白领就业指南：

Visual C++ 6.0

设计师之路

王海龙　董智勇　董跃钧　编著
夏敏捷　审校

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

白领就业指南：
Visual C++ 6.0设计师之路

王海龙　董智勇　董跃钧　编著

夏敏捷　审校

電子工業出版社

Publishing House of Electronics Industry

北京・BEIJING

内 容 简 介

本书以面向对象程序设计语言为基础，以Visual C++开发环境为主线，结合应用实例，强调“实用”，力求以通俗易懂的语言，将面向对象程序设计的基础知识和Visual C++ 6.0的具体应用展示给读者。书中实例丰富、讲解清晰、力避代码复杂冗长，简短的实例特别有助于初学者仿效理解、把握问题的精髓，能够帮助读者快速建立对应用程序框架的整体认识。

本书既适合初学者和具有一定编程经验的Visual C++用户阅读，也可供广大计算机工作者和软件开发者参考。

图书在版编目（CIP）数据

白领就业指南：Visual C++ 6.0设计师之路/王海龙，董智勇，董跃钧编著.—北京：电子工业出版社，2006.9

ISBN 7-121-02900-6

Ⅰ.白… Ⅱ.①王…②董…③董… Ⅲ.C语言—程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字（2006）第079326号

责任编辑：徐云鹏 李红玉
印　　刷：北京天竺颖华印刷厂
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱 邮编：100036
　　北京市海淀区翠微东里甲2号 邮编：100036
经　　销：各地新华书店
开　　本：787×1092 1/16 印张：30.125 字数：750千字
印　　次：2006年9月第1次印刷
定　　价：42.00元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换，若书店售缺，请与本社发行部联系。联系电话：010-68279077。质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

前　言

Visual C++（简称VC）是微软公司推出的一个面向对象的、功能丰富的可视化开发工具，是计算机界公认的最优秀的应用开发工具之一。Visual C++是C和C++的混合编译器，这使得Visual C++开发的程序具备了C和C++的高效和简洁的特点。Visual C++是一个面向对象的语言，软件能够在源码级、类级、控件级等多个级别上重用，软件的开发效率大为加快。Visual C++借助于微软公司出色的MFC类库和应用程序框架，能够轻松地开发出基于Windows标准界面的应用程序。编写本书的目的就是为了让用户学会在Visual C++环境下，利用微软的基本类库MFC开发出功能强大的Windows应用程序。

本书包括以下三篇（共16章）。

第一篇　Visual C++入门篇——夯实求职路的基石

第1章：主要介绍了面向对象的基本概念，包括类和对象以及需要重点掌握的面向对象的封装性、继承性、多态性三大特性思想和具体体现。这部分内容对学过C++和没有学过C++的用户都有很好的指导意义。

第2章：主要介绍Windows程序的特点和进一步学习Visual C++的一些核心知识，如Win32程序结构、窗口类、窗口句柄、消息循环等，这些是理解Visual C++关于MFC应用程序框架的一些必备的知识。

第3章：主要介绍了Visual C++ 6.0开发平台的安装和配置、Visual C++集成开发环境等。

第4章：介绍了MFC应用程序框架的特点和优势以及MFC应用程序的启动流程，另外还介绍了MFC应用程序框架的基本类、MFC类库、窗口类与窗口对象等。

第二篇　Visual C++知识提高篇——引领您走向专家之路

第5章：主要介绍应用程序常见的界面元素：菜单、工具栏、自定义状态栏，并用实例辅助讲解各个元素的初始化和创建原理等方面的知识。

第6章：主要介绍了控件的创建、控件的消息处理，以及Windows通用控件。

第7章：主要介绍了设备上下文和图形对象，同时还对图形的输出和打印进行了详细的阐述。

第8章：主要介绍了模式对话框、非模式对话框，以及Windows通用对话框。

第9章：介绍文档和视图结构、如何保存数据和文件、如何从文件得到数据重建文档对象、各种常见的文档和视图结构形式的表达方法，讲解过程中使用实例辅助说明。

第10章：讨论了利用MFC ODBC和ADO访问数据库的方法、使用原理和过程。

第11章：主要介绍了如何把函数、C++类、资源打包到一个动态链接库中，以及如何使用动态链接库中的资源。

第三篇　Visual C++综合应用篇——走向白领阶层

通过第12章的“聊天室设计”、第13章的“人事管理系统设计”、第14章的“工资管理系统设计”、第15章的“屏幕捕捉”和第16章的“FTP客户端编程”五个案例详细介绍了系统的开发过程。并按照软件开发方法与流程，从需求分析到编码实施，让读者真正体会到软件项目开发的思想，同时又对前两篇的知识进行了巩固与提高。

最后说明一点，学习编程是一个实践的过程，而不仅仅是看书、看资料的过程，亲自动手编写、调试程序才是至关重要的。通过实际的编程以及积极的思考，读者可以很快地掌握一门编程语言。而且，在编程过程中读者会积累许多宝贵的编程经验，在当前的软件开发环境下，编程经验对开发者尤其显得不可或缺。

本书第1章、第2章、第12章由夏敏捷编写，第3章、第5章、第6章由董跃钧编写，第4章、第9章、第10章由王海龙编写，第7章、第8章、第11章由董智勇编写，第13章、第14章由杨要科编写，第15章、第16章由程传鹏编写。董跃钧在前期做了大量的策划准备工作，全书最终由王海龙、夏敏捷修改并统稿。在本书的编写过程中，为确保内容的正确性参阅了很多资料，并且得到了众多网站和资深Web程序员的支持，他们是刘东鸿、李欣、保春艳、马煜、欧立奇、张庆、崔竞、张小丽、张强、张音、董军、张静、李辉、李正非、钱栩，以及为本书最终出版与出版社进行不断沟通的李宗民、康祥顺老师，在此谨向他们表示衷心的感谢。

由于编者水平有限，书中难免有错，敬请广大读者批评指正，在此表示感谢。

为方便读者阅读，若需要本书配套资料，请登录“华信教育资源网”（http://www.hxedu.com.cn），在“教学资源”频道的“综合资源下载”栏目下载。

目　录

第一篇　Visual C++入门篇——夯实求职路的基石

第二篇　Visual C++知识提高篇——引领您走向专家之路

第一篇

Visual C++入门篇——夯实求职路的基石

引　言

从这里开始，我们将一起进入Visual C++的世界！

“万丈高楼平地起”，任何技术知识都是从基础开始一点一点积累起来的，在学习MFC之前，必要的基础是对于Windows程序的事件驱动特性的了解，以及对C++多态的精确体会。

在第一篇中，我们将介绍面向对象程序设计的特点、Windows程序基本概念、Visual C++集成开发环境、MFC类库、应用程序框架、窗口类和窗口对象的相关知识。

近年来“面向对象”一词席卷了整个软件界。面向对象程序设计其实是一种理念，用什么语言实现它都可以。C++是最重要的面向对象语言，因为它站在C语言的肩膀上，而C语言拥有绝对多数的使用者。在第1章中，我们将介绍面向对象的基本概念，包括类和对象以及需要重点掌握的面向对象的封装性、继承性、多态性三大特性思想和具体体现。这部分内容对学过C++和没有学过C++的人都有很好的指导意义。

从来没有学习过在“事件驱动（event driver）系统”中撰写“以消息（message）为基础”的应用程序的人，能否一步跨进MFC领域，直接以应用程序框架（Application Framework）开发Windows程序，我们一直对此持怀疑态度。在第2章中，我们将同读者一道来温习Windows程序的事件驱动，从中获得消息的产生、获得、分派、判断及处理，窗口的创建，程序的诞生与死亡等知识。

如果MFC是箭，Visual C++集成开发环境便是弓。强壮的弓，让箭飞得更远。在第3章中，我们将以概要的方式为读者介绍Visual C++的集成开发环境，目的在于认识搭配在MFC周围的这些工具的操作性与功能性，实地理解这一整套服务带给我们什么样的便利。

在应用程序框架）出现后，带有艺术气息的软件创作成为工匠技术。MFC是一个零组件超级市场，所贩卖零组件的功能以及零组件彼此之间的关系已经定义好，我们可以选择自己喜欢的零件，组成一个应用程序。在第4章中，我们将介绍MFC应用程序框架以及MFC的主要组成部分。

第1章　面向对象程序设计基础

Visual C++不仅仅是一个编译器，它还是一个全面的应用程序的开发环境，使用它你可以充分利用具有面向对象特性的C++来开发出专业级的Windows应用程序。为了能充分利用这些特性，你必须首先掌握C++程序设计语言。Visual C++与C++有很大的不同，但是C++又是Visual C++的基础，特别是C++中的继承、封装和派生的思想都运用到了Visual C++中，只不过Visual C++的库很多，其中的继承、封装和派生的关系很复杂。所以首先要掌握C++中面向对象程序设计的知识。

1.1　从结构化程序设计到面向对象程序设计

1.1.1　两种程序设计方法

面向对象技术是目前流行的系统设计技术，面向对象程序设计技术的提出，主要是为了解决传统程序设计方法——结构化程序设计所不能解决的代码重用问题。

结构化程序设计从系统的功能入手，按照工程的标准和严格的规范将系统分解为若干功能模块，系统是实现模块功能的函数和过程的集合。由于用户的需求和软、硬件技术的不断发展，按照功能划分设计的系统模块必然是易变的和不稳定的，这样开发出来的模块的可重用性不高。

近几年来，面向对象技术无论是在理论上还是在实践上都在飞速地发展。面向对象技术中最重要的就是对象的概念，它把现实世界中的气球、自行车等客观实体抽象成程序中的对象。这种对象具有一定的属性和方法，这里的属性指对象本身的各种特性参数，如气球的体积、自行车的长度等；而方法是指对象本身所能执行的功能，如气球能飞、自行车能行驶等。一个具体的对象可以有许多的属性和方法，面向对象技术的重要特点就是对象的封装性，对于外界而言，并不需要知道对象有哪些属性，也不需要知道对象本身的方法是如何实现的，而只需要调用对象所提供的方法来完成特定的功能。从这里我们可以看出，当把面向对象技术应用到程序设计中时，程序员只是在编写对象方法时才需要关心对象本身的细节问题，大部分的时间放在对对象的方法的调用上，以便组织这些对象进行协同工作。

面向对象程序设计是一种围绕真实世界的概念来组织模型的程序设计方法，它采用对象来描述问题空间的实体。面向对象程序设计从所处理的数据入手，以数据为中心而不是以服务（功能）为中心来描述系统。它把编程问题视为一个数据集合，数据相对于功能而言，具有更强的稳定性。

在面向对象设计方法中，类是一系列具有相同性质的对象的抽象，它将对象的属性（状态）和方法（行为）统一在一个单元中，是对对象共同特征的描述。以学生为例，所有学生都有学号、姓名、性别、年龄、所属系别、联系电话等，将这些共同的特征和一些方法定义

在一个模板中就构成了学生类，如果为这个学生类中指定了具体的值，如“200011070，张亮，男，20，计算机科学系，676986XX”，这就是学生类的一个实例，或者叫对象。对象是类的一个实例。

在Windows中，应用程序是以窗口形式展现的。一个窗口是由一组数据的集合和处理这些数据的方法和窗口函数组成的。从面向对象的角度来看，窗口本身就是一个对象。Windows程序的执行过程本身就是窗口和其他对象的创建、处理和消亡过程。Windows中的消息的发送可以理解为一个窗口对象向别的窗口对象请求对象的服务过程。因此，用面向对象的方法来进行Windows程序的设计与开发是极其方便和自然的。

1.1.2　面向对象编程方法的基本特征

1. 封装性

封装性是面向对象的特征之一，是对象和类概念的主要特性。封装是把过程和数据包围起来，如果想对数据进行访问只能通过已定义的界面。面向对象设计基于这个基本概念，即现实世界可以被描绘成一系列完全自治、封装的对象，这些对象只能通过一个受保护的接口访问其他对象。一旦定义了一个对象的特性，则有必要决定这些特性的可见性，即哪些特性对外部是可见的、哪些特性用于表示内部状态。在这个阶段定义对象的接口。通常，应禁止直接访问一个对象的实际表示，而应通过操作接口访问对象，这称为信息隐藏。事实上，信息隐藏是用户对封装性的认识，封装性则为信息隐藏提供支持。封装保证了模块具有较好的独立性，使得程序维护、修改较为容易。对应用程序的修改仅限于类的内部，因而可以将应用程序修改带来的影响减少到最低限度。

2. 继承性

继承是一种联结类的层次模型，并且允许和鼓励类的重用，它提供了一种明确表述共性的方法。对象的一个新类可以从现有的类中派生，这个过程称为类继承。新类继承了原始类的特性，新类称为原始类的派生类（子类），而原始类称为新类的基类（父类）。派生类可以从它的基类那里继承方法和实例变量，并且可以修改或增加新的方法使之更适合特殊的需要。这也体现了大自然中一般与特殊的关系。继承性很好的解决了软件的可重用性问题。比如说，所有的Windows应用程序都有一个窗口，可以把它们看做都是从一个窗口类派生出来的；但是有的应用程序用于文字处理、有的应用程序用于绘图，这是由于派生出了不同的子类，各个子类添加了不同的特性。

3. 多态性

多态性是指允许不同类的对象对同一消息做出响应。比如同样的加法，把两个时间加在一起和把两个整数加在一起肯定完全不同。又比如，同样地执行编辑、粘贴操作，在字处理程序和绘图程序中会有不同的效果。

面向对象程序设计具有以下优点：

- 开发时间短、效率高、可靠性高、所开发的程序更强壮。由于面向对象编程的可重用性，可以在应用程序中大量采用成熟的类库，从而缩短开发时间。
- 应用程序更易于维护、更新和升级。继承性和封装性使得应用程序的修改带来的影

响更加局部化。

1.1.3 面向对象程序设计方法与结构化程序设计方法的比较

（1）传统的结构化程序设计方法以过程为中心构造应用程序，数据和处理数据的代码是分离的、相互独立的实体，设计出的程序可重用代码少，且当代码量增加时维护数据和代码的一致性困难。

（2）面向对象程序设计方法中，对象所具有的封装性和继承性使得代码重用成为可能，并大大减少了程序出错的可能性。

（3）面向对象程序设计方法吸收了结构化程序设计方法的优点，同时引入了新概念、新机制并建立了比传统方法更高层次的抽象。

面向对象程序设计同结构化程序设计相比最大的区别就在于：前者首先关心的是所要处理的数据，而后者首先关心的是功能。

1.2 类和对象

在面向对象程序设计中，类是面向对象程序设计的核心。类是具有相同操作功能和相同数据格式（属性）的对象的集合。类是对某一类对象的抽象；而对象是某一种类的实例，因此，类和对象是密切相关的。没有脱离对象的类，也没有不依赖于类的对象。对象和类的关系相当于一般的程序设计语言中变量和变量类型的关系。

1.2.1 类和对象的定义

1. C++类的定义

C++类的一般定义格式如下：

```
class  <类名>
{
public:
    公有成员函数或数据成员的说明;
protected:
    保护成员函数或数据成员的说明;
private:
    私有成员函数或数据成员的说明;
};                              //以上类的声明部分
<各个成员函数的实现>            //类的实现部分
```

其中，class是定义类的关键字；<类名>是由用户自己定义的类的名称，必须是C++的有效标识符，但一般首字母大写。

例1.1 下面给出一个矩形类定义的例子。

```
class Rect   //类名 Rect
{
```

```
    private:    //私有的数据成员
        int LeftX;
        int TopY;
        int RightX;
        int BottomY;
    public:     //公有的
      void init(int l,int t,int r, int b);        //初始化成员函数的说明
      void area( );
};
//类的实现部分
  void Rect::init(int l,int t,int r, int b)
  {
      LeftX=l;          //类的成员函数，可使用类的私有成员
       TopY =t;
      RightX=r;
      BottomY=b;       //初始化数据成员
  }   //无;
  int Rect::area( )
  {
    int s;
s= (RightX- LeftX)*( BottomY -TopY);        //使用类的私有成员
    return s;
  }
```

这里出现的作用域运算符“::”是用来标识某个成员函数是属于哪个类的。该类的定义还可以如下所示，将成员函数的定义直接写在类中：

```
class Rect  //类名 Rect
 {
    private:            //私有的数据成员
        int LeftX;
        int TopY;
        int RightX;
        int BottomY;
    public:             //公有的
     void init(int l,int t,int r, int b)
 {
      LeftX=l;          //类的成员函数，可使用类的私有成员
     TopY =t;
      RightX=r;
      BottomY=b;       //初始化数据成员
 }
      void area( )
        {
   int s;
s= (RightX- LeftX)*( BottomY -TopY);        //使用类的私有成员
    return s;
```

```
    }
};
```

从访问权限上来分，类的成员又分为：公有的（public）、私有的（private）和保护的（protected）三类。公有的成员用public来声明，公有部分往往是一些操作（即成员函数），它是提供给用户的接口功能，这部分成员可以在程序中引用。

在类的protected部分声明的数据成员和成员函数是不能在类之外存取的，只有类的成员函数及其子类（派生类）可以存取protected的成员。

私有的成员用private来声明，私有部分通常是一些数据成员，这些成员是用来描述该类中对象的属性的，用户是无法访问它们的，只有成员函数才可以引用它们，它们是被用来隐藏的部分。在Visual C++中常常将类的声明放在一个以类名为名称，后缀为.h的头文件中；各个成员函数的实现放在以类名为名称，后缀为.cpp的文件中。并且在.cpp文件的第一行写上#include“类名.h”。

定义类时，如果未指明成员是哪部分，则默认是属于private成员，但一般不要采用默认形式。

2. 对象的定义

对象是类的实例，它属于某个已知的类。因此，定义对象之前，一定要先定义好该对象的类。下面简单地介绍对象的定义。

定义对象的格式如下：

```
<类名>  <对象名表>
```

其中，<类名>是待定的对象所属的类的名称，即所定义的对象是该类类型的对象。<对象名表>中可以有一个或多个对象名，有多个对象名时用逗号分隔。<对象名表>中，可以是一般的对象名，也可以是指向对象的指针或引用，还可以是对象数组。如：

```
Rect  d1, d2, *p, d[31];
```

说明：

（1）一个对象的成员就是该对象的类所定义的成员。对象成员有数据成员和成员函数，其表示方式如下：

```
<对象名>.<成员名>
```

如：d1.area( )

（2）对象的指针的成员表示如下：

```
<对象指针名>→<成员名>
```

如：p→ area( )

1.2.2 类的成员函数及特性

构造函数和析构函数是在类中声明的两种特殊的成员函数。构造函数的功能是，在创

建对象时，使用给定的值将对象初始化。析构函数是用来从内存中释放一个对象的，在删除对象前，用它来做一些清理工作，它与构造函数的功能正好相反。下面举一例子来说明构造函数和析构函数的特点。

例1.2　演示构造函数和析构函数的特点。

```
#include<iostream.h>
class CPoint
{
private:
    int X, Y;
public:
    CPoint(int x,int y);          //构造函数
    CPoint( );                    //无参构造函数
    ~CPoint( );                   //析构函数
};
CPoint::CPoint(int x,int y)       //构造函数
{
X=x; Y=y;}
CPoint:: CPoint( )        //无参构造函数
{
X=0;Y=0;}
CPoint:: ~CPoint( )       //析构函数
{
  cout<<X<<","<<Y<<" 析构函数被调用"<<endl; }
void main( )
{
  CPoint P1(5,7);
  CPoint P2;
}
```

编译程序并运行可得到下面的输出：

```
0 , 0析构函数被调用
5 , 7析构函数被调用
```

CPoint类内声明的成员函数CPoint( )是构造函数，而~CPoint( )是析构函数。构造函数的特点如下：

（1）构造函数是成员函数，该函数的名称与类名相同。

（2）构造函数是一个特殊的函数，该函数无数据类型，也没有返回值。构造函数可以有一个参数，也可以有多个参数。

（3）构造函数可以重载，即可以定义多个参数个数不同的函数。

（4）程序中不能直接调用构造函数，在创建实例时系统自动调用构造函数。

（5）当类没有构造函数时，编译系统自动生成函数体为空的一个默认的构造函数，其形式如下：

```
    CPoint( )  //默认的构造函数
{
};
```

（6）构造函数的访问权限总是public。如果是private，则表示该类不能被实例化，这通常在只含有静态成员的类中。

析构函数的特点如下：

（1）析构函数是成员函数，函数体可定义在类体内，也可定义在类体外。

（2）析构函数也是一个特殊的函数，它的名称同类名，并在前面加“~”字符，用来与构造函数加以区别。析构函数不能有参数，也无数据类型。

（3）一个类中只可以定义一个析构函数。

（4）当撤销对象时，析构函数自动被调用。撤销对象的顺序与创建时的顺序正好相反。

（5）析构函数不能被继承和重载。

1.3 类的继承

在C++中，当一个类被其他类继承时，被继承的类称为基类（base class），继承其他类特性的类称为派生类（derived class）。

一个类可以从它的基类继承成员。继承意味着这个类隐含地包含除构造函数和析构函数以外的基类的所有成员。继承常用于在一个现有基类的基础上进行功能扩展，往往通过将几个类中相同的成员提取出来放在基类中，然后在各个派生类（子类）中加以继承来实现。

继承的重要特性如下：

• 继承是可传递的。如果C从B派生，并且B从A派生，那么C继承在B中声明的成员，同时也继承在A中声明的成员。

• 派生类可以对基类的功能进行扩展。即一个派生类可以添加自己的新成员，但是不能删除从基类继承的成员，只能不予使用。

1.3.1 继承的定义格式

继承的定义格式如下：

```
class  <派生类名>  ：<派生权限>  <基类名>
{
    派生类新增加的数据成员;
    派生类新增加的成员函数;
};
```

其中，<派生权限>常使用如下三种关键字：

private （默认的）	私有派生
public	公有派生
protected	保护派生

派生权限决定了继承的成员在派生类中的访问权限。其中使用public为最多，公有派生的继承权限规则如表1-1所示。

表1-1　公有派生的继承权限规则

基类成员的访问权限	继承后在派生类中的访问权限		
	公有派生	保护派生	私有派生
private	不可访问的权限	不可访问的权限	不可访问的权限
protected	protected	protected	private
public	public	protected	private

例1.3　从Box类公有派生ColorBox类。

```
class Box
   {
   private:
int width,height;
   public:
   void SetWidth(int w)
   {
         width=w;      }
   void SetHeight(int h)
   {
         height=h;     }
   };
class ColorBox:public Box      //public公有派生
{
   private:
   int color;
   public:
   void SetColor(int c)
   {color=c;      }
};
void main( )
{  ColorBox ob1;               //定义派生类的对象，分配内存，初始化数据成员
   ob1.SetWidth(1);
   ob1.SetHeight(2);
   ob1.SetColor(3);
}
```

现在的派生类ColorBox公有继承了基类Box类的全部数据成员（width、height）、成员函数（SetHeight、SetWidth）。添加自己的新成员——数据成员color和成员函数SetColor。如果在派生类ColorBox中的成员函数访问基类Box类的private数据成员（width、height），则出现错误，因为是不可访问的权限。