第3章 控制语句与预处理命令

CHAPTER

结构化程序有顺序、分支和循环三种结构。针对分支结构和循环结构，C语言提供了相应的语句。另外，为了便于程序的书写、阅读、修改及调试，C语言提供了编译预处理功能。

3.1 分支语句

分支结构用分支语句来实现，C语言中提供的分支语句有两种：一种是if语句，另一种是switch-case语句。

3.1.1 if语句

if语句有以下四种格式：单分支格式、双分支格式、多分支格式和嵌套格式。

1. 单分支格式

一般形式为：

if(表达式) 语句

执行过程：先计算if后面的表达式，若结果为真(非0)，执行后面的语句；若结果为假(0)，不执行该语句，其流程图见图3.1。

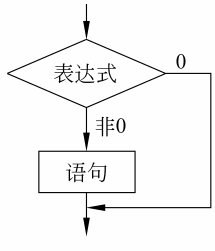

图3.1 单分支流程图

【例3.1】 输入一个整型数，输出该数的绝对值。

```
main()
{ int a;
  scanf("%d",&a);
  if(a<0)
  a=-a;
  printf("%d\n",a);
}
```

【运行结果】

```
-3↙
3
```

2. 双分支格式

一般形式为：

if(表达式) 语句1

else 语句2

执行过程：先计算if后面的表达式，若结果为真(非0)，则执行语句1；否则执行语句2。其流程图见图3.2。

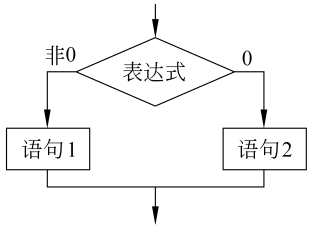

图3.2 双分支流程图

【例3.2】 输入两个整型数，将平方值较大者输出。

```
main()
{ int a,b,max;
  scanf("%d%d",&a,&b);
  if(a*a>b*b)
    max=a;
  else
    max=b;
  printf("%5d\n",max);
}
```

【运行结果】

```
2  -3↙
-3
```

3. 多分支格式

一般形式为：

if(表达式1) 语句1

else if(表达式2) 语句2

else if(表达式3) 语句3

⋮

else if(表达式n) 语句n

else 语句m

执行过程：先计算表达式1，若表达式1的结果为真(非0)，执行语句1，否则计算表达式2；若表达式2的结果为真(非0)，执行语句2，以此类推。若n个表达式的结果都为假(0)，则执行语句m，其流程图见图3.3。

由执行过程可知，n+1个语句只有一个被执行。若n个表达式的值都为假，则执行语句m，否则执行第一个表达式值为真(非0)的表达式后面的语句。

【例3.3】 输入一个百分制成绩，输出其对应的等级(90～100分为A，80～99分为

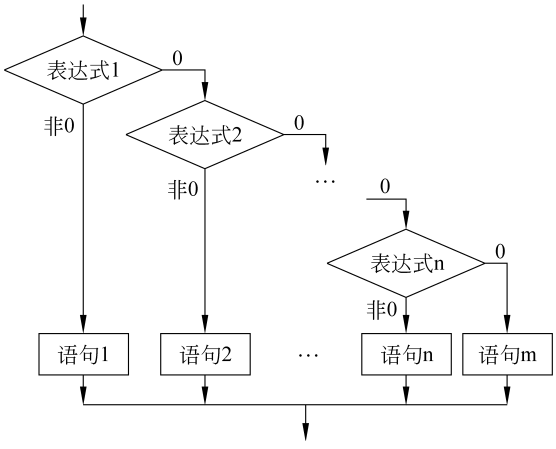

图 3.3　多分支流程图

B,70～79 分为 C,60～69 分为 D,0～59 分为 E)。

```
main()
{ int x; char y;
  scanf("%d",&x);
  if(x>=90) y='A';
  else if(x>=80) y='B';
       else if(x>=70) y='C';
            else if(x>=60) y='D';
                 else y='E';
  printf("y=%c\n",y);
}
```

【运行结果】

```
88↙
y=B
```

4. 嵌套格式

if 语句可以嵌套,即在一个 if 语句中又可以包含一个或多个 if 语句。一般形式为:

if(表达式 1)

　　if(表达式 2) 语句 1

　　else 语句 2

else

　　if(表达式 3) 语句 3

　　else 语句 4

在缺省花括号的情况下,if 和 else 的配对关系是:从最内层开始,else 总是与它前面最近且没有和其他 else 配对的 if 配对。

【例 3.4】 已知函数

$$y=\begin{cases}-1, & x<0\\ 0, & x=0\\ 1, & x>0\end{cases}$$

编写程序，输入 x，输出 y 值。

```
main()
{ float x; int y;
  scanf("%f",&x);
  if(x>=0)
    if(x>0) y=1;
    else    y=0;
  else y=-1;
  printf("x=%f y=%d\n",x,y);
}
```

【运行结果】

```
-2↙
x=-2.000000 y=-1
```

此程序还可用其他方法实现，请读者自己编写。

使用 if 语句时应注意以下几点：

(1) if 后面圆括号内的表达式可以为任意类型，但一般为关系表达式或逻辑表达式。

(2) if 和 else 后面的语句可以是任意语句。

(3) if(x)与 if(x!=0)等价。

(4) if(!x)与 if(x==0)等价。

(5) 各分支格式只是 if 语句嵌套格式的一种特例。

3.1.2 switch-case 语句

虽然用 if 语句可以解决多分支问题，但如果分支较多，嵌套的层次就多，这样会使程序冗长、降低可读性。C 语言提供了专门用于处理多分支情况的语句——switch-case 语句，其一般形式为：

switch(表达式)
{ **case 常量表达式 1：语句 1** [**break**;]
  case 常量表达式 2：语句 2 [**break**;]
  ⋮
  case 常量表达式 n：语句 n [**break**;]
  default ：语句 n+1;[**break**;]
}

switch-case 语句的执行过程是：首先计算 switch 后面圆括号中表达式的值，然后用其结果依次与各 case 后面的常量表达式的值进行比较。若相等，则执行该 case 后面的语句。执行时，如果遇到 break 语句，就退出 switch-case 语句，转至花括号的下方；否则顺

序往下执行。若与各 case 后面常量表达式的值都不相等，则执行 default 后面的语句。

【例 3.5】 用 switch-case 语句实现例 3.3。

```
main()
{ int a;
  char y;
  scanf("%d",&a);
  switch(a/10)
  { case 10:y='A';break;
    case 9:y='A';break;
    case 8:y='B';break;
    case 7:y='C';break;
    case 6:y='D';break;
    default:y='E';break;
  }
  printf("y=%c\n",y);
}
```

【运行结果】

```
88↙
y=B
```

请读者比较例 3.3 和例 3.5。

说明：

(1) switch 后面的圆括号后不能加分号。

(2) switch 后面圆括号内表达式的值必须为整型、字符型或枚举型。

(3) 各 case 后面常量表达式的值必须为整型、字符型或枚举型。

(4) 各 case 后面常量表达式的值必须互不相同。

(5) 若每个 case 和 default 后面的语句都以 break 语句结束，则各个 case 和 default 的位置可以互换。

(6) case 后面的语句可以是任何语句，也可以为空，但 default 的后面不能为空。若为复合语句，则花括号可以省略。

(7) 若某个 case 后面的常量表达式的值与 switch 后面圆括号内表达式的值相等，就执行该 case 后面的语句。执行完后若没有遇到 break 语句，就不再进行判断，接着执行下一个 case 后面的语句。若想执行完某一语句后退出，就必须在语句最后加上 break 语句。

(8) 多个 case 可以共用一组语句。如例 3.5 中的程序段：

```
case 10:y='A';break;
case 9:y='A';break;
```

可以改为：

```
case 10:
case 9:y='A';break;
```

(9) switch-case 语句可以嵌套，即一个 switch-case 语句中又含有 switch-case 语句。

【例 3.6】 switch-case 语句的嵌套举例。

```
main()
{ int x,y,a=0,b=0;
  scanf("%d%d",&x,&y);
  switch(x)
   { case 1:
        switch(y)
         {
          case 0:a++;break;
          case 1:b++;break;
         }
     case 2:a++;b++;break;
     case 3:a++;b++;
   }
  printf("a=%d,b=%d\n",a,b);
}
```

【运行结果】

```
1  0↙
a=2,b=1
```

3.2 循环语句

C 语言中有三种循环语句：while 语句、do-while 语句和 for 语句。它们都是在条件成立时反复执行某个程序段，这个反复被执行的程序段称为循环体。循环体是否被继续执行要依据某个条件，这个条件称为循环条件。

3.2.1 while 语句

while 语句的一般形式为：

while(表达式)

循环体

其中，表达式可以是任意类型，一般为关系表达式或逻辑表达式，其值为循环条件。循环体可以是任何语句。

while 语句的执行过程为：

(1) 计算 while 后面圆括号中表达式的值，若其结果为非 0，转(2)；否则转(3)。

(2) 执行循环体，转(1)。

(3) 退出循环，执行循环体下面的语句。

其流程图见图 3.4。

while 语句的特点是：先判断表达式，后执行循环体。

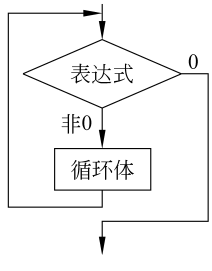

图 3.4 while 语句流程图

【例 3.7】 从键盘上输入 10 个小于 100 的整数，输出偶数的个数及偶数和。

```
main()
{ int i,n=0,sum=0,a;
  i=1;                                  /*循环变量赋初值*/
  while(i<=10)                          /*循环条件为 i<=10*/
   { scanf("%d",&a);
     if(a%2==0)
       {n++;sum+=a;}
     i++;                               /*循环变量增值,使 i 趋于大于 10*/
   }
  printf("n=%d sum=%d\n",n,sum);
}
```

【运行结果】

```
1  2  3  4  5  6  7  8  9  10↙
n=5 sum=30
```

说明：

(1) 由于 while 语句是先判断表达式，后执行循环体，所以循环体有可能一次也不执行。

(2) 循环体可以是任何语句。如果循环体不是空语句，则不能在 while 后面的圆括号后加分号(;)。

(3) 在循环体中要有使循环趋于结束的语句，如例 3.7 中 i++。

3.2.2　do-while 语句

do-while 语句的一般形式为：

do 循环体 while(表达式);

其中，表达式可以是任意类型，一般为关系表达式或逻辑表达式，其值为循环条件。循环体可以是任意语句。

do-while 语句的执行过程为：

(1) 执行循环体，转(2)。

(2) 计算 while 后面圆括号中表达式的值，若其结果为非 0，转(1)；否则转(3)。

(3) 退出循环，执行循环体下面的语句。

其流程图见图 3.5。

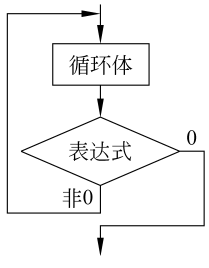

图 3.5　do-while 语句流程图

do-while 语句的特点是：先执行循环体，后判断表达式。

【例 3.8】 计算整数 n 的值，使 1+2+3+…+n 刚好大于或等于 500。

```
main()
{ int n=0,sum;
```

```
  sum=0;                                    /*循环变量赋初值*/
  do
   {n++;
    sum+=n;                                 /*循环变量增值,使sum趋于500*/
   }while(sum<500);                         /*循环条件为:sum<500*/
  printf("n=%d sum=%d\n",n,sum);
}
```

【运行结果】

```
n=32 sum=528
```

说明：

（1）do-while语句最后的分号(;)不可少，否则将出现语法错误。

（2）循环体中要有使循环趋于结束的语句，如例3.8中sum＋＝n。

（3）由于do-while语句是先执行循环体，后判断表达式，所以循环体至少执行一次。

3.2.3 for语句

for语句的一般形式为：

for(表达式1;表达式2;表达式3)

循环体

其中，循环体可以是任意语句。三个表达式可以是任意类型，一般来说，表达式1用于给某些变量赋初值，表达式2用来说明循环条件，表达式3用来修正某些变量的值。

for语句的执行过程为：

（1）计算表达式1，转(2)。

（2）计算表达式2，若其值为非0，转(3)；否则转(5)。

（3）执行循环体，转(4)。

（4）计算表达式3，转(2)。

（5）退出循环，执行循环体下面的语句。

其流程图见图3.6。

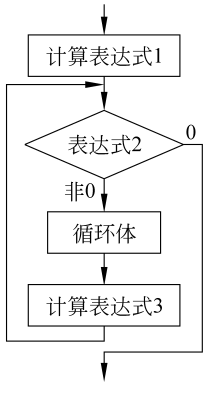

图3.6 for语句流程图

for语句的特点是：先判断表达式，后执行循环体。

【例3.9】 计算1～100之间的整数和。

```
main()
{ int i,sum;
  sum=0;
  for(i=1;i<=100;i++)  sum+=i;
                  /*i=1是为循环变量赋初值,i<=100是循环条件,i++是修正循环变量*/
  printf("sum=%d\n",sum);
}
```

【运行结果】

```
sum=5050
```

在 for 语句中，表达式 1 和表达式 3 经常使用逗号表达式简化程序，提高程序运行效率。这是逗号表达式的主要用途。如例 3.9 中的程序段：

```
sum=0;
for(i=1;i<=100;i++)  sum+=i;
```

可以改写成：

```
for(i=1,sum=0;i<=100;sum+=i,i++);
```

在 for 语句中，在分号(;)必须保留的前提条件下，三个表达式的任何一个都可以省略，因此 for 语句又有如下省略形式。

1. for(;表达式 2;表达式 3) 循环体

表达式 1 省略。此时应在 for 语句之前给变量赋初值。如例 3.9 中的程序段：

```
for(i=1;i<=100;i++) sum+=i;
```

可以改写成：

```
i=1;
for(;i<=100;i++) sum+=i;
```

2. for(表达式 1;表达式 2;) 循环体

表达式 3 省略。此时应在循环体中修正循环变量。如例 3.9 中的程序段：

```
for(i=1;i<=100;i++)  sum+=i;
```

可以改写成：

```
for(i=1;i<=100;)  {sum+=i;i++;}
```

3. for(表达式 1; ;表达式 3) 循环体

表达式 2 省略。此时认为表达式 2 的值始终为真，如果循环体中不包含 break 语句或 goto 语句，循环就无法终止，是死循环。如例 3.9 中的程序段：

```
for(i=1;i<=100;i++)  sum+=i;
```

可以改写成：

```
for(i=1;;i++)  {if(i>100) break; sum+=i;}            /*break 的功能见 3.2.5*/
```

4. for(;表达式 2;) 循环体

表达式 1 和表达式 3 同时省略。此时应在 for 语句之前给变量赋初值，在循环体中修正循环变量。如例 3.9 中的程序段：

```
for(i=1;i<=100;i++)  sum+=i;
```

可以改写成：

```
i=1;
for(;i<=100;)  {sum+=i;i++;}
```

这种情况完全等同于 while 语句。

5. for(;;表达式 3) 循环体

表达式 1 和表达式 2 同时省略。此时应在 for 语句之前给变量赋初值，在循环体中用 break 语句或 goto 语句退出循环。如例 3.9 中的程序段：

```
for(i=1;i<=100;i++)  sum+=i;
```

可以改写成：

```
i=1;
for(; ;i++)  {if(i>100)  break; sum+=i;}
```

6. for(表达式 1;;) 循环体

表达式 2 和表达式 3 同时省略。此时应在循环体中修正循环变量，在循环体中用 break 语句或 goto 语句退出循环。如例 3.9 中的程序段：

```
for(i=1;i<=100;i++)  sum+=i;
```

可以改写成：

```
for(i=1;;)  {sum+=i; i++; if(i>100)  break;}
```

7. for(;;) 循环体

三个表达式同时省略。此时相当于：while(1) 循环体。应在 for 语句之前给变量赋初值，在循环体中修正循环变量，在循环体中用 break 语句或 goto 语句退出循环。如例 3.9 中的程序段：

```
for(i=1;i<=100;i++)  sum+=i;
```

可以改写成：

```
i=1;
for(; ;)  { sum+=i; i++; if(i>100)  break;}
```

说明：

(1) 若循环体不是空语句，则不能在 for 语句的圆括号后加分号(;)。

(2) 表达式 1 或表达式 2 省略时，其后的分号不能省略，并且不能用其他符号代替。

(3) 要有使循环趋于结束的语句。

3.2.4 循环语句的嵌套

若循环语句的循环体中又有循环语句，则称为循环语句的嵌套。三种循环语句可以互相嵌套，并且可以嵌套多层。

【例 3.10】 输出九九表。

```
main()
{ int i,j;
  for(i=1;i<=9;i++)
```