第3章

顺序结构程序设计

从田忌赛马说起——

齐国的大将田忌常同齐威王进行跑马比赛。赛前，双方各下赌注，每次比赛共设三局，胜两次及以上为赢家。然而，每次比赛，田忌总是输给齐威王。

这天，田忌赛马时又输给了齐威王。回家后，田忌十分郁闷，他把赛马失败的事情告诉了孙膑。孙膑是大军事家孙武的后代，足智多谋，熟读兵书战策，深谙兵法，只是曾被魏国将军庞涓谋害造成双腿残疾，不能率兵打仗。他被田忌救到齐国后，很受器重。田忌待他为上宾，请他当了军师。

孙膑说："将军与大王的马我看了。其实，将军的三等马匹与大王的都差那么一点儿。您第一局派出的是上等马与大王的上等马比赛，第二局派中等马与大王的中等马比赛，第三局派下等马与大王的下等马比赛。您这样总按常规派出马与大王的马比赛，您永远会输。"田忌不解地问："不这样，又怎么办呢？"孙膑对田忌说："下次赛马时，您照我说的办法派出马匹，一定会取胜的，您只管多下赌注就是了。"田忌听了，大喜。这次他主动与齐威王相约，择日再进行赛马。齐威王听了，不屑地说："田将军又想给寡人送银子了，再比，将军也是输。"

赛马这一天到了。双方的骑士和马匹都来到赛马场上。齐威王和田忌在看台上饶有兴致地观看比赛。孙膑也坐着车子，坐在田忌的身旁。

赛马开始了，第一局田忌派出了自己的下等马，对阵齐威王的上等马。结果可想而知，田忌输掉了第一局。齐威王十分得意。第二局，田忌派出了自己的上等马对阵齐威王的中等马。结果，田忌赢了第二局。第三局，田忌派出自己的中等马对阵齐威王的下等马，田忌又赢了第三局。三局两胜，田忌第一次在赛马比赛中战胜了齐威王。由于事先田忌下了很大的赌注，他把前几次输掉的银子都赚了回来，还略有盈余。

这是一篇寓意深刻的寓言故事，引自《史记·孙子吴起列传》。这个故事告诉人们要有全局观念，如果能够在整体上取得重大胜利，就要舍得局部付出一点点牺牲和损失。在整体实力与对手相等或者略低于对手的时候，要想取得胜利，那就要看谁会巧妙地排兵布阵了，也就是比赛的顺序。齐威王的比赛的顺序就是上等马、中等马、下等马，而田忌的参赛顺序则是下等马、上等马、中等马，那结果可想而知，大概率会获得三局两胜。不同的顺序导致了不同的结果。所以我们做事情更要遵循事物的发展顺序和内在规律，才能事半功倍。编写程序亦是如此，结合事物内在的规律和程序目标，合理选择程序设计的基本结构。

顺序结构、选择结构和循环结构是程序设计的3种基本结构。用这3种基本结构能够

描述程序设计的任何算法和问题。这3种基本结构是结构化程序设计的基础。

本章将介绍顺序结构程序的设计，整型、实型、字符型的数据输入输出的格式及复合语句等。

3.1 顺序结构的程序特点


视频

【例 3.1】 求两整数的和。

```
main( )
{  int a,b, x;
   printf("请为 a,b 输入值:");
   scanf("%d%d",&a,&b);
   x=a+b;
   printf("x=%d\n",x);
}
```

运行结果为：

```
请为 a,b 输入值::20␣7↙
x=27
```

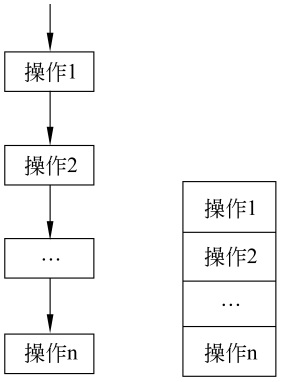
这里，“␣”表示空格，“↙”表示按 Enter 键。

上述程序完成的功能：从键盘输入两个整数，通过计算，输出此两整数的和。

该程序在执行时，函数体（即“{”和“}”之间的部分）中的语句是按书写顺序从上到下，一句一句地被执行的。

这种按语句出现的先后顺序逐条被执行的程序结构就称为顺序结构。

顺序结构程序的流程图和 N-S 图如图 3.1 所示。



(a) 流程图　　(b) N-S图

图 3.1　顺序结构的流程图和 N-S 图

顺序结构程序的特点如下。

（1）程序一般由下面几部分组成：

定义变量；

输入数据或赋值；

对数据进行操作（分析、运算）；

输出结果。

（2）顺序结构的程序在执行过程中严格按照语句书写的先后顺序逐一执行。每个语句都会被执行，并且只执行一次。

（3）顺序结构的程序结构简单，易于理解。

3.2 赋 值 语 句

赋值语句由赋值表达式加上分号构成。例如：

```
a=3.14;
```

而 a=3.14 则是赋值表达式。

再如：

```
a=5,b=7;
```

是赋值语句。而

```
a=5,b=7
```

则是两个赋值表达式。

要注意变量定义中赋初值和赋值语句的区别：在变量定义中，不允许连续给变量赋初值，而赋值语句允许连续赋值。例如：

```
int x=y=z=5;
```

是错误的声明方式(写成 int x=5,y=5,z=5;是正确的)。

```
int x,y,z;
x=y=z=5;
```

则是正确的赋值方式。赋值语句：

```
x=y=z=5;
```

相当于

```
x=(y=(z=5));
```

3.3 数据输入输出

数据输入是指把计算机需要的数据从输入设备(通常是键盘)送给计算机。数据输出是把计算机内存中的数据送到外部设备上，通常是显示器上。在 C 语言中，没有输入输出语句，输入和输出操作是通过调用库函数来完成的。

C 语言中的标准输入输出库函数有 printf(格式输出)、scanf(格式输入)、putchar(输出字符)、getchar(输入字符)、puts(输出字符串)、gets(输入字符串)等。本节介绍前 4 个最基本的输入输出函数。

注意：在调用标准输入输出库函数时，要用到“stdio.h”头文件，程序开头应有以下编译预处理命令：

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

或

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

stdio 是 standard input & output 的缩写。由于 printf 和 scanf 函数使用频繁，Turbo C 2.0 允许(Turbo C++ 3.0、VC++ 6.0 等不允许)在使用这两个函数时可以不加 #include 命令。

3.3.1 格式输出函数——printf 函数

1. printf 函数的一般格式

printf 函数的一般格式如下：

printf(格式控制字符串,输出表列);

例如：

```
printf("Welcome!");
printf("The numbers: %d,%f\n",k,y);
```

说明：

(1) 格式控制字符串必须用双引号括起来。格式控制字符串由 3 类内容组成。

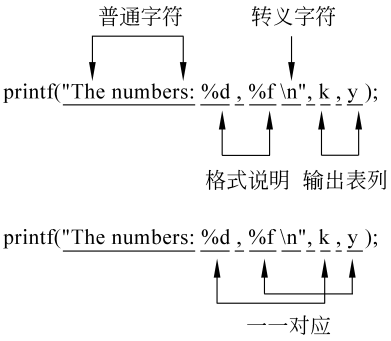
第一类是普通字符。函数将它们原样输出到屏幕上。

第二类是转义字符。在输出时按其含义完成相应的功能。

第三类是格式说明。由“%”和格式字符组成,如%d、%f 等。它指定要输出的数据类型。

(2) 输出表列是需要输出的变量、常量、表达式等,输出表列中参数的个数可以是 0 个或若干个,当超过一个时用逗号分隔。输出表列和格式说明在个数和类型上要一一匹配。

【例 3.2】 对普通字符、转义字符、格式说明以及个数和类型一一匹配关系进行说明。



如果 k=10,y=25.6,则上述语句输出结果为：

```
The numbers:10,15.600000
```

2. printf 函数的格式字符

不同的数据输出时使用不同的格式字符。printf 函数中的格式字符如表 3.1 所示。

表 3.1 printf 函数的格式字符

输出类型	格式字符	说明
整型数据	d,i	输出带符号的十进制整数(正数不输出正号)
	o	输出无符号的八进制整数(不输出前缀 0)
	u	输出无符号的十进制整数
	x,X	输出无符号的十六进制整数(不输出前缀 0x),用 x 输出十六进制整数时的 a～f 以小写形式输出,用 X 时以大写字母形式输出
字符型数据	c	输出一个字符
	s	输出一个字符串
实型数据	f	以小数形式输出单、双精度实数,隐含输出 6 位小数
	e,E	以指数形式输出实数,用 e 时指数以“e”表示(如 1.6e+03),用 E 时指数以“E”表示(如 1.6E+03)
	g,G	自动选用%f 和%e 中输出宽度较短的一种形式输出实数,且不输出无意义的 0。用 G 时,若以指数形式输出,则指数以大写表示

例 3.2 就是应用了格式字符 d 和 f,即格式说明%d 和%f。

对于格式字符,下面再说明几种情况。

(1) 输出实数。

使用格式%f 输出,整数部分全部如数输出,小数部分按 6 位小数输出。但并非全部数字都是有效数字,float 型实数的有效数字是前 7 位,double 型实数的有效数字是前 15 位或 16 位。

【例 3.3】 输出单精度实数,观察有效位数。

```
main( )
{
   float a,b;
   a=111111.111;
   b=333333.333;
   printf("a+b= %f\n",a+b);
}
```

运行结果为:

```
444444.453125
```

很明显,只有前 7 位是有效数字,给出 6 位小数。实数输出时不要认为输出的数字都是准确的。

【例 3.4】 输出双精度实数,观察有效位数。

```
main( )
{
   double a,b;
   a=1111111111111.111111111;
   b=3333333333333.333333333;
   printf("a+b= %f\n",a+b);
}
```

运行结果为:

```
4444444444444.444340
```

很明显,只有前 16 位是有效数字,给出 6 位小数。最后 3 位小数超过了 16 位,无意义。

(2) 输出八进制整数和十六进制整数。

【例 3.5】 已知十进制整数,输出与其对应的八进制整数和十六进制整数。

```
void main( )
{
   int a=180;
   printf("八进制= %o\n",a);
   printf("十六进制= %x\n",a);
   printf("十六进制= %X\n",a);
}
```

运行结果为:

```
八进制= 264
十六进制= b4
十六进制= B6
```

(3) 输出一个字符。

使用格式%c 输出。一个字符既可以作为字符输出,也可以作为整数(该字符对应的

ASCII 码)输出;反之,一个整数,只要它在 0～255 范围内,也可以将该整数作为 ASCII 码的相应字符输出。

【例 3.6】 输出字符。

```
void main( )
{
   char c='M';
   int j=81;
   printf("%c,%d\n",c,c);
   printf("%c,%d \n",j,j);
}
```

运行结果为:

```
M,77
Q,81
```

(4) 输出字符串。

【例 3.7】 使用格式%s 输出字符串。

```
main( )
{
   printf("%s%s\n","CHINA"," IS A GREAT COUNTRY!");
}
```

运行结果为:

```
CHINA IS A GREAT COUNTRY!
```

输出的字符串不包括引号。

3. printf 函数的附加格式说明字符

在格式说明中,%和格式字符之间可以带有附加格式说明字符(又称修饰符),如表 3.2 所示。

表 3.2 printf 函数的附加格式说明字符(修饰符)

字　符	说　明
字母 l	用于长整型整数,可加在类型字符 d、i、o、x、u 前面;用于双精度实数,可加在 f、e、g 前面
m(代表一个正整数)	数据输出的最小宽度,右对齐方式,空出的左端补空格
n(代表一个正整数)	对实数,输出 n 位小数;对字符串,从左端截取 n 个字符输出
—	输出的数字或字符在域内向左靠
#	加在格式字符 o、x 前面,使系统输出八进制的前缀 0 和十六进制的前缀 0x
数字 0	输出数值时在左面不使用的空位置补 0

对于附加格式说明字符(修饰符),重点说明下面几种情况。

(1) %md 表示输出十进制整数的最小宽度为 m 位,即输出字段的宽度至少占 m 列,右对齐,数据少于 m 位则在左端补空格(或 0)使之达到 m 位;数据超过 m 位则 m 不起作用,按数据的实际位数输出。数据前要补 0,则在 m 前面加个 0。

例如:若 k=18,p=31689,则

```
printf("%6d",k);          输出结果为: ____18
printf("%06d",k);         输出结果为: 000018
```

```
printf("%4d",p);                    输出结果为: 31689
```

类似地还有%mc、%mo、%mu、%ms 等。

(2) 字母 l。在输出 d、i、o、u、x 等整型数据时,在前面加上字母 l 表示输出的是一个长整型数据。

【例 3.8】 输出数据时,修饰符——字母 l 的应用。

```
main( )
{
   long  a=245978;
   printf("a=%ld,a=%d\n",a,a);
}
```

运行结果为:

```
a=245978,a=-16166
```

前一个数据是正确的,后一个数据是错误的。因为 245 978 已经超出整型数据的范围-32 767～32 768,必须要用长整型数据格式输出。

(3) %m.nf 表示输出数据为小数形式,m 为总宽度(包括小数点),n 为小数部分的位数。小数长度不够则补 0,小数部分超过 n 位,则 n+1 位向 n 位四舍五入;整个数据小于 m 位左补空格,超过 m 位,则 m 不起作用,按数据的实际位数输出。

例如:若 x=123.45,y=123.456,z=-123.45,则

```
printf("x=%10.4f",x);              输出结果为:x=␣␣123.4500
printf("y=%10.2f",y);              输出结果为:y=␣␣␣␣␣123.46
printf("z=%4.2f",z);               输出结果为:z=-123.45
```

(4) %m.ns 表示输出数据为字符串形式,m 是总宽度,但当实际位数超过时,多余者将被删除,n 表示只取字符串中左端的 n 位,n<m 时,左边补空格;n>m 时,m 自动取 n 值,保证 n 位字段的正常输出。例如:

```
printf("%7.3s", "12345" );         输出结果为:␣␣␣␣123
printf("%5.7s", "12345678");       输出结果为:1234567
```

不过,在程序中用得较多的形式还是%s,使用该格式输出时,按实际内容输出。例如:

```
printf("%s", "12345");             输出结果为 12345
printf("%s", "12345678");          输出结果为 12345678
```

(5) -表示左对齐格式,若没有则为右对齐格式。例如:

```
printf("%-5d", 12);                输出结果为 12␣␣␣
printf("%-5s", "1243");            输出结果为 1243␣
printf("%-7.3s", "12345" );        输出结果为 123␣␣␣␣
printf("%-5.7s", "12345678");      输出结果为 1234567
printf("%-5.7s", "123456");        输出结果为 123456␣
printf("%-10.2f",123.456);         输出结果为 123.46␣␣␣␣
```

4. 使用 printf 函数时的注意事项

(1) 在格式控制字符串中,格式说明与输出表列中的参数个数应该相同。若格式说明的个数少于参数的个数,多余的参数不予输出;若格式说明的个数多于参数的个数,则多余的参数将输出不定值或零值。

(2) 在格式控制字符串中,格式说明与输出表列中的参数从左到右在类型上必须一一匹配。若不匹配将导致数据不能正确输出(输出数据的类型由格式说明确定),这时,系统并

不报错。例如：

【例 3.9】 格式说明与输出表列不匹配的情况。

```
main( )
{
   char c='M';
   int j=81;
   printf("%d\n",c);
   printf("%c \n",j);
}
```

运行结果为：

```
77
Q
```

(3) 在格式控制字符串中，可以包含任意的合法字符(包括转义字符)，这些字符在输出时将原样输出。

(4) 若想输出字符“%”，则需在格式说明中用连续的两个“%”表示，例如：

```
printf("%f%% \n",25.3);
```

输出结果为：

```
25.300000%
```

3.3.2 格式输入函数——scanf 函数

1. scanf 函数的一般格式

scanf 函数的一般格式如下：

```
scanf(格式控制字符串,地址表列);
```

例如：

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

说明：

(1) 格式控制字符串必须用双引号括起来。格式控制字符串由两类不同的内容组成。

第一类是普通字符。输入数据时，必须在对应位置上原样输入。

第二类是格式说明。scanf 函数中的格式说明与 printf 函数类似，由“%”和格式字符组成(也可以在其中间加入修饰符)，如%d 等。它指定要输入的数据的格式。

(2) 地址表列是接收输入数据的变量的存储单元地址，或字符串的首地址。变量的地址由“&”加变量名组成，如上例中的 &a 和 &b，scanf 函数的作用是将从键盘输入的数据存入变量地址中。地址表列中地址的个数超过一个时用逗号分隔。地址表列和格式说明在个数和类型上要一一匹配。

如：

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

若从键盘输入：

```
7,12.345↙
```

则 7 存入变量 a 的地址中(a 得到 7),12.345 存入变量 b 的地址中(b 得到 12.345)。

2. scanf 函数的格式字符和附加格式说明字符

scanf 函数中的格式字符如表 3.3 所示。scanf 函数的附加格式说明字符(修饰符)如表 3.4 所示。

表 3.3　scanf 函数的格式字符

输入类型	格式字符	说　明
整型数据	d,i	输入带符号的十进制整数
	o	输入无符号的八进制整数(可以带前缀 0,也可以不带)
	u	输入无符号的十进制整数
	x,X	输入无符号的十六进制整数(可以带前缀 0x 或 0X,也可以不带),大小写作用相同
字符型数据	c	输入一个字符
	s	输入一个字符串,将字符串送到一个字符数组中。输入时以非空白字符开始,以第一个空白字符结束。字符串以串结束标志'\0'作为其最后一个字符
实型数据	f	以小数形式或指数形式输入实数
	e,E,g,G	与 f 作用相同,e 与 f、g 可以互相替换(大小写作用相同)

表 3.4　scanf 函数的附加格式说明字符(修饰符)

字　符	说　明
字母 l	用于输入长整型数据(可用%ld,%lo,%lx,%lu)以及 double 型数据(可用%lf 或%le)
h	用于输入短整型数据(可用%hd,%ho,%hx)
域宽 m(代表一个正整数)	指定输入数据所占的宽度(列数)
*	表示本输入项在读入后不赋给相应的变量

说明:

(1) 若多个格式控制字符中无分隔符,从键盘输入数据(不是字符型数据)时,可用隐含分隔符隔开。隐含分隔符有空格(一个或多个)、Enter 键、Tab 键。

例如,假设 a、b、c 为整型变量,要使它们分别得到值 30、40、50。使用以下输入语句:

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

键盘输入数据的格式:

① 30␣40␣50↙
② 30↙
　 40␣50↙
③ 30␣40↙
　 50↙
④ 30(按 Tab 键)40↙
　 50↙
⑤ 30↙
　 40↙
　 50↙

注意:其中空格"␣"可以是一个也可以是若干个。

(2) 若在格式控制字符串中有其他字符(如逗号等),则在输入数据时,要在相应位置原样输入这些字符。如:

① scanf("%d,%d",&a,&b);

可从键盘输入:

30,40↙

注意:“,”必须在对应位置上原样输入;若不输入“,”则是错误的。

如输入:

10␣20↙或 10:20↙

均是错误的。

② 若有以下输入语句:

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

在输入时必须为:

a=10,b=20↙

而输入:

10␣20↙或 a=10␣b=20↙

是错误的。

(3) 关于输入字符型数据时,转义字符和隐含分隔符的使用问题。

① 字符和字符之间不能使用转义字符和隐含分隔符。在用“%c”输入多个字符型数据时,数据之间不可以使用转义字符和隐含分隔符(空格、Enter 键、Tab 键),因为系统此时会把转义字符、隐含分隔符作为有效字符型数据送给字符变量。例如:

```
scanf("%c%c",&x,&y);
```

使 x 得到 a,y 得到 b。若输入:

ab↙

则正确。

若输入:

a␣b↙

则不正确。此时,x 得到 a,y 得到␣。

② 数字和字符之间能不能使用转义字符和隐含分隔符,要视情况而定。例如:

```
scanf("%d%c",&x,&y);
```

使 x 得到 68,y 得到 b。若输入:

68b↙

则正确。若输入:

68␣b↙

则不正确。此时,x 得到 68,y 得到␣。

```
scanf("%c%d ",&x,&y);
```

使 x 得到 b,y 得到 68。输入:

b␣68↙或 b68↙

都正确。为保险起见，字符和字符之间、数字和字符之间不要使用转义字符和隐含分隔符。

(4) 如果指定了输入数据的宽度，系统将自动按此宽度截取所需的数据。例如：

```
scanf("%3d%3d%2d",&a,&b,&c);
```

输入：

```
12345678↙
```

则 123 赋给 a，456 赋给 b，78 赋给 c。

但是，在输入实数时，不能规定小数位。例如：

```
scanf("%10.4f",&p);
```

是错误的。

(5) 一个格式说明中出现“*”时，表示读入该类型的数据不赋给某个变量(相当于跳过该数据)。

【例 3.10】 “*”的使用。


视频

```
main( )
{
   int a,b;
   scanf("%3d%*2d%4d",&a,&b);
   printf("a=%d,b=%d\n",a,b);
}
```

输入：

```
987654321↙
```

则变量 a 得到 987，65 跳过，变量 b 得到 4321。运行结果为：

```
a=987, b=4321
```

(6) 输入数据时，遇到下述情况认为该数据结束。

① 遇到空格、Enter 键或 Tab 键。

② 按指定的宽度结束，例如%4d，只取 4 列。

③ 遇到非法输入。例如：

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

若输入：

```
5678p345o.79↙     (o 是字母，不是数字 0)
```

则变量 a 得到 5678，变量 b 得到字符 p，变量 c 得到 345(没有得到应该得到的 3450.79)。字母 o 在这里非法。

(7) 若有确定数据的最大位数的修饰符 m，当输入数据的位数少于 m 时，程序等待输入，直到满足要求或遇到非法字符为止；当输入数据位数多于 m 时，多余数据将作为下一个数据读入其他变量。例如：

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

输入：

12345↙　　(输入数据的位数少于 3+3 位)

则变量 a 得到 123,变量 b 得到 45(遇 Enter 键)。若输入

1␣2345↙

则变量 a 得到 1(遇空格)而不是 12、102 或 123,变量 b 得到 234。例如:

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

输入:

1234␣6789↙　　(输入数据 1234 的位数 4 多于 a 要求的 3 位)

则变量 a 得到 123,后面的 4 读入 a 的后一变量 b(b 读入 4 后遇空格结束),变量 b 得到 4,变量 c 得到 678。

(8) 当一行内输入的数据个数小于格式控制字符串的个数时,光标会出现在下一行,等待用户继续输入。C 语言允许在一行内输入的数据个数大于格式控制字符串的个数,这时会把多余的数据留给后面的 scanf 函数使用。

(9) 输入长整型数据和 double 型数据时,在%和格式字符之间加 l,如%ld、%lf 等。

3. 使用 scanf 函数时的注意事项

(1) scanf 函数中地址表列使用的是变量的地址而不是变量名。例如 scanf("%d%d",x,y);是错误的。

(2) scanf 函数没有计算功能,输入的数据只能是常量,而不能是表达式。

(3) 在格式控制字符串中不要使用转义字符,如\n。

(4) 格式说明和地址表列在类型上要一一匹配。若不匹配,系统并不报错,但不可能得到正确的结果。

(5) 格式说明和地址表列在个数上要相同。若格式说明的个数少于地址表列的个数,则 scanf 函数结束输入,多余的地址表列并没得到新数据;若格式说明的个数多于地址表列的个数,则 scanf 函数等待输入。

3.3.3 字符输出函数——putchar 函数

putchar 函数的作用是向屏幕(显示器)输出一个字符。

putchar 函数的一般调用格式如下:

```
putchar(c);
```

它将 c 输出到屏幕上。c 既可以是常量,也可以是字符变量、整型变量或表达式。

【例 3.11】 字符数据的输出。

```
# include <stdio.h>
main( )
{
   char a='y', b='e', c='s';   /*字符变量*/
   putchar(a); putchar(b); putchar(c);
   putchar('\n');              /*转义字符,换行*/

   putchar('P');               /*字符常量*/
   putchar(81);                /*整型常量,ASCII 码值为 81 的字符 Q */
   putchar('\333');            /*ASCII 码用八进制表示为 333(十进制表示为 219)的字符■*/
```

```
    putchar('K'+2);              /* 表达式,即输出字符 M */
}
```

运行结果为：

```
yes
PQ■M
```

3.3.4 字符输入函数——getchar 函数

getchar 函数的作用是接收从键盘输入的一个字符。当程序执行到 getchar 函数时，将等待用户从键盘输入一个字符。

其一般调用格式如下：

```
getchar();
```

注意：括号( )中间无参数。getchar 函数只接收一个字符。

【例 3.12】 从键盘输入一个字符，把它存入字符型变量 x 中，并输出。

```
# include <stdio.h>
main( )
{
   char x;
   x=getchar( );
   putchar(x);                   /* 用 putchar 函数输出 */
}
```

输入：

```
y↙
```

运行结果为：

```
y
```

说明：getchar 函数得到的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，而作为表达式的一部分。

【例 3.13】 从键盘输入一个字符，不赋给任何变量。

```
# include <stdio.h>
main( )
{
   putchar(getchar( ));
}
```

输入：

```
9↙
```

运行结果为：

```
9
```

在这里，9 是一个字符。

语句 putchar(getchar( ))；用 printf("%c", getchar( ))；代替也可以。

3.4 C 语句概述

C 语言的语句分为 5 类，分别为复合语句、空语句、表达式语句、控制语句和函数调用语句。

下面重点介绍复合语句、空语句和表达式语句，简单介绍控制语句和函数调用语句。控制语句和函数调用语句在后续章节再重点学习。

3.4.1 复合语句

C 语言把由一对花括号({ })括起来的一组语句称为复合语句。

例如：

```
{
   t=a;
   a=b;
   b=t;
}
```

一个复合语句在语法上可以看作一个整体。在程序中，凡是单个语句能够出现的地方复合语句都可以出现。

复合语句作为一个整体又可以出现在其他复合语句的内部，并且复合语句中也可以定义变量，但所定义的变量仅在本复合语句中有效。

例如：

```
main( )
{  char c;
   …
   {  double x;
      …
      {  double  y;
         scanf("%lf",&y);
         printf("x+y=%lf",x+y);
      }
    …
   }
   …
}
```

3.4.2 空语句

在程序设计中，特别是在复杂程序的调试过程中，往往需要加一个或若干个空语句来表示存在某个或某些语句。

所谓空语句，就是一个不含其他内容而仅含分号的语句。

例如：

```
main( )
{
   …
   ;                              /*空语句 */
   …
}
```

注意：对空语句的使用一定要慎重，因为随意加分号极易导致逻辑上的错误。

3.4.3 表达式语句

任何一个表达式的后面加上一个分号就构成了表达式语句。例如：

```
x+y
```

是表达式，而

```
x+y;
```

是表达式语句。

最典型的是由一个赋值表达式加分号构成一个赋值语句。例如：

```
x=80
```

是赋值表达式，而

```
x=80;
```

是赋值语句。

3.4.4 控制语句

用来实现一定的控制功能的语句称为控制语句。C 语言中有 9 种控制语句：

(1) if…else…　条件语句。
(2) switch　多分支选择语句。
(3) while…　循环语句。
(4) do…while　循环语句。
(5) for…　循环语句。
(6) break　中止执行 switch 或循环语句。
(7) goto　转向语句。
(8) continue　结束本次循环语句。
(9) return　函数返回语句。

3.4.5 函数调用语句

由函数调用加分号构成了函数调用语句。例如：

```
printf("请输入 a,b 的值:");
add(a,b);
```

是两个函数调用语句(函数调用语句将在第 7 章中大量使用)。

科学素质拓展——安常处顺，顺应自然，随遇而安，不负韶华。

《庄子.养生主》："适来，夫子时也；适去，夫子顺也。安时而处顺，哀乐不能入也。"意思是一个人偶然来到世间，这是他顺时而生；偶然离去了，这是他顺时而死。安于时运而顺应自然，一切哀乐之情就不能进入心怀。后以"安常处顺"指习惯于正常生活，处于顺利境遇中。

"顺应自然，随遇而安"是中国传统文化中的一种重要的生活哲学观念。它强调人们应该顺应自然，不去过分计较，能够安然自在地应对生活的各种变化。从儒家、道家和马克思主义哲学等不同的角度来解读"随遇而安"，可以更深刻地理解其内涵。

在儒家看来，"顺应自然，随遇而安"需要人们根据自己的生活情境和社会环境，选择适合自己的生活方式，同时也要认真对待自己的人生，积极面对生活中的各种挑战。

在道家看来，"顺应自然，随遇而安"需要人们通过修炼自己的内心，放下执着和烦恼，平静地面对生活中的各种变化和困难。同时，也需要人们以宽广的心态来看待生活，顺应自然，不为外物所动，始终保持内心的平静与安宁。

在马克思主义哲学看来，"顺应自然，随遇而安"需要人们以开放的心态面对生活，接纳事物的发展和环境的变化，不断地修正自己的观念和行为方式，实现自身的成长和进化，从困境和挫折中寻找启示和智慧，从而更好地理解万事万物的内在规律，遵循事物发展的规律，到相应的时间就做相应的事情。比如小时候是读书识字的年龄，就要认真读书，错过识字的年龄，以后难以弥补，正所谓少壮不努力，老大徒伤悲。

希望我们每个人就如丰子恺在《不宠无惊过一生》中写道的："不乱于心，不困于情。不畏将来，不念过往。如此，安好。"以梦为马，享受生命馈赠，不负韶华。

3.5 应用举例

【例 3.14】 编写一个程序，分别输入一个八进制、十进制和十六进制数，将这 3 个数相加，以十进制的形式输出。

分析：

(1) 首先定义变量，用来保存输入的数据及运算结果。

(2) 输入八进制、十进制和十六进制数。

(3) 对数据求和。

(4) 输出结果(数据输出)。

程序如下：

```
main( )
{
   int a,b,c,d;
   printf("请输入八进制、十进制、十六进制数据: \n");
   scanf("%o%d%x",&a,&b,&c);
   d=a+b+c;
   printf("输入的三个数据为: %o  %d  %x \n",a,b,c);
   printf("对应的十进制数为:%d  %d  %d \n",a,b,c);      /*以十进制输出这 3 个数*/
   printf("sum=%d \n",d);
}
```

程序执行：

```
请输入八进制、十进制、十六进制数据：
12 ␣34 ␣7a ↙                                    //键盘输入数据
输入的三个数据为：12  34  7a
对应的十进制数为：10  34  122
sum=164
```

【例 3.15】 从键盘输入一个大写字母，把它转换为小写字母后输出。

分析：

(1) 数据输入、输出类似于例 3.14。

(2) 本题的关键是清楚字母大小写的转换规则。查附录 A 可知，小写字母在大写字母之后，且同一个字母的 ASCII 码值小写比大写大 32，若大写字母用变量 c1 表示，小写字母用变量 c2 表示，则 c2＝c1＋32，这是一种方式；另一种方式是 c2＝c1 ＋'a' －'A'，请考虑这是为什么？

程序如下：

```
main( )
{
   char c1,c2;
   scanf("%c",&c1);
   c2=c1+'a'-'A';              /* 'a'-'A'为同一个小写字母和大写字母的 ASCII 码之差 */
   printf("Upper=%c,Lower=%c",c1,c2);
}
```

运行结果为：

```
D ↙
Upper=D,Lower=d
```

请思考：如果从键盘输入一小写字母，把它转换为大写字母后输出，程序如何编写？

【例 3.16】 输入三角形的三条边长，求三角形的面积(假设输入的三条边能构成三角形)。

分析：

(1) 首先要明确三角形面积的计算公式。从数学中知道，假设三条边用 a、b、c 表示，三角形面积的计算公式如下：

$$s=(a+b+c)/2$$

$$area=\sqrt{s(s-a)(s-b)(s-c)}$$

(2) 为 a、b、c 输入值(假设输入的三条边能构成三角形)。

(3) C 语言开平方可以使用系统提供的数学函数 sqrt。

程序如下：

```
# include <stdio.h>
# include <math.h>
void  main( )
{
   float a,b,c,s,area;
   scanf("%f,%f,%f",&a,&b,&c);
   s=(a+b+c)/2;
   area=sqrt(s*(s-a)*(s-b)*(s-c));
   printf("a= %6.2f,b=%6.2f,c=%6.2f\n",a,b,c);
   printf("area=%6.2f\n",area);
```

```
}
```

运行结果为：

```
3,4,6↙                                          //假如输入的数据为 3、4、6
a=  3.00, b=  4.00, c=  6.00
area=  5.33
```

sqrt 函数的原型在 math.h 头文件中。因此必须在程序头部使用文件包含命令 #include "math.h"或#include <math.h>。

请思考：area=sqrt(s*(s-a)*(s-b)*(s-c));可以写成 area=sqrt(s(s-a)(s-b)(s-c));吗？

习 题 3

一、选择题

1. 下述哪一个不是结构化程序基本结构？（　　）

A. 顺序　　B. 选择　　C. 循环　　D. 嵌套

2. 组成 C 语句的一个必不可少的符号是（　　）。

A. 逗号　　B. 引号　　C. 冒号　　D. 分号

3. 下述是 C 语言中有关变量定义的几个说法，正确的是（　　）。

A. 变量可以不定义直接使用

B. 一个说明语句只能定义一个变量

C. 几个不同类型的变量可在同一语句中定义

D. 变量可以在定义时进行初始化

4. 以下所列语句中，合法的语句是（　　）。

A. a=1,b=2　　B. ++a;　　C. a=a+1=5　　D. y=int(a);

5. 与 x * = y + z 等价的赋值表达式是（　　）。

A. x = y + z　　B. x = x * y + z

B. x = x * (y + z)　　D. x = x + y * z

6. 使用语句 scanf("x=%f,y=%f",&x,&y);输入变量 x、y 的值(□代表空格)，正确的输入是（　　）。

A. 1.25,2.4　　B. 1.25□2.4

C. x=1.25,y=2.4　　D. x=1.25□y=2.4

7. 已知字母 A 的 ASCII 码是 65，以下程序的执行结果是（　　）。

```
#include<stdio.h>
main()
{
  char c1='A',C2='Y';
  printf("%d,%d\n",c1,c2);
}
```

A. A,Y　　B. 65,65　　C. 65,90　　D. 65,89

二、给出运行结果题

1. 给出下列程序段的输出结果。

```
(1) int a=1, b=2;
    printf("a=%d%%,b=%d\n",a,b);
(2) int a=-1;
    printf("%d,%u",a,a);
(3) int x=10;
    printf("x=%d,x=%#x\n",x,x);
(4) int d=389;
    printf("*%-07d*\n",d);
(5) double y=123.456789;
    printf("y=%8.6f,y=%8.2f,y=%14.8f,y=%14.8lf\n",y,y,y,y);
```

2. 分析下列程序，并给出运行结果。

```
(1) main( )
    {
      int a,b;
      float f;
      scanf("%d,%d",&a,&b);                    /*输入 1 和 2*/
      f=a/b;
      printf("F=%f",f);
    }
(2) main()
    {
      int sum,pad;
      sum=pad=5;
      pad=sum++;
      pad++;
      ++pad;
      printf("%d\n",pad);
    }
```

三、改错题

指出下面程序中的错误并改正。

```
main( )
{
   float b;
   char c;
   int d;
   scanf("%c%d%c%f",a,b,c,d);
   printf("a+b+c+d=%f\n", a+b+c+d);
}
```

四、编程题

1. 编写程序，输入实数 a 和 b 的值，求其和并输出。

2. 编写程序，将输入的小写字母转换为大写字母后输出。

3. 编写程序，输入某个学生 3 门功课的成绩，求出它们的和及平均成绩。要求平均成绩保留小数点后 1 位。

4. 输入圆的半径，求圆的面积和周长。

5. 输入两个正整数，求它们相除所得的商，商的整数部分、小数部分及余数。例如 21 除以 2，其商为 10.5，商的整数部分为 10，小数部分为 0.5，余数为 1。

第4章

选择结构程序设计

从选择的权利谈起——

人类无时无刻不在做出选择，就像当下这一刻，你可以选择认真学习，也可以选择游玩潇洒。但是，无论你做出什么选择，你都需要为这个选择而造成的结果负责。今天的选择是因，才造成了明天的结果。比如，你选择认真听讲，好好学习，才有以后取得好成绩的果；同样，如果你游玩偷懒，以后就可能不及格甚至无法毕业。有时候，即使是父母逼迫你做出某个选择，比如你想学文科，可是父母逼迫你学理科，其结果也是你在承受，因为这是你的人生。

俗话说："所有偷过的懒，都会变成打脸的巴掌。"如果不想挨巴掌，就要学会如何做出更好的选择并为自己的人生负责任。

第3章讲解了顺序结构的程序，只能按照语句先后顺序处理问题，虽然能解决计算、输出等问题，但不能做判断再选择。在很多情况下，需要根据不同的前提条件做出选择，即从给定的两种或多种操作选择其一，这就要求程序本身具有完备的条件判断和选择能力。从流程图角度看，选择结构是一种双分支或多分支结构，从主观角度看，它是一种能执行选择权利的结构。因此选择结构也称分支结构。

在程序设计中，选择结构根据给定的条件判断选择哪一条分支，从而执行相应的语句。在使用选择结构时，关键在于进行条件判断，要用条件表达式来描述条件，条件表达式可以是关系运算表达式或者逻辑运算表达式。本章介绍选择结构程序设计，主要内容包括关系运算、逻辑运算、if语句、switch语句等。

4.1 关系运算符和关系表达式

关系运算也叫比较运算。在程序中经常用来比较两个量的大小关系，以决定程序下一步的走向。比较两个量的运算符称为关系运算符。

4.1.1 关系运算符及其优先顺序

C语言中的关系运算符共有6种，它们是：