21世纪高等学校计算机规划教材

21st Century University Planned Textbooks of Computer Science

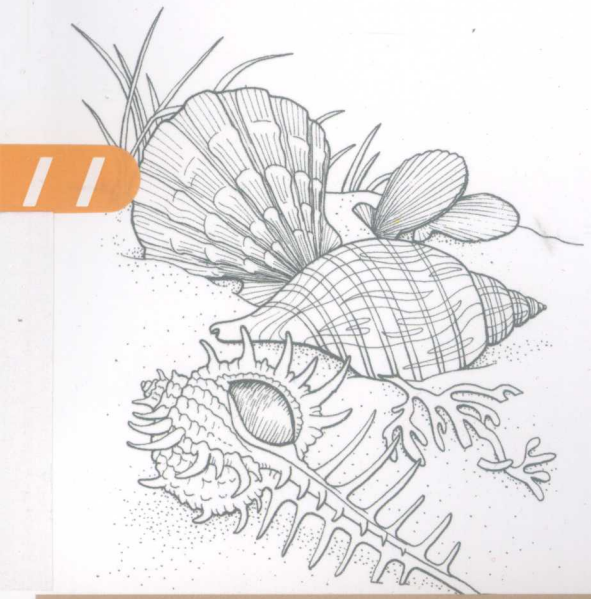
C语言程序设计

The C Language Programming

贾小军 主编

顾国松 刘锦萍 潘云燕 副主编

- 体系完整、内容详实
- 精选实例、精讲算法
- 循序渐进、深入浅出



高校系列

人民邮电出版社

POSTS & TELECOM PRESS

21世纪高等学校计算机规划教材

21st Century University Planned Textbooks of Computer Science

C语言程序设计

The C Language Programming

贾小军 主编

顾国松 刘锦萍 潘云燕 副主编

高校系列

人民邮电出版社

北京

图书在版编目（CIP）数据

C语言程序设计 / 贾小军主编. -- 北京 : 人民邮电出版社, 2014.2
21世纪高等学校计算机规划教材
ISBN 978-7-115-33833-4

Ⅰ. ①C… Ⅱ. ①贾… 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(2013)第320704号

内 容 提 要

C 语言是目前国内外使用最为广泛的程序设计语言之一，是高等学校计算机语言类课程都在讲授的重要的基础课内容。本书以程序设计为主线，循序渐进、突出重点、注重案例的编写方法，系统地讲授了 C 语言的基本语法和程序设计方法，内容包括 C 语言概述、程序设计基础、程序控制结构、数组、函数、指针、编译预处理、结构体与共用体、位运算、文件等。每章后面均配有大量的习题及参考答案，附录中还介绍了 ASCII 码、运算符优先级和结合性、常用库函数、常用语法等内容。

本书可作为高等学校理工类专业“C 语言程序设计”课程的教学用书，也可作为参加国家计算机等级考试（二级 C 语言）的辅导用书，或作为计算机程序设计爱好者的自学参考书。为配合教学，本书配有 PPT 教学课件，并有配套的《C 语言程序设计实验教程》供读者参考。

◆ 主　　编　贾小军
副 主 编　顾国松　刘锦萍　潘云燕
责任编辑　武恩玉
责任印制　彭志环　杨林杰

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京艺辉印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：17　　2014 年 2 月第 1 版
字数：404 字　　2014 年 2 月北京第 1 次印刷

定价：35.00 元

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第 0021 号

前　言

C 语言是目前国内外使用最为广泛的程序设计语言之一，是高等学校计算机语言类课程都在讲授的重要的基础课内容，具有语法丰富、结构清晰、使用方便、执行效率高以及可移植性好等特点，是高等学校计算机语言类课程的首选。

本书以程序设计为主线，循序渐进、突出重点、注重案例的编写方法，系统地讲授了C 语言的基本语法和程序设计方法，全书共 11 章。第 1 章为 C 语言概述，包括 C 语言简介、开发环境等内容。第 2 章为 C 语言程序设计基础，包括数据类型、常量与变量、函数与表达式、数据运算、输入/输出语句等内容。第 3 章为程序控制结构，包括顺序结构、分支结构及循环结构等内容。第 4 章为数组，包括数组定义、一维数组、二维数组、字符数组等内容。第 5 章为函数，包括函数概述、参数传递方式、调用方法、嵌套与递归等内容。第 6 章为指针，包括指针概念、指针与变量、指针与数组、指针与字符串、指针数组与指向指针的指针等内容。第 7 章为编译预处理，包括宏定义、文件包含及条件编译等内容。第 8 章为结构体与共用体，包括基本概念、结构体数组、结构体指针变量、结构体函数、链表、共用体、枚举类型等内容。第 9 章为位运算，包括位运算概述、位运算符和位运算、位段等内容。第 10 章为文件，包括文件概述、文件指针、文件操作、文件读写等内容。第 11 章为 C 语言程序设计综合案例，介绍 C 语言开发应用程序的基本步骤和方法。本书每章后面均配有大量的习题及参考答案（教材习题参考答案在实验教程中），附录中还介绍了 ASCII 码、运算符优先级和结合性、常用库函数、常用语法提要等内容。

本书在讲授 C 语言过程中既注重语法知识的详细阐述，又注重案例算法的精细剖析；既注重培养读者设计程序的能力，又提倡养成良好的程序设计风格习惯。本书同时兼顾了全国计算机等级考试（二级 C 语言）的大纲要求，涵盖了大纲要求的所有知识点，提供了丰富的案例。这些案例的源程序代码均在 C Free 5.0 环境下调试通过，可直接使用。并且本书提供了所有案例的数据及运行结果，方便读者参考及查阅。

本书可作为高等学校理工类专业“C 语言程序设计”课程的教学用书，也可作为参加国家计算机等级考试（二级 C 语言）的辅导用书，或作为计算机程序设计爱好者的自学参考书。为配合教学，本书配有 PPT 教学课件，并有配套的《C 语言程序设计实验教程》供读者参考。

本书是多位老师在多年 C 语言教学与程序设计实践的基础上，结合多次编写相关讲义和教材的经验总结而成，同时本书在编写过程中也参考了大量书籍，得到了许多同行的帮助与支持，在此向他们表示衷心的感谢。本书由贾小军任主编，顾国松、刘锦萍、潘云燕任副主编。第 1 章由方玫编写，第 2 章、附录由许巨定编写，第 3 章、第 7 章由顾国松编写，第 4 章、第 10 章由贾小军编写，第 5 章、第 11 章由骆红波编写，第 6 章由潘云燕编写，第 8 章、第 9 章由刘锦萍编写。全书由贾小军博士统稿。

由于时间仓促，书中难免存在不足或者遗漏之处，恳请广大读者提出修改建议。

编　者

2013 年 10 月

目 录

第 1 章 C 语言概述 …… 1
1.1 计算机语言及程序的发展 …… 1
1.1.1 计算机语言的历史 …… 1
1.1.2 C 语言的历史 …… 2
1.1.3 C 语言的特点 …… 3
1.2 从 Hello 程序认识 C 语言 …… 3
1.2.1 程序的实例解析 …… 4
1.2.2 简单例题的引申 …… 4
1.2.3 C 语言程序的组成 …… 6
1.3 C 语言程序的运行步骤和开发环境 …… 6
1.3.1 C 语言程序的运行步骤 …… 6
1.3.2 C-Free 5.0 简介 …… 7
1.3.3 Visual C++ 6.0 简介 …… 10
习题 …… 13
第 2 章 C 语言程序设计基础 …… 14
2.1 数据类型 …… 14
2.1.1 数据的分类 …… 14
2.1.2 整型数据 …… 15
2.1.3 实型数据 …… 16
2.1.4 字符型数据 …… 16
2.2 常量与变量 …… 17
2.2.1 常量和符号常量 …… 17
2.2.2 变量和变量的定义 …… 19
2.3 运算符与表达式 …… 20
2.3.1 运算符的分类 …… 20
2.3.2 算术运算符和算术表达式 …… 21
2.3.3 赋值运算符和赋值表达式 …… 23
2.3.4 条件运算符和条件表达式 …… 24
2.3.5 逗号运算符和逗号表达式 …… 24
2.3.6 关系运算符和关系表达式 …… 25
2.3.7 逻辑运算符和逻辑表达式 …… 26
2.4 数据运算 …… 27
2.4.1 自动类型转换 …… 27
2.4.2 赋值类型转换 …… 28
2.4.3 强制类型转换 …… 28
2.5 输入输出语句 …… 28
2.5.1 字符输出函数（putchar()函数） …… 29
2.5.2 字符输入函数（getchar()函数） …… 29
2.5.3 格式输出函数（printf()函数） …… 30
2.5.4 格式输入函数（scanf()函数） …… 32
习题 …… 35
第 3 章 程序控制结构 …… 37
3.1 算法与程序流程图 …… 37
3.1.1 程序的灵魂 …… 37
3.1.2 程序流程图 …… 37
3.1.3 3 种基本结构 …… 38
3.1.4 程序语句 …… 39
3.2 顺序结构 …… 40
3.3 分支结构 …… 43
3.3.1 if 语句 …… 44
3.3.2 switch 语句 …… 49
3.3.3 分支结构的嵌套 …… 50
3.4 循环结构 …… 52
3.4.1 for 循环 …… 53
3.4.2 while 循环 …… 57
3.4.3 do...while 循环 …… 59
3.5 break 和 continue 语句 …… 60
3.6 C 语言程序控制应用实例 …… 65
习题 …… 72
第 4 章 数组 …… 74
4.1 数组定义及分类 …… 74
4.2 一维数组 …… 74
4.2.1 一维数组的定义 …… 74
4.2.2 一维数组的引用 …… 75

4.2.3 一维数组的初始化 …… 76
4.2.4 一维数组的应用 …… 78
4.3 二维数组 …… 79
4.3.1 二维数组的定义 …… 79
4.3.2 二维数组的引用 …… 81
4.3.3 二维数组的初始化 …… 82
4.3.4 二维数组的应用 …… 83
4.4 字符数组 …… 85
4.4.1 字符数组的定义 …… 85
4.4.2 字符数组的引用 …… 85
4.4.3 字符数组的初始化 …… 85
4.4.4 字符数组与字符串的关系 …… 86
4.4.5 字符数组的输入与输出 …… 87
4.4.6 字符串处理函数 …… 89
4.4.7 字符数组的应用 …… 92
4.5 数组应用实例 …… 93
4.5.1 统计 …… 94
4.5.2 排序 …… 96
4.5.3 查找 …… 99
4.5.4 其他应用 …… 101
习题 …… 102
第 5 章 函数 …… 104
5.1 函数概述 …… 104
5.1.1 模块化程序设计 …… 104
5.1.2 使用函数的好处 …… 105
5.1.3 函数的基本用法 …… 106
5.2 函数的一般形式 …… 107
5.2.1 函数的定义 …… 107
5.2.2 函数原型的声明 …… 111
5.3 函数的参数传递方式 …… 112
5.3.1 形参与实参 …… 112
5.3.2 多个参数的传递 …… 113
5.3.3 值传递方式 …… 114
5.4 函数的调用 …… 115
5.4.1 函数调用的一般形式 …… 115
5.4.2 函数的调用过程 …… 115
5.4.3 函数的调用方式 …… 117
5.5 函数的嵌套与递归 …… 117
5.5.1 函数的嵌套调用 …… 117
5.5.2 函数的递归调用 …… 119
5.6 变量的作用域与存储类别 …… 123
5.6.1 变量的作用域 …… 123
5.6.2 变量的存储类别 …… 126
5.7 函数应用举例 …… 129
习题 …… 132
第 6 章 指针 …… 134
6.1 指针的概念 …… 134
6.2 指针变量与变量 …… 135
6.2.1 指针变量的定义 …… 135
6.2.2 指针变量的引用和运算 …… 136
6.2.3 指针变量作为函数参数 …… 140
6.3 指针与数组 …… 141
6.3.1 指向数组的指针 …… 142
6.3.2 通过指针变量访问数组元素 …… 142
6.3.3 数组作为函数参数 …… 145
6.3.4 指向多维数组的指针 …… 153
6.4 指针与字符串 …… 156
6.4.1 指针与字符串 …… 156
6.4.2 字符串指针作为函数参数 …… 157
6.4.3 使用字符串指针变量与字符数组的区别 …… 159
6.5 指针数组和指向指针的指针 …… 160
6.5.1 指针数组 …… 160
6.5.2 指向指针的指针 …… 161
6.5.3 指针的其他用法 …… 162
6.5.4 与指针有关的用法小结 …… 165
习题 …… 165
第 7 章 编译预处理 …… 166
7.1 宏定义 …… 166
7.1.1 无参数的宏定义 …… 166
7.1.2 带参数的宏定义 …… 170
7.2 文件包含 …… 174
7.2.1 “文件包含”命令的一般形式 …… 174
7.2.2 “文件包含”命令的应用 …… 174

7.3 条件编译 …… 176
习题 …… 178
第 8 章 结构体与共用体 …… 179
8.1 引例 …… 179
8.2 结构体类型与结构体变量 …… 180
8.2.1 结构体类型的定义 …… 180
8.2.2 结构体变量的定义 …… 181
8.2.3 结构体变量的初始化 …… 182
8.2.4 结构体变量的引用 …… 183
8.3 结构体数组 …… 185
8.3.1 结构体数组的定义和初始化 …… 185
8.3.2 结构体数组的使用 …… 186
8.4 结构体指针变量 …… 188
8.4.1 指向结构体变量的指针 …… 188
8.4.2 指向结构体数组的指针 …… 190
8.5 结构体与函数 …… 191
8.5.1 结构体变量作函数参数 …… 191
8.5.2 指向结构体变量的指针作函数参数 …… 193
8.5.3 返回结构体的函数调用 …… 194
8.6 动态存储分配 …… 195
8.7 链表 …… 197
8.7.1 链表的概念 …… 197
8.7.2 链表的基本操作 …… 198
8.8 共用体 …… 202
8.8.1 共用体的定义 …… 202
8.8.2 共用体变量的定义 …… 203
8.8.3 共用体变量的引用 …… 203
8.9 枚举类型 …… 205
8.9.1 枚举类型、枚举类型变量的定义 …… 206
8.9.2 枚举类型变量的赋值和使用 …… 206
8.10 用户自定义类型 …… 208
习题 …… 209
第 9 章 位运算 …… 210
9.1 位运算概述 …… 210
9.1.1 计算机中数据的表示 …… 210
9.1.2 补码的求法 …… 211
9.2 位运算符和位运算 …… 211
9.2.1 位运算操作 …… 212
9.2.2 位运算操作举例 …… 218
9.3 位段 …… 220
习题 …… 222
第 10 章 文件 …… 223
10.1 C 文件概述 …… 223
10.1.1 数据文件的存储形式 …… 223
10.1.2 缓冲文件系统与非缓冲文件系统 …… 224
10.2 文件指针 …… 224
10.3 文件的打开与关闭 …… 225
10.3.1 文件的打开 …… 225
10.3.2 文件的关闭 …… 227
10.4 文件的读写 …… 228
10.4.1 字符读写函数 …… 228
10.4.2 字符串读写函数 …… 230
10.4.3 数据块读写函数 …… 232
10.4.4 格式化读写函数 …… 235
10.4.5 整数读写函数 …… 238
10.5 文件的定位 …… 239
10.6 文件检测函数 …… 241
习题 …… 242
第 11 章 C 语言程序设计综合案例 …… 243
11.1 系统功能 …… 243
11.2 设计思路 …… 244
11.3 代码实现 …… 245
11.4 运行结果 …… 252
11.5 小结 …… 253
习题 …… 253
附录 A ASCII 码字符表 …… 254
附录 B 运算符的优先级和结合性 …… 255
附录 C C 语言常用库函数 …… 257
附录 D C 语言常用语法提要 …… 262

第1章 C语言概述

本章首先概要地介绍了C语言的发展史及特点，然后通过一个实例程序让读者初步认识C语言，最后介绍了C语言运行的步骤及开发环境。

1.1 计算机语言及程序的发展

1.1.1 计算机语言的历史

当今使用的计算机语言有上百种，大致可分机器语言、汇编语言、高级语言。同样一个语句用机器语言、汇编语言和C语言分别表示如表1-1所示。

表1-1　　一个语句的3种表示

编程语言	表示形式
机器语言	a1 1c a0 04 08 83 c0 01 a3 18 a0 04 08
汇编语言	mov　0x804a01c,%eax add　$0x1,%eax mov　%eax,0x804a018
C语言	a=b+1

显然，从程序员的角度看高级语言比机器语言和汇编语言更令他们满意。对于计算机来说，计算机只能对数字做运算，符号、声音、图像在计算机内部都要用数字表示，指令也不例外，表1-1中的机器语言完全由十六进制数字组成，如a1即为10100001。

1. 机器语言

机器语言是用二进制代码（0和1）表示的，能直接识别和执行的一种机器指令的集合。具有灵活、直接执行和速度快等特点。但是，不同型号的计算机其机器语言是不相通的。同时，用机器语言编程的工作很繁琐，程序员需要熟记所用计算机的全部指令代码和代码的含义，还得记住编程过程中每步所使用的工作单元处在何种状态。编写程序花费的时间往往是实际运行时间的几十倍或几百倍。而且，编出的程序全是0和1的指令代码，直观性差，容易出错。现在，除了计算机生产厂家的专业人员外，绝大多数的程序员已经不再去学习机器语言了。

但机器语言是计算机唯一能直接执行的语言。执行其他语言，需要“翻译”成机器语言

才能被计算机“读懂”。如图 1-1 所示，程序员所见的汇编语言或高级语言，在计算机的“眼中”都是被翻译后的由 0 或 1 组成的代码。

2. 汇编语言

机器语言编程麻烦，并且编写出来的程序很不直观，容易出错，于是有了汇编语言，把机器语言中一组一组的数字用助记符（Mnemonic）表示，直接用这些助记符（如表 1-1 中的 mov、add）写出汇编程序，然后让汇编器（Assembler）去查表把助记符替换成数字，也就把汇编语言翻译成了机器语言。从表 1-1 的例子可以看出，汇编语言和机器语言的指令是一一对应的，汇编语言有三条指令，机器语言也有三条指令，汇编器就是做一个简单的替换工作。

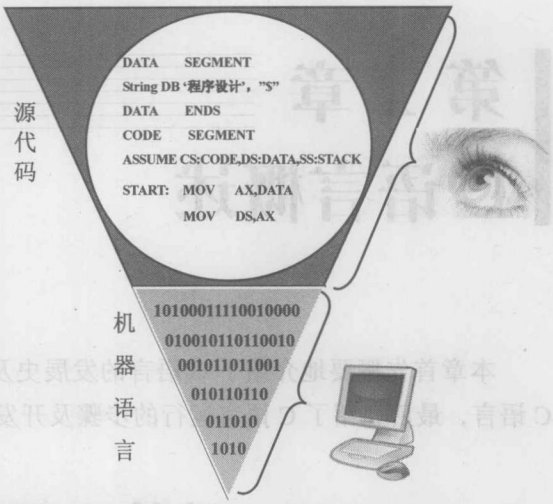


图 1-1 源代码需要“翻译”成机器语言

3. 高级语言

随着汇编语言的出现，计算机的用途迅速扩大，但是即使完成最简单的任务仍然需要编写许多指令。为了加速程序开发的过程，出现了用一条语句能够完成一定意义任务的高级语言。

从表 1-1 中的例子可以看出，C 语言的语句和低级语言的指令之间不是简单的一一对应关系，一条 a=b+1;语句要翻译成三条汇编或机器指令，这个过程称为编译（Compile），由编译器（Compiler）来完成，显然编译器的功能比汇编器要复杂得多。用 C 语言编写的程序必须经过编译转成机器指令才能被计算机执行，编译需要花一些时间，这是用高级语言编程的一个缺点，然而更多的是优点。首先，用 C 语言编程更容易，写出来的代码更紧凑，可读性更强，出了错也更容易改正。其次，C 语言是可移植的（Portable）或者称为平台无关的（Platform Independent）。

C 语言是功能强大，使用范围最广的高级语言之一。

1.1.2 C 语言的历史

C 语言是面向过程的较好的结构化程序设计语言，其历史颇为有趣。C 语言的“先人”可以追溯到 ALGOL 60 语言。1963 年，剑桥大学将 ALGOL 60 语言发展成为 CPL（Combined Programming Language）语言。但 CPL 的最大缺点就是它太大了，以至于不能在很多应用程序中实现。到 1967 年，剑桥大学的 Martin Richards 在访问 MIT 时，对 CPL 语言进行了简化，推出了 BCPL（Basic CPL）语言。

1970 年，美国贝尔实验室的 Ken Thompson 发现 BCPL 太慢了，而且缺乏运行时的支持，他将 BCPL 进行了实验性的修改，并为它起了一个有趣的名字——B 语言，意思是将 CPL 语言进行压缩，提炼出它的精华。B 是 BCPL 的一个简化版，而且被设计成专门用来进行系统编程，但它依然不能满足程序员们的要求：它的字符处理机制太丑陋了，而且浮点数运算被实现得并不是很理想，处理指针时开销太大。

在 1972 年，当 PDP-11 进入贝尔实验室时，Ken Thompson 的同伴，同样在贝尔实验室的 Dennis M. Ritchie，在从 BCPL 中抽取了一些共性放入 B 语言中，对 B 进行了改进，并且

在里面加入了数据类型，他把这个扩展的语言称为 NB（New B）。随后，他又继续对新语言进行大量修改，似乎可以用新的名字来重新命名它了，于是取了 BCPL 的第二个字母作为这种语言的名字，这就是 C 语言。

1973 年，Ritchie 完成了 C 语言核心，并用 C 重写了整个 UNIX 内核。为了使 UNIX 操作系统推广，1977 年 Dennis M. Ritchie 发表了不依赖于具体机器系统的 C 语言编译文本《可移植的 C 语言编译程序》。

1978 年，Brian W. Kernighian 和 Dennis M. Ritchie 出版了长久不衰的经典之作——《The C Programming Language》，把 C 从贝尔实验室推向世界，从而使 C 语言成为目前世界上流行最广泛的程序设计语言之一。

1.1.3　C 语言的特点

C 语言具有如下特点。

（1）简洁紧凑、灵活方便。C 语言一共只有 32 个关键字（关键字都为小写），9 种控制语句，程序书写自由，它把高级语言的基本结构和语句与低级语言的实用性结合起来，输出程序工作量少。

（2）运算符丰富。C 的运算符包含的范围很广泛，共有 34 个运算符。C 语言把括号、赋值、强制类型转换等都作为运算符处理。从而使 C 的运算类型极其丰富，可以实现在其他高级语言中难以实现的运算。

（3）C 是结构化语言。结构化语言的显著特点是代码及数据的分隔化，即程序的各个部分除了必要的信息交流外彼此独立。这种结构化方式可使程序层次清晰，便于使用、维护以及调试。

（4）数据结构丰富。

（5）C 语法限制不太严格、程序设计自由度大。

（6）C 语言允许直接访问物理地址，可以直接对硬件进行操作。

（7）C 语言程序生成代码质量高，程序执行效率高。

（8）C 语言适用范围大，可移植性好。

1.2　从 Hello 程序认识 C 语言

下面先介绍几个简单的 C 语言程序，并从中了解 C 语言的基本构成。

【例 1.1】　请打印出以下内容：Hello Kitty。

```
#include <stdio.h>                                    /*包含标准库的信息*/
int main()                                  /*定义名为 main()的函数，它不接受参考值*/
{                                             /*main()函数的语句都被包含在花括号中*/
   printf("Hello Kitty\n");          /*main()函数调用库函数 printf()显示打印的序列*/
   return 0;                                                /*其中\n 代表换行符*/
}
```

C 程序最大的特点就是所有的程序都是用函数来装配的。通常情况下，函数的命名是没有限制的，但 main()是一个特殊的函数称为主函数，是所有程序运行的入口，无论它所在何处。C 语言程序中有且只有一个 main()函数。

1.2.1 程序的实例解析

main()函数通常会调用其他函数来帮助完成某种工作，被调用的函数可由程序员自己编写，也可以是已有的函数库。程序的第一行：

```
#include <stdio.h>
```

作用在于告诉编辑器在本程序中包含标准输入/输出库的信息。stdio.h 是 C 编译系统提供的一个文件名，stdio 是“standard input & output”的缩写，即有关标准输入输出的信息；.h 表示是 head 文件即为头文件。

函数之间进行数据交换方法之一就是调用函数向被调用函数提供一个值，称为参数，函数名后的圆括号( )将参数列表包含。如本例中的 main()函数不需要任何参数，可以用空参数表( )表示。

函数中所有的语句用一对花括号{}包含，本例中的 main()函数仅包含一条语句：

```
printf("Hello Kitty\n");
```

其作用是在屏幕上输出 Hello Kitty 并将光标换行。语句中“Hello Kitty\n”作为参数调用 printf()函数。printf()是一个打印输出的库函数。\n 的作用是使光标换行。

试将该语句分别做如下改动。

（1）printf("Hello Kitty\n);

缺少一个引号（"）后程序编译出错，printf()函数有特定的格式，在第 2 章中有其详细解释。

（2）printf("Hello Kitty");

将\n 去掉，编译并未出错，读者发现光标的停留位置变化了，没有换行。

（3）将整个程序改为：

```
#include <stdio.h>
int main()
{
   printf("Hello");
   printf("  Kitty");
   printf(" \n");
   return 0;
}
```

这段程序和之前的输出结果相同。请读者试试在双引号中任意输您想输出的内容。

1.2.2 简单例题的引申

【例 1.2】 求两数之和。

```
#include <stdio.h>
int main( )
{
   int  a,b,sum;                          /*定义整型变量 a,b,sum*/
   a=110;  b=120;                           /*对变量 a,b 赋值*/
   sum = a+b;                               /*计算 sum 的值*/
   printf("sum=%d",sum);                  /*将结果 sum 以整型的方式输出*/
   return 0;
}
```

程序中如/*定义整型变量 a,b,sum*/是对程序的注释，包含在/*与*/之间的内容将被编译器忽略。程序的注释一般在某行的最末尾或另起一行。注释的作用是使程序更易于理解。

在 C 语言中，变量必须先定义后使用。定义通常放在函数的起始处，在任何可执行语句之前。定义用于说明变量的属性。如 int a,b,sum；即说明了 a、b、sum 3 个变量都为整型变量。程序的第五、六行：

```
a=110; b=120;
sum = a+b;
```

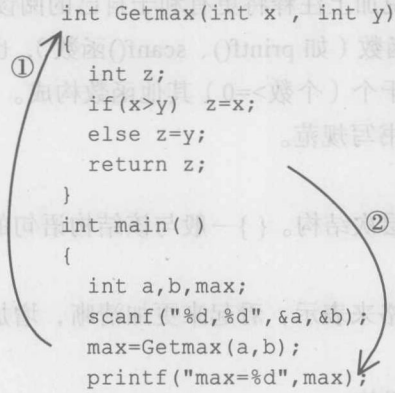
它们为变量 a、b、sum 赋初值。各条语句以分号（;）做结束。变量的概念在本书的第 2 章中详细介绍。变量就像钱包，里面可以装上千元也可以装几毛钱，其值可以不断的变化。读者可以将程序中变量的值改变，观察程序变化。程序的第七行：

```
printf("sum=%d",sum);
```

printf()是一个通用输出格式化函数，其中每个百分号（%）表示其他的参数之一进行替换的位置，并制定打印格式，%d 为整型参数，该位置由 sum 的值替代，即输出 sum=230，除了%d 的位置按参数 sum 的值输出外，printf()函数中双引号（“”）中的内容遵循你给什么内容它输什么内容的原则。

【例 1.3】　通过调用子函数，求两数之中的较大者。

```
#include <stdio.h>
int Getmax(int x , int y)
{
  int z;
  if(x>y)  z=x;
  else z=y;
  return z;
}
int main( )
{
  int a,b,max;
  scanf("%d,%d",&a,&b);
  max=Getmax(a,b);
  printf("max=%d",max);
  return 0;
}
```



本程序包含两个函数：被调用函数 Getmax()和主函数 main()。例题中的函数如 printf()、scanf()等都是函数库中提供的函数。除此之外，我们还能自己动手来编写一些函数。如本例中的 Getmax()函数的作用就是返回两数中的较大值。main()也是函数，不过身份有些特殊，首先它是程序的入口；其次，如果将所有的函数当成团队中的个人，那么 main()函数更像是老板，Getmax()函数就是员工，main()函数调用 Getmax()完成任务。

虽然程序的第一行不是 main()函数，但是程序仍从 main()函数开始执行，因为 main()函数为程序的入口。进入 main()函数后，按顺序执行程序，运行至程序第 13 行：

```
max=Getmax(a,b);
```

调用函数 Getmax()，并将 a、b 的值传递给 x、y，若我们从键盘上输入 12、30，则 x、y 的值为 12、30。

```
int Getmax(int x , int y)
```

Getmax()函数结果通过 return 语句返回给 main()函数。即程序调用 Getmax()函数，返回主函数的被调用点处，如程序中箭头所指。

main()函数中 scanf()函数的功能是执行格式化输入。程序第 12 行：

```
scanf("%d,%d",&a,&b);
```

作用为从键盘上输入 a 和 b 的值，&a 为 a 的地址，即从键盘上输入的内容存放到 a 和 b 所在的地址中。双引号“”中%d 是对输入参数的格式化说明，即要求输入的 a 和 b 的值为整型。

1.2.3 C 语言程序的组成

通过学习以上 3 个简单的 C 语言程序，可以分析出 C 语言程序的基本特点、基本组成结构，常见语句的书写模式，可以总结出以下特点。

（1）一个 C 语言源程序有且只有一个 main()函数，即主函数。无论 main()函数的位置在何处，程序执行都从 main()函数开始，即 main()函数是程序的入口。

（2）源程序中可以有预处理命令（如 include 命令），预处理命令通常放在源文件或源程序的最前面。

（3）每一条语句都必须以分号结尾。除此之外，预处理命令、函数头及花括号“}”之后不能加分号。

（4）C 程序中可通过/*……*/对程序语句进行注释，注释内容被编译器忽略。读者为了方便阅读可以对复杂的程序加上必要的注释，对初学者来说加上注释将更有利于自己的阅读。

（5）C 程序由函数构成。函数可以是系统提供的库函数（如 printf()、scanf()函数），也可以自己编写函数。一个 C 源程序由一个 main()函数和若干个（个数>=0）其他函数构成。

除此之外，编写 C 语言程序时还需遵循如下程序的书写规范。

（1）一个说明或一个语句占一行。

（2）用{ }括起来的部分，通常表示了程序的某一个层次结构。{ }一般与该结构语句的一个字母对齐，并独占一行。

（3）当语句比上一条语句层次低时可用缩进若干空格来表示，看起来更加清晰，增加程序的可读性。

在编程时应力求遵循这些规则，以养成良好的编程风格。

1.3 C 语言程序的运行步骤和开发环境

1.3.1 C 语言程序的运行步骤

前面介绍了几个用 C 语言编写的程序，C 语言是高级语言源程序，必须用编译程序的软件“翻译”计算机能够识别的由 0 或 1 组成的二进制命令。这种二进制形式程序我们称为“目标程序（object program）”。然后，将该目标程序与系统的库函数及其他目标程序连接，形成可执行的目标程序。

完成 C 语言程序的编辑后，如何运行呢？其调试和运行步骤如图 1-2 所示。

图中虚线表示程序调试方向，但编译、连接发生错误或程序运行结果不正确时，需要检查源程序，不断调试，直到程序结果正确为止。

我们可以将前面的程序修改，运行程序。如果将某行的分号（;）去掉，则编译不通过；如果将例题中的 printf 改为 print，并分别执行编译和连接，发现编译后未发生错误，而连接

后发生错误，即检测输入语句 printf 是否发生错误是在连接输入输出库（stdio.h）才被发现。

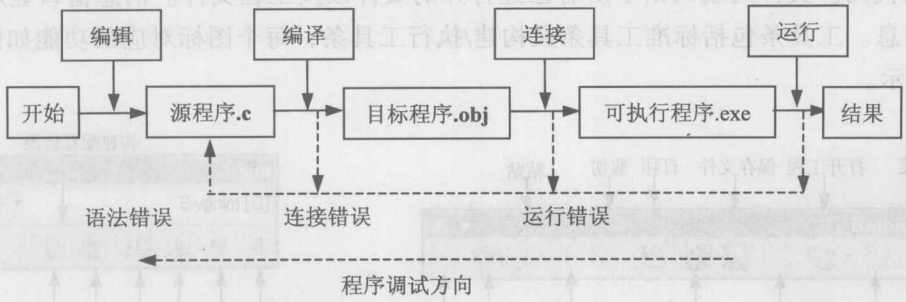


图 1-2　C 语言程序调试和运行步骤

运行 C 程序，必须用到相应的 C 编译系统，常用的编译系统有 C-Free 5.0、Visual C++ 6.0 等，接下来介绍上述两种编译系统。

1.3.2　C-Free 5.0 简介

1. 窗口信息

主界面窗口如图 1-3 所示，主要包含以下部分。

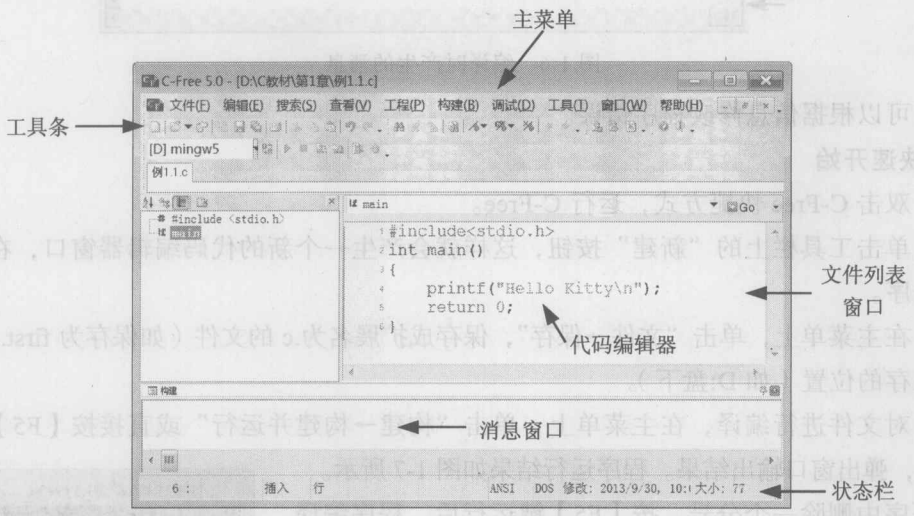


图 1-3　C-Free 主界面

- 主菜单及工具条。
- 标签栏。
- 代码编辑器。
- 符号窗口及符号工具条。
- 消息窗口。
- 文件列表窗口。

主菜单中几乎包含了所有的命令（部分命令包含在右键菜单中），工具条能够使用户方便的执行一些常用命令。标签栏列出了所有已经打开的文件，方便用户在不同文件之间进

行切换。大部分的工作在代码编辑器中进行。符号窗口和符号工具条能够帮助用户方便的寻找定位代码。文件列表列出了所有已经打开的文件以及工程文件。消息窗口显示编译、构建的信息。工具条包括标准工具条及构建/执行工具条，每个图标对应的功能如图 1-4 及图 1-5 所示。

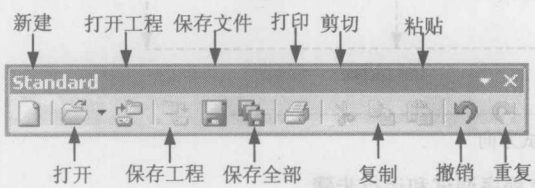


图 1-4　标准工具条

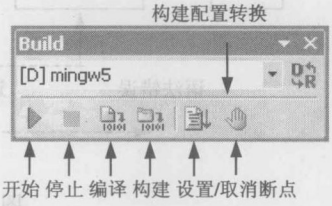


图 1-5　构建/执行工具条

消息窗口可以显示两类消息，一类是编译、构建过程产生的消息；另一类是多文件搜索之后显示的结果信息。编译过程中产生的消息如图 1-6 所示。

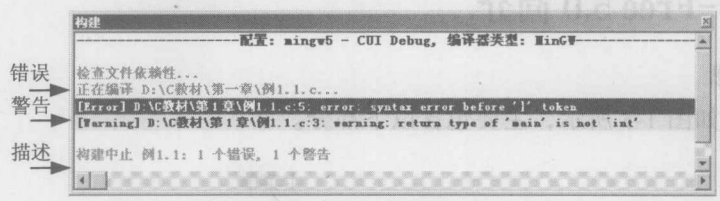


图 1-6　编译时产生的消息

用户可以根据信息修改程序错误。

2. 快速开始

（1）双击 C-Free 快捷方式，运行 C-Free。

（2）单击工具栏上的“新建”按钮，这样就会产生一个新的代码编辑器窗口，在编辑区中输入程序。

（3）在主菜单上，单击“文件→保存”，保存成扩展名为.c 的文件（如保存为 first.c 文件）及选择保存的位置（如 D:盘下）。

（4）对文件进行编译，在主菜单上，单击“构建→构建并运行”或直接按【F5】键，若程序正确，弹出窗口输出结果。程序运行结果如图 1-7 所示。

若程序中删除一个分号，按【F5】键运行后，程序编译出错，查看消息窗口的错误提示，可根据提示的错误，修改程序，直至结果正确。

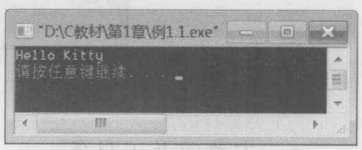


图 1-7　程序运行结果

3. 如何调试

当程序出现错误，而用户又不知程序错在何处时，可尝试使用断点调试的方法。

（1）设置/取消断点。在代码编辑器中，将光标定位到代码中的某行，单击主菜单中“调试→设置/取消断点”。快捷方法：用鼠标单击代码行前面的空白位置（如第 13 行前空白部分），如图 1-8 所示。取消断点方法相同，但必须光标停留在已设置断点的程序行（如要取消第 13 行的断点，将光标停留在第 13 行，按【F10】键）

（2）开始调试。单击主菜单中“调试→调试”，或直接按【F9】键，一旦程序开始调试，

C-Free 会自动显示“调试工具条”，如图 1-9 所示。

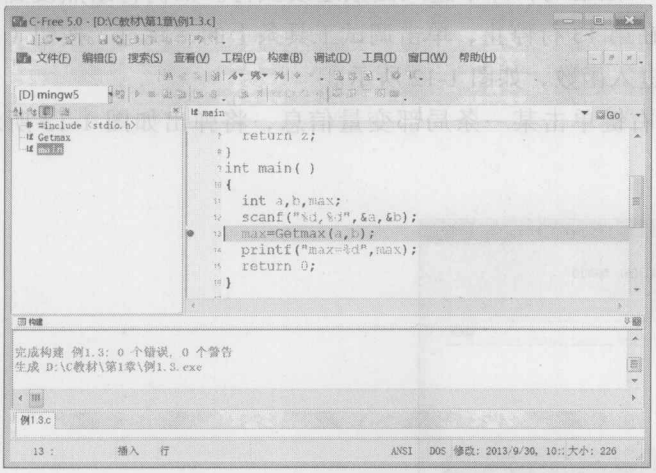


图 1-8　C-free 设置断点（【F10】）

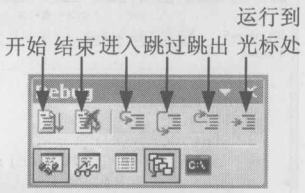


图 1-9　C-free 调试工具栏

程序开始调试，停在某一个断点处。若要继续执行程序，可单击工具条上的进入（或【F7】），跳过（或【F8】）按钮。两个按钮都为继续单步执行程序，区别在于当有函数时，单步进入（【F7】）是进入函数内部执行的，跳过（【F8】）是跳过函数继续执行，即当碰到函数时不进入函数内部单步执行，而直接得到函数的结果。

（3）退出调试。读者可以单步执行到程序结束，也可以中途退出程序。单击调试工具条上的“结束调试”按钮，程序退出调试状态，回到编辑状态。

（4）单步进入（【F7】）。当程序处于调试状态，并停在某个断点处，如果该断点处的语句是一条函数调用，单击调试工具条上的“单步进入”按钮，程序将进入这个函数，可对这个函数进行调试。

如对例 1.3 的程序，在其第 13 行设置断点，开始调试程序，程序在第 13 行停止，如图 1-10 所示。单击“进入”按钮（或【F7】），程序进入 Getmax()函数，如图 1-11 所示。

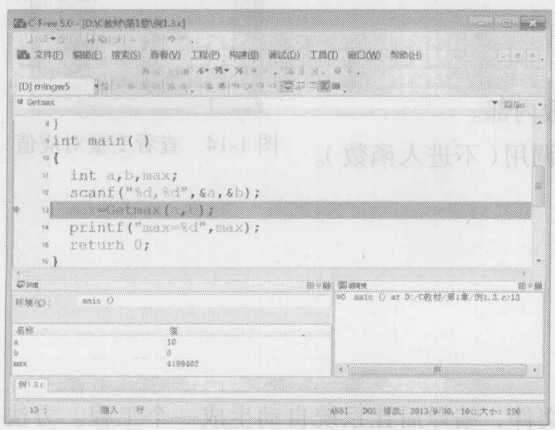


图 1-10　开始调试（【F9】）

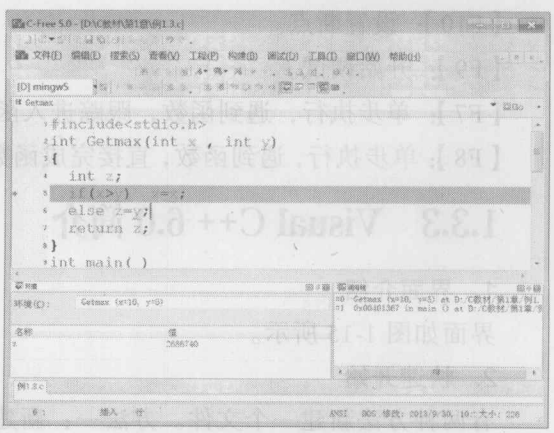


图 1-11　进入函数（【F7】）

（5）下一步（【F8】）。当程序处于调试状态，并停在某个断点处，单击调试工具条上的“跳过”按钮（或【F8】），程序执行到下一条语句，即不进入函数继续执行。如仍将断点设置在第13行，然后开始调试程序，程序在第13行停止，单击调试工具条上的“跳过”按钮（或【F8】），程序直接进入下一步，而不进入函数，如图1-12所示。

（6）查看/修改。在环境窗口中右键单击某一条局部变量信息，将弹出如图1-13所示窗口。

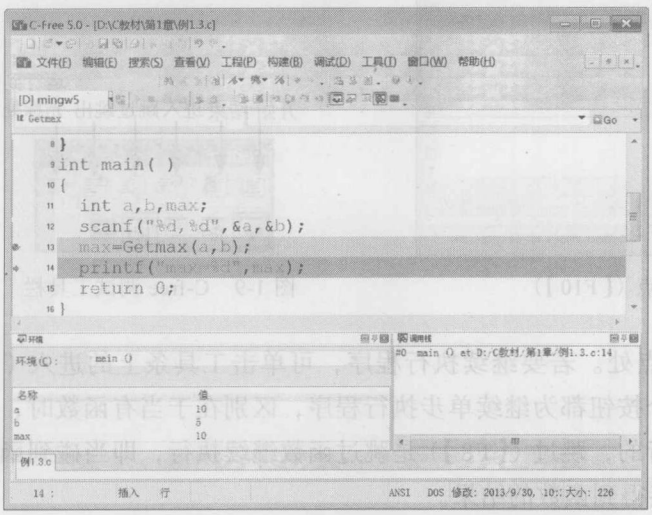


图1-12 不进入函数（【F8】）

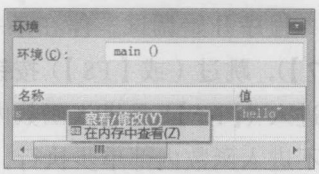


图1-13 查看变量信息

选择“查看/修改”，对于数组（将在第4章讲到）、结构或者类变量，将显示“查看”窗口；对于其他变量，将显示“修改”窗口。双击某一条局部变量信息，也可以实现同样的功能。

例如，对于图1-13的局部变量，双击“s”，由于s是字符串数组，因此将显示“查看”窗口，如图1-14所示。

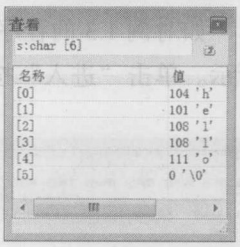


图1-14 查看变量对应值

4. C-free5.0常用快捷键

【F5】：运行程序。

【F10】：设置断点。

【F9】：开始调试。

【F7】：单步执行，遇到函数，跟踪进入函数内部。

【F8】：单步执行，遇到函数，直接完成函数调用（不进入函数）。

1.3.3 Visual C++ 6.0简介

1. 界面介绍

界面如图1-15所示。

2. 快速开始

有两种方法新建一个文件。方法一：新建文件，编译后让系统自动生成一个工程。方法二：先新建工程后新建文件。

方法一：系统自动生成工程。