高等学校计算机规划教材

C++程序设计基础

幸莉仙 主编
于海泳 王立军 田志刚 王华英 编著

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

高等学校计算机规划教材

C++程序设计基础

幸莉仙　主编

于海泳　王立军　田志刚　王华英　编著

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

内 容 简 介

本书针对初学者学习程序设计而编写，通过本书的学习，初学者可以较好地掌握结构化程序设计的3种结构、面向对象的概念和编程思想。本书以VC++ 2005为开发平台，结合大量实例，系统本书介绍VC++ 2005的开发环境、基本语法和编程技巧。全书共11章：C++与VC++ 2005概述，VC++ 2005程序设计基础，流程控制语句，数组和字符串，指针，函数，结构体与联合，类与对象，类的继承、派生与多态，C++流与文件操作，VC++ 2005应用程序开发实例。本书配有电子课件、源代码等教学资源。

本书可作为普通高等学校C++程序设计的教学用书，也可作为计算机等级考试的培训教材和VC++ 2005的自学用书。

图书在版编目(CIP)数据

C++程序设计基础 / 幸莉仙主编. —北京：电子工业出版社，2011.1
高等学校计算机规划教材
ISBN 978-7-121-12226-2

Ⅰ. ①C… Ⅱ. ①幸… Ⅲ. ①C语言－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2010)第217099号

策划编辑：史鹏举
责任编辑：史鹏举
印　　刷：北京丰源印刷厂
装　　订：三河市鹏成印业有限公司
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
开　　本：787×1092　1/16　　印张：16　　字数：410千字
印　　次：2011年1月第1次印刷
定　　价：28.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010)88254888。

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。

服务热线：(010)88258888。

前　言

网络时代学习网络版的软件开发工具非常必要。“短视近利，够用就好”不是学习者应有的态度和方法，未来对软件开发环境有太多的需求和挑战，本书采用 Visual Studio 2005.NET 作为学习 C++的语言环境正是为了贴近人才市场对学生能力和技能需要。

C++语言是当前最流行、最实用的一种计算机高级语言，尽管国内大多数高校都把 C++设定为新生计算机编程知识入门类课程，但长期以来却在讲授单机版的 Turbo C 或 VC++ 6.0，学生学完后，不会使用目前流行的.NET 开发工具编写应用程序。本书将面向 Visual Studio 2005.NET 的初中级用户，系统地介绍 C++程序开发的基础知识、编程方法和技巧。

本书经过精心规划，具有以下特点：

1．从 VC++ 2005 最基本的数据类型、概念、语法、简单程序编写开始入手，使读者逐步掌握结构化程序设计的三种基本结构，即顺序结构、选择结构、循环结构；通过介绍面向对象的概念以及在 VC++ 2005 环境下的实现，使读者从基本概念、基础操作的学习上升到对理论的理解，从而领会应用程序开发的实质。

2．在例题选择上秉承由浅入深、由简到繁的编程规律，对典型例题给出多种算法求解，在习题上力求做到多样化，以培养和提高初学者分析问题和解决问题的能力。

3．VC++ 2005 语言系统庞大，知识点前后衔接紧密，为使初学者轻松学习本门课程，掌握程序设计的精髓，本书将所有知识点按章节划分成一个有序的线性结构，内容由易到难、循序渐进。

4．为使初学者在学习完本课程后能编写出完成的 Windows 应用程序，在第11章介绍了一个完整的应用程序开发实例。

5．所有例题、习题、函数等程序都在 VC++ 2005 环境下测试通过。

6．附录给出了编程中常用的库函数(包括数学函数、字符串函数、常用数学函数的反函数等)，以及习题答案，以方便读者学习时使用。

本书共分 11 章，依次是：C++与 VC++ 2005 概述，VC++ 2005 程序设计基础，流程控制语句，数组和字符串，指针，函数，结构体与联合，类与对象，类的继承、派生与多态，C++流与文件操作，VC++ 2005 应用程序开发实例。各章之间内容衔接紧密、自然，形成了一个完整的学习体系。

本书配有电子课件、源代码等教学资源，需要者可登录华信教育资源网 http://www.hxedu.com.cn，免费注册下载。

本书由幸莉仙策划和统稿，由幸莉仙、于海泳、王立军、田志刚、王华英共同撰写完成。韩玢、黄慧莲对本书进行了排版和校对工作。

由于时间仓促，书中难免有一些疏漏和不足，恳请广大读者和同行不吝赐教，以便及时修订和补充。

联系方式：0312-7525111。E-mail：xlxwhy@sina.com.cn

编　者

目　录

第 1 章　C++与 VC++ 2005 概述

1.1　计算机程序设计语言的发展

自 1946 年第一台电子计算机问世以来，计算机已被广泛地应用于生产、生活的各个领域，推动着社会的进步与发展。特别是 Internet 出现后，传统的信息收集、传输及交换方式发生了革命性的改变。

计算机科学的发展依赖于计算机硬件和软件技术的发展，硬件是计算机的躯体，软件是计算机的灵魂。没有软件，计算机只是一台“裸机”，什么也不能干；有了软件，计算机才有“思想”，才能做相应的事。软件是用计算机程序设计语言编写的。计算机程序设计语言的发展经历了从机器语言、汇编语言到高级语言的历程，如图1-1所示。

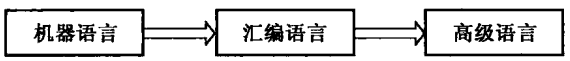

图 1-1　计算机语言发展历程

1.1.1　机器语言

计算机使用的是由“0”和“1”组成的二进制数，二进制编码方式是计算机语言的基础。计算机发明之初，科学家只能用二进制数编制的指令控制计算机运行。每一条计算机指令均由一组“0”、“1”数字按一定的规则排列组成，若要计算机执行一项简单的任务，需要编写大量的这种指令。这种有规则的二进制数组成的指令集，就是机器语言(Machine Language)(也称指令系统)。不同系列的 CPU，具有不同的机器语言，如目前个人计算机中常用 AMD 公司的 CPU 系列和 Intel 公司的 CPU 系列，具有不同的机器语言。

机器语言是计算机唯一能识别并直接执行的语言，与汇编语言或高级语言相比，其执行效率高。但其可读性差，不易记忆；编写程序既难又繁，容易出错；程序调试和修改难度巨大，不容易掌握和使用。此外，因为机器语言直接依赖于中央处理器，所以用某种机器语言编写的程序只能在相应的计算机上执行，无法在其他型号的计算机上执行，也就是说，可移植性差。

1.1.2　汇编语言

为了减轻使用机器语言编程的痛苦，20 世纪 50 年代初，出现了汇编语言(Assemble Language)。汇编语言用比较容易识别、记忆的助记符替代特定的二进制串。下面是几条 Intel 80x86 的汇编指令：

```
ADD AX, BX;      表示将寄存器 AX 和 BX 中的内容相加，结果保存在寄存器 AX 中。
SUB AX, NUM;     表示将寄存器 AX 中的内容减去 NUM，结果保存在寄存器 AX 中。
MOV AX, NUM;     表示把数 NUM 保存在寄存器 AX 中。
```

通过这种助记符，人们就能较容易地读懂程序，调试和维护也更方便了。但这些助记符号计算机无法识别，需要一个专门的程序将其翻译成机器语言，这种翻译程序称为汇编程序。

汇编语言的一条汇编指令对应一条机器指令，与机器语言性质上是一样的，只是表示方式做了改进，其可移植性与机器语言一样不好。总之，汇编语言是符号化的机器语言，执行效率仍接近于机器语言，因此，汇编语言至今仍是一种常用的软件开发工具。

1.1.3 高级语言

尽管汇编语言比机器语言方便，但汇编语言仍然具有许多不便之处，程序编写的效率远远不能满足需要。1954 年，第一个高级语言 FORTRAN 问世了。高级语言(High-level Language)是一种用能表达各种意义的“词”和“数学公式”按一定的“语法规则”编写程序的语言，也称为高级程序设计语言或算法语言。半个多世纪以来，有几百种高级语言问世，影响较大、使用较普遍的有 FORTRAN、A LGOL、COBOL、BASIC、LISP、SNOBOL、PL/1、Pascal、C、PROLOG、Ada、C++、Visual C++、Visual Basic 、Delphi、Java、C#等。高级语言的发展也经历了从早期语言到结构化程序设计语言、面向对象程序设计语言的过程。

高级语言与自然语言和数学表达式相当接近，不依赖于计算机型号，通用性较好。高级语言的使用，大大提高了程序编写的效率和程序的可读性。

与汇编语言一样，计算机无法直接识别和执行高级语言，必须翻译成等价的机器语言程序(称为目标程序)才能执行。高级语言源程序翻译成机器语言程序的方法有“解释”和“编译”两种。解释方法采用边解释边执行的方法，如早期的 BASIC 语言即采用解释方法，在执行 BASIC 源程序时，解释一条 BASIC 语句，执行一条语句。编译方法采用相应语言的编译程序，先把源程序编译成指定机型的机器语言目标程序，然后再把目标程序和各种标准库函数连接装配成完整的目标程序，在相应的机型上执行。如 C、C++、Visual C++及 Visual Basic 等均采用编译的方法。编译方法比解释方法更有效率。

1.1.4 结构化程序设计语言

高级语言编写程序的编写效率虽然比汇编语言高，但随着计算机硬件技术的日益发展，人们对大型、复杂的软件需求量剧增，同时因缺乏科学规范、系统规划与测试，程序含有过多错误而无法使用，甚至带来巨大损失。20 世纪 60 年代中后期“软件危机”的爆发，使人们认识到大型程序的编制不同于小程序。“软件危机”的解决一方面需要对程序设计方法、程序的正确性和软件的可靠性等问题进行深入研究，另一方面需要对软件的编制、测试、维护和管理方法进行深入研究。

1968 年，E. W. Dijkstra 首先提出“GOTO 语句有害”论点，引起了人们对程序设计方法讨论的普遍重视。程序设计方法学在这场讨论中逐渐产生和形成。程序设计方法学是一门讨论程序性质、设计理论和方法的学科。它包含的内容比较丰富，如结构化程序设计、程序的正确性证明、程序变换、程序的形式说明与推导，以及自动程序设计等。

在程序设计方法学中，结构化程序设计(Structural Programming)占有重要的地位，可以说，程序设计方法学是在结构化程序设计的基础上逐步发展和完善的。结构化程序设计是一种程序设计的原则和方法，它讨论了如何避免使用 GOTO 语句；如何将大规模、复杂的流程图转换成一种标准的形式，使得它们能够用几种标准的控制结构(顺序、分支和循环)通过重复和嵌套来表示。结构化程序设计思想采用“自顶向下、逐步求精”的方法，避免被具体的细节所缠绕，降低难度，直到恰当的时机才考虑实现的细节，从而有效地将复杂的程序系统设计任务分解成许多易于控制和处理的子程序，便于开发和维护。

按结构化程序设计的要求设计出的高级程序设计语言称为结构化程序设计语言。利用结构化程序设计语言，或者说按结构化程序设计思想编写出来的程序，称为结构化程序。结构化程序具有结构清晰、容易理解、容易修改、容易验证等特点。

到了 20 世纪 70 年代末期，随着计算机应用领域的不断扩大，对软件技术的要求越来越高，结构化程序设计语言和结构化程序设计方法也无法满足用户需求的变化，其缺点日益显露出来：

(1) 代码的可重用性差。随着软件规模的逐渐庞大，代码重用成了提高程序设计效率的关键；但采用传统的结构化设计模式，程序员每进行一个新系统的开发，几乎都要从零开始，这中间需要做大量重复、繁琐的工作。

(2) 可维护性差。结构化程序是由大量的过程(函数、子程序)组成的；随着软件规模逐渐庞大，程序变得越来越复杂，过程(函数、子程序)越来越多，相互间的耦合越来越高，它们变得难以管理；当某个业务有所变化时必须对大量的程序进行修改和调试。

(3) 稳定性差。结构化程序要求模块独立，并通过过程(函数、子程序)的概念来实现。但这一概念狭隘、稳定性有限，在大型软件开发过程中，数据的不一致性问题仍然存在。

(4) 难以实现。在结构化程序中，代码和数据是分离的。例如，在 C 语言中，代码单位为函数，而数据单位称为结构，函数和结构没有结合在一起。然而，函数和数据结构并不能充分地模拟现实世界。例如，当考虑会计部门的应用程序时，应该考虑下列内容：

① 出纳支付工资。

② 职工出具凭证。

③ 财务主管批准支付。

④ 出纳记账。

但实际应用中，要决定如何通过数据结构、变量和函数来实现这个应用程序却是很困难的。

1.1.5　面向对象语言的产生

结构化程序设计方法与语言是面向过程的，存在较多的缺点，同时程序的执行是流水线式的，在一个模块被执行完成前不能干别的事，也无法动态地改变程序的执行方向。这和人们日常认识、处理事物的方式不一致。人们认为客观世界是由各种各样的对象(或称实体、事物)组成的；每个对象都有自己的内部状态和运动规律，不同对象间的相互联系和相互作用构成了各种不同的系统，进而构成整个客观世界；计算机软件主要是为了模拟现实世界中的不同系统，如物流系统、银行系统、图书管理系统、教学管理系统等。因此，计算机软件可以认为是，现实世界中相互联系的对象所组成的系统在计算机中的模拟实现。

为了使计算机更易于模拟现实世界，1967 年挪威计算中心的 Kisten.Nygaard 和 Ole.Johan Dahl 开发了 Simula67 语言，它提供了比子程序更高一级的抽象和封装，引入了数据抽象和类的概念，被认为是第一个面向对象(Object Oriented)程序设计语言。20 世纪 70 年代初，Palo Alto 研究中心的 Alan Kay 所在的研究小组开发出了 Smalltalk 语言，之后又开发出了 Smalltalk-80。Smalltalk-80 被认为是最纯正的面向对象语言，它对后来出现的面向对象语言，如 Object-C、C++、Java、Self、Eiffl 产生了深远的影响。

随着面向对象语言的出现，面向对象程序设计方法也应运而生且得到迅速发展，面向对象的思想也不断向其他方面渗透。1980 年 Grady Booch 提出了面向对象设计的概念，之后面向对象分析的概念也被提出。1990 年以来，面向对象分析、测试、度量和管理等研究得到了

长足的发展，并在全世界掀起了一股面向对象热潮，至今盛行不衰。面向对象程序设计在软件开发领域掀起了巨大的变革，极大地提高了软件开发效率。

1.2 C++语言与面向对象程序设计

1.2.1 C++概述

当 C 语言程序代码达到 25 000 行以上后，维护和修改工作变得相当困难。为了满足管理程序复杂性的需要，美国贝尔实验室的 Bjarne Stroustrup 博士于 1979 年开始对 C 语言进行了改进和扩充，并引入了面向对象程序设计的内容，1983 年命名为 C++，后经过三次重大修订，于 1994 年制定了标准 C++草案，之后经过不断完善，成为目前的 C++。

C++提出了一些更为深入的概念，它所支持的这些面向对象的概念容易将问题空间直接地映射到程序空间，为程序员提供了一种与传统结构程序设计不同的思维方式和编程方法。因而也增加了整个语言的复杂性，掌握起来有一定难度。

C++具有以下特点：

(1) 保持了与 C 语言的兼容性。绝大多数 C 语言程序不经修改可以直接在 C++环境中运行。

(2) 支持面向过程的程序设计。它是一种理想的结构化程序设计语言，又包含了面向对象程序设计的特征。C++由两部分组成：一是过程性语言部分，与 C 语言无本质区别；二是类和对象部分，是面向对象程序设计的主体。

(3) 具有程序效率高、灵活性强的特点。C++使程序结构清晰、易于扩展、易于维护而不失效率。

(4) 具有通用性和可移植性。C++是一种标准化的、与硬件基本无关的程序设计语言，C++程序通常无须修改或稍许修改便可在其他计算机上运行。

(5) 具有丰富的数据类型和运算符，并提供了强大的库函数。

(6) 具有面向对象的特性，C++支持抽象性、封装性、继承性和多态性。

1.2.2 面向对象程序设计

面向过程程序设计缺点的根源在于数据与数据处理分离，而面向对象程序设计(Object Oriented Programming)方法正是克服这个缺点，同时吸纳结构化设计思想的合理部分而发展起来的，这两种设计思想并非对立关系。面向对象设计思想模拟自然界认识和处理事物的方法，将数据和对数据的操作方法放在一起，形成一个相对独立的整体——对象(Object)，对同类型对象抽象出共性，形成类(Class)。任何一个类中的数据都只能用本类自有的方法进行处理，并通过简单的接口与外部联系。对象之间通过消息(Message)进行通信。以下就面向对象程序设计中的基本概念、面向对象软件开发方法和面向对象程序设计的特点做简单介绍。

(1) 对象

在自然界中，对象是一个常见概念，一般将所面对的事物看做具有某些属性和行为(或操作)的对象。例如，手表是一个对象，它具有的属性包括表针、旋钮及复杂的机械结构等，行为包括调节旋钮这样的操作。其中调节旋钮提供了对外的接口。在手表之外，只能通过这个接口对对象进行操作，而不能直接调整其内部的机械零件。在面向对象程序设计中，同样

使用对象的概念描述问题和事物，但更加抽象，含义的范围也更加广泛，可包括有形实体、图形，甚至抽象概念，如程序设计中的窗口、单击鼠标这样的事件等，都可以抽象成为一个对象。对象是组成程序系统的基本单位。

(2) 类

人类在认识事物时，通常是分类进行的。通过抽象的方法，从一些对象中概括出此类对象共有的静态特征(属性)和动态特征(行为)，形成类。类是一个抽象的概念，用来描述这类对象所共有的、本质的属性和行为。任何一个对象都是这个类的一个具体实现，称为实例(instance)。同类对象之间具有相同的属性和行为。类和对象之间的关系正如手表和一块具体的手表之间的关系。手表只是个概念，这个概念描述了所有手表共有的属性(包括表针、旋钮、内部结构)和行为(调节旋钮)，而一块具体的手表则是一个实在的对象。两块手表可能外形不同，但都具有手表所共有的属性和操作。

面向对象程序设计也使用分类抽象的方法，通过对一类对象的抽象形成“类”。在程序设计中，“类”表现为一种用户定义的数据类型，用这种数据类型描述该类所具有的属性和行为，其中属性用数据进行描述，而行为用一系列函数描述，也称操作。例如，定义一个矩形类，它的数据包括矩形的顶点坐标，而方法可包含以下函数：移动矩形位置、扩大、缩小等。如用这个矩形类定义两个矩形对象，这两个矩形对象就是同类对象，它们都用顶点坐标来描述，都可以进行移动、扩大和缩小等操作。

(3) 消息

自然界是由各种各样的对象组成的，这些对象之间通过信息传递产生相互作用，构成富有生机的世界。人类将对象之间产生相互作用所传递的信息称做消息。例如，汽车和人是两个对象，人起动汽车，就是向汽车发送消息，转动方向盘，也是发送消息，其中转动的角度是消息中的参数。汽车接收到消息后，按照消息及其参数执行相应的操作。

在面向对象设计的程序中，对象之间的相互作用也是通过消息机制实现的。

(4) 面向对象的软件开发方法

在面向对象程序设计发展的早期，软件业界主要集中于研究面向对象的编程(OOP)，但对于大型软件开发过程，编程只是其中一个很小的部分。面向对象方法的根本合理性在于它符合客观世界的组成方式和大脑的思维方式，因此面向对象的思想方法应贯穿软件开发的全过程，这就是面向对象的软件工程。

面向对象的软件工程同样遵循分层抽象、逐步细化的原则，软件开发过程包括面向对象的分析(OOA, Object Oriented Analysis)、面向对象的设计(OOD, Object Oriented Design)、面向对象的编程(OOP, Object Oriental Programming)、面向对象的测试(OOT, Object Oriented Test)、面向对象的维护(OOSM, Object Oriented Soft Maintenance)五个阶段。

分析阶段的主要任务是按照面向对象的概念和方法，从问题中识别出有意义的对象，以及对象的属性、行为和对象间的通信，进而抽象出类结构，最终将它们描述出来，形成一个需求模型，由于系统的复杂性，这个模型一般只能反映用户对系统主体部分的需求。

设计阶段从需求模型出发，分别进行类的设计和应用程序的设计。类的设计需要应用分层抽象的方法，而应用程序设计是根据已设计好的类来构造满足要求的应用程序。

编程阶段实现由设计表示到面向对象程序设计语言描述的转换。

测试的任务在于发现并改正程序中的错误。面向对象程序设计中，类是程序的基本单元，因此也是测试的基本单元。经过测试后的程序进入运行维护期，即投入使用。

(5) 面向对象程序设计的特点

面向对象程序设计中，对象是程序的基本单元。从类和对象的概念及面向对象设计方法所提供的支持看，这种设计方法具有以下几个特点及相应的优点。

封装性(Ecapsulation)：对象是一个封装体，在其中封装了该对象所具有的属性和操作。对象作为独立的基本单元，实现了将数据和数据处理相结合的思想。此外，封装特性还体现在可以限制对象中数据和操作的访问权限，从而将属性“隐藏”在对象内部，对外只呈现一定的外部特性和功能。手表是一个典型的例子，大量的零件和动作被封装在外壳中，并被隐藏起来，提供给我们的只能是读表盘和旋旋钮。同样，可以将一个对象中的数据隐藏起来，而方法定义为开放，那么在这个类之外，不能直接引用其中的数据，只能通过方法达到间接使用数据的目的。就好比不能直接去拨驱动表针的内部结构，只能通过旋旋钮实现一样。

封装性增加了对象的独立性，C++通过建立数据类型——类，来支持封装和数据隐藏。一个定义完好的类一旦建立，就可看成完全的封装体，作为一个整体单元使用，用户不需要知道这个类是如何工作的，而只需要知道如何使用就行。另一方面，封装增加了数据的可靠性，保护类中的数据不被类以外的程序随意使用。这两个优点十分有利于程序的调试和维护。

继承(Inheritance)和派生性：以汽车为例，如果已经定义了汽车类，现在需要定义小汽车，通常我们不会重复描述属于汽车的那些共有特征，而是在继承汽车类特性的基础上，描述出属于小汽车的新的特征。于是称小汽车继承了汽车，也可以称是由汽车派生出来的。面向对象程序设计提供了类似的机制。当定义了一个类后，又需定义一个新类，这个新类与原来类相比，只是增加了或修改了部分属性和操作，这样在定义新类时只需说明新类继承原来类，然后描述出新类所特有的属性和操作即可。此时称原来类为基类，由它派生出来的类称为子类或派生类。由基类派生出子类，子类还可继续派生它的子类，如此下去，可以形成树状派生关系，称为派生树或继承树；如图1-2所示(注意箭头指向基类)。

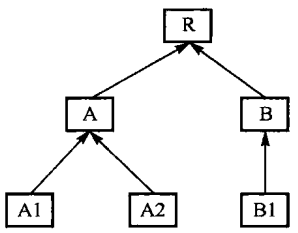

图 1-2 类的继承关系

继承性可以简化人们对问题的认识和描述，同时还可以在开发新程序和修改源程序时最大限度利用已有的程序，提高了程序的可重用性，从而提高了程序修改、扩充和设计的效率。

多态性(Polymorphism)：多态性是指同样一个消息，被不同对象接收时，产生不同的结果。系统提供的这种机制主要用在具有继承关系的类体系中。一个类体系中的不同对象可以用不同方式响应同一消息，并产生不同结果，即实现“同一接口，多种方法”。例如，定义了中学生类，再由中学生派生出大学生类。对于“计算平均成绩”这样一个消息，对中学生对象，计算的是语文、数学、英语等课程；而对大学生对象，则计算高等数学、英语、线性代数等课程。

继承和多态性的组合，可以很容易地生成一系列虽然类似但独一无二的对象。继承性使这些对象共享许多相似特征，而多态性使同样的操作对不同的对象有不同的表现方式。这样既提高了程序的灵活性，又减轻了分别逐个设计的负担。

面向对象程序设计着眼点是对象，程序设计的核心是从问题中抽象出合适的对象，即首先解决“做什么”的问题。至于“怎么做”，则封装在对象内部。而对于操作方法的设计，核心仍然是算法的设计，完全吸收了结构化程序设计的思想。

1.3　C++集成开发环境 Visual Studio 2005

1.3.1　集成开发环境 IDE

集成开发环境(Integrated Development Environment，IDE)是用于提供程序开发环境的应用程序，一般包括代码编辑器、编译器、调试器和图形用户界面工具。就是集成了代码编写功能、分析功能、编译功能、调试功能等一体化的开发软件服务套。所有具备这一特性的软件或软件套(组)都可以叫做集成开发环境。如微软的 Visual Studio 系列，Borland 的 C++ Builder、Delphi 系列等。该程序可以独立运行，也可以和其他程序并用。例如，BASIC 语言在微软办公软件中可以使用，可以在微软 Word 文档中编写 Word Basic 程序。IDE 为用户使用 Visual Basic、Java 和 PowerBuilder 等现代编程语言提供了方便。

不同的技术体系有不同的 IDE。比如 Visual Studio.Net 可以称为 C++、VB、C#等语言的集成开发环境，所以 Visual Studio.Net 可以叫做 IDE。同样，Borland 的 JBuilder 也是一个 IDE，它是 Java 的 IDE。Zend Studio、Editplus、Ultraedit，它们都具备基本的编码、调试功能，所以都可以称做 IDE。

1.3.2　Visual Studio 2005 简介

Microsoft Visual Studio 2005 是 Microsoft 推出的新一代集成开发环境，包含了许多强大的工具，支持多种编程语言(C#、VB、C++、HTML 与 JavaScript 等)。VC++ 2005 是 Visual Studio 2005 开发套件之一，具有集成开发环境，可提供编辑 C、C++，以及 C++/CLI 等编程语言。

VC++ 2005 包括许多完全集成的工具，设计这些工具的目的是使编写 C++程序的整个过程更加轻松。作为 IDE 组成部分提供的 VC++ 2005 的基本部件有编辑器、编译器、连接器和库。这些是编写和执行 VC++ 2005 程序所必需的基本工具。这些工具的功能如下。

(1) 编辑器

编辑器给用户提供创建和编辑 VC++ 2005 源代码的交互式环境。除了那些肯定已经为人所熟知的常见功能(如剪切和粘贴)之外，编辑器还用颜色来区分不同的语言元素。编辑器能够自动识别 C++语言中的基本单词，并根据其类别给它们分配某种颜色。这不仅有助于使代码的可读性更好，而且在输入这些单词时出错的情况下可以提供清楚的指示。

(2) 编译器

编译器将源代码转换为目标代码，并检测和报告编译过程中的错误。编译器可以检测各种因无效或不可识别的程序代码引起的错误，还可以检测诸如部分程序永远不能执行这样的结构性错误。编译器输出的目标代码存储在称做目标文件的文件中。编译器产生的目标代码有两种类型。这些目标代码通常使用以.obj 为扩展名的名称。

(3) 连接器

连接器组合编译器根据源代码文件生成的各种模块，从作为 VC++ 2005 组成部分提供的程序库中添加所需的代码模块，并将所有模块整合成可执行的整体。连接器也能检测并报告错误。例如，程序缺少某个组成部分，或者引用了不存在的库组件。

(4) 库

库只不过是预先编写的例程集合，它通过提供专业制作的标准代码单元，支持并扩展了C++语言。我们可以将这些代码合并到自己的程序中，以执行常见的操作。Visual C++ 2005提供了各种库包括的例程所实现的操作，从而节省了用户亲自编写并测试实现这些操作的代码所需的工作量，大大提高了生产率。

1.4 简单的 VC++ 2005 程序

1.4.1 VC++ 2005 程序的开发过程

把设计好的 VC++ 2005 程序交计算机运行，并最终获得结果，这一系列工作主要由计算机完成，我们只需做一些比较简单的操作。

(1) 编辑 VC++ 2005 程序

第一项工作是把程序作为一个文本(字符)文件输入到计算机中。编辑的工作主要包括创建新程序文件(一般是.cpp 文件)，输入，浏览，修改，插入，删除等。

任何一种文本编辑器都可以完成这项工作，大多数 C++系统如 Visual C++、Borland C++等都提供了集成开发环境 IDE(Integrated Develop Environment)。IDE 不仅为用户提供了方便的编辑功能，也包括了文件管理、编译、链接和项目管理等多方面的支持。

由于 IDE 为用户提供了编辑、编译、链接、调试等综合环境。例如，在编好一个程序后，可以当即编译、运行，再根据编译和运行情况(包括系统提供的出错信息和调试手段)修改程序，使程序迅速调试通过或修改得更好。

(2) 编译和链接过程

VC++ 2005 程序不能直接交计算机运行，需经编译系统(Compiler)编译，变成由机器指令组成的可执行程序，然后由计算机运行。

编译前的 VC++ 2005 程序称为源程序，编译后的机器语言程序称为目标程序，如图 1-3 所示。

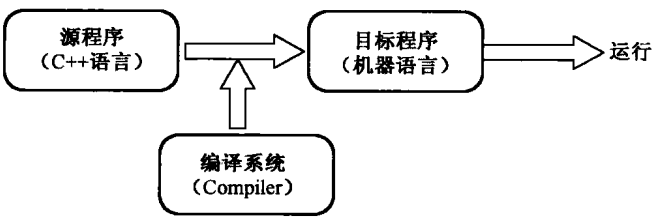

图 1-3 源程序编译为目标程序

实际上从 VC++ 2005 源程序到可执行的机器语言程序还有两个步骤必须完成:

① 编译预处理。在源程序被编译之前，须经“预处理”，其任务是根据程序中的预处理指令，完成指定的预处理任务。本节给出的简单程序中包含预处理指令：

```
#include <iostream.h>,
using namespace std;
```

其具体操作是在系统提供的 std 名字空间的标准库中找出 iostream 头文件，并把它嵌入到该预处理指令的位置。

② 链接(linking)。链接的任务是把已编译好的目标程序与其他需共同运行的目标程序和系统提供的库程序链接起来，形成可执行程序。