高等学校计算机专业教材

C++面向对象程序设计

朱战立　张玉祥　编著

人 民 邮 电 出 版 社

图书在版编目（CIP）数据

C++面向对象程序设计 / 朱战立编著．—北京：人民邮电出版社，2006.4

（高等学校计算机专业教材）

ISBN 7-115-13943-1

Ⅰ．C… Ⅱ．朱… 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 016441 号

内 容 提 要

面向对象程序设计是目前软件的开发流方法。本书讨论面向对象程序设计的基本概念，以及使用 C++语言进行面向对象程序设计的基本方法。本书的内容主要包括：C++语言基础、面向对象程序设计、类和对象、友元和运算符重载、继承、运行时的多态性和抽象类、模板、异常处理、I/O 流类库、标准模板库等。另外，为方便学生上机实践，附录中还讨论 Visual C++集成开发环境的组成、编译和运行 C++程序的方法、程序调试技术、C++下的 Windows 编程方法等。

本书既可作为大专院校计算机等专业“面向对象程序设计”课程的教材，也可作为从事计算机开发和应用的工程技术人员的自学参考书。

高等学校计算机专业教材

C++面向对象程序设计

◆ 编　　著　朱战立　张玉祥

责任编辑　张孟玮

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号

邮编　100061　　电子函件　315@ptpress.com.cn

网址　http://www.ptpress.com.cn

北京艺辉印刷有限公司印刷

新华书店总店北京发行所经销

◆ 开本：787×1092　1/16

印张：17.75

字数：426 千字　　2006 年 4 月第 1 版

印数：1 – 3 000 册　　2006 年 4 月北京第 1 次印刷

ISBN 7-115-13943-1/TP・4930

定价：24.00 元

读者服务热线：（010）67170985　印装质量热线：（010）67129223

编 者 的 话

面向对象程序设计方法是软件分析、设计和实现的一种新方法。面向对象软件开发以人类习惯的解决问题方法来进行软件开发，从而使软件开发过程和人类的求解问题过程一致。面向对象软件开发的出发点和基本原则，是尽可能模拟人类习惯的思维模式，使软件开发过程尽可能接近人类解决问题的过程。目前，面向对象程序设计方法已成为软件开发的主流方法。

本书讨论面向对象程序设计的基本概念，以及使用 C++语言进行面向对象程序设计的基本方法。本书的内容主要包括：C++语言基础、面向对象程序设计、类和对象、友元和运算符重载、继承、运行时的多态性和抽象类、模板、异常处理、I/O 流类库、标准模板库等。

本书内容组织的基本思想是，以面向对象程序设计的理论为指导，以 C++语言面向对象程序设计的具体方法为落脚点。这样做有利于学生理论结合实际，在学习掌握面向对象程序设计基本概念和基本方法的基础上，具体掌握一门面向对象程序设计的语言，从而能够举一反三，触类旁通。

本书每章都设计了一个较大的应用程序，这种应用程序既可以帮助学生理解该章的基本概念，又是学生自己上机编程的样板。另外，为方便学生上机实践，附录中还讨论了 Visual C++集成开发环境的组成、编译和运行 C++程序的方法、程序调试技术、C++下的 Windows 编程方法等。本书的所有例子都用 Visual C++ 6.0 调试通过。

根据作者的教学体会，使用本教材授课需 32～48 学时。

本书第 1 章至第 5 章以及第 7 章至第 9 章由张玉祥编写，其余各章以及 3 个附录由朱战立编写；全书由朱战立修改定稿。另外，研究生纪福全完成了全书的电子图稿制作。

尽管作者在写作过程中非常认真和努力，但由于水平有限，错误和不足之处在所难免，敬请读者批评指正。

编著者

2006 年 1 月

目　　录

第 1 章　C++语言基础

教学要点

本章主要内容包括　C++语言的数据类型、基本语句、变量、函数的设计方法、指针变量和引用变量，以及程序预处理语句的功能等。

要求　熟练掌握 C++语言的数据类型、基本语句、变量、指针变量和引用变量的设计方法，以及函数的设计方法。函数的设计方法主要是函数的返回值、输入型参数和输出型参数的设计方法，以及实参的设计方法。另外，掌握上机编程和调试程序的基本方法。

本书讨论 C++语言面向对象程序设计。C++语言是在 C 语言基础上发展起来的一种混合了面向过程语言要素和面向对象语言要素的程序设计语言。C++语言除了支持面向对象技术外，在常规性能方面也作了许多扩充。为方便后面各章的学习，本章在分析 C++语言对 C 语言常规语句功能扩充的基础上，简要介绍 C++语言的一些基础概念。对 C++语言和 C 语言都具有的功能，我们用 C++/C 语言表示。

C 语言程序，其文件名后缀为.c；C++语言程序，其文件名后缀为.cpp。

1.1　数 据 类 型

数据类型规定了一类数据的数据位长度（或称字节个数）、取值范围以及对该类数据所能进行的操作。

1.1.1　基本数据类型

基本数据类型是系统已经定义的数据类型。C++/C 语言共定义了 7 种基本数据类型，其中 4 种为整型数，2 种为浮点型数，1 种为字符型数。数据类型不同，所定义的变量占用的内存空间、取值范围以及对该类数据所能进行的操作也不同。

C++/C 语言定义的 7 种基本数据类型及相应的关键字如下：

- 整型：byte、short、int、long
- 浮点型：float、double
- 字符型：char

C++/C 语言的基本数据类型、字节数和数值范围如表 1.1 所示。

表 1.1　　基本数据类型和数值范围

数据类型关键字	字　节　数	数　值　范　围
byte	1	?128～127，即?2^7～2^7?1
short	2	?32,768～32,767，即?2^{15}～2^{15}?1
int	4	?2,147,483,648～2,147,483,647，即?2^{31}～2^{31}?1
long	8	?2^{63}～2^{63}?1
float	4	3.4e?038～3.4e+038
double	8	1.7e?308～1.7e+308

C++/C 语言的字符串用字符数组表示。如语句：

```
char str[10] = "abcdefghi";
```

就定义了长度为 10 的字符数组变量 str，且该变量的初始值为“abcdefghi”。

另外，C++/C 语言还有空类型，其关键字是 void。空类型主要用来定义函数返回值的类型。

1.1.2　枚举类型

和 C 语言不同的是，C++语言的枚举类型是一个真正的类型名，即在 C++语言中枚举类型可像其他类型一样使用。系统对待枚举类型变量将像对待其他类型的变量一样。

C++枚举类型的一般形式是：

```
enum <标识符> {<枚举列表>};
```

其中，enum 是枚举类型标识，枚举列表中定义了该枚举类型的所有枚举值。枚举列表的枚举值和编号从 0 开始的整数值对应。

一旦定义了枚举类型，就可以定义该枚举类型的变量。枚举类型变量允许的操作只有赋值操作。

一个枚举类型变量定义和使用的例子如下：

```
#include <iostream.h>

enum Color{Red, Green, Yellow};
void main(void)
{
    Color myColor;              //Color 是枚举类型名
    myColor = Yellow;           //赋值操作
    cout << myColor;            //MyColor 中保存的值是符号数字常量 2
}
```

1.1.3　结构体

高级程序设计语言中都只定义了如 int、long、float、char 等基本数据类型，在有些程序设计问题中，需要把若干个基本数据类型的数据作为一个整体来考虑。在 C++/C 语言中，这样的问题是通过结构体定义语句来解决的。结构体定义语句的一般格式为：

```
struct <结构体名>
{
    <成员表列>
};
```

其中结构体名是结构体的标识符，成员表列中的每一项都由已定义数据类型名和成员名两部分组成。由于结构体中所有成员的数据类型都是已定义的，所以可以把结构体看做一个新的、用户自定义的数据类型。换句话说，一旦定义了一个结构体，就可以用该结构体定义变量。

例如，要处理学生信息时，假如要处理的学生信息包括学生的学号、姓名、性别、年龄，就可以把学生的这些信息定义成一个结构体。结构体定义语句如下：

```
struct Student
{
    long    number;                    //学号
    char    name[10];                  //姓名
    char    sex[3];                    //性别
    int     age;                       //年龄
};
```

对结构体类型的变量，既可以整体处理，也可以按成员分量处理。整体处理的例子如下：

```
struct Student x = {100001, "张三", "男", 26}, y, *p;
y = x;                                 //结构体赋值
p = &x;                                //结构体地址赋值
```

按成员分量处理的例子如下：

```
struct Student x = {100001, "张三", "男", 26}, y, z, *p;
y.number = x.number;                   //变量的成员分量赋值
p = &x;                                //结构体地址赋值
z.number = p->number;                  //指针类型变量的成员分量赋值
```

注意，这里指针类型变量的成员表示方法和非指针类型变量的成员表示方法的不同。

C++语言的结构体类型定义方法和 C 语言的定义方法基本类同，差别主要有两点：

（1）C++语言的结构体类型定义和使用更简单一些，结构体一旦定义，就可以直接使用该结构体名定义变量，而不用在结构体名前加标识符 struct。

（2）C++语言把结构体看做类的特殊形式，即结构体是没有私有部分的类。换句话说，在 C++语言中，关键字 struct 的含义和后面要讨论的类的关键字 class 基本类同，只是用 struct 定义的类中，所有成员的权限都是 public。

1.2　基 本 语 句

程序是由各种语句根据问题的要求有机组合而成的。C++/C 语言的基本语句可分为赋值语句、自加减表达式语句、分支语句、循环语句和控制语句。

1.2.1 赋值语句

赋值语句用来给变量赋值。最基本的赋值语句语法为：

```
<变量>  <赋值运算符>  <表达式>;
```

其中，最常用的赋值运算符是“=”。另外，还有复合运算符+=，?=，*=，/=等运算符。表达式分为算术表达式、关系表达式和逻辑表达式。赋值语句中的表达式主要是算术表达式。算术表达式是由算术运算符组成的表达式。例如：

```
int y = 0, x = 5;                    //变量定义并初始化赋值
y = x + 3;                           //把算术表达式 x+3 的值赋给变量 y
x += 5;                              //把算术表达式 x+5 的值赋给变量 x
```

语句 x += 5 也可以写成：

```
x = x + 5;
```

1.2.2 自加减表达式语句

自加减表达式语句是特殊情况下赋值语句的简略形式。例如，对于如下程序段：

```
int i = 0, x = 0, sum = 0;
i++;                                 //自加减表达式语句
x += 5;
sum = sum + x;
```

其中，变量 i 的初始值为 0，i++表示变量 i 的值等于变量 i 的值加 1，即 i++也可以写为：

```
i = i + 1;
```

自加减表达式语句的其他例子还有：

```
i++;
++i;
i--;
--i;
```

1.2.3 分支语句

分支语句用来根据不同的条件构造不同的语句执行流程。if 语句的语法形式为：

```
if  (<条件>)
     <语句> ;
```

或：

```
if  (<条件>)
      <语句 1>;
else
      <语句 2>;
```

第一种分支语句的含义是：假如条件成立则执行语句，否则跳过执行后续语句；第二种分支语句的含义是：假如条件成立则执行语句 1，否则执行语句 2。

条件是由关系表达式或逻辑表达式组成的一个其值为真（即其值大于 0）或为假（即其

值小于或等于 0）的表达式。

当条件后面的语句多于一句时，要用一对花括号“{}”把这些语句括起来作为一个部分看待。

【例 1.1】　编写一个程序，完成：对于两个 float 类型的数值 a=0.5 和 b=1.6，首先计算出 a+b 和 a*b，然后判断 a+b 和 a*b 的大小关系，如果 a+b>a*b，则输出 a+b，否则输出 a*b。

程序设计如下：

```
//Exam1-1.c
#include <stdio.h>

void main(void)
{
    float a = 0.5, b = 1.6, s1,s2;
    s1=a+b;                                  //两个数的和
    s2=a*b;                                  //两个数的积
    if (s1 > s2)
       printf("a + b = %f\n", s1);           //输出 a+b
    else
       printf("a * b = %f\n", s2);           //输出 a*b
}
```

程序运行结果为：

```
a + b = 2.100000
```

if 语句可以嵌套使用，即可以在一个 if 语句中又包含另一个 if 语句，从而构成程序执行的多个分支。但是大多数情况下，当程序存在多个分支时，一般使用 switch 语句。switch 语句的语法形式为：

```
switch (<表达式>)
{
    case  <常量 1>：<语句序列 1>;
    case  <常量 2>：<语句序列 2>;
    …
    case  <常量 n>：<语句序列 n>;
    default：<语句序列 n+1>;
}
```

switch 语句中表达式的值与某一 case 后面的常量 i 匹配时，就执行此 case 后面的所有语句序列；如果表达式的值与任何 case 后面的常量都不匹配时，则执行语句序列 n+1。

要注意的是，switch 语句中语句序列的最后一条语句通常都是 break 语句。

1.2.4　循环语句

循环语句用来构造程序中需要重复执行的语句部分。循环语句用来构造满足一定条件时，对同一个程序段重复执行若干次的程序结构。

循环语句主要有 while 语句和 for 语句两种。while 语句主要用来构造循环次数不固定的循环。for 语句主要用来构造循环次数固定的循环。实际上，这两种循环语句的功能完全一样，也就是说，既可以把 while 语句构造的循环结构改造成 for 语句构造的循环结构，也可以把 for 语句构造的循环结构改造成 while 语句构造的循环结构。

while 循环的语法形式为：

```
while  (<条件>)
    <循环体>;
```

其含义是：当条件成立时反复执行循环体中的语句，直到条件不成立时为止。

for 循环的语法形式为：

```
for(<初始表达式>; <终止表达式>; <增量表达式>)
    <循环体>;
```

其循环执行流程为：

（1）计算初始表达式（这通常是一个或用逗号分隔的一组赋值语句）；

（2）判断终止表达式（这通常是一个条件），若条件成立则执行循环体中的语句，若条件不成立则跳出循环语句；

（3）计算增量表达式（这通常是一个自加减表达式语句），然后转入（2）。

注意，3 个不同的表达式之间用逗号分隔开。当语句多于一句时，要用一对花括号“{}”把循环体中的语句括起来。

【例 1.2】 编写求 1～1000 累加和的程序，并将计算结果打印出来。

for 循环结构程序如下：

```
//Exam1-2-1.c
#include <stdio.h>

void main(void)
{
    long sum = 0;
    int i;
    for(i = 1; i <= 1000; i++)
        sum = sum + i;
    printf("sum = %ld\n", sum);
}
```

由于本例的循环次数事先已知，所以用 for 语句实现较好。为说明 while 语句和 for 语句的功能完全相同，再用 while 语句设计如下：

```
//Exam1-2-2.c
#include <stdio.h>

void main(void)
{
    long sum = 0;
```

```
    int i = 1;
    while(i <= 1000)
    {
        sum = sum + i;
        i++;
    }
    printf("sum = %ld\n", sum);
}
```

两个程序的运行结果均为：

```
sum = 500500
```

循环语句还有 do-while 语句，其实现的功能和 while 语句完全相同，只是语法略有不同。

1.2.5　流程控制语句

流程控制语句主要有 break 语句和 continue 语句。

break 语句的语法形式为：

```
break;
```

其语义是跳出当前的 switch 语句或循环语句。

continue 语句的语法形式为：

```
continue;
```

其语义为：结束本次循环，即跳过循环语句中尚未执行的语句，接着进行循环条件的判定。continue 语句只用在 for，while，do-while 等循环语句中，一般是与 if 语句一起使用，可以加速循环。

【例 1.3】　设计一个程序完成以下功能：若输入英文字母 Y 或 y，则继续执行；若输入英文字母 N 或 n 则结束；若输入其他字符不做任何处理。

```
//Exam1-3.c
#include <stdio.h>

void main(void)
{
    char    ch;
    do
    {
        ch = getche();
        if(ch = = 'Y' || ch = = 'y')
            continue;
        if(ch = = 'N' || ch = = 'n')
            break;
      }while(1);
}
```

1.3 变　　量

C++语言中的变量和 C 语言中的变量相比，功能扩充之处主要体现在：变量的定义方法、const 类型限定符、函数形式的变量类型转换等。

1.3.1 变量的定义方法

C 语言只允许变量在程序开始处定义，而 C++语言允许变量在程序的任何位置定义，这就使得 C++语言的变量除全局变量和局部变量外，又增加了块变量。

C++语言把用花括号{}括起来的一块区域称为块。块变量就是定义在某个块中的变量。变量的作用域就是变量的作用范围。块变量的作用域就是该变量定义的由花括号{}括起来的范围，称做块作用域。块变量在其作用域内是可见的，在其作用域外是不可见的。

当程序比较大时，在程序的开始处定义程序中要使用的所有变量通常比较麻烦，而 C++程序就可以在需要使用一个变量时再定义它。例如，下例程序对 C 语言程序来说是错误的，而对 C++语言程序就是正确的。

```
//Example.cpp
void main(void)
{
    long sum = 0;
    int n = 100;
    for(int i = 0; i < n; i++)
    {
        for(int j = 0; j < n; j++)
        {
            sum = sum + i + j;
        }
    }
}
```

1.3.2 const 类型限定符

关键字 const 用来限定变量或函数参数（或函数返回值）的修改。

当用 const 修饰一个变量时，则限定该变量在定义域范围内为常量。并要求该变量必须在定义时初始化赋值，且以后不允许再重新赋值修改。

用 C 语言编程序时通常用宏定义命令来定义常量，如要定义 MaxSize 为 100，则：

```
#define MaxSize 100
```

而用 C++语言编程序时通常用 const 来定义常量，如：

```
const int maxSize = 100;
```

虽然这两种方法均可定义程序中使用的常量，但两种定义方法有如下两点主要差别：

（1）宏定义命令定义的常量只是文本替换，如用 100 替换 MaxSize，由于没有数据类型，所以 C 程序编译时也不做数据类型检查；而 const 定义的常量带有数据类型，如 maxSize 就是整数类型。定义有数据类型的常量在进行表达式运算时，可以进行类型检查，从而防止程序出错。

（2）宏定义命令定义的常量的作用域类似于全局变量；而用 const 定义的常量可以像定义变量那样，在程序的任意地方进行定义，其作用域随定义位置的不同而不同。现在通行的程序设计方法主张取消（至少尽量减少）全局变量，因为全局变量不好维护，容易引起错误。

当用 const 修饰一个函数的形式参数时，则限定相应的实际参数在该函数内为常量，即该参数不允许在函数内被修改。

1.3.3　函数形式的变量类型转换

C++语言对变量提供了函数形式的显式类型转换。任何系统定义的基本数据类型或用户自定义的数据类型的名字，都可以作为函数使用，从而把变量（或常量）从一种数据类型转换到另外一种数据类型。函数形式的类型转换例子如下：

```
int i = 20;
float x;
x = float(i);                          //把变量 i 转换为 float 类型
```

而在 C 语言中，上述例子的类型转换方法是：

```
x = (float)i;
```

1.4　函　　数

C++/C 语言都支持面向过程的程序设计。面向过程的程序设计以函数为程序模块化设计的基础。函数是完成特定功能的程序块。函数中的参数是一个变量，其具体取值由调用函数在调用时传送的数据（即实际参数）决定，因此，一个函数可以被许多别的函数用不同的参数调用。

函数的一般形式为：

```
数据类型  函数名(数据类型  参数 1，…，数据类型  参数 n)
{
        函数体
}
```

函数由函数名、参数和函数体三部分组成。一个函数的函数名既是该函数的代表，也可以用返回值带回函数的处理结果。函数的参数称为形参或虚参。函数的参数有输入型参数和输出型参数两种。调用函数的函数（最常见的是主函数）称为调用函数，被调用函数调用的函数称为被调用函数。调用函数中使用的参数称为实参。

函数形参的形式不同，实参和形参间传送数据的功能就不同，从而可以实现输入型参数