21st CENTURY
规划教材

面向21世纪高等院校计算机系列规划教材

COMPUTER COURSES FOR UNDERGRADUATE EDUCATION

Visual C++程序设计

严迪新　班建民　主编

科学出版社
www.sciencep.com

面向21世纪高等院校计算机系列规划教材
COMPUTER COURSES FOR UNDERGRADUATE EDUCATION

Visual C++程序设计

严迪新　班建民　主编

科学出版社
北京

内 容 简 介

本书是作者在总结多年软件开发和从事 Visual C++ 教学经验的基础上，按照高等学校计算机及相关专业教学要求编写而成的。

本书系统介绍了 Visual C++ 面向对象编程的基础知识和基本方法。本书分为 11 章，主要内容包括 C++ 程序设计、Visual C++ 编程基础和 Visual C++ 高级编程三个部分。本书采用案例方式，讲练结合，并在大多数章节中以实际应用为背景，从简单案例阅读和较复杂案例的分析，到自行设计解决方案，逐步培养学生的程序设计能力和综合开发能力。

本书适合作为高等学校相关课程的教材或参考书，也可供实际应用开发人员学习参考。

图书在版编目(CIP)数据

Visual C++ 程序设计/严迪新，班建民主编. —北京：科学出版社，2005
(面向 21 世纪高等院校计算机系列规划教材)
ISBN 7-03-015115-1
Ⅰ. V… Ⅱ. ①严…②班… 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(2005)第 016524 号

责任编辑：吕建忠 韩 洁/责任校对：都 岚
责任印制：吕春珉/封面设计：王 浩

科学出版社 出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com

新蕾印刷厂 印刷
科学出版社发行 各地新华书店经销
*
2005 年 4 月第 一 版 开本：787×1092 1/16
2005 年 4 月第一次印刷 印张：24
印数：1－3 000 字数：551 000

定价：30.00 元

(如有印装质量问题，我社负责调换〈环伟〉)
销售部电话 010－62136131 编辑部电话 010－62138978－8001(HI01)

前　言

Visual C++是Microsoft公司推出的一个面向对象的、功能丰富的可视化开发工具，是目前最为流行、使用最为广泛的软件开发工具之一。Visual C++支持MFC(Microsoft Foundation Class)类库编程，并有强大的集成开发环境Developer Studio(包括程序自动生成向导AppWizard、类向导Class Wizard、各种资源编辑器以及功能强大的调试器等可视化和自动化编程辅助工具)。

Visual C++不仅适用于传统的C/C++开发过程，更充分优化了对面向对象技术的支持。Visual C++可用来开发各种类型、不同规模和复杂程度的应用程序，并具有开发效率高、软件代码品质优良等特点。因此学习和掌握Visual C++编程方法和技巧，是开发高质量应用程序必不可少的条件之一，也是广大程序设计和开发人员的迫切需要。教学实践证明，Visual C++作为应用非常广泛的Windows程序开发工具，在高校相关专业开设其编程课程是非常必要且有效的。

Visual C++编程的内容广泛并相互交织，面对Visual C++这样一个庞大的程序设计语言，采用传统的以语意、语法为主的教学方法已感到力不从心。考虑到案例学习法是绝大多数软件开发人员学习、进修采用的主要方法。在本书的编写过程中，采取了一种经过编者教学实践，证明为行之有效的方法——案例学习法，在大多数章节中以实际应用为背景，首先提出一个简单案例和相应的解决方案，分析其中用到的语法规范和可选用途，引出相关知识，引导学生应用这些知识进行较为复杂案例的分析并设计解决方案，同时通过上机习题，逐步扩展学生解决问题的范围，并在解决问题的过程中学习程序设计知识，培养学生分析和解决问题的能力。

本书系统介绍了Visual C++面向对象编程的基础知识和基本方法，适用于高等学校计算机及相关专业学生程序设计能力的培养。主要内容包括C++程序设计、Visual C++编程基础和Visual C++高级编程三个部分。本书共11章，前2章为从C到Visual C++的过渡部分；第3章至第7章为Visual C++的编程基础，重点介绍Win32程序、对话框、控件、文档视图等多种程序类型结构，以适应不同的应用环境；第8章至第10章为提高部分，有进程、数据库、网络通信等内容，这部分内容涉及计算机专业课程中的部分专题，并与这些课程有相互补充和相互依托关系；第11章介绍Visual C++的集成环境与调试技术。特别要说明的是，本书是作者在总结多年来进行软件开发和从事Visual C++教学经验的基础上，按照高等学校计算机及相关专业教学要求编写而成的。

本书内容由浅入深，循序渐进，结构清晰，重点突出，实例丰富，适合作为高等学校相关课程的教材或参考书，也可供实际应用开发人员学习参考。书中主要程序都在Visual C++6.0和Windows 2000环境下调试通过，如果没有特别指明，所有实例同样可以在Windows 98/NT/2000/XP/2003环境下运行。

本书由严迪新、班建民主编，其中严迪新编写第8～10章；班建民编写第3～7章；崔

玉玲编写第1、2章;周虹编写第11章。最后,由严迪新、班建民统一定稿。

因作者水平有限和编写时间仓促,书中不妥和错误之处在所难免,敬请广大读者及同行专家批评指正。

编 者

2004年11月

目　录

第 1 章　从 C 到 C++

自 20 世纪 80 年代美国 AT&T 贝尔实验室的 Bjarne Stroustrup 在 C 语言的基础上推出了 C++ 以来，C++ 越来越受到重视并得到广泛应用。C++ 语言是在基于 C 语言的语法基础上融入了 Simula 67、Algol 68 和 ADA 等语言中许多独特的语法特点之后而形成的全新的语言体系。C++ 在两个方面进行了扩展：一是对原来的面向过程的机制作了许多改进；二是增加了面向对象的机制，从而使它成为既支持面向过程的程序设计，又支持面向对象程序设计的混合型程序设计语言。用户在学习 Visual C++ 编程之前，必须熟练掌握 C++ 程序设计方法。本章首先介绍 C++ 的特点，然后介绍面向对象的程序设计。通过本章的学习，使用户完成从 C 语言编程向 C++ 语言编程的过渡。

在此之前需要说明的是：由于 C++ 语言的语法基于 C 语言，并考虑到大多数学习 C++ 语言的用户应当系统地学习过 C 语言程序设计等课程，所以本章以此为起点来讲述C++ 语言，其更多的篇幅是在同 C 语言进行比较的基础上来论述 C++ 语言语法的。

1.1　C++ 语言新增的约定、符号名称和运算符

1.1.1　C++ 语言的文件扩展名

为了使编译器能够区别是 C 语言还是 C++ 语言，一般说来，C 语言的源程序文件名后缀一般为 .c，而 C++ 的源程序文件名后缀一般是 .cpp。虽然仅差两个字母，但编译时的处理却相差甚远。所以凡是讲到的属于 C++ 语言特有的语法描述的内容只能书写在以 .cpp 为扩展名的文件中（即按 C++ 语法处理）。反过来，按 C 语言语法书写的程序却大多可以使用 .cpp 的文件扩展名并用 C++ 编译器处理。

1.1.2　C++ 语言的输入/输出

在 C 语言中，输入/输出本是依靠函数来实现的（标准的输入/输出用 scanf、printf 等）。通常，只要在程序的最前端放置对应的头文件声明 # include＜stdio.h＞就可以引用这类函数。由于此种函数的引用语句书写起来比较长，为了简化起见，C++ 语言又另外定义了一套保留字与运算符来替代 C 语言中对标准输入/输出函数的引用。C++ 语言的保留字为：

```
cout<<"输出内容"<<…;        /*为标准输出运算符（默认为显示器）*/
cin>>"输入内容">>…;         /*为标准输入运算符（默认为键盘）*/
```

支持此二运算符的内部函数体在一个名为“iostream.h”（即标准 I/O 流）的头文件中予以声明，所以应用时一定要将其放置在对应的头文件声明部位。

【例 1.1】

```
#include<iostream.h>
void main()
{    char  name[10];
     int age;
     cout<<"please  input  your  name:";
     cin>>name;
     cout<<"How  old  are  you:";
     cin>>age;
     cout<<"name  is  "<<name<<"\n";
     cout<<"age  is  "<<age<<"\n";
}
```

将例 1.1 的标准输入/输出语法与 C 语言的 printf 和 scanf 函数的书写语法对比后可以很明显地看出，此种标准的输入/输出语法的书写大大简化了函数格式的描述，而由 C++语言编译器按被操作数的类型和具体内容代为选定默认格式（实际上此种方法也有一整套控制格式的手段）。

1.1.3 数据类型声明的变化

1. 数据类型声明的变化

C++语言除了新增的类（即 Class，其应用参见以后介绍）数据类型以外，继承了 C 语言所支持的所有数据类型。但在数据类型的声明上做了两种较大的改变，一是允许数据类型的声明语句可以出现在程序的任何位置；二是允许直接使用结构体名定义实体。

【例 1.2】

```
void  main()
{
    struct  Dt{unsigned  int mm,dd,yy;};
    Dt dt[3];
    for(int i=0;i<3;i++)
    {
      dt[i].mm=8;
      dt[i].yy=1994;
      dt[i].dd=i+15;
    }
}
```

2. 行注释符//的使用

在 C 语言中用“/*…*/”符号作程序注释，这种注释叫做段注释。只当做单行注释时便可用“//”符号表示，从此符号起至行尾均为行注释内容。

1.1.4 动态内存分配运算符

这种运算符有两个：new 和 delete。无须任何函数支持（即由 C++ 语言编译器直接处理）。C++ 语言之所以要以 new 和 delete 运算符来代替 C 语言内繁多的内存动态分配函数，是由于在面向对象的程序运行中，对象的产生和撤销极其频繁，以至于用 C 语言的动态内存分配函数来完成这种内存的占用和释放不仅过于烦琐，甚至是不可能的。

1. new 的功能与格式

new 运算符的功能等效于 C 语言中 malloc 一类的函数功能，用来动态地为对象或数据分配内存。即在尚未占用的内存空间中为对应类型的数据按实际需要来安排空间，并带回分配的首址。其语法格式为：

指向对应类型的指针 = new 类型描述符;

类型描述符可有下面几种书写形式。

①仅占一个单元空间：

```
int  *p;
p=new  int;          /*占一个机器字长*/
```

②仅占一个单元空间且赋初值：

```
int  *p;
p=new  int(20);      /*占一个机器字长且赋予初值 20*/
```

③占用多个单元可按数组或指针使用：

```
int  *p;
p=new  int[20];      /*占 20 个机器字长单元*/
```

2. delete 的功能与格式

delete 运算符的功能等效于 C 语言中的 free 一类的函数功能。其语法格式为：

delete 指针名;

如果指针是数组还可在指针名前加上“[]”表示，即

delete [n]指针名;

若 n 值小于定义时的指针单元数，则释放自 n 单元起的内存空间。

【例 1.3】

```
#include<iostream.h>
#include<stdio.h>
void  main()
{
  unsigned  int  *a,*b;
  struct  dt{unsigned  int  mm,dd,yy;};
  a=new  unsigned  int(20);        /*使a指向一个机器字长的地址单元并赋初值 20*/
                                   /*即等效于a=new  unsigned  int;  *a=20;*/
  b=new  unsigned  int[20];        /*使b指向 20 个机器字长的数组单元的第一个*/
```

```
    for(int  i=0;i<20;i++)b[i]=i;   /*为这20数组单元赋初值0~19*/
    cout<<*a<<''<<b[0]<<''<<b[19]<<"\n";
    dt  *c=new  dt;
            /*使c指向一个struct dt数据结构对象的首地址并赋初值1994.8.15*/
    c->yy=1994;c->mm=8;c->dd=15;
    cout<<c+>yy<<'.'<<c+>mm<<'.'<<c->dd<<"\n";
    printf("%Np,%Np,%Np\n",a,b,c);     /*显示a、b、c的地址值*/
    delete  a;delete  []b;
    a=new  unsigned  int[300];
    dt  *d=new  dt;
    *d=*c;                                  /*将c的内容复制到d内*/
    cout<<d->yy<<'.'<<d->mm<<'.'<<d->dd<<"\n";
    printf("%Np,%Np,%Np,%Np\n",a,b,c,d);/*显示a、b、c、d的地址值*/
}
```

显示结果：

```
004300E0, 00
003461F8, 00
20  0  19
1994.8.15
1994.8.15
```

1.1.5 引用类型

引用（references）类型是C++语言的一个特殊的数据类型描述，在程序的不同部分使用两个以上的变量名指向同一地址，使得对其中任一个变量的操作实际上都是对同一地址单元进行的。在这种两个以上变量名的关系上，被声明为引用类型的变量名则是实变量名的别名。其语法格式是：

类型名 &引用型函数名或变量名=前已声明的(常)变量名;

在一行上声明多个引用型变量（函数）名时，要在每个变量（函数）名前都冠以“&”符号。

```
int  i;
int  &j=i;
j=1;
```

j和i指向同一个地址单元，改变j的值也就是改变了i的值。

【例1.4】

```
int  i=0;
int  &j=i,k=2;
j++;
k++;
cout<<&i<<"->"<<i;
cout<<&j<<"->"<<j;
cout<<&k<<"->"<<k
```

例1.4的显示结果如下：

```
34D0->2
34D0->2
34D4->3
```

例1.4中地址值以十六进制数显示i的地址。分析时要注意两种不同的“&”符号的意义。例中j是i的别名，而k则是独立的变量。出现在类型声明语句中的“&”代表符号右端的别名。而在可执行句中的“&”则是取变量的地址。此区别千万不能混淆。对于指针，使用引用类型时则要注意书写格式。

```
int  i=0,*j=&i,*&k=j;
cout<<*k;
```

切不可写成下面的样子：

```
int  i,*p;
int  &r1=p;     /*非法语句*/
int  &r2=&i;    /*非法语句*/
```

由于引用类型所声明的变量不是独立内存实体，所以不能用来对常数声明。如：“int &r=3;”的形式就是错误的。

【例1.5】

```
#include<iostream.h>
void  t(int&);
void  main()
{     int i=0;
      t(i);
      cout<<"i="<<i;      /*显示*/
}
void  t(int &ip)
{cout<<ip;ip++;}          /*显示*/
```

例1.5中函数t中的ip被定义为引用类型。把它放在形参位置上与主函数main传来的实参i相对应（指向同一空间等效于传址）。但与用指针传递所不同的是ip并非内存实体，是在编译阶段就把这两个符号的地址等同起来，到运行阶段后实际传递的仍是i的地址，只是表象为非地址方式，从而真正体现双方对同一内存地址单元的操作。例1.5通过这种实、形参在编译阶段建立的别名对应关系确实使得函数t可以直接访问被声明在main中的局部变量i。

1.1.6 const语义的扩展

在C语言中，const保留字是作为声明常数数据类型的附加类别标识使用的，如“const int i=10;”。此时，被声明的常数必须到程序执行后才能被访问。而在C++语言中，则允许被声明的常数在后续的编译中被直接访问，如下面第二句的格式在C语言编译时会报错，但在C++语言中是允许的。

```
const  int  i=9;
char  t[i];  //等效于char[9];
```

若在C语言中要实现这种技术，只能借助于“#define i 9”来预定义常数字符代换了。而#define所预定义的常数的类型在编译时是无法确定的。如果用#define预定义的常数做实参传给另一个函数时，该函数只能按声明的形式参数类型来做隐含类型转换。

【例 1.6】

```
#include<iostream.h>
#define  k  50
void  fun(char);
void  main()
{ fun(k);}
void  fun(char c)
{cout<<"c="<<c;}
```

执行后显示“c=2”就是将50（0X32）当做ASCII码解释时恰好是“2”的缘故。

1.1.7 指针声明时的类型必须同实际指向的对象类型相一致

在C语言中，指针声明时的类型与其所指向的数据实体的类型可以不一致（借助C语言内定的默认类型转换技术）。而C++语言则不再允许，这样便大大简化了C++编译器的设计复杂程度。

【例 1.7】

```
#include<iostream.h>
void  main()
  {void  *f;
  int  i=1,*p;
  f=&i
  p=(int*)f;
  f=p;
  cout<<p<<*p;
  cout<<f<<*f;  //void的值不能显示,故编译报错
  }
```

例1.7中f是void型指针，可被解释成能指向任何类型变量，所以能被任何类型指针所赋值。而指针p的类型是int，在C++语言中就不能自动地通过默认类型转换将非int类型的地址赋予它，所以只能用“(int*)”来声明强制类型转换。

1.1.8 int与char不再等价

在C语言中，char类型（如'A'）与int类型同样占用2B的空间（假设机器字长为16位）；所以有时认为二者可以等价。在C++语言中则非如此，'字符'类型的长度计量真正成为按单字节为单位计算，所以在使用时要特别注意。比如sizeof('A')的值为1，而sizeof（65）的值为2，两者不一致。