第5章　顺　　序

本章首先对C语言的语句进行介绍，而后详细地讨论C语言中字符输入输出函数和标准输入输出函数的使用方法和规则，最后通过若干顺序结构程序的例子来加深理解。

本章重点

- 字符输入输出函数。
- 标准输入输出函数。
- 输入输出格式控制。

本章难点

- 标准输出函数中的格式控制符。
- 数据输入时的格式控制。
- 输入数据的识别。

5.1　顺序结构

1. 语句

C语言程序是以函数为基本单位的，函数是由一个一个的C语句构成的。C语句必须以分号结束。

C语言的语句主要分为以下几类：

(1) 说明语句。

说明语句一般指用来定义变量数据类型等的语句。例如：

```
int a=5,b;
float f1,f2;
```

(2) 表达式语句。

表达式语句是指由一个C语言表达式加上分号构成的语句。例如：

```
a=b+1;          /*赋值表达式 a=b+1 加上分号*/
1;              /*算术表达式 1 加上分号,此语句无实际意义*/
2+2;            /*算术表达式 2+2 加上分号,此语句无实际意义*/
x>y?x:y;        /*条件表达式 x>y?x:y 加上分号,此语句无实际意义*/
i=1,j=2;        /*逗号表达式 i=1,j=2 加上分号*/
i++;            /*表达式 i++加上分号*/
```

(3) 函数调用语句。

函数调用语句是由一个函数调用加上分号构成的。例如：

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

```
srand(time(NULL));
cos(6);          /* 无实际意义的语句 */
```

这类语句也可以归属于表达式语句，因为函数调用本身也是一个表达式。

(4) 空语句。

空语句是仅由一个分号所构成的语句，没有任何动作，即

```
;
```

(5) 复合语句。

复合语句是指将一组语句用大括号“{}”括起来，从而使大括号中的语句变成一个整体(复合语句)。从整体上看，复合语句是一个语句。例如：

```
{
    a=5;
    b=6;
    c=7;
}
```

复合语句在 C 语言程序中的用处很大，在以后的学习中大家会逐渐体会到。有的书中也将复合语句称为分程序或语句块。

(6) 控制语句。

控制语句完成一定的控制功能，实现程序流程的跳转。C 语言提供的控制语句有

goto	无条件转向语句
if-else	选择语句
switch	多分支选择语句
while	循环语句
for	循环语句
do-while	循环语句
break	循环控制语句
continue	循环控制语句
return	从函数返回语句

控制语句在本书的后续章节中会一一介绍。

2. C 程序的书写格式

在 C 语言程序中，允许一行写多个语句，也可以将一个语句写在多行上，书写格式无固定的要求。语句中关键字、标识符、运算符、运算量之间可以用任意个空白字符分隔。这里的空白字符指空格、回车键或制表符(Tab 键)，C 语言程序在编译时会自动略过这些多余的空白字符。

程序清单 05-01-01.c

```
//自由的书写格式(此程序实际代码为 03-07-04.c)
#include <stdlib.h>
#include <time.h>
int main()  {int a,b,c;  srand(time(NULL));
    a=rand()%    100+1;  b=rand()%
```

```
  100+1;
printf("%d+%d="                    ,
                a,
b);
scanf("%d",              &c);
printf(c==a+b?"GOOD!":
"SORRY!"); }
```

程序分析：

此格式书写的程序一样可以通过编译得到正确的运行结果。但是，按照此格式书写的程序阅读起来是很困难的，尤其是在编写规模稍大一些的程序时。所以在编程时尽量养成良好的书写格式习惯。

3. 顺序结构示例

顺序结构是最简单的程序结构，也是最常见的程序结构。顺序结构程序自上而下依次执行。前面章节中的大多数程序是顺序结构的。

程序清单 05-01-02.c

```
//顺序结构程序举例。请运行程序并分析程序功能
#include <stdlib.h>
#include <time.h>
int main(){
    int a,b,c,d,r;
    srand((unsigned)time(NULL));
    printf("*********两位数四则运算测试*********\n");
    printf("请输入以下算式的结果:\n");
    a=rand()%89+11;
    b=rand()%89+11;
    c=rand()%4+1;
    (c==1)?(printf("%d+%d=",a,b),d=a+b):1;
    (c==2)?(printf("%d-%d=",a,b),d=a-b):1;
    (c==3)?(printf("%d*%d=",a,b),d=a*b):1;
    (c==4)?(printf("%d/%d=",a,b),d=a/b):1;
    scanf("%d",&r);
    printf(d==r?"答对了,你真棒!":"真对不起,你答错了!");
}
```

程序分析：

这是顺序结构的程序，程序从第一条语句开始，依次向下一条一条地执行，直到最后一条语句。

4. 基本输入输出功能的实现

C 语言数据的输入输出都是通过函数调用来实现的。

C 语言提供了标准的函数库，其中包括标准输入函数 scanf 和标准输出函数 printf。关

于这两个函数，在前面的章节中已经多次使用过。

C 语言还提供了字符输入函数 getchar、getch、getche 和字符输出函数 putchar 等，C 语言把这些函数的定义放在头文件 stdio.h 中，所以在使用这些函数的 C 程序中，应该在程序的开头处加上以下编译预处理命令：

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

或者

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

程序清单 05-01-03.c

```
//标准输入输出函数举例
#include <stdio.h>
int main(){
    int a,b,c,d,r;
    scanf("%d%d",&a,&b);
    printf("你输入的两个数是%d 和%d.\n",a,b);
    printf("其中较大的是%d,较小的是%d.\n", a>b?a:b, a>b?b:a);
}
```

执行程序，输入

```
5 8
```

输出

```
你输入的两个数是 5 和 8.
其中较大的是 8,较小的是 5.
```

5.2 字符输入输出

1. 字符输出函数 putchar

C 程序中调用字符输出函数 putchar 的一般形式是

```
putchar(表达式)
```

功能说明：

(1) 参数表达式可以为字符表达式或整型表达式。表达式的值为将要输出字符的 ASCII 码，或者是字符本身。

(2) 函数的功能就是在标准输出设备(显示器)上输出一个字符。

(3) 如果表达式的值不是整型值，则自动舍弃小数部分取整。

(4) C 语言中 ASCII 码共有 256 个，详见附录 A，所以函数 putchar 的参数值应该是 0～255，如果超过 255，则系统会自动除以 256 取余数，使之回到 0～255 的范围。

(5) 有些控制字符是不可显示的。例如，ASCII 码为 7 的字符是使计算机的扬声器响一声。

程序清单 05-02-01.c

```
//字符输出函数应用举例
#include<stdio.h>
int main(){
    char c='A';
    int  n=65;
    //以下的6个语句输出AAABBB
    putchar('A');    putchar(c);      putchar(n);
    putchar('A'+1);  putchar(c+1);    putchar(n+1);
    putchar(10);
    //以下的5个语句输出AAAAA
    putchar(65);     putchar('\101'); putchar('\x41');
    putchar(0101);   putchar(0x41);
    putchar(10);
    //以下的两个语句输出AA
    putchar(65.6);   putchar(8*8.2);
    putchar(10);
    //以下的一条语句使扬声器响一声,无可见字符输出
    putchar(7);
}
```

执行程序,输出

```
AAABBB
AAAAA
AA
```

程序分析:

程序执行的同时会听到计算机扬声器响一声。

请大家仔细对照程序中的语句和输出结果,就会发现哪个字符是由哪个语句执行而得来的。请注意程序中 putchar 函数参数的多种形式。

练习 05-02-01 请使用 putchar 函数输出你想输出的内容。

2. 字符输入函数 getchar

C 程序中调用字符输入函数 getchar 的一般形式是

```
getchar()
```

功能说明:

(1) 函数 getchar 没有参数,其功能是从标准输入设备(键盘)上读入一个字符。程序执行到该函数时暂停,光标在屏幕上闪烁,等待用户输入字符,用户输入的字符会依次显示在屏幕上。当用户输入一串字符(可能 1 个,可能多个,也可能 0 个)并按回车键时,输入完成。函数 getchar 的返回值为用户所输入的第一个字符。

(2) 如果用户在输入字符时直接按回车键,那么函数 getchar 的值为回车字符。

(3) 函数 getchar 的值为其所获得的字符,为了使用这个值,一般应该对 getchar 进行处

理(参加运算或赋值),例如:

```
c=getchar();
c=getchar()+1;
putchar(getchar());
```

(4) 单独使用 getchar,能使程序暂停,待用户按回车键时继续,但是用户输入的字符没有被使用。

程序清单 05-02-02.c

```
//字符输入函数应用举例
#include<stdio.h>
int main(){
    char c;
    c=getchar();              /*输入字符直到按回车键结束,接收第一个字符*/
    putchar(c);               /*显示输入的第一个字符*/
}
```

执行程序,输入

```
ABCD   EFG<回车>
```

输出

```
A
```

程序分析:

用户从键盘输入的字符暂时存放在键盘缓冲区中,当用户按回车键时,才会将所有已输入的字符送给程序处理。

getchar 函数依次接收键盘缓冲区中的字符,本例接收到的是第一个字符 A。

程序清单 05-02-02-A.c

```
//字符输入函数应用举例
#include<stdio.h>
int main(){
    putchar(getchar());
}
```

程序分析:

此程序功能与上例程序相同。

程序清单 05-02-03.c

```
//字符输入函数应用举例
#include<stdio.h>
int main(){
    char c;
    c=getchar();
    printf("你输入的字符是:");
    putchar(c);
```

```
    printf(",它的 ASCII 码是:%d.",c);
}
```

执行程序,输入

```
ABCD   EFG<回车>
```

输出

```
你输入的字符是:A,它的 ASCII 码是:65.
```

再次执行程序,输入

```
<空格>ABCD<回车>
```

输出

```
你输入的字符是: ,它的 ASCII 码是:32.
```

程序清单 05-02-04.c

```
//字符输入函数应用举例
#include<stdio.h>
int main(){
    char c1,c2;
    c1=getchar();
    c2=getchar();
    printf("你输入的第一个字符是:%c,其 ASCII 码为:%d.\n",c1,c1);
    printf("你输入的第二个字符是:%c,其 ASCII 码为:%d."  ,c2,c2);
}
```

执行程序,输入

```
ABCD   EFG<回车>
```

输出

```
你输入的第一个字符是:A,其 ASCII 码为:65.
你输入的第二个字符是:B,其 ASCII 码为:66.
```

再次执行程序,输入

```
A<回车>
```

输出

```
你输入的第一个字符是:A,其 ASCII 码为:65.
你输入的第二个字符是:
,其 ASCII 码为:10.
```

程序分析:

getchar 函数从键盘缓冲区依次读取字符,不会忽略空格和回车,也就是说空格、制表符、回车键都会被当作一个字符读入。

程序清单 05-02-05.c

```
//编程输入一个字符,输出它是否为英文字母
int main(){
    char c;
    c=getchar();
    printf("你输入的字符是:%c\n",c);
    printf(c>='A'&&c<='Z'||c>='a'&&c<='z'?
    "它是一个英文字母!":"它不是英文字母!");
}
```

执行程序,输入

```
ABCD
```

输出

```
你输入的字符是:A
它是一个英文字母!
```

再次执行程序,输入

```
123
```

输出

```
你输入的字符是:1
它不是英文字母!
```

练习 05-02-01 编程输入一个字符,输出它是否为数字字符。

练习 05-02-02 编程输入一个字符,若为小写字母则输出其大写形式,若为其他字符则原样输出。

3. 字符输入函数 getche 和 getch

C 语言还提供了两个字符输入函数 getche 和 getch。调用它们的一般形式是

```
getche()
getch()
```

功能说明:

这两个函数的调用形式与 getchar 完全相同,功能也相同,都是从标准输入设备(键盘)中读入一个字符。区别主要在于下面的两点:

(1) 这两个函数在执行时,只要用户输入了一个字符(不需按回车键),函数就立即获得返回值。这样就不会发生执行 getchar 函数时输入多个字符(没按回车键)时,程序并不向下运行,只有按回车键才真正读入数据的情况。

(2) 函数 getche 在执行时,用户输入的字符会显示在屏幕上(回显)。而函数 getch 在执行时,用户输入的字符不在屏幕上显示(不回显)。

程序清单 05-02-06.c

```
//字符输入函数应用举例
#include<stdio.h>
int main(){
    char ch;
    ch=getche();          //输入的字符回显,不用按回车键
    printf("你输入的字符是:%c",ch);
}
```

执行程序,输入一个字符 y(输入 y 后,程序立即向下运行),输出

```
你输入的字符是:y
```

程序清单 05-02-07.c

```
//字符输入函数应用举例
#include<stdio.h>
int main(){
    char ch;
    ch=getch();          //输入的字符不回显,不用按回车键
    printf("你输入的字符是:%c",ch);
}
```

执行程序,输入一个字符 y(输入的字符 y 并不显示,程序立即向下运行),输出

```
你输入的字符是:y
```

程序分析:

请在计算机上运行以上程序,体会 3 个字符输入函数的区别。函数 getch 在执行时用户输入的字符不回显,且不用按回车键,也就是说实现了按任意键继续的功能。所以常常把这个函数单独使用,放在一个程序的末尾,以使程序结束后能暂停。只有当用户按下任意一个键后,程序才真正结束。

程序清单 05-02-08.c

```
//字符输入函数 getch 应用举例
#include<stdio.h>
int main(){
    printf("程序开始 ...\n");
    printf("请按任意键结束程序 ...");
    getch();              //等待输入,按任意键后结束
}
```

5.3 标准输入输出函数

1. 标准输出函数 printf

标准输出函数 printf 一般用于向标准输出设备(显示器)按规定的格式输出信息。

调用 printf 函数的一般形式为

printf(格式控制字符串,输出值参数列表);

功能说明：

(1) 格式控制字符串是一串用双引号括起来的字符,其中包括普通字符和格式说明符。格式说明符是以%开头,后跟一个或几个规定字符。一个格式说明符用来代表一个输出的数据,并规定了该数据的输出格式。

例如在语句 printf("a=%d,b=%d",a,b);中,字符串"a=%d,b=%d"是格式控制字符串,a、b 是输出值参数列表。在这个格式控制字符串中,a=、逗号(,)、b=是普通字符,而%d 就是格式说明符。

(2) 格式控制字符串中的普通字符要按原样输出。

(3) 一个格式说明符用来代表一个输出的数据,其所代表的数据在输出值参数列表中。输出值参数列表是一系列用逗号分开的表达式,表达式的个数和顺序与前面的格式说明符要一致。

例如,有整型变量 a 和 b,它们的值分别是 3 和 8,则执行以下语句：

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

输出结果是

```
a=3,b=8
```

2. 格式说明符

不同类型的数据在输出时应该使用不同的格式说明符。C 语言提供的格式说明符及其含义如表 5-1 所示。

表 5-1　格式说明符

格式说明符	代表的数据类型及输出形式
%d	int 型;十进制有符号整数,正号省略
%ld	long 型;十进制有符号长整数,正号省略
%u	int 型;十进制无符号整数
%o	int 型;八进制无符号整数,不输出前导 0
%x,%X	int 型;十六进制无符号整数,不输出前导 0x
%#o,%#x,%#X	八进制和十六进制整数输出时加上前导 0、0x(或 0X)
%c	int 型(char 型);一个字符
%f,%lf	double 型;十进制小数,默认小数位数为 6 位
%e	double 型;以指数形式输出浮点数
%g	double 型;自动在%f 和%e 之间选择输出宽度小的表示法