第3章 输入输出与简单程序设计

C语言没有专门的输入输出语句，程序中所有的输入输出都是通过诸如scanf和printf等标准库函数来完成的。本章先讲述程序设计中的流程控制结构，然后介绍C语言的输入输出函数，最后讲述简单程序设计的方法与步骤。

本章重点：

(1) 字符输入和字符输出函数的使用。

(2) 格式输入和格式输出函数的使用。

本章难点：

不同类型的数据的格式输入输出函数的使用。

3.1 概　　述

在程序设计过程中经常会遇到以下3种情况。

(1) 依次执行某些操作，最后得到所需的结果。

(2) 在执行操作的过程中对各种数据进行判断，然后根据判断的结果选择不同的数据进行处理。

(3) 反复执行某一项或某几项操作，直到达到某个目的为止。

为了满足对上述数据处理的需求，保证数据处理控制流程的规范性和控制编程中的复杂性，1965年，计算机科学家E.W.Dijikstra提出了“结构化程序设计”的思想。结构化程序设计思想采用“自顶向下、逐步求精”的程序设计方法，可以有效地将一个比较复杂的系统设计任务分解成许多易于控制和处理的子任务，从而使程序的设计、开发和维护更加方便。其中，控制语句的结构化是结构化程序设计方法的主要精髓之一。结构化程序设计是计算机软件发展史上的第三个里程碑(另外两个是子程序和高级语言)。

所谓控制语句的结构化是指将顺序结构、选择结构和循环结构3种基本结构作为程序流程的基本控制结构，每种结构均只有一个入口和一个出口，任何程序都可由这3种基本控制结构通过组合、叠加，像搭积木一样来构成。控制语句的结构化改进了程序设计的效率，提高了程序设计的质量，使得程序设计向着更加易阅读、易理解、易维护和易验证的方向迈进。

3.2 流程控制结构与语句

1. 3 种基本控制结构

结构化程序设计使用的 3 种基本控制结构是顺序结构、选择结构和循环结构。

顺序结构是结构化程序设计的 3 种基本结构中最简单的一种。其特点是按语句在源程序中出现的先后顺序依次执行。它可以独立存在，也可以出现在选择结构或循环结构中。顺序结构是描述客观世界顺序现象的重要手段。

选择结构用来描述分支现象。它是一种常用的基本结构，通常只要稍有规模的程序都会不可避免地用到它。选择结构是根据条件判断的结果有选择地执行或不执行某些语句。

循环结构由循环条件和循环体(某一程序段)组成，它是在循环条件的控制下重复地执行循环中的若干语句。循环结构是描述客观世界重复现象的重要手段。

所有的流程控制都是由语句实现的，能够支持结构化程序设计的语言必须有好的流程控制语句。C 语言为实现结构良好的程序提供了基本的、功能强大的、使用灵活的各种流程控制语句，即语句组、条件判断(if…else)、多分支选择(switch)、循环(while，for，do)、转移语句(break，continue)等。

2. C 语句概述

语句是构成程序的基本单元。C 语言程序中有两类语句：说明语句和可执行语句。

说明语句用于定义程序中被处理的数据，包括变量说明、函数说明、常量定义及类型定义等。

可执行语句是程序中用于实现算法的代码，即完成对数据的处理和对程序流程的控制。C 语言的可执行语句共有以下 6 种。

(1) 表达式语句；

(2) 复合语句；

(3) 选择语句(if…else、switch)；

(4) 循环语句(while、for、do)；

(5) 转移语句(break、continue、return、goto)；

(6) 标号语句。

本章仅介绍表达式语句，其他几种语句将在以后的章节中介绍。

3. 表达式语句

C 语言是一种表达式语言。表达式是程序中使用最频繁的计算手段。程序中要求计算机进行某种计算主要是通过表达式来实现的。不同的表达式进行不同的运算，达到不同的目的。

C 语言中任何表达式在其末尾加上一个分号，就构成一个表达式语句。其语句形式为：

```
表达式;
```

其中“;”是C语句的结束标志,它是语句的组成部分,是不可缺少的。

表达式语句与表达式的区别是:表达式语句能够独立出现在程序中的任何地方,而表达式只能出现在允许表达式出现的位置,例如作为运算符的操作数、作为函数调用的参数、作为选择语句和循环语句的条件等。

表达式语句中使用较多的是赋值表达式语句、逗号表达式语句、函数调用表达式语句等,特别是赋值表达式语句(简称赋值语句)使用非常广泛。

【例 3.1】

```
x=y+1                    //赋值表达式
x=y+1;                   //赋值表达式语句
x+=y                     //复合赋值表达式
x+=y;                    //复合赋值表达式语句
i=j=k=0                  //多重赋值表达式
i=j=k=0;                 //多重赋值表达式语句
a=5, n=2*a               //逗号表达式
a=5, n=2*a;              //逗号表达式语句,等同于语句"a=5 ; n=2*a ;"
a>b?1:0                  //条件表达式
a>b?1:0 ;                //条件表达式语句
printf("hello world")    //标准输出函数调用表达式
printf("hello world");   //标准输出函数调用表达式语句,习惯上称为输出语句
scanf("%d",&x)           //标准输入函数调用表达式
scanf("%d",&x);          //标准输入函数调用表达式语句,习惯上称为输入语句
```

表达式语句的表达式部分可以为空,即只有一个分号“;”。仅由一个分号组成的语句称为空语句,在语法上占据一个语句的位置,但是它不具备任何执行功能。空语句在程序中可以作为循环体使用,如下例所示。

```
for (i=0;i<100000;i++)
    ;
```

在这个循环中,循环体是一个空语句,不执行任何操作,在程序中用它来实现延时的功能。

C语言中运算符种类很多,它可与运算对象组成各种表达式,表达能力很强。表达式语句是C语言的一个重要特色,在设计C语言程序时使用十分方便。

3.3 基本的标准输入输出函数

C语言本身没有任何内置的输入输出语句。在C语言中所有的输入输出都是通过诸如printf和scanf之类的函数调用来完成的。这些函数统称为标准I/O库函数。

给程序变量提供数据有两种方法:一种方法是通过赋值语句把数值赋给变量,如“x=1;”“sum=0;”等;另一种方法是使用输入函数(如getchar和scanf),这两个函数可以从键盘读取数据。

输出数据可以使用输出函数(如putchar和printf函数),它们可以将结果送到外部设

备(如显示器)上。

用户程序在使用 C 语言的标准函数库提供的函数时,需用编译预处理命令“#include <…>”将相关的系统头文件包含进来,并按规定的格式调用其中的标准函数即可完成所需的功能。例如,使用标准输入输出库函数时,要用到 stdio.h 文件(文件扩展名 h 是 head 的缩写,称为头文件)。如:

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

或者

```
#include "stdio.h"
```

两者的区别在于:前者是从系统定义的有关目录下查找指定的头文件(一般放在目录\include 中),而后者先从当前目录中查找指定的头文件,如果不存在,则再从系统定义的有关目录下查找。

本章将介绍 4 个最基本的用于输入输出的标准库函数:字符输入函数 getchar、字符输出函数 putchar、格式输入函数 scanf 和格式输出函数 printf。它们都是在系统默认的输入输出终端设备(一般为键盘、显示器)上进行输入和输出。其他输入输出函数将在第 11 章中详细介绍。

3.4 单个字符的输入和输出

最简单的输入输出操作是从“标准输入设备”(通常是键盘)中读取一个字符,或向“标准输出设备”(通常是显示器)写一个字符。读取一个字符可以用 getchar 函数来完成(也可以用 scanf 函数来实现,详见 3.6 节)。

3.4.1 字符输入

字符输入函数 getchar 的调用形式为:

```
var_name =getchar();
```

其中,var_name 是用户已经声明为 char、short 等整数类型的变量名。

该函数的功能是:每调用一次该函数,计算机就等待从键盘输入一个字符。从键盘输入一个字符后,将该字符的 ASCII 码返回,并将它赋给变量 var_name。若遇到文件尾(Ctrl+Z),则返回 EOF(EOF 是系统在头文件 stdio.h 中定义的符号常量,其值为-1)。例如:

```
char ch;
ch=getchar();
```

当在键盘上输入 A 时,就把字符'A'赋给了变量 ch。

注意:

(1) getchar 函数的参数为空,但一对圆括号不可省略。

(2) getchar 函数一次只能接收一个字符,该字符可以赋给一个字符变量或整型变量,也可以不赋给任何变量,只作为表达式的一个运算对象参加表达式的运算或处理。

例如：

```
int num;
num =getchar();
```

执行时若在键盘上输入 a 时，则赋值后变量 num 的值为 97，即字符'a'的 ASCII 码值。

又例如：

```
printf("%c",getchar());
```

执行时若在键盘上输入 A，则 getchar 函数的返回值为 65，即字符'A'的 ASCII 码值，并将该返回值作为 printf 函数的参数以字符形式(%c)进行输出，即输出字符 A。

(3) getchar 函数只能输入可显示和可打印的字符，对于不可显示和不可打印的字符(如水平制表字符'\t'、响铃字符'\a'等)，只有使用赋值语句才能将其赋给相关变量。如：

```
ch='\t';
```

3.4.2 字符输出

与 getchar 函数相对应，C 语言中有一个 putchar 函数，用于每次向终端写一个字符。其调用形式如下：

```
putchar(var_name);
```

其中，var_name 是一个 char 类型的变量。该函数的功能是：每调用一次该函数，就在显示器上显示存储在 var_name 变量中的字符。如果没有发生错误，则函数返回输出的字符；否则函数返回 EOF。例如：

```
char  yn='y';
putchar(yn);
```

是将字符'y'显示在显示器的屏幕上。而语句：

```
putchar('\n');
```

则输出一个回车换行符，即将屏幕上的光标移到下一行的开始处。

【例 3.2】 从键盘输入一个字符并将该字符显示在显示器上。

【程序】

```
#include <stdio.h>
int main()
{
    char c;
    c=getchar();          //从键盘输入一个字符并将该字符的 ASCII 码赋值给变量 c
    putchar(c);           //将该字符输出在显示器上
    return 0;
}
```

【运行】 从键盘上输入一个字符'A'，并按回车键(即 Enter 键↙)，就会在屏幕上看到输

出的字符'A'。

```
A
A
```


上面的例 3.2 程序也可以修改为以下程序。

```
#include <stdio.h>
int main()
{
    char c;
    putchar(getchar()); //getchar 的返回值作为 putchar 的参数输出
    return 0;
}
```

【例 3.3】 在屏幕上显示 A,B,C 3 个字符。

【程序】

```
#include <stdio. h>
int main()
{
    char a,b,c;
    a='A';
    b='B';
    c='C';
    putchar (a);
    putchar (b);
    putchar(c);
    return 0;
}
```

【运行】

```
ABC
```

如果将例 3.3 程序中的 3 条输出语句修改为：

```
putchar (a);
putchar('\n');
putchar (b);
putchar('\n');
putchar (c);
putchar('\n');
```

则修改后程序的运行结果为：

```
A
B
C
```

注意：putchar 函数的参数可以是 char 类型，也可以是 short 或 int 类型的常量、变量或表达式。例如，有定义：

```
char ch1='a';
int ch2=97;
```

则

```
putchar(ch1);
putchar(ch2);
putchar(97);
```

都是将小写字母'a'显示在屏幕上。而

```
putchar(ch1-32);
putchar(ch2-32);
```

都是将大写字母'A'显示在屏幕上，即将小写字母转换为大写字母输出。

3.5 格式化输出

getchar 函数一次只能输入一个字符，putchar 函数一次只能输出一个字符。如果要一次输入或输出若干个任意类型的数据，必须通过 printf 和 scanf 函数来实现。scanf 和 printf 函数在数据输入和输出的过程中能够将计算机内部格式的数据和输出设备上的外部数据进行相互转换，故 scanf 和 printf 函数被称为格式输入函数和格式输出函数。本节介绍格式化输出函数 printf，格式化输入函数将在 3.6 节讲述。

编程人员都期望程序的输出清晰、易理解，也易使用。printf 函数所提供的特性，能使程序员用来有效地控制输出数据在显示器上的对齐方式和间距。

printf 函数的一般形式为：

```
printf(格式控制字符串 [,输出参数 1,输出参数 2,…,输出参数 n]);
```

该函数的功能是：在格式控制字符串的控制下，将输出参数进行转换与格式化，并在标准输出设备(显示器)上输出。它的返回值为正确输出的字符数。

例如：

```
printf("a=%4d,b=%6.1f, c=%2c\n",a, b, c);
```

第一个参数是格式控制字符串"a=%4d,b=%6.1f, c=%2c\n"，后面的 3 个参数是要输出的数据 a,b,c。

格式控制字符串是用一对双引号括起来的字符串序列(又称为转换控制字符)。输出参数 1 至输出参数 n 是要输出的数据项，每个输出参数是一个表达式，一般可以是任何基本类型的表达式，也可以是指针类型的表达式。

在格式控制字符串中可以包括如下 3 种字符信息。

(1) 普通字符：这些字符原样显示在屏幕上，它们通常用作对输出内容的注释或用来提示相关的信息。例如，“printf("a=%d,b=%d\n",a,b);”中的“a=”“,”和“b=”都是普

通字符。而“printf("Please input a number:\n");”则用来显示一个提示信息,即“Please input a number:”。在程序中这样的提示信息非常有用,经常用在输入语句 scanf 之前。

(2) 格式转换说明符:每个格式转换说明都由一个%开头,并以一个转换字符结束,转换字符通常用于说明输出数据的类型。转换说明并不直接输出,而是用于控制 printf 函数中参数的转换和打印。转换字符及其输出形式如表 3.1 所示。

格式转换说明的一般形式为:

```
%[[+/-][0][m.n][h/l]]转换字符
```

其中,+,-,0,m,n,h,l 是可选的,它们通常称为附加格式说明字符,用来说明输出数据的精度(即指出输出值的字段宽度、小数部分的位数)、输出值的左右对齐方向等,其输出形式如表 3.2 所示。

表 3.1 printf 函数的转换字符

转换字符	参数类型	输出形式
d	int	带符号的十进制数(正数不输出符号)
o	int	无符号八进制数(不输出前导符 0)
x,X	int	无符号十六进制数(不输出前导符 0x 或 0X),10～15 分别用 a、b、c、d、e、f 或 A、B、C、D、E、F 表示
u	int	无符号十进制数
c	int	单个字符
s	char *	字符串
f	double	十进制小数[-]m.dddddd,d 的个数由精度决定(默认值为 6)
e,E	double	标准指数,[-]m.dddddde±xxx 或[-]m.ddddddE±xxx, d 的个数由精度决定(默认值为 6)
g,G	double	选用%f 或%e 格式中输出宽度较短的一种格式,不输出无意义的 0

表 3.2 printf 函数的附加格式说明字符

附加字符	说明
字母 h	表示输出的是短整型整数,可加在 d、o、x、u 前面
字母 l	表示输出的是长整型整数,可加在 d、o、x、u 前面
m	表示输出数据的最小宽度,称为域宽
n	对实数,表示输出 n 位小数;对字符串,表示截取 n 个字符
0	表示左边补 0
+	转换后的数据右对齐
-	转换后的数据左对齐

下面是正确的 printf 语句的示例:

```
printf("%d",num);
printf("sum=%10d,average=%5.2f",sum,aver);
printf("length=%d",123);
```

(3) 转义字符:用来对输出进行控制。如'\n','\t'等。'\n'是回车换行控制符,用来使后面的输出内容从下一行开始;'\t'是水平制表控制符,使输出内容在下一个输出区输出(一个

输出区占 8 列)。

printf 不能自动回车换行,因此多个 printf 语句产生的输出将显示在同一行上。利用换行字符'\n'就可以产生换行。例如:

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

产生一个换行。

```
printf("a=%d\tb=%d\n",a,b);
```

输出 a 和 b 的值后,产生一个换行。其中的'\t'使得 a 和 b 的输出分隔开,输出更加清晰。

3.5.1 整数的输出

用于显示整数的格式转换说明符为:

```
%[m]d
```

其中,m 指定输出的最小宽度,可以省略。d 表示要显示的数为整数。如果一个数字的宽度比指定的宽度要大,则将全部显示,忽略该最小说明符。数字按给定宽度对齐显示,如果没有达到给定的宽度,则在数字前面补上空格。下面是在不同格式下输出整数 1234 的例子。

格式	输出
printf("%d",1234);	1 2 3 4
printf("%7d",1234);	␣ ␣ ␣ 1 2 3 4
printf("%3d",1234);	1 2 3 4
printf("%-7d",1234);	1 2 3 4 ␣ ␣ ␣
printf("%07d",1234);	0 0 0 1 2 3 4

输出格式中系统默认是右对齐方式,通过在%号后面放置一个减号,可以强制地使输出左对齐,如上面的第 2 个例子。也可以在域宽前面加一个 0,使输出结果的前面用 0 而不用空格来填充,如上面最后一个例子。

如果要输出长整型数,可以把格式说明符中的 d 用 ld 代替。同样,要输出短整型数,可以把格式说明符中的 d 用 hd 代替。

例如,语句:

```
long a=12345678;
short b=9876;
printf("%10ld,%10hd\n",a,b);
```

则产生输出:

```
␣␣12345678,␣␣␣␣␣␣9876
```

这个输出中␣符号表示由控制字符串中的域宽产生的空格。在这里,默认对齐方式都是右对齐的,所以需要在数字的左边添加空格。

【例 3.4】 计算 15 与 121 的和,用竖式加法显示结果。

【程序】

```
#include <stdio.h>
int main()
{
    int a=15,b=121;
    printf("%6d\n",a);
    printf(" +%4d\n",b);                    //在屏幕上输出加号及 b,+号前有一个空格
    printf("--------\n");                   //在屏幕上输出横线
    printf("%6d\n",a+b);
    return 0;
}
```

【运行】 该程序执行时输出为：

```
    15
 + 121
--------
   136
```

一般情况下只显示负数的符号，为了强制显示正和负的符号，必须使用加号(+)附加格式修饰符。例如，语句：

```
printf("%+10d\n",15);
```

产生的输出为：

```
␣␣␣␣␣␣␣␣␣+15
```

在输出整数时，%d 使整数用十进制形式进行显示，%o 使整数用八进制进行显示，%x 使整数用十六进制进行显示。例如，语句：

```
printf("%d,%o,%x\n",15,15,15);
```

产生的输出为：

```
15,17,f
```

3.5.2 实数的输出

用下面的格式转换说明符，可以将实数用小数形式输出：

```
%[m.n]f
```

其中，整数 m 表示包括小数点在内的总体显示宽度，整数 n 表示在小数点后输出数字的数目，m 和 n 均可省略。当省略 n 时，默认显示 6 位小数。显示时，在列宽为 m 的区域以右对齐方式显示。

对于所有的数字(整数、单精度浮点数和双精度浮点数)，如果总的域宽小于原数实际宽度，则忽略域宽按原数的实际宽度输出。如果小数部分含有的位数小于被指定的位数，则在这个数字末尾用 0 填补。如果小数部分的数位多于指定的位数，则这个数字被四舍五入到指定的小数位数。例如，语句：

```
printf("%10.3f",29.76);
```

产生输出：

```
␣␣␣␣␣29.760
```

在这个显示中，域宽 10 包含小数点和小数点右边的 3 位数。由于这个数字在右边只含有两个数字，这个数字的小数部分用 0 填补，并在输出数字的左边填写 4 个空格。

下面是在不同格式下输出数值 123.456 的例子。

格式	输出
printf("%f",123.456);	1 2 3 . 4 5 6 0 0 0
printf("%7.3f",123.456);	1 2 3 . 4 5 6
printf("%7.2f",123.456);	␣ 1 2 3 . 4 6
printf("%-7.2f",123.456);	1 2 3 . 4 6 ␣
printf("%5.3f",123.456);	1 2 3 . 4 5 6

如果要输出双精度实型数，可以把格式说明符中的 f 用 lf 代替。同样，要输出长双精度实型数，可以把格式说明符中的 f 用 Lf 代替。

3.5.3 单个字符的输出

使用如下格式，可以将单个字符显示在所需的位置。

```
%[m]c
```

上述格式将字符以右对齐的方式显示在列宽为 m 的区域中，m 可以省略。在整数 m 之前加上负号，则以左对齐的方式显示。m 的默认值为 1。

下面是在不同格式下输出字符'A'的例子。

格式	输出
printf("%c",'A');	A
printf("%4c",'A');	␣ ␣ ␣ A
printf("%-4c",'A');	A ␣ ␣ ␣

3.5.4 字符串的输出

输出字符串的格式说明符的形式为：

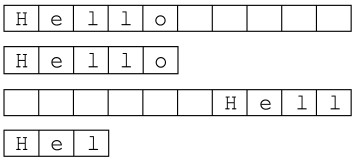
```
%[m.n]s
```

其中，m 指定显示的区域宽度，n 表示只显示字符串的前 n 个字符。这种显示为右对齐方式显示字符串，m 和 n 均可省略。

下面是在不同格式下输出字符串"Hello"的例子。

格式	输出
printf("%s","Hello");	H e l l o
printf("%10s","Hello");	␣ ␣ ␣ ␣ ␣ H e l l o

```
printf("%-10s","Hello");
printf("%3s","Hello");
printf("%10.4s","Hello");
printf("%.3s","Hello");
```



3.5.5 混合数据的输出

在一条 printf 语句中可以输出以上所述的多种数据。例如：

```
int a=10;
float b=13.54;
char c[20]="China",d='A';
printf("%d %f %s %c",a,b,c,d);
```

该语句的输出为：

```
10␣13.540000␣China␣A
```

3.5.6 使用 printf 函数时的注意事项

在使用 printf 函数进行数据输出时，应该注意以下几点。

(1) 输出数据项的数目是任意的，但是必须在数目、类型和顺序上与格式控制字符串中格式转换说明保持一致。否则编译系统虽然不报语法错误，但将输出错误的结果。若输出项个数多于格式控制字符串中的格式转换说明的个数，则多余的项不被输出。

例如：

```
int  i=1,j=2;
double x=-5.5,y=12.345;
printf("i=%-4d,j=%4d,y=%6.2f \n", i, j, y, x);
```

输出为：

```
i=1␣␣␣,j=␣␣␣2,y=␣12.35
```

在输出项中的 x 值不被输出。

(2) 程序的输出经常用作分析变量之间的某些联系的信息，为决策提供依据。因此，输出的正确性和清晰度尤为重要。正确性取决于求解过程，而清晰度取决于输出的方法。因此，在程序输出时应该注意：①在两个数字之间提供足够的空格。②在输出中给出适当的标题和变量名。③在输出的两个部分之间加上空白行。

例如：

```
printf("a=%d\tb=%d",a,b);
```

上述语句在输出语句中给出了两个变量的名字，同时通过制表符'\t'，加大了两个数字之间的间距。而下面的语句通过在语句中使用回车换行字符'\n'将 a 和 b 在两行中显示。

```
printf("a=%d\nb=%d",a,b);
```

(3) 通过直接在 printf 语句中使用字符串,可以在输出中显示某些重要的提示和标题。如下面的语句所示。

```
printf("Please input a number:");
printf("Code \t Name \t Age \n");
```

3.6 格式化输入

在 C 语言中可以用 scanf 函数来实现不同类型数据的输入。scanf 函数的调用形式为:

```
scanf (格式控制字符串,地址列表);
```

其中的格式控制字符串与 printf 函数中类似。

该函数的功能是:从标准输入设备(键盘)上读取字符序列,并将它们按格式控制字符串中指定的格式转换为相应类型的值后,存储于地址表所指定的对应的变量中。

格式控制字符串通常包含以下几个部分。

(1) 空格或制表符,在处理过程中将被忽略。

(2) 普通字符,在输入流中相应位置必须有相同的字符与之匹配。

(3) 转换说明,依次由一个%,一个可选的赋值禁止字符 *,一个可选的数据(指定最大字段宽度),一个可选的 h、l 或 L 字符以及一个转换字符组成。

表 3.3 是 scanf 函数用到的格式字符。表 3.4 是 scanf 函数可以使用的附加格式说明字符。

表 3.3 scanf 函数的格式字符

格 式 字 符	对输入数据的要求
d	十进制整数
0	八进制整数(可以以 0 开头,也可以不以 0 开头)
x	十六进制整数
c	字符
s	字符串(不加引号)
e, f, g	实数(浮点数),它可以包括正负号(可选)、小数点(可选)及指数部分(可选)

表 3.4 scanf 的附加格式说明字符

附加说明字符	说 明
字母 l	表示输入长整型量(%ld、%lo、%lx)或 double 型量(%lf、%le)
h	表示输入短整型数据(%hd、%ho、%hx)
m(正整数)	表示输入数据的最小宽度
*	表示本输入项在读入后不赋给相应的变量

转换说明指定对输入字段的解释,对应的地址列表中的参数必须是地址。

地址列表由“,”分开的若干个地址组成,地址可以是简单变量的地址或字符数组的首地址。在地址表中,给出变量的地址是用变量名前面加取地址运算符“&”来表示的。字符串的首地址是用字符数组名或指向字符串首地址的指针变量名表示的。

例如：

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

若运行时输入(↙表示回车，下同)：

```
12↙
```

则结果是将 12 赋给变量 a。

在这个 scanf 函数的地址表中 &a 不可写成 a。& 是取地址运算符，&a 指出变量 a 在内存中的地址(将相应数据值送到该变量对应的地址单元中)。

```
char name[20];
scanf("%s",name);
```

在这个 scanf 函数的地址表中的参数是 name，它本身表示字符数组在内存中的首地址。

3.6.1 整数的输入

用于读取一个整数的格式说明符为：

```
%[m]d
```

其中，m 是一个整数，指定要读取的数字的域宽，可以省略。d 表明要读取的数据为整型数据。

例如：

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

若运行时输入：

```
12␣34↙
```

则结果是将 12 赋给变量 a，34 赋给变量 b。注意，在 12 和 34 之间有一个空格。

此时输入数据的各项必须用空白字符(空格、制表符或换行符)隔开，而不能用标点符号来分隔。当 scanf 函数从输入数据行读取数据时，将忽略所有的空白字符。

当 scanf 函数读取某个特定的值时，如果指定了域宽，则输入域直到域宽用完时为止；如果读取时遇到了一个不合法的字符，读取工作将被中止。

例如：

```
scanf("%2d%5d",&a,&b);
```

若输入为 12345678，则结果是将 12 赋给了 a，34567 赋给了 b，8 丢弃不用。

通过在格式说明符前加上输入抑制符 *，就可以跳过该输入数据，将其忽略。例如，语句：

```
scanf("%d%*d%d",&num1,&num2);
```

此时，如果输入为：

```
12␣34␣56↙
```

那么,12 赋给 num1,34 被忽略,不会赋值给任何变量(因为格式控制符中有 *),56 赋给 num2。

如果格式控制符是%ld,则用来读取长整型数据;如果换为%hd,则用来读取短整型数据。

3.6.2 实数的输入

输入实数时,不能指定实数的域宽,因此,scanf 函数只需使用简单的格式说明符%f 来读取实数,可用十进制小数或指数形式来输入实数。例如:

```
scanf("%f%f%f",&fa,&fb,&fc);
```

的输入数据为

```
123.456␣42.15E-2␣987
```

那么,123.456 赋值给了变量 fa,42.15E－2 赋值给了变量 fb,987 赋值给了变量 fc。注意,输入时在输入数据之间需用空格、换行符或者 Tab 键隔开。

如果要输入的数据为 double 类型,那么,格式说明符应为%lf。

【例 3.5】 各种实数的输入。

【程序】

```
#include <stdio.h>
int main()
{
    float fx,fy;
    double dx,dy;
    printf("enter two float (fx, fy): ");
    scanf("%f%f",&fx, &fy);
    printf("fx=%f\tfy=%f\n",fx, fy);
    printf("enter two double (dx, dy): ");
    scanf("%lf%lf",&dx, &dy);
    printf("dx=%lf\tdy=%e\n",dx,dy);
    printf("dx=%.4f\tdy=%12.3e\n",dx,dy );
    return 0;
}
```

【运行】

```
enter two float (fx, fy): 123.456 24.98E-2
fx=123.456001   fy=0.249800
enter two double (dx, dy): 3.1415926 12.3456789
dx=3.141593     dy=1.234568e+001
dx=3.1416       dy=  1.235e+001
```

3.6.3 字符和字符串的输入

利用 getchar 函数仅能输入单个字符,而利用 scanf 函数不仅可以输入单个字符,而且

还可以输入含有多个字符的字符串。输入字符串的格式说明符如下。

(1) 输入字符的格式说明符为：%c。例如：

```
scanf("%c",&ch);
```

的输入数据为

```
B↙
```

那么，字符 A 赋值给了字符变量 ch。需要注意的是，空格符、换行符、制表符也都是字符。

(2) 输入字符串的格式说明符为：%[m]s 或%[m]c，相应的参数应该是字符数组的名字或是指向字符数组的指针。例如：

```
char name[20];
scanf("%s",name);
```

的输入数据为

```
Jack↙
```

那么，name 中将保留输入的字符串"Jack"。这里的 name 是字符数组名，表示字符串的首地址。

【例 3.6】 输入字符和字符串。

【程序】

```
#include <stdio.h>
int main()
{
    char name1,name2[20],name3[20];
    printf("Please input name1:\n");
    scanf("%c",&name1);
    printf("name1 :%10c\n", name1) ;
    printf("Please input name2:\n");
    scanf("%s",name2);
    printf("name2 :%10s\n",name2);
    printf("Please input name3:\n");
    scanf("%10s",name3);
    printf("name3 :%10s\n",name3);
    return 0;
}
```

【运行】

第一次运行：

```
Please input name1:
A
name1 :         A
Please input name2:
Hubei Wuhan
name2 :     Hubei
Please input name3:
name3 :     Wuhan
```

第二次运行：

```
Please input name1:
A
name1 :         A
Please input name2:
Hubei-Wuhan
name2 :Hubei-Wuhan
Please input name3:
China
name3 :     China
```

【说明】

(1) 当使用%c输入字符时，系统将空格符、换行符、制表符都作为字符输入。

(2) 当使用%ms输入字符串时，遇到空白字符(空格符、换行符、制表符)，读取工作将被终止。

因此，在第一次运行的时候，name2只读取了输入的字符串"Hubei Wuhan"的前一部分"Hubei"，而后一部分"Wuhan"则自动赋给了name3。而在第二次运行时，通过在"Hubei"和"Wuhan"之间加上连字符号，name2和name3都读取了正确的字符串。

3.6.4 混合数据类型的输入

在scanf函数中，通过选用不同的格式转换字符，可实现同时输入多个不同简单类型的数据。例如：

```
char  ch;
int  a;
short  b;
long  c;
double  x;
scanf("%d%hd%ld%c,%*c%lf", &a,&b,&c,&ch,&x);
```

若从键盘输入数据：

```
32␣40␣123A,B␣6.3↙
```

则结果是将32赋值给变量a，40赋值给变量b，123赋值给变量c，字符'A'赋值给变量ch，6.3赋值给变量x。而其中与%*c指定的转换说明相匹配的输入域的值，即字符'B'被跳过，不赋给任何变量。

特别需要注意的是，以%c格式输入字符时，空格字符和转义字符都作为有效字符，所以输入时123与'A'之间不用空格分开，否则赋值给ch的就是空格字符' '；在格式说明“%c,%*c”中的逗号属于普通字符，需要输入同样的字符与之匹配，所以输入数据“123A,B”中的逗号必须输入。

3.6.5 使用scanf函数时的注意事项

(1) 格式控制字符串中的格式说明与地址表中变量的类型要一致，否则输入变量中的数据可能不是所希望得到的值(在某些编译器中不提示语法错误)。

(2) 格式说明的个数应与地址表中变量的个数相同。若格式说明的个数少于地址表中

变量的个数，则地址表中右边多出的变量将不被赋值；若格式说明的个数多于地址表中变量的个数，因输入数据没有被指定存储地址而可能导致难以预料的后果。

(3) scanf 遇到下列情况之一时终止读取数据。

① 在输入数字时发现有一个空白字符。

② 已经读取了数据的域宽个数。

③ 检查出了一个错误。

(4) 数据输入形式。

① 输入时，在每个输入项之间可以用空白字符(空格符、换行符或制表符)隔开。例如：

```
scanf("%f%d",&num1,&num2);
```

执行时若输入：

```
12.35␣34↙
```

或者执行时输入：

```
12.35↙
34↙
```

则 12.35 被赋值给 num1，34 被赋值给 num2。

② 整型、浮点型或字符型数据后面的字符型不用分隔符隔开，否则将把分隔符赋值给字符型变量。例如：

```
int num;
char ch;
scanf("%d%c",&num,&ch);
```

执行时若输入：

```
12a↙(2 和 a 之间无空格)
```

则 12 被赋值给 num，'a'被赋值给 ch。

执行时若输入：

```
12␣a↙(2 和 a 之间有一个空格)
```

则 12 被赋值给 num，空格字符' '被赋值给 ch，而输入的'a'被忽略，不会赋给任何变量。

③ 整数、浮点型或字符型数据后面的字符串数据可以有或无空白符；但是一个字符串内部不能有空白符，因为空白符是字符串输入结束的标志。例如：

```
int day,year;
char month[20];
scanf("%d%s%d",&day,month,&year);     //注意:month 前无取地址运算符
```

执行时若输入：

```
20␣Dec␣2006
```

或执行时输入：

```
20Dec  2006
```

则都将整数 20 赋值给变量 day，字符串"Dec"赋值给数组 month，整数 2006 被赋值给变量 year。

④ 如果在格式控制字符串中包含有普通字符，则输入时必须输入同样的字符与之匹配。例如：

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

运行时，从键盘输入数据必须按如下所示进行。

```
a=3,b=6↙
```

又例如：

```
scanf("%d\n",&num);
```

运行时，从键盘输入数据必须按如下所示进行。

```
34\n↙
```

在这个输入函数中，'\n'是普通字符，输入时必须原样输入 num 才会被赋值。

所以，在 scanf 函数的格式控制字符串中一般不要出现除格式转换符以外的其他字符，如上面例子中的"a＝"、"，b＝"、"\n"等。

3.7　简单程序设计

前面介绍了 4 种输入输出函数的调用形式及使用方法，它们也属于表达式语句，称为函数调用表达式语句。一般也简称为输入语句和输出语句。有了这些语句，就可以实现顺序结构程序设计。

简单程序是仅包含一个 main 函数的简单的 C 语言程序，只要适当运用这些表达式语句，就可以设计出完成某个特定功能的简单 C 语言程序。

输入、处理和输出数据是计算机程序设计的 3 个基本功能。在简单程序设计中要完成这 3 个基本功能。

下面通过几个例子来说明简单程序设计的方法。

【例 3.7】 从键盘上输入 3 个整数，计算并输出它们的和及平均值。

【程序】

```
#include <stdio.h>
int main()
{
    int a,b,c,sum;
    double aver;
    printf("enter three integers(a,b,c)\n");
    scanf("%d%d%d",&a,&b,&c);          //①
    sum=a+b+c;                         //②
```

```
    aver=sum/3.0;                          //③
    printf("sum=%4d\naverage=%6.2lf\n",sum,aver);
    return 0;
}
```

【运行】

```
enter three integers(a,b,c)
5 9 14
sum=  28
average=  9.33
```

【说明】

(1) 在这个程序中,①完成的是数据的读取工作,②完成的是数据的处理工作,③完成的是数据的输出工作。任何程序都是由输入(或赋值)、处理和输出 3 部分顺序组成的。

(2) 因为平均值需要保留小数部分,所以在“aver=sum/3.0;”这条语句中除数是 3.0,而不是整数 3。

【例 3.8】 输入两个字符,输出用这两个字符绘制的三角形。

【程序】

```
#include <stdio.h>
int main()
{
    char ch1, ch2;
    printf("Enter two characters :");
    scanf("%c%c",&ch1,&ch2);
    printf("%4c\n",ch1);
    printf("%3c%c%c\n",ch1,ch2,ch1);
    printf("%2c%c%c%c%c\n", ch1,ch2,ch2,ch2,ch1) ;
    printf("%c%c%c%c%c%c%c\n", ch1,ch1,ch1,ch1,ch1,ch1,ch1);
    return 0;
}
```

【运行】

```
Enter two characters :*A
   *
  *A*
 *AAA*
*******
```

【例 3.9】 已知三角形三条边的长度,求该三角形的面积。

【分析】 该程序需要接收的输入为三角形的三条边长,设三角形三条边的长度分别用 a、b、c 表示,定义为整型,程序需要显示输出的是三角形的面积。假设三角形的三条边长通过键盘输入并且可以构成三角形,面积用 area 表示,这时可以根据下列数学公式计算三角形的面积。

$$area=\sqrt{s(s-a)(s-b)(s-c)} \quad 其中 \quad s=\frac{a+b+c}{2}$$