普通高等教育“十一五”规划教材

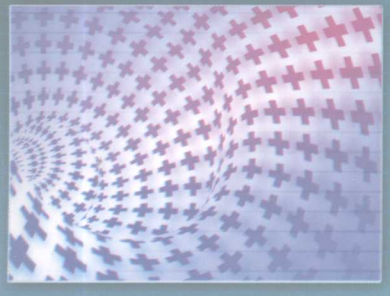
高等院校计算机技术系列教材

C++语言程序设计

刘怀亮 主编

胡子义 副主编

江流丰 张少龙 范创海 苏瑞娟 编著



研究出版社

普通高等教育“十一五”规划教材
高等院校计算机技术系列教材

C++语言程序设计

刘怀亮　主编

胡子义　副主编

江流丰　张少龙　范创海　苏瑞娟　编著

研究出版社

图书在版编目（CIP）数据

C++语言程序设计 / 刘怀亮主编.
—北京：研究出版社，2008.4
普通高等教育“十一五”规划教材
高等院校计算机技术系列教材
ISBN 978-7-80168-356-4

I. C…
II. 刘…
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（2008）第 049975 号

出版发行 研究出版社
地　址：北京 1746 信箱（100017）
电　话：010-63097512（总编室）010-64045344（发行部）
E-mail：yjcbsfxb@126.com
经　　销 新华书店
印　　刷 广州锦昌印务有限公司
版　　次 2008 年 4 月第 1 版　　2008 年 4 月第 1 次印刷
规　　格 787 毫米 x 1092 毫米　　1/16　　18.5 印张
字　　数 425 千字
定　　价 38.00 元　　ISBN 978-7-80168-356-4

本书销售专线：010-64045344　64041660

前　言

一、关于本书

本书是根据普通高等教育“十一五”国家级规划教材的指导精神编写的。

在各种面向对象的程序设计语言中，C++最受人青睐。首先，C++是 C 的扩展集，是 C 语言的延伸，而 C 已拥有了最广泛的用户；其次，它既可进行面向过程的程序设计，也可进行面向对象程序设计，是生成代码效率最高的语言之一。而且 C++实现了类的封装、继承及多态，使得其代码更容易维护和具有高度可重用性。学习 C++不仅可以深入理解面向对象程序设计的一些基本概念，而且非常有助于进一步学习其他的计算机语言。

本书介绍了 Visual C++6.0 开发环境下编程工具的使用、C++基本理论思想和基本编程技能，及上机实训，当读者完成本书的学习后，就能够熟悉 C++这门语言，为以后计算机知识的学习打下坚实的基础。

二、本书结构

本书共 13 章，总体结构如下：

第 1 章：从 C 到 C++。介绍了语言的由来、面向对象程序设计的思想、C++语言程序的基本框架，学会使用 Visual C++ 6.0 集成开发环境编辑、编译、运行与调试程序。

第 2 章：C++的数据类型与表达式。介绍了 C++数据类型的定义与使用方法、C++运算符的种类、运算符的优先级和结合性、C++表达式类型及求值规则等内容。

第 3 章：C++的程序控制结构。介绍了程序顺序执行的特性、顺序编程的方法、控制程序执行流程的语句和方法、预处理的原理和作用等内容。

第 4 章：函数。介绍了函数的结构、类型和返回值，函数调用的方法，参数的传递，嵌套机制和递归机制，内联和重载技术，变量的作用域和命名空间的使用等内容。

第 5 章：指针、数组、结构体和共用体。介绍了指针与指针变量的概念、数组、const 的用法与作用、一维数组、多维数组、结构体和共用体等内容。

第 6 章：类与对象。介绍了 C++类与对象的基本概念、访问方式、访问属性与特殊成员（静态成员、常成员等）、构造函数和析构函数的调用机制和用法、友员函数和友员类等内容。

第 7 章：运算符重载。介绍了如何重载运算符、成员函数和友员函数重载运算符的区别、常用的运算符重载以及类型转换的原理等内容。

第 8 章：继承。介绍了继承的基本概念、派生类的访问控制、多继承的概念、实现方式及其派生类的构造和访问、虚拟继承等内容。

第 9 章：虚函数和多态性。介绍了基类指针与派生类指针的引用、虚函数的定义和使用、纯虚函数和抽象类、两种多态性的原理等内容。

第 10 章：C++的 I/O 流。介绍了 C++流的原理、流类库的组成和使用、标准流的操作、流错误状态的构成和诊断、输入/输出的格式控制、如何进行文件处理等内容。

第 11 章：模板。介绍了模板的概念和特点、函数模板的使用和重载、类模板的使用。

第 12 章：异常处理机制。介绍了异常处理机制的概念、异常处理机制的思想、异常处理机制的实现、常见的异常处理等内容。

第 13 章：上机实训。分为 11 个实训，是对前面所学章节知识的巩固练习。

三、本书特点

本书主要具有如下几个特点：

（1）针对性强，循序渐进，结构清晰。

本书针对初学者的学习习惯和心理，从基础入手，从零开始，介绍最基本的计算机知识和最基本的操作以及最需要掌握的计算机技能，循序渐进，内容结构清晰明了。

（2）突出应用型培训计划，遵循 20%&80%规律，重实践，可操作性强。

章首都有教学要求及主要内容，符合 20%&80%规律，突出计算机最重要的 20%功能，囊括实践中常见的 80%例子，每个重点都会设计一个完整的实例供读者阅读程序，专门的上机实训和每章后的综合操作题具有趣味性和典型性，着重培养动手和理解能力，更能巩固所学内容。

（3）重点难点突出，编程风格优雅，解释详细。

每章结尾都有小结，使得读者对本章知识一览无遗。使用了特性段落方式，即黑体字的“注意”，标示的要点或操作技巧。在源程序中，代码格式布置得条理清楚、层次分明，语句简单易明，重点程序段都附有详细的注释。

四、本书适用对象

本书内容结构清晰明了、循序渐进、通俗易懂，适合各本专科院校的师生或者成人教育学校、培训学校的学员作为教材使用，也可供各行各业想要学习 C++语言的读者参考。

本书由刘怀亮负责全书统稿，并编写全书大部分内容，约 20 万字，胡子义编写约 5 万字，江流丰编写约 4 万字，张少龙编写约 5 万字，范创海编写约 4 万字，苏瑞娟编写约 4 万字。本书在编写过程中还得到阙彩球老师的监审与帮助，在此表示衷心感谢。

由于编写时间仓促，水平有限，书中疏漏之处在所难免，敬请各位读者批评指正。联系方法如下：

电子邮箱：service@cnbook.net

网址：www.cnbook.net

本书的电子教案、源代码及习题参考答案可从该网站下载，此外，该网站还有一些其他相关书籍的介绍，供读者参考。

编　者

2008 年 3 月

目　录

第 1 章　从 C 到 C++ 1

1.1　语言的由来 1

1.1.1　C 语言的由来 1

1.1.2　C++语言的由来 1

1.2　面向对象程序设计的介绍 2

1.2.1　面向对象的方法 2

1.2.2　面向对象的基本概念 2

1.3　简单的 C 和 C++例子 3

1.3.1　基本相同的 C 语言和 C++语言实例 3

1.3.2　扩展的 C++语言实例 4

1.4　C++程序的结构 5

1.4.1　基本的框架 5

1.4.2　程序的说明 5

1.4.3　基本的 I/O 5

1.5　编程工具 Visual C++ 6.0 6

1.5.1　Visual C++ 6.0 简述 6

1.5.2　开发 C++源程序的过程 6

1.5.3　C++单文件的开发步骤及调试 7

小结 10

习题一 10

一、选择题 10

二、填空题 11

三、思考题 12

四、上机操作题 13

第 2 章　C++的数据类型与表达式 14

2.1　数据类型 14

2.1.1　字符集和词汇 14

2.1.2　数据存储 15

2.1.3　整型、浮点型、字符型、逻辑型、空值型 16

2.1.4　枚举类型 18

2.1.5　typedef 类型 19

2.1.6　引用类型 20

2.1.7　常量与变量 20

2.2　运算符与表达式 21

2.2.1　算术运算符和表达式 21

2.2.2　关系运算符和表达式 22

2.2.3　逻辑运算符和表达式 23

2.2.4　赋值运算符和表达式 24

2.2.5　条件运算符和表达式 25

2.2.6　逗号运算符和表达式 25

2.2.7　位运算符和表达式 26

小结 27

习题二 27

一、选择题 27

二、填空题 28

三、思考题 29

四、上机操作题 30

第 3 章　C++的程序控制结构 31

3.1　顺序结构语句 31

3.1.1　声明语句 31

3.1.2　赋值语句 32

3.1.3　简单的 I/O 语句 33

3.2　选择结构语句 34

3.2.1　if 语句 34

3.2.2　if-else 语句 35

3.2.3　switch 语句 36

3.3　循环结构语句 37

3.3.1　while 语句 37

3.3.2　do-while 语句 38

3.3.3　for 循环语句 39

3.3.4　循环的嵌套 41

3.4　转向语句 43

3.4.1　break 语句 43

3.4.2　continue 语句 44

3.4.3　goto 语句 45

3.5　预处理语句 46

3.5.1 #define 预处理 46
3.5.2 #include 预处理 47
3.5.3 #ifdef- #else- #endif 预处理 47
小结 48
习题三 48
一、选择题 48
二、填空题 50
三、思考题 51
四、上机操作题 53
第 4 章 函数 54
4.1 函数的结构与调用 54
4.1.1 函数的结构 54
4.1.2 函数的调用 55
4.1.3 函数的返回类型 56
4.1.4 函数的原型 57
4.2 函数参数的传递 57
4.2.1 参数缺省 57
4.2.2 传值参数 58
4.2.3 引用参数 59
4.2.4 指针参数 61
4.3 函数的嵌套与递归 62
4.3.1 函数的嵌套 62
4.3.2 函数的递归 64
4.4 函数指针 66
4.4.1 函数的地址 66
4.4.2 函数的指针 67
4.5 函数的内联和重载 69
4.5.1 函数的内联 69
4.5.2 函数的重载 70
4.6 作用域和命名空间域 72
4.6.1 局部变量 72
4.6.2 全局变量 73
4.6.3 命名空间域 75
小结 76
习题四 77
一、选择题 77
二、填空题 78
三、思考题 79
四、上机操作题 80
第 5 章 指针、数组、结构体和共用体 82
5.1 指针类型 82
5.1.1 地址的引入 82
5.1.2 指针变量的定义 83
5.1.3 指针变量的运算 85
5.1.4 const 的约束 88
5.2 数组 89
5.2.1 一维数组 89
5.2.2 多维数组 92
5.2.3 字符数组和字符串 94
5.2.4 指针数组 97
5.3 结构体和共用体 98
5.3.1 结构体的定义 98
5.3.2 访问结构成员 99
5.3.3 结构体赋值 100
5.3.4 结构体与指针 101
5.3.5 共用体的定义 102
5.3.6 共用体变量的引用 102
小结 103
习题五 104
一、选择题 104
二、填空题 105
三、思考题 106
四、上机操作题 108
第 6 章 类与对象 109
6.1 类和对象的定义 109
6.1.1 类和对象的概念 109
6.1.2 访问对象成员 112
6.1.3 类的访问限制 114
6.1.4 动态内存分配 116
6.1.5 this 指针 118
6.2 构造函数和析构函数 119
6.2.1 简单的构造函数和析构函数 119
6.2.2 带参数的构造函数 121
6.2.3 重载构造函数 123
6.2.4 复制构造函数 124

6.3 静态成员 125
6.3.1 静态数据成员 125
6.3.2 静态成员函数 127
6.4 友员 128
6.4.1 友员函数 128
6.4.2 友员类 129
小结 131
习题六 131
一、选择题 131
二、填空题 132
三、思考题 134
四、上机操作题 136

第 7 章 运算符重载 138

7.1 重载运算符 138
7.1.1 C++的重载运算符 138
7.1.2 运算符重载的语法形式 139
7.2 成员函数和友员函数重载运算符的区别 140
7.2.1 用成员函数重载运算符 140
7.2.2 用友员函数重载运算符 142
7.3 常用的运算符重载 144
7.3.1 重载自增运算符和自减运算符 144
7.3.2 重载赋值运算符 145
7.3.3 重载下标运算符和函数调用符 148
7.3.4 重载输入输出流运算符 149
7.4 类型转换 150
7.4.1 类型转换的规则 150
7.4.2 类型转换函数 151
小结 153
习题七 153
一、选择题 153
二、填空题 154
三、思考题 155
四、上机操作题 158

第 8 章 继承 159

8.1 继承的基本概念 159
8.1.1 继承的概念 159
8.1.2 继承的实现方式 161
8.2 基类和派生类 162
8.2.1 访问控制 162
8.2.2 重写基类成员 166
8.2.3 派生类中的静态成员 167
8.3 多继承 168
8.3.1 多继承的概念 168
8.3.2 多继承的实现方式 170
8.3.3 多继承的派生类构造和析构 171
8.3.4 多继承的二义性 172
8.3.5 虚拟继承 173
小结 174
习题八 175
一、选择题 175
二、填空题 176
三、思考题 177
四、上机操作题 180

第 9 章 虚函数和多态性 182

9.1 类指针的引用 182
9.1.1 基类指针引用派生类对象 182
9.1.2 派生类指针引用基类对象 183
9.2 虚函数 186
9.2.1 虚函数的使用 186
9.2.2 虚析构函数 189
9.3 纯虚函数和抽象类 190
9.3.1 纯虚函数 191
9.3.2 抽象类 192
9.4 多态性 193
9.4.1 编译时多态 193
9.4.2 运行时多态 194
小结 195
习题九 196
一、选择题 196
二、填空题 197
三、思考题 197
四、上机操作题 200

第 10 章 C++的 I/O 流 201

10.1 C++流的概念 201
10.1.1 流类库的介绍 201
10.1.2 流类库的应用举例 202
10.2 标准流操作和格式控制 203
10.2.1 标准输入流操作 204
10.2.2 标准输出流操作 205
10.2.3 流错误状态 206
10.2.4 设置标志字 207
10.2.5 格式控制符 209
10.3 文件处理 211
10.3.1 文件的打开和关闭 211
10.3.2 文本文件 214
10.3.3 二进制文件 216
10.3.4 文件的随机读写 217
小结 219
习题十一 220
一、选择题 220
二、填空题 221
三、思考题 221
四、上机操作题 222

第 11 章 模板 224

11.1 模板的基本概念 224
11.2 函数模板 224
11.2.1 函数模板的说明 225
11.2.2 函数模板的使用 225
11.2.3 函数模板的重载 227
11.3 类模板 228
11.3.1 类模板的说明 228
11.3.2 类模板的使用 229
小结 231
习题十一 231
一、选择题 231
二、填空题 233
三、思考题 233
四、上机操作题 235

第 12 章 异常处理机制 236

12.1 C++异常处理机制 236
12.1.1 异常处理的概述 236
12.1.2 异常处理的基本思想 237
12.2 异常处理的实现 238
12.2.1 异常处理的语法 238
12.2.2 异常处理中的构造与析构 240
12.2.3 常见的异常处理 242
小结 244
习题十二 244
一、选择题 244
二、填空题 246
三、思考题 247
四、上机操作题 250

第 13 章 上机实训 251

实训 1 数据类型与表达式 251
实训 2 C++控制语句 253
实训 3 函数 255
实训 4 数组、指针与字符串 258
实训 5 类与对象 260
实训 6 运算符重载 264
实训 7 继承与派生 268
实训 8 多态性 271
实训 9 输入/输出流 275
实训 10 模板 277
实训 11 异常处理机制 280

附录 284

A.1 ASCII 码表 284
A.2 运算符和结合性 284

参考文献 286

内容简介 287

第 1 章　从 C 到 C++

计算机编程语言的一大进步就是高级语言的出现。高级语言抛开了机器的硬件细节，同时提高了语言的抽象层次。而面向对象语言的出现，主要解决的是将程序中数据和操作的分离，这样就能够更有效地组成与自然界中的具体事物紧密对应的程序成分。面向对象语言设计的出发点就是为了能更直接地描述客观世界中存在的事物以及它们之间的关系。

本章教学目标：

（1）了解 C/C++语言的由来。

（2）了解面向对象程序设计的思想。

（3）掌握 C++语言程序的基本框架。

（4）学会使用 Visual C++ 6.0 集成开发环境编辑、编译、运行与调试程序。

1.1　语言的由来

每一种语言都有它的由来，当然，计算机编程语言也是一样，从低级语言到高级语言的变化发展都是为了在书写程序时可以联系到程序所描述的具体事物。

1.1.1　C 语言的由来

回顾语言的由来，第一代语言——机器语言，完全由二进制数字 0 和 1 组成，能够被计算机直接识别和执行，具有灵活、速度快等优点。但是，比较困难的问题在于程序员要熟悉全部的指令及其含义，手工去处理每条指令和每个数据。后来为了克服语言的难读写和容易出错等缺点，第二代语言——汇编语言产生了。和机器语言一样，汇编语言也是面向机器的，但它采用与指令代码实际含义较近的英文缩写单词来取代指令代码，其指令和机器的指令是一一对应的，可是同样也依赖硬件系统，导致可移植性依然很差。1957 年，出现了第一个高级程序设计语言 FORTRAN，主要是用于科学运算。后来出现的 LISP、COBOL、ALGOL60、BASIC、PASCAL 和 PROLOG 等高级语言都称为第三代语言。最后出现的这些语言在人工智能、情报检索、商业数据处理和关系数据库等领域都得到广泛使用。

1967 年，英国剑桥大学的学者 Matin Richards 对 CPL 做了简化，推出了 BCPL 语言。1970 年美国贝尔实验室以 BCPL 语言为基础进行了简化，设计出了很接近硬件的 B 语言（取 BCPL 的第一个字母）。1972 年，贝尔实验室在 B 语言的基础上设计出 C 语言（取 BCPL 的第二个字母）。C 语言是一种具有一般高级语言特性，又具有低级语言特性的编程语言。1973 年，贝尔实验室的 K.Thompson 和 D.M.Ritchie 两人合作用 C 语言重写了 UNIX 操作系统。随着 UNIX 的日益广泛使用，C 语言也迅速得到发展。1987 年美国标准化协会（ANSI）制定了 C 语言的标准，也就是 ANSI C。

1.1.2　C++语言的由来

1960 年，ALGOL60 语言诞生，块结构的概念提出了。1967 年，面向对象语言的鼻祖 Simula 67 提出了对象的概念，并且支持类和继承。随后出现的 Smalltalk 语言继续丰富和

发展了 Simula 67 中的面向对象的概念，并且实施了更加严格的信息隐藏。1980 年，Simula 80 面世，面向对象技术开始显示出魅力，成为 C++之前最具有影响力的面向对象语言。由于 C 语言类型检查机制较弱，这使得程序中的一些错误不能在编译时被发现，而且其本身几乎没有支持代码重用的机制，也使得各个程序的代码很难为其他程序所使用，另外对大型的软件项目，程序员很难控制程序的复杂性。为解决日益增长的软件需求，避免 C 语言的不足，1980 年贝尔实验室的 Bjarne Stroustrup 开始对 C 语言进行改编，1983 年正式推出一种新的语言—C++语言。1990 年，Bjarne Stroustrup 博士出版了 The Annotated C++ Reference Manual（ARM），由于当时还没有出台 C++标准，ARM 成为事实上的标准，1990 年，C++引入了模板和异常的概念，使 C++具备了泛型编程和更好的错误处理能力。1993 年，运行时类型识别（RTTI）和命名空间（Namespace）的概念被加入到 C++中。1998 年，ANSI 和 ISO 先后批准 C++语言成为美国国家标准和国际标准。

1.2 面向对象程序设计的介绍

面向对象程序设计（Object-Oriented Programming，简称 OOP）将数据及对数据的操作放在一起，作为一个相互依存、不可分割的整体来处理，采用数据抽象和信息隐藏技术。它将对象及对象的操作抽象成一种新的数据类型——类，并且考虑不同对象之间的联系和对象类的重用性。概括为“对象+消息=面向对象的程序”。

1.2.1 面向对象的方法

面向对象方法（Object-Oriented Method）是一种将面向对象的思想应用于软件开发过程中，指导开发活动的系统方法，简称 OO（Object-Oriented）方法，是建立在“对象”概念基础上的方法学。对象是由数据和允许的操作组成的封装体，与客观实体有直接对应关系，一个对象类定义了具有相似性质的一组对象。而继承是对具有层次关系的类的属性和操作进行共享的一种方式。所谓面向对象就是基于对象概念，以对象为中心，以类和继承为构造机制，来认识、理解、刻画客观世界并且设计、构建相应的软件系统。

在面向对象的方法出现之前，采用的都是面向过程的程序设计方法。面向过程的程序设计方法是把数据和处理数据的过程分离，如果数据结构发生改变，那么相关的处理过程也要进行相应的修改，同时新方法都要带来额外的开销，导致程序的可重用性差。而面向对象的程序设计就把数据及对数据的操作放在一起，作为一个相互依存、不可分割的整体——对象来处理。将同类型对象抽象出它们的共性，组成一种新的数据类型——类。类通过一个简单的外部接口与外界联系，对象与对象之间通过消息进行通信。

面向对象方法所具有的模块化、信息封装与隐蔽、抽象性、继承性、多样性等独特之处，为研制大型软件，提高软件可靠性、可重用性、可扩充性和可维护性提供了有效的手段和途径。

1.2.2 面向对象的基本概念

1. 对象

在面向对象程序的系统中，对象是基本的运行时的实体，它既包括数据（属性），也包括作用于数据的操作（行为），所以一个对象把属性和行为封装为一个整体。封装是一种

信息隐蔽技术，它的目的是使对象的使用者和生产者分离，使对象的定义和实现分开。从程序设计者来看，对象是一个程序模块；从用户来看，对象为他们提供了所希望的行为。在对象内的操作通常叫作方法。一个对象通常可由对象名、属性和操作三部分组成。

2. 消息

对象之间进行通信的一种构造叫作消息。但一个消息发送给某个对象时，包含要求接收对象去执行某些活动的信息，接收到消息的对象经过解释，然后予以响应，这种通信机制叫做消息传递。发送消息的对象不需要知道接收消息的对象如何对请求予以响应。

3. 类

一个类定义了一组大体上相似的对象，一个类所包含的方法和数据描述一组对象的共同行为和属性。类是在对象之上的抽象，对象是类的具体化，是类的实例。通常把一个类和这个类的所有对象称为“类及对象”或对象类。

4. 继承

继承是父类和子类之间共享数据和方法的机制。这是类之间的一种关系，在定义和实现一个类时，可以在一个已经存在的类的基础上进行，把这个已经存在的类所定义的内容作为自己的内容，并加入若干新的内容。

一个父类可以有多个子类，这些子类都是父类的特例，父类描述了这些子类的公共属性和操作。一个子类可以继承它的父类中的属性和操作，这些属性和操作在子类中不必定义，子类中还可以定义自己的属性和操作。

如果一个子类只从一个父类得到继承，叫做“单重继承”；如果一个子类有两个或更多个父类，则称为“多重继承”。

5. 多态

不同的对象收到同一个消息可以产生完全不同的结果，这一种现象叫作多态。在使用多态时，用户可以发送一个通用的消息，而实现的细节则由接收对象自行决定。这样，把具有通用功能的消息存放在高层次，而不同的实现这一功能的行为放在较低层次，在这些低层次上生成的对象能够给通用消息以不同的响应。

1.3 简单的C和C++例子

本节将通过例子比较面向过程的C语言和面向对象的C++语言之间的一些最基本的区别，也让大家简单认识这两种语言。

1.3.1 基本相同的C语言和C++语言实例

C语言与C++语言最主要的区别是：C语言中没有类的概念，而C++语言引入了类。C++语言和C语言的语法是基本相同的。再者，由于C++语言是在C语言的基础上发展而来的，所以也具有面向过程的C语言的性质。下面是一个用C语言编写的简单例子。

```
//问题：使用面向过程的C语言
//源程序 1_1.cpp
#include <stdio.h>
int main()
{
    printf("Welcome to the C language world\n");
    return 0;
}
```

运行结果如图 1-1 所示。

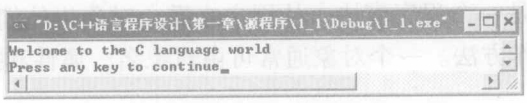


图 1-1 使用面向过程的 C 语言

程序说明：在源程序 1_1.cpp 中，使用的是面向过程的 C 语言，在 C 语言中，基本的 I/O 库是“stdio.h”，输入字符串用 printf 函数。以下再看面向过程的 C++语言例子，比较它们的不同之处。

```
//问题：使用面向过程的 C++语言
//源程序 1_2.cpp
#include <iostream.h>
int main()
{
    cout<<"Welcome to the C++ language world"<<endl;
    return 0;
}
```

运行结果如图 1-2 所示。

图 1-2 使用面向过程的 C++ 语言

程序说明：在源程序 1_2.cpp 中，使用的是面向过程的 C++语言，在 C++语言中，基本的 I/O 库是“iostream.h”，输入字符串用 cout 命令。整个程序结构看起来都是一样的，而且现在很多集成开发环境，都是兼容的。但 C++语言比 C 语言有更宽广的空间，本书后面的章节一一讲述。

1.3.2 扩展的 C++语言实例

C++语言的本质特点是引进了类和对象的概念，并具备了继承、重载、多态的特性，类的封装也得到了程序员的青睐。下面来看一个面向对象的 C++语言的例子。

```
//问题：使用面向对象的 C++语言
//源程序 1_3.cpp
#include <iostream.h>
const double PI = 3.14159;
class CircleArea {
public:                          //公有函数说明
    void setRadius();            //设置圆的半径
    void getArea();              //计算圆的面积
    void printArea();            //打印圆的面积
private:                         //私有成员变量说明
    double radius;               //半径
    double area;                 //面积
};
void CircleArea::setRadius() {
    cout<<"Please input radius of a circle:";
    cin>>radius;
}
void CircleArea::getArea() {
    area = PI * radius * radius;
}
void CircleArea::printArea() {
    cout<<"Area of a circle is "<<area<<endl;
```

```
}
int main() {
    CircleArea c;                    //声明对象
    c.setRadius();
    c.getArea();
    c.printArea();
    return 0;
}
```

运行结果如图 1-3 所示。

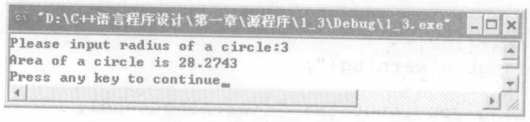


图 1-3 使用面向对象的 C++ 语言

程序说明：在源程序 1_3.cpp 中，使用的是面向对象 C++语言，用户定义的类类型是一种抽象的数据类型，使用类的过程，即在程序中说明类类型变量的过程，叫作对象的创建。创建对象也为对象分配内存。程序中 class CircleArea 定义了一个类，类名是 CircleArea。public 和 private 是定义属性和方法的访问类别，除此之外还有 protected。接着是方法的具体实现。最后在主程序中声明对象，称为实例化。第 6 章将会详细讲述类的概念及使用。

1.4 C++程序的结构

函数是 C++结构化程序设计中程序的基本组成单元，这样的一个 C++程序可被认为是函数串。类是 C++面向对象程序设计的基本组成单元，这样的一个 C++程序可被认为是类串。

1.4.1 基本的框架

由上面的面向过程的 C++语言以及面向对象的 C++语言两个例子（源程序 1-2.CPP、1-3.CPP）看，C++程序一般都是由 4 个部分组成：预处理指令、全局说明、程序的主函数和用户自定义的函数。预处理指令就是源程序中所包含的各种编译指令，都是以“#”开始的。全局说明一般包括一些程序所要使用的全局变量、类说明等。而主函数是任何一个 C++程序都有且只有一个的。用户自定义的函数都是一些功能的实现。

程序中的头文件是通过 include 预处理指令包含进来，用“#”标识。如果文件名用“<”和“>”括起来，说明这个文件是一个工程或者标准头文件，查找过程会检查预定义的目录。如果文件名是用一对引号括起来，则说明该文件是用户提供的头文件，查找该文件时从当前文件所在的目录开始，然后查找环境变量设置的目录。一般情况下，引用系统的头文件都用“<”和“>”括起来，比如系统的输入输出文件<iostream.h>，引用用户自定义的文件使用双引号。

1.4.2 程序的说明

C++语言跟 C 语言一样有两种注释符号，一种是注释对（/*，*/）。编译器把编写在这之间的语言当作注释。注释可以放在程序的任何位置，可以跨越多行程序。另一种注释是双斜线（//），注释一个单行，程序行注释符右边的内容都被当作注释而被编译器忽略。

1.4.3 基本的 I/O

C++的输入输出是通过流来实现的（将在第 10 章介绍），其输入输出是 C++本身自带

的，引用头文件<iostream.h>，并根据数据类型自动使用合适的输入输出方式。标准输入使用 cin，标准输出使用 cout，标准错误输出 cerr 常用来产生给程序用户的警告或者提供错误信息。下面来看一个例子。

```
//问题：使用基本的 I/O
//源程序：1_4.cpp
#include <iostream.h>
int main()
{
    char *warning;
    warning = new char[100];
    cout<<"Please input a warning:";
    cin>>warning;
    cerr<<"The warning you input is: "<<warning<<endl;
    return 0;
}
```

运行结果如图 1-4 所示。

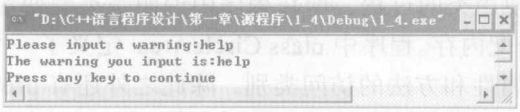


图 1-4 使用基本的 I/O

程序说明：在源程序 1_4.cpp 中，使用了经常使用的基本 I/O，cout 是标准打印输出，cin 是标准输入，最后的 cerr 是标准的错误输出，在这个程序中输出的功能跟 cout 的使用是一样的，一般都用来提示出错信息。关于输入输出流的内容，会在后面章节中详细讲述。

1.5 编程工具 Visual C++ 6.0

Visual C++ 6.0 是 Microsoft 公司的 Developer Studio 6.0 工具集的重要组成部分，是一种用于开发 Windows 应用程序的可视化开发工具。它改善了传统的编程手段，使得程序员可以直接在用户界面良好的可视化开发环境中进行工作。

1.5.1 Visual C++ 6.0 简述

Visual C++ 6.0，是微软公司推出的开发 Win32 应用程序的、面向对象的可视化集成工具。它的最大优点就是提供了功能强大的 MFC 类库，MFC 是一个很大的 C++类层次结构，其中封装了大量的类及其函数，很多 Windows 程序所共有的标准内容可以由 MFC 的类来提供，MFC 类为这些内容提供了用户接口的标准实现方法，程序员所要做的就是通过预定义的接口把具体应用程序特有的东西填入这个轮廓，这将简化编程工作，大大地减少程序员编写的代码数量，使编程工作变得更加轻松容易。

Visual C++ 6.0 也是 C/C++的集成开发工具，是目前最常用的版本。集成开发工具里面包括各种编辑器、编译工具、集成调试器等等。在编写程序的过程中，各种操作都可以通过单击相应的菜单完成。

1.5.2 开发 C++源程序的过程

（1）编辑：是 C++程序开发的第一步，主要包括文本的输入和修改。在 Visual C++ 6.0 集成开发环境中，可以使用编辑窗口进行编辑工作。保存时应将文本保存为以.cpp 为扩展名的文件。

（2）编译：是由源程序文件转换到目标文件的过程。在 Visual C++ 6.0 集成开发环境中，可以使用编译（Compile）命令将一个.cpp 源文件转换成一个.obj 的目标文件。

（3）链接：是将目标代码变成可执行程序（.exe 文件）的过程。在 Visual C++ 6.0 集成开发环境中，可以使用生成（Build）命令进行链接。

（4）运行和调试：得到可执行程序后进行运行，查看运行结果。在 Visual C++ 6.0 集成开发环境中，可以使用执行（Execute）命令来运行程序。如果运行结果不正确，说明源程序有错误，这就需要调试可执行程序，查找出错的原因。在 Visual C++ 6.0 集成开发环境中，可以很方便地进行调试工作。

1.5.3 C++单文件的开发步骤及调试

前面简单介绍了集成开发环境 Visual C++ 6.0，并且使用它来开发 C++源程序的过程，而本书的例子都是基于控制台的应用程序，并且都是单文件的，运行结果都是字符用户界面 DOS 程序。本小节主要讲述在 Visual C++ 6.0 环境下，开发 C++单文件应用程序的操作步骤以及调试运行的过程。

（1）默认 PC 机上已经装好 Visual C++ 6.0，启动 Visual C++ 6.0，如图 1-5 所示。

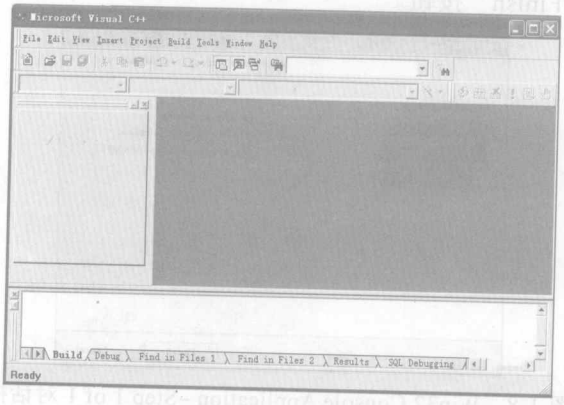


图 1-5 主窗口

（2）新建工程。在菜单栏中单击“File”，出现子菜单栏，选择“New”命令，如图 1-6 所示。

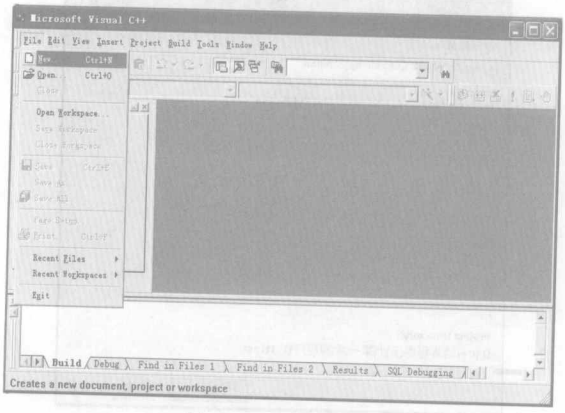


图 1-6 新建窗口

（3）此时出现“New”（新建）对话框，菜单栏默认是“Projects”（工程）选项卡，单击显示的项目类型“Win32 Console Application”。并在右边的“Project name”文本框输入工程的名字，而下面是工程所存放的路径“Loaction”，如图 1-7 所示。

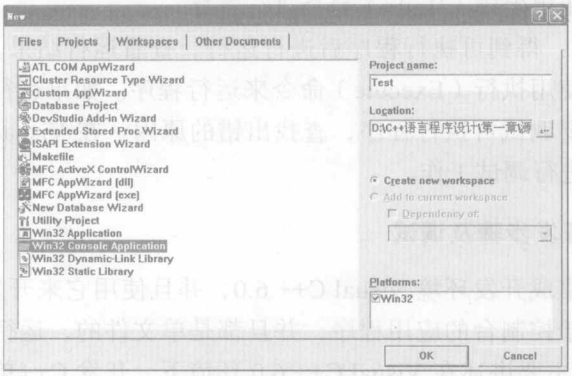


图 1-7　New 对话框

（4）单击“OK”按钮完成工程的创建。在新出现的对话框中选择“A simple application”，如图 1-8 所示。单击“Finish”按钮。

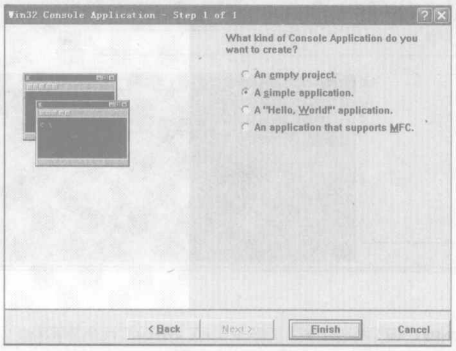


图 1-8　Win32 Console Application –Step 1 of 1 对话框

（5）接着出现的是新建工程信息对话框，完成项目的创建单击“OK”按钮，如图 1-9 所示。

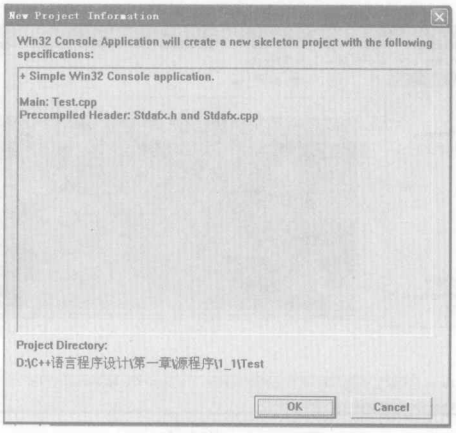


图 1-9　New Project Information 对话框