第3章 选择结构及其应用

在现实生活中，人们经常需要根据不同的条件做出选择，而在计算机程序设计过程中，也可通过某一个或若干条件的判断，有选择地执行特定语句，这就是选择结构。选择结构是一种使程序具有判断能力的程序结构。

本章主要介绍在C语言中实现选择结构的程序设计方法，选择结构主要通过if语句或switch语句来实现。

学习目标：

- 了解关系运算符和关系表达式。
- 了解逻辑运算符和逻辑表达式。
- 掌握if单分支语句、if-else双分支语句和嵌套的if语句的使用。
- 掌握switch语句的使用。
- 区分两种if语句与switch语句。
- 学会利用选择结构解决一般应用问题。

3.1 关系运算符和关系表达式

3.1.1 关系运算符

在程序中经常需要比较两个量的大小关系，以决定程序下一步的工作。比较两个量的运算符称为关系运算符，所谓关系运算，实际上就是比较运算，即将两个操作数进行比较并产生运算结果0(假)或1(真)。C语言提供的关系运算符有6种，如表3.1所示。

表3.1 关系运算符

运 算 符	功 能
<	小于
<=	小于或等于
>	大于
>=	大于或等于
==	等于
!=	不等于

说明：

(1) C 语言中的小于或等于、大于或等于、等于、不等于运算符(<=、>=、==、!=)的表示与数学中的表示(≤、≥、=、≠)不同。

(2) 在以上 6 种关系运算符中，前 4 种(<、<=、>、>=)的优先级相同，后两种(==、!=)的优先级相同，前 4 种的优先级高于后两种。例如，a>=b!=b<=3 等价于(a>=b)!=(b<=3)。

(3) 关系运算符的结合性为从左到右。

(4) C 语言中“==”是关系运算符，用来判断两个数是否相等，请读者注意与赋值运算符“=”的区别，例如，x==3 用于判断 x 的值是否为 3，x=3 是使 x 的值为 3。

3.1.2 关系表达式

关系表达式是指用关系运算符将两个数(或表达式)连接起来进行关系运算的式子。例如，以下均是合法的关系表达式。

```
3<2， a>b， a<b+c， c>b==a， a=b>c
```

关系表达式的结果是逻辑值，即真值或假值，其中真值为 1，假值为 0，真值表示指定的关系成立，假值则表示指定的关系不正确。例如，若 a=1，b=2，c=3，则：

- 关系表达式“c<b”的结果为假值，因为 3 大于 2，所以该关系不成立。
- 关系表达式“a>b”的结果为假值，因为 a=1，b=2，1 小于 2，所以该关系不成立。
- 关系表达式“a<b+c”的结果为真值，因为 a=1，b+c=5，1 小于 5，所以该关系式成立。
- 关系表达式“c>b==a”的结果为真值，因为 b=2，c=3，3 大于 2，所以关系式 c>b 成立，结果为真值，真值为 1，等于 a 的值，所以表达式 c>b==a 的值为 1。
- 关系表达式“a=b>c”的结果为假值，因为 b=2，c=3，2 小于 3，所以关系式 b>c 不成立，结果为假值，假值为 0，所以赋值后 a 的值为 0。注意：“>”运算符比赋值运算符“=”的优先级要高。

3.1.3 优先级和结合性

关系运算符的结合性都是自左向右的。使用关系运算符时常常会判断两个表达式的关系，但是由于运算符存在着优先级的问题，因此如果处理不小心就会出现错误。如进行这样的判断操作：先对一个变量进行赋值，然后判断这个赋值的变量是否等于一个常数，表达式如下：

```
Number=Num==10
```

因为“==”运算符比“=”的优先级要高，所以 Num==10 的判断操作会在赋值之前进行，变量 Number 得到的就是关系表达式的真值或者假值，这样并不会按照之前的意愿执行。所以要用到括号运算符，其优先级具有最高性，因此可以使用括号来表示要优先

计算的表达式，例如：

```
(Number=Num)==10
```

这种写法比较清楚，不会产生混淆，没有人会对代码的含义产生误解。由于这种写法格式比较精确简洁，因此建议使用这种方式。

3.2 逻辑运算符和逻辑表达式

C 语言中，对参与逻辑运算的所有数值，都转换为逻辑“真”或逻辑“假”后才参与逻辑运算，如果参与逻辑运算的数值为 0，则把它作为逻辑“假”处理，而所有非 0 的数值都作为逻辑“真”处理。

3.2.1 逻辑运算符

有的时候，要求一些关系同时成立，而有的时候，可能只要求其中的某一个关系成立就可以，这时，需要用到逻辑运算符。C 语言中有 3 种逻辑运算符：逻辑与(&&)、逻辑或(||)、逻辑非(!)。

逻辑运算符及其对应的功能说明如表 3.2 所示。

表 3.2 逻辑运算符

运算符	功　能
&&	逻辑与，双目运算符，左右两个数都为“真”时才为“真”，否则为“假”
||	逻辑或，双目运算符，左右两个数都为“假”时才为“假”，否则为“真”
!	逻辑非，单目运算符，改变当前数的值，“真”变“假”，“假”变“真”

逻辑运算的真值表如表 3.3 所示。

表 3.3 逻辑运算真值表

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

3.2.2 逻辑表达式

逻辑表达式是由逻辑运算符将逻辑量连接起来构成的式子。逻辑运算符两侧的运算

对象可以是任何类型的数据，但运算结果一定是整型值，并且只有两个值：1 和 0，分别表示“真”和“假”。例如：

(1) 若 a=2，则逻辑表达式！a 的值为 0，因为 a 的值为非 0，逻辑值为“真”，对它进行“逻辑非”运算，得“假”，“假”以 0 代表。

(2) 若 a=2，b=3，则逻辑表达式 a&&b 的值为 1，因为 a 和 b 均非 0，逻辑值为“真”，所以进行“逻辑与”运算的值也为“真”，“真”以 1 代表。

(3) 若 a=2，b=3，则逻辑表达式 a||b 的值为 1，因为 a 和 b 均非 0，逻辑值为“真”，所以进行“逻辑或”运算的值也为“真”，“真”以 1 代表。

(4) 若 a=2，b=3，则逻辑表达式！a||b 的值为 1，因为虽然！a 的值为 0，但是 b 非 0，逻辑值为“真”，所以进行“逻辑或”运算的值也为“真”，“真”以 1 代表。

说明：

(1) 对于 a&&b，只有 a 为真(非 0)时，才需要判断 b 的值，如果 a 为假，就不必判断 b 的值。也就是说，对于 && 运算符，只有 a≠0，才继续进行其右面的运算。

(2) 对于 a||b，只要 a 为真(非 0)，就不必判断 b 的值，只有 a 为假时，才判断 b 的值。也就是说，对于|| 运算符，只有 a=0，才继续进行其右面的运算。

(3) 2<a<3 在 C 语言中的表示为(2<a)&&(a<3)。

3.2.3 优先级和结合性

三种运算符的优先级由高到低依次为：!、&&、||。

逻辑运算符中的“&&”和“||”的结合性为从左到右，“!”的结合性为从右到左。

关系运算符的优先级低于算术运算符，逻辑运算符中的“&&”和“||”的优先级低于关系运算符，“!”的优先级高于算术运算符。

【例 3-1】 逻辑运算符的应用。

【问题分析】 在本示例中，使用逻辑运算符构成表达式，通过输出函数显示表达式的结果，根据结果分析表达式中逻辑运算符的计算过程。

【程序代码】

```
#include<stdio.h>
int main()
{
    int num1,num2;                          //声明变量
    num1=10;                                //给变量 num1 赋值 10
    num2=0;                                 //给变量 num2 赋值 0
    printf("1 is true,0 is false\n");       //显示提示信息
    //显示逻辑与表达式的结果
    printf("5<num1&&num2 is %d\n",5<num1&&num2);
    //显示逻辑或表达式的结果
    printf("5<num1||num2 is %d\n",5<num1||num2);
    num2=!!num1;                            //得到 num1 的逻辑值
```

```
    printf("num2 is %d\n",num2);              //输出逻辑值
    return 0;
}
```

【运行结果】

程序运行结果如图 3.1 所示。

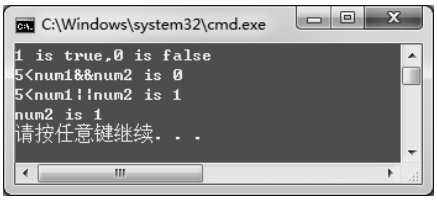

图 3.1　例 3-1 程序运行结果

【代码解析】

(1) 在程序中,首先声明两个变量用来进行下面的计算。给变量赋值,num1 的值为 10,num2 的值为 0。

(2) 输出结果说明,显示为 1 表示真值,0 表示假值。在第二个和第三个 printf()函数中,先进行表达式的运算,再将结果输出。分析表达式 5＜num1&&num2,由于“＜”运算符的优先级高于“&&”运算符,因此先执行关系判断,再进行与运算。num1 的值为 10,num2 的值为 0,这个表达式的含义是数值 5 小于 num1 的同时 num2 为非零,很明显关系不成立,因此表达式返回的是假值。而表达式 5＜num1||num2 的含义是 5 小于 num1 或者 num2 为非零,此时表达式成立,返回值为真值。

(3) !! num1 表示的是将 num1 进行两次单目逻辑非运算,得到的是逻辑值,因为 num1 的值是 10,所以逻辑值为 1。

3.3　if 语 句

if 语句是条件选择语句,它先对给定条件进行判断,根据判定的结果(真或假)决定要执行的语句。if 语句有 if 分支、if-else 分支和嵌套的 if 语句 3 种形式,下面介绍每种 if 语句的具体使用方式。

3.3.1　if 分支

if 分支是最简单的条件语句,if 分支语句的一般形式如下:

```
if(表达式)
    语句 1;
```

其中：表达式一般为逻辑表达式或关系表达式。语句 1 可以是一条简单的语句或多条语句，当为多条语句时，需要用大括号“{}”将这些语句括起来，构成复合语句。if 分支语句的执行过程是：当表达式的值为真(非 0)时，执行语句 1，否则直接执行 if 语句下面的语句。其执行流程图如图 3.2 所示。

【例 3-2】 从键盘输入一个整数，若输入是 3 的倍数，则显示“OK!”，否则什么也不显示。

【问题分析】

根据题意，需要使用 if 语句对输入的整数进行判断，解题的关键是 3 的倍数的表达式“num%3==0”的建立，如果是 3 的倍数，输出“OK!”，不是 3 的倍数时，程序无输出。

解决该问题的算法流程图如图 3.3 所示。

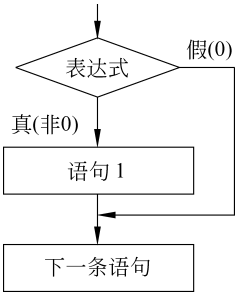

图 3.2 if 分支的执行流程图

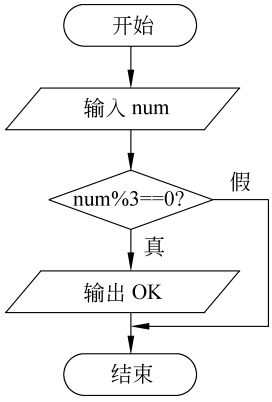

图 3.3 例 3-2 的流程图

【程序代码】

```
#include <stdio.h>
int main()
{
    int num;                          //定义整型变量 num
    printf("Please enter num:");      //输出屏幕提示语
    scanf("%d",&num);                 //从键盘输入 num 的值
    if(num%3==0)                      //判断 num 是否为 3 的整数倍
        printf("OK!\n");              //是则输出提示信息 OK!
    return 0;
}
```

【运行结果】

程序运行结果如图 3.4 所示。

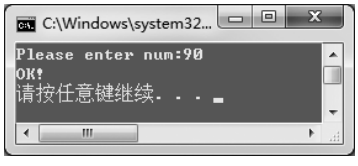

图 3.4 例 3-2 程序运行结果

【代码解析】

(1) 本示例中 num 为整型。

(2) 使用输入函数 scanf 实现给 num 赋值。

(3) 使用 if 语句进行条件判断,因为 90 是 3 的倍数,所以屏幕输出“OK!”,当输入的整数不是 3 的倍数时,程序无输出。

【例 3-3】 从键盘输入两个整数,输出这两个数中较大的数。

【问题分析】

根据题意,本题要实现的功能是比较两个整数的大小,这两个整数由用户从键盘输入,然后将其中较大的数输出显示。

解决该问题的算法流程图如图 3.5 所示。

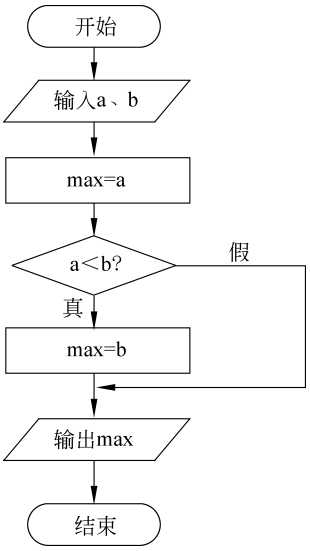

图 3.5 例 3-3 的流程图

【程序代码】

```
#include <stdio.h>
int main()
{
    int a,b,max;                                  //定义整型变量 a、b、max
    printf("Please enter two integers:");         //输出屏幕提示语
    scanf("%d%d",&a,&b);                          //输入 a、b 的值
    max=a;                                        //假设 a 是较大的数,并赋值给 max
    if(a<b)                                       //使用 if 语句进行判断
        max=b;                                    //如果 a<b 的值为真,则将 b 赋值给 max
    printf("The bigger integer is:%d\n",max);     //输出 max 的值
    return 0;
}
```

【运行结果】

程序运行结果如图 3.6 所示。

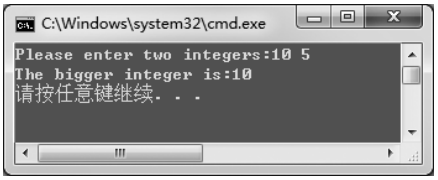

图 3.6　例 3-3 程序运行结果

【代码解析】

(1) 本实例中 a、b、max 均为整型。

(2) 使用输入函数 scanf 实现给 a、b 赋值。

(3) 首先假设 a 是较大的数,将 a 的值赋给 max,然后使用 if 语句进行条件判断,如果 a 小于 b,则 b 为较大的数,将 b 的值赋给 max。

(4) 使用输出函数 printf 输出 max 的值。

在使用 if 语句时,应注意以下几点:

(1) if 后面的表达式必须用小括号括起来。

(2) if 后面的表达式可以为关系表达式、逻辑表达式、算术表达式等。例如:

```
if(a>=1&&a<=10) printf("x=%d,y=%d",x,3*x-1);
if(1) printf("OK!");                          //条件永远为真
if(!a) printf("input error!");
```

(3) 在表达式中一定要区分赋值运算符“=”和关系运算符“==”。例如:

```
y=10;
if(x==3)  y=2*x;
```

当 x 值为 3,表达式 x==3 的值为真,执行语句 y=2*x,则 y=6;当 x 取其他值时,表达式 x==3 的值为假,不执行语句 y=2*x,则 y=10。再如:

```
y=10;
if (x=3) y=2*x;
```

不管 x 原来取值多少,执行完 if 语句后,x 值为 3,为非 0 值,则条件永远为真,执行语句 y=2*x,则 y=6。

3.3.2　if-else 分支

if 分支语句只允许在条件为真时指定要执行的语句,而 if-else 分支还可在条件为假时指定要执行的语句。if-else 分支语句的一般形式如下:

```
if(表达式)
    语句 1;
```

```
else
    语句 2;
```

if-else 分支语句的执行过程是：当表达式为真(非 0)时，执行语句 1，否则执行语句 2，其执行流程图如图 3.7 所示。

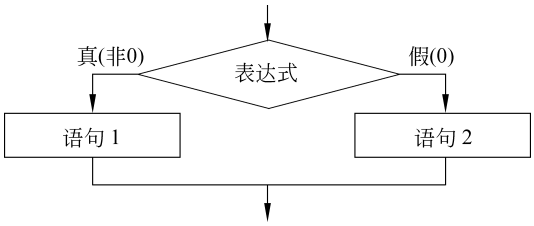

图 3.7　if-else 分支的流程图

【例 3-4】 编程计算下列分段函数的值并输出。

$$y=\begin{cases}2x-1 & x<0\\ x & x\geqslant 0\end{cases}$$

【问题分析】

本题使用 if-else 语句判断用户输入的数值，若输入 x 的值小于 0 表示条件为真，y＝2x－1；若输入 x 的值大于等于 0 表示条件为假，y＝x；最后输出 y 的值。

解决该问题的算法流程图如图 3.8 所示。

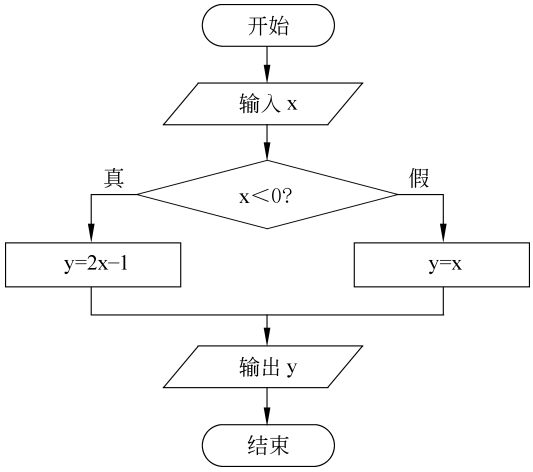

图 3.8　例 3-4 的流程图

【程序代码】

```
#include <stdio.h>
int main()
{
    int x,y;                                    //定义整型变量 x、y
```

```
    printf("Please enter x:");              //输出屏幕提示语
    scanf("%d",&x);                         // 输入 x 的值
    if(x<0)                                 //使用 if 语句进行判断
        y=2*x-1;                            //如果 x<0 的值为真,则 y=2x-1
    else
        y=x;                                //如果 x<0 的值为假,则 y=x
    printf("y=%d\n",y);                     //输出 y 的值
    return 0;
}
```

【运行结果】

程序运行结果如图 3.9 所示。

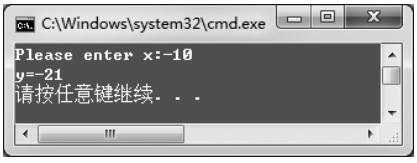

图 3.9　例 3-4 程序运行结果

【代码解析】

(1) 本示例中 x、y 均为整型。

(2) 使用输入函数 scanf 获得任意值并赋给 x。

(3) 使用 if 语句进行条件判断,如果 x 小于 0,则 y＝2x-1,否则 y＝x。

(4) 使用输出函数 printf 输出 y 的值。

【例 3-5】 从键盘输入两个整数,输出这两个数中较大的数。

【问题分析】

本示例实现的功能与例 3-3 相同,都是求两个数中较大的数,不同之处在于本例使用 if-else 分支来实现。

解决该问题的算法流程图如图 3.10 所示。

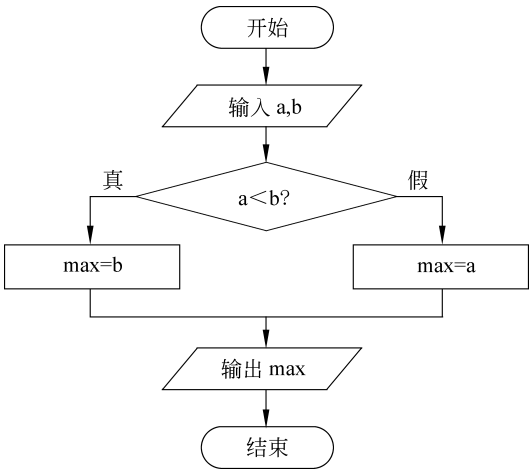

图 3.10　例 3-5 的流程图

【程序代码】

```
#include <stdio.h>
int main()
{
    int a,b,max;                                    //定义整型变量 a、b、max
    printf("Please enter a and b:");                //输出屏幕提示语
    scanf("%d,%d",&a,&b);                           //输入 a、b 的值
    if(a<b)                                         //使用 if-else 语句进行判断
        max=b;                                      //如果 a<b 为真,则 b 为较大的数
    else
        max=a;                                      //如果 a<b 为假,则 a 为较大的数
    printf("The bigger integer is:%d\n",max);       //输出最大值
    return 0;
}
```

【运行结果】

程序运行结果如图 3.11 所示。

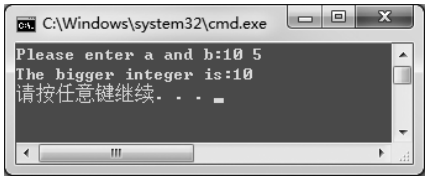

图 3.11　例 3-5 程序运行结果

【代码解析】

(1) 本示例中 a、b、max 均为整型。

(2) 使用输入函数 scanf 获得两个任意值并赋给 a 和 b。

(3) 使用 if-else 语句进行条件判断,如果 a 小于 b 为真,则 b 为较大的数,将 b 的值赋给 max;如果 a 小于 b 为假,则 a 为较大的数,将 a 的值赋给 max。

(4) 使用输出函数 printf 输出 max 的值。

【例 3-6】 从键盘上输入 a、b、c 的值,对读入的 a、b、c 的值进行判断,如果 3 个值均大于 0 而且符合任意两个之和大于第 3 个,则计算以这三个值为边的三角形的面积并输出结果,否则,输出提示信息"Error input!"。

【问题分析】

实现本示例之前必须知道三角形的一些相关知识,例如,如何判断输入的三边是否能组成三角形,以及三角形面积的求法等。从键盘中输入三条边的边长后,只需判断这三条边的边长是否大于 0 并且任意两边之和是否大于第三边,如果满足条件,可以构成三角形,然后计算三角形的面积。

解决该问题的算法流程图如图 3.12 所示。

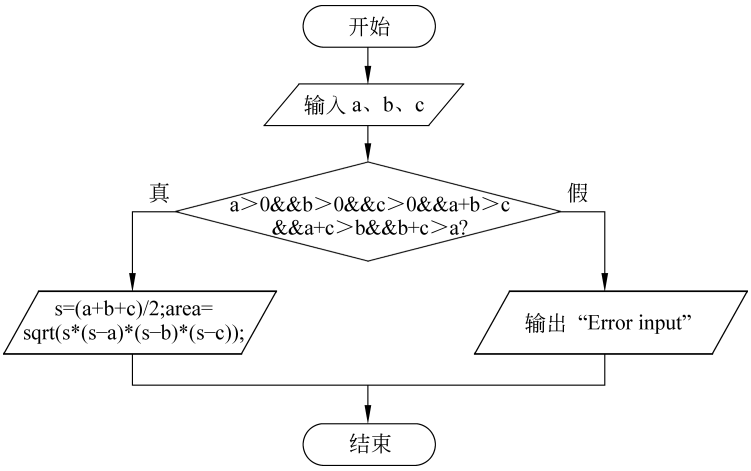

图 3.12　例 3-6 的流程图

【程序代码】

```
#include <stdio.h>
#include <math.h>                          //引用头文件,math.h 中定义了各种数学函数
int main()
{
    double a,b,c,s,area;                   //定义 5 个双精度浮点型变量
    printf("Please enter a,b,c:");         //输出屏幕提示语
    scanf("%lf%lf%lf",&a,&b,&c);           //输入三条边的值
    //判断 3 条边均大于 0,并且任意两边之和大于第 3 边
    if(a>0&&b>0&&c>0&&a+b>c&&a+c>b&&b+c>a)
    {
        s=(a+b+c)/2;
        area=sqrt(s*(s-a)*(s-b)*(s-c));    //计算面积
        printf("area=%lf\n",area);         //输出面积
    }
    else //如果两边之和小于第 3 边或有的边的值小于 0,输出错误提示
        printf("Error input!\n");
    return 0;
}
```

【运行结果】

程序运行结果如图 3.13 所示。

图 3.13　例 3-6 程序运行结果

【代码解析】

(1) 程序中用户输入三个数,然后对三个数进行判断,解题的关键是条件表达式的建立。假设输入的三个数为 a、b、c, 根据题目要求三个数值均大于 0,即 a>0,b>0,c>0,并且任意两个数值的和大于第三个数,即 a+b>c,b+c>a,c+a>b。

(2) 当需要表达多个条件同时满足的时候,这些子条件间以"&&"运算符连接,本示例中的 6 个小条件同时满足才能保证 a、b、c 能构成一个三角形。

(3) 本示例中 else 分支不能省略,如果省略了这一分支,则在不能构成三角形时,不计算也不输出,此时程序没有任何输出结果。

3.3.3 嵌套的 if 语句

简单的 if 语句只能通过给定条件的判断决定执行给出的两种操作之一,而不能从多种操作中进行选择,此时可通过 if 语句的嵌套来解决多分支选择问题。if 语句中又包含一个或多个 if 语句时,称为 if 语句的嵌套。常用的 if 语句嵌套有以下两种形式。

(1) 形式一:

```
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;
```

此种结构的流程图如图 3.14 所示。

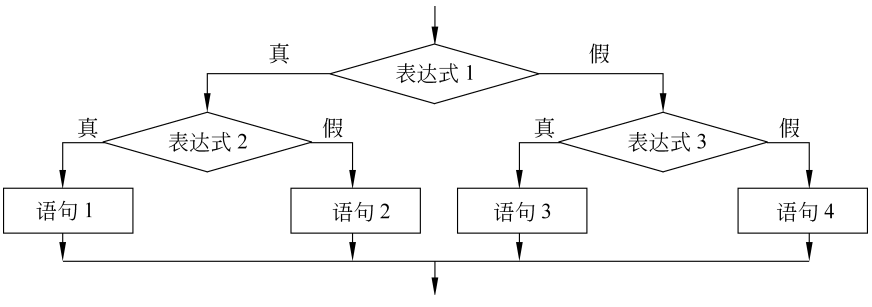

图 3.14 嵌套的 if 语句的流程图

在上述格式中,if 与 else 既可成对出现,也可不成对出现,且 else 总是与最近的 if 相配对。在编写这种语句时,每个 else 应与对应的 if 对齐,形成锯齿形状,这样就能更清晰

地表示 if 语句的逻辑关系。例如：

```
if(x>=0)
    if(x>0)
        y=1;
    else
        y=0;
else
    y=-1;
```

(2) 形式二：

```
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 n+1;
```

此结构的程序流程是在多个分支中，仅执行表达式为真的那个 else-if 后面的语句。若所有表达式的值都为 0，则执行最后一个 else 后的语句。这种结构的流程图如图 3.15 所示。

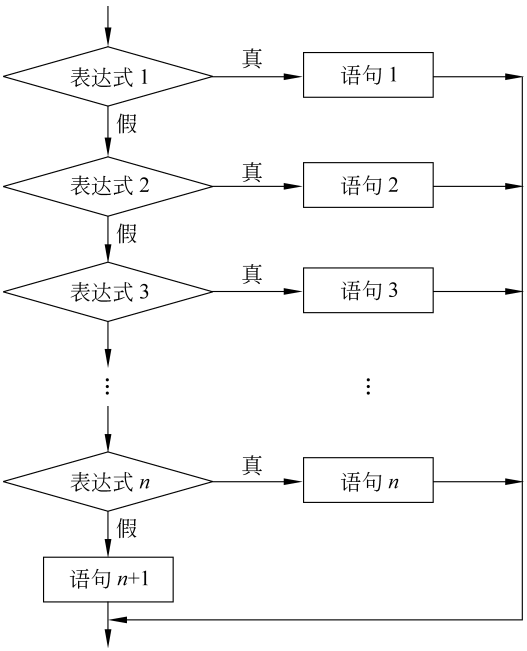

图 3.15　多分支 if-else if-else 的流程图

【例 3-7】 学生成绩可分为百分制和五分制，根据输入的百分制成绩 score，转换成相应的五分制输出，百分制与五分制的对应关系如表 3.4 所示。

表 3.4 百分制与五分制的对应关系

百 分 制	五 分 制
90≤score≤100	A
80≤score<90	B
70≤score<80	C
60≤score<70	D
0≤score<60	E

【问题分析】

本示例中，使用第二种形式的 if 嵌套语句对输入的数据逐步进行判断，并选择执行相应的操作。

解决该问题的算法流程图如图 3.16 所示。

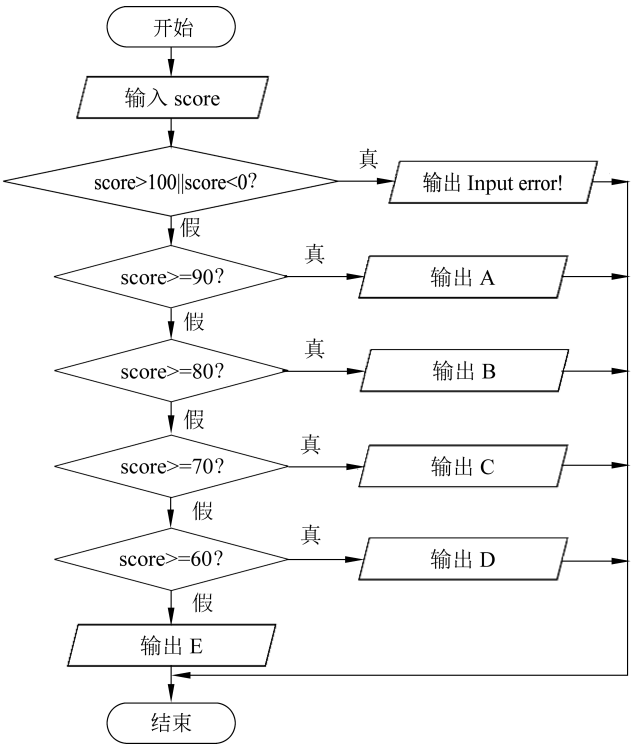

图 3.16 例 3-7 的流程图

【程序代码】

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

```
{
    int score;                                  //定义变量表示分数
    printf("Please enter score:");              //输出屏幕提示语
    scanf("%d",&score);                         //输入百分制的分数
    if(score>100||score<0)                      //分值不合理时显示出错信息
        printf("Input error!\n");
    else if(score>=90)                          //分数范围在 90~100 的情况
        printf("A\n");
    else if(score>=80)                          //分数范围在 80~89 的情况
        printf("B\n");
    else if(score>=70)                          //分数范围在 70~79 的情况
        printf("C\n");
    else if(score>=60)                          //分数范围在 60~69 的情况
        printf("D\n");
    else                                        //分数范围低于 60 的情况
        printf("E\n");
    return 0;
}
```

【运行结果】

程序运行结果如图 3.17 所示。

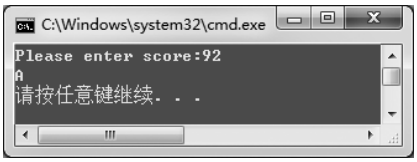

图 3.17　例 3-7 程序运行结果

【代码解析】

(1) 本示例定义一个变量 score 用来表示分数，使用嵌套的 if 语句对分数的范围进行检查判断，根据表 3.4 的对应关系，输出相应的分数等级。

(2) if 和 else 的配对关系，else 总是与其前方最靠近的、并且没有其他 else 与其配对的 if 相配对。

(3) 每一个 else 本身都隐含了一个条件，如本示例中的第一个 else 实质上表示条件 score>=0&&score<=100 成立，此隐含条件与对应的 if 所给出的条件完全相反，在编程时要善于利用隐含条件，使程序代码清晰简洁。

3.4　switch 语 句

上面介绍的 if 语句，常用于两种情况的选择结构，如果要表示两种以上的条件选择，可以采用嵌套 if 语句或多级嵌套的 if-else 语句，还可以用简洁的多分支选择 switch 语

句。switch 语句的一般形式如下：

```
switch(表达式)
{
    case 常量表达式 1: [语句系列 1;]
    case 常量表达式 2: [语句系列 2;]
    ……
    case 常量表达式 n: [语句系列 n;]
    [default: 语句系列 n+1;]
}
```

其中，方括号中的内容是可选项。

switch 语句一般形式的流程图如图 3.18 所示。

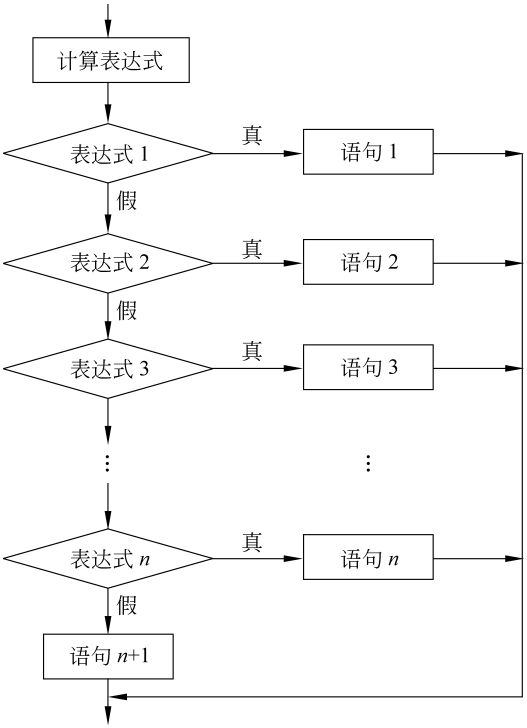

图 3.18　switch 语句一般形式的流程图

switch 语句的执行过程是：首先计算 switch 后表达式的值，然后将其结果值与 case 后的常量表达式的值依次进行比较，若此值与某 case 后常量表达式的值一致，即转去执行该 case 后的语句系列；若没有找到与之匹配的常量表达式，则执行 default 后的语句系列。

【例 3-8】 从键盘上输入 1～7 的数字时，然后显示对应的星期几的英文单词。当输入数字不在 1～7 时，输出“Error!”。

【问题分析】

本示例中，要求根据输入的数字，输出星期几的英文单词，可以使用 switch 语句来判断输入的数字。

解决该问题的算法流程图如图 3.19 所示。

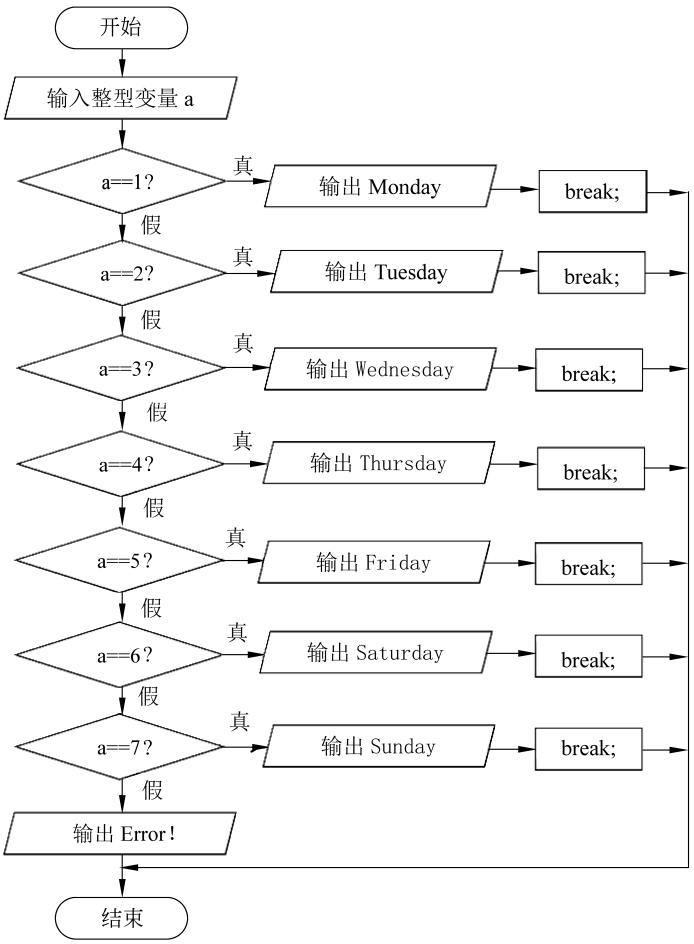

图 3.19　例 3-8 的流程图

【程序代码】

```
#include <stdio.h>
int main()
{
    int a;                                  //定义整型变量 a 表示输入的数字
    printf("Please enter an integer:");     //输出屏幕提示语
    scanf("%d",&a);                         //输入 1~7 的数字
    switch(a)                               //switch 语句判断
    {
        case 1:                             //a 的值为 1 的情况
```

```
            printf("Monday\n");                 //输出 Monday
            break;                              //跳出 switch 语句
        case 2:                                 //a 的值为 2 的情况
            printf("Tuesday\n");                //输出 Tuesday
            break;                              //跳出 switch 语句
        case 3:                                 //a 的值为 3 的情况
            printf("Wednesday\n");              //输出 Wednesday
            break;                              //跳出 switch 语句
        case 4:                                 //a 的值为 4 的情况
            printf("Thursday\n");               //输出 Thursday
            break;                              //跳出 switch 语句
        case 5:                                 //a 的值为 5 的情况
            printf("Friday\n");                 //输出 Friday
            break;                              //跳出 switch 语句
        case 6:                                 //a 的值为 6 的情况
            printf("Saturday\n");               //输出 Saturday
            break;                              //跳出 switch 语句
        case 7:                                 //a 的值为 7 的情况
            printf("Sunday\n");                 //输出 Sunday
            break;                              //跳出 switch 语句
        default:                                //默认情况
            printf("Error!\n");                 //提示错误
            break;                              //跳出
    }
    return 0;
}
```

【运行结果】

程序运行结果如图 3.20 所示。

图 3.20　例 3-8 程序运行结果

【代码解析】

本示例中使用 switch 来判断整型变量 a 的值，利用 case 语句检验 a 值的不同情况。假设 a 的值为 2，那么执行 case 为 2 时的情况，执行后跳出 switch 语句。如果 a 的值不是 case 中所列出的情况，那么执行 default 后的语句。在每一个 case 语句或 default 语句后都有一个 break 语句，该 break 语句用来跳出 switch 结构，不再继续执行该 case 语句或 default 语句后的代码。

在使用 switch 语句时，应注意以下几点：

(1) switch 后的表达式和 case 后的常量表达式可以是整型、字符型、枚举型，但不能是实型。

(2) 在同一个 switch 语句中，每个 case 后的常量表达式的值必须互不相等。

(3) case 后的语句系列可以是一条语句，也可以是多条语句，此时多条语句也不必用大括号括起来。

(4) default 可以省略，此时如果没有与 switch 表达式相匹配的 case 常量，则不执行任何语句，程序转到 switch 语句后的下一条语句执行。

(5) break 语句和 switch 最外层的右大括号是退出 switch 选择结构的出口，遇到第一个 break 即终止执行 switch 语句。如果程序没有 break 语句，则在执行完某个 case 后的语句系列后，将继续执行下一个 case 中的语句系列，直到遇到 switch 语句的右大括号为止。因此，通常在每个 case 语句执行完后，增加一个 break 语句来达到终止 switch 语句执行的目的。

在例 3-8 中，若每个 case 语句中没有 break 语句，则输入“5”时，输出结果如图 3.21 所示。

图 3.21　例 3-8 不添加 break 程序运行结果

(6) 每个 case 及 default 的次序是任意的，也就是说，default 可以位于 case 之前。例如：

```
int a=4;
switch(a)
{
    case 1:a++;
    default:a++;
    case 2:a++;
}
printf("a=%d",a);
```

此程序段的运行结果为：a=6。

由此可以看出，在上述情况下，执行完 default 后的语句系列之后，程序将自动转移到下一个 case 继续执行。

(7) 如果多种情况都执行相同的程序块，则对应的多个 case 可以执行同一语句系列。

【例 3-9】 例 3-7 可以用 if 嵌套语句实现，也可以用 switch 语句实现，实现代码如下。

```
#include <stdio.h>
int main()
{
    int score;                              //定义整型变量表示分数
    printf("Please enter score:");          //输出屏幕提示语
    scanf("%d",&score);                     //输入百分制的分数
    switch(score/10)                        //使用 switch 语句判断分数的十位数
    {
        case 10:
```