C语言程序设计基础教程

吕凤翥　高　超　编著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书较全面系统地讲述了C语言的基本词法和语法以及它们在编程中的应用，并且介绍了一些编写C语言程序的方法及技巧。本书内容包含有C语言的词法、常量和变量、运算符和表达式、语句、函数和存储类、指针、结构联合和枚举、预处理命令以及文件操作等内容。

本书是作者在总结多年C语言教学经验的基础上编写的，概念准确、例题丰富、重点突出、解释详尽，每章配有练习题。

本书可作为大专院校学生学习C语言的教材，也可作为自学C语言的读者的教材。

图书在版编目(CIP)数据

C语言程序设计基础教程/吕凤翥，高超编著 . - 北京：电子工业出版社，2000.6

ISBN 7-5053-5870-7

Ⅰ.C... Ⅱ.①吕 ...②高 ... Ⅲ.C语言-程序设计-高等教育：技术教育-教材 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2000)第 05866 号

书　　名：C语言程序设计基础教程

编 著 者：吕凤翥　高　超

策　　划：卢先河

责任编辑：吕　迈

特约编辑：朱若愚

排版制作：电子工业出版社计算机排版室监制

印 刷 者：北京天宇星印刷厂

装 订 者：河北省涿州桃园装订厂

出版发行：电子工业出版社　URL:http://www.phei.com.cn

北京市海淀区万寿路 173 信箱　邮编 100036

经　　销：各地新华书店

开　　本：787×1092　1/16　印张：20　字数：512 千字

版　　次：2000 年 6 月第 1 版　2000 年 6 月第 1 次印刷

书　　号：ISBN 7-5053-5870-7/G·530

印　　数：8000 册　　定价：25.00 元

第1章　C语言概述

本书主要介绍C语言的词法和语法知识，及用C语言进行编程的方法。第1章先让读者对C语言和C语言程序有个初步的了解，为后面的学习打下一个基础。因此，介绍了C语言的由来和发展，C语言的特点和应用，C语言程序结构的特点和书写格式，常用的C语言输入输出函数以及C语言程序的实现方法。通过对上述问题的讲述，使读者对C语言和C语言程序有一个概貌的了解。

1.1　C语言的由来和发展

1.1.1　C语言的由来

C语言诞生于1972年，它是由美国电话电报公司(AT&T)贝尔实验室的D.M.Ritchie设计的，并首先使用在一台UNIX操作系统的DEC PDP-11计算机上实现的。从C语言诞生至今还不到30年。

C语言的来源要追溯到ALGOL 60，C语言是由ALGOL 60逐渐演变而来的。早在1963年英国剑桥大学根据当时流行的高级语言ALGOL 60推出一种接近于硬件的语言CPL(Combined Programming Language)。1967年，英国剑桥大学针对当时的CPL语言提出一种改进的语言，称为BCPL(Basic Combined Programming Language)语言。1970年，美国的贝尔实验室的K.Thompson以BCPL语言为基础，设计出一种既简单又接近于硬件的高级语言，称为B语言，它取用了BCPL语言的第一个字母，并使用B语言写成了第一个UNIX操作系统，在DEC PDP-7计算机上得到了实现。1972年，美国贝尔实验室的D.M.Ritchie在B语言基础上，克服了B语言依赖于机器又无数据类型等局限性，开发出了C语言。由此可见，C语言是由ALGOL 60逐步演变来的，它的过程如下所示：

ALGOL 60(1960年)→CPL(1963年)→BCPL(1967年)→B(1970年)→C(1972年)

1.1.2　C语言的发展

C语言是在人们设想寻找一种既具有一般高级语言特征，又具有低级语言特点的语言的情况下应运而生的，它具备了一般高级语言和低级语言的优点，是人们当时所期望的一种语言。

1973年，C语言刚刚诞生的第二年，美国贝尔实验室的K.Thompson和D.M.Ritchie两人合作使用C语言对UNIX操作系统进行了修改，修改后的UNIX操作系统是第5版本。原来的UNIX操作系统是用汇编语言和B语言编写的，修改后的UNIX操作系统中的90%以上是用C语言编写的。因此，C语言刚诞生就将它的命运与UNIX操作系统的命运紧密地联系在一起了。随着UNIX操作系统的发展和推广，C语言也得到了广泛的应用和发展。

C语言出世后，在实际的应用中也在不断地改进，越来越符合人们的需要。在1975年，UNIX操作系统第6版本公布以后，C语言开始引起人们的注意，它的优点也逐渐地被人们所

认识。1977 年出现了一种与具体机器无关的 C 语言编译文本,推动了 UNIX 操作系统在各种机器上迅速地实现。随着 UNIX 操作系统的日益广泛地应用,C 语言也将得到了迅速地推广。1978 年以后,C 语言先后被移植到大、中、小和微型机上,它很快成为世界上应用最广泛的计算机语言之一。

1978 年又推出了 UNIX 操作系统的第 8 版本,以该版本中的 C 语言编译系统为基础,B.W.Kernighan 和 D.R.Ritchie 合作(被称为 K&R)出版了《The C Programming Language》(C 程序设计语言)一书,被称为标准 C 语言。

1983 年,ANSI(美国国家标准化协会)对 C 语言的各种版本进行了扩充,推出了新的标准,被称为 ANSI C,它比原来的标准 C 有了一些改进和扩充。

1987 年,ANSI 又公布了 87 ANSI C 新版本。目前流行的各种 C 语言编译系统的版本大多数都是以此为基础的,但是它们各自又有其不同之处。当前在微机上使用的 C 语言编译系统多为 Microsoft C,Turbo C,Borland C 和 Quick C 等,它们都是按照标准 C 编写的,它们之间略有差异。每种编译系统又都有不同的版本,其差异主要是在其功能上有些不同,版本高的编译系统所提供的函数更多;编译能力更强;使用起来更方便;界面更友好。

1.2 C 语言的特点和应用

1.2.1 C 语言的特点

C 语言是一种开发比较晚的高级语言,它吸取了早期高级语言的长处,克服了某些不足,形成它独有的风格和特点。总的说来,C 语言是一种简单明了、功能强、移植性好的结构化程序设计语言。

1.C 语言是一种结构化程序设计语言

结构化程序设计语言,又称为模块化程序设计语言,这种语言可以采用结构化(或模块化)程序设计方法进行程序设计,这是当前比较通用的一种程序设计方法。在 C 语言中,函数是构成结构化程序的最小模块,每个函数实现一个功能,多个函数完成一个较大的功能。实际上,C 语言的程序就是由若干个函数所组成的,这些函数可以放在一个文件中,也可以放在多个文件中。

C 语言是结构化程序设计语言,它具备了构成结构化程序设计的三种基本结构模式的语句。这三种基本结构模式如下所示:

(1) 顺序结构模式

它由若干条顺序执行的语句构成,执行这种模块的特点是按事先编写好的语句顺序逐条执行。这是程序设计的最基本形式。

(2) 分支结构模式

用来处理实际问题中的分支选择问题。C 语言中提供了两种分支语句:条件语句和开关语句,使用它们将能构成各种多路分支选择的结构模式。

(3) 循环结构模式

用来实现重复执行某种功能的要求。C 语言中提供了三种循环语句(while 循环,do-while 循环和 for 循环),使用它们可以构成各种循环结构模式,完成各种重复执行某种功能的要

求。

在讲述C语言是一种结构化程序设计语言的同时，还应该指出C语言为了某些灵活性，而使得结构化出现了一些不够完全之处。因此，有人说，C语言是一种不完全的结构化程序设计语言。

C语言的不完全性主要表现在如下两方面：

第一，完全结构化程序设计语言不允许使用goto语句，因为goto语句会破坏结构化。但是，goto语句在某些时候会使得程序更加简练，因此，在C语言中为了某些方便和灵活，允许使用goto语句，但它是被限制性地使用的，以减少它对结构化的影响。

第二，完全的结构化程序设计语言要求一个模块只有一个入口和一个出口，这样便于结构化的管理。但是，C语言程序中允许一个函数使用多个返回语句(return语句)，即允许函数通过多个出口返回到调用函数，这样做为编程增加了灵活性，但是这样不利于结构化。

2.C语言编程简洁明了

C语言是一种非常简洁的语言，用C语言编写的程序简洁明了。C语言的简洁性表现如下：

(1) C语言中关键字较少，只有30多个。有些关键字采用了简单的符号代替。例如，条件语句中的if体的定界符采用一对花括号({})来标识，如果是一条语句规定不用定界符。又例如，循环语句的循环体也用一对花括号来定界。

(2) C语言中类型说明符采用缩写形式，为书写带来一些方便。例如，整型变量说明符用int，而不用integer；字符型变量说明符用char，而不用character等等。

(3) C语言中运算符不仅数量多，而且功能强，这不仅为编程带来方便，而且功能较强的运算符使得编程更加简洁。例如，三目运算符(?:)具有简单条件语句的功能。三目运算符要求三个操作数，其格式如下：

d1? d2:d3

其中，d1，d2和d3是三个表达式作为该运算符的三个操作数。该运算符的功能是先计算表达式d1的值，如果是非0，则该表达式值为d2表达式的值，否则该表达式的值为d3表达式的值。

(4) C语言还具有预处理功能，这一功能将使程序编写简洁明了。C语言的预处理功能中有宏定义和文件包含两个命令。这两个命令给简化程序的书写带来方便。

3.C语言功能强

C语言的功能强表现在它既具有高级语言的功能又具有低级语言的功能。C语言具有数值运算的功能，还具有非数值运算的功能，并且在处理非数值数据时更加方便和灵活。另外，C语言还具备一些低级语言(即汇编语言)的功能。例如，寄存器运算功能、二进制位运算功能和内存地址运算功能等。这些低级语言的功能是一般高级语言所不具备的。而C语言具备了上述这些功能，就使得C语言的应用更加广泛。它既可以像一般高级语言那样编写一些应用程序，还可以像汇编语言那样编写一些系统程序。

C语言还提供了丰富的数据类型，除了基本数据类型外，还提供了数组、结构、联合和枚举等构造数据类型，使用这些数据类型可以较方便地实现各种复杂的数据结构的操作，进一步增强了该语言的功能。

4.C 语言的移植性好

由于 C 语言的编译系统较小,又加上它具有一些预处理的命令,这些都对 C 语言的移植带来一些方便。C 语言的移植性好表现在只要对它稍加修改,便可适用于各种型号的机器和各类操作系统,同时也表现在使用 C 语言编写的应用程序可以很方便地用在不同系统中。正是由于 C 语言的这一特点,使得它能广泛地被应用到各个领域。

C 语言中,由于突出某个特点或者为了使用更加灵活,而使得该语言也存在一些不足。了解这些不足,对避免出错是十分重要的。

(1) 运算符多,难用难记。C 语言中共有 40 多个运算符,又分为 15 种优先级和两类结合性,这无疑对数据运算及处理带来了方便。但是,这么多的运算符和那么多的优先级却带来了难记忆和难使用的不足。例如,有的是相同的运算符,却有着不同的功能,它们是 -,* 和 &,这三个运算符都有单目与双目之分。以运算符 - 为例,单目是求负数,双目是减法运算符;而运算符 *,单目是取指针的内容,双目是乘法运算符。另外,还要将所有的运算符分为 15 种优先级,有时容易搞混。

(2) C 语言中类型转换比较灵活。例如,char 型可以自动转换为 int 型,这种自动转换会给某种计算带来一定的方便,它将允许一个字符与一个整数进行加减运算。例如

```
'm' + 1
```

是合法的。该表达式的值用字符型表示为'n',用整型数表示为 156。但是,C 语言程序为了转换上的方便,在许多情况下不做类型检查。有些情况下 C 语言要求类型一致,由于不作类型检查,当出现不一致时也不出错,结果造成运算结果的混乱。因此,在编程中对类型处理要慎重,尽量避免出现不一致的差错。

(3) C 语言中对数组进行动态赋值时不做越界检查,因此,当给某个数组动态赋值的数据项数超过了数组元素个数时,容易造成数据方面的混乱。例如,某数组有 5 个元素,当给该数组赋第 6 个数据项时,便将该数据赋给该数组最末一个元素后面的地址中。而数组的赋初值(静态赋值)不会发生越界现象,因为静态赋值时系统进行越界检查,在发现越界后即报错。

(4) C 语言为了优化等原因允许不同的编译系统对表达式或参数表中的操作数或数据项的计算顺序有不同的安排,即有的编译系统规定计算顺序是从左至右,有的编译系统规定计算顺序是从右至左。这种不同的安排对一般表达式或参数表来讲没有什么影响,但是,当表达式或参数表中出现有副作用运算符时,对计算顺序不同的编译系统将会出现二义性。这一点在后面还会详述。

以上 4 点不足将在本书后面章节中做详细讨论,读者对此应引起重视,避免在编程中出现上述问题。

1.2.2 C 语言的应用

从前面对 C 语言特点的分析中可以看到,C 语言具有编程方便、语句简练、功能性强、移植性好等优点。因此,C 语言是编程者喜欢使用的一种结构化程序设计语言,它被广泛地应用于系统软件和应用软件的开发研制之中。下面介绍 C 语言应用的几个方面。

1. 数据库管理系统及应用程序方面

C 语言的非数据处理功能很强,并且又具有汇编语言的一些特点,因此,它被广泛地应用

于开发数据库管理系统和应用软件。大多数关系数据库管理系统，如 dBASE、FoxBASE、ORACLE 等，都是用 C 语言开发的。而数据库管理系统的应用软件大都是用 C 语言开发的，因此 C 语言在开发数据库管理系统和应用软件方面深受开发者的欢迎。

2. 图形图像系统和应用程序方面

C 语言在图形图像的开发中也有着广泛的应用。很多图形图像系统软件包是采用 C 语言开发的，例如，AutoCAD 通用图形系统软件包就是用 C 语言开发的，并直接支持 C 语言程序。C 语言编译系统本身带有许多绘图功能的函数，利用这些函数开发图形应用软件也十分方便。也可以直接使用 C 语言编写绘图环境实现专业绘图设计。

3. 编写与设备的接口程序方面

C 语言在建立友好的交互式图形界面方面有着广泛应用，用 C 语言可以方便地实现下拉式菜单、弹出式菜单和多窗口技术等功能，并且在编写与设备接口程序方面也有着广泛的应用。因为 C 语言既有着高级语言的特性又具有汇编语言的部分功能，因此在编写接口程序与汇编语言混合的程序时会带来很多方便，并显示出更高的效率。

4. 数据结构方面

C语言不仅提供了十分丰富的基本数据类型，又提供了构造数据类型。使用C语言提供的数据类型可以很方便地解决复杂的数据结构中的问题；使用C语言将十分方便地编写关于链表结构、队列和栈结构以及树结构的程序，并且有许多这方面的成熟例程供选择使用。

5. 排序和检索方面

排序和检索是数据处理方面经常遇到的较为复杂的问题。使用C语言来编写排序和检索程序既方便又简洁。特别是有些排序算法采用了递归方法进行编程，更显得清晰明了。

以上列举了五个方面的应用，而 C 语言的应用还远远不止这些方面。可以说，C 语言在各个领域都可以使用，并且都会有较好的效果。所以，C 语言是当前被广泛使用的高级语言之一。

1.3 C语言程序结构特点和书写格式

本节从 C 语言的实际程序入手，初步看到 C 语言程序结构上的特点以及书写格式方面的要求，为逐步地学习 C 语言的词法、语法及基本编程方法打下一个基础。

1.3.1 C语言程序实例

下面通过两个 C 语言程序实例来认识和理解 C 语言程序的特点。

【例 1.1】 编写一个程序，输出如下的字符串：

This is a C-Language Program.

程序内容如下：

```
main()
{
  printf("This is a C-Language Program. \n");
}
```

这是一个输出一个指定的字符串的简单程序。该程序只有一个文件,文件中只有一个函数,该函数是主函数 main()。主函数没有参数也没有返回值。该函数的函数体是用一对花括号({})括起来的。函数体内有一条语句。每一条语句的最后要有一个分号结束。该语句是一个函数调用,被调用函数是一个标准格式输出函数 printf()。该函数是编译系统提供的,自动地包含在每个 C 语言的程序中,只需要编程者调用就可以了。该函数(printf()函数)中只有一个用双撇号括起的控制串部分,该函数功能是将其控制串中的一般字符输出显示在屏幕上。字符串中有一个换行符'\n',它是一个不可打印字符,用转义序列法表示的。

执行该程序后,输出如下结果:

This is a C-Language Program.

光标停在下一行开头。

上述分析中有些内容后面还会详述,这里只要有一个大概的印象就可以了。

下面再看一个稍微复杂一点的程序。

【例 1.2】 编写一个程序,求出两个 int 型数之和。

假定两个 int 型是从键盘输入获取的,并将求两个 int 型数之和写在另外一个函数 add()中。

程序内容如下:

```
main()
{
  int a,b,sum;
  printf("Enter a and b: ");
  scanf("%d%d",&a,&b);
  sum = add(a,b);
  printf("sum = %d + %d = %d \n",a,b,sum);
}
add(x,y)
int x,y;
{
  return(x + y);
}
```

该程序也是一个文件,文件中包含两个函数,它们是主函数 main()和另外一个被主函数调用的函数 add()。主函数 main()没有参数和返回值。函数 add()有两个参数 x 和 y,它们都是 int 型变量,该函数有 int 型数的返回值,返回值是由 return 语句来实现的。该函数的功能是用来计算两个数的和。主函数的函数体内有 5 条语句,开始是一条说明语句,用来定义 3 个 int 型变量:a,b 和 sum。接着,一条 printf()语句输出一条提示信息:

Enter a and b:

提醒用户要从键盘上输入两个 int 型数,通过调用 scanf()函数,将键入的两个 int 型数存放在变量 a 和 b 中,使变量 a 和 b 分别获取了值。scanf()函数是编译系统提供的被自动包含在程

序中的一个标准格式输入函数。该函数控制串中只有两个格式符,参数表中有两个用地址值表示的参数 &a 和 &b,这是该函数要求的。这里 & 是一个取地址的运算符,作用于变量名前,表示取该变量的地址。关于该函数的功能和对其参数的要求以及运算符 & 的作用将在本书后面章节中详述。

在主函数中,调用了 add()函数,并将该函数的返回值赋值给变量 sum,于是 sum 中将存放着所求得的两个数的和。在主函数中使用 printf()输出显示 sum 的值,这就是 a 和 b 变量的和。

执行该程序后,屏幕显示出如下提示信息:

Enter a and b:

这时,从键盘上输入两个 int 型数,例如,18 和 36 时,

18 ⌴ 36↙

则在屏幕上显示如下结果:

sum = 18 + 36 = 54

该格式将在 printf()函数中讲述。

1.3.2 C语言程序结构特点

通过前面讲述的两个 C 语言程序实例(例 1.1 和例 1.2),可以对 C 语言程序的特点归纳如下:

(1) C 语言程序是由一个或多个文件组成的,每一个文件又是由一个或多个函数组成。可见,C 语言程序是一个函数串,即由若干个函数所组成。每个函数又是由若干条语句组成,每条语句又是由若干个单词组成,每个单词是由若干个字符组成。字符是组成 C 语句程序的最小元素,函数是组成 C 语言程序的最小模块。上述关系可以简化表示如下:

程序——文件——函数——语句——单词——字符

在例 1.2 中,该程序是由 2 个函数组成,主函数 main()中函数体内有 5 条语句,被调用函数 add()体内只有一条语句。

(2) C 语言程序是一个函数串,即包含若干个函数,在这些函数中,有且只能有一个是主函数 main()。系统执行该程序时,一定是从主函数开始执行。程序中主函数之外的所有函数都是被调用函数,或者是被主函数调用,或者是被主函数所调用的某个函数调用。因此,程序中函数之间是调用关系。每个函数在程序中的位置是无关的,主函数可以放在程序头,也可以放在程序尾,还可以放在程序中间,其他函数也是如此。函数与函数是平等的关系,不是包含的关系,C 语言中不允许在函数之中再定义函数。例如,下述程序段是错误的。

```
main()
{
  int a,b,sum;
     ⋮
  add(x,y)
  int x,y;
  {
    return(x + y);
  }
     ⋮
```

}

(3) 一个程序里的诸多函数中,只能有一个主函数,并且规定其名字为 main()。根据需要它可以有参数,也可以没有参数,本书前几章讲述的程序中主函数都没有参数。如果一个程序由多个文件组成时,多个文件中只能有一个文件含有主函数,该文件称为主文件,其余文件中不得再含有主函数。

(4) 程序由函数组成,函数由函数头和函数体组成。函数头包含函数的类型、函数的名字、函数的参数以及参数的说明。函数体由一对花括号括起来的若干条语句组成,在这些语句中说明语句放在执行语句的前面。这便是函数的定义格式。先对函数有个初步了解,关于详细内容将会在函数一章中详述。

1.3.3 C语言程序书写格式

C 语言的语句比较精练、短小,一条语句的含意又比较丰富,语句格式的规定又比较多,因此,C 语言程序的可读性较差,需要养成良好的书写习惯,以便提高程序的可读性,否则不仅读起来困难,而且有时会引起误导,造成分析上的错误。

C 语言程序的书写格式习惯上有如下要求:

(1) C 语言程序中,每行可以写一条语句,也可以写多条语句。一般地讲,一行只写一条语句,有时很短的语句可写多条。C 语言程序中,也允许将一条语句写成多行,一般情况下,可不必加续行符。但是,一个字符串尽量写在一行,而不要断成多行,太长的字符串可以分成多个字符串来写。

(2) C 语言程序中,每条语句的末尾必须加一个分号(;),而不是一条语句的末尾不要分号。分号是用来表示一条语句的结束。C 语言程序中,语句前不需要加行号,只有在 goto 语句转向到的某条语句前需加语句标号,具体用法详见"语句"一章。

(3) C 语言程序中,经常出现花括号({}),花括号都是成对出现的,不同情况下使用花括号是用来表示不同的含意。例如,函数体用一对花括号表示该函数体的定界符,即从左花括号起到右花括号止,其中的若干语句为该函数的函数体。花括号的其他用法本书后面会逐一讲述。花括号的书写格式有三种,本书中采用其中一种,规定如下:每个花括号独占一行,并且左花号和右花括号都与使用他们的语句对齐,而花括号中的语句都向右缩进两个字符。

(4) C 语言程序中,语句的书写要注意适当的缩进。使用缩进的方法使程序更加清晰易读。一般地,循环语句中的循环体,条件语句中的 if 体或 else 体内的语句都应缩进。

(5) 在书写程序中,要习惯使用/*…*/对程序中某些部分进行注释,所注释的内容只供阅读,而不参与编译和运行。使用注释的方法可增加对程序的可读性。

为了说明书写格式的重要性,下面举出一个例子,该例给出两个描述相同问题的程序,前一个程序书写格式不好,但是也能编译运行,读起来很费力,后一个程序格式写得较好,读起来比较清晰省力。它们执行结果是相同的。

【例 1.3】 编写程序求两个 int 型数之和。

程序一:不按习惯的格式书写。

```
add(x.
y)
int
x,y;{
```

```
return
(x + y);}main(){
int a,
b;
a = 10
;b = 5;
printf("%d \ n",
add(a,b))
;}
```

程序二：按习惯的格式书写，每行一条语句，单词间用空格符分隔，说明语句中各个变量之间和函数参数之间用逗号分隔，并适当采取缩进方法。

```
add(x,y)
int x,y;
{
 return(x + y);
}
main()
{
 int a,b;
 a = 10;
 b = 5;
 printf("%d \ n",add(a,b));
}
```

这两个程序都可以通过编译连接生成可执行文件，执行该程序后，输出结果如下：

```
15
```

C 语言程序的书写比较自由，只要不把一个单词拆开就可以随便分行。书写得太随便了会影响可读性。

1.4 C 语言中常用的读写函数

在前面的例程中，可看到 C 语言程序离不开的函数 printf()，该函数用来将其计算结果输出显示到屏幕上。与这个函数相关的另一个函数是 scanf()函数，在前面的例 1.2 中也出现过，它的功能是用来从键盘上输入数据。这两个函数分别是标准格式输出函数 printf()和标准格式输入函数 scanf()函数，它们都是标准文件的读写函数。这里，标准文件指的是输入标准文件——键盘和输出显示标准文件——屏幕。这些标准文件的读写函数(即输入输出函数)是最常使用的。这节中讲述的就是标准文件的一些读写函数，由于它们在程序中经常出现，放在前面讲述的目的是便于读者阅读使用这些函数的程序，关于一般文件的读写函数将被放在本书中最后一章中讲述。

1.4.1 常用的输入函数

输入函数又称读函数，用来从标准输入设备键盘上读取数据的函数，故称标准文件输入函数。常用的有如下三种：

1. 获取一个字符的函数 getchar()

该函数的功能是从键盘上一次获取一个字符，并可将它赋值给一个相应的变量。使用该函数获取字符不是直接从键盘上获取，而是从键盘的输入缓冲区中读取。将键盘上输入的字符先存放到一个输入缓冲区中，然后使用该函数从缓冲区中读取字符。该函数格式如下：

```
int getchar()
```

其中，getchar 是函数名，该函数没有参数，其返回值是一个 int 型数，即返回该函数所读取字符的 ASCII 码值。

在程序中使用该函数时，往往包含 stdio.h 文件，因为 getchar()函数被定义在该头文件中。

2. 获取一个字符串的函数 gets()

该函数的功能是从键盘上获取所键入的字符串，它的返回值是 char 型指针。该函数格式如下：

```
char * gets(s)
```

其中，gets 是函数名，函数名前面的 char * 表示该函数的类型是字符型指针。该函数有一个参数 s，它是字符数组名或字符指针名，用来存放从键盘上读取的字符串。从键盘上输入的字符串以换行符作为结束符。

3. 标准格式输入函数 scanf()

这个函数在前面讲过的例 1.2 中已经出现过。它的功能是从键盘上按照所指定的格式读取数据，给指定的变量赋值。该函数的格式如下：

```
int scanf("〈控制串〉",〈参数表〉)
```

其中，scanf 是函数名，该函数的参数分为两部分，中间用逗号分开，前一部分是用双撇号括起来的〈控制串〉，后部分是〈参数表〉。

〈参数表〉中参数个数可以是一个，也可以是多个，多个参数之间用逗号分隔，每个参数是一个地址值。一般变量的地址值可用变量名前加上运算符 & 来表示。例如，变量 a 的地址值被表示为 &a。关于运算符 & 的用法后面再详述。要求〈参数表〉中参数的个数与〈控制串〉中的格式符的数目相等，并且要求〈参数表〉中各参数的类型与〈控制串〉中对应的格式符类型相同。

〈控制串〉中包含有格式符和一般字符。格式符是由格式标识符(%)和格式说明符组成的，用来指定某种输入格式的。对该函数的格式说明符常用的有如下几种：

d ——十进制整型
o ——八进制整型
x ——十六进制整型
u ——无符号型十进制整数
f ——小数型单精度浮点数
e ——指数型单精度浮点数
c ——单个字符

s ——字符串

在%和格式说明符之间还可用修饰符来说明输入项的数据宽度。常用的修饰符有如下几种：

l ——用在 d,o,x 前表示长整型数，用在 f 和 e 前表示双精度浮点数。

数字——用在%与格式说明符之间的数字表示接收输入数据的最大宽度。

〈控制串〉中的一般字符用作“匹配符”。所谓“匹配符”是指控制串中出现的一般字符与输入流中键入的相同字符进行匹配，从而确定输入流中的数据项，即将输入流中的匹配符作为数据项的分隔符。如果〈控制串〉中没有一般字符作为匹配符时，则输入流中使用缺省的数据项分隔符，即空白符（包含空格符、换行符和水平制表符等）。输入流是指从键盘上输入的若干个字符信息。一个输入流一般是由若干个输入项组成的，scanf（）函数中每个参数将顺序从输入流中获取一个输入项的数据。

该函数的返回值是一个 int 型数，它表示该函数的参数表中成功获得输入数据的参数的个数。因为该函数是从键盘上获取数据的，由于种种原因，有可能使得参数表中多个参数中部分获取输入数据，而有些参数没有获取输入数据，该返回值便可判定参数表中有多少参数获取数据。

以上讲述了三个常用的标准文件的读函数，即输入函数。具体实例后面讲述。

1.4.2 常用的输出函数

输出函数又称写函数，它们的功能是将计算的结果输出到计算机的显示屏幕上。常用的输出函数有如下三种：

1. 输出一个字符的函数 putchar()

该函数的功能是将一个指定的字符输出显示在屏幕上。输出显示的字符可以是字符常量，也可以是字符变量。该函数格式如下：

```
int putchar(c)
```

其中，putchar 是函数名，该函数有一个参数 c，c 可以是一个 char 型变量，也可以是字符常量或字符型表达式。该函数将 c 中字符输出显示在屏幕上。该函数有一个 int 型返回值，它是输出显示字符的 ASCII 码值。该函数一般被包含在 stdio.h 文件中。

2. 输出一个字符串函数 puts()

该函数的功能是将一个指定的字符串输出显示到屏幕上。该函数的格式如下：

```
int puts(s)
```

其中，puts 是函数名，该函数有一个参数 s，它可以是字符数组名，也可以是字符指针名。有关字符数组和字符指针的概念后面将会讲解。该函数正常时返回值为 0。程序中经常使用该函数将一个字符串显示在屏幕上。

3. 标准格式输出函数 printf()

这个函数在前面讲过的三个例程中都出现过。该函数的功能是将该函数参数表中的若干个表达式的值，按其指定的格式和顺序逐一显示在屏幕上。该函数的格式如下：

```
int printf("〈控制串〉",〈参数表〉)
```

其中，printf 是函数名，该函数参数也被分为两部分，一部分是用双撇号括起来的〈控制串〉，另一部分是〈参数表〉。该函数参数从形式上看与 scanf()函数相同，但是，就其内容上却有很大差别，请不要把它们弄混了。

〈参数表〉是由一个或多个表达式组成的，多个表达式之间用逗号分隔，也可以没有表达式，这时，用来分隔〈控制串〉和〈参数表〉之间的逗号也被省略，该函数的参数只剩下一个〈控制串〉了，而控制串中也只有一般字符，因此，〈控制串〉变成了一个字符串了。要求〈参数表〉中表达式的个数和类型与〈控制串〉中的格式符的个数和类型相一致，即要求两者的个数相等，对应的类型相同。

〈控制串〉中包含有格式符和一般字符。格式符用来指定输出格式，它由格式标识符%和格式说明符组成。该函数的格式说明符有如下几种：

d——十进制整型数

o——八进制整型数

x——十六进制整型数

u——无符号型十进制整数

c——单个字符

s——字符串

f——小数型单精度浮点数

e——指数型单精度浮点数

g——f 和 e 中较短的一种

格式说明符是用一个字母表示的。在格式标识符%与格式说明符之间还可用修饰符来说明其输出数据项的具体宽度。常用的修饰符有如下几种：

〈数字〉·〈数字〉—小数点前的〈数字〉用来表示输出数据项的最小宽度。所谓最小宽度是指当输出数据的实际宽度小于最小宽度时，按最小宽度输出，一般用空格符填补空位；当输出数据的实际宽度大于最小宽度时，则按实际宽度输出，可见最小宽度是用来指出所输出的数据项的最小宽度。小数点后面的数字是用来表示输出数据的精度，对浮点数来讲表示小数点后的位数；对字符串来讲表示输出字符串中字符最多的个数，并将超过部分截掉。

l——用在格式符 d,o 和 x 前表示长整型数，用在 e,f,g 前面表示双精度浮点数。

-——负号用来表示输出数据在域宽中左对齐。当不用负号时，则表示右对齐。

0——用来表示在右对齐中替换输出数据项左边的空格符。

〈控制串〉中出现的一般字符照样输出，即将一般字符输出显示在屏幕上。在一般字符中，对可打印字符直接用字符符号来表示，对不可打印字符则用转义序列表示。转义序列是表示字符的一种方法，这种方法是使用字符的 ASCII 码值来表示字符，其格式有如下两种：

\0ddd　　或　　\xhh

其中，\0ddd 格式是在 \0 后面用字符的八进制的 ASCII 码值表示，最多为 3 位。\xhh 格式是在 \x 后面用字符的十六进制的 ASCII 码值表示，最多用 2 位。例如，字符 Esc 的转义序列表示为 \033 或 \x1b。

为了方便起见，C 语言将一些常用的不可打印字符的转义序列用 \〈字母〉来表示，如表 1.1 所示。

表 1.1 常用的不可打印字符的转义序列表示法

转义序列	含 义	转义序列	含 义
\n	换行符	\r	回车不换行
\t	水平制表符	\v	垂直制表行
\b	退格符	\f	走纸换页符
\a	鸣铃符	\\	反斜线符
\'	单撇号	\0	空字符

1.4.3 常用的读写函数应用实例

前面讲过了三对常用的读写函数,它们都是从键盘上读取数据,或向屏幕上输出显示数据的函数。这里仅举几个例子讲述一下它们的使用。

【例 1.4】 分析下列程序的输出结果,熟悉对一个字符的读写函数的用法。

```
#include <stdio.h>
main()
{
   int a,b;
   printf("Enter a character: ");
   a = getchar();
   b = getchar();
   putchar(a);
   printf("Enter a charater,again: ");
   a = getchar();
   b = getchar();
   printf("%c,%c \n",b,a);
}
```

执行该程序,屏幕上显示如下信息:

Enter a character:a ↵

在提示信息后面输入字符 a 和回车键后,输出显示如下信息:

a

Enter a character,again:b ↵

又键入字符 b 和回车键后,输出如下结果:

,b

程序分析:

(1) 该程序由一个函数 main()组成。程序开头使用预处理命令 include 包含了一个头文件 stdio.h,因为该文件包含了该程序中要调用的函数 getchar()和 putchar()。

(2) 该程序中调用了标准文件的读写函数有如下几个:

对一个字符的读函数 getchar(),调用 4 次。

对一个字符的写函数 putchar(),调用 1 次。

标准格式输出函数 printf(),调用 3 次。

(3) 该程序中调用 getchar()函数是从键盘上输入的字符流中读取一个字符,并将它赋给一个变量。开始两次调用 getchar()函数是将输入的字符'a'和回车符分别读取后赋给变量 i 和 j,后面两次调用 getchar()函数是将输入的字符'b'和回车符分别读取后赋给变量 i 和 j。

(4) 该程序中三次调用 printf()函数。第一次和第二次调用时,参数中没有参数表,只有控制串,控制串中只有一般字符,没有格式符。前两次使用 printf()函数是用来在屏幕上显示一个字符串作为提示信息。第三次调用 printf()是用来输出显示 2 个字符。该函数参数中的参数表部分有两个变量,其间用逗号分隔,它们分别是 j 和 i。该函数参数中的控制部分有两个格式符,分别是%c 和%c,表示输出单个字符,还有两个一般字符,它们是逗号(,)和换行符(\n)。该函数按其格式规定先按%c 的输出格式输出显示变量 j 的值,即为空行,接着,将一般字符逗号输出显示,最后是按%c 的输出格式输出显示变量 i 的值,即显示字符 b。因此,最后出现上述输出结果。

(5) 该程序中又出现了 putchar()函数。该函数是用来将指定的字符输出显示在屏幕上。在该程序中使用 putchar()函数用来将变量 i 中的字符显示在屏幕上,由于当时 i 变量中存放的字符是 a,于是调用该函数将在屏幕上显示出字符 a。

(6) 前面讲过的 getchar()函数实际上是从输入缓冲区中读数字符的。在本程序运行后,有下面提示信息:

Enter a character:

之后,输入了字符 a 和回车符,实际上在缓冲区中存放了 2 个字符,当使用了 getcher()函数读出一个字符并赋给变量 i 后,缓冲区内还存在一个回车字符。如果不将这个字符读取出来,则仍保留在缓冲区内,这对后来的输入会带来影响。如果接着又输入字符 b 和回车符,这时缓冲区内将存放了三个字符:上次剩下的回车符,b 字符和回车符。再读取时先读出上一次的回车符,再读出 b 字符,再读出回车符。因此,第一次输入后,为了不影响其后的输入,则需要将缓冲区清除干净。该程序中的第二次调用 getchar()函数是将缓冲区内剩余的回车符读取出来并赋给变量 j,以使得缓冲区内不再保留剩余字符,保证后面输入和读取的正确性,这一点请读者在使用 getchar()函数时注意。

【例 1.5】 分析下列程序的输出结果,并熟悉对一个字符串的读写函数。

```
#include <stdio.h>
main()
{
  char s1[50],s2[50];
  puts("Enter a string: \n");
  gets(s1);
  puts(s1);
  puts("Enter a string,again: \n");
  gets(s2);
  puts(s2);
}
```

执行该程序,显示如下信息:

Enter a string:

string ↵

划下划线的是键盘输入的内容。输入后，输出信息如下：

string

又显示如下信息：

Enter a string again:

enter ↵

再键入字符串 enter，按回车键后，输出如下信息：

enter

程序分析：

(1) 该程序中，调用 puts()函数共 4 次，调用 gets()函数共 2 次。这两个函数定义在 stdio.h 文件，因此程序开头包含了该文件。

(2) 该程序中出现的 puts()函数是用来将指定的字符串输出显示在屏幕上。该函数的输出显示的字符串可以用字符数组名来指定，也可以直接用字符串。该程序中 4 次调用 puts()函数，其中，有 2 次是用字符串常量的，另外 2 次用字符数组名 s1 和 s2。调用 puts()函数将一个字符串输出显示在屏幕上。

(3) 该程序中出现了 2 次 gets()函数，该函数是用从键盘的输入字符流中获取一个字符串，并存放指定的字符数组中。该程序中第一次调用 gets()函数是将输入的字符串"string"存放在字符数组 s1 中，第二次调用 gets()函数是将输入的字符串"enter"存放在字符数组 s2 中。

【例 1.6】 分析下列程序的输出结果，并熟悉标准格式读写函数的用法。

```
main()
{
  int x;
  float y;
  double z;
  printf("Enter int x,float y,double z: ");
  scanf("%d%f%lf",&x,&y,&z);
  printf("x = %d,y = %.2f,z = %8.3lf \n",x,y,z);
}
```

执行该程序输入如下数据：

Enter int x,float y,double z:56 ␣ 78.123 ␣ 14.5 ↵

输入数据方法是先输入 56 空格符，再输入 78.123 空格符，最后输入 14.5 回车符。

输出结果如下：

x = 56,y = 78.12,z = ␣␣ 14.500

程序分析：

(1) 该程序中调用了 2 次 printf()函数和 1 次 scanf()函数。

(2) 该程序中调用 scanf()函数的格式如下：

scanf("%d%f%lf",&x,&y,&z);

该函数的参数的控制部分含有 3 个格式符，按顺序分别是十进制整型数格式符%d，单精度浮点型数格式符%f 和双精度浮点型数格式符%lf。该函数的参数的参数表中包含有 3 个参数，它们之间用逗号分隔，每个参数是一个地址值，它们分别是变量 x 的地址 &x，变量 y 的地址 &y 和变量 z 的地址 &z。