面向对象程序设计系列教材

Visual C++与面向对象程序设计教程

（第二版）

吕　军　杨　琦　罗建军　刘路放　编

冯博琴　审

高 等 教 育 出 版 社

（京）112 号

内 容 提 要

本书主要介绍如何应用 Visual C++进行面向对象和可视化编程。

本书在第一版的基础上，结合读者和教师反馈以及进一步的教学实践，对内容的选取、讲授方法、例题与习题等进行了全面的修订，以便更适应该课程的教学要求。主要内容包括：C++入门与 Visual C++编程环境，程序设计基础，面向对象的概念和方法，图形用户界面程序设计等。本书在讲授方式上注意结合应用开发实例，讲练结合，精讲多练，注重培养学生的程序设计和综合开发能力。书中配有丰富的例题和习题。

本书可作为高等学校计算机或相关专业的教材或参考书，也可供应用开发人员学习参考。

本书配有教学辅助课件，供教师教学和学生自学使用。

策划编辑 何新权　　**责任编辑** 倪文慧
封面设计 李卫青　　**责任印制**

出版发行	高等教育出版社	**购书热线**	010−64054588
社　　址	北京市西城区德外大街 4 号	**免费咨询**	800−810−0598
邮政编码	100011	**网　　址**	http:// www.hep.edu.cn
总　　机	010−82028899		http:// www.hep.com.cn
经　　销	新华书店北京发行所		
印　　刷			
开　　本	787×1092　1/16	**版　　次**	年　月第 1 版
			年　月第 2 版
印　　张	22	**印　　次**	年　月第　次印刷
字　　数	550 000	**定　　价**	26.40 元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

初 版 前 言

随着 Windows 操作系统的崛起，由传统的面向控制台的字符软件开发向面向窗口的可视化编程转化已成为必然趋势。而 Visual C++正是 Windows 环境下最强大、最流行的程序设计语言之一。

Visual C++支持面向对象的程序设计方法（Object-Oriented Programming，OOP），支持 MFC（Microsoft Foundation Class）类库编程，有强大的集成开发环境 Developer Studio（其中包括了程序自动生成向导 AppWizard、类向导 Class Wizard 和各种资源编辑器，以及功能强大的调试器等可视化和自动编程辅导工具）。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，开发效率很高，生成的应用软件代码品质优良。这一切都使得 Visual C++成为许多专业程序开发人员的首选。

然而，Visual C++一向有“难学”的名声，许多初学者视学习 Visual C++为畏途。究其原因，一方面是 Visual C++（包括 MFC 类库）的规模庞大，结构复杂，难于理出一条循序渐进的学习路线；另一方面是其 AppWizard 自动生成的程序专业化程度高，代码量大，结构复杂，以其为基础编写的例题难于为初学者理解和掌握。因此，目前的 Visual C++教科书多是为已有 C 语言或 C++语言编程基础的人准备的，起点较高。

本书是 Visual C++入门教科书，适用于本科类计算机及相关专业学生程序设计能力的培养。为了克服上述困难，使基础不高的初学者也能很快地掌握程序设计方法，我们在确定教学目标、设计教学模式、编写教程内容等方面进行了一系列革新探索，以现代教育理论为指导，多媒体教学手段为基础，提出了“精讲多练”的教学模式。使用“精讲多练”模式进行 Visual C++这类程序设计语言课程的教学，效果很好。

本教程的目标是使学生掌握使用 Visual C++设计应用程序的基本技能，了解面向对象的和结构化的程序设计方法，能够编写、调试和运行实用、规范、可读性好的 Visual C++程序。不像其他 Visual C++教材需要学习者具有一定的程序设计基础（如学过 C 语言或 C++语言），本书“从零开始”，不要求学生有程序设计方面的先修课程。但在学习本课程时，学生最好对计算机的使用有一定了解（了解 Windows 的使用，具有键盘操作和文件处理的基础）。

我们在设计本教程内容时，以面向对象的和结构化的程序设计方法思想贯穿全书，并以大量篇幅介绍了 Visual C++程序的调试技术和一些典型应用程序的设计思路，其中有些是作者在长期的编程和教学实践中摸索和总结出来的心得。

本教程共分 16 章，分别对应 16 个教学重点。这 16 个教学重点又可分为两组：前 8 章为一组，处理 C++的基本内容，包括控制结构、基本数据类型、表达式、函数，指针和引用，以及类与对象的基本概念和封装、继承和多态性等面向对象程序设计的基础理论。在学习了这些内容之后，学生应能编写、调试和运行一般规模和难度的控制台应用程序（如数值计算类程序），并对面向对象的和结构化的程序设计方法有所了解，为编写较大规模的应用程序打下基础。后 8 个章处理 Windows 编程技术，包括消息传递机制、MFC 应用程序框架、

设备环境、资源、文档/视图结构、对话框和控件等。在这一部分中，强调对基本概念的理解和掌握，以及在理解和掌握的基础上编写具有较复杂的窗口界面的 Windows 应用程序的能力。

为了便于教学，每章均按以下主题进行组织:

教学目标和学习要求 本书的特点是“精讲多练”，因此为教师和学生规定明确的教学和学习目标是非常重要的。

授课内容 是建议教师课堂讲授的内容。一般来说，授课内容是本章所有教学内容的“纲”，起着联系本章所有项目的作用。授课内容部分的分量按两学时组织。第 1 章的授课内容分量略轻，这是因为在第 1 章的授课时间中还应划分出部分时间用于介绍编辑、调试和运行应用程序项目的基本步骤。

自学内容 “自学内容”和“授课内容”部分一起组成了一个章的基本教学内容。这部分内容通常都是“授课内容”的延伸和继续，由学生在课外时间自学。必须强调的是自学部分并非不重要，也不能省略。一般来说，教师应在授课时间中抽出 5 ~ 10 分钟对自学内容略作导引，以便利学生自学。

调试技术 介绍 Developer Studio 集成开发环境的使用方法，以及如何调试、连接和运行 Visual C++应用程序项目。强调编程实践是本书的重要特色。第 1 章的调试技术中的部分内容可以在授课时间讲授，其他章的调试技术一般由学生自学，同时也可以作为学生上机的实验指导书。辅导教师在带学生上机时应对这些内容进行现场辅导。

程序设计举例 为了补充授课内容和自学内容部分的例题，我们设置了程序设计举例栏目。本栏目所有例题均与本章的授课、自学或调试技术等部分的内容相关，是学生学习复习本章的重要参考资料。

上机练习题 每章均配有若干上机练习题目，供学生上机练习。这些练习题目均为程序设计题目，传统的做法是先编程，再上机。由于C++的特点，也可以在写出较详细的伪代码程序之后直接上机。“精讲多练”式教学方法的基本特点是上机时数较多，所以这部分的习题工作量较大，因此在上机时数不足的情况下可以酌情选做若干题目。

为了保证教学效果,在条件许可的情况下最好采用直接在计算机房进行的联机电化教学。在这种情况下，每个教学单元（即每章）可使用连续的 4 课时，先由教师讲解授课部分并对自学部分和调试技术等内容进行简短的指导（共 2 学时），然后学生即可在教师指导下上机练习（2 学时）。此外，如果能够提供一定数量的课外机时（如 20~30 小时）则更好。

近年来，我中心在计算机基础教育的理论和实践等方面进行了一系列探索和革新，其成果（“精讲多练”的教学模式是其中之一）荣获了 1997 年度国家级教学成果一等奖。这些成果都是在冯博琴教授的领导下完成的，本课程的建设也不例外。本教程的构思和编写得到了冯博琴教授的多方指导，并由他审核了书稿，在此向冯老师表示深深的谢意。在本书编写过程中，曾与李波、罗建军、卫颜俊、杨琦、吕军和张伟诸同事进行了多次交流，受益匪浅。以上同事还提供了一些有用的材料；杨琦同志为本书绘制了部分插图，在此一并表示感谢。
由于作者学识浅陋，编写时间仓促，书中错误在所难免。希望读者不吝指教。

编者于西安交通大学

2000 年 4 月

第二版前言

本书自 2000 年 7 月面市，转眼已近三年了。在这三年里，仅在本校先后就有十余位教师、近万学生参与了本课程的教学实践。在教学中，授课教师多次开展教学法活动，互相听课、研讨，并以各种形式听取学生的意见和建议。我们建立了一整套教学体系，并不断完善教学环节，包括多媒体授课、网络视频课堂、电子答疑、电子作业提交与批改以及上机编程的考核方法，以图贯彻“精讲多练”的教学方针。

在教学实践和教学法交流活动中，授课教师和学生对本书提出了大量建议。除了指出原书中存在的错误外，这些建议集中反映在学习难度上。作为 Visual C++的入门教科书，并且面向“零起点”的学生，本书的目标在第一版就明确定位为“使学生掌握使用 Visual C++设计应用程序的基本技能”，以及能够编写、调试程序，而不是对 Windows 编程的全面介绍。为了能在一本几十个课时的教材中涵盖 Visual C++的基本技术，对于 C、C++和 Windows 编程技术相关内容的选择一直是我们反复推敲的重点。经过三年的积累，我们深感需要对有关内容进行修订和调整，以反映这些来自教学一线的需求。

随着微软.net 的推广，有的教师建议将教学环境过渡到.net 框架，以适应技术发展的潮流。经过多次讨论，我们认为语言开发环境的变化是快速的，而语言本身在相当时间内会保持相对稳定。作为入门教材，我们应更注重对学生基本程序设计能力的培养，而不过分依赖于开发环境。为此，我们在新版中没有引入最新的开发环境，而把重点放在内容的取舍和例题难度、跨度的调整上。

除了修正原书中的错误外，第二版主要进行了以下方面的修改和调整:

（1）加大了 C++部分内容的份量，由原来的两章改为三章，并增加了相应例题，以期强化初学者面向对象程序设计的能力。为保持总课时数，Windows 编程部分相应地压缩了一章。

（2）降低学习难度，删除了原书中的“Win32 应用程序”编程模式、使用非模式对话框编程方法，并改写了相关例题，以单文档（SDI）编程模式为主线介绍 Windows 编程，以使学生更关注于程序设计本身，弱化对开发环境的学习。

（3）增加部分贴近实际应用的例题，如应用数值分析及图示编程，为学生在后续课程及以后工作中应用编程技术打下良好基础。

（4）对例题和概念的讲解进行了全面的润色，并从第十章开始，采用 step-by-step 的方法指导学生在向导生成的程序基础上进行编程，更有利于读者自学。原书中编程技巧较高的例题放在附录中，以满足学习进度较快的学生的要求。

第一版的主要作者刘路放教授已经远在加拿大，第二版的修订工作是冯博琴教授应高等教育出版社要求，组织西安交通大学计算机教学实验中心多位在课程教学一线的教师共同完成的，这些教师中有些人还同时承担其他语言的教学工作。参加修订的有杨琦（第 1~7 章），仇国巍（第 8、9 章），吕军（第 10、11、15、16 章），朱丹军（第 12 章），薛涛（第 13、

14 章），崔舒宁（本书的部分例题）。本书由罗建军、杨琦统稿，全书由冯博琴教授主审。刘路放教授对本书的修改提出了重要的建议，在此致以诚挚的感谢。其他授课教师在百忙中也对本书提出许多有益的建议，在此一并致谢。特别的致谢给予那些在调查问卷、课堂调查以及通过 BBS、E-mail 向教师提出建议的广大学生。

由于作者学识浅陋，编写时间仓促，书中错误在所难免。希望读者不吝赐教。

编者于西安交通大学

2003 年 6 月

目　录

第一章　C++入门……1

1.1　软件开发与C++语言……1
1.2　算法与程序……2
1.3　输入、编译、调试和运行一个C++程序……4
1.4　C++语言的输入与输出命令……4
1.5　程序设计语言的发展……5
1.6　C++程序的基本要素……7
1.6.1　标识符、关键词和标点符号……7
1.6.2　注释……7
1.6.3　源程序……8
1.7　编译预处理……8
1.7.1　宏定义……8
1.7.2　文件包含……9
1.8　Visual C++的集成开发环境……10
1.8.1　菜单和工具栏……11
1.8.2　Developer Studio 窗口……11
1.8.3　用 Developer Studio 编写和调试简单 C++程序……11
1.8.4　菜单选项、快捷键和工具栏……12
上机练习题……15

第二章　控制结构……16

2.1　程序的基本控制结构……16
2.2　C++的控制结构……18
2.2.1　顺序结构……18
2.2.2　选择结构……18
2.2.3　循环结构……19
2.3　结构化程序设计……20
2.4　伪代码……21
2.5　结构化程序设计方法简介……24
2.6　C++的其他控制转移语句……24
2.6.1　switch 语句……25
2.6.2　goto 语句和语句标号……26
2.6.3　break 语句和 continue 语句……27
2.6.4　exit()函数和 abort()函数……28
2.7　Developer Studio 的文本编辑器……28
上机练习题……34

第三章　基本数据类型……35

3.1　数据类型……35
3.1.1　整型数据的表示方法……36
3.1.2　实型数据的表示方法……36
3.2　常量……36
3.2.1　整型常量……37
3.2.2　实型常量……37
3.2.3　字符常量……37
3.2.4　字符串常量……38
3.3　变量……38
3.3.1　变量的声明……38
3.3.2　变量的初始化……40
3.4　数组……40
3.5　字符型数组和字符串处理库函数……41
3.6　类型修饰符和常量修饰符……43
3.7　枚举类型……44
3.8　typedef 语句……45
3.9　Developer Studio 的文件处理功能……46
3.10　Visual C++程序的编译、连接和运行……47
3.11　查看和修改编译、连接错误……48
上机练习题……51

第四章　表达式……53

4.1　算术运算符和算术表达式……53
4.2　逻辑运算符和逻辑表达式……54
4.3　赋值运算符和赋值表达式……54
4.4　自增运算符和自减运算符……55
4.5　表达式中各运算符的运算顺序……56
4.6　其他具有副作用的运算符……57
4.7　问号表达式和逗号表达式……58
4.8　类型不同的数据之间的混合算术运算……59
4.9　运行错误的判断与调试……60

4.10 基本调试手段……61
4.11 注释号在调试中的作用……62
4.12 条件编译……62
上机练习题……66

第五章 函数……67

5.1 函数的定义……67
5.2 函数的调用……69
5.3 函数原型……70
5.4 函数间的参数传递……70
5.4.1 值调用……70
5.4.2 引用调用……71
5.5 函数重载……72
5.6 局部变量和全局变量……73
5.7 内联函数……74
5.8 带有默认参数的函数……75
5.9 C++的库函数……75
5.10 函数模板……76
5.11 变量的存储类别……77
5.11.1 自动变量（auto）……77
5.11.2 静态变量（static）……77
5.12 Developer Studio 的跟踪调试功能……78
上机练习题……81

第六章 指针……83

6.1 地址与指针……83
6.1.1 地址……83
6.1.2 指针……84
6.2 指针运算……84
6.3 指针与数组……87
6.4 动态存储分配……90
6.5 指针和函数……91
6.5.1 指针作为函数的参数……91
6.5.2 返回指针的函数……92
6.5.3 指向函数的指针……93
6.6 指针的数组……94
6.7 指针的初始化……96
6.8 Visual C++的帮助功能……96
上机练习题……99

第七章 类和对象……100

7.1 面向对象的思想……100
7.2 面向对象程序设计的特点……101
7.3 类与对象……102
7.3.1 类的定义……102
7.3.2 成员函数的定义……103
7.3.3 内联成员函数……104
7.3.4 对象……104
7.4 构造函数与析构函数……106
7.5 数据成员的初始化……108
7.6 对象与指针……109
7.7 const 修饰符……110
7.8 MFC 的 CString 类……111
7.9 MFC 的 CTime 类和 CTimeSpan 类……113
7.9.1 CTime 类……114
7.9.2 CTimeSpan 类……115
7.9.3 CTime 类和 CtimeSpan 类的运算……115
7.10 类的嵌套……115
7.11 如何在程序中使用 MFC 类库……116
7.12 使用 FileView 标签……116
上机练习题……119

第八章 继承与派生……120

8.1 继承与派生……120
8.1.1 为什么使用继承……120
8.1.2 派生类的定义……121
8.1.3 派生类中的变化……122
8.2 派生类的继承方式……122
8.2.1 公有继承……122
8.2.2 私有继承……124
8.2.3 保护继承……126
8.3 派生类的构造函数和析构函数……127
8.3.1 构造函数……128
8.3.2 析构函数……129
8.4 显式访问基类成员……129
8.5 静态成员……131
8.6 类模板……132
8.7 使用 ClassView 标签……134
上机练习题……139

第九章 多态性……140

9.1 多态性概述……140
9.2 派生类对象替换基类对象……142
9.3 虚函数……143
9.3.1 虚函数定义……143
9.3.2 虚函数的使用限制……145
9.4 抽象类……146

9.5　运算符重载……148
9.6　文件处理……150
9.7　异常处理机制……152
上机练习题……159

第十章　Windows 编程……160

10.1　Windows 编程的基本思想……160
10.2　MFC 编程……161
10.3　单文档界面（SDI）应用程序……162
10.4　在窗口的客户区输出文字和图形……163
10.5　编制消息处理函数……167
10.5.1　消息映射……167
10.5.2　利用 ClassWizard 编制消息处理函数……168
10.6　鼠标和键盘消息处理……168
10.7　Windows 的用户界面对象……170
10.7.1　窗口……170
10.7.2　系统菜单……171
10.7.3　标题栏……171
10.7.4　菜单栏……171
10.7.5　工具条……171
10.7.6　客户区……171
10.7.7　垂直滚动条和水平滚动条……172
10.7.8　状态栏……172
10.7.9　图标……172
10.7.10　光标……172
10.7.11　插入符……172
10.7.12　对话框……172
10.7.13　控件……172
10.8　Windows 数据类型与变量的命名规则……173
10.9　用 AppWizard 生成文档/视图结构的程序框架……175
10.10　Developer Studio 的 ClassWizard（类向导）……184
上机练习题……191

第十一章　文档/视图结构……192

11.1　文档/视图概念……192
11.2　视图类……193
11.3　文档类……194
11.4　文档/视图类之间的协作关系……195
11.5　使客户区重绘……200
11.6　定时器消息……203
11.7　文档/视图结构中的应用程序类……204
11.8　文档/视图结构中的框架窗口类……205
11.9　文档/视图结构中各类对象之间的协作关系……205
11.10　Visual C++的常用调试宏……205
11.10.1　TRACE()宏……205
11.10.2　ASSERT()宏……206
11.10.3　ASSERT_VALID()宏……206
11.10.4　CObject::Dump()成员函数……206
上机练习题……212

第十二章　图形设备接口和资源……213

12.1　设备环境类和图形对象……213
12.2　Windows 应用程序资源……214
12.3　库存图形对象……215
12.4　画笔与画刷……217
12.5　位图……223
12.6　菜单……225
12.7　字体……227
12.8　绘图模式……229
12.9　GDI 坐标系……230
12.10　图标、快捷键和字符串表……231
12.11　向项目中添加资源……231
12.12　资源编辑器……232
12.12.1　图标编辑器……232
12.12.2　位图编辑器……233
12.12.3　菜单编辑器……233
12.12.4　快捷键编辑器……234
12.12.5　字符串表编辑器……234
上机练习题……242

第十三章　对话框……243

13.1　对话框（Dialog）……243
13.2　控件……246
13.3　对话框的初始化……246
13.4　对话框的数据交换和数据检验机制……246
13.5　非模态对话框……252
13.6　公用对话框……252
13.6.1　颜色选择对话框……253
13.6.2　字体选择对话框……253
13.7　对话框模板资源的编辑……254
13.8　使用 ClassWizard 建立对话框类……256
13.9　为对话框类加入成员变量……256

上机练习题……263

第十四章 控件……264

14.1 常用控件……264
14.2 基于对话框的应用程序……274
14.3 动画控件……277
14.4 用 AppWizard 生成基于对话框的应用程序……279
上机练习题……281

第十五章 文档读写与打印……282

15.1 序列化（Serialize）……282
15.2 打印和打印预览……283
15.3 自定义类的序列化……286
15.4 编写独立的打印处理程序……288
15.5 更新命令用户接口消息……291
15.6 工具条与状态条……294
15.7 Developer Studio 的输出窗口……295
上机练习题……297

第十六章 多文档界面程序……298

16.1 MDI 应用程序……298
16.2 滚动视图……304
16.3 对话视图……305
16.4 文本编辑视图……306
16.5 使用 AppWizard 建立 MDI 程序框架……307
上机练习题……309

附录……310

附录 1 ASCII 码表……310
附录 2 常用库函数……311
附录 3 可供两人对弈的中国象棋程序……317
附录 4 防空战游戏程序……328
附录 5 七巧板程序……336

第一章　C++入门

教学目标

介绍Ｃ++程序的基本结构以及在计算机上输入、编译、调试和运行Ｃ++程序的基本方法和步骤。

学习要求

了解Ｃ++程序的基本特点，熟悉 Visual C++集成开发环境的基本使用方法。

授课内容

1.1　软件开发与Ｃ++语言

我们知道，使用计算机工作是通过相应的应用软件进行的，如用于文字处理的 Word，科学计算的 MATLAB，表格处理的 Excel 和各种数据库软件等，这些软件均由专业软件开发人员设计编程。一般来说，日常工作中遇到的任务大多数可借助现成的应用软件完成，但有时仍需为具体问题自行开发相应的软件。特别是在工程应用领域中，解决各类项目中可能遇到的大量具体问题，使用通用的软件不仅效率低，还可能无法完成任务。在这种情况下，自行编写具有针对性的相应软件可能是唯一的解决方法。

编写计算机软件需要使用程序设计语言。目前可用的计算机语言很多，各有特点。有些适用于开发数据库应用程序，有些适用于开发科学计算程序，有的简便易学，有的功能全面。在本课程中，我们介绍 C++语言。

C++是从 C 语言发展演变而来的，因此介绍 C++就不能不首先回顾一下 C 语言。

1969 年，美国贝尔实验室的 Ken Thompson 为 DEC PDP-7 计算机设计了一个操作系统软件，这就是最早的 UNIX。接着，他又根据剑桥大学的 Martin Richards 设计的 BCPL 语言为 UNIX 设计了一种便于编写系统软件的语言，命名为Ｂ。Ｂ语言是一种无类型的语言，直接对机器字进行操作，这一点和后来的Ｃ语言有很大不同。作为系统软件编程语言的第一个应用，Ken Thompson 使用Ｂ语言重写了其自身的解释程序。

1972~1973 年间，同在贝尔实验室的 Denis Ritchie 改造了Ｂ语言，为其添加了数据类型的概念，并将原来的解释程序改写为可以直接生成机器代码的编译程序，然后将其命名为Ｃ。1973 年，Ken Thompson 小组在 PDP-11 机上用Ｃ重新改写了 UNIX 的内核。与此同时，Ｃ语言的编译程序也被移植到 IBM 360/370、Honeywell 11 以及 VAX-11/780 等多种计算机上，

迅速成为应用最广泛的系统程序设计语言。

然而，C 语言也存在一些缺陷，例如类型检查机制相对较弱；缺少支持代码重用的语言结构等，因此用 C 语言开发大程序比较困难。

为了克服 C 语言存在的缺点，并保持 C 语言简洁、高效的特点，贝尔实验室的 Bjarne Stroustrup 博士及其同事开始对 C 语言进行改进和扩充，将“类”的概念引入了 C 语言，构成了最早的 C++语言（1983）。后来，Stroustrup 和他的同事们又为 C++引进了运算符重载、引用、虚函数等许多特性，并使之更加精炼，于 1989 年推出了 AT&T C++2.0 版。随后美国标准化协会以 AT&T C++2.0 为基础，制定了 ANSI C++标准。各软件商推出的 C++编译器都支持该标准，并有不同程度的拓展。

C++支持面向对象的程序设计方法（参阅第 7 章），特别适合于中型和大型的软件开发项目，从开发时间、费用到软件的可重用性、可扩充性、可维护性和可靠性等方面，C++均具有很大的优越性。同时，C++又是 C 语言的一个超集，这就使得许多 C 代码不经修改就可被 C++编译器编译通过。

早期的C++编译器称为 Cfront，实际上是一个预处理程序，负责将C++程序转换为C程序，然后再由C语言编程序继续编译。后来也出现了可以直接将C++程序编译为目标代码的C++编译程序。

随着 Windows 操作系统的出现，应用程序的外观和结构发生了巨大的变化，程序设计环境也得到了很大改善。为了适应 Windows 编程，各软件厂商纷纷推出了新型 C++编译器，Microsoft 公司的 Visual C++就是其中的佼佼者。Visual C++并不是一个单纯的编译器，而是一整套用于软件开发的集成环境（IDE），其中包括了文本编辑、编译连接、调试、可视化界面设计和完备的在线帮助，甚至可直接通过 Internet 与 Microsoft 公司的站点连接，以得到技术支持和产品更新升级。

Visual C++语言支持面向对象的程序设计方法，适用于 Windows 应用程序的开发，可用于开发各类应用程序，功能十分强大，是目前最流行的程序设计语言之一。本教程以面向对象的程序设计方法为中心，本着简明、实用和循序渐进的原则，介绍使用 Visual C++开发应用程序的基本方法。

1.2　算法与程序

要编写用于解决应用问题的程序，首先必需确定解决问题的方案，也就是算法。例如，对于问题：给定两个正整数 p 和 q，如何求出其最大公因数？古希腊数学家欧几里德给出了一个著名的算法：

步骤 1：如果 $p < q$，交换 p 和 q。

步骤 2：求 p/q 的余数 r。

步骤 3：如果 $r = 0$，则 q 就是所求的结果。

否则反复做如下工作：

令 $p = q$，$q = r$，重新计算 p 和 q 的余数 r，直到 $r = 0$ 为止；

则 q 就是原来两个正整数的最大公因数。

从原则上说，有了算法就可以直接求解问题了。但如果问题比较复杂，计算步骤很多，则应通过编程用计算机解决。

例 1.1 使用欧几里德算法，编写程序求解任意两个正整数的最大公因数。

说明 根据 1.8.3“用 Developer Studio 编写和调试简单 C++程序”建立项目并调试运行。

程序

```
// Example 1.1: 计算两个正整数的最大公因数
#include <iostream.h>                                    //基本输入输出库
void main()
{   int p, q, r;                                         // 说明三个整型变量 p,q,r
    cout<< "Please input two integer numbers:" << endl;      // 提示用户由键盘输入两个正整数
    cin >> p >> q;
    // 如果 p < q, 交换 p 和 q
    if(p<q)
    {   r = p;
        p = q;
        q = r;
    }
    r = p%q;                                             // 计算 p 除 q 的余数 r
    // 只要 r 不等于 0,重复进行下列计算
    while(r != 0)
    {   p = q;
        q = r;
        r = p%q;
    }
    cout << "The maximum common divisor is " << q << "." << endl;   // 输出结果
}
```

输入 输入两个整数，中间用空格分隔。如：12 18

输出 The maximum common divisor is 6.

分析 可以看出，该程序的主体部分几乎与原算法完全相同，只是增加了一些 C++语言特有的内容。

程序的第 1 行是注释。注释以“//”开头，直到该行的末尾，用于说明或解释程序段的功能、变量的作用，或者程序员想要说明的任何内容。在该程序中还有一些注释，用于说明程序的结构。在将C++程序编译成目标代码时所有的注释行都会被忽略掉，因此即使使用了很多注释也不会影响目标码的效率。恰当地应用注释可以使程序清晰易懂、便于调试，便于程序员之间的交流与协作，因此，在编写程序时恰当地使用注释是一个良好的编程习惯。

第 2 行是编译预处理，把 iostream.h 头文件插入到程序中函数体之外的地方，当一个程序包含有 iostream.h 时，几个标准流就自动地定义了，其中包括 cin 和 cout。

从第 3 行到最后一行是主函数。主函数是该程序的主体部分，由其说明部分

```
void main()
```

和用一对花括号“{}”括起来的函数体构成。在函数体内，除了注释行以外，还有语句。C++

的语句可分为说明语句和执行语句，说明语句用于说明程序中使用的变量、函数等的类型和参数，执行语句实际执行某功能。第 5 行就是一个类型说明语句

```
int p, q, r;
```

说明在该程序中使用了 3 个整型变量。关于类型和变量等概念，将在第 3 章中详细介绍。

就一般的计算机程序而言，总是要包括 3 个基本内容：数据输入、计算和输出。

该程序的输入部分首先在计算机屏幕上显示一行提示信息（第 7 行）：

```
Please input two integer numbers:
```

然后等待用户由键盘输入两个整数。用户输入的两个正整数由语句

```
cin >> p >> q;
```

分别存入变量 p 和 q。

程序下面的部分就是欧几里德算法的具体实现了。可以看出，C++程序类似于自然语言，只是结构更加严谨。

程序中的最后一条语句是输出语句，将计算结果显示在计算机屏幕上。

1.3　输入、编译、调试和运行一个 C++程序

C++是一种编译语言，源程序经过编译、连接，生成可执行文件后方可运行。开发一个 C++应用程序大致要经过以下步骤：

1. 首先根据实际问题确定编程思路，包括选用适当的数学模型。
2. 根据上述思路或数学模型编写程序。
3. 编辑源程序。源程序一般以文件的形式存放在磁盘上（后缀为.cpp）。
4. 编译和连接。就 C++而言，编译阶段将源程序转换成目标文件（后缀为.obj），连接阶段将目标文件连接成可执行文件（后缀为.exe）。
5. 反复上机调试程序，直到改正了所有的编译错误和运行错误。调试过程中应精心选择典型数据进行试算，避免因调试数据不能反映实际数据的特征而引起计算偏差和运行错误。
6. 运行程序。

应该说明的是，如果要利用 C++开发一个大型应用系统，例如管理信息系统、数据库应用系统、计算机辅助教学系统或者实时控制系统等，一般要比编写一个数值计算方面的应用程序复杂得多，上面介绍的开发步骤就显得过于简单了。这时要遵循软件工程的方法进行应用系统开发。

1.4　C++语言的输入与输出命令

程序通常会要求用户提供一些信息（如数据），这一过程被称为程序的输入。程序通常总是要向用户发出一些信息（如计算结果等），这一过程被称为程序的输出。C++程序的输入操作由 cin 对象完成，输出操作由 cout 对象完成。

cin 的基本用法为：

```
cin>>V1>>V2>>…>>Vn;
```

其中>>称作提取运算符，V_1，V_2，…，V_n都是变量。这个语句的意思是，程序暂时中止执行，等待用户从键盘输入数据。用户输入所有数据后，应以回车键表示输入结束，程序将用户键入的数据存入各变量中，并继续执行下面的语句。例如：

```
cin >> p >> q;
```

就是要求用户分别为变量 p 和 q 各输入一个数据。在输入时，应注意用空格或 Tab 键将所输入的数据分隔开。

请注意，当用户响应 cin 的要求输入数据时，输入数据的类型（在第 3 章中介绍）应与接收该数据的变量的类型相匹配，否则输入操作将会失败或者得到一个错误数据。

cout 的基本用法为：

```
cout << E1 << E2 <<…<< Em;
```

其中<<称做插入运算符，E_1，E_2，…，E_m都是表达式（在第 3 章中介绍）。这个语句的意思是，将各表达式的值输出（显示）到屏幕上当前光标位置处。在输出时，要注意恰当使用字符串和换行符 endl，提高输出信息的可读性。

自学内容

1.5　程序设计语言的发展

自从 1946 年在美国宾夕法尼亚大学诞生世界上第一台真正能自动运行的电子数字计算机至今，计算机已经成为人们生产劳动和日常生活中必备的重要工具。

完整的计算机系统包括硬件和软件两大部分。硬件是指计算机系统中的各种物理设备，包括控制器、运算器、内存储器、I/O 设备以及外存储器等，它是计算机系统的物质基础。软件是相对于硬件而言的。从狭义的角度上讲，软件是指计算机运行所需的各种程序；而从广义的角度上讲，还包括手册、说明书和有关的资料。

要用计算机解决问题就必须利用某种计算机语言提供的命令来编制程序，并把程序存储在计算机的存储器中，然后在这个程序的控制下运行计算机，达到解决问题的目的。

用于编写计算机可执行程序的语言称为程序设计语言，程序设计语言按其发展的先后可分为机器语言、汇编语言和高级语言。如图 1.1 所示。

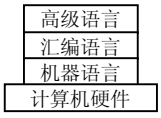

图 1.1　程序设计语言的分类

机器语言是伴随着第一台计算机的诞生而诞生的，它能被计算机直接理解和执行，它在形式上是由“0”和“1”构成的一串二进制代码，每种计算机都有自己的一套机器指令。机器指令的集合就是机器语言。例如，下面这段代码就是机器语言的片段：

```
10111000001101000010010
10111011011110000101011O
0000001111000011
```

机器语言与人所习惯的语言差别很大，难学、难记，开发实用的计算机程序很不方便。

不久就出现了汇编语言，它将机器指令映射为一些可以被人读懂的助记符，如 ADD、SUB 等。同时又用变量取代各类地址，这样构成的计算机符号语言称为汇编语言。用汇编语言编

写的源程序必须经过翻译（汇编），变成机器语言程序才能被计算机识别和执行。

汇编语言的工作原理很简单，就是用指令代替相应的机器码，前面的那段机器代码，用汇编指令描述为：

```
mov    ax,1234
mov    bx,5678
add     ax,bx
```

汇编语言在一定程度上克服了机器语言难于辨认和记忆的缺点，但对大多数用户来说仍然是不便理解和使用的。

为了克服汇编语言的缺点，出现了“高级程序设计语言”，这是一种类似于“数学表达式”、接近自然语言、又能为机器所接受的程序设计语言。高级语言具有学习容易、使用方便、通用性强、移植性好的特点，便于各类人员学习和应用。

早期应用比较广泛的几种高级语言有 FORTRAN、Basic、Pascal 等。FORTRAN 语言是各种高级语言的鼻祖。在此之后，上百种高级程序设计语言诞生并发展、完善，并根据应用领域的不同和语言本身侧重点的差异，分成了许多种类别。一般说来，语言和语言之间都会有差异，但在计算机高级程序设计语言中，这种差异只是反映了概念上的差别。比如偏重于数值计算的语言和偏重于商业应用的语言，各自体现了相关领域的基本概念和活动流程。

高级语言程序（称为源程序）必须经过相应的翻译程序翻译成用机器指令表示的程序（称为目标程序）后，才能由计算机执行。这种翻译通常有两种作法。

1. 编译方式：将高级语言源程序输入计算机后，调用编译程序将其整个地翻译成机器指令表示的目标程序，然后执行目标程序，得到计算结果。如图 1.2 所示。

2. 解释方式：讲高级语言源程序输入计算机后，启动解释程序，翻译一句执行一句，直到程序执行完为止。如图 1.3 所示。

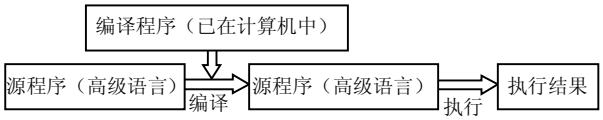

图 1.2　高级语言的编译方式

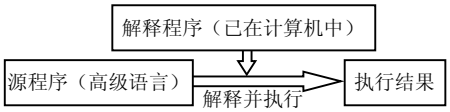

图 1.3　高级语言的解释方式

20 世纪 60 年代末开始出现的结构化高级编程语言进一步提高了语言的层次。结构化数据、结构化语句、数据抽象、过程抽象等概念使程序更便于体现客观事物的结构和逻辑含义，这种编程语言与人类的自然语言更接近。结构化程序设计语言成为当时程序设计的主流语言，几乎为所有的程序员所接受和使用。它的产生和发展形成了现代软件工程的基础。但是结构化方法的主要问题是程序中的数据和操作分离，不能有效地组织成与自然界中的具体事物紧密对应的程序成分。

自 1986 年以来，面向对象的技术开始进入实际应用。现在，在工业和商业上应用中更多地采用面向对象程序设计语言解决实际问题。面向对象的编程语言将客观事物看作具有属性和行为的对象，通过抽象找出同一类对象的共同属性和行为，形成类。通过类的继承与多态可以方便地实现代码重用，大大缩短了软件开发周期，并使得软件风格统一。

Smalltalk 是第一个真正的面向对象的程序设计语言，对这类语言可进行如下分类：