第5章 选择结构程序设计

CHAPTER 5

选择结构又称分支结构，是基本结构之一，在大多数程序中都包含选择结构。它的作用是根据对给定条件的判断来选择其一个分支执行。C语言提供了两种选择语句：if语句（条件语句）和switch语句（开关语句）。

本章重点：正确使用各种关系的运算，掌握if语句和switch语句的功能并在编程中熟练运用。

5.1 关系运算和逻辑运算

5.1.1 关系运算

C语言提供了一组关系运算符，如表5.1所示。它们用来比较两个运算对象的大小。

表5.1 关系运算符

运算符	名 称	例 子
>	大于	a>b，a大于b
<	小于	a<b，a小于b
==	等于	a==b，a等于b
>=	大于或等于	a>=b，a大于或等于b
<=	小于或等于	a<=b，a小于或等于b
!=	不等于	a!=b，a不等于b

关系运算符都是二元（双目）运算符，它们的优先级比算术运算符低，高于赋值运算符。在关系运算符中，<、<=、>、>=同级，它们高于==和!=。关系运算符的结合性都是自左至右。

用关系运算符将两个表达式连接起来就成为关系表达式。例如，a>b、x==y、a+b>=c+b都是合法的关系表达式。

关系表达式的值是一个逻辑值,即真或假。C 语言没有专门的逻辑型数据,而是用 1(或非 0)表示真,用 0 表示假。因此表达式 5<3 的值为假,即为 0。而表达式 a>b 的值则取决于 a、b 的值,但只可能是真或假(1 或 0)两种情况之一。

再看一个复杂一些的关系表达式如何求值(设 a=2,b=3):

```
c=5-3>=a+1<=b+2
```

在这个表达式中有赋值运算、算术运算和关系运算。其中算术运算优先级最高,关系运算次之,赋值运算最低,所以先进行算术运算,即

```
c=2>=3<=5
```

然后进行关系运算,关系运算符的结合性为自左至右,先计算 2>=3,结果为假,其值为 0,即

```
c=0<=5
```

再进行关系运算 0<=5,结果为真,其值为 1,故 c 的值为 1。

字符型数据可按其 ASCII 码值进行比较。例如:

'a'>'b':结果为假,值为 0。

'a'>50:结果为真,值为 1。

5.1.2 逻辑运算

逻辑运算符是用来对运算对象做逻辑运算的。C 语言提供了 3 种逻辑运算符,如表 5.2 所示。

表 5.2 逻辑运算符

运算符	名称	例 子
!	逻辑非	!a,a 反
&&	逻辑与	a&&b,a 与 b
||	逻辑或	a||b,a 或 b

!(逻辑非)为单目运算符,右结合。其运算规则是:当运算量为 0 时,运算结果为 1;当运算量为 1 时,运算结果为 0。

&&(逻辑与)为双目运算符,左结合。其运算规则是:只有当两个运算量都是非 0 时,运算结果才为 1,否则为 0。

||(逻辑或)为双目运算符,左结合。其运算规则是:只要有一个运算量为非 0,运算结果就为 1;只有两个运算量都为 0 时,结果才是 0。

这 3 个运算符的优先级如下:逻辑非!最高,逻辑与 && 次之,逻辑或|| 最低。!的优先级高于算术运算符和关系运算符,而 && 和||的优先级低于算术运算符和关系运算符。由此可见:

```
a>b&&c>d        相当于        (a>b)&&(c>d)
```

```
a==0||b==0      相当于      (a==0)||(b==0)
!a&&b==c        相当于      (!a)&&(b==c)
```

用逻辑运算符将逻辑量(表示逻辑的常量、变量、函数、关系表达式等)连接起来的式子称为逻辑表达式。逻辑表达式的值是一个逻辑值,用 1 表示真,用 0 表示假。而在判断一个量的真或假时,以非 0 表示真,以 0 表示假。例如 a=3,b=2,则

!a:相当于!3,值为 0。

a&&b:相当于 3&&2,值为 1。

a&&!b:相当于 3&&0,值为 0。

!a||b:相当于 0||2,值为 1。

逻辑与和逻辑或运算分别具有如下性质:

a&&b,当 a 为 0 时,不管 b 为何值,结果为 0。

a||b,当 a 为 1 时,不管 b 为何值,结果为 1。

利用上述性质,在计算连续的逻辑与运算时,若有运算分量的值为 0,则表达式的结果为 0,不再计算后面的运算分量;在计算连续的逻辑或运算时,若有运算分量的值为 1,则表达式的结果为 1,不再计算后面的运算分量。上述性质也称为短路特性。

注意:运算符 && 和||的短路特性的副作用。思考下面的表达式:

```
i>0&&++j>0
```

如果 i>0 的结果为假,将不会计算表达式++j>0,那么 j 也不会自增。把表达式的条件变成++j>0&&i>0,就可以解决这个短路问题。更好的办法是单独对 j 进行自增操作。

关系运算和逻辑运算经常用于流程控制,如分支语句或循环语句的条件表达式中。

5.2　if 语句

if 语句是条件选择语句,它通过判断给定的条件是否满足来决定所要执行的操作。

5.2.1　if 语句的 3 种形式

if 语句有单分支、双分支和嵌套(多分支)3 种形式。

1. 单分支 if 语句

单分支的 if 语句格式如下:

```
if(表达式)  语句
```

例如:

```
if(a>b) printf("%d",a);
```

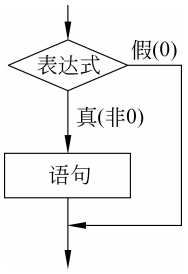

图 5.1　单分支 if 语句的执行过程

单分支 if 语句执行过程如图 5.1 所示,首先计算 if 后面括号内表达式的值。如果它的值为真(非 0),就执行语句;如果它的值

为假(0),就转到 if 语句的下一条语句去执行。if 语句中的表达式通常为关系表达式或逻辑表达式,也可以是算术表达式。

例 5.1 求一个整数的绝对值。

```
#include <stdio.h>
void main()
{
    int a;
    scanf("%d",&a);
    if(a<0) a=-a;
    printf("这个数的绝对值是%d\n",a);
}
```

运行时输入-101,结果如下:

```
这个数的绝对值是 101
```

2. 双分支 if 语句

双分支 if 语句的格式如下:

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

例如:

```
if(a>b)
    printf("%d",a);
else
    printf("%d",b);
```

双分支 if 语句的执行过程如图 5.2 所示,首先计算 if 后面括号内表达式的值。如果它的值为真(非 0),就执行语句 1;如果它的值为假(0),就执行语句 2。

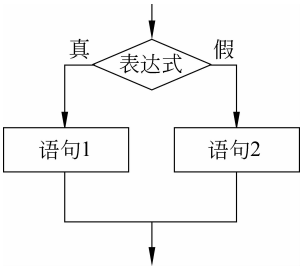

图 5.2 双分支 if 语句的执行过程

例 5.2 编写程序,输入两个整数,求其中较大者。

```
#include <stdio.h>
void main()
{
    int a,b,large;
    scanf("%d%d",&a,&b);
    if(a>b) large=a;
      else  large=b;
    printf("large=%d\n",large);
}
```

运行时输入 10 20,结果如下：

```
large=20
```

在 if 语句中,语句 1 和语句 2 可以是单个语句,也可以是由多个语句组成的复合语句。

例 5.3　输入两个数,要求将大者赋予 x,小者赋予 y。

```
#include <stdio.h>
void main()
{
    int a,b,x,y;
    scanf("%d%d",&a,&b);
    if(a>b) {x=a; y=b;}
    else    {x=b; y=a;}
    printf("x=%d  y=%d\n",x,y);
}
```

运行时输入 3 4,结果如下：

```
x=4  y=3
```

3. if 语句嵌套(多分支)

if 语句中的语句 1 和语句 2 本身又可以是一个 if 语句,这就是 if 语句的嵌套,用这种嵌套实现多分支 if 语句。其执行过程如图 5.3 所示。下面是一种 if 语句嵌套最常用的形式——else-if 结构。

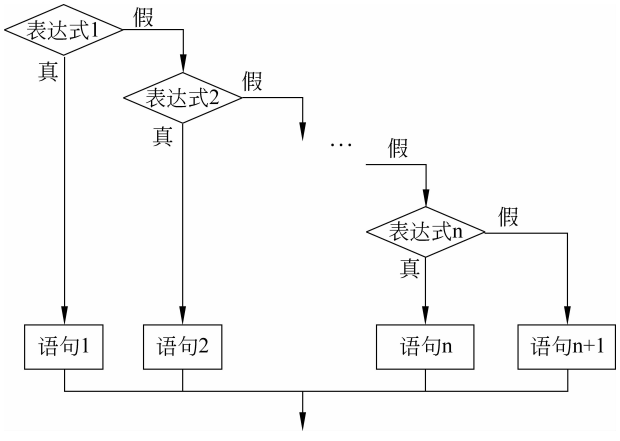

图 5.3　if 语句的嵌套(多分支)

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

```
…
else if(表达式 n)
    语句 n
else
    语句 n+1
```

其含义是：如果表达式 1 为真，则执行语句 1；否则，如果表达式 2 为真，则执行语句 2……以此类推，如果表达式 n 为真，则执行语句 n；如果各表达式都不为真，则执行语句 n+1。

例 5.4 编写一个求解符号函数的程序。

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

程序如下：

```
#include <stdio.h>
void main()
{
    int x,sign;
    printf("Please input a number:\n");
    scanf("%d",&x);
    if(x>0)
        sign=1;
    else if(x==0)
        sign=0;
    else
        sign=-1;
    printf("The sign is %d.\n",sign);
}
```

运行时出现以下提示信息：

```
Please input a number:
```

输入 10，结果如下：

```
The sign is -1
```

例 5.5 程序要求输入百分制成绩，然后按此成绩输出成绩等级(90～100 为 A，80～89 为 B，70～79 为 C，60～69 为 D，60 以下为 E)。

```
#include <stdio.h>
void main()
{
    int score;
    printf("请输入一个成绩 0~100:");
```

```
    scanf("%d",&score);
    printf("The grade is:");
    if(score>=90)
        printf("%c\n",'A');
    else if(score>=80)
        printf("%c\n",'B');
    else if(score>=70)
        printf("%c\n",'C');
    else if(score>=60)
        printf("%c\n",'D');
    else
        printf("%c\n",'E');
}
```

运行时出现以下提示信息：

```
请输入一个成绩(0~100):
```

输入 85，结果如下：

```
The grade is:B
```

上面的 else-if 结构并不能代表 if 嵌套的全部情况，更一般的情况是 if 后面和 else 后面的语句都可以再包含 if 语句。例如：

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

这里有两个 if 和一个 else，显然如果 if 和 else 配对不同，则语句的执行效果是不一样的，于是就出现了二义性。为此 C 语言规定，else 总是与它前面最近的一个无 else 的 if 配对。根据这一规定，上面语句中的 else 是与第二个 if 配对。如果想要使 else 与第一个 if 配对，则可在相应的 if 语句上加上大括号：

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

注意：不要混淆＝＝(判等)运算符和＝(赋值)运算符。语句 if(i＝＝0)…测试 i 是否等于 0，而语句 if(i＝0)…则是先把 0 赋值给 i，然后测试赋值表达式的结果是否非零值，在这种情况下，测试总是会失败的。

把＝＝运算符与＝运算符混淆是最常见的 C 语言编程错误。

5.2.2　条件运算

条件运算符(?:)是 C 语言唯一的三目运算符，它连接 3 个运算分量。条件运算符构

成的表达式的一般形式如下：

```
表达式 1?表达式 2:表达式 3
```

它的执行过程是：先计算表达式 1，如果其值为真，则计算表达式 2 的值，并作为结果值，否则计算表达式 3 的值作为结果值。例如：

```
max=(a>b)?a:b;
```

该语句执行时，当 a＞b 条件成立时，变量 max 取 a 值，否则取 b 值。

在 if 语句中，当 if 和 else 都只带一个赋值语句，且给同一变量赋值时，就可用条件运算来代替。例如上述条件运算可替代下面的 if 语句：

```
if(a>b)
    max=a;
else
    max=b;
```

条件运算符的优先级较低，只高于赋值运算符和逗号运算符。

条件表达式体现了 C 语言简明的风格，这是 C 语言区别于其他高级语言的一个显著特点。

5.3 switch 语句

5.2 节介绍的 if 语句一般适用于单分支和双分支的选择，尽管也可以通过 if 嵌套形式实现多分支的选择，但这种方式由于嵌套层次过多，影响了程序的可读性。C 语言提供了一种更适于多分支选择的 switch 语句，又称开关语句。它的一般形式如下：

```
switch(表达式)
{
    case 常量表达式 1: 语句组 1
    case 常量表达式 2: 语句组 2
    ⋮
    case 常量表达式 n: 语句组 n
    default:          语句组 n+1
}
```

其中，switch、case、default 均为 C 语言的保留字。switch 后面的表达式通常为整型、字符型或枚举型。常量表达式又称开关常数或分支标号，必须是与表达式类型一致的整数、字符或枚举常数。语句组 1～n＋1 可以是单个语句，也可以是多个语句，如果是多个语句也不必用大括号括起来。default 和语句组 n＋1 部分可以省略。

switch 语句的执行过程是：首先计算 switch 后面小括号内表达式的值，然后依次与各个 case 后面的常量表达式的值相比较，若一致就执行该 case 后面的语句，直到遇到 break 语句，或 switch 语句执行结束，就转到 switch 语句后面的语句去执行；如果表达式的值与所有常量表达式的值都不相等，则转向 default 后面的语句去执行；如果没有

default 部分，则不执行 switch 语句中的任何语句，而直接转到 switch 语句后面的语句去执行。

每个 case 代表一个分支，其后面的语句组代表该分支所要执行的操作。但是，如果语句组中没有 break 语句，程序就可能一直执行，从而进入其他分支，这在 C 语言中是允许的。但一个好的习惯是在每个语句组中以 break 结束，从而保持各分支的独立性。

注意：忘记使用 break 语句是编程时常犯的错误。虽然有时会故意忽略 break 以便分支共享代码，但很多错误是忘记加上 break 导致的。

例 5.6　将例 5.5 程序中的 if-else 语句改用 switch 语句。

```
#include <stdio.h>
void main()
{
    int score;
    printf("请输入一个成绩(0~100):");
    scanf("%d",&score);
    printf("The grade is:");
    switch (score/10)
    {
        case 10:
        case 9:   printf("%c\n",'A');break;
        case 8:   printf("%c\n",'B');break;
        case 7:   printf("%c\n",'C');break;
        case 6:   printf("%c\n",'D');break;
        default: printf("%c\n",'E');
    }
}
```

运行时出现以下提示信息：

```
请输入一个成绩(0~100):
```

输入 85，结果如下：

```
The grade is:B
```

在使用 switch 语句时，应注意以下几点：

(1) switch 后面小括号内表达式的值与 case 后面常量表达式的值都必须是整型、字符型或枚举型。

(2) switch 语句中所有 case 后面的常量表达式的值必须互不相同，而多个 case 的后面可以共用一组语句。例如：

```
switch(x)
{
    case 0:
    case 1: 语句组 1; break;
}
```

是合法的,表示当 x=0 或 x=1 时,都执行语句组 1 和 break。而下面的语句是不合法的:

```
switch(c)
{
    case 'a': 语句组 1; break;
    case 'a': 语句组 2; break;
    …
}
```

(3) case 后面的语句可以是单个语句,也可以是多个语句,但不需要用大括号括起来。

(4) switch 语句中的 case 和 default 出现的次序是任意的,即 default 也可位于 case 之前,且 case 的次序也不要求按常量表达式的大小顺序排列。

5.4 应用举例

例 5.7 编写程序,从键盘输入年份,判断其是否为闰年。

分析:闰年的条件是,能被 4 整除但不能被 100 整除,或者能被 400 整除。

整除描述:如果 X 能被 Y 整除,则余数为 0,即如果 X%Y 的值等于 0,则表示 X 能被 Y 整除。设年份变量为 year,标志是否闰年的变量 leap,leap=1 为闰年,否则不是闰年。该程序的算法见图 5.4。程序的下面几种写法说明编写程序没有固定的格式,主要看个人喜欢使用什么语句功能来实现。

写法 1:使用复合语句。

```
void main()
{
    int year,leap;
    printf("请输入年份: \n");
    scanf("%d",&year);
    if(year%4==0)
    {
        if(year%100==0)
        {
            if(year%400==0)
                leap=1;
            else
                leap=0;
        }
        else
            leap=1;
        }
        else
        leap=0
```

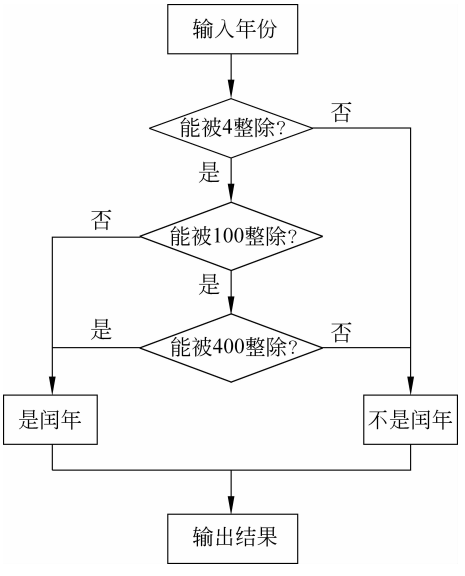

图 5.4 判断闰年的算法