第 3 章　数据的输入与输出

数据的输入/输出是程序设计中使用最普遍的基本操作。本章首先简要介绍输入/输出的概念和 C 语言输入/输出的实现方法，其次详细介绍用于基本类型数据输入的库函数 scanf 和用于基本类型数据输出的库函数 printf。

【任务 3.1】计算圆的面积和周长（改进版）

任务描述：从键盘上输入圆的半径，求圆的面积和周长。

任务分析：在任务 2.1 中，程序只能对给定的半径值计算圆的面积，如果要计算不同半径的圆的面积和周长，必须修改源程序。现在要求程序能够在运行过程中根据键盘输入的半径值计算圆的面积和周长。面积和周长的计算公式同 2-1 和 2-2。

程序需要考虑如何解决以下两个主要问题：

（1）程序如何接收来自键盘输入的数据并存储到变量中？

（2）程序如何输出变量中的值和计算结果？

3.1　C 语言的输入输出

输入/输出(简称 I/O)是程序的基本组成部分，程序运行所需的数据通常要从外部设备(如键盘、文件等)输入，程序的运行结果通常也要输出到外部设备(如显示器、打印机、文件等)。

C 语言没有专门的 I/O 语句，I/O 操作是通过 C 语言标准输入输出库 stdio.h 中声明的库函数来实现，如 getchar、scanf、putchar、printf 等。

基本类型数据的输入主要通过 scanf 来实现，基本类型数据的输出主要通过 printf 来实现。在使用 scanf 和 printf 等 I/O 库函数前，通常在程序开头必须增加下列代码：

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

即包含 C 语言标准输入输出库 stdio.h。其中，std 是 standard 的缩写，i 是 input 的缩写，o 是 output 的缩写；h 是 head 的缩写，用作头文件的扩展名。

3.2　字符的非格式化输入输出函数

字符的非格式化输入函数 getchar 的原型为：

```
int getchar();
```

它的功能是从标准输入设备(默认时为键盘)上读入一字符。若调用成功，则返回输入的字符，否则返回 EOF(-1)。例如：

c=getchar();

当执行上述语句时，变量 c 获得一个从标准设备上读取的字符代码值。当从键盘上输入 Ctrl+Z(即先按下 Ctrl 键不放开，然后再按 Z 键)键时，c 得到的值是 EOF(-1)。

字符的非格式化输出函数 putchar 的原型为：

int putchar(int c);

它的功能是向标准输出设备(默认时为显示器)输出一字符。若调用成功，则返回字符 c，否则返回 EOF(-1)。

【例 3.1】 从键盘输入一个字符，并输出该字符及其后续字符。

```
#include<stdio.h>

int main( )
{
    char c;

    c=getchar( );     //从键盘读入一个字符
    putchar(c);       //在屏幕上输出所输入的字符
    putchar(c+1);     //在屏幕上输出所输入的字符的后续字符
    putchar('\n');    //换行

    return 0;
}
```

运行该程序时，从键盘输入一个字符，接着在显示器上即输出所输入的字符及后续字符。程序运行结果截图（假设输入字符为 a）如下：

3.3 格式化输出函数 printf

3.3.1 printf 函数概述

为了便于机器保存和处理，数据在内存中采用二进制形式；但为了便于人输入和阅读，输入和显示时均采用字符形式。因此，显示数据时应先将内存中的二进制数据转换成相应的字符序列，然后才可显示。例如，整数 123456789 在内存中的二进制形式为：

0000 0111 0101 1011 1100 1101 0001 0101

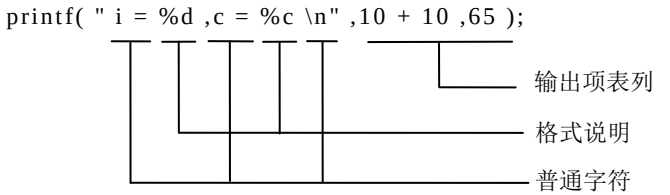
显示时，对应的八进制字符序列为："726746425"，十进制字符序列为："123456789"，十六进制字符序列为："75bcd15"。

printf 函数专用于将内存中基本类型的二进制数据按指定格式转换成对应的字符序列

并输出。其格式为：

printf(格式控制串,输出项表列)

它的功能是将格式控制串输出到标准输出设备(默认时为显示器)。格式控制串是用双引号括起来的字符串，包括普通字符和格式说明。格式控制串中的格式说明可以有 0 个或多个，应与输出项表列中的数据项依次一一对应。对于格式控制串中的普通字符在输出时原样输出；对于格式控制串中的格式说明，在输出时用指定的字符序列替换，而指定的字符序列是由输出项表列中指定的二进制数据按格式说明转换所得。输出项表列给出待转换的数据，多个数据项之间用逗号(,)分隔，每个数据项均为表达式。printf 函数若调用成功，则返回输出的数据个数，否则返回一个负数。例如：



其输出过程是：普通字符"i="原样输出；"%d"是第一个格式说明，含义是将输出项表列中的第一个表达式(10+10)转换成十进制字符序列("20")，输出时用十进制字符序列("20")替换"%d"。类似的，普通字符",c="原样输出；"%c"是第二个格式说明，含义是将输出项表列中的第二个表达式(65)转换成 ASCII 字符('A')，输出时用 ASCII 字符('A')替换"%c"。最后输出普通字符'\n'。即 printf 函数最终输出的字符序列为："i=20,c=A\n"。

若格式控制串中不含格式说明，则输出项表列可省略，此时 printf 函数可简化为：

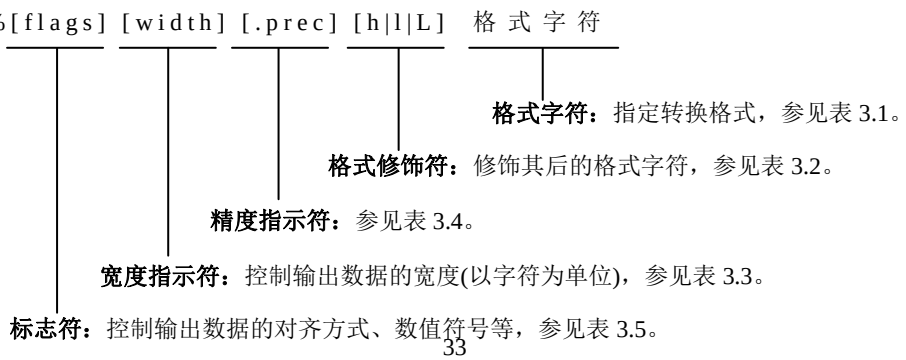
printf(格式控制串);

例如：

printf("Welcome to C world!\n");

3.3.2 printf 函数的格式说明

printf 函数有丰富的格式说明，用于转换并输出 C 语言的各种基本类型数据。格式控制串的每个格式说明是由%开始、依次由标志字符序列、宽度指示符、精度指示符、格式修饰符和格式字符组成，其一般格式为：



它的作用是将输出项表列中的指定数据按指定格式转换成字符序列并输出。其中，方括号括起来的内容为可选项。

表 3-1　printf 函数的常用格式字符

格式字符	转换格式	举例	输出结果
d	带符号十进制整数	printf("%d",25);	25
i	带符号十进制整数	printf("%i",25);	25
x	无符号十六进制整数(字母用 a、b、c、d、e、f)	printf("%x",255);	ff
X	无符号十六进制整数(字母用 A、B、C、D、E、F)	printf("%X",255);	FF
o	无符号八进制整数	printf("%o",35);	43
u	无符号十进制整数	printf("%u",35);	35
c	ASCII 字符	printf("%c",'A');	A
s	字符串	printf("%s","This");	This
E 或 e	指数形式的浮点数。默认精度 6 位小数。指数部分占 5 位(如 e+002)，其中“e”占 1 位，指数符号占 1 位，指数占 3 位。数值规格化(小数点前有且仅有 1 位非零数字)。	printf("%E",-475.3751); printf("%e",-475.3751);	-4.753751E+002 -4.753751e+002
f	小数形式的浮点数。默认精度 6 位小数。	printf("%f",-475.3751);	-475.375100
G 或 g	G(或 e)和 f 中较短的一种，不输出无意义的 0。	printf("%g",-475.3751);	-475.375
p	无符号十六进制整数，用于显示变量的指针值。	int y=25; printf("%p",&y);	0012FF7C
%	符号"%"本身	printf("%%");	%

表 3.2 printf 函数的格式修饰符

格式修饰符	作用	举例	输出结果
h	表示 short。对格式字符 d、i、o、u、x 和 X 有效，主要用于输出 short int 和 short unsigned int 型数据	short int a=1000; printf("%hd",a);	1000
l	表示 long。用于输出整型和长整型数据时，对格式字符 d、i、o、u、x 和 X 有效；用于输出 double 型浮点数据，对格式字符 e、E、f、g 和 G 有效	long a=1345678; printf("%ld",a);	1345678
L	用于输出 long double 型浮点数据，对格式字符 e、E、f、g 和 G 有效		

表 3.3 printf 函数的宽度指示符

宽度指示符	作用	举例	输出结果
n	至少输出占 n 个字符。若实际输出字符不足 n 个，则空位用空格填充(在给定标志字符'-'时，右边填空格，否则左边填空格)	printf("%5d",20); printf("%-5d",20);	␣␣␣20 20␣␣␣
0n	至少输出 n 个字符，若结果少于 n 个字符，则左边填 0(零)	printf("%05d",20); printf("%-05d",20);	00020 20␣␣␣
注．␣表示空格。下同。			

表 3.4 printf 函数的精度指示符

精度指示符	作用	举例	输出结果
无	默认精度： 对 d、i、o、u、x 格式符：1 对 e、E、f 格式符：6 位小数 对 g、G 格式符：所有有效数字 对 s 格式符：输出到第一个'\0' 对 c 格式符：无影响	参见表 3.1	
.0	对 d、i、o、u、x 格式符：默认精度 对 e、E、f 格式符：不输出小数点	printf("%.0d",20); printf("%.0f",20.5);	20 21
.n	对于字符串，表示截取的字符个数；对于实数，表示输出 n 位小数	printf("%.1s","Hi"); printf("%.1f",2.56);	H 2.6
.*	在待转换数据前的数据中指定待转换数据的精度。如右例中的参数 3 在待转换数据 2.1 前指定数据 2.1 的转换精度为 3 位小数	printf("%.*f",3,2.1);	2.100

表 3.5 printf 函数的标志符

标志符	作用	举例	输出结果
无	结果右对齐，左边填空格或零	printf("%5d",20);	␣␣␣20
-	结果左对齐，右边填空格	printf("%-5d",20);	20␣␣␣
+	带符号的转换。若结果非负则以正号(+)开头，否则以负号(-)开头	printf("%+5d",20); printf("%-+5d",-2);	␣␣+20 -2␣␣␣
空格	若结果非负，则输出以空格代替正号，否则以符号开头	printf("%␣5d",20); printf("%␣5d",-2);	␣␣␣20 ␣␣␣-2

3.3.3 printf 函数的使用

printf 函数有丰富的格式说明，初学时可从 printf 函数输出基本类型数据开始逐步掌握。下面通过一系列程序片段，进一步说明 printf 函数输出基本类型数据的方法。

（1）int a=123,b=12345;

printf("%d,%4d,%4d",a,a,b);

输出结果为：123,␣123,12345

说明：

第一个数据输出时，格式上没有说明所占位数，则按实际位数输出。

第二个数据输出时，格式上说明所占位数为 4 位，而实际位数只有 3 位，由于此时输出数据在所占 4 个字符位置的默认对其方式是右对齐，因此，最右边 3 个位置输出”123”，左边空出 1 个位置为空格字符。

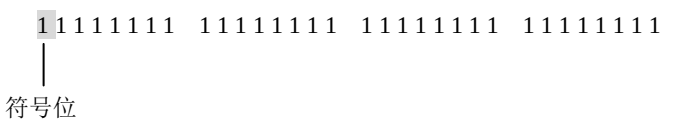
最后一个数据输出时，虽然格式上限制为占 4 位，但由于其实际值超出 4 位，为体现出正确性优先原则，按实际位数输出。

（2）int a=-1;

printf("%d,%u,%o,%x",a,a,a,a);

输出结果为：-1,4294967295,37777777777,ffffffff

说明：-1 在内存单元中的存放形式(以补码形式存放)如下：



u 格式字符以无符号十进制数形式输出整数(连同符号位)，o 格式字符和 x 格式字符，分别以八进制数形式和十六进制数形式输出整数(连同符号位)。

（3）char c='a';

printf("%c,%d\n",c,c);

输出结果为：a,97

说明：字符数据以”%d”格式输出时，输出的是相应字符的 ASCII 码。

（4） printf("%3s,%6.2s,%.3s,%-5.3s\n","Good","Good","Good","Good");

输出结果为：Good,␣␣␣␣Go,Goo,Goo␣␣

说明：

第一个格式说明为"%3s"，指定输出的字符串"Good"占 3 列。如字符串本身长度大于 3，则将字符串全部输出。若串长小于 3，则左补空格。

第二个格式说明为"%6.2s"，指定输出的字符串"Good"占 6 列，但只取字符串中左端 2 个字符。这 2 个字符输出在 6 列的右侧，左补空格。

第三个格式说明为"%.3s"，指定输出的字符串"Good"仅占 3 列，只取字符串中左端 3 个字符。

第四个格式说明为"%-5.3s"，指定输出的字符串"Good"占 5 列，只取字符串中左端 3 个字符。这 3 个字符输出在 5 列的左侧，右补空格。

（5） float x=111111.111f,y=3.1415926f;

printf("%f,%6f,%6.2f,%-6.2f,%.2f\n",x+x,y,y,y,y);

输出结果为：222222.218750,3.141593,␣␣3.14,3.14␣␣,3.14

说明：

第一个格式说明为"%f"，采用默认精度 6，使第一个数据项 x+y 的整数部分全部输出，并输出 6 位小数。应当注意，并非全部数字都是有效数字，可以看到最后 5 位小数(超过 7 位)是无意义的。单精度实数的有效位数一般为 7 位，双精度实数的有效位数一般为 16 位。

第二个格式说明为"%6f"，输出宽度至少 6 个字符位，采用默认精度 6，使第二个数据项 y 的整数部分全部输出，并输出 6 位小数。实际输出占 8 个字符位。

第三个格式说明为"%6.2f"，输出宽度至少 6 个字符位，采用指定精度 2，使第三个数据项 y 的整数部分全部输出，并输出 2 位小数。实际只输出 4 个字符，不足 6 个，左补 2 个空格。

第四个格式说明为"%-6.2f"，左对齐，输出宽度至少 6 个字符位，采用指定精度 2，使第四个数据项 y 的整数部分全部输出，并输出 2 位小数。实际只输出 4 个字符，不足 6 个，右补 2 个空格。

第五个格式说明为"%.2f"，未指定宽度，采用指定精度 2，使第五个数据项 y 的整数部分全部输出，并输出 2 位小数。实际只输出 4 个字符。

（6） float x=314.15926f;

printf("%e,%10e,%10.2e,%-10.2e,%.2e",x,x,x,x,x);

输出结果为：

3.141593e+002,3.141593e+002,␣3.14e+002,3.14e+002␣,3.14e+002

说明：

第一个格式说明为"%e"，采用默认精度 6，使第一个数据项 x 的尾数部分输出 1 位整数、一个小数点和 6 位小数，指数部分输出：“e”占 1 位，指数符号占 1 位，指数占 3 位。

第二个格式说明为"%10e"，至少占 10 个字符位，采用默认精度 6。结果同上。

第三个格式说明为"%10.2e"，至少占 10 个字符位，采用指定精度 2，使第三个数据项 x 的尾数部分输出 1 位整数、一个小数点和 2 位小数，指数部分输出：“e”占 1 位，指数符号占 1 位，指数占 3 位。实际输出 9 个字符，不足 10 个，左补一个空格。

第四个格式说明为"%-10.2e"，左对齐，至少占 10 个字符位，采用指定精度 2，使第四个数据项 x 的尾数部分输出 1 位整数、一个小数点和 2 位小数，指数部分输出：“e”占 1 位，指数符号占 1 位，指数占 3 位。实际输出 9 个字符，不足 10 个，右补一个空格。

第五个格式说明为"%.2e"，采用指定精度 2，使第三个数据项 x 的尾数部分输出 1 位整数、一个小数点和 2 位小数，指数部分输出：“e”占 1 位，指数符号占 1 位，指数占 3 位。由于未指定至少占多少个字符位，则按实际输出 9 个字符输出。

3.4 格式化输入函数 scanf

3.4.1 scanf 函数概述

用户从键盘输入的数据实际上都是字符序列，需按指定的数据类型转换成相应的二进制数据，才可存入内存。例如，从键盘输入十进制整数 123456789 时，实际输入的是数字字符序列，接下来需要将这个数字字符序列按整数类型转换成二进制整数形式：

0000 0111 0101 1011 1100 1101 0001 0101

再存入内存。这一转换过程由编程者来实现是有一定难度的。但由于实际编程时这一转换过程具有普遍性，因此 C 语言标准输入输出库(stdio.h)提供了一个 scanf 函数来完成这一转换过程。

scanf 函数用于从键盘输入字符序列数据，并按指定格式(由格式控制串指定)转换成相应基本类型的二进制数据存入内存(由输入项表列指定)。其一般格式为：

scanf("格式控制串",输入项表列);

其中，格式控制串是用双引号括起来的字符串，包括普通字符和格式说明。格式控制串中的格式说明应与输入项表列中的数据项依次一一对应。格式控制串中的普通字符在输入时要原样输入。输入项表列给出待转换的数据存放的内存位置，多个数据项之间用逗号(,)分隔。scanf 函数若调用成功，则返回输入、转换和保存的数据个数；若没有数据被保存，则返回 0；若读到文件结束(从键盘输入 Ctrl+Z)，则返回 EOF(-1)。例如：

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

其中，&a 和&b 分别表示变量 a 和 b 在内存的位置。运行时从键盘输入下列字符序列：

30␣20↙

scanf 函数对输入的第一个字符串"30"，按第一个格式说明指定的格式"%d"，转换成十进制数值 30，再以二进制形式存入输入项表列的第一项&a 指定的内存位置。类似地，对输入的第二个字符串"20"，按第二个格式说明指定的格式"%d"，转换成十进制数值 20，再以二进制形式存入输入项表列的第二项&b 指定的内存位置。

若在格式控制串中未指定字符作为输入数据之间的分隔，则在输入数据时，在两个数据之间以一个或多个空格间隔(如上面的输入所示)，也可以用回车键、跳格键 tab。下面输入均为合法：

①30␣␣␣20↙

②30↙

20↙

③30(按 tab 键)20↙

但以下输入不合法：

30,20↙

若希望用指定字符作为输入数据之间的分隔，可在 scanf 函数的格式控制串中指定。例如，为了使上述输入合法正确，可将 scanf 函数改写成：

```
scanf("%d,%d",&a,&b);          //用逗号(,)作为两个输入数据的分隔
```

必须注意，一旦在 scanf 函数的格式控制串中用指定字符作为输入数据之间的分隔，实际输入数据时，若忘记在输入数据之间原样输入指定的分隔字符，则会导致输入出错。

3.4.2 scanf 函数的格式说明

与 printf 函数类似，scanf 函数也有丰富的格式说明，用于将输入的字符序列转换成基

本类型数据。格式控制串的每个格式说明是由%开始、依次由星号(*)、宽度指示符、格式修饰符和格式字符组成，其一般格式为：

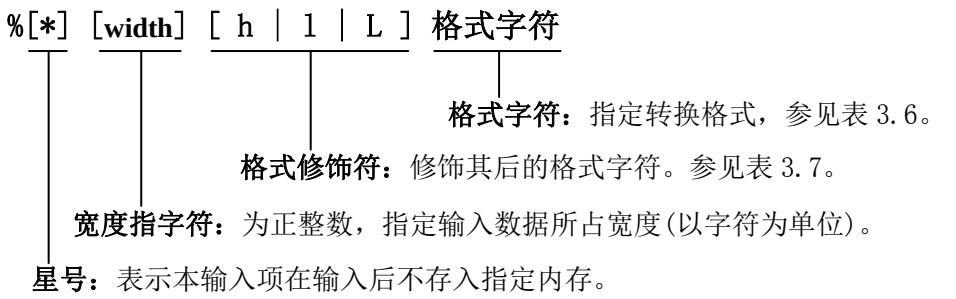


表 3.6　scanf 函数的格式字符

格式字符	将输入的字符序列转换成	数据的输入格式
d	有符号的十进制整数。	[-|+]dd…d
i	有符号的八、十或十六进制整数。	[-|+][0[x]]dd…d①
u	无符号的十进制整数。	[-|+]dd…d
o	无符号的八进制整数。	[-|+]dd…d②
x	无符号的十六进制整数。	[-|+][0x]dd…d③
c	单个字符。	
s	字符串，将字符串送到一个字符数组中。在输入时以非空白字符开始，以第一个空白字符结束。字符串以'\0'作为其最后一个字符。	
f	实数。可用小数形式或指数形式输入。	
e,E,g,G	实数。与格式字符 f 作用相同。	
注：①数字的进制由第一个数字确定，与 C 语言常量一样。例如，输入 12 为十进制数，012 为八进制数，0x12 为十六进制数。 ②数字为八进制。 ③数字为十六进制，不管是否有 0x。		

表 3.7　scanf 函数的格式修饰符

格式修饰符	用于转换输入的
l	长整型数据(可用%ld，%li，%lu，%lo，%lx)以及 double 型数据(用%lf 或%le)。
h	短整型数据(可用%hd，%hi，%hu，%ho，%hx)。
L	long double 型数据(用%Lf 或%Le)。

3.4.3　scanf 函数的使用

1．输入整数

设有下列变量说明：

```
int a,b;
```

unsigned c,d,e;
short int f;

（1）输入有符号十进制整数可用格式字符 d，如：

scanf("%d",&a);

在格式控制串中若指定输入数据所占字符数，则系统自动按它截取所需数据。如：

scanf("%3d%d",&a,&b);

运行时若输入：11112↙

则系统自动将前三个字符 111 转换成整数后存入变量 a，将剩余的字符 12 转换成整数后存入变量 b。但这种方法输入数据时容易出错，建议少用。

（2）输入有符号八进制、十进制和十六进制整数可用格式字符 i，如：

scanf("%i",&a);

输入数据按何种进制转换，取决于运行时的输入。若运行时输入：077↙

则为输入八进制数据。若运行时输入：99↙

则为输入十进制数据。若运行时输入：0x77↙

则为输入十六进制数据。

（3）输入无符号八进制、十进制和十六进制整数分别用格式字符 o、u 和 x，如：

scanf("%o%u%x",&c,&d,&e);

运行时若输入：77 99 ff↙

则系统自动将 77(八进制)转换成整数后存入变量 c，将 99(十进制)转换成整数后存入变量 d，将 ff(十六进制)转换成整数后存入变量 e。但若输入非法字符，则输入数据将出错。例如，输入八进制数据时，合法字符为+、-、0～7；输入十进制数据时，合法字符为+、-、0～9；输入十六进制数据时，合法字符为+、-、0～9、a～f、A～F。

（4）输入有符号十进制短整数可用格式字符 d 和格式修饰符 h，如：

scanf("%hd",&f);

2．输入实数

设有下列变量说明：

float x;

double y;

对于 float 型实数的输入可用格式字符 f 或 e，如：

scanf("%f",&x);

也可指定输入数据所占字符数，如：

scanf("%5f",&x);

若运行时输入：12.34↙

则系统自动将前 5 个字符 12.34 转换成实数后存入变量 x。

对于 double 型实数的输入必须用格式字符 lf 或 le，如：

scanf("%lf",&y);

注意，若仅用格式字符 f 或 e，则无法正确输入 double 型实数。

3．输入字符

格式字符 c 可用于转换输入的一个字符数据，例如：

```
char c,d,e;
scanf("%c%c%c",&c,&d,&e);
```

注意，格式字符 c 转换输入的字符时，不跳过空白符(空格、回车和跳格)，这与其他格式字符不同。例如，对于上述语句，若运行时输入：␣a␣b␣c↙，则变量 c 得到的字符为空格，变量 d 得到的字符为'a'，变量 e 得到的字符为空格。

4．输入字符串

格式字符 s 可用于转换输入的一个字符数据，详细介绍参见 6.3 节。

5．其他

（1）格式控制串中的“*”表示跳过它指定的数据项。例如，设：

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

运行时若输入：1␣2␣3↙

则读入字符串"1"并转换成整数 1 送给变量 a，%*d 表示读入一个字符串"2"并转换成整数 2 但不送给任何变量，最后再读入字符串"3"并转换成整数 3 送给变量 b。再如，下列语句：

```
scanf("%*c");
```

可使程序运行到该语句时暂停，等待用户按回车键继续。

（2）在输入数据时，若遇下列情况则认为该数据结束。

①遇空白符(空格、回车或跳格)。

②取完指定的个数的字符。如"%3d"，只取 3 个字符。

③遇非法字符。如输入十进制数时出现 a、b 等。

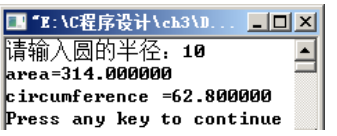
④遇文件结束符 EOF(DOS、Windows 操作系统为 Ctrl+Z 组合键，UNIX 操作系统为 Return+Ctrl+d 组合键，Macintosh 操作系统为 Ctrl+d 组合键)。

3.5 完成任务 3.1 的参考程序

```
#include<stdio.h>
#define PI 3.14

int main()
{
    int radius;
    double area, circumference;
```

```
    printf("请输入圆的半径：");
    scanf("%d", & radius);
    area = PI* radius * radius;
    circumference = 2*PI *radius;
    printf("area=%f\n", area);
    printf("circumference =%f\n", circumference);

    return 0;
}
```

程序运行结果截图如下（输入的半径值为 10）：



3.6 程序设计实例

输入/输出是编程的基本操作，几乎每个程序都离不开，应通过编写和调试程序来逐步深入而自然地掌握。

【例 3.2】编写一个程序，将以吋为单位的长度值转换成以厘米为单位的长度值，输出结果的精度为 10^{-3}。转换公式为：1 吋=2.54 厘米。请按以下输入/输出格式编程：

输入格式：

请输入长度值(吋)?12.3↙

输出格式：

12.300 吋＝31.242 厘米

分析：

编程解决实际问题时，一要确定数据结构，二要确定算法，三要确定输入输出界面。

（1）确定数据结构

根据题目要求，本题首先需要定义一个变量保存输入的以吋为单位的长度值。根据变量取名见名知意的原则，这里取变量名为 inch。

接下来还需为变量确定数据类型。确定变量的类型时，主要考虑该变量的取值、取值范围及取值精度，以占用较少内存的类型为佳。对于 inch 变量而言，因其保存的是实数，故其数据类型选 float(占 4 字节，精度达 10^{-7})或 double(占 8 字节，精度达 10^{-15})均可；但从 inch 变量所存实数的精度（10^{-3}）来看，其数据类型选 float 为佳。

类似地，本题定义的另一个变量取名为 cm，类型为 float，用于保存转换后的长度值(厘米)。

（2）确定算法

本题的算法是已知的，即 1 吋=2.54 厘米。

（3）确定输入输出界面

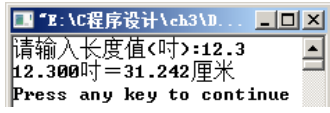
程序的输入/输出界面直接面对用户，为了方便用户使用，应该十分重视程序的输入/输出界面设计。通常，输入数据时应有必要的提示信息，输出数据时也要注意可读性。

本题的输入/输出界面已由题目给定，按照题目要求实现即可。

参考程序：

```
#include<stdio.h>

int main()
{
    float inch,cm;        //变量 inch 保存输入的吋值()，变量 cm 保存转换后的厘米值

    printf("请输入长度值(吋):");
    scanf("%f",&inch);

    cm=2.54f*inch;                                    //吋转换成厘米

    printf("%.3f 吋＝%.3f 厘米\n",inch,cm);     //置输出结果的精度，并输出结果

    return 0;
}
```

程序一次运行结果截图如下（输入长度值为 12.3）：



【例 3.3】给定三个小写字母，输出其 ASCII 码和对应的大写字母。

分析：

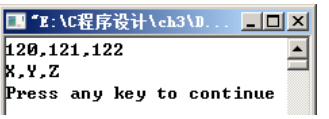
从 ASCII 码表可知，小写字母的 ASCII 码比相应的大写字母的 ASCII 码大 32，而且 C 语言允许字符数据与整数直接进行算术运算，因此本题可以通过将变量的值-32，并通过字符格式“%c”输出，就可得到其相应的大写字母。

参考程序：

```
#include<stdio.h>

int main()
{
    char a='x',b='y',c='z';

    printf("%d,%d,%d\n%c,%c,%c\n",a,b,c,a-32,b-32,c-32);
```

```
    return 0;
}
```

程序运行结果截图如下：



本章小结

本章的主要内容总结如下：

（1）字符的非格式化输入输出函数。

（2）格式化输出函数 printf。

（3）格式化输入函数 scanf。

程序对数据的处理通常分为数据输入、数据处理和数据输出三个阶段。本章的重点就是数据输入和数据输出。

习题 3

一、选择题。

1．下列程序片段的输出结果是（　　）。

```
int a=2,c=5;
printf("a=%%d,b=%%d\n",a,c);
```

A．a=%2,b=%5　　B. a=2,b=5　　C. a=%%d,b=%%d　　D. a=%d,b=%d

2．运行下列程序片段时，（　　）输入能使 a=1，b=2，c='+'。

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

A. 1␣+␣2↙　　B. 1+2↙　　C. 1␣+2↙　　D. 1+␣2↙

3．运行下列程序片段时，（　　）输入不能使 a=1，b=2。

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

A. 1␣2↙　　B. 1,2↙　　C. 1-2↙　　D. 1aa2↙

二、简答题。

1．写出下列语句的输出结果。

```
printf("%c,%c,%d",'a','a'+2,2+'a');
```

2．指出下列语句存在的错误，分析出错原因并改正：

(1)int a,b;

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

(2)float x;

scanf("%5.2f",&x);

(3)double x;

scanf("%f",&x);

(4)printf("You should say "Hi"!");

3．设有语句：

```
char c1,c2,c3;
scanf("%c%c%c",&c1,&c2,&c3);
```

当执行以上两个语句时，若输入：

'a'␣b␣c↙

则 c1、c2、c3 的值分别是什么？若在其执行过程中，输入

abcdef↙

则 c1、c2、c3 的值又分别是什么?

4．分析下列程序，分别写出输入数据 5␣12.3↙和-5␣–12.3↙的运行结果。

```
#include<stdio.h>

int main()
{
    int a;    float x;

    scanf("%d%f",&a,&x);
    printf("%d,%3d,%-3d,%03d,%+03d\n",a,a,a,a,a);
    printf("%8.2f,%.4f,%3f\n",x,x,x);
    printf("%-8.2f,%-.4f,%-3f\n",x,x,x);
    printf("%+8.2f,%+.4f,%+3f\n",x,x,x);
    printf("%+-8.2f,%+-.4f,%+-3f\n",x,x,x);
    printf("%e,%10.2e,%.2e,%+.2e,%-.2e,%+-.2e\n",x,x,x,x,x,x);

    return 0;
}
```

5．若希望运行下列程序片段后，变量 a、b、h、m、s 的值分别为 3、7、12、58、31，应如何在键盘上输入数据？

```
int a,b,h,m,s;
scanf("a=%d b=%d",&a,&b);
scanf("%d:%d:%d",&h,&m,&s);
```