21世纪大学计算机基础规划教材

C/C++程序设计教程

王连相 主 编
赵付青 副主编

- 本书是作者在多年软件开发和教学实践经验的基础上，探讨了现代计算机教学的规律，收集分析了大量的教学文献，并基于实际应用编写而成。
- 本书内容翔实，概念清晰，通过大量实例分析引导读者快速掌握C/C++程序设计的技巧。
- 本书适合作为高校教材用书，也适合作为各种培训和编程爱好者及参加全国计算机等级考试（二级C语言）考生和工程技术人员的自学参考书。

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

21 世纪大学计算机基础规划教材

C/C++程序设计教程

王连相　主　编

赵付青　副主编

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

内 容 简 介

本书全面介绍了C语言的基本概念、语法规则和程序设计的基本方法，数组与函数，指针类型的各种操作，复合数据类型及应用，文件操作和C++程序设计基础等。从实用的目的出发，列举了大量有一定实用价值的程序，每一章配有适量的习题。

本书适合于高校作为教材用书，也适合于各种培训和编程爱好者及参加全国计算机等级（二级C语言）考试人员作为自学参考书。

图书在版编目（CIP）数据

C/C++程序设计教程 / 王连相主编. —北京：中国铁道出版社，2006.8

21世纪大学计算机基础规划教材

ISBN 7-113-07235-6

Ⅰ.C... Ⅱ.王... 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(2006)第104323号

书　　名：C/C++程序设计教程

作　　者：王连相　赵付青

出版发行：中国铁道出版社（100054，北京市宣武区右安门西街8号）

策划编辑：严晓舟　秦绪好

责任编辑：苏　茜　翟玉峰　王　丹

封面设计：薛　为

封面制作：白　雪

责任校对：黄园园

印　　刷：北京鑫正大印刷有限公司

开　　本：787×1092　1/16　**印张**：18　**字数**：421千

版　　本：2006年8月第1版　2006年8月第1次印刷

印　　数：1～5 000册

书　　号：ISBN 7-113-07235-6/TP·1942

定　　价：24.00元

前　言

C 语言是当今世界上最流行的程序设计语言之一，它以功能强大、表达灵活、使用方便和移植性好而受用户青睐。无论在设计系统软件、进行数据处理、数值计算及教学等领域都得到了广泛的应用。

21 世纪对信息技术人才的综合能力要求越来越高，需求也越来越多。随着现代编程技术的广泛应用与发展，C/C++程序设计必将更为普及。学习和掌握 C/C++语言已成为现代软件开发人员的必备知识之一。

C 语言程序设计不仅有高级语言的语义特点，还可以编写操作系统、编译程序、数据库软件系统等，还具有低级语言底层操作的能力，也可以方便地进行字节运算、字位运算、寄存器操作、取地址、设计中断、访问设备端口等。C 语言是现代软件设计人员不可或缺的软件系统开发与制作工具。随着计算机技术的飞速发展，C 语言本身也在不断地完善发展，如现在广为使用的面向对象技术的 C++、可视化编程的 Visual C、网络编程语言等，但其基本语法规范与编程风格都没变。可以说，掌握了 C 语言，就掌握了深入系统学习和应用计算机的钥匙，用户再学习理解其他现代编程技术也就方便、容易了许多。C++语言对 C 语言做了很多改进，例如增加了运算符、增加了类型的安全性、允许函数重载等。而 C++语言与 C 语言的本质区别是增加了面向对象的技术，如类、对象的封装性，基类、派生类的继承性，重载、动态联编多态性等。像新一代编程语言 Java 是一种面向对象的跨平台、适合于网络计算模式的分布式语言，而 Java 程序设计就保留了许多与 C、C++同样的语法规范与结构风格。

本书针对现代教育教学改革理念，在提高教学效率的同时，力求提高学生综合实践的能力。我们是在多年软件开发和 C/C++程序设计教学实践经验的基础上，根据现代高校教学改革特有的情况，探讨了现代计算机教学的规律，收集分析了大量的教学文献，并基于实际应用编写了本书和配套的《C/C++程序设计上机指导与测试》。本书内容翔实，概念清晰，通过实例分析，可以引导读者尽快掌握 C/C++程序设计的实际应用技巧。作为教学用书，本书内容编排比较紧凑，可以帮助读者系统全面地了解 C/C++程序设计的重点和要点。

本书共分为 10 章，其中第 1～5 章由王连相负责编写；第 6、7 章由李晓丽负责编写；第 8 章由余冬梅负责编写；第 9 章和第 10 章由赵付青负责编写。本书在编写过程中还得到了李睿、赵宏、刘纯芳、冯峰、武兆辉、周玉炳、杨朝霞、崔剑波、何万生、刘正岐和马仲海的大力支持和帮助。全书由王连相统稿并定稿，管会生对本书的编写给予了很大的指导和帮助。本书还得到了中国铁道出版社姜淑静、翟玉峰、曹莉群和戴薇等编辑的鼎力支持和帮助，在此一并表示衷心的感谢。

由于教学任务十分繁重，加之本书的编写时间非常紧迫，因此难免会出现一些错误和不足之处，殷切希望广大读者批评指正。

编　者

2006 年 7 月

目 录

第 1 章 C 语言概述 1

1.1 C 语言的起源 1
1.2 C 语言是中级语言 1
1.3 C 语言是结构化语言 2
1.4 C 语言是面向程序员的语言 3
1.5 编译和解释 4
1.6 C 语言程序结构 4
1.6.1 库和链接 6
1.6.2 分别编译 6
1.6.3 编译 C 语言程序 7
1.6.4 C 语言的内存映象 7
1.7 术语 7
习题 8

第 2 章 基本数据类型 9

2.1 数据类型 9
2.2 保留字、标识符 9
2.2.1 单词 9
2.2.2 标识符 10
2.2.3 保留字 10
2.2.4 选择合适的标识符 10
2.2.5 常量与变量 11
2.3 基本数据类型 12
2.3.1 数据的内部表示 13
2.3.2 字符型 14
2.3.3 整型 16
2.3.4 浮点类型和双精度类型 17
2.3.5 字符串常量 18
2.3.6 符号常量 19
2.4 运算符与表达式 19
2.4.1 表达式 20
2.4.2 算术运算符及算术表达式 20
2.4.3 关系运算符及关系运算表达式 21
2.4.4 逻辑运算符及逻辑表达式 22
2.4.5 赋值表达式 22

2.4.6 逗号表达式 23
2.4.7 表达式的运算顺序 23
2.4.8 条件运算 23
2.4.9 sizeof 运算 24
2.4.10 运算符优先级与结合性质 24
2.5 数据类型转换 26
2.5.1 隐式类型转换 26
2.5.2 强制类型转换 27
2.6 位运算 27
2.7 数组和指针 29
2.7.1 数组 29
2.7.2 指针 31
2.7.3 指针和数组的简单用法 32
2.8 简单应用程序举例 34
本章小结 36
习题 36
第 3 章 结构化程序设计 39
3.1 顺序结构程序设计 39
3.1.1 赋值语句 39
3.1.2 数据输入 41
3.1.3 数据输出 44
3.2 选择结构程序设计 50
3.2.1 条件语句（if） 50
3.2.2 开关选择语句 55
3.2.3 goto 语句 58
3.3 循环结构程序设计 59
3.3.1 3 种循环结构的流程表示及执行过程 59
3.3.2 循环的嵌套结构 62
3.3.3 循环程序举例 64
3.3.4 循环辅助控制 71
本章小结 73
习题 73
第 4 章 函数 76
4.1 C 语言程序的结构 76
4.2 函数的定义和调用 76
4.2.1 函数的定义 76
4.2.2 函数的调用 77

4.3 变量的存储类别和变量的使用范围 80
4.3.1 变量的存储类别 80
4.3.2 变量的作用范围 82
4.3.3 小结 83
4.4 函数间的传值 83
4.5 函数的嵌套调用 87
4.6 函数的递归调用 87
4.6.1 递归函数及其执行特点 87
4.6.2 实现递归调用的两种结构 88
4.6.3 递归结构的函数用法举例 89
4.7 函数的应用举例 91
本章小结 94
习题 94
第 5 章 编译预处理 96
5.1 宏定义 96
5.1.1 不带参数的宏定义 96
5.1.2 带参数的宏定义 97
5.2 文件包含 99
5.3 条件编译 101
本章小结 103
习题 103
第 6 章 复合数据类型 104
6.1 指针类型 104
6.1.1 指针的声明 104
6.1.2 指针的引用 105
6.1.3 指针的运算 105
6.2 数组类型 106
6.2.1 一维数组的声明 107
6.2.2 一维数组元素的引用与初始化 107
6.2.3 数组作为函数的参数 108
6.2.4 二维数组的声明与使用 110
6.2.5 二维数组元素的引用与初始化 111
6.2.6 指针与数组 113
6.2.7 指针数组与数组指针 113
6.3 字符串 116
6.3.1 字符串常量与变量 116
6.3.2 字符串数组 118

6.3.3 有关字符串操作的系统库函数 119
6.3.4 字符串的用法举例 121
6.4 结构类型 123
6.4.1 结构类型 124
6.4.2 访问结构体成员 125
6.4.3 结构指针和结构数组 127
6.4.4 结构指针作为函数的参数 131
6.4.5 结构体类型及结构指针的应用举例 132
6.5 关于指针的另外几种用法 141
6.5.1 指向函数的指针 141
6.5.2 指针类型的函数 144
6.5.3 指向指针的指针（二级指针） 145
6.5.4 main()函数的参数 146
6.6 联合、枚举和类型别名 150
6.6.1 联合类型 150
6.6.2 枚举类型 152
6.6.3 类型别名 153
本章小结 154
习题 154
第 7 章 文件及其操作 159
7.1 C 语言文件的概念 159
7.1.1 ANSI C 的 I/O 和 UNIX C 的 I/O 159
7.1.2 流和文件 159
7.1.3 C 语言文件 162
7.2 文件系统的基础 162
7.2.1 定义文件结构体类型的指针 163
7.2.2 用于文件操作的函数 164
7.3 建立文件的基本步骤 178
7.4 读取文件的基本步骤 179
7.5 非缓冲文件系统 180
7.5.1 open() 180
7.5.2 close() 181
7.5.3 creat() 181
7.5.4 read()和 write() 182
7.5.5 unlink() 183
7.5.6 lseek() 183
7.5.7 缓冲文件系统和非缓冲文件系统的区别 184

本章小结 185
习题 185
第 8 章 基本应用综合举例 186
8.1 数组元素的查找和排序 186
8.1.1 排序及应用 186
8.1.2 查找 191
8.2 数值积分 193
8.2.1 矩形法 194
8.2.2 梯形法 195
8.2.3 辛普生法 195
8.3 非线性方程的求根 197
8.3.1 牛顿迭代法 198
8.3.2 二分法 199
8.3.3 弦截法 200
8.4 C 语言动态数据结构及其用法 200
8.4.1 动态存储分配 201
8.4.2 线性动态数据结构及链表的应用 202
8.4.3 非线性动态数据结构及二叉树的应用 206
8.5 文件的应用 209
8.5.1 用户数据文件的应用 210
8.5.2 用户文本文件的应用 211
第 9 章 C++程序设计基础 218
9.1 从 C 到 C++ 218
9.2 面向对象程序设计 219
9.2.1 面向对象的概念及程序结构 219
9.2.2 面向对象程序设计的实现 220
9.3 C++程序开发过程 221
9.4 C++程序示例 222
9.5 C++对函数功能的增强 223
9.5.1 C++用函数组织程序 223
9.5.2 函数驱动 C++程序 223
9.6 C++对数据类型的扩充 225
9.6.1 基本数据类型 226
9.6.2 自定义数据类型 226
9.6.3 从结构到类 226
9.6.4 类与结构的区别 227
9.7 C++对运算符的扩充 228

9.7.1 基本运算符 228
9.7.2 运算符重载 228
9.7.3 运算符作为成员函数 230
9.7.4 转换运算符 233
9.7.5 赋值运算符 234
本章小结 237
习题 238
第 10 章 C++语言基础 239
10.1 C++的类 239
10.2 C++的输入与输出 240
10.3 类与对象 241
10.3.1 类的定义与对象的引用 241
10.3.2 构造函数与析构函数 244
10.3.3 函数重载 248
10.3.4 友元 249
10.4 对象指针 252
10.5 派生类与继承类 258
10.5.1 派生类及其定义 258
10.5.2 单继承的派生类 258
10.5.3 多继承的派生类 265
10.6 虚拟函数与多态性 270
10.6.1 虚拟函数使用方法 270
10.6.2 虚拟函数与重载函数 271
10.6.3 虚拟函数的继承 272
本章小结 273
习题 273
参考文献 277

第 1 章　C 语言概述

C 语言是由 Dennis Ritchie 创造并首先在配备 UNIX 操作系统的 DEC PDP-11 计算机上实现的。C 语言是早期计算机语言 BCPL（目前在欧洲还有应用）发展过程的产物。BCPL 由 Martin Richards 开发，并影响了 Ken Thompson 首创的 B 语言，B 语言促进了 C 语言在 20 世纪 70 年代的发展。本章主要介绍 C 语言的概况和起源、C 语言的程序构成原则和应用。

1.1　C 语言的起源

多年来，UNIX 系统配备的 C 语言一直是 C 语言的公认标准，由 Brian Kernighan 和 Dennis Ritchie 在《C 语言程序设计》（The C Programming Language，Prentice-Hall，1987）一书中描述。随着微型计算机的普及，出现了大批 C 语言系统，幸运的是，其中多数系统接受的源程序都能高度兼容。然而，没有统一标准必然存在差异，为了克服这种不利局面，ANSI（American National Standards Institute，美国国家标准学会）于 1983 年初夏成立了一个委员会，由该委员会制订了一个定义 C 语言的标准。本书写作之际，ANSI 标准委员会已经停止正式接受修改建议，所有主要 C 语言编译程序都实现了 ANSI C 标准。

1.2　C 语言是中级语言

C 语言通常被称为中级计算机语言。中级语言并不意味着功能差或难以使用，或者比 BASIC、PASCAL 等高级语言原始，也不表明与汇编语言相似，会给使用者带来类似的麻烦。C 语言之所以被称为中级语言，是因为它把高级语言的成分同汇编语言的功能结合起来。

作为中级语言，C 语言允许对位、字节和地址这些计算机功能中的基本成分进行操作，而且 C 语言具有良好的可移植性。可移植性是指能够把为某种计算机写的软件通过改编应用于另一种计算机上。例如，如果为苹果机写的一个程序，能够方便地改为可以在 IBM PC 上运行的程序，则称之为可移植的。

所有的高级语言都支持数据类型的概念，一种数据类型定义了该类变量的取值范围和在其上操作的一组运算。常见的数据类型有整数类型、实数类型和字符类型。虽然 C 语言有 5 种固有的基本数据类型，但与 PASCAL 或 ADA 相比，却不是强类型语言。C 语言几乎允许所有类型转换，例如，字符类型和整数类型数据能够自由地混合在大多数表达式中，C 语言不进行数组边界和函数参数的相容性等运行时的错误检查，检查运行时的错误是程序员的责任。

同样，C 语言的早期版本也不检查函数参数与调用变量之间的兼容性。例如，在早期版本中，用指针变量调用定义成期望浮点数变量的函数时，编译程序不产生出错信息。ANSI C 标准引入了函数原型（Function Prototype）的概念，能够报告与上例类似的潜在错误，函数原型在本书的第 4 章作具体讨论。

C 语言的特色：允许对位、字节、字和指针直接操作，因此非常适合用于实现上述操作

的系统程序的设计。C 语言的另一个重要特点是，仅有 32 个关键字（27 个来源于 Kernighan 和 Ritchie 的公认标准，另 5 个是 ANSI 标准化委员会增加的），这些关键字就是构成 C 语言的命令。和 IBM PC 的 BASIC 相比，后者包含 159 个关键字。

1.3 C 语言是结构化语言

严格地说，C 语言并不完全是块结构语言（Block-Structured Language），一般称为结构化语言（Structured Language）。C 语言与 ALGOL、PASCAL 和 Modula-2 等结构化语言非常类似。从技术上看，一个块结构语言允许在过程（Procedure）和函数（Function）中定义其他过程和函数。这样，通过使用作用域规则（Scope Rules），扩展了全局（Global）和局部（Local）的概念。作用域规则确定变量、过程，函数的可见性（Visibility）。因为 C 语言不允许在函数内再创建函数，所以不能正式称为块结构语言。

结构化语言的显著特征是代码和数据的封装（Compartmentalization）。这种语言能够把执行某个特殊任务所需的指令和数据从程序的其余部分分离出去并隐藏起来。获得隔离的方法通常是调用使用局部（临时）变量的子程序。通过使用局部变量，用户能够编制对程序其他部分没有副作用的子程序，容易编写共享代码段的程序。如果开发了一些分离良好的函数，在引用时仅需要知道函数做什么，不必知道它如何做。切记，过度使用全局变量（可以被全部程序访问的变量）会由于意外的副作用而在程序中引入错误，设计过 BASIC 语言程序的人对这个问题都会有深刻体会。

结构化语言提供了大量程序设计功能，直接支持若干循环结构，如 while，do-while 和 for 等。在结构化语言中禁止或不提倡使用 goto 语句，不能像 BASIC 语言和 FORTRAN 语言那样把它作为常用的程序流程控制语句。结构化语言允许将语句缩进，但又不要求遵守 FORTRAN 语言那种严格的程序格式。

表 1-1 所示是结构化语言和非结构化语言的例子。

表 1-1 结构化语言和非结构化语言

非结构化语言	结构化语言
FORTRAN 语言	PASCAL 语言
BASIC 语言	ADA 语言
COBOL 语言	C 语言

现代语言都是结构化的，而陈旧的计算机语言的标志之一就是非结构性。结构化语言比非结构化语言更便于程序设计，用结构化语言编写的结构清晰的程序更易于维护。这已是人们普遍接受的观点。

C 语言的主要结构成分是函数，在 C 语言中，函数是一种构件（程序块），程序的所有操作都在其中发生。函数允许一个程序中的任务被分别定义和编码，使程序模块化。可以确信，设计好的函数能在各种情况下正确工作，不会对程序的其他部分产生副作用。能否设计出独立的函数在大型项目中是至关重要的，在这种项目中，一个程序员的代码不会意外地影响其他人的程序。

在 C 语言中，实现结构化和代码隔离的另一种方法是使用复合语句。一个复合语句是作

为一个语句处理的并在逻辑上相互关联的一组语句。在 C 语言中，复合语句就是处于一对大括号之间的语句序列。例如：

```
if(x<10)
{
    printf("too low,try again.\n");
    scanf("%d",&x);
}
```

在上面的语句中，如果 x 小于 10，位于 if 语句之后、在一对大括号之间的两个语句将都被执行。这两个句子连同大括号表示一个代码块，是一个逻辑单位，其中所有语句必须一起执行。代码块不仅能使许多算法得到清晰且有效的实现，而且有助于程序员抓住程序的本质。

1.4　C 语言是面向程序员的语言

令人遗憾的是，计算机程序设计语言并非都是为程序员设计的。以典型的非程序员语言 COBOL 语言和 BASIC 语言为例。COBOL 语言的设计目标不是改善程序员的环境，不是增加代码的可靠性，也不是提高编程的速度。它的部分目标是使非程序员能够阅读并理解程序。BASIC 语言主要是供非程序员编制计算机程序、解决简单问题的。

与此形成鲜明对比的是 C 语言。C 语言是由实际工作中的程序设计人员创造、影响并测试的，最终向程序员提供了程序员期望的一切，即很少限制、很少强求、块结构、独立函数和简明扼要的关键字集合。通过 C 语言编写的程序可以基本达到汇编码的效率，程序又有 ALGOL 语言和 MODULA-2 语言的块结构。因此，C 语言自然深受第一流专业程序设计人员的欢迎。

程序设计人员广泛使用 C 语言的一个主要因素是，常能用 C 语言代替汇编语言。汇编语言使用的是汇编指令，是能够直接在计算机上执行的二进制代码的符号表示，它的每个操作都映射为计算机执行的单一任务。虽然汇编语言使程序员达到最大灵活性和发挥最高效率的潜力，但开发和调试汇编语言程序的困难也是程序员难以忍受的。进一步说，由于汇编语言是非结构化的，最后完成的程序肯定像一团面条——纠缠不清的跳转、调用和变址，这种非结构化使汇编语言难于阅读、改进和维护。更重要的是，汇编语言编写的程序不能在使用不同中央处理器（CPU）的机器之间移植。

最初，C 语言主要用于系统程序设计（System Programming），系统程序是计算机操作系统及操作系统支持的实用程序的一部分。以下列出几种常用的系统程序。

- 操作系统
- 翻译程序
- 编辑程序
- 汇编程序
- 编译程序
- 数据库管理程序

随着 C 语言的普及和发展，更由于它的可移植性和高效率，许多程序员开始用 C 语言设计各类程序。几乎所有计算机上都有 C 语言编译程序，使用户可以很少改动，甚至不加改动

地将为一种机器写的C语言源程序在另一种机器上编译执行。可移植性节省了大量的时间和财力，而且各种C语言编译程序均可产生非常紧凑、执行快捷的目标码，比任何一种BASIC语言编译程序的目标码都紧凑、快速。

也许C语言被用于各类程序设计的主要原因在于程序员的偏爱，C语言提供了汇编语言的速度和FORTRAN的可扩充性，而很少有PASCAL和MODULA-2的限制。C程序员可以创建并维护一个专门的函数库，库中的函数根据个人需要配置，可以在各种程序中使用。由于允许（更确切地说是鼓励）分别编译，C语言可以使程序员方便地管理大型项目，最大限度地减少重复劳动。

1.5 编译和解释

编译（Compile）和解释（Interpret）是两种程序执行的方式。理论上，任何程序语言都既能编译执行又能解释执行，但特定语言一般仅取两者之一。例如，BASIC语言一般是解释执行的，而C语言常为编译执行的（最近，C语言解释程序作为调试手段也受到了欢迎）。程序的执行方式不由它的编程语言确定，解释程序和编译程序只是作用于源代码上的复杂程序。

解释程序一次读进一行源代码，然后执行一串由这行源代码确定的指令。编译程序一次读完全部程序，并把它转换成目标码（Object Code）。目标码是源代码转换成能够在计算机上直接运行的代码，常指二进制码或机器码。程序被编译之后，源代码对程序的执行就毫无意义了。

使用解释程序时，每次运行程序都调用解释程序。例如，每次运行BASIC语言程序时都必须首先运行BASIC语言的解释程序，然后装入源程序，再键入RUN，BASIC语言解释程序检查、纠正并运行该程序。这种缓慢的处理过程在每次运行程序时都重复一次。相反，编译程序一次将整个源程序转换为可直接在计算机上运行的目标码，因为编译程序一次性地完成转换工作，所以每次直接运行程序时只要键入其名字即可。因此，编译是一次性开销，而解释则每次运行时都引起额外开销。

运行编译程序时需要花费额外时间，但这种开销容易被每次运行程序所节省的时间抵消。被编译的程序运行速度比被解释的程序快得多。只有一种例外，即程序很短（不足50行）并且没有循环语句。本书和编译手册中常见的两个术语是编译时（Compile Time）和运行时（Run Time），前者指发生在编译期间的事件，后者指发生在运行期间的事件。读者通常只在讨论错误时（例如在“编译错”和“运行错”这类短语中）看到这两个术语。

1.6 C语言程序结构

C语言的关键字都是小写的。在C语言中区分大小写，如else是关键字而ELSE则不是，在C语言程序中，关键字不能用于其他目的，既不能作为变量名也不能作为函数名。C语言关键字如下。

auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void	volatile	while

以上 32 个关键字，与标准 C 语言句法结合，构成程序设计语言 C 语言。其中 27 个关键字由 C 语言的原始版定义，而 enum、const、signed、void 和 volatile 这 5 个关键字由 ANSI 委员会定义。

C 语言程序都由一个或多个函数组成，其中必须出现的函数是 main()，通常把它称为 C 语言的主函数。它在程序开始运行时首先被调入执行。在一个良好的 C 语言程序中，main()总是包括程序操作的基本轮廓，它可以由一系列函数调用组成。main()被看作 C 语言的基本组成部分，虽然技术上并非如此。另外不要把 main 作为变量名字，因为可能破坏编译程序的正常操作。

下面举两个 C 语言程序的实例，以便了解并认识 C 语言程序的结构。

【例 1.1】一个简单的 C 语言程序。

```
/* C program _1 */
main()
{
 printf("C program _1\n");
}
```

这是一个简单而完整的 C 程序。第一行“/*”开头并以“*/”结尾的内容是 C 语言程序的注释信息。在程序的任何地方都可以加上注释的内容对程序的功能进行注解和说明。在 QC（QUICK C）中表示注释有两种方法，一种是用“/*”和“*/”将注释的信息括起来；另一种是在注释信息行的前面用“//”开头，如例 1.2 中的表示。可以看出，例 1.1 这个程序只有一个 main()函数，函数中只有一条“printf()”函数调用语句，它的功能是输出引号内除“\n”以外的所有字符。“\n”是换行符，作用是在输出“C program _1”这串字符后进行回车换行。

【例 1.2】含有两个程序块的 C 语言程序结构。

```
// C program _2
// main function
#define N 10                                    /*C 语言的宏定义  */
main( )                                         /*主函数块  */
{
  int I,p;
  p=0;
  for(i=1;i<=N;i++)
     p=p+f(i);                                  /*调用 f(int x)函数 */
  printf("p=1+(1+2)+(1+2+3)+...=%d\n",p);
}
// f sub function                               /*f(int x)子函数块 */
int f(int x)
{
  int m,y;
  for(y=0,m=1;m<=x;m++)                         /*功能:循环求和 */
     y=y+m;
  return y;                                     /*返回 函数值    */
}
```

该程序由两个程序块组成，完成$\sum_{i=1}^{n}(\sum_{k=1}^{i}k)$的计算。

观察这两个程序，可以看出C语言程序有以下特点：

（1）C语言程序的关键字和变量习惯上使用小写的英文字母。而宏定义的“宏名”通常用大写字母。

（2）C语言程序是由一个个的语句组成，一行可以书写多个语句，每个语句均由分号“;”结尾，因此分号是C语句的“终止符”或“分隔符”。

（3）C语言程序用大括号“{ }”表示程序的范围和层次关系。每个函数块需要用一对大括号括起来，同样一段复合语句也需要用大括号括起来。

（4）程序中的注解说明信息，可用“/*...*/”或“//”进行注释。

1.6.1 库和链接

从技术上讲，程序员可以仅用自己编制的函数构成实用的完整程序，但在实际中，这种情况十分罕见。在C语言的实际定义中，未提供任何实施输入/输出（I/O）操作的方法，因此，大多数C语言程序都调用C语言标准库中的各种函数。

C语言编译程序都带有标准函数库，由此完成最常见的任务。ANSI C语言标准定义了标准库中必须包括的函数的最小集合。读者的标准库中，可能包括ANSI C语言标准并未规定的图形函数等。

在C语言的某些实现中，函数库以单一的大文件形式出现，在C语言的另一些实现中，函数库以若干小文件的形式出现，提高了效率和实用性。

C语言编译程序的实现者已经编写了大部分常见的通用函数。当调用一个别人编写的函数时，编译程序“记忆”它的名字。随后，链接程序（Linker）把用户编写的程序同标准函数库中找到的目标码结合起来，这个过程称为链接（Link），某些C编译程序带有自己的链接程序，有些则使用操作系统提供的标准链接程序。

保存在函数库中的是可重定位（Relocatable）的函数，意味着其中机器码指令的内存地址并未绝对确定，只有偏移量是确定的。把程序与标准函数库中的函数相链接时，内存偏移量被用来产生实际地址。某些技术手册和参考书中更为详细地讲解了这一处理过程，但对于用C语言编程而言，读者并不需要进一步理解重定位的实际操作。

编程所需的许多函数都在标准库中，作为构成程序的基本构件。程序员自己编写并反复使用的函数，也可以放在一个库中。有些编译程序允许用户向标准库中插入自己的函数，有些则要求程序员另建一个库，无论如何，库中的代码都可以反复使用。

如前述，ANSI C语言标准只定义了一个最小标准库。许多编译程序提供的库都比ANSI C语言定义的标准库更丰富，由于冗余，ANSI C语言没有定义原始UNIX C语言的某些函数。

1.6.2 分别编译

多数短程序都可以完全放进一个源文件，然而，随着程序长度的增加，编译时间也大幅度增加（编译时间的增加常使人无法忍耐），因此，C语言允许将一个程序分解为若干块，装入若干文件，每一个文件可单独编译。一旦所有的文件编译完毕，就可以将它们与函数库中

的函数链接，形成完整的目标码。分别编译的优点是，当一个文件中的代码改变以后，不必重新编译全部程序。除最简单的项目之外，这种办法都能节省大量时间。

1.6.3　编译 C 语言程序

编译 C 语言程序包括以下 3 步：

（1）程序设计

（2）程序编译

（3）程序与所需要的库函数链接

某些编译程序提供一个集成开发环境，其中包括编辑程序。另一些编译程序没有集成开发环境，程序员必须用独立的编辑程序编程。编译程序只接受输入的标准文本文件。用某些字处理软件产生的文件包含控制码和非打印字符，编译程序将拒绝接受。

读者编译自己编写的源程序时，所用方法依实际使用的编译程序而定。同样，链接的实现也随编译程序和系统环境有所不同，详情见用户手册。

1.6.4　C 语言的内存映象

一个已完成编译的 C 语言程序取得并使用 4 块在逻辑上不同且用于不同目的的内存区域，如图 1-1 所示。第一块区域含有程序代码，相邻的一块内存区域存放全局变量，其他两块分别是栈（Stack）和堆（Heap）。“栈”用来处理程序运行的许多事务，保存函数调用时的返回地址、函数的变量、局部变量以及 CPU 的当前状态。“堆”是一个自由内存区域，程序可利用 C 语言的动态分配函数，由此取得用于链接表和树等结构所需要的内存。

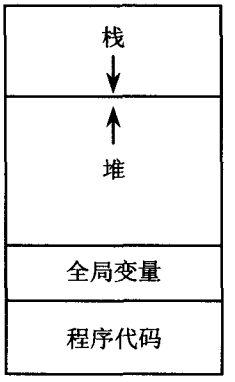

图 1-1　一个 C 语言程序在内存中的映像

程序的实际布局随编译程序和环境的不同而有所变化。由于 8086 体系的分段内存结构，多数适用于 8086 系列处理器的编译程序都有 6 种组织内存的方法。虽然 4 块区域的实际物理布局随 CPU 类型和编译程序的实现不同而有所不同，但图 1-1 仍然从概念上描述了 C 语言程序在内存中的形态。

1.7　术语

1. 源代码（Source Code）

用户可阅读的程序文本，通常称为程序，源代码是 C 语言编译程序的输入。

2. 目标码（Object Code）

由源代码转换而来的机器码，计算机可直接读入并执行。目标码是链接程序的输入。

3. 链接程序（Linker）

可把分别编译的函数链接为完整程序的程序，能够把 C 语言的标准库函数与程序人员编