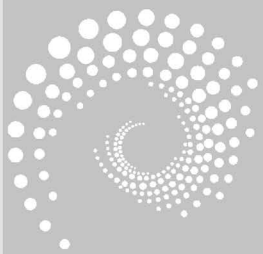
第3章

分 支 结 构

CHAPTER 3



学习目标

- 掌握关系运算符和关系表达式的表示方法。
- 掌握逻辑运算符和逻辑表达式的表示方法。
- 掌握 if 语句和 switch 语句的基本用法。
- 掌握分支结构程序设计的方法。

结构化程序设计思想认为：所有计算机程序都可以用顺序结构、选择结构、循环结构这三种基本结构表示。顺序结构是一组按前后顺序执行的语句；选择结构能根据运行时的情况选择要执行的语句组；循环结构允许多次重复执行一组语句。用这三种基本结构组合而成的程序结构清晰，可读性好，也便于调试和修改。

前面章节的程序都是顺序结构的，各语句是按自上而下的顺序执行的，执行完上一个语句就自动执行下一个语句，是无条件的，不必作任何判断。实际上，在很多情况下需要根据某个条件是否满足来决定是否执行指定的操作任务，这就是分支结构要解决的问题。例如，在教务系统中规定停开选课人数不足 20 的课程，那么是否停开某个课程就要根据其选课人数是否大于或等于 20 这个条件来决定。

分支结构的执行是依据一定的条件选择执行路径，而不是严格按照语句出现的物理顺序。分支结构的程序设计方法的关键在于构造合适的分支条件和分析程序流程，根据不同的程序流程选择适当的分支语句。本章介绍如何利用关系表达式或逻辑表达式在 C 语言中表示条件，以及如何用 if 语句或 switch 语句实现分支结构。

3.1　关系运算符和关系表达式

在 C 语言中，比较运算符也称关系运算符。所谓“关系运算”就是“比较运算”，将两个数值进行比较，判断其比较的结果是否符合给定的条件。例如，s < 60 就是一个关系表达式，“<”是关系运算符，如果 s 的值小于 60，即满足“s < 60”条件，则关系表达式“s < 60”的值为“真”(true)；否则，关系表达式“s < 60”的值为“假”(false)。

3.1.1　关系运算符

C 语言中提供了 6 种关系运算符，且均为二元运算符(即两个运算对象)，其结合性均为左结合。其具体含义见表 3.1。

表 3.1　关系运算符及其含义

关系运算符	含　义	关系运算符	含　义
<	小于	<=	小于或等于
>	大于	>=	大于或等于
==	等于	!=	不等于

注意：关系运算符“==”由两个等于号组成，而赋值运算符是一个“=”。关系运算符“!=”中感叹号是英文感叹号。

关系运算符的优先级如下。

(1) 前 4 种关系运算符(<,<=,>,>=)的优先级相同，后两种关系运算符(==,!=)的优先级也相同，但前 4 种运算符的优先级高于后两种。例如，“<=”优先于“==”，“<=”与“>”优先级相同。

(2) 关系运算符的优先级高于赋值运算符。

(3) 关系运算符的优先级低于算术运算符。

例如：

```
x>y<1 等价于 (x>y)<1        x==y>1 等价于 x==(y>1)
x=y>1 等价于 x=(y>1)        x>y+1 等价于 x>(y+1)
```

3.1.2　关系表达式

用关系运算符将两个数值或数值表达式连接起来的式子称为关系表达式。例如：x＞y，x＋10＜y，'A'＜'a'，(x＝10)＞6，5＞＝2，1＜0 等均为合法的关系表达式。在 C99 标准之前，C 语言中没有逻辑型类型，true 用 1 表示，false 用 0 表示；反之，0 代表 false，非 0 代表 true。若 x＝1，y＝2，z＝3，则：

关系表达式“x＋y＜z”的值为“假”，其值实际为 0。

关系表达式“x＋y＞＝z”的值为“真”，其值实际为 1。

赋值表达式“x＝y＜＝z”的作用是：由于 y＜＝z 的值为“真”，所以赋值后 x 的值为 1。

3.2　逻辑运算符和逻辑表达式

有些情况下，判断的条件不是一个单一条件，而涉及多个方面，可以分解为多个简单条件组成的复合条件。例如，要统计年龄不超过 30，月薪超过 8000 的人数。这就需要检查两个条件：①年龄 age＜＝30；②月薪 salary＞8000。这个组合条件不能用一个关系表达式来表示，要用两个表达式的组合来表示，即 age＜＝30 AND salary＞8000，AND 的含义是“与”，即“二者同时满足”的意思。这个复合的关系表达式就是一个逻辑表达式。

3.2.1　逻辑运算符

C 语言提供了三种逻辑运算符，见表 3.2。

表 3.2　逻辑运算符及其含义

逻辑运算符	含　义	运算对象个数	结　合　性
&&	逻辑与	2(二元运算符)	左结合
||	逻辑或	2(二元运算符)	左结合
!	逻辑非	1(一元运算符)	右结合

如果 a 和 b 均为逻辑量，则可以用来表示逻辑运算符的运算规则的“真值表”如表 3.3 所示。

表 3.3　逻辑运算符的真值表

a	b	a&&b	a||b	!a
真	真	真	真	假
真	假	假	真	假
假	真	假	真	真
假	假	假	假	真

一个逻辑表达式如果包含多个逻辑运算符，其优先次序为：逻辑非(!)最高，逻辑与

(&&)次之,逻辑或(||)最低。其中逻辑非(!)比算术运算符优先级高,而逻辑与(&&)和逻辑或(||)比关系运算符优先级低。例如,设 x,y,z 为逻辑量,则

```
x || y && z 等价于 x || (y && z)        x && !y 等价于 x && (!y)
x && y<1 等价于 x && (y<1)              x && 0+2 等价于 x && (0+2)
```

3.2.2　逻辑表达式

用逻辑运算符将关系表达式或逻辑量连接起来的式子就是逻辑表达式。逻辑表达式的值同关系表达式的值一样,即“真”或“假”。

如前所述,逻辑表达式的值应该是一个逻辑量“真”或“假”。C 语言编译系统在表示逻辑运算结果时,通常以数值 1 代表“真”,以数值 0 代表“假”,但在判断一个量是否为“真”时,以 0 代表“假”,以非 0 代表“真”。即将一个非零的数值认作为“真”。例如:

(1) 若 x=5,则!x 的值为 0。因为 x 的值为非 0,当成“真”,进行逻辑非运算后得“假”,即为 0。

(2) 若 x=3,y=4,则 x&&y 的值为 1。因为 x 和 y 的值均为非 0,当成“真”,所以 x&&y 的值也为“真”,即为 1。

(3) 5||3&&7-7 的值为 1。

由此可知,逻辑表达式的值不是 0 就是 1,不可能是其他数值。而参加逻辑运算的运算对象可以是 0 也可以是 1 或其他任何数值。这时就应该区分表达式中不同位置上出现的数值,哪些作为数值运算或关系运算的对象,哪些作为逻辑运算的对象。例如:

```
2<1 || 8>8-!0
```

自左向右扫描该表达式并进行计算。首先处理“2<1”(因为关系运算符优先于逻辑运算符)得 0(代表假)。再对式子“0 || 8>8-!0”进行计算,同理根据优先规则应先计算“8>8-!0”。现在,第二个“8”左边是“>”,右边是“-”,“-”优先级高,所以应先计算“8-!0”,又由于“!”比“-”优先级高,故先求“!0”的值,得 1。然后,对式子“0 || 8>8-1”依次进行“-”“>”“||”三个运算得 1。即表达式“2<1 || 8>8-!0”的值为 1(代表真)。

在逻辑表达式的求解过程中,并不是所有的逻辑运算符都被执行,只是在必须执行下一个逻辑运算符才能计算出表达式的值时,才执行该逻辑运算符。假如 x、y、z 均为逻辑量,则有:

(1) x || y && z。如果 x 为真,则不需要判别 y 和 z,因为此时已经可以确定表达式的值为真;只有当 x 为假时,才需要求解 y && z 的值。求解过程如图 3.1 所示。

(2) x && y || z。如果 x 和 y 都为真,则不需要判别 z,因为此时已经可以确定表达式的值为真;当 x 和 y 不同时为真时,需要判别 z 的值。求解过程如图 3.2 所示。

换句话说,在(1)中,对运算符||来说,只有 x=0(x 为假)时,才继续右边 y&&z 的运算。在(2)中,对运算符 && 来说,只有 x≠0(x 为真)时,才继续右边 y 的运算。因此,如果有下面的逻辑表达式:

```
(x=a>b)||(y=c>d)
```

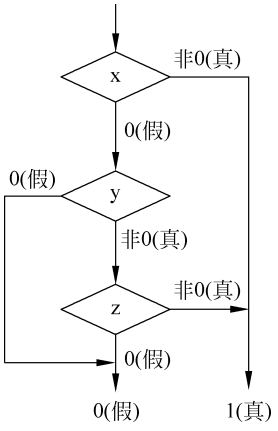


图 3.1　x||y&&z 的求解过程

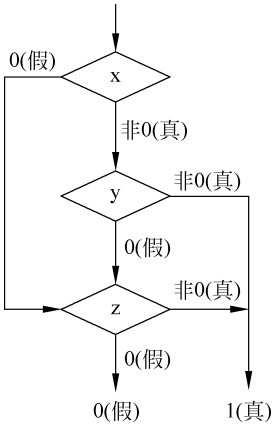


图 3.2　x&&y||z 的求解过程

当 a=1,b=0,c=3,d=2,x 和 y 的原值为 0 时,由于“a > b”的值为 1,因此 x=1,此时已经能够确定表达式的值为 1,不必进行“(y=c > d)”的运算,因而 y 的值保持不变,仍为 0。

掌握了 C 语言的关系运算符和逻辑运算符后,就可以很方便地用一个逻辑表达式来表示复杂的条件。例如,判别某年是否是闰年的问题就可以用一个逻辑表达式来表示条件。闰年的条件是必须满足下列两个条件之一:①能被 4 整除,但不能被 100 整除,如 2020;②能被 400 整除,如 2000。假设 year 代表年份(整数),可以写出闰年满足的逻辑表达式为

```
(year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)
```

3.3　if 语句

用 if 语句可以构成分支结构,每次运行程序时它根据给定的条件进行判断,决定执行某个分支程序。C 语言提供了三种形式的 if 语句。

3.3.1　单分支 if 语句

单分支 if 语句是最简单的 if 语句形式,可以用来设计简单的分支结构程序。

单分支 if 语句的一般形式:

```
if(表达式)
    语句
```

其语义是:如果表达式的值为真(非 0),则执行其后的语句,否则不执行该语句。其执行过程如图 3.3 所示。表达式可以是关系表达式、逻辑表达式或数值表达式,其中最直观、最容易理解的是关系表达式和逻辑表达式。

【例 3.1】　编程实现:输入两个整数,输出其中较大者。

分析:用两个 int 型变量 x 和 y 存放输入的两个整数,先假定 x 是较大者并存放到 int 型变量 max 中,如果 x < y,则将 max 的值修改为 y,最后 max 中存放的值一定是它们中的较大者,输出 max 即可。算法流程如图 3.4 所示。

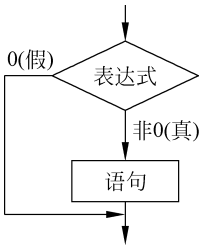


图 3.3　单分支 if 语句执行过程

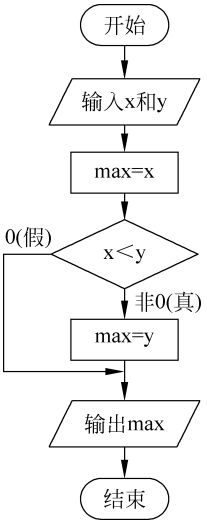


图 3.4　例 3.1 流程图

程序 3.1　输入两个整数，输出其中较大者。

```
#include <stdio.h>
int main(void)
{
    int x,y,max;
    printf("Input two numbers:");
    scanf("%d%d",&x,&y);
    max = x;
    if(x < y) max = y;
    printf("Max = %d\n",max);
    return 0;
}
```

程序运行结果如下。

```
Input two numbers:3 4
Max = 4
```

【例 3.2】　编程实现：输入一个数，输出其绝对值。

分析：已知负数的绝对值是它的相反数，非负数的绝对值是它自己。可以用一个 double 型变量 x 存放输入的数，用另一个 double 型变量 y 存放 x 的绝对值。可先假定 x 不是负数，y 赋值为 x，然后判断 x 是否小于 0，如果 x＜0，则修改 y 为－x。最后输出 y。算法流程如图 3.5 所示。

程序 3.2　输入一个数，输出其绝对值。

```
#include <stdio.h>
int main(void)
{
    double x,y;
    printf("Input x = ");
    scanf("%lf",&x);
    y = x;
    if(x < 0)
```

```
    y = -x;
    printf("|%g| = %g\n",x,y);
    return 0;
}
```

程序运行结果如下。

```
Input x = -3.14
|-3.14| = 3.14
```

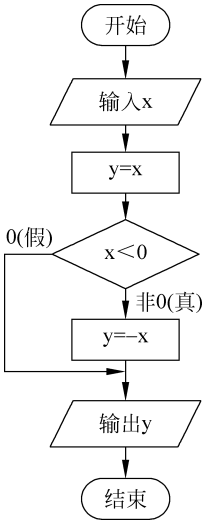


图 3.5　例 3.2 流程图

输出格式类型%g 用来输出实数，它根据数值的大小，自动选%f 格式或%e 格式，选择输出时占宽度较小的一种，且不输出无意义的 0。

【例 3.3】 编程实现：输入两个整数，按其值由大到小的顺序输出。

分析：两个数的大小顺序只有两种情况。可以用一个 int 型变量 x 存放输入的第一个整数，用另一个 int 型变量 y 存放输入的第二个整数，然后判断 x 与 y 的大小，如果 x < y，则将 x 与 y 的值进行互换，最后依次输出 x 和 y。互换两个变量的值则需要第三个变量 t，这个与下述问题类似：甲同学与乙同学想互换各自杯中的饮料，而又不想换杯子，则必然需要第三个杯子来中转一下。

程序 3.3　输入两个整数，按其值由大到小的顺序输出。

```
#include <stdio.h>
int main(void)
{
    int x,y,t;
    printf("Input two numbers: ");
    scanf("%d%d",&x,&y);
    if(x<y)
    {
        t=x; x=y; y=t;
    }
    printf("%d %d\n",x,y);
```

```
    return 0;
}
```

程序运行结果如下。

```
Input two numbers: 3 4
4 3
```

上述程序代码中，交换 x 和 y 的值也可以写成“t=y; y=x; x=t;”。请思考这是为什么呢？

3.3.2　双分支 if 语句

单分支 if 语句实际上也可以认为是双分支结构的特例，即其中一个分支的操作为空。从这个意义上讲，双分支 if 语句更具有一般性。

双分支 if 语句一般形式：

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

其语义是：如果表达式的值为真(非 0)，则执行其后的语句 1，否则执行语句 2。其执行过程如图 3.6 所示。

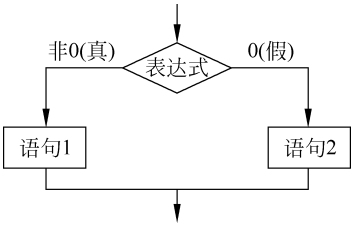


图 3.6　双分支 if 语句执行过程

【例 3.4】 编程实现：输入两个整数，按其值由大到小的顺序输出。(用双分支 if 语句)

分析：两个数的大小顺序只有两种情况。可以用一个 int 型变量 x 存放输入的第一个整数，用另一个 int 型变量 y 存放输入的第二个整数，然后判断 x 与 y 的大小。如果 x>y，则依次输出 x 和 y；否则依次输出 y 和 x。算法流程如图 3.7 所示。

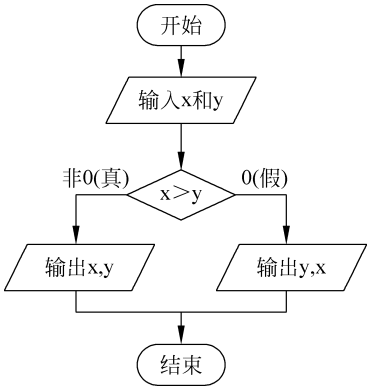


图 3.7　例 3.4 流程图

程序 3.4　输入两个整数，按其值由大到小的顺序输出。

```
#include <stdio.h>
int main(void)
{
    int x,y,t;
    printf("Input two numbers: ");
    scanf("%d%d",&x,&y);
    if(x>y)
        printf("%d %d\n",x,y);
    else
        printf("%d %d\n",y,x);
    return 0;
}
```

该程序运行结果与程序 3.3 相同。

【例 3.5】　编程实现：输入代表年份的 4 位数，判断其是否是闰年。

分析：可以用一个 int 型变量 year 存放输入的 4 位数。根据 3.2.2 节中关于判断闰年的逻辑表达式编写分支结构程序。

程序 3.5　输入代表年份的 4 位数，判断其是否为闰年。

```
#include <stdio.h>
int main(void)
{
    int year;
    printf("Input year = ");
    scanf("%d",&year);
    if((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))
        printf("%d 年是闰年。",year);
    else
        printf("%d 年不是闰年。",year);
    return 0;
}
```

程序运行结果如下。

```
该程序运行结果 1:    Input year = 2000
                      2000 年是闰年。
该程序运行结果 2:    Input year = 2016
                      2016 年是闰年。
该程序运行结果 3:    Input year = 1900
                      1900 年不是闰年。
```

3.3.3　多分支 if 语句

if 语句的前两种形式一般都用于两个分支的情况。当有多个分支选择时，可以采用 if-else-if 语句，其一般形式如下。

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

```
        语句 2;
        …
    else if(表达式 n - 1)
        语句 n - 1;
    else
        语句 n;
```

其语义是：依次判别表达式的值，当出现某个真时，则执行其对应的语句，然后跳转到整个 if 语句之后继续执行后续程序代码。如果所有的表达式均为假，则执行语句 n。其执行过程如图 3.8 所示。

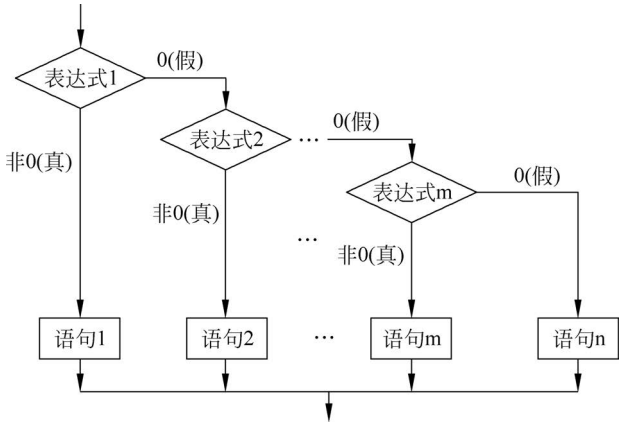


图 3.8　多分支 if 语句的执行过程

【例 3.6】 编程实现：从键盘输入一个英文字符，判断该字符是数字、字母或其他字符等三种类别中的哪一类别。

分析：10 个数字为'0'～'9'，大写字母为'A'～'Z'，小写字母为'a'～'z'，如果既不是数字也不是字母，那该字符就属于“其他字符”类别。容易看出分三种情况来分别处理。

程序 3.6　从键盘输入一个英文字符，判断该字符是数字、字母或其他字符等三种类别中的哪一类别。

```
#include <stdio.h>
int main(void)
{
    char ch;
    printf("Input a character: ");
    scanf("%c",&ch);
    if('0'<=ch && ch<='9')
        printf("%c 是数字。\n",ch);
    else if(('A'<=ch && ch<='Z') ||('a'<=ch && ch<='z'))
        printf("%c 是字母。\n",ch);
    else
        printf("%c 是其他字符。\n",ch);
    return 0;
}
```

程序运行结果如下。

```
该程序运行结果 1:    Input a character: c
                     c 是字母。
该程序运行结果 2:    Input a character: 8
                     8 是数字。
该程序运行结果 3:    Input a character: @
                     @是其他字符。
```

本例是一个多分支选择的问题,综合利用关系表达式和逻辑表达式的知识实现判断输入字符的类别。

【例 3.7】 编程实现:从键盘输入一个百分制的成绩(整数),将其转换为相应的五级计分制的成绩。转换规则:90～100 为 A,80～89 为 B,70～79 为 C,60～69 为 D,0～59 为 E。

分析:可以用一个 int 型变量 score 存放输入的百分制分数,用一个 char 型变量 grade 存放五级计分制成绩。显然,可以用多分支 if 语句分五种情况来处理。

程序 3.7 从键盘输入一个百分制的成绩(整数),将其转换为相应的五级计分制的成绩。

```
#include <stdio.h>
int main(void)
{
    int score;
    char grade;
    printf("Input score(0 - 100):");
    scanf("%d",&score);
    if(score >= 90)
        grade = 'A';
    else if(score >= 80)
        grade = 'B';
    else if(score >= 70)
        grade = 'C';
    else if(score >= 60)
        grade = 'D';
    else
        grade = 'E';
    printf("Grade = %c\n",grade);
    return 0;
}
```

程序运行结果如下。

```
Input score(0 - 100):78
Grade = C
```

本例程序中,在执行“if(score >= 80)”时,实际上隐含了条件“score < 90”,后面两次比较时也是类似情况。当然,“if(score >= 80)”也可以换成“if(score >= 80 && score < 90)”,只是没这个必要。

3.3.4　嵌套的 if 语句

在 if 语句中又包含一个或多个 if 语句称为 if 语句的嵌套。嵌套 if 语句的形式是多样

化的，没有固定形式。当语句中同时存在多个 if 语句时，else 总是与它前面最近的未配对的 if 配对。例如：

```
if(表达式)
    if(表达式 1)    /* 内嵌 if 语句 */
        语句 1;
    else
        if(表达式 2) 语句 2;
        else 语句 3;
else
    语句 4;
```

【例 3.8】 编程实现：从键盘输入两个数，代表直角坐标系某点的坐标，试问该点在第几象限（不考虑在坐标轴上的情况）？

分析：可以用两个 double 型变量 x、y 存放输入的横坐标和纵坐标，可以分四种情况来处理，也可以先根据 x 与 0 的大小关系分两种情况：第 1、4 象限和第 2、3 象限，然后将 y 与 0 比较，确定在哪一个象限。程序代码如下。

程序 3.8 从键盘输入两个数，代表直角坐标系某点的坐标，试问该点在第几象限？

```
#include <stdio.h>
int main(void)
{
    double x,y;
    int n;
    printf("Input x,y:");
    scanf("%lf,%lf",&x,&y);        /* x!=0,y!=0 */
    if(x>0)
        if(y>0)  n=1;
        else     n=4;
    else
        if(y>0)  n=2;
        else     n=3;
    printf("点(%g,%g)位于第%d象限。\n",x,y,n);
    return 0;
}
```

程序运行结果如下。

```
Input x,y: 4.5,-2.3
点(4.5,-2.3)位于第4象限。
```

例 3.5 中判断是否是闰年也可以换一个思路：如果不能被 4 整除，则不是闰年；否则，如果不能被 100 整除，则是闰年，否则，如果能被 400 整除则也是闰年，否则也不是闰年。这就可以用嵌套 if 语句完成程序设计，为了少写输出信息的语句，定义一个 int 型变量 leap 作为标志来表示是否为闰年。算法流程如图 3.9 所示。

程序 3.9 输入代表年份的四位数，判断其是否为闰年。

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

```
    int year,leap;
    printf("Input year = ");
    scanf("%d",&year);
    if(year % 4 != 0)
        leap = 0;
    else
        if(year % 100!= 0)     leap = 1;
        else
            if(year % 400 == 0) leap = 1;
            else leap = 0;
    if(leap!= 0)  printf("%d 年是闰年。",year);
    else     printf("%d 年不是闰年。",year);
    return 0;
}
```

该程序运行结果与程序 3.5 相同。

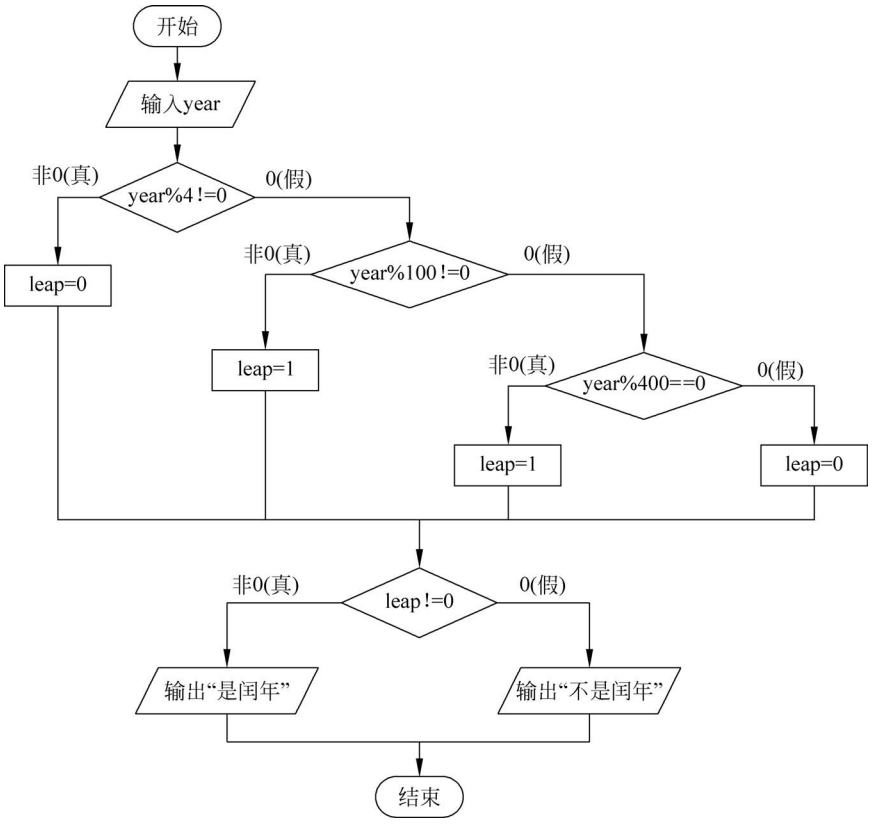


图 3.9　判断是否为闰年的流程图

3.3.5　在 if 语句中使用复合语句

在前面各种形式的 if 语句中，“语句 1”、“语句 2”或“语句 n”既可以是一条语句，也可以是多条语句组成的复合语句。使用复合语句时，需要将组成复合语句的多条语句放在花括号{}中，提供多个要执行的语句，其一般形式如下。

```
{
    第 1 条语句;
    第 2 条语句;
    …
    第 n 条语句;
}
```

【例 3.9】 编程实现：从键盘输入一个百分制的分数，如果小于 60，则输出“绩点为 0，需要补考。”，否则输出绩点（保留 1 位小数）。绩点计算公式：（分数－60）/10.0＋1.0。

分析：容易看出此题需要根据分数与 60 的大小关系分两种情况来处理，可以采用双分支 if 语句。当分数小于 60 时，绩点直接为 0，不需要使用公式计算绩点；否则，需要利用公式计算绩点并输出。注意绩点可能有小数，应采用双精度或单精度型变量，这里采用双精度型变量存放绩点。

程序 3.10　从键盘输入一个百分制的分数，输出绩点或补考信息。

```
#include <stdio.h>
int main(void)
{
    int score;
    double merit;
    printf("Input Score(0～100): ");
    scanf("%d",&score);
    if(score<60)
    {
        printf("绩点为 0,需要补考。");
    }
    else
    {
        merit=(score-60)/10.0+1.0;
        printf("绩点为 %0.1f。",merit);
    }
    return 0;
}
```

程序运行结果如下。

```
该程序运行结果 1:    Input Score(0～100): 50
                      绩点为 0,需要补考。
该程序运行结果 2:    Input Score(0～100): 73
                      绩点为 2.3。
```

注意：当 score＜60 不成立时（score≥60），需要执行两个语句，这两个语句必须放入{}中形成一个复合语句；将语句“printf("绩点为 0,需要补考。");”放入{}中也是可以的，而且在这里提倡这样做，以使得程序结构更清晰。

其实，只要可以使用单个语句的地方，都可以使用放在花括号{}中的复合语句。这也说明，复合语句中也可以嵌套另一个复合语句。

3.4　条件运算符和条件表达式

若有代码：

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

该代码的功能是使得 max 存放 x 和 y 中的较大者的值。C 语言中提供的条件运算符也可以实现相同的功能。上述代码用条件运算符可写为

```
max = (x > y)?x:y;
```

1. 条件运算符

条件运算符为“?:”“?”和“:”必须一起使用，它是 C 语言中唯一的三元运算符，即需要三个运算对象。适当使用条件运算符可以使程序更简洁和高效。

2. 条件表达式

条件表达式的一般形式：

```
表达式 1? 表达式 2 : 表达式 3
```

其求值规则：如果表达式 1 的值为真(非 0)，则以表达式 2 的值作为条件表达式的值，否则以表达式 3 的值作为条件表达式的值，它的执行过程如图 3.10 所示。

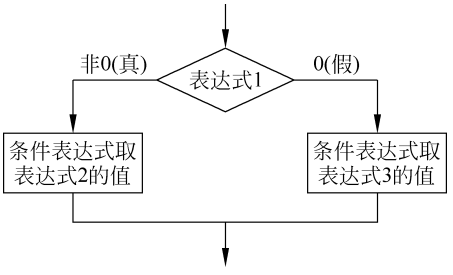


图 3.10　条件表达式的执行过程

3. 条件表达式的使用说明

(1) 条件运算符的优先级比逻辑运算符、关系运算符和算术运算符都低，但高于赋值运算符。

例如，条件表达式 max=(x>y)?x:y; 等价于 max=x>y?x:y; 因为，条件运算符比关系运算符“>”优先级低。

(2) 条件表达式中，表达式 2 和表达式 3 可以是数值表达式，也可以是赋值表达式或函数表达式。

例如，条件表达式 x>y?(x=1):(y=1)或 x>y?printf("%d",x):printf("%d",y)都是合法的。

(3) 条件表达式中,表达式 1 的类型可以与表达式 2 和表达式 3 的类型不同。

例如,假若 x=1,y=2,则条件表达式 x>y?'A':'B'是合法的,其值为'B'。

【例 3.10】 编程实现:从键盘输入一个字母,如果是小写字母则输出相应的大写字母,否则输出相应的小写字母。

分析:对同一个字母来说,大写字母的 ASCII 值比其小写字母的 ASCII 值小 32。可以利用条件表达式来实现大小字母的转换。

程序 3.11 从键盘输入一个字母,如果是小写字母则输出相应的大写字母,否则输出相应的小写字母。

```
#include <stdio.h>
int main(void)
{
    char ch;
    printf("Input a letter: ");
    scanf("%c",&ch);
    ch=(ch>='A'&& ch<='Z')?ch+32:ch-32;
    printf("%c\n",ch);
    return 0;
}
```

程序运行结果如下。

```
该程序运行结果 1:   Input a letter: d
                    D
该程序运行结果 2:   Input a letter: A
                    a
```

3.5 switch 语句

非嵌套的 if 语句只能提供两个分支进行选择,如果分支较多,使用嵌套的 if 语句则会语句层数多,程序冗长,可读性差。C 语言提供 switch 语句直接处理多分支结构。

3.5.1 switch 语句的一般形式

switch 语句的一般形式:

```
switch(表达式)
{
     case 常量表达式 1 :  语句段 1;   break;
     case 常量表达式 2 :  语句段 2;   break;
            …
     case 常量表达式 n :  语句段 n;    break;
     default :           语句段 n+1;  break;
}
```

执行过程如下。

(1) 计算 switch 后圆括号中表达式的值,记为 v。

(2) 计算“常量表达式 1”的值并与 v 比较，如果相等则执行“语句段 1”，否则计算“常量表达式 2”的值并与 v 比较，如果相等则执行“语句段 2”，否则计算常量表达式 3 并与 v 比较……逐个计算并判断。

(3) 如果所有的“常量表达式”的值都不等于 v，则执行 default 后的“语句段 n+1”。

break 语句的作用是结束 switch 语句，执行后面的其他语句。如果执行完“语句段 1”后没有 break 语句，则接下来执行“语句段 2”。所以，通常情况下，case 子句后面会加上 break 语句。

【例 3.11】 编程实现：从键盘依次输入一个数、一个算术运算符、另一个数，如果将其看成一个算术运算式，试输出其值。运算符是“+”“-”“*”“/”四者之一。

分析：依题意可知需要分 4 种情况计算所输入表达式的值。应该采用多分支结构，为了程序直观采用 switch 语句。

程序 3.12　编程实现四则运算。

```
#include <stdio.h>
int main(void)
{
    char op;
    double x,y,z;
    printf("Input:");
    scanf("%lf%c%lf",&x,&op,&y); /* 中间不能有空格 */
    switch(op)
    {
        case '+':z = x + y;break;
        case '-':z = x - y;break;
        case '*':z = x * y;break;
        default: z = x/y;break;
    }
    printf("%g%c%g = %g\n",x,op,y,z);
    return 0;
}
```

程序运行结果如下。

```
该程序运行结果 1:    Input: 5.4 * 2.1
                      5.4 * 2.1 = 11.34
该程序运行结果 2:    Input: 12/5
                      12/5 = 2.4
```

运行上述程序，如果输入“9#4”，会输出什么结果？按照 switch 语句的执行过程，在 case 子句中没有“#”常量表达式，所以执行 default 后的语句，输出的是 9#4=2.25。

3.5.2　switch 语句的使用说明

(1) switch 后面括号中的“表达式”，其值的类型只能是整数类型(包括字符型和枚举型)。

(2) 可以没有 default 子句。此时，如果“表达式”的值与“常量表达式 1”“常量表达式 2”……“常量表达式 n”都不相等，则不执行任何语句，流程转到 switch 语句后面的语句。

(3) default 子句不一定出现在最后，case 子句和 default 子句的出现次序不影响执行

结果。

(4)“常量表达式1”“常量表达式2”……“常量表达式n”的值必须互不相同,否则会出现编译错误。

(5) case子句中的语句段包含多个语句时,不必用花括号括起来,当然,加上花括号也可以。

(6) 多个case可以共用一个语句段。

(7) switch语句实现的分支结构程序,往往容易用if语句或嵌套if语句来实现。switch语句中也允许出现多种语句,也包括嵌套的switch语句。

【例3.12】 编程实现:从键盘输入年号和月份,输出该月的天数。

分析:一年12个月中,2月的天数与是否是闰年有关,因此需要先判断是否是闰年,其他月份的天数只有30和31两种情况,所以共有三种情况,可以采用switch语句实现多分支结构程序设计。

程序3.13 从键盘输入年号和月份,输出该月的天数。

```
#include <stdio.h>
int main(void)
{
    int year,month,days,leap;
    printf("Input year and month:");
    scanf("%d%d",&year,&month);
    if((year%4==0&&year%100!=0)||(year%400==0))     leap=1;
    else  leap=0;
    switch(month)
    {
        case 1: case 3: case 5:
        case 7: case 8:
        case 10: case 12: days=31;break;
        case 2:           days=28+leap; break;
        case 4: case 6:
        case 9: case 11:   days=30;break;
        default: printf("Month input error.\n");
    }
    printf("%d年%d月有%d天。\n",year,month,days);
    return 0;
}
```

程序运行结果如下。

```
该程序运行结果1:     Input year and month:2016 2
                     2016年2月有29天。
该程序运行结果2:     Input year and month:2018 7
                     2018年7月有31天。
```

【例3.13】 用switch语句编程实现:从键盘输入一个百分制的成绩(整数),将其转换为相应的五级计分制的成绩。转换规则:90～100为A,80～89为B,70～79为C,60～69为D,0～59为E。

分析:可以用一个int型变量score存放输入的百分制分数,用一个char型变量grade存放五级计分制成绩。根据成绩转换规则按分数处在不同区间对应不同等级,如果只考虑

分数的十位数字，就容易联想到使用 switch 语句完成多分支程序设计。

程序 3.14　用 switch 语句编程实现成绩由百分制转换为五级计分值。

```
#include <stdio.h>
int main(void)
{
    int score;
    char grade;
    printf("Input score(0 - 100):");
    scanf("%d",&score);
    switch(score/10)
    {
        case 10:
        case 9:  grade = 'A'; break;
        case 8:  grade = 'B'; break;
        case 7:  grade = 'C'; break;
        case 6:  grade = 'D'; break;
        default: grade = 'E';
    }
    printf("Grade = %c\n",grade);
    return 0;
}
```

程序运行结果如下。

```
该程序运行结果 1:    Input score(0 - 100):100
                      Grade = A
该程序运行结果 2:    Input score(0 - 100):55
                      Grade = E
```

3.6　综合应用实例——猜数小游戏

“猜数小游戏”是一个能和计算机互动的小游戏，虽然目前暂未学习循环程序设计，但是可以先体会如何通过程序与计算机实现交互，程序根据玩家输入的信息执行不同的流程和输出。程序随机产生一个 1 到 100 之间的数，用户有 7 次机会猜测这是一个什么数，对于每次输入的数程序经过判断给出提示信息：“You are smart!”“Too big”或“Too small”。如果 7 次都没有猜对，游戏结束。

程序 3.15　猜数小游戏。

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(void)
{
    int count = 0, flag = 0, target, number;
    srand(time(0));
    target = rand() % 100 + 1;
    while(count < 7){
```

```
        printf("Enter the number you guess: ");
        scanf(" %d", &number);
        count++;
        if(number == target) {
            printf("You are smart!\n");
            flag = 1;
            break;
        }
        else
            if(number > target )
                printf("Too big\n");
            else
                printf("Too small\n");
    }
    if (flag == 0)
        printf("Game Over!\n");
    return 0;
}
```

程序运行结果 1 如下。

```
Enter the number you guess: 50
Too big
Enter the number you guess: 25
Too big
Enter the number you guess: 15
Too small
Enter the number you guess: 20
Too small
Enter the number you guess: 23
Too big
Enter the number you guess: 22
You are smart!
```

程序运行结果 2 如下。

```
Enter the number you guess: 50
Too samll
Enter the number you guess: 80
Too big
Enter the number you guess: 68
Too big
Enter the number you guess: 60
Too big
Enter the number you guess: 55
Too small
Enter the number you guess: 56
Too small
Enter the number you guess: 57
Too small
Game Over!
```

3.7　工程案例分析——空调离合器状态仲裁

下面的代码是判断变排量空调的工作状态，当前汽车都有很高的油耗指标，变排量空调一是可以为节油作出贡献同时客户感觉更为舒适。变排量空调工作总体来说有两层逻辑，一是空调离合器结合，压缩机开始工作；二是离合器结合状态下压缩机的排量进行变化，得到设定的温度。离合器结合与否、该迅速还是缓慢结合或者反之离合器脱开，关系到动力系统运行是否平顺等问题，读者可以从下面的状态仲裁代码进行初步了解。

```
switch( acc_evdc_clth_mode )
  {
    case 1U:                                        /* 离合器断开 */
    {
      if (accflg != FALSE)                          /* 如果发现有空调运行请求信号 */
      {
        acc_evdc_clth_mode = 2U;                    /* 离合器状态切换为缓冲结合模式 */
      }
      else
      {
        acc_evdc_clth_mode = 1U;                    /* 离合器状态保持为断开模式 */
      }
      break;
    }
    case 2U:                                        /* 离合器缓冲结合 */
    {
      if (acc_instant_off_flg != FALSE)             /* 收到立即切断空调的请求 */
      {
        acc_evdc_clth_mode = 1U;                    /* 回到离合器断开模式 */
      }
      else if (acrqst == FALSE)                     /* 驾驶员请求空调关闭 */
      {
        acc_evdc_clth_mode = 4U;                    /* 离合器状态切换为缓冲切断模式 */
      }
      else if (acc_evdc_rmp_pct >= (1.0F - p_epsilon_p))  /* 空调排量开始增加 */
      {
        acc_evdc_clth_mode = 3U;                    /* 离合器已结合状态 */
      }
      else
      {
        acc_evdc_clth_mode = 2U;                    /* 离合器状态切换为缓冲结合模式 */
      }
      break;
    }
    case 3U:                                        /* 离合器已结合状态 */
    {
      if (acc_instant_off_flg != FALSE)             /* 收到立即切断空调的请求 */
      {
        acc_evdc_clth_mode = 1U;                    /* 离合器状态切换为断开模式 */
      }
      else if (acrqst == FALSE)                     /* 驾驶员请求关闭空调 */
```

```
        {
          acc_evdc_clth_mode = 4U;          /* 离合器状态切换为缓冲断开模式 */
        }
        else
        {
          acc_evdc_clth_mode = 3U;          /* 保持离合器已结合状态 */
        }
        break;
      }
      case 4U:                              /* 离合器状态为缓冲断开 */
      {
        if (acc_instant_off_flg != FALSE)   /* 收到立即切断空调信号 */
        {
          acc_evdc_clth_mode = 1U;          /* 回到离合器断开模式 */
        }
        else if (acrqst != FALSE)           /* 驾驶员请求打开空调 */
        {
          acc_evdc_clth_mode = 2U;          /* 离合器状态切换为缓冲结合模式 */
        }
        /* 在缓冲切断模式中停留的时间超过标定值 acc_evdc_clutch_off_tm */
        else if (acc_evdc_clutch_off_tmr >= acc_evdc_clutch_off_tm)
        {
          acc_evdc_clth_mode = 1U;          /* 回到离合器切断状态 */
        }
        else
        {
          acc_evdc_clth_mode = 4U;          /* 保持在离合器缓冲切断模式 */
        }
        break;
      }
    }
  /* 变排量的空调可以线性地改变输出,从而让整个系统受力平顺地过渡,不会造成车辆振动,而如
  果遇到需要保护硬件等紧急情况时,需要立即切断空调 */
```

3.8　小结

本章对关系表达式、逻辑表达式和条件表达式做了详细的介绍，每种表达式都有常见用法的举例，以加深读者对上述表达式的认识和理解，掌握使用适当的表达式描述实际问题的方法。本章还详细介绍了 C 语言中 if 语句和 switch 语句的常见形式和基本用法，通过实例讲解了使用 if 语句和 switch 语句进行分支结构程序设计的基本思路。

人生之路就像分支结构，在很多路口都延伸了不同分支，每个分支对应着不一样的未来，走向哪个分支需要人们做出选择，选择不同，结果迥异。“鱼和熊掌不可兼得”，选择是人这一生都需要面对的问题，树立正确的人生观和价值观，学会取舍，“博学之，审问之，慎思之，明辨之，笃行之”，成就更好的自己。

本章习题


在线测试

知识点强化训练

单选题

1. 逻辑运算符两侧的运算对象(　　)。

A. 只能是 0 或 1　　B. 只能是整型或字符型数据

C. 只能是 0 或非 0 正数　　D. 可以是任何类型的数值

2. 为了避免嵌套 if 语句的二义性,C 语言规定 else 总是与(　　)配对。

A. 缩排位置相同的 if　　B. 在其之前未配对的 if

C. 在其之前未配对的最近的 if　　D. 同一行上的 if

3. 设 x=3,y=4,z=5,则表达式“x+y>z && y==z”的值为(　　)。

A. 0　　B. 1　　C. -1　　D. 非 0

4. 设 x=3,y=4,z=5,则表达式“!x || y>z-3 && 2”的值为(　　)。

A. 0　　B. 1　　C. -1　　D. 非 0

5. 能正确表示逻辑关系“10≤x≤99”的 C 语言表达式是(　　)。

A. 10<=x<=99　　B. 10<=x && x<=99

C. 10<=x || x<=99　　D. x<=99 || x>=10

6. 如果 int a=7,b=9; 则条件表达式"a>b?a:b"的值是(　　)。

A. 7　　B. 9　　C. 0　　D. 1

填空题

1. 执行以下程序的输出结果是_____。

```
#include<stdio.h>
int main(void)
{
    int x=3,y=2,z=4;
    if(x>y)
    {
        if(x>z)  printf("%d\n",x);
        else  printf("%d\n",y);
    }
    return 0;
}
```

2. 执行以下程序的输出结果是_____。

```
#include<stdio.h>
int main(void)
{
    int x=3,y=2,z=0,u=10;
    if(!x)  u=u-5;
    else    if(y)
```

```
            if(!z)   u = u + 10;
            else     u = u + 20;
    printf("u = %d\n",u);
    return 0;
}
```

3. 执行以下程序,输入为 2 时的输出结果是_____。

```
#include <stdio.h>
int main(void)
{
    int k;
    scanf("%d",&k);
    switch(k)
    {
        case 1: printf("%d",k + 1);
        case 2: printf("%d",k + 2);
        case 3: printf("%d",k + 3);   break;
        default: printf("%d\n",k);
    }
    return 0;
}
```

4. 执行以下程序的输出结果是_____。

```
#include <stdio.h>
int main(void)
{
    int a = 1,b = 1,c = 5,d = 6;
    if(c > 0)
    {
        if(c > d)   b = a + b;
        else  if(c == d)   b = a;
              else   b = 2 + a;
    }
    else   b = 2 * a;
    printf("b = %d\n",b);
    return 0;
}
```

5. 执行以下程序的输出结果是_____。

```
#include <stdio.h>
int main(void)
{
    int a = 1, b = 1;
    switch(a)
    {
        case 1:
           switch(b)
           {
               case 0: printf("A");break;
               case 1: printf("B");
```

```
            }
        case 2: printf("C");
    }
    return 0;
}
```

编程训练

1. 从键盘输入 3 个同学的成绩，输出其中最高分。

2. 从键盘输入 3 个同学的成绩，按从最高分到最低分的顺序输出。

3. 从键盘依次输入一元二次方程的二次项系数、一次项系数和常数项，输出其实根。

4. 从键盘输入 3 个数，问这 3 个数能否作为三角形的边长。若能构成三角形，则输出周长，否则输出“不可能”。

5. 某城市为了鼓励节约用电，对居民用电作如下规定：若一户居民每月用电量不超过 200 度，则按每度 0.5 元收费；若超过 200 度但不超过 350 度，则其中 200 度按每度 0.5 元收费，剩余部分按每度 0.7 元收费；若超过 350 度，则其中 350 度按前述价格收费，剩余部分按每度 0.9 元收费。编程实现：输入某户居民的月用电量，输出应缴纳的电费(保留 2 位小数)。

6. 某市的出租车计价规则如下：行程不超过 3 千米，收起步价 8 元；超过 3 千米而没超过 15 千米部分每千米收费 1.8 元；超过 15 千米部分每千米 2.2 元；不足 0.5 千米按 0.5 千米收费。编程实现：输入千米数，输出应付车费(保留 2 位小数)。