第 3 章　顺序结构程序设计

【本章概要】

在 C 语言中，程序结构一般分为顺序结构、选择结构、循环结构。本章主要介绍的是：C 语言语句、顺序程序结构、基本的输入输出函数等。

顺序结构是最简单的也是最基本的程序结构，其特点是语句按书写的顺序依次执行。顺序结构主要由简单语句、复合语句及输入输出函数语句组成。printf 与 putchar 是输出函数，scanf 与 getchar 是输入函数，它们既有相同点也有不同点。

3.1　C 语言的基本语句

在进行程序设计时，有两部分工作，一部分是数据的设计，另一部分是操作的设计。数据设计的结果是一系列的数据描述语句，主要用来定义数据的类型，完成数据的初始化等；操作设计的结果是一系列的操作控制语句，其作用是向计算机系统发出操作指令，以完成对数据的加工和流程控制。

在 C 语言中，无论是运算操作还是流程控制，都是由相应的语句完成的。C 语言的语句用来向计算机系统发出操作指令，一个语句经过编译后产生若干条机器指令。

C 语言的语句可分为以下 5 种类型：表达式语句、函数调用语句、控制语句、复合语句和空语句。

1. 表达式语句

由表达式组成的语句称为表达式语句，其作用是计算表达式的值或改变变量的值。它的一般形式是：

表达式；

即在表达式的末尾加上分号，就变成了表达式语句。最典型的例子是，由赋值表达式构成赋值语句。如：

```
x=5 是赋值表达式，
x=5; 是一个赋值语句。
```

【注意】 分号是 C 语言中语句的标志，一个语句必须要有分号，没有分号，则一定不是语句。表达式能构成语句是 C 语言的一个重要特色。

2. 函数调用语句

由一个函数调用加上一个分号构成函数调用语句，其作用是完成特定的功能。它的

一般形式是：

函数名(参数列表);

例如：

```
printf("This is a C statement");                    /*调用库函数,输出字符串*/
```

3. 控制语句

控制语句用于完成一定的控制功能，以实现程序的各种结构方式。C语言有9种控制语句，可分为如下三类。

(1) 条件判断语句：if语句、switch语句。

(2) 转向语句：break语句、continue语句、goto语句、return语句。

(3) 循环语句：for语句、while语句、do-while语句。

4. 复合语句

复合语句是用花括号将若干语句组合在一起的，又称分程序，在语法上相当于一条语句。例如下面是一个复合语句：

```
{ i++;
  printf("%d\n",i);
}
```

【注意】 复合语句中最后一个语句的分号不能省略不写；复合语句的大括号后面没有分号。

5. 空语句

只有一个分号的语句称为空语句。它的一般形式是：

```
;
```

空语句是什么也不执行的语句，常用于循环语句中的循环体，表示循环体什么都不做。例如：

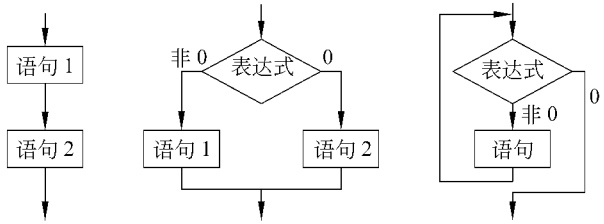
```
while (getchar()!='\n')
;                                                  /*空语句*/
```

该循环语句的功能是：直到从键盘上按回车才退出循环。这里的循环体是空语句。

3.2 顺序结构

在C语言中，程序结构一般分为顺序结构、选择结构、循环结构，图3-1所示是这三种基本结构的流程图。任何复杂的程序都是由这三种基本结构组成的，这也是结构化程序

设计的基本思想之一。



(a) 顺序结构　(b) 选择结构　(c) 循环结构

图 3-1　C 程序的三种基本结构

顺序结构，按语句出现的先后顺序依次执行的程序结构。

选择结构，又称分支结构，根据给定的条件是否成立，以便决定程序转向的程序结构，详见第 4 章。

循环结构，又称重复结构，即反复执行某一部分语句的程序结构，详见第 5 章。

下面简单介绍顺序结构的相关内容。

顺序结构是程序设计中最简单、最基本的结构，其特点是程序运行时，按语句书写的次序依次执行。顺序结构一般由函数调用语句、说明语句、表达式语句和输入输出语句组成。

【例 3-1】 分析下面程序结构。

```
/* EX3-1.C */
#include <stdio.h>
main()
{
  int a,b,c;
  a=123;b=456;
  c=a+b;
  printf("\nc=%d\n",c);
}
```

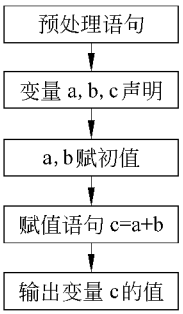


图 3-2　例 3-1 的流程图

上述程序显然是顺序结构，其语句执行的次序如图 3-2 所示。

从例 3-1 可以看出，顺序结构的程序框架如下：

```
#开头的编译预处理命令行
main()
{
  局部变量声明语句;
  可执行语句序列;
}
```

3.3 数据输入与输出

C语言的输入和输出功能是由标准输入输出库函数来实现，这使得C语言编译系统简单。因为没有输入输出语句，就可以避免在编译阶段处理与硬件有关的问题，使编译系统简化，通用性强，可移植性好。C语言提供的函数以库文件的形式存放在系统中，它们不是C语言文本的组成部分。在使用函数库时，要用预编译命令“#include”将有关的“头文件”包含到用户源文件中，#include命令一般放在程序的开头。

stdio.h是standard input & output的缩写，是标准的输入输出函数库头文件，包括putchar(输出字符)、getchar(输入字符)、printf(格式输出)、scanf(格式输入)、puts(输出字符串)、gets(输入字符串)等函数。考虑到printf、scanf使用频繁，系统允许在使用这两个函数时可省略#include命令，而使用其他函数必须加#include命令。

3.3.1 格式化输出函数printf

printf函数的功能是向标准输出设备输出若干个任意类型的数据。

1. printf函数调用形式

printf函数是一个标准库函数，其调用的一般形式为：

```
printf(格式控制字符串,输出列表);
```

括号里格式控制字符串和输出列表实际上都是函数的参数。其中：

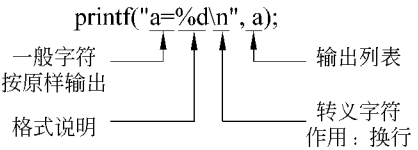
(1) 格式控制字符串是用双引号括起来的字符串，它包括如下两个信息。

① 格式说明部分。由“%”开头和格式字符结尾，如%d、%c、%f等。它的作用是将要输出的数据转化成指定的格式输出，格式说明都是由“%”字符开始的。

② 一般字符(非格式字符)。即按原样输出的字符，在显示时起提示作用。它由普通字符和转义字符构成。如例3-1中有

```
printf("\nc=%d\n",c);          /*双引号内换行符、字符c=都是原样输出字符*/
```

(2) 输出列表是需要输出的一些数据，如变量、函数、表达式。输出列表的个数与格式说明符的个数一般应保持一致。如



若a为5，则该语句输出的是a=5，然后换行。

2. 格式说明符

格式说明符由“%”开头，以一个英文字母结束，它表明输出数据的类型；其间还可以有一些格式控制字符，用以说明数据输出的长度、位数、对齐方式等。在C语言中格式说

明符的一般形式为：

% [-][+][0][#][输出数据最小宽度 m][.精度 n][数据长度]类型

其中[]表示可选项。

格式控制字符－、十、0、＃的具体含义如表 3-1 所示。

表 3-1　printf 函数常用附加格式符

字符形式	字 符 含 义
＋	表示输出时输出数值的符号(＋或－)。默认时若输出正值，前面无＋号，若是负数，则在数值前面输出负号(－)
－	表示输出时数值左对齐，输出宽度富余则右边补空格
0	表示输出时，在数值的前面多余的空格用 0 来代替
＃	对格式字符 o(八进制)数字前加 0，对格式字符 x 或 X(十六进制)数字前加 0x 或 0X

[**输出数据最小宽度 m**]：m 是十进制整数，它表示输出的最少位数。当需要输出的数据长度超过该数时，该选项不起作用，数据按实际位数进行输出；当需要输出的数据长度小于该数时，则在该数据的左边补空格或 0(数值型，且有格式控制字符 0 时)。

[**.精度 n**]：是“.”加上十进制整数 n。其作用如下。

① 如果输出的是实数，则该数表示小数位数，若实际位数大于所定义的精度，则超过部分按四舍五入处理；若实际位数少于所定义的精度则后面补 0。

② 如果输出的是整数，则该数表示输出整数位数，若实际位数大于所定义的位数，则照样输出；若实际位数少于所定义的位数则前面补 0。

③ 如果输出的是字符串，则表示输出字符的个数，若实际字符数多于所定义的位数，则输出前 n 个字符；若实际字符数少于所定义的位数则仅输出全部字符。

[**数据长度**]：是字符，它有两种形式，即 h、l。h 表示按短整型量输出，l 表示整数按长整型量输出，实数按 double 型输出。

类型：是格式说明符中必须要有的，它表示输出列表里要输出的数据类型。表 3-2 给出了常用的类型格式符及含义。

表 3-2　printf 函数常用类型格式符表

格式字符形式	格式字符含义
d(或 i)	表示以十进制形式输出一个带符号的整数(默认正数不输出符号)
o	表示以八进制形式输出一个无符号的整数(默认不输出前导符 0)
x(或 X)	表示以十六进制形式输出一个无符号的整数，用 x 则输出十六进制的 a～f 时以小写形式输出，用 X 则以大写形式输出(默认不输出前导符 0x 或 0X)
u	表示以十进制形式输出一个无符号的整数
f	表示以小数形式输出带符号的实数(包括单、双精度)，默认输出 6 位小数
e(或 E)	表示以指数形式输出，默认小数点前 1 位非 0 整数(负数有负号)，6 位小数，e(或 E)，符号(＋或－)，3 位指数(Turbo C 为 5 位小数，2 位指数)

续表

格式字符形式	格式字符含义
g(或 G)	表示选择%f或%e格式中占宽度较小的一种格式输出实数(不输出无意义的0)
c	表示输出一个单字符
s	表示输出一个字符串

【例 3-2】 分析下面程序运行结果。

```
/* EX3-2.C */
#include <stdio.h>
main()
{int a,b,c;
 a=123;b=456;
 c=a+b;
 printf("a+b=%d+%d=%d\n",a,b,c);
 printf("a+b=%hd+ %hd=%hd\n",a,b,c);          /* 用%hd格式输出 */
 printf("a+b=%2d+%2d=%5d\n",a,b,c);           /* 用%md格式输出 */
 printf("a+b=%02d+%02d=%05d\n",a,b,c);        /* 用%0md格式输出 */
}
```

程序运行结果如图 3-3 所示。

【例 3-3】 无符号数据的输出。

```
/* EX3-3.C */
main()
{ unsigned int a=65535;
  int b=-2;
  printf("a=%d,%o,%x,%u\n",a,a,a,a);
  printf("a=%d,%o,%x,%u\n",a,a,a,a);
}
```

程序运行结果如图 3-4 所示。

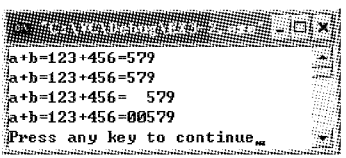


图 3-3 例 3-2 运行结果

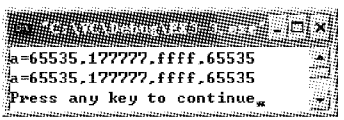


图 3-4 例 3-3 运行结果

【例 3-4】 分析下面程序运行结果。

```
/* EX3-4.C */
#include <stdio.h>
main()
{float x,y,z;
```

```
 x=123.234;y=0.000345;z=-98.567;
 printf("x=%.4f, x=%.2f, x=%+e\n",x,x,x);     /*用% .nf、%+e格式输出*/
 printf("y=%f, y=%8.4f, y=%10.3e\n",y,y,y);   /*用%f、%m.nf 、%m.ne格式输出*/
 printf("z=%+f, z=%+e, z=%+g\n",z,z,z);       /*注意g格式的使用*/
}
```

程序运行结果如图 3-5 所示。

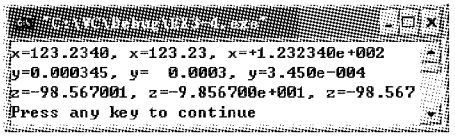


图 3-5　例 3-4 运行结果

【注意】 由于变量 x、y、z 均是单精度实型变量，因此按 f 格式输出 7 位有效数字，所以运行结果中第三行第一个 z 的值中小数点后第 6 位的数值是无效的。

【例 3-5】 分析下面程序运行结果。

```
/*EX3-5.C*/
#include <stdio.h>
main()
{double x,y,z;
 x=123.234;y=-0.000345;z=-98.567;
 printf("x=%lf, x=%.4lf, x=%2lf, x=%+e \n",x,x,x,x);
 printf("y=%lf, y=%+lf, y=%le\n",y,y,y);
 printf("z=%f, z=%lf \n",z,z);
}
```

程序运行结果如图 3-6 所示。

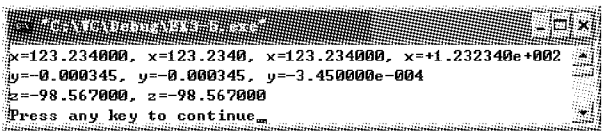


图 3-6　例 3-5 运行结果

请读者注意运行结果中第三行，考虑为什么与例 3-4 的第三行第一个结果不一样？因为这里的 x、y、z 均定义为双精度类型。对于 double 型数据，按%f 或%lf 输出效果是一样的。

【例 3-6】 分析下面程序运行结果。

```
/*EX3-6.C*/
#include <stdio.h>
main()
{printf("%c\n",'a');                          /*输出单个字符*/
 printf("%s\n","abc");                        /*输出一个字符串*/
}
```

程序运行结果如图 3-7 所示。

【注意】 在 C 语言中，字符与字符串的表示是不同的。单个字符用单撇号括起来，

而字符串要用双引号括起来，若单个字符用双引号括起来，则表示是字符串。

【思考】 将该例中第一个 printf 函数语句写成 printf("\n %c \n","a");，则结果应如何？

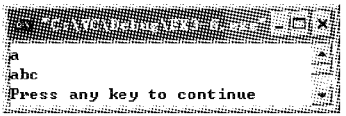


图 3-7 例 3-6 运行结果

在使用 printf 函数时，要注意以下几个问题。

(1) 在格式控制字符串中可包括“转义字符”，如'\n'、'\t'、'\r'、'\b'、'\377'等。

(2) 控制输出类型的格式符除 X(表示输出的十六进制数用大写字母输出)、E(表示输出的指数 e 用大写字母 E 输出)、G(表示若选用指数形式输出，则用大写字母 E 输出)外，其余必须是小写字母，如%d 不能写成%D。

(3) 若想输出字符“%”，则在格式字符串中用连续两个%表示。如

```
printf("%f%%",1.0/4);
```

则输出：0.250000%。

(4) 输出是从右向左计算输出项后，按格式说明的顺序对应输出。一般格式说明和输出项的个数和类型应相同，如不匹配系统并不报错，系统将按以下操作：

① 格式说明的个数少于输出项的个数，多余的输出项将不输出；格式说明的个数多于输出项的个数，多余的格式说明将输出随机的值。

② 整型数据按%f 输出或者实型数据按%d 输出，均出现输出结果错误。

3.3.2 格式化输入函数 scanf

scanf 函数的功能是从键盘上将数据按用户指定的格式输入并赋给指定的变量。

1. scanf 函数调用形式

scanf 函数是一个标准库函数，其调用的一般形式为：

scanf(格式控制字符串,地址列表);

其中格式控制字符串的定义和使用方法与 printf 函数大致相同，但不能显示格式说明以外的字符，即不能显示提示信息。格式控制字符串中若有格式说明以外的字符要原样输入。

地址列表是要赋值的各变量地址。变量地址由“&”后跟变量名组成，& 是取地址运算符，其作用是求变量的地址，如 &x 表示变量 x 的地址。

【例 3-7】 scanf 函数的使用。

```
/* EX3-7.C */
main()
{int a,b;
 scanf("%d,%d",&a,&b);
 printf("a=%d,b=%d\n",a,b);
}
```

运行时按以下方式输入 a、b 的值：25，−34 ↙，运行结果如图 3-8 所示。

此时 scanf 函数的作用是：按照 a、b 在内存的地址将 a、b 的值存进去，如图 3-8 所示。变量 a、b 地址是程序在编译连接阶段分配的。

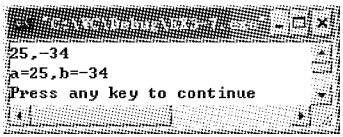


图 3-8　例 3-7 运行结果

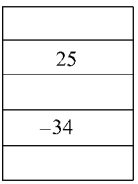


图 3-9　a、b 地址示意图

“%d,%d”表示按十进制整数形式输入数据。输入时，数据间必须用逗号分隔，用空格键、回车键、Tab(跳格)键分隔等都是不正确的。若将 scanf 函数改写成：

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

则输入时，数据间不能用逗号“,”分隔，必须用一个或多个空格分隔，也可以用回车键、Tab 键分隔。即下述几种方式输入均是合法的：

① 25□□-34↙　　(数据间用空格作为分隔，□代表空格，↙表示回车)
② 25↙　　(数据间用回车键作为分隔)
　-34↙
③ 25(按 Tab 键)-34↙　　(数据间用 Tab 键作为分隔)

2. 格式说明符

与 printf 函数中格式说明符相似，以%开始，后面跟一个格式字母，中间可以有若干个附加字符，格式说明符的一般形式为

%[*][输入数据宽度 m][长度] 类型

其中

[ ]：表示可选项。

*：表示输入的数值不赋给相应的变量，即跳过该数据不读。

[**输入数据宽度 m**]：m 是十进制正整数，表示按 m 的宽度输入数据。

[**长度**]：长度格式符为 l 和 h，l 表示输入长整型数据或双精度实型数据；h 表示输入短整型数据。

类型：是格式说明符中必须要有的，其格式符的意义与 printf 函数基本相同，具体如表 3-3 所示。

表 3-3　scanf 函数常用类型格式符表

格式字符形式	格式字符含义
d,i	表示以十进制形式输入一个整数
o	表示以八进制形式输入一个整数
x(X)	表示以十六进制形式输入一个整数

续表

格式字符形式	格式字符含义
u	表示以十进制形式输入一个无符号的整数
f或e(E)	表示输入一个实数,可以是小数形式或指数形式
g(G)	与f或e的作用相同
c	表示输入一个字符
s	表示输入一个字符串

【说明】

(1) 对unsigned型变量所需的数据,可以用%u、%d、%o或%x格式输入。

(2) 可以指定输入数据所占列数,系统自动按它截取所需数据。

(3) 如果在一个%后有一个*附加说明符,表示跳过它指定的数据。

(4) %f、%e、%E、%g、%G格式说明均可按小数(整数)形式或指数形式输入实数。

(5) 输入数据时不能规定数据的精度。例如scanf("%8.2f",&a);是不合法的。

(6) 与输出的情况不同,输入数据时长度格式符不能省略,如输入double型数据必须用%lf等。

【例3-8】 分析下面程序。

```
/*EX3-8.C*/
#include <stdio.h>
main()
{int a,b,c;
 scanf("%3d%3d%4d",&a,&b,&c );      /*按长度m进行输入*/
 printf("\na=%d,b=%d,c=%d\n",a,b,c);
}
```

程序运行时,若输入1122334455↙(正好输入10位数字)。

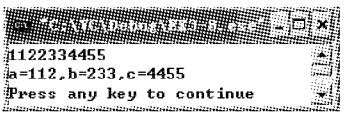


图3-10 例3-8运行结果

则运行结果如图3-10所示。

【思考】 若输入的数据不足10位,或超过10位,或数据间用空格分隔,则a、b、c获得什么值?

【例3-9】 分析下面程序。

```
/*EX3-9.C*/
#include <stdio.h>
main()
{double a,b,c;
 scanf("%lf,%lf",&a,&b );          /*按双精度进行输入*/
 c=a*b;
 printf("a*b=%lf*%lf=%le\n",a,b,c);
}
```

程序运行时,若输入12345.23456,223344.55667788↙,则运行结果如图3-11所示。