



新编高等院校计算机科学与技术应用型规划教材

# C 语言程序设计

葛日波 主编



北京邮电大学出版社  
[www.buptpress.com](http://www.buptpress.com)

新编高等院校计算机科学与技术应用型规划教材

# C 语言程序设计

葛日波 主编

北京邮电大学出版社  
·北京·

## 内 容 简 介

C 语言是国内外广泛使用的计算机语言,是计算机从业人员应该掌握的一种重要的程序设计工具。

本书是针对应用型人才的培养要求,经过精心策划,准确定位,以清晰的概念、大量的图例和工程用例,深入浅出地介绍了 C 语言的基础知识、数据输入/输出、运算符和表达式、分支语句、循环语句、数组、指针、字符串、函数、复合数据类型和文件操作,同时还阐述了程序设计的思想与技术。

本书结合作者多年的教学实践和使用英文原版教材授课的经验编写而成,结构合理、逻辑性强、文字流畅、通俗易懂、便于自学,是学习 C 语言的一本好教材。

### 图书在版编目(CIP)数据

C 语言程序设计/葛日波主编. —北京:北京邮电大学出版社,2007

ISBN 978-7-5635-1620-9

I. C… II. 葛… III. C 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字(2008)第 114201 号

---

书 名: C 语言程序设计

作 者: 葛日波

责任编辑: 满志文

出版发行: 北京邮电大学出版社

社 址: 北京市海淀区西土城路 10 号(邮编:100876)

发 行 部: 电话: 010-62282185 传真: 010-62283578

E-mail: publish@bupt. edu. cn

经 销: 各地新华书店

印 刷: 北京源海印刷有限责任公司

开 本: 787 mm×1 092 mm 1/16

印 张: 20.25

字 数: 475 千字

版 次: 2008 年 8 月第 1 版 2008 年 8 月第 1 次印刷

---

ISBN 978-7-5635-1620-9

定 价: 29.00 元

• 如有印装质量问题,请与北京邮电大学出版社发行部联系 •

# 前 言

C语言自问世以来,一直得到广泛应用。在当前,C语言是科技人员应该掌握的一种基本计算机语言工具。

本书针对应用型人才培养要求,结合作者多年的教学实践和使用英文原版教材授课的经验,以面向应用、重视实践、便于自学为立足点,在认真分析初学者特点的基础上,进行了精心策划,准确定位,以清晰的概念、大量的图例、丰富的工程用例、深入浅出地讲解,全面阐述了C语言(ANSI C)的基本内容和程序设计技术。

本教材要实现两个目标:一是在知识结构上使初学者一看就懂,具有一定基础的读者看了就有提高;二是帮助读者掌握编程的方法与技巧,提高读、编写程序的能力,培养良好的编程习惯。

本教材特点:

1. 面向应用、突出能力培养。
2. 内容精练、语言简练、通俗易懂。
3. 结构合理、循序渐进、容易理解。
4. 大量的图表和程序实例,使读者学习更容易。
5. 大量的章后习题有助于学习者练习和提高。
6. 把国外优秀的程序设计题例纳入教材,内容丰富。
7. 融入了程序设计思想和规范,有利于良好编程习惯的培养。

本书由葛日波主编,第2章和附录E由付蓉编写,第3章和附录D由刘丽艳编写,第5章和附录B由何毅编写,第10章由张治海编写,其余章节由葛日波编写,最后由葛日波统稿。

十分感谢大连理工大学城市学院院长唐志宏教授及电子与计算机工程学院院长孙承科教授对编写本书的帮助和指导。

朱志刚和张治海对教材中的所有程序进行了调试,许多同事对编写此书给予了关心和支持,在此一并感谢。

限于作者的水平,书中难免有错误和不足之处,殷切期望广大读者批评指正。

编 者

# 目 录

## 第 1 章 C 语言初步

|                     |    |
|---------------------|----|
| 1.1 C 语言的背景 .....   | 1  |
| 1.2 C 程序结构 .....    | 2  |
| 1.2.1 程序结构 .....    | 2  |
| 1.2.2 两个程序例子 .....  | 4  |
| 1.3 标识符 .....       | 6  |
| 1.4 数据类型 .....      | 7  |
| 1.5 变量 .....        | 9  |
| 1.6 常量 .....        | 12 |
| 1.6.1 常量的种类 .....   | 12 |
| 1.6.2 常量的用法 .....   | 17 |
| 1.7 输入/输出 .....     | 18 |
| 1.7.1 格式化输出 .....   | 18 |
| 1.7.2 格式化输入 .....   | 22 |
| 1.7.3 字符输入/输出 ..... | 25 |
| 习题 .....            | 27 |

## 第 2 章 表达式和简单程序设计

|                        |    |
|------------------------|----|
| 2.1 表达式 .....          | 32 |
| 2.1.1 初级表达式 .....      | 33 |
| 2.1.2 二元表达式 .....      | 34 |
| 2.1.3 赋值表达式 .....      | 36 |
| 2.1.4 后缀表达式 .....      | 38 |
| 2.1.5 一元表达式 .....      | 40 |
| 2.1.6 逗号表达式 .....      | 42 |
| 2.2 副作用 .....          | 42 |
| 2.3 表达式求值 .....        | 43 |
| 2.3.1 无副作用的表达式求值 ..... | 43 |
| 2.3.2 有副作用的表达式求值 ..... | 43 |
| 2.4 混合类型表达式 .....      | 45 |

|              |    |
|--------------|----|
| 2.4.1 隐式类型转换 | 45 |
| 2.4.2 强制类型转换 | 47 |
| 2.5 语句       | 48 |
| 2.6 标准库函数    | 49 |
| 2.7 简单程序设计   | 54 |
| 习题           | 60 |

### 第3章 分支程序设计

|                       |    |
|-----------------------|----|
| 3.1 关系与逻辑运算           | 64 |
| 3.1.1 逻辑数据            | 64 |
| 3.1.2 关系运算            | 64 |
| 3.1.3 逻辑运算            | 66 |
| 3.2 两路分支              | 70 |
| 3.2.1 if...else 语句    | 70 |
| 3.2.2 if 语句           | 72 |
| 3.2.3 if 语句的嵌套        | 73 |
| 3.2.4 条件表达式           | 74 |
| 3.3 多路分支              | 75 |
| 3.3.1 switch 语句       | 75 |
| 3.3.2 if...else-if 语句 | 79 |
| 习题                    | 82 |

### 第4章 循环程序设计

|                         |     |
|-------------------------|-----|
| 4.1 循环概述                | 89  |
| 4.2 while 语句            | 91  |
| 4.3 for 语句              | 94  |
| 4.4 do...while 语句       | 96  |
| 4.5 循环的嵌套               | 99  |
| 4.6 break 与 continue 语句 | 105 |
| 习题                      | 109 |

### 第5章 数组

|              |     |
|--------------|-----|
| 5.1 概念       | 116 |
| 5.2 一维数组     | 118 |
| 5.2.1 定义数组   | 118 |
| 5.2.2 访问数组元素 | 119 |
| 5.2.3 用数组存数据 | 120 |
| 5.3 顺序查找     | 124 |

|                    |     |
|--------------------|-----|
| 5.4 冒泡排序 .....     | 126 |
| 5.5 二维数组 .....     | 128 |
| 5.5.1 定义数组 .....   | 129 |
| 5.5.2 访问数组元素 ..... | 129 |
| 5.5.3 用数组存数据 ..... | 129 |
| 习题 .....           | 132 |

## 第6章 指针

|                     |     |
|---------------------|-----|
| 6.1 概念 .....        | 137 |
| 6.1.1 指针常量 .....    | 137 |
| 6.1.2 取地址运算 .....   | 137 |
| 6.1.3 指针变量 .....    | 138 |
| 6.2 指针与变量 .....     | 139 |
| 6.2.1 定义指针变量 .....  | 139 |
| 6.2.2 指针的初始化 .....  | 139 |
| 6.2.3 用指针处理变量 ..... | 141 |
| 6.2.4 程序举例 .....    | 142 |
| 6.3 多级指针 .....      | 146 |
| 6.4 指针与数组 .....     | 150 |
| 6.4.1 指针与一维数组 ..... | 150 |
| 6.4.2 指针与二维数组 ..... | 156 |
| 6.5 动态内存分配 .....    | 161 |
| 6.5.1 概念 .....      | 161 |
| 6.5.2 内存管理函数 .....  | 162 |
| 习题 .....            | 165 |

## 第7章 字符串

|                   |     |
|-------------------|-----|
| 7.1 字符串的存储 .....  | 172 |
| 7.2 指针与字符串 .....  | 173 |
| 7.3 字符串输入输出 ..... | 174 |
| 7.4 字符串处理函数 ..... | 180 |
| 习题 .....          | 186 |

## 第8章 函数

|                      |     |
|----------------------|-----|
| 8.1 结构化程序设计 .....    | 190 |
| 8.1.1 自上而下程序设计 ..... | 190 |
| 8.1.2 C 程序的结构化 ..... | 191 |
| 8.2 函数定义 .....       | 192 |

|                     |     |
|---------------------|-----|
| 8.2.1 函数定义格式 .....  | 192 |
| 8.2.2 函数定义举例 .....  | 194 |
| 8.3 原型声明与函数调用 ..... | 194 |
| 8.3.1 原型声明 .....    | 194 |
| 8.3.2 函数调用 .....    | 195 |
| 8.4 向函数传值 .....     | 197 |
| 8.5 向函数传地址 .....    | 198 |
| 8.6 向函数传数组 .....    | 204 |
| 8.6.1 传一维数组 .....   | 204 |
| 8.6.2 传二维数组 .....   | 207 |
| 8.7 向函数传字符串 .....   | 209 |
| 8.8 指针型的函数 .....    | 210 |
| 8.9 函数指针 .....      | 212 |
| 8.10 作用域 .....      | 213 |
| 8.11 变量的存储类型 .....  | 215 |
| 8.12 命令行参数 .....    | 218 |
| 8.13 递归 .....       | 220 |
| 习题 .....            | 226 |

## 第9章 结构、联合与枚举

|                      |     |
|----------------------|-----|
| 9.1 类型定义 .....       | 234 |
| 9.2 结构类型 .....       | 235 |
| 9.2.1 结构的声明 .....    | 235 |
| 9.2.2 定义结构变量 .....   | 236 |
| 9.2.3 结构指针 .....     | 238 |
| 9.3 结构的处理 .....      | 238 |
| 9.3.1 结构变量的初始化 ..... | 238 |
| 9.3.2 访问结构成员 .....   | 239 |
| 9.4 复杂结构 .....       | 244 |
| 9.4.1 嵌套结构 .....     | 244 |
| 9.4.2 含数组的结构 .....   | 246 |
| 9.4.3 含指针的结构 .....   | 249 |
| 9.5 结构数组 .....       | 251 |
| 9.6 结构与函数 .....      | 252 |
| 9.7 联合 .....         | 264 |
| 9.8 枚举 .....         | 268 |
| 习题 .....             | 269 |

## 第 10 章 文件

|                        |     |
|------------------------|-----|
| 10.1 文件概述.....         | 276 |
| 10.1.1 文件的概念 .....     | 276 |
| 10.1.2 文件的分类 .....     | 276 |
| 10.1.3 文件和流 .....      | 277 |
| 10.1.4 文件指针 .....      | 277 |
| 10.2 文件的打开与关闭.....     | 278 |
| 10.2.1 文件的打开 .....     | 278 |
| 10.2.2 文件的关闭 .....     | 279 |
| 10.3 文件操作.....         | 280 |
| 10.3.1 文本文件读写函数 .....  | 280 |
| 10.3.2 二进制文件操作函数 ..... | 284 |
| 10.4 其他函数.....         | 287 |
| 习题.....                | 289 |
| 附录 A ASCII 码表 .....    | 293 |
| 附录 B C 语言中的运算符 .....   | 294 |
| 附录 C C 语言库函数 .....     | 295 |
| 附录 D 位运算.....          | 303 |
| 附录 E 预处理命令 .....       | 305 |
| 参考文献.....              | 312 |

# 第1章 C语言初步



C语言是一种功能很强的高级计算机语言。本章主要介绍C语言的发展背景、C程序的基本组成要素与C程序的结构、数据类型、常量、变量、数据的输入/输出等内容。

## 1.1 C语言的背景

C语言是目前国际上广为流行的一种通用的结构化程序设计语言(Programming Language),它不仅是开发系统软件(System Softwares)的程序设计语言,也是开发应用软件(Application Softwares)的理想工具。

C语言的发展根源可以追溯到ALGOL。1967年Martin Richards推出了BCPL(Basic Combined Programming Language)。1970年,美国贝尔实验室的Ken Thompson以BCPL语言为基础设计出了B语言,并用B语言编写了第一个UNIX操作系统(Operating System)。由于B语言过于简单,所以其功能有限。1972年,Dennis Ritchie在B语言的基础上开发出了C语言。C语言既保持了BCPL和B语言的优点,又克服了它们的缺点。C语言的发展过程如图1-1所示。

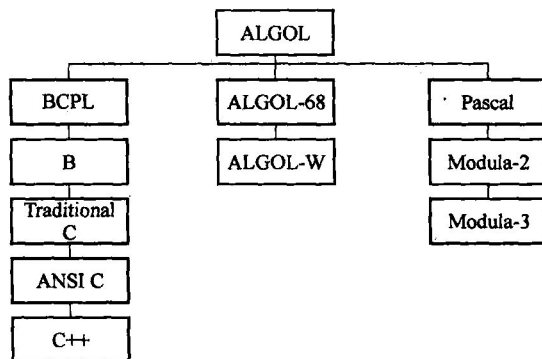


图 1-1 C语言的发展过程

1978年,Brian W. Kernighan和Dennis Ritchie合著了《The C Programming Language》一书,被称为传统C语言。1983年美国国家标准化协会(American National Stan-

tard Institute)制定了 C 语言标准,称为 ANSI C,并在 1989 年通过认定。1990 年,国际标准化组织(ISO,International Stantard Organization)接受了 ANSI C 标准。本书使用 ANSI C 标准。

## 1.2 C 程序结构

### 1.2.1 程序结构

每个 C 程序都由一个全局声明区(Global Declaration Section)和一个或多个函数(Functions)组成,其中必须有且只能有一个称作 main 的函数,图 1-2 是典型的 C 程序结构示意图,其中图 1-2(a)是只含有一个 main 函数的简单结构,图 1-2(b)是包含了两个函数的程序结构,一个是 main 函数,另一个是称作 fun 的函数。

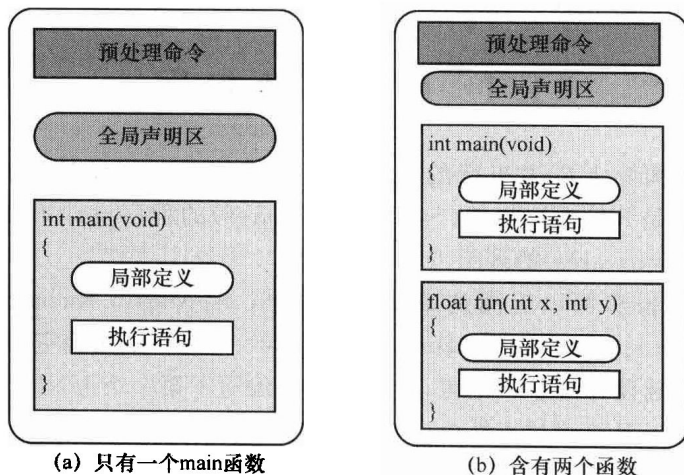


图 1-2 典型 C 程序的结构

说明以下几点:

(1) 关于全局区。全局声明区是位于整个程序开头的区域,主要用来进行全局对象的声明,它对整个程序都是可见的(Visible),也就是说在全局区声明的对象,程序中的所有代码都可以访问它。

(2) 关于函数。

① 函数是具有独立功能的命名的代码段(Code Segment)。

② 函数是 C 程序的基本结构单位(Unit)。

③ 函数包括函数头(Function Header)和函数体(Function Body)两部分。

函数头(Function Header)包含了函数值类型、函数名和参数列表。函数体跟在函数头后,是用 { } 括起来的部分,包含了局部定义区(Local Definition Section)和可执行语句(Statement)两部分,局部定义在函数体的开始部分,主要用于定义函数内部要用到的变量(Variable)。语句部分在局部定义的后面,它是由若干条语句组成的。图 1-3 显示了函

数的结构情况。

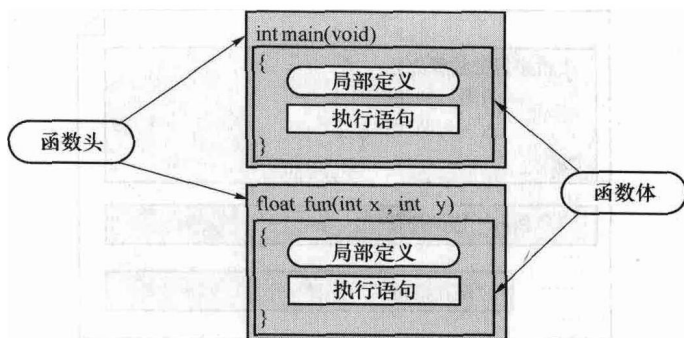


图 1-3 函数的结构

④ main 函数必须要有,它是整个程序的控制部分。

每个程序中,有且必须有一个称作 main 的函数。程序的执行是由 main 函数的开始而开始,由它的结束而结束。其他函数都是在开始执行 main 函数以后,通过函数调用 (Function Call) 而得以执行的。

⑤ 函数包括库函数 (Library Function) 和用户定义函数 (User-Defined Function)。

库函数是由开发商提供的函数,可以直接使用;用户定义函数是用户根据需要自己编制的函数。

⑥ 函数间只存在调用 (Calling) 和被调用 (Called) 的关系。

main 函数只能由操作系统调用,它可以调用任何函数,其他函数之间可以相互调用。

(3) 关于语句 (Statements)。

① 语句是 C 程序的最小功能单位。

一条语句经编译 (Compile) 转化为可执行的机器指令 (Instruction)。

② C 语言中的语句以分号 (Semicolon) 结束。

③ C 语言中的语句包括表达式语句、空语句、复合语句和控制语句,详细内容参见 2.5 节。

④ C 语言中允许一行写多条语句,也允许一条语句占多行,本书特别提倡一行一句的书写规范。

(4) 关于注释 (Comments)。

① 注释是在程序代码中加入的说明性信息,为了增强程序的可读性。格式是:

/\* 注释的内容 \*/

② 注释以 /\* 开始,由 \*/ 结束。

③ 可以在除一条可执行语句内部之外的任何地方加注释。

④ 注释可以跨多行。图 1-4 给出了几个注释的范例。

⑤ 注释部分在编译时被忽略。

根据这一特点,在调试程序时,若不想让某部分代码参加编译和运行,就可以把这部分代码的首尾用 /\* 和 \*/ 括起来,该部分代码就暂时被屏蔽了。

(5) 关于预编译命令 (Precompiler Directives)

① 预编译命令是开发商提供的命令格式。

之所以称作预编译命令,是因为这些命令在编译前实施相应处理。预编译命令包括

文件包含、宏定义和条件编译三种,详细内容请参阅附录 E。

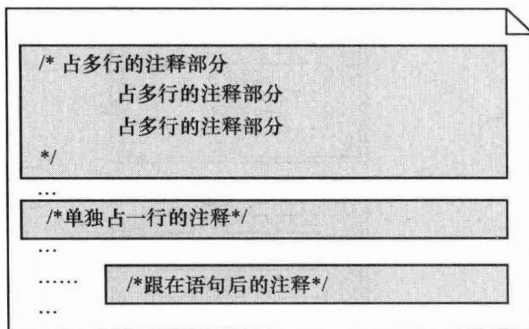


图 1-4 注释范例

② 所有的预编译命令都是以#开头,末尾没有分号。

如:

```
#include <stdio.h>           /* 文件包含命令 */
#define PI 3.14159           /* 宏定义命令 */
```

③ 预编译命令一般都放在程序最开始的位置。

 运行效率、可读性和可维护性是衡量一个程序好坏的重要指标。一个好的程序就是力争做到有很高的运行效率,方便别人阅读、日后修改和升级。良好的书写格式、适当地添加注释都是增强程序可读性的有利手段。

### 1.2.2 两个程序例子

下面通过两个具体的程序来研究 C 程序的结构问题。

图 1-5 给出了一个非常简单的程序。程序中只有一个 main 函数,只包含了一条预编译命令,没有全局声明部分,且函数体中没有局部定义,只包含两条语句。功能是在屏幕上输出“Hello World!”。

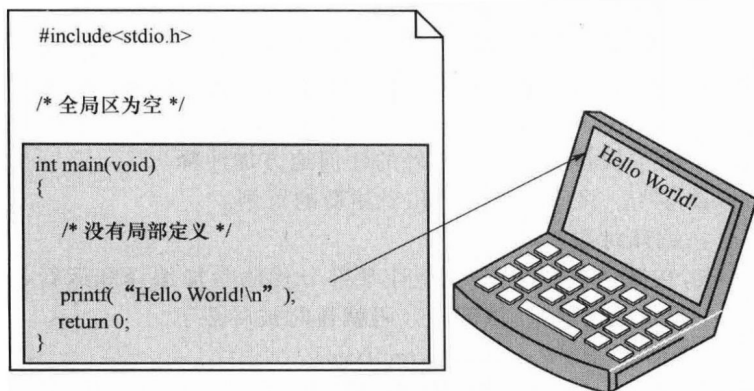


图 1-5 一个简单的 C 程序

程序开头是一条预包含命令：

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

它的作用是把尖括号中名字为 `stdio.h` 的头文件包含到程序中，程序编译时系统将用 `stdio.h` 的内容替换这条语句。在 `stdio.h` 中包含了库函数 `printf()` 的相关信息。

在预包含命令后面是 `main` 函数的函数头：

```
int main( void )
```

其中，`int` 是函数的返回值类型，表示整数；`main` 是函数名；圆括号 `()` 是函数的标识，括号中的 `void` 表示该函数是无参数的。注意：函数头末尾不能有分号。

在函数头后面用 `{ }` 括起来的部分，是 `main` 函数的函数体，里面有两条语句，一条用来输出信息，一条用来结束程序。第一条语句 `printf(...)`；是调用库函数 `printf` 来输出信息，由函数名 `printf` 和括在一对圆括号 `()` 中的参数组成。本例中的参数只有一个，是用双引号括起来的部分，是要输出的内容。“`\n`”是换行符，控制在输出“Hello World!”后换行。`printf()` 后的分号是语句结束符，C 语言中的每一个语句都以“`;`”终止。第二条语句 `return 0;` 用来结束(Terminate)`main` 函数执行，把控制还给操作系统(Return the control to OS)。

图 1-6 给出了一个较复杂的 C 程序。程序包含两个函数——`main` 和 `average`，`main` 函数调用 `average` 函数，因此 `main` 函数称作调用函数(Calling Function)，`average` 函数称作被调用函数(Called Function)。

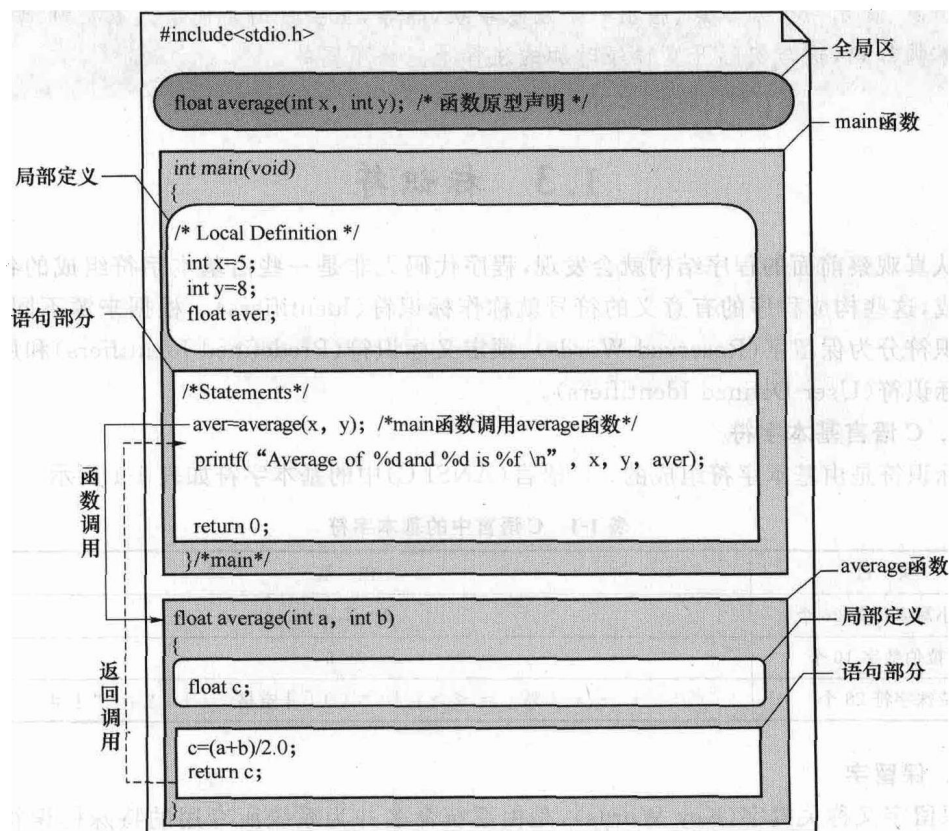


图 1-6 一个较复杂的 C 程序

average 函数的作用是接收两个整数,返回一个浮点型的平均值。在函数头中,float 用来指定返回值的类型,a 和 b 称作形式参数(Formal Parameters),用来接收调用函数传递来的数据,int 用来指定接收数据的类型。函数体中共有三条语句,第一条语句是局部变量定义语句,后面两条是执行语句。第一条语句的作用是申请了一个称作 c 的用来存储 float 型数据的变量(Variable)。第二条语句的作用是求 a 和 b 的平均值,并把结果存储到 c 中。第三条语句的作用是把 c 的值传递给调用函数。

由于 main 函数要调用库函数 printf 输出数据,因此在程序开头同样有一条预包含命令。因为 main 函数要调用用户自定义函数 average,因此在全局区写了一条函数原型声明语句,用来对函数的头部信息进行描述,具体细节参阅 8.3 节。

main 函数的函数体包含了六条语句,前面的三条是局部变量定义语句,分别用来定义整型的变量 x、y 和浮点型变量 aver,并给 x 和 y 分别存入了 5 和 8。第四条语句是函数调用语句,把 x 和 y 的值传给函数 average,并把处理结果存储到变量 aver。在函数调用中,x 和 y 称作实参(Actual Parameters)。第五条语句也是函数调用语句,是调用库函数 printf 把 x、y 和 aver 中的数据输出到屏幕上。

#### 特别说明一点:

教材中的很多例子,作者在认为需要的地方都适当加了注释,用它来告诉大家一个信息、提示一个知识点、指出一个注意事项,等等,主要目的是便于大家更好地理解和掌握知识,这与实际开发程序时加的注释是完全不同的。

## 1.3 标识符

认真观察前面的程序结构就会发现,程序代码无非是一些由基本字符组成的各类符号组成,这些构成程序的有意义的符号就称作标识符(Identifiers)。根据来源不同,可以把标识符分为保留字(Reserved Words)、预定义标识符(Predefined Identifiers)和用户自定义标识符(User-Defined Identifiers)。

### 1. C 语言基本字符

标识符是由基本字符组成的。C 语言(ANSI C)中的基本字符如表 1-1 所示。

表 1-1 C 语言中的基本字符

| 项 目         | 内 容                                                    |
|-------------|--------------------------------------------------------|
| 大小写字母各 26 个 | A~Z, a~z                                               |
| 阿拉伯数字 10 个  | 0~9                                                    |
| 特殊字符 28 个   | + - * / % _ = < > &   ^ ~ ( ) [ ] 空格 . { } ; : ' " ! # |

### 2. 保留字

保留字又称关键字(Key Words),是由系统命名并为系统所专用的特殊标识符。如程序中主函数的名字 main 就是一个保留字,这些保留字用户只能按系统的规定来用,不

可以作为它用。C语言中共有这样的保留字 32 个。

```

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

```

记住：所有的保留字不允许做他用，且必须要用小写字母。

### 3. 用户自定义标识符

用户自定义标识符是用户按照一定规则命名的标识符，主要用来表示常量、变量、函数和类型等的名字。标识符的构成规则如下：

- (1) 只能由字母、数字和下画线组成。
- (2) 必须以字母或下画线开头。
- (3) 长度不超过 31 个字符。
- (4) 不能和系统保留字同名。

表 1-2 中给出了一些用户自定义标识符的例子。

表 1-2 用户自定义标识符

| 合法标识符        |         | 不合法标识符       |          |
|--------------|---------|--------------|----------|
|              |         | 标识符          | 错误原因     |
| a            | al      | \$ sum       | \$ 开头    |
| student_name | stntNm  | 2name        | 2 开头     |
| _aSystemName | _aSysNm | Student name | 有空格      |
| TRUE         | FALSE   | int          | 与系统关键字同名 |

强调以下三点：

- (1) 要尽量使用有意义的英文单词，这样的程序更容易读。
- (2) 当用多个单词做标识符时，单词之间最好隔开。

有两种方法：一是单词间用下画线；二是除第一个单词外首字母大写。本书提倡使用后者，这是当前最流行的书写格式。如：

```

student_name           /* 用下画线分隔 */
studentName            /* 首字母用大写 */

```

- (3) 标识符不要太长，长了书写麻烦，还容易出错。

通常采用去掉元音字母部分的方法进行简化，有时也采取取前几个字符的办法，如可以把 studentName 简化为 stdntNm，也可以把 studentName 简化为 stuName。

## 1.4 数据类型

数据是有类型的，在编程时必须明确指定参加处理数据所属的类型。数据类型 (Data Type) 定义了一个值的集合和基于该集合的一组操作。值的集合称作该类型的取值范围 (Domain for the Type)。不同类型数据的取值范围不同，可以施加的运算也不同。例如，

整型(int)数据,其取值范围是 $-32\,768 \sim 32\,767$ 之间的整数,可对其施加算术运算。

C语言中的数据类型分为标准数据类型(Standard Types)和复合数据类型(Derived Types)两大类。标准数据类型是原子型(Automic),不能再分解。复合数据类型是由标准数据类型按一定规则构造出的数据类型,其中包括指针(Pointer)、枚举类型(Enumerated Type)、共用体(Union)、数组(Array)和结构体(Structure)。本章主要介绍标准数据类型,其他数据类型将在以后的章节中逐步介绍。

C语言包含四种标准数据类型:void、char(character的缩写)、int(integer的缩写)和float(floating point的缩写),如图1-7所示。

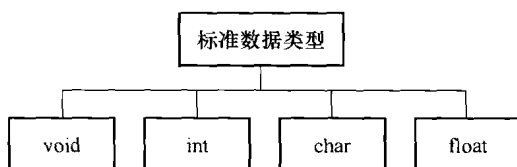


图 1-7 标准数据类型

### 1. 空类型

空类型没有值只有赋值操作。看起来似乎有点不好理解,不过在后面的学习中大家就会明白它是非常有用的,因为空类型可以被看作是“智能型”,可以用它来代表任何的数据类型,在第6章将做详细介绍。

### 2. 整型

整型数是没有小数部分的数据。C语言支持三种不同的整数类型:短整型(short int)、整型(int)和长整型(long int)。短整型(short int)也可以表示为short,长整型(long int)也可以表示为long。每一种类型又存在有符号(signed)和无符号(unsigned)的区分,有符号整数可以表示正数和负数,而无符号整数只能表示正数。有符号数使用关键字signed表示,无符号数使用关键字unsigned表示。

不同类型的数据在内存中占用空间的大小因机器类型而有所不同,占用空间的大小决定了该数据类型的取值范围。表1-3给出了在微型计算机(Microcomputer)上三种类型数据的大小情况。

表 1-3 三种整型数的表示及大小

| 类 型     | 表示方法(关键字)                                                                                    | 字节数 | 取值范围                |                     |
|---------|----------------------------------------------------------------------------------------------|-----|---------------------|---------------------|
|         |                                                                                              |     | 最小值                 | 最大值                 |
| 有符号短整型数 | signed short int                                                                             | 2   | $-32\,768$          | $32\,767$           |
| 无符号短整型数 | unsigned short int                                                                           | 2   | 0                   | $65\,535$           |
| 有符号整型数  | signed int                                                                                   | 2   | $-32\,768$          | $32\,767$           |
| 无符号整型数  | unsigned int                                                                                 | 2   | 0                   | $65\,535$           |
| 有符号长整型数 | signed long int                                                                              | 4   | $-2\,147\,483\,648$ | $-2\,147\,483\,647$ |
| 无符号长整型数 | unsigned long int                                                                            | 4   | 0                   | $4\,294\,967\,295$  |
| 大小关系    | $\text{sizeof}(\text{short}) \leq \text{sizeof}(\text{int}) \leq \text{sizeof}(\text{long})$ |     |                     |                     |