第3章

C 程序控制结构

在第 1 章中已经介绍，一个 C 语言程序由一个或多个函数组成，一个函数包含声明部分和执行部分。声明部分主要定义本函数内用到的变量，执行部分由若干条语句组成，指定在函数内进行的操作，即函数的功能。


视频讲解

3.1 C 语 句

C 语句可以分为 5 大类：控制语句、函数调用语句、表达式语句、空语句和复合语句。

1. 控制语句

控制语句用于控制程序的执行流程，共有 9 种语句，可分为选择语句、循环语句和辅助语句 3 类。

(1) 选择语句：if()…else…、switch。

(2) 循环语句：for()…、while()…、do…while()。

(3) 辅助语句：continue、break、goto、return。

其中，括号内是控制条件，使用时用具体的条件代替。例如：

```
if(x>y)
    z=x;
else
    z=y;
```

2. 函数调用语句

由一个函数调用加一个分号构成，格式如下：

函数名(参数表);

例如：

```
printf("This is a C statement.");
```

3. 表达式语句

表达式后加一个分号“;”，就构成了表达式语句，例如，a=3 和 i=i+1 是赋值表达式，

但不是语句，而“a=3;”和“i=i+1;”则是赋值语句。

4. 空语句

空语句指只含有一个分号的语句，例如：

```
;
```

空语句常用在循环语句或函数体中。

5. 复合语句

复合语句是将多个语句用大括号括起来的语句，在语法上作为一个语句，例如：

```
if(a>b)
{ t=a;a=b;b=t; }
```

3.2 顺序结构程序举例

一个 C 语言程序可由顺序、选择、循环 3 种基本控制结构组成。

(1) 顺序结构表示程序中的各个操作是按照它们出现的先后顺序执行的。

(2) 选择结构表示程序的处理步骤出现了分支，需要根据某一特定的条件选择其中的一个分支执行。

(3) 循环结构表示程序反复执行某个或某些操作，直到某条件为假(或为真)时才终止循环。

程序的整体结构是顺序结构，是按顺序从第一条语句开始执行到最后一条语句，其中可能嵌有选择结构和循环结构。

【例 3-1】 输入一个学生的 3 门成绩，求总分和平均分，并输出。

```
#include<stdio.h>
int main()
{
    int score1,score2,score3,sum;
    float aver;
    scanf("%d%d%d",&score1,&score2,&score3);
    sum=score1+score2+score3;
    aver=sum/3.0;
    printf("sum=%d,aver=%.1f\n",sum,aver);
    return 0;
}
```

输入：

```
76 85 90
```

运行结果：

```
sum=251,aver=83.7
```

3.3 选 择 结 构

用顺序结构只能编写一些简单的程序。在求解实际问题时，用户往往会遇到先要判断一个条件，然后根据条件是否满足进行不同处理的情况，称为选择结构(分支结构)。C语言提供了实现分支结构的if语句和switch语句。


视频讲解

3.3.1 if语句

用if语句可以构成分支结构，其功能是根据给定的条件进行判断，决定执行某个分支程序段。

1. if语句的形式

C语言中的if语句有3种基本形式，下面分别进行介绍。

(1) 单分支if语句。

单分支if语句的一般形式如下：

```
if(表达式)
语句
```

例如：

```
if(x>y)
    printf("%d",x);
```

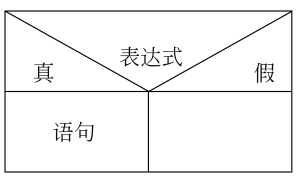

图3-1 单分支if语句的执行过程

单分支if语句的执行过程是，先求解表达式的值，如果表达式的值为真，那么执行其后的内嵌语句，否则执行if之后的语句，其执行过程如图3-1所示。

注意：if后面的表达式是分支条件，一定要用小括号()括起来。

【例3-2】 求x的绝对值。

```
#include<stdio.h>
int main()
{
    int x;
    scanf("%d",&x);
    if(x<0)
        x=-x;                          //if的内嵌语句
    printf("|x|=%d\n",x);
    return 0;
}
```

输入：

```
-10
```

运行结果：

```
|x|=10
```

输入：

```
10
```

运行结果：

```
|x|=10
```

if 的内嵌语句可以是一条语句，也可以是多条语句，此时一定要用大括号“{ }”将多条语句括起来形成一条复合语句。

【例 3-3】 输入两个整数，按由小到大的顺序输出这两个数。

```
#include<stdio.h>
int main()
{
    int x,y,t;
    scanf("%d,%d",&x,&y);
    if(x>y)
    {   t=x;  x=y;  y=t;   }          //复合语句,功能是交换变量 x 和 y 的值
    printf("%d,%d\n",x,y);
    return 0;
}
```

输入：

```
10,5
```

运行结果：

```
5,10
```

注意：if 的内嵌语句是由 3 条赋值语句构成的复合语句，如果把复合语句的大括号去掉，写成：

```
if(x>y)
    t=x;  x=y;  y=t;
```

则 if 的内嵌语句是“t=x;”，其他两条语句“x=y;”和“y=t;”是 if 语句后的语句。如果输入：

```
5,10
```

则运行结果不正确。

(2) 双分支 if 语句。

双分支 if 语句的一般形式如下：

```
if(表达式)
    语句 1
else
    语句 2
```

例如：

```
if(x>0)
    printf("%d",x);
else
    printf("%d",-x);
```

双分支 if 语句的执行过程是，先求解表达式的值，如果表达式的值为真，则执行语句 1，否则执行语句 2，其执行过程如图 3-2 所示。

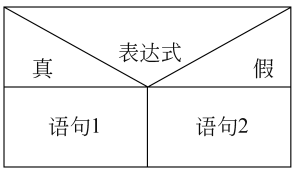

图 3-2　双分支 if 语句的执行过程

【例 3-4】 求 x 和 y 两个数中的较大者。

```
#include<stdio.h>
int main()
{
    int x,y,max;
    scanf("%d,%d",&x,&y);
    if(x>y)
        max=x;                    //if 的内嵌语句
    else
        max=y;                    //else 的内嵌语句
    printf("max=%d\n",max);
    return 0;
}
```

输入：

```
5,10
```

运行结果：

```
max=10
```

注意：else 不能作为语句单独使用，必须是 if 语句的一部分，与 if 配对使用，所以 if…else是一条语句。

在双分支的 if 语句中，if 和 else 后面的内嵌语句可以是一条语句，也可以是多条语句。如果是多条语句，一定要用大括号“{ }”将多条语句括起来形成一条复合语句，例如：

```
if(x>y)
{  x=y;  y++;  }                  //if 的内嵌语句
else
{  y=x;  x--;  }                  //else 的内嵌语句
```

如果写成：

```
if(x>y)
    x=y;  y++;
else
{  y=x;  x--;  }
```

则有语法错误，因为此时 if 是一条单分支 if 语句，其内嵌语句为“x=y;”，而 else 没有 if 与它配对，所以出现语法错误。

如果写成：

```
if(x>y)
{  x=y;  y++;  }
else
    y=x;  x--;
```

则当 x>y 时，运行结果错误。因为此时 else 的内嵌语句为“y=x;”，当条件成立时，不执行“y=x;”，而执行“x--;”。

(3) 多分支 if 语句。

多分支 if 语句可以看成双分支 if 语句的扩展形式，即在双分支 if 语句中，else 的内嵌语句是另一个双分支的 if 语句，如此扩展下去，形成多分支 if 语句。

多分支 if 语句的一般形式为：

```
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 语句 n
```

多分支 if 语句的执行过程是，如果表达式 1 的值为真，则执行语句 1，否则判断表达式 2；如果表达式 2 的值为真，则执行语句 2，否则判断表达式 3……以此类推。如果所有表达式的值都为假，则执行语句 n，其执行过程如图 3-3 所示。

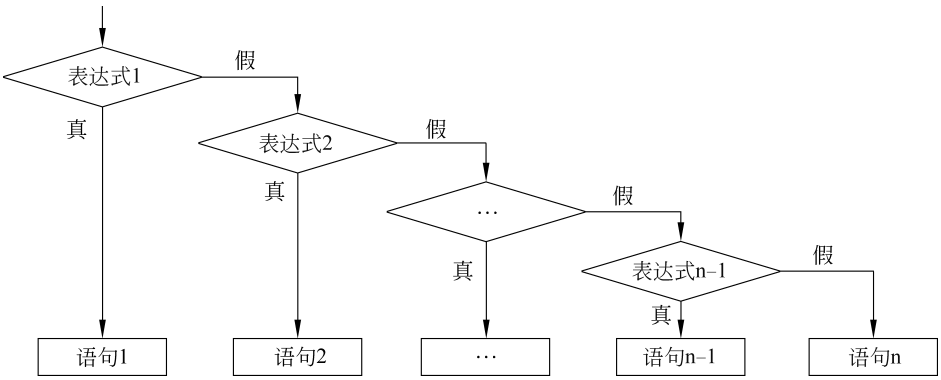

图 3-3　多分支 if 语句的执行过程

【例 3-5】 输入一个百分制的成绩，输出相应的等级。如果输入成绩为 90～100，则输出 A；输入成绩为 80～89，则输出 B；输入成绩为 70～79，则输出 C；输入成绩为 60～69，则输出 D；输入成绩小于 60，则输出 E。

```
#include<stdio.h>
int main()
{
    int score;
    scanf("%d",& score);
    if(score >=90)
        printf("score grade is A\n");
    else if(score >=80)
        printf("score grade is B\n");
    else if(score >=70)
        printf("score grade is C\n");
    else if(score >=60)
        printf("score grade is D\n");
    else
        printf("score grade is E\n");
return 0;
}
```

输入：

```
85
```

运行结果：

```
score grade is B
```

输入：

```
55
```

运行结果：

```
score grade is E
```

注意：在 3 种形式的 if 语句中，关键字 if 之后均为表达式。该表达式通常是关系表达式或逻辑表达式，但可以扩展为任何类型的表达式，甚至可以是一个常量或变量。如果表达式的值非 0，则按真处理，条件成立；否则按假处理，条件不成立。例如：

```
if(5)语句;
if(a)语句;
if(a=0)语句;
```

都是允许的。

2. if 语句的嵌套

在 if 语句中，if 和 else 的内嵌语句可以是任何语句，如果内嵌语句又是 if 语句，那么

称为 if 语句的嵌套，其一般形式如图 3-4 所示。

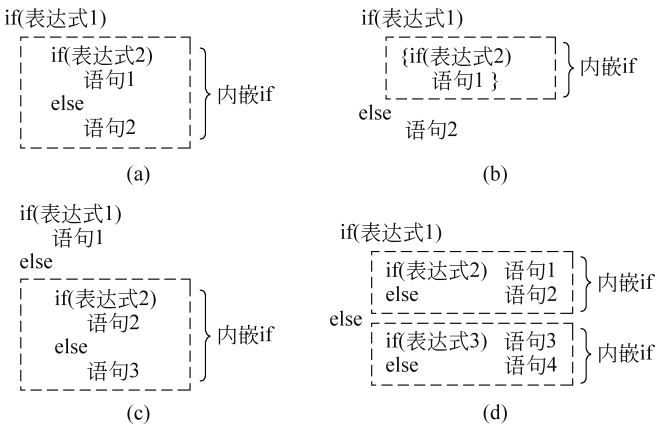

图 3-4　if 语句的嵌套

嵌套的 if 语句会出现多个 if 和多个 else 的情况，这时用户要注意 if 和 else 的配对问题。例如：

```
if(a>b)
    if(b>c)
        printf("%d is biggest\n",a);
else
    printf("%d is not biggest\n",a);
```

其中，else 究竟与哪一个 if 配对？为了避免这种二义性，C 语言规定，else 总是与它上面最近的未曾配对的 if 配对，因此上述例子应属于图 3-4(a)所示的情况。例如：

```
if(a>b)
    if(b>c)                          //***
        printf("%d is biggest\n",a);
    else                             //else 与标记***的 if 配对
        printf("%d is not biggest\n",a);
```

如果要使 else 与第一个 if 配对，可以加大括号{ }来确定配对关系(见图 3-4(b))。例如：

```
if(a>b)
{
    if(b>c)
        printf("%d is biggest\n",a);
}
else
    printf("%d is not biggest\n",a);
```

【例 3-6】 求 x、y、z 3 个数中的最大者。

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

```
int main()
{  int x=4,y=5,z=2,max;
   max=x;
   if(z>y)
   {   if(z>x)
           max=z;
    }
    else
        if(y>x)
           max=y;
    printf("max=%d",max);
    return 0;
}
```

运行结果：

```
max=5
```


视频讲解

3.3.2 switch 语句

通过嵌套的 if 语句可以实现多分支结构，但如果分支较多，则嵌套的 if 语句层就会较多，理解起来就会较困难。switch 语句是专门用于处理多分支结构的条件选择语句，又称开关语句，其一般形式如下(通常使用 switch 语句)：

```
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;]  ]
}
```

其中，用中括号“[ ]”括起来的部分表示可以省略。

switch 语句的执行过程为：首先求解 switch 后面括号中表达式的值，然后用此值依次与各个 case 后面的常量表达式的值进行比较。若括号中表达式的值与某个 case 后面的常量表达式的值相等，则执行此 case 后面的语句。语句执行后若遇到 break 语句或 switch 的结束符“}”就终止 switch 语句，否则继续执行下一个 case 后面的语句。以此类推。若括号中表达式的值与所有 case 后面的常量表达式都不相等，则执行 default 后面的语句 n+1。

例如：

```
int no;
scanf("%d",&no);
```

```
switch(no)
{
    case 1 : printf("first\n"); break;
    case 2 : printf("second\n");
    case 3 : printf("third\n");
}
```

输入：

```
1
```

运行结果：

```
first
```

输入：

```
2
```

运行结果：

```
second
third
```

程序分析：

如果输入 1，则执行 case 1 后面的语句，输出 first，遇到 break 语句，终止 switch 语句。如果输入 2，则执行 case 2 后面的语句，输出 second，继续执行 case 3 后面的语句，输出 third，遇到 switch 语句的结束符}，switch 语句结束。

说明：

(1) case 和常量表达式之间要有空格，常量表达式只起语句标号作用，跳出 switch 必须用 break 语句。如果每个 case 和 default 后面都有 break 语句，则它们出现的次序不影响执行结果，例如：

```
switch(no)
{
    default: printf("last\n"); break;
    case 2 : printf("second\n"); break;
    case 1 : printf("first\n"); break;
    case 3 : printf("third\n"); break;
}
```

(2) case 后面常量表达式的值必须互不相同。

(3) 多个 case 可共用一组执行语句。case 后可包含多个可执行语句，且不必加“{ }”，进入某个 case 后，会顺序执行该 case 后面的所有语句。

【例 3-7】 将例 3-5 进行修改，用 switch 语句实现。

```
#include<stdio.h>
int main()
```

```
{
    int score;
    scanf("%d",& score);
    switch(score/10)
    {
        case10:
        case 9: printf("score grade is A\n"); break;
        case 8: printf("score grade is B\n"); break;
        case 7: printf("score grade is C\n"); break;
        case 6: printf("score grade is D\n"); break;
        default: printf("score grade is E\n"); break;
    }
    return 0;
}
```

程序分析：

本程序中，case 10 和 case 9 共用一组语句。case 后有两条语句，不需要用大括号括起来。

(1) default 部分可以省略。如果省略，当 switch 后面括号中表达式的值与所有 case 后面的常量表达式的值都不相等时，则不执行任何一个分支，直接退出 switch 语句。

例如，将例 3-7 中 switch 语句的 default 部分去掉，则当输入小于 60 的整数时，switch 语句中的任何一条语句都不被执行。

(2) switch 语句可以嵌套。

【例 3-8】 嵌套的 switch 语句。

```
#include<stdio.h>
int main()
{
    int x=1,y=0,a=0,b=0;
    switch(x)
    {
        case 1:
            switch(y)
            {
                case 0: a++; break;              //break 语句终止 switch(y)
                case 1: b++; break;              //break 语句终止 switch(y)
            }
        case 2: a++;b++; break;                  //break 语句终止 switch(x)
        case 3: a++;b++;
    }
    printf("\na=%d,b=%d",a,b);
    return 0;
}
```

运行结果：

```
a=2,b=1
```

程序分析：

本程序中有一条嵌套的 switch 语句。执行外层 switch(x)语句，由于 x 的值为 1，执行 case 1 后面的 switch(y)语句。由于 y 的值为 0，执行 case 0 后面的语句。a 的值变成 1，遇到 break 语句，终止内层的 switch(y)，继续执行 switch(x)的 case 2 后面的语句。a 的值变成 2，b 的值变成 1，遇到 break 语句，终止外层的 switch(x)。


视频讲解

3.3.3 条件运算符与条件表达式

在 C 语言中有唯一的一个三元运算符，即条件运算符，由条件运算符连接的式子称为条件表达式，其一般形式如下：

表达式 1?表达式 2:表达式 3

例如：

```
x>y?x:y
```

说明：

(1) 条件表达式的求解过程为先求解表达式 1，如果表达式 1 的值为真(非 0)，则将表达式 2 的值作为整个条件表达式的值，否则将表达式 3 的值作为整个条件表达式的值。例如：

```
max=x>y?x:y;                    //将 x 和 y 中较大者赋给 max
y=x>0?x:-x;                     //将 x 的绝对值赋给 y
```

(2) 条件运算符的优先级高于赋值运算符和逗号运算符，低于其他运算符。例如：

```
a%3?a+2:a-2          等价于  (a%3)?(a+2):(a-2)
a>5&&a<10?a++:a--    等价于  (a>5&&a<10)?(a++):(a--)
b=a+3>7?10:20        等价于  b=((a+3)>7?10:20)
```

(3) 条件运算符的结合方向为自右至左。条件表达式可以嵌套，当一个条件表达式中出现多个条件运算符时，应该将位于最右边的问号与离它最近的冒号配对，并按这一原则正确区分各条件运算符的运算对象。例如：

```
a>b?10:b>c?20:30   等价于  a>b?10:(b>c?20:30)
```


视频讲解

3.3.4 选择结构程序举例

【例 3-9】 从键盘输入一个字符，判断该字符是否为大写字母。如果是，则转换为小写字母输出；如果不是，则原样输出。

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

```
int main()
{
    char ch;
    ch=getchar();
    if(ch>='A '&&ch<='Z ')
        ch=ch+32;
    printf("%c\n",ch);
    return 0;
}
```

输入：

```
A
```

运行结果：

```
a
```

【例 3-10】 从键盘输入一个数字字符，输出对应的星期名称，其中，0 对应星期日，1～6 分别对应星期一到星期六。

```
#include<stdio.h>
int main()
{
    char ch;
    ch=getchar();
    switch(ch)
    {
        case '0': printf("Sunday\n");break;
        case '1': printf("Monday\n");break;
        case '2': printf("Tuesday\n");break;
        case '3': printf("Wednesday\n");break;
        case '4': printf("Thursday\n");break;
        case '5': printf("Friday\n");break;
        case '6': printf("Saturday\n");break;
        default: printf("error\n");
    }
    return 0;
}
```

输入：

```
5
```

运行结果：

```
Friday
```

程序分析：

本题 switch 语句中要使用 break 语句，否则输出存在错误。

3.4 循 环 结 构

循环结构是在给定条件成立时，反复执行某段程序，直到条件不成立为止。给定的条件称为循环条件，反复执行的程序段称为循环体。C 语言提供了多种循环语句，可以组成各种不同形式的循环结构。本节介绍 while 语句、do-while 语句和 for 语句。


视频讲解

3.4.1 while 语句

while 语句的一般形式如下：

while(表达式) 语句

其中，表达式是循环条件，要用小括号“()”括起来，语句为循环体。

while 语句的执行过程为先求解表达式的值，当其值为真时，执行循环体语句，然后再次求解表达式的值并进行判断，否则循环结束。while 语句的流程图和 N-S 图分别如图 3-5 和图 3-6 所示。流程图由一些特定意义的图形、流程线及简要的文字说明构成，能清晰、明确地表示程序的运行过程。在使用过程中，人们设计了一种新的流程图，它把整个程序写在一个大框图内，这个大框图由若干个小的基本框图构成，简称 N-S 图。

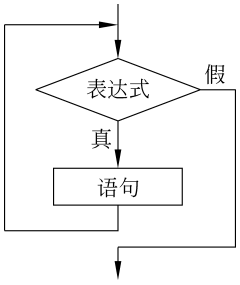

图 3-5 while 语句的流程图

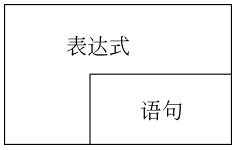

图 3-6 while 语句的 N-S 图

【例 3-11】 用 while 语句计算 1＋2＋3＋…＋100。

```
#include<stdio.h>
int main()
{
    int k,sum=0;
    k=1;                              //为循环变量 k 赋初值
    while(k<=100)                     //循环条件 k<=100
    {
        sum=sum+k;
        k++;                          //循环变量值增 1
    }
    printf("%d\n",sum);
```

```
    return 0;
}
```

运行结果：

```
5050
```

例 3-11 的流程图和 N-S 图分别如图 3-7 和图 3-8 所示。

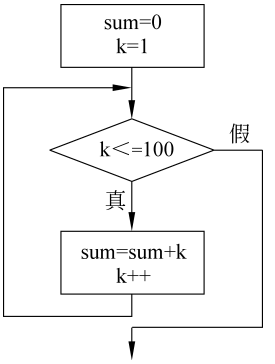

图 3-7　例 3-11 的流程图

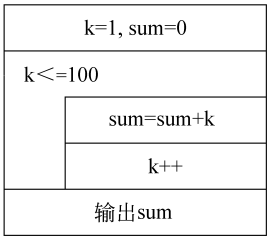

图 3-8　例 3-11 的 N-S 图

说明：

(1) 循环条件表达式不仅限于关系表达式和逻辑表达式，可以是任意类型的表达式。如果表达式的值非 0，表示条件为真，执行循环体语句；如果表达式的值为 0，表示条件为假，终止循环，执行 while 语句后面的语句。例如：

```
k=10;
while(k--)
    printf("%3d\n",k);
```

while 语句的执行过程是计算 k--的值为 10，k 的值变成 9，输出 9。再计算 k--的值为 9，k 的值变成 8，输出 8，…，计算 k--的值为 1，k 的值变成 0，输出 0。最后计算 k--的值为 0，循环结束，k 的值变成-1。

(2) while 语句的特点是先判断循环条件，后执行循环体。循环体有可能一次也不被执行。

(3) 当循环体包含一条以上的语句时，必须用大括号{}括起来，组成复合语句，否则会出现程序结果不正确或死循环的情况。例如：

```
#include<stdio.h>
int main()
{
    int k=1,sum=0;
    while(k<=10)
        sum=sum+k;
        k++;
    printf("%d\n",sum);
    return 0;
}
```

本程序执行时出现死循环，因为 while 的循环体语句是“sum＝sum＋k;”，每次执行完循环体后，k 的值没有改变，循环条件一直成立，循环将一直执行下去。

(4) 在循环体中应有使循环趋于结束的语句。例如，循环条件是 k≤10，在循环体中应有使 k 增值从而导致 k>10 的语句，例如“k ++;”。如果无此语句，循环条件将一直成立，循环无法结束。


视频讲解

3.4.2 do-while 语句

do-while 语句的一般形式如下：

```
do
    语句
while(表达式);
```

do-while 语句的执行过程是先执行循环体语句，然后求解表达式。如果表达式的值为假(0)，终止循环；如果为真(非 0)，则继续循环。do-while 语句的流程图和 N-S 图分别如图 3-9 和图 3-10 所示。

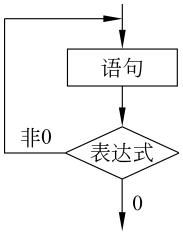

图 3-9　do-while 语句的流程图

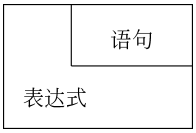

图 3-10　do-while 语句的 N-S 图

【例 3-12】 用 do-while 语句计算 1＋2＋3＋…＋100 的值。

```
#include<stdio.h>
int main()
{
    int k,sum=0;
    k=1;                              //为循环变量 k 赋初值
    do
    {
        sum=sum+k;
        k++;                          //循环变量 k 值增 1
    }
    while(k<=100);                    //循环条件 k<=100
    printf("%d\n",sum);
    return 0;
}
```

说明：

(1) do-while 语句的特点是先执行循环体，后判断循环条件，循环体至少要执行一次。

(2) do-while 语句的循环体是 do 和 while 之间的语句，如果有一条以上语句，要用大括号“{}”括起来，构成复合语句。

(3) while 语句与 do-while 语句的区别在于以下两点。

① while 语句先判断循环条件，如果条件成立，继续循环，否则终止循环。do-while 语句先执行一次循环体语句，然后判断循环条件，如果条件成立，继续循环，否则终止循环。

② 如果循环条件一开始就不成立，则 while 语句的循环体一次也不被执行，而 do-while 语句的循环体执行一次。


视频讲解

3.4.3 for 语句

for 语句的一般形式如下：

```
for([表达式 1];[表达式 2];[表达式 3])
    语句
```

其中，用中括号[ ]括起来的部分表示可以省略，即 3 个表达式可以省略，但用于分隔 3 个表达式的 2 个分号“;”不能省略。

for 语句的执行过程如下。

(1) 求解表达式 1。

(2) 求解表达式 2，若其值为假(0)，则结束循环，转到步骤(5)；若为真(非 0)，则执行循环体语句。

(3) 求解表达式 3。

(4) 转回到步骤(2)，继续执行。

(5) 循环结束，执行 for 语句下面的语句。

for 语句的流程图和 N-S 图分别如图 3-11和图 3-12 所示。

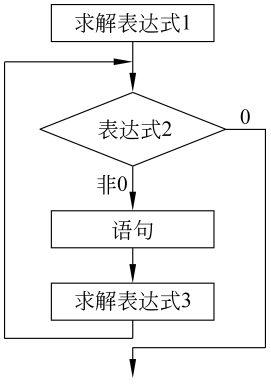

图 3-11　for 语句的流程图

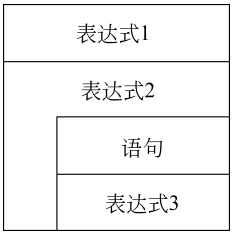

图 3-12　for 语句的 N-S 图

for 语句常见的应用形式如下：

```
for([循环变量赋初值];[循环条件];[循环变量值增减])
    语句
```

【例 3-13】 用 for 语句计算 1＋2＋3＋…＋100 的值。

```
#include<stdio.h>
int main()
{
    int k,sum=0;
    for(k=1;k<=100; k++)
        sum=sum+k;
    printf("%d\n",sum);
    return 0;
}
```

说明：

(1) 若 for 语句的循环体中有多条语句，要用大括号{}构成复合语句。

(2) 在 for 语句中，for 后面小括号中的 3 个表达式可以省略，但分号不能省略。

① 表达式 1 省略，此时应把它放在 for 语句前面。例如：

```
k=1;
for(;k<=10; k++)
    sum=sum+k;
```

② 表达式 2 省略，此时应在循环体中判断循环何时结束，否则出现死循环。例如：

```
for(k=1; ; k++)
{  if(k>10)
        break;                    //终止循环
    sum=sum+k;
}
```

③ 表达式 3 省略，此时应把它放在循环体中。例如：

```
for(k=1; k<=10; )
{  sum=sum+k;
    k++;
}
```

④ 表达式 1 和表达式 3 同时省略，只有表达式 2。例如：

```
k=1;
for( ;k<=10; )
{  sum=sum+k;
    k++;
}
```

⑤ 3 个表达式可以同时省略。例如：

```
k=1;
for( ; ; )
{  if(k>10)
        break;
```

```
    sum=sum+k++;
}
```

(3) 表达式 1 和表达式 3 可以是逗号表达式。例如：

```
for(k=1,sum=0; k<=10 ; sum+=k,k++);
```

表达式 1 是逗号表达式，为 k 和 sum 赋值。表达式 3 也是逗号表达式，依次改变 sum 和 k 的值。


视频讲解

3.4.4 循环嵌套

在一个循环体中又包含另一个循环语句，这称为循环的嵌套。如果是两个循环嵌套在一起，称为双重循环。C 语言支持多重循环。以双重循环为例，外层循环和内层循环均可是本节介绍的 3 种循环语句。3 种循环语句的嵌套如图 3-13～图 3-15 所示。

```
while()          while()          while()
{…               {…               {…
    while()          do               for()
        语句             语句             语句
                         while();
    ⋮                ⋮                ⋮
}                }                }
```

图 3-13　while 语句嵌套

```
do               do               do
{…               {…               {…
    while()          do               for()
        语句             语句             语句
                         while()
    ⋮                ⋮                ⋮
}                }                }
while();         while();         while();
```

图 3-14　do-while 语句嵌套

```
for()            for()            for()
{…               {…               {…
    while()          do               for()
        语句             语句             语句
                         while();
    ⋮                ⋮                ⋮
}                }                }
```

图 3-15　for 语句嵌套

双重循环的执行过程是外层循环的循环体执行一次，内层循环执行一遍。

【例 3-14】 编写一个程序，使输出结果如下：

```
1 * 1=1
1 * 2=2   2 * 2=4
1 * 3=3   2 * 3=6   3 * 3=9
1 * 4=4   2 * 4=8   3 * 4=12   4 * 4=16
⋮
1 * 9=9   2 * 9=18   3 * 9=27   4 * 9=36   5 * 9=45   6 * 9=54   7 * 9=63   8 * 9=72   9 * 9=81
```

例 3-14 的流程图如图 3-16 所示。

```
#include<stdio.h>
int main()
{
     int i,j;
     printf("\n");
     for(i=1;i<10;i++)
    {
        for(j=1;j<=i;j++)
            printf("%d * %d=%-4d",j,i,i * j);
        printf("\n");
    }
    return 0;
}
```

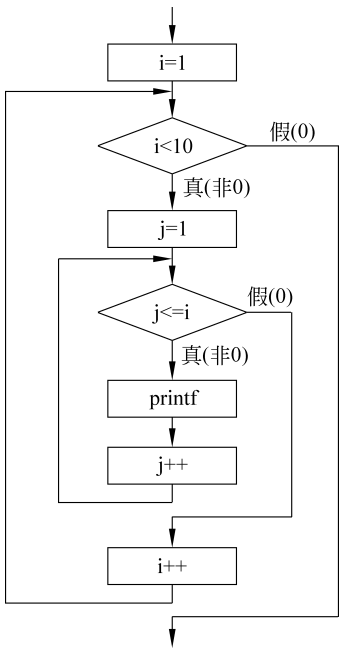

图 3-16　例 3-14 的流程图


视频讲解

3.4.5　break 语句和 continue 语句

1. break 语句

break 语句的一般形式如下：

```
break;
```

break 语句只能用于循环语句和 switch 语句中。当 break 语句用于循环语句中时，可终止循环，执行循环后面的语句。break 语句一般和 if 语句一起使用，即满足条件时跳出循环。例如：

```
int k,m;
for(k=0;k<10;k++)
{
    m=k * k;
    if(m>50)  break;
}
printf("%d\n",m);
```

运行结果：