;
  cin>>x>>y;
  z=max(x,y);                //调用 max()函数，将得到的值赋给 z
  cout<<"max="<<z<<endl;
}
int max(int a,int b)         //定义 max()函数，函数值为整型，形式参数 a,b 为整型
```

```
{
  int c;
  if(a>b)   c=a;
  else      c=b;
  return(c);                    //将 c 的值返回,通过 max 带回调用处
}
```

运行结果如下:

```
input two number:
20  6↙
max=20
```

程序的第一行为预处理命令行,包含有 iostream.h 头文件,使程序能够使用键盘和显示器进行数据的输入输出操作。endl 也定义在该头文件中,它与换行符'\n'功能相同。

本程序包括两个函数:主函数 main()和被调用的函数 max( )。max()函数的作用是将 a 和 b 中较大者的值赋给变量 c。return 语句将 c 的值返回给主函数 main()。返回值通过函数名 max()带回到 main()函数中的调用处。main()函数中的 cin 是输入函数的名字(cin 和 cout 都是 C++系统提供的标准输入输出流)。程序中 cin 的作用是从键盘输入 a 和 b 的值。

main()函数中的 z=max(x,y)为调用 max()函数的语句,在调用时将实际参数 x 和 y 的值分别传送给 max()函数中的形式参数 a 和 b。经过执行 max()函数得到一个返回值 c,把这个值赋给变量 z。最后输出 z 的值。

【程序 1-2】 带有类和对象的 C++程序。

```
#include<iostream>
using namespace std;              //使用命名空间 std;
class A                           //定义一个类 A
{
  public:                         //定义公有成员
    A(int i)                      //定义构造函数 A
    { a=i;}
    int fun1()                    //定义成员函数 fun1()
    { return 2 * a;}
    int fun2()                    //定义成员函数 fun2()
    { return a * a;}
  private:                        //定义私有成员
    int a;
};
void main()
```

```
{
  A n(4);                          //定义类 A 的对象 n,对象 n 的数据成员 a 数值为 4
  cout<<n.fun1()<<endl;    //调用对象 n 的 fun1 函数
  cout<<n.fun2()<<endl;    //调用对象 n 的 fun2 函数
}
```

运行程序结果：

```
8
16
```

本程序定义了一个类 A。类 A 中定义了 3 个公有的成员函数：一是带有参数的构造函数；二是成员函数 fun1()，其功能是求变量 a 的 2 倍数值；三是成员函数 fun2()，其功能是求变量 a 的平方值。类 A 中还定义了一个私有的数据成员 a。

在主函数中，定义了一个类 A 的对象 n，其初值为 4。最后两句输出语句，通过 A 类对象 n 调用类中的成员函数 fun1()和 fun2()，分别得到数值 8 和 16。

由上面两个例子可以归纳出 C++程序的基本结构。

1) 函数和类

C++语言程序是由若干个类和函数组成。这些类和函数可以放在一个文件中，也可以放在多个文件中。每个函数和类可以出现在程序文件中的任何位置。main()函数不一定出现在程序的开始处，但不论 main()函数的位置如何，程序运行时总是从 main()函数开始。

C++语言程序中可以有若干个类，类之间可以是继承关系，也可以是包含关系。一个类可以被嵌套在另一个类中，也可以定义在一个函数体中。在一个类中可以包含数据成员和成员函数，它们可以被指定为私有的(private)或公有的(public)属性。

2) C++的输入输出

C++语言没有专门的输入输出语句，输入输出操作是通过输入/输出流 cin 和 cout 来实现的。C++默认的标准输入设备是键盘，标准输出设备是显示器。

3) 注释和预处理

C++语言程序和 C 语言程序一样，可以在程序的任何位置插入注释信息，以增强程序的可读性。在程序开始处使用预处理命令，包含相关的头文件。

4) C++程序的书写格式

C++程序的书写格式比较自由，一行内可以写多个语句，语句之间用“;”隔开，一个语句也可以分成几行来写。为了使程序清晰易读，通常每行写一条语句。不同结构层次的语句从不同的位置开始，即按缩进格式书写成阶梯形状。

1.1.2　命名空间

在 C++中，名称(name)可以是符号常量、变量、宏、函数和类等。为了避免在大规模程序设计中，不同程序员使用各种各样的 C++库以及定义的标识符的命名时发生冲突，标准

C++引入了关键字 namespace(命名空间或名字空间),它是为解决 C++中的变量、函数等标识符的命名冲突而服务的,以便更好地控制标识符的作用域。

1. 命名空间的定义方式

命名空间是一种将程序库名称封装起来的方法,它就像在各个程序库中树立起的一道道围墙。本质上来讲,一个命名空间就定义了一个范围。命名空间的定义方式为:

```
namespace 名称
{
    //符号定义:类、变量、函数等
}
```

【程序 1-3】 命名空间的定义。

```
#include <iostream>
#include <string>
using namespace std;   //使在 C++标准类库中定义的名字在本程序中可以使用,否
                       则,iostream,string 等 C++标准类就不可见,编译时会出错。
//两个在不同命名空间中定义名字相同的变量
namespace myown1                    //声明命名空间,名为 myown1
{
  string user_name= "John" ;        //定义字符串变量 user_name
}
namespace myown2                    //声明命名空间,名为 myown2
{
   strring user_name="Alison";      //定义字符串变量 user_name
}
int main()
{
  cout<<"Hello, my name is"<< myown1::user_name<<endl;
                                    //用命名空间限定符 myown1 访问变量 user_name
  cout<<"Hello, my name is"<< myown2::user_name<<endl;
                                    //用命名空间限定符 myown2 访问变量 user_name
  return 0;
}
```

程序运行结果:

```
Hello, my name is John
Hello, my name is Alison
```

通过上面的例子可以看到,C++中命名空间的作用类似于操作系统中的文件夹。由于

文件很多，不便管理，且容易文件重名，可以通过建立不同的文件夹，对文件进行管理，使用文件时要指定文件路径。

2. 命名空间成员的使用

1) using 关键字

如果在程序中需要多次引用某个命名空间的成员，每次都要使用范围限定字符来指定该命名空间是一件很麻烦的事情。为了解决这个问题，C++引入了 using 关键字，利用 using 声明可以在引用命名空间成员时不必使用命名空间限定符“::”。using 语句通常有两种使用方式：

using namespace 命名空间名称;

using 命名空间名称::成员;

第一种形式中的命名空间名称就是程序要访问的命名空间。该命名空间中的所有成员都会被引入到当前范围中。也就是说，命名空间成员都变成当前命名空间的一部分了，使用的时候不再需要使用范围限定符了。第二种形式只是让指定的命名空间中的指定成员在当前范围中变为可见。

可以在程序开头来指定使用命名空间的名字，特别是有多个命名空间成员时，其好处在于程序中不必显式地使用命名空间限定符来访问成员。以上主程序可以修改为：

```
int main()
{
  using namespace myown1;
  cout<<"Hello, my name is" << user_name
  cout<<"Hello, my name is"<< myown2::user_name
                          //用命名空间限定符 myown2 访问变量 user_name
  return 0;
}
```

第二个变量必需用命名空间限制符来访问，因为此时 myown1 空间中的变量已经可见，如果不加限制，编译系统就会无法识别是哪一个命名空间中的变量。

2) 命名空间别名

如果命名空间名字比较长，可以为命名空间起一个别名，用来代替较长的命名空间名，这样使用起来更方便。如定义：

```
namespace myown1                    //声明命名空间，名为 myown1
{ ……}
```

再用一个较短而易记的别名代替它，如：

```
namespace my1= myown1               //别名 my1 与 myown1 等价
```

这样，别名 my1 指向原名 myown1，原来出现 myown1 的位置都可以无条件地用 my1 来代替。

3. std 标准命名空间

标准 C++把整个库定义在 std 命名空间中，即标准头文件中的函数、类、对象和模板都是在命名空间 std 中定义的。大部分程序开头都写有下面的语句：

using namespace std;

这样写是为了把 std 命名空间的成员都引入到当前的命名空间中，以便可以直接使用其中的函数和类，而不用每次都写上 std::作为限定。

当然，可以显式地在每次使用其中成员的时候都指定 std::。例如，可以显式地采用如下语句指定 cout。

std::cout << "显示输出内容"<<endl;

如果程序中只是少量地使用了 std 命名空间中的成员，或者是引入 std 命名空间可能导致命名空间的冲突，那就没有必要使用 using namespace std;了。然而，如果在程序中要多次使用 std 命名空间的成员，则采用 using namespace std;的方式把 std 命名空间的成员作为全局量来使用会方便很多，而不用每次都单独在使用的时候显式指定。

用标准的 C++编程，是应该对命名空间 std 的成员进行声明或限定的。但目前所使用的 C++库大多是几年前开发的，当时并没有命名空间，库中的有关内容也没有放在 std 命名空间中，因而在程序中可以不必对 std 进行声明。

4. 没有名称的命名空间

有一种特殊的命名空间，叫做未命名的命名空间。这种没有名称的命名空间可以创建在一个文件范围里可用的命名空间。如在文件 K 中声明以下的无名命名空间：

```
namespace
{
  void fun()
  { cout<<"Hello"<<endl;}
}
```

使用这种没有名称的命名空间只有在声明它的文件中才是可见的标识符。也就是说，只有在声明这个命名空间的文件中，它的成员才是可见的，它的成员才是可以被直接使用的，不需要命名空间名称来修饰。如上面定义的无名命名空间的成员 fun 函数的作用域为文件 K，在文件 K 中使用无名命名空间的成员 fun，不必用命名空间名限定。

对于其他文件，该命名空间是不可见的，即无法使用 fun 函数，从而将 fun 函数的作用域限制在本文件范围中。把全局名称的作用域限制在声明它的文件的另一种方式就是把它声明为静态的。尽管 C++是支持静态全局声明的，但是更好的方式就是使用未命名的命名空间。

1.1.3 输入输出格式

C++语言提供一个面向对象的输入输出流库来实现数据的输入和输出，用于输入输出的流库包含在头文件 iostream.h 中。

1. 标准输入输出

标准输入输出流是指数据从键盘流入正在运行的程序或从程序流向显示器。C++编译系统对输入输出格式做了规定，用户可以使用这些格式，但不能任意改变。

cin 用于输入流操作，与提取操作符“>>”配合可以实现从键盘输入数值和字符，并赋值给指定的变量。一般格式为：**cin >> <变量名 1>[>> <变量名 2> >>…>> <变量名 n>];**

例如：int　number，x，y；

```
cin>>number;
cin>>x>>y;
```

从键盘输入数值：10↙

20 30↙

cout 用于输出流操作，与插入操作符“<<”配合可以实现向显示器输出数据。一般格式为：**cout<< <表达式 1> [<< <表达式 2> <<…<< <表达式 n>];**

例如：int x=10，y=5，z；

```
z=x+y;
cout<<"x+y="<<z<<endl;
```

输出结果为：x+y=15。

2. 格式化输入输出

C++提供了一些格式控制符，可以直接插入到 cin 或 cout 流中，用于输入输出的格式改变，以满足用户程序的特殊要求。C++编译系统提供的 I/O 流的常用控制符如表 1-1 所示。

表 1-1　I/O 流的常用控制符

控　制　符	描　　述
dex	数值采用十进制表示
oct	数值采用八进制表示
hex	数值采用十六进制表示
setfill(w)	设置填充字符 w
setprecision(m)	设显示小数精度为 m 位
setw(m)	设置输出数据的宽度为 m
setiosflags(ios::fixed)	固定的浮点数显示
setiosflags(ios::scientific)	浮点数采用科学记数法表示
setiosflags(ios::right)	右对齐
setiosflags(ios::left)	左对齐
setiosflags(ios::skipws)	忽略前导空白
setiosflags(ios::lowercase)	16 进制数小写输出