掌握 Visual C++ 编程

黄科 编著

CD-ROM

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

掌握 Visual C++编程

黄科　编著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 提 要

全书分为11章，深入浅出的介绍了Visual C++ 6.0的编程技术。本书的内容包括：面向对象技术和C++语言的一些重要特点；Visual C++ 6.0的集成开发环境； Windows的消息驱动机制；对话框、各种控件以及属性页；以及在视图上输出文字和图形；利用键盘、鼠标、定时器；MFC的层次类别以及一些常用类的使用；动态链接库、多线程编程、文档/视图结构以及利用socket进行网络编程。

除内容讲述比较详细以外，本书特色就是书中所讲述的内容大部分有一个详细的、接近实际软件开发过程的编程实例与之对应。通过学习这些编程实例，读者可以对所讲述的内容加深理解，从而帮助掌握VC编程技术。本书适合利用Visual C++进行开发的程序设计人员，以及Visual C++的初学者。

书　　名：掌握Visual C++编程
总 策 划：北京暴风雪科技有限公司
创　　作：黄科
电脑制作：北京暴风雪科技有限公司
发　　行：电子工业出版社 URL：http：//www.phei.com.cn
　　北京市海淀区万寿路173信箱　邮编 100036
印　　刷：北京安泰印刷厂
经　　销：各地新华书店
开　　本：787×1092　1/16　印张：25
版　　次：2001年9月第1版
本 版 号：ISBN 7-900074-74-0/TP·56
定　　价：39.00（含1CD）

凡购买电子工业出版社的图书和光盘有问题者，请向购买处调换。经销者请与本社联系。
电话：(010) 62645767、63962507

前 言

关于软件

Visuai C++6.0 是微软公司最新出品的功能最为强大的可视化开放工具，是计算机界公认的最优秀的应用开发工具之一。Microsoft 的基本类库 MFC 使得开发 Windows 应用程序比以往任何时候都要容易。本书的目的就是为了让你学会在 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。Visual C++作为一种程序设计语言，它同时也是一个集成开发工具，提供了软件代码自动生成和可视化的资源编辑功能。

关于配盘书

本书分为 11 章，由深入浅地介绍了 Visual C++的编程技术：第一章为简要介绍了面向对象技术和 C++语言的一些重要特点。第二章系统地介绍了 Visual C++的集成开发环境，包括各种开发的用户界面的功能和基本的使用，调试程序的一些技巧。第三章介绍了如何使用 Visual C++提供的程序框架生成工具来自动产生一个简单的应用程序，并对这个应用程序对各个文件进行了分析。第四章介绍了 Windows 程序的运行机制，包括 Windows 程序的特点，开发 Windows 的手段，并详细介绍了 Windows 的消息驱动机制及其使用。第五章介绍了基本的用户界面设计，主要是对话框、各种控件以及属性页，这一章涉及内容较多，需要多加练习。第六章介绍了一些基本的输入和输出接口，包括如何在视图上输出文字和图形，如何利用键盘、鼠标、定时器等设备进行输入。第七章介绍了 MFC 简要的层次类别，以期读者能对 MFC 有一个全盘的了解，此外这一章还介绍了一些常用类的使用。第八章介绍了动态链接库的理论及在 VC 中的编程使用。第九章介绍了多线程理论和 VC 中如何进行多线程编程。第十章介绍了文档/视图结构，这是 VC 程序的基本架构，此外还对对象序列化进行了介绍。第十一章介绍了利用 socket 进行网络编程的内容。

关于光盘

本书所配光盘包含了在本书中进行 Visual C++6.0 程序设计时所用的各个编程实例。这些编程实例涉及了 Visual C++6.0 的多个方面，它们适合读者进行学习而又不过于简单。这些编程实例贴近于实际的 Visual C++6.0 软件开发实战，通过对照书中内容学习这些编程实例，用户在使用 Visual C++6.0 开发应用程序方面会有一个较大的提高。

关于读者和作者

本书所面向的是那些有一定 C++语言程序设计基础并希望在 Windows 程序设计方面有一定深入发展的读者。虽然本书的第一章会介绍一些 C++语言的重要特点，但是这些介绍只是为了使读者对 C++语言有一个更深入的理解，而不是一个 C++教程。学习 Visual C++编程是一个相对比较长久而艰苦的过程，然而对于那些希望在程序设计、软件开发方面有一定的深入掌握的读者来说，学习 Visual C++编程是一个合适的选择。因为使用 Visual C++进行编程，特别是进行有一定难度的编程，会涉及到很多计算机的相关知识，比如面向对象技术、内存分配、线程交互、动态链接库、网络操作流程等。希望读者在学习 Visual C++编程的过程中必要的时候参考一些计算机科学领域的相关书籍，比如操作系统、数据结构、计算机网络基础等。相信通过一定的努力，读者，特别是初学者在对计算机编程乃至对计算机整个软件系统会有一个新的认识。

然而，学习 Visual C++编程也是一个充满乐趣的过程，通过对一个相对较难入门的编程工具的学习，读者会体会到其中收获的甘甜，掌握新技术的欣喜。除了书本以外，Internet 上很多的 VC 学习网站、讨论区以及 MSDN 等都是很好的参考资料。

最后一点，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很多的编程技术，而且，在编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，这种编程经验对开发者尤其显得不可或缺。

由于编者水平有限，书中难免有错，敬请广大读者批语指正。

暴风雪科技

2001 年 7 月

目 录

第 1 章 面向对象的程序设计及 C++语言的重要特点

本章主要讲述了面向对象的概念和 C++语言的一些重要特点，希望通过这些内容读者能加深对面向对象的程序设计，特别是 C++程序设计的认识。本章讲述的这些 C++语言的特点对于后续的 Visual C++程序设计很大的相关性。

1.1 面向对象的概念和设计

1.1.1 面向对象的概念及发展

面向对象技术是目前流行的系统设计开发技术，它包括面向对象分析和面向对象程序设计。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，即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变化，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可重用性不高。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。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. 对象的概念

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。关于对象这一概念，目前还没有统一的定义。一般的认为，对象是包含现实世界物体特征的抽象实体，它反映了系统为之保存信息和（或）与它交互的能力。它是一些属性及服务的一个封装体，在程序设计领域，可以用“对象=数据+作用于这些数据上的操作”这一公式来表达。

客观世界的问题都是由客观世界中的实体及其相互关系构成的，将客观世界中的实体抽象为问题空间中的对象（object）。由于研究的问题不同，面向的对象也就不同，因此对象是不固定的，一个人、一把椅子、一本书都可以是一个对象。下面就以椅子为例，对这一对象进行考察。椅子具有一些属性（attributes），如价格、尺寸、重量、位置和颜色等，这些属性的值代表这把椅子的状态，如价格为 100 元，颜色为白色等。对这把椅子可以用许多方式来操作，可以买、卖，进行物理变更（如把这把椅子漆成红色），或从一个地方搬到另一个地方。每一种操作（operation）或服务（service）或方法(method)会改变对象的一个或更多个属性的值。例如，若属性位置定义为一个组合项：

位置 = 经度值＋纬度值＋海平高度

那么一个名为“移动”的操作就会改变组成属性“位置”的数据项（经度值、纬度值、海平高度）中的一个或多个数据项的值。要完整、准确地描述一把椅子（chair），就必须定

义这些属性及属性之上的操作。椅子状态的改变只能通过这些操作来进行。换个角度说，对象椅子封装了数据（定义椅子的属性值）和操作（可用来改变椅子属性的动作）。

上面的例子可以帮助理解对象的概念，对象是一个封装了数据和操作的实体。对象的结构特征由属性表示，数据描述了对象的状态，操作可操纵私有数据（把数据称之为“私有”的，是因为认为数据是封装的在对象内部，是属于对象的）改变对象的状态。

对象不会无缘无故地执行某个操作，只有在接受别的对象的请求时，才会进行某一操作，这种请求对象执行某一操作或回答某些信息的要求称才会进行某一操作，这种请求对象执行某一操作或回答某些信息的要求称为消息（message），对象之间通过消息的传递来实现相互作用。

2. 类的概念

类是具有相同操作功能和相同的数据格式（属性）的对象的集合。类可以看作抽象数据类型的具体实现。抽象数据类型是数据类型抽象的表示形式。数据类型是指数据的集合和作用于其上的操作的集合，而抽象数据类型不关心操作实现的细节。从外部看，类型的行为可以用新定义的操作加以规定。类为对象集合的抽象，它规定了这些对象的公共属性和方法；对象为类的一个实例。苹果是一个类，而某一个具体的苹果则是一个对象。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。

还是以椅子为例，发现椅子、桌子、沙发等等对象都具有一些相同的特征，由于这些相同的特征它们可归类为家具。当你听别人说起一种不熟悉的东西时，如果别人告诉你它是一种家具，你一定就会对这样东西有一个大致的概念，即它是陈设在家中的一种用具。下一步你可能会要求别人描述一下这种家具的形状、尺寸、颜色、功能等等，因为你知道只要是一种家具，就应该具有这些属性。如果别人向你介绍的是一种饮料，你多半不会想了解这种饮料的尺寸，这时你想问的就是味道、包装、价格等等跟饮料相关的特性了。

家具与椅子之间的关系便是类与类的成员对象之间的关系。类是具有共同的属性、共同的操作的对象的集合。而单个对象则是对应类的一个成员，或称为实例（instance）。在描述一个类时定义了一组属性和操作，而这些属性和操作可被该类的成员所继承。也就是说，对象自动拥有它所属的类的属性和操作。正因为这一点，才会在知道一种物品是家具时，主动去询问它的形状、颜色、尺寸、功能等等属性。继承性（inheritance）是现代软件工程中的一个重要概念，软件的可重用性、程序成分的可重用性都是通过继承类中的属性和操作而实现的。因为重用就意味着利用已有的定义、设计和实现，如果缺少这种继承的手段是无法做到的。利用继承性，在定义一个新的对象时，只需指明它具有哪些类定义以外的新的特性，即说明其个性，而不必定义新对象的全部特性。这就大大减少重复定义，充分利用了前人的劳动成果，同时也使定义的系统的结构更加清晰、易于理解和维护。

类可以构成层次结构，相对上层的是超类（superclass），相对下层的是子类（subclass），例如，桌子可看作家具的子类，因为家具中包含了桌子、椅子、柜子等成员。子类在继承超类的属性和操作的同时可以拥有自己的特有的属性和操作。

继承性是面向对象方法的一个主要特征，另一个主要特征是封装性（encapsulation）。把对象定义为封装了数据和操作的实体，含义是将对象的各种独立的外部特征与内部细节分开，亦即对象的具体数据结构和各种操作实现的细节对于对象外的一切是隐藏的。当想知道一把椅子的重量时，只需向这个对象发一条得知重量的请求，然后获知其重量，这便是对对象外

部所需做的一切，完全不必了解给椅子称重的具体细节。对象将其实现细节隐藏在其内部，因此无论是对象功能的完善扩充，还是对象实现的修改，影响仅限于该对象内部，而不会对外界产生影响，这就保证了面向对象软件的可构造性和易维护性。

可以将面向对象方法的基本要点概括为以下四点：

- 数据的抽象，即类与子类的概念及相互关系；
- 数据以及对它的操作的一体化，即封闭的概念与方法；
- 属性与操作由父类向子类传递，即继承的概念与方法；
- 客观事物之间的相互关系用统一的、消息传递的方法来描述。

1.1.2 深入理解面向对象技术

进行面向对象的软件设计，应该是将所要解决的客观世界的问题抽象为具体的对象，即进行数据抽象。数据抽象为程序员提供了一种较高级的对数据和为操作数据所需要的算法的抽象。用户的视图是一系列的操作，这些操作集中在一起定义了这个抽象的行为；用户不必知道内部实现细节。数据抽象实际上包含了两个独立的但又密切相关的概念：模块化和信息隐藏。

模块化指的是将一个复杂系统分解为几个自包含的实体（即模块），与系统中一个特定的实体有关的信息保存在该模块内。这样一个模块是对整个系统结构某一部分的一个自包含的和完整的描述。用计算机的术语来说，这表示一个模块将包含为实现系统的这个部分所需要的所有数据结构和算法。模块化的优点是，当需要进行修改或出现问题时，可以立即确定需要在哪些模块上进行。模块化强调了这样一种设计方法：程序员将问题域分解为几个可识别的概念实体。这种设计方法构成面向对象计算的本质。

信息隐藏通过将一个模块的细节对用户隐藏起来将抽象的级别向前推进了一步。使用信息隐藏，用户必须通过一个受保护的接口访问一个实体，而不能直接访问诸如数据结构或局部过程等这些内部细节。这个接口一般由一些操作组成，正如前面所言，这些操作定义了一个实体的行为。在处理复杂性中，这是一个主要工具，因为它允许系统在某层上进行抽象。通过严格控制一个模块的入口点，信息隐藏也支持开发更可靠的程序。由于不能直接访问数据结构，因此，也就不可能对数据执行不期望的操作。访问数据结构的唯一途径是通过操作接口。对操作接口的完全测试为一个模块的正确性提供了高的置信度。类似地，在支持信息隐藏的系统中，错误的影响通常被限制在一个模块内。因此，就没有可能使一个模块破坏另一个模块，这增强了一个系统的可靠性。

在实践中，一个实体的行为可以用两种方式来表示：使用过程接口抽象地表示，或使用接口和相应的实现具体地表示。面向对象的计算在两种情况中都使用术语行为。在后面的章节中将看到，将抽象的规范说明与实现相分离，以更抽象的形式解释行为是有益的。

数据抽象一般认为是迈向更加结构化的程序设计的一个重大步骤。数据抽象的重要性在于，它提供了面向对象计算的始点，即系统应该被分解为概念上的实体，实体的内部细节应该被隐藏起来。由模块化和信息隐藏这个过程所提供的抽象是面向对象方法的核心，实现面向对象方法的首要一点是提供技术支持数据抽象。

1.1.3 面向对象在 C++/VC 中的体现

Visual C++是美国 Microsoft 公司推出的第四代软件开发工具，目前已成为国内应用最广泛的高级程序设计语言之一，最新版本为 6.0 版。C++全面支持类和对象，Visual C++更是以类作为基础，其基础是一个类库 MFC（Microsoft Foudation Class，微软基础类）。在 Visual C++的世界里，基本的而且最广泛应用的单位就是类。可以说，Visual C++是一个 C++类库加上一个 Windows 消息机制。同其他软件开发工具相比，Visual C++具有以下优点：

1. 面向对象、可视化开发：提供了面向对象的应用程序框架 MFC（Microsoft Foundation Class:微软基础类库），大大简化了程序员的编程工作，提高了模块的可重用性。Visual C++还提供了基于 CASE 技术的可视化软件自动生成和维护工具 AppWizard、ClassWizard、Visual Studio、WizardBar 等，帮助用户直观、可视地设计程序的用户界面，可以方便的编写和管理各种类，维护程序源代码，从而提高了开发效率。用户可以简单而容易地使用 C/C++编程。

2. 众多的开发商支持以及业已成为工业标准的 MFC 类库：MFC 类库已经成为事实上的工业标准类库，得到了众多开发商和软件开发工具的支持；另外，由于众多的开发商都采用 Visual C++进行软件开发，这样用 Visual C++开发的程序就与别的应用软件有许多相似之处，易于学习和使用。

3. Visual C++封装了 Windows 的 API（应用程序接口）函数、USER、KERNEL、GDI 函数，帮助弄清了许多函数的组织方法，隐去了创建、维护窗口的许多复杂的例行工作，简化了编程。

1.2 类的基本特征：封装性

对象有两个主要的组成部分，一个是属性（property），另一个是方法（method）。在程序设计领域，属性通常也被称为成员变量（member variable），方法通常也被称为成员函数（member function）。一般情况下，成员变量由成员函数所操纵。

如果用 CCircle 代表“圆”这个类别，那么圆有直径（diam），圆可以显示（display）出来。那么，diam 就是一个成员变量，display 就是一个成员函数。描述如下：

```
CCircle circle;     //定义 circle 是一个圆
circle.diam = 10;   //设置成员变量的值，直径为 10
circle.display();   // 调用成员函数，显示圆
```

下面是 C++语言对于 CCircle 的描述：

```
class CCircle    // 通常以 C 作为类别名称的前缀
{
private:
    int m_nDiam;  // 通常以 m_作为成员变量的前缀

public:
    void display()      // 成员函数 display 的定义
```

```
    {...};    //成员函数 display 的实现
    void setDiam(int nDiam)      //设置圆的直径
    {
        m_nDiam = nDiam;
    };
    int GetDiam() // 获取圆的直径
    {
        return m_nDiam;
    };
};
```

成员变量通常在类别之内被处理，也可以暴露给类别以外的函数、类别处理。从封装性的目的来说，不希望将成员变量暴露给外部的函数、类别。但是，在一些具体的情况下，又不得不对外开方成员变量。所以，C++提供了 private、public、protected 三种限制访问的修饰词。定义为 private 的成员变量或是成员函数只能由本类别内部的成员函数进行访问，定义为 public 的成员变量和成员函数，任何外部函数或类别均可访问，定义为 protected 的成员变量和成员类别只有在本类别的子类中可以被访问。当把成员变量定义为 private 以后，外部函数和类别便不能够随意访问，只能通过类别提供的界面来访问，这就是面向对象的“封装性”。比如在 CCircle 的例子中，就定义了 SetDiam(int)成员函数来设定圆的直径。

封装性的一个明显好处是简化。程序员可以将对象和外部的交互，只通过有限的界面和外部交互。封装简化了程序员对于对象的使用，只需要知道输入什么和输出什么，而不用关心对象内部是如何操作的。同时封装也使得对象作为一个独立的部件可以应用在不同的应用程序当中而不用担心功能受到影响，这些都使得软件设计和维护更为方便，同时也使得软件质量得到了保证。

1.3 基础类别的衍生：类的继承

其实在一些非面向对象的程序设计语言中也可以实现封装性，比如在 C 语言中的利用结构类型（struct）和数据处理的函数指针（function pointer），也可以实现一定的封装性。

C++语言所特有的性质是其继承性。在现实生活中，可以看到大量层次类别的例子。比如，苹果是水果，梨也是水果，葡萄也是水果；猪是动物，狗是动物，狼也是动物。那么可以把性质相似的东西的共同的性质抽出来，构成一个基础类别，然后再从它衍生出派生类别。举一个简单的例子，可以对生物界给出如图 1-1 所示的类别层次图。图中直线箭头指向父类（在不同的系统中，这一点有区别，有些系统中箭头是由父类指向子类）。

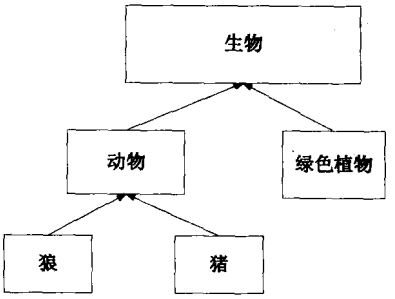

图 1-1 生物类层次

可以看出，派生出来的类别和基础类别之间的关系是一种“是一种”（Is Kind Of）的关系，派生类别有基类的所有特征，还有其区别于基类的更多的特征，其中可能包括新的成员变量和成员函数。

可以用 C++来描述以上类层次图的部分类别：

```
Class CLife   // 生物
{
private:
    int  m_nAge;
public:
    void SetAge(int nAge)
    {
        m_nAge = nAge;
    };
};

class CAnimal : public CLife// 动物是一种生物
{
public:
    void Move()        // 动物会移动
    {...};
};

class CGreenPlant : public CLife // 绿色植物是一种生物
{
public:
    GrowLeaves()            // 绿色植物会长叶子
    {...}    ;
};

class CPig : public CAnimal // 猪是一种动物
```

```
{
public:
    void Move()
    {...};
};

class CWolf : public CAnimal// 狼是一种动物
{
public:
        void Move()
    {...};
    void bark()
    {…};
};
```

有了以上的定义，可以这样来调用：

```
CWolf wolf1;  //定义一个 CWolf 对象
CWolf wolf2;  //定义另一个 CWolf 对象
CPig pig;     //定义一个 CPig 对象

wolf1.SetAge(5);   // 调用 CWolf::SetAge()
wolf1.Move();      // 调用 CWolf::Move()
wolf2.SetAge(4);   // 调用 CWolf::SetAge()
wolf2.Move();      // 调用 CWolf::Move()
pig.SetAge(3);     // 调用 CPig::SetAge()
pig.Move();        // 调用 CPig::SetAge()
```

对上述例子，需要说明的是，通过继承，派生类隐含地具有了基类所定义的成员变量和成员方法，比如上例中即是 m_nAge 和 SetAge(int nAge)。所以对于类 CWolf，其定义实际上相当于：

```
class CWolf
{
private:
    int m_nAge;
public:
    void SetAge(int nAge);
    int GetAge();
    void Move();
```