第5章 条件分支选择结构程序设计

生活中常常会根据不同情况采取对应的不同处理措施解决实际问题，反映在程序设计中，这类问题的求解算法体现为根据不同的判定条件控制程序流程执行不同的程序段，其中判断条件即求解表达式。本章主要内容如下：

- 简单 if 分支选择结构；
- if-else 分支选择结构；
- 嵌套的 if-else 分支选择结构；
- else if 多路分支选择结构；
- switch-case 条件选择结构；
- switch-case 条件选择语句；
- 条件选择综合案例分析。

5.1 条件分支选择结构

条件分支选择结构又称条件判断选择结构或分支选择结构，是结构化程序设计流程控制的基本结构之一。当程序设计算法实现需要选择判断处理时，就要使用条件分支选择结构控制程序流程走向。C 程序设计条件分支选择结构的语句包括 if 语句的 3 种形式，即简单 if 语句、if-else 语句和 else-if 语句，以及多分支选择 switch-case 开关语句等，在使用时可根据具体问题的复杂程度做适当的选择。下面分别加以介绍。

5.1.1 简单 if 分支选择结构

C 语言中 if 条件分支选择结构命令是最基本的条件判断语句，用来判定是否满足指定的条件，并根据求解条件表达式的值判断执行给定的操作。

简单 if 语句的一般形式如下：

```
if (<表达式>)
    [控制语句];
```

其中 if()表达式后面的尖括号表示其中的内容为必选项；方括号表示其中的内容为可选项，控制语句也可以是用花括号括起来的一组命令语句构成的一条复合语句。

简单 if 语句的执行流程如图 5-1 所示。

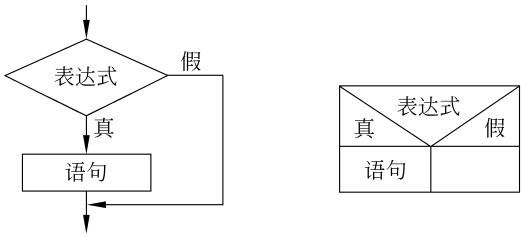

图 5-1　简单 if 语句的执行流程

系统首先对表达式求解，当表达式值为真时，则执行指定的语句；否则跳过指定语句，接着执行指定语句下面的语句。

例 5-1　编写程序，比较两个数的大小，按不同数据输入情况输出不同结果。

程序源代码：

```
/* L5_1.C */
#include<stdio.h>
main()
{
    int m,n;
    printf("Please input two numbers m,n=");
    scanf("%d,%d",&m,&n);
    if(m>n)
        printf("The first number is bigger\n");
                                                    /* 条件表达式 m>n 值为 1 时执行 */
    if(n>m)
        printf("The second number is bigger\n");
    if(n==m)
    printf("The two numbers are equal\n");
    getch();
}
```

分别运行 3 次程序，各次运行输出结果如图 5-2 所示。

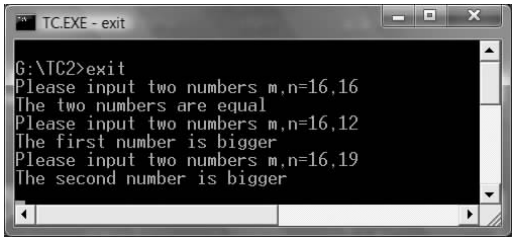

图 5-2　输入不同数据的条件输出结果

该程序用简单的 if 结构对两个变量 m 和 n 的值进行比较，完成 3 种情况的选择处理。从程序中可以看出，if 结构只在条件表达式为真时才执行指定的操作。如果程序应

用只需实现单向分支选择结构，使用简单 if 语句完成算法较为清晰方便。

多个简单的 if 结构配合，也可以处理稍微复杂一点的问题，如实现简单的排序算法。以 3 个变量排序为例，随机输入 3 个数，分别赋予变量 a、b、c，利用顺序执行的简单 if 结构语句实现排序算法，要求按变量 a、b、c 值由小到大输出结果。

程序源代码：

```
#include "stdio.h"
main()
{
    float a,b,c,t;
    scanf("%f,%f,%f",&a,&b,&c);
    if(a>b)
    {t=a; a=b; b=t;}                    /*借助临时变量 t,交换 a、b 两个变量的值*/
    if (a>c)                            /*值较小的 a 变量再与第三个变量 c 的值比较*/
    {t=a; a=c; c=t;}                    /*至此 a 变量的值最小*/
    if (b>c)                            /*剩余 b 变量的值再与 c 的值比较*/
    {t=b; b=c; c=t;}                    /*至此 b 变量的值次小*/
    printf("%5.2f,%5.2f,%5.2f",a,b,c);   /*排序结果*/
}
```

程序中使用 3 个简单 if 语句进行变量值的两两比较，用 if 语句判别变量值的大小，根据条件进行变量值的交换，每个 if 结构执行后较小的变量再与其余的变量作比较，最后输出排序结果。

5.1.2 if-else 分支选择结构

if-else 分支选择结构可实现同一表达式判断下的两路程序流程的选择。if-else 命令语句的一般形式如下：

```
if(<表达式>)
        [控制语句 1];
else
        [控制语句 2];
```

同样，if-else 中的控制语句也可以是用花括号括起来的一组语句构成的一条复合语句。if-else 语句的执行流程如图 5-3 所示。

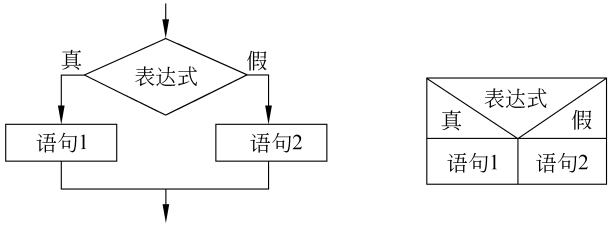

图 5-3　if-else 语句的执行流程

系统先对表达式求解，当表达式值为真时，执行控制语句 1；当表达式值为假时，执行控制语句 2。整个结构只有一个入口和一个出口，符合结构化程序设计规范。利用 if-else 分支选择形式的 if 语句可以实现两路程序流程的分支选择。

例 5-2 编写程序，输入一个年份值，检查该年是否为闰年，输出结果。检查某一年为闰年的自然语言描述是：年份值能被 4 整除，但不能被 100 整除；或者年份只能被 400 整除。表达式为 x%4==0&&x%100!=0||x%400==0。

程序源代码如下：

```
/*L5_2.C*/
#include<stdio.h>
main()
{
    int x;
    printf("Please input the year to determined=");
    scanf("%d",&x);
    if(x%4==0&&x%100!=0||x%400==0)                    /*满足闰年条件*/
        printf("%d is a leap year\n",x);
    else                                               /*不满足闰年条件*/
        printf("%d is not a leap year\n",x);
}
```

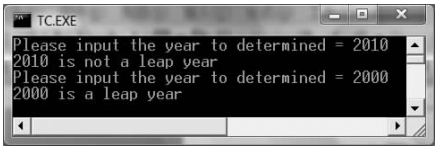

图 5-4 if-else 选择结构

执行上述程序时，根据输入的 x 值进行判断，如果满足闰年表达式，表达式值为 1，则执行选择语句中 if 命令后面的语句；表达式值为 0，执行 else 命令后面的语句。编译后运行两次程序，运行结果如图 5-4 所示。

如果 if-else 条件分支的 if 和 else 后的命令语句执行的是给同一个变量赋值操作，则该算法也可以用条件运算符“?:”取代 if-else 语句，实现相同的功能。

例 5-3 编写程序，从键盘输入一个字符，如果是小写字母，则将其转换为大写字母，否则原样输出该字符。

程序源代码：

```
/*L5_3.C*/
main()
{
    char ch;
    printf("please input a character:\n");
    scanf ("%c",&ch);
    ch=(ch>='a' && ch<='z')?(ch-32):ch;          /*用条件运算符实现判断赋值*/
    printf("ch=%c\n",ch);
}
```

编译后分别运行两次程序，如果输入小写字母 r，判断条件运算表达式为真，执行运

算符“?”后的赋值表达式，将 ch－32 的值赋给变量 ch，得到 ch 值为大写字母 R 的 ASCII 码值。如果输入字符 9，则由于不满足条件运算表达式中的条件，执行运算符“:”后的赋值表达式，因此直接输出该字符。两次运行程序后输入和对应输出结果如图 5-5 所示。

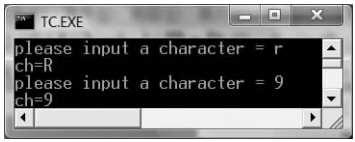

图 5-5　条件运算表达式实现选择运算

与本例相似，选择运算结果赋给同一个变量，如求最大值运算，源程序代码如下：

```
/*L5_3_1.C*/
#include<stdio.h>
main()
{
    int a,b,c;
    scanf("%d,%d",&a,&b);
    c=(a>b)?a:b;                    /*取 a 和 b 中大者赋给 c */
    printf("The max number is %d",c);
}
```

程序运行后，输入两个数值，分别赋给两个变量 a 和 b，条件运算表达式运算取变量 a 或变量 b 中值较大者赋给变量 c，输出变量 c 值。

5.1.3　嵌套的 if-else 选择结构

if-else 结构解决了程序流程按条件两路分支的问题，当按不同条件选择两个以上分支流程，就需要使用 if 语句的嵌套结构，即 if 语句结构中还可包含 if 语句结构，形成层层嵌套的 if 语句结构。嵌套在 if-else 结构 if 命令语句之后的内嵌 if-else 结构形式为

```
if(<表达式 1>)
        [控制语句 1];
    if(<表达式 2>)
        [控制语句 2];
    else
        [控制语句 3];
else
    [控制语句 4];
```

嵌套的 if 语句结构逻辑关系是：当选择条件＜表达式 1＞的值为真，才会判断“＜表达式 2＞”的值，也只有当＜表达式 2＞为真时，才会执行控制语句 2，此时＜表达式 1＞和＜表达式 2＞条件同时满足，是逻辑与的关系。这就是条件选择语句嵌套结构的特点。

同样，嵌套在 if-else 结构 else 命令语句之后的内嵌 if-else 结构的一般形式为

```
if(<表达式 1>)
        [控制语句 1];
else if(<表达式 2>)
          [控制语句 2];
```

```
    else if(<表达式 3>)
        [控制语句 3];
          else if(<表达式 4>)
            [控制语句 4];
              else[控制语句 5];
```

程序流程图如图 5-6 所示。

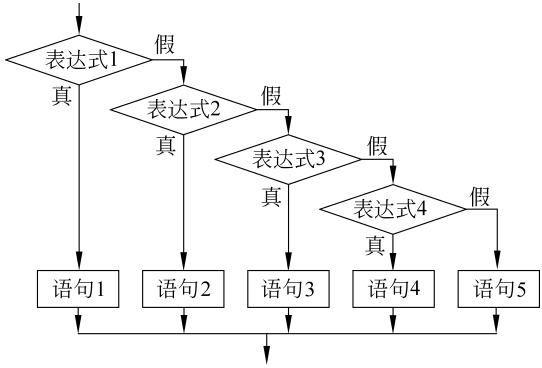

图 5-6　嵌套在 else 命令语句之后的内嵌结构

例 5-4　编写程序，实现符号函数的算法，即从键盘输入一个数值赋给变量 x，如果 x 值小于 0，符号函数值为－1；如果 x 值等于 0，符号函数值为 0；如果 x 值大于 0，符号函数值为 1，检验输出结果。

数学表达式：

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

程序源代码：

```
/*L5_4.C*/
#include<stdio.h>
main()
{
    int number,sign;
    printf("Please type in a number x=");
    scanf("%d",&number);
    if(number<0)
        sign=-1;
    else if(number==0)                    /*else 包含条件(number>0)和(number=0)*/
            sign=0;
        else                              /*else 只包括(number>0)*/
            sign=1;
    printf("sign(x)=%d\n",sign);
}
```

运行 3 次，输入 3 个不同的数，输入和对应的输出结果如图 5-7 所示。

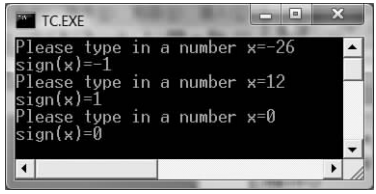

图 5-7　符号函数的运行结果

案例中 if 与 else 成对匹配，书写成锯齿形式易读、易理解。习惯上嵌套的 if 语句尽可能在 else 之后，如果嵌套中只有 if 而没有 else，容易造成错误，这是因为 else 总是与前面最相近的不包含 else 的 if 语句对应匹配。为避免发生这种错误，习惯上应将嵌套中的 if 语句用花括号{ }括起来，例如：

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

该案例中的 else 与最外层的 if 匹配，逻辑上不易出错。

实际上程序算法并不是唯一的，可以有不同的逻辑表达方式，只要最终目的和结果正确即可。当然，大型软件开发还有程序优化等问题，只是不在本书范围内。现以符号函数为例再作一些案例分析，以进一步理解 if 嵌套的逻辑关系。为便于案例分析，首先给出正确的程序源代码：

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

该程序算法已通过运行验证，算法逻辑正确。

将程序的 if 语句嵌套关系改为以下程序段：

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

该段程序的第一个 if 条件 x＞＝0 包含着两种情况，需要嵌套 if 条件 x＞0 语句，内嵌 if 语句的 else 命令意味着条件 x＝＝0，此处 y 等于 0，算法正确。

若将程序的 if 语句嵌套关系改为以下程序段：

```
y=-1;
```

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

该段程序在 if-else 之外先赋值 y=－1。第一个 if 条件 x!=0 也包含着两种情况，需要嵌套 if 条件 x>0 语句，满足该条件则 y=1。内嵌 if 语句受外层 if 条件 x!=0 约束，else 意味着条件 x<0，而 y 等于 0，算法出现逻辑错误，可将该程序的嵌套关系作以下修改：

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

增加了一对花括号，使 else 与外层 if 匹配，else 隐含条件为 x==0，执行 y=0，整个程序段算法正确。

再看如下算法程序段：

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

该段程序在 if-else 之外先赋值 y=－1。第一个 if 条件 x>=0 同样包含着两种情况，需要嵌套 if 条件 x>0 语句，满足该条件则 y=1。内嵌 if 语句受外层 if 条件 x>=0 约束，else 意味着条件 x=0，而 y 等于－1，算法也出现逻辑错误，可将该程序的嵌套关系作以下修改：

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

增加了一对花括号，使 else 与外层 if 匹配，else 隐含条件为 x<0，执行 y=－1，整个程序段算法正确。

当需要处理的分支选择头绪多且问题更加复杂时，可以在各种结构形式的 if 语句中再嵌套一个或多个 if 语句，形成更加复杂的嵌套。使用嵌套选择结构，可以在 if-else 结构中 if 命令后面嵌套，也可以在 else 命令后面嵌套。其组合形式可以表示为

```
if (表达式 1)
    if (表达式 2)
        语句 2;
    else
        语句 3;
else
    if (表达式 3)
```

```
        语句 4;
    else if (表达式 4)
        语句 5;
    else
        语句 6;
```

其中的各 if 和 else 语句中又可以嵌套另外的 if-else 语句。使用时注意 if 及 else if 与 else 的配对关系，系统总是将 else 与其向上最接近的未配对的 if 匹配组合。使用内嵌简单 if 语句时还应注意，简单形式的 if 语句中不包含 else 语句，嵌套逻辑上避免混乱。

5.1.4 else if 多路分支选择结构

嵌套的 if 结构丰富了程序流程按条件多路分支流程控制问题，当按不同条件选择更多分支流程，可以使用 if 嵌套结构，也可以使用 else if 多路分支选择结构。else if 多路分支选择结构的一般形式为

```
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 if(<表达式 n>)
              语句 n;
            else
                语句 n+1;
```

else if 多路分支选择结构逻辑关系上是由多个 if-else 层层嵌套的 if-else 语句组成的 if-else-if-else 逻辑结构，其选择分支执行流程如图 5-8 所示。

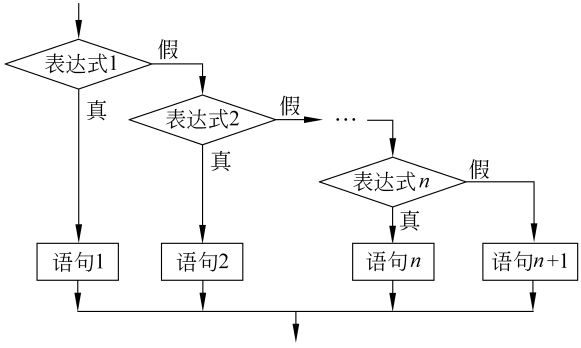

图 5-8　if-else-if-else 多路选择结构流程

if-else-if-else 分支结构中的每个 else if 表达式控制着一个分支流程，程序执行时首

先求解＜表达式 1＞的值，当＜表达式 1＞值为真时，执行语句 1，执行后跳出并结束该选择结构；当＜表达式 1＞值为假时，则求解＜表达式 2＞的值，当＜表达式 2＞值为真时，执行语句 2，执行结束也跳出并结束所在选择结构；同样，再继续求解＜表达式 3＞的值，当＜表达式 3＞值为真时，执行语句后跳出选择结构，否则继续向下进行判断，依此类推，直到＜表达式 n＞的值为真，执行相关语句。如果所有的表达式值都为假，那么就执行最后一个 else 后面的语句。

实际上，if-else if-else 形式的 if 语句是 if-else 形式的 if 语句的嵌套使用，利用它能够实现多分支选择，应用时应该注意每个 else 实际上是和其前面最接近的 if 配对使用的，通常中间各个 else 不能省略，但最后一个 else 可以省略，这时表示当所有的表达式值都为假时不作任何处理，接着执行选择结构之外的命令语句。

else if 结构执行是从上到下对所有条件表达式逐一进行扫描判断求解，只有遇到某个条件表达式值为真，才会选择执行与之对应的语句，执行后即跳出整个选择判断结构。如果没有任何一个条件满足，即所有表达式值均为假，则执行最后一个 else 命令后的语句，这个 else 通常作为默认条件使用。

例 5-5 编写程序，判断从键盘输入字符的类属范围，输出对应的分类属性提示。例如，输入字符 6，输出提示属于数字。

算法分析：字符类别是根据键盘输入字符的 ASCII 码值来区分的。在 ASCII 码表中，若输入的字符 ASCII 码值小于 32，通常为控制字符；若在字符 0～9 之间，则为数字；若在字符 A～Z 之间，则为大写字母；而在字符 a～z 之间，为小写字母；其余则列为其他字符。该程序算法属于多条件选择分类处理问题，适于 else if 多条件判断分支选择结构实现，程序流程图如图 5-9 所示。

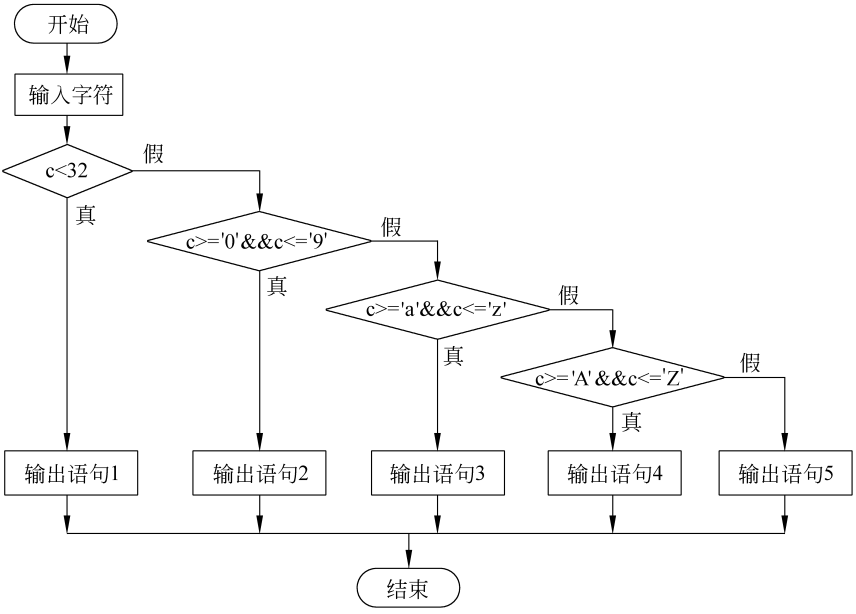

图 5-9 判断输入字符的 ASCII 码值，选择输出语句