第5章 选择结构

选择结构是结构化程序设计的3种基本结构之一，其作用是根据逻辑判断的结果决定程序的不同流程。

在设计选择结构程序时，要考虑两方面的问题：一是在C语言中如何表示条件；二是在C语言中用什么语句实现选择结构。

本章将详细介绍C语言中的if语句和switch语句，它们都用来实现选择结构。

5.1 if语句构成的选择结构

在前面的章节已经介绍了关系表达式和逻辑表达式，运算后都会得到一个逻辑结果。逻辑结果只有两个，即“真”和“假”，用1和0来表示。在C语言中，没有专门的“逻辑值”，对于操作数而言用非零值表示“真”值，用零值表示“假”。

C语言常用关系表达式或逻辑表达式来表示条件，即表示逻辑判断；用if语句、switch语句来实现选择结构。

5.1.1 if语句

if语句用来判断条件表达式是否成立，根据判断结果（“真”或“假”）决定执行什么操作。

if语句的一般形式如下：

```
if(表达式)
{    语句组1;  }
[else
{    语句组2;  } ]
```

其中，方括号[ ]中的项为可选项。

1. 不含else的if语句

1）语句形式

不含else子句的if语句也称为单分支选择语句，它的语句形式如下：

```
if(表达式)
{    语句组;  }
```

其中：

（1）if 是 C 语言的关键字，不能用作标识符。

（2）if 后面的“表达式”必须用括号括起来。表达式除常见的关系表达式或逻辑表达式外，也允许是其他类型的表达式，如整型、实型、字符型等，只要表达式的值非零即为真。

（3）当 if 后面的语句组仅由一条语句构成时，也可不使用复合语句形式（即去掉花括号）。

2）执行过程

首先计算表达式的结果，若为真（非零），则执行后面的语句组，然后再执行 if 语句后的下一个语句；若表达式的值为假（零），则跳过 if 语句，直接执行 if 语句的下一个语句。单分支 if 语句的执行过程如图 5-1 所示。

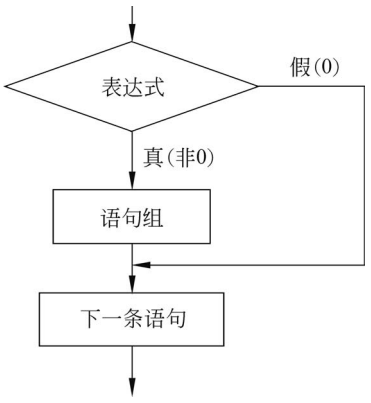


图 5-1　单分支 if 语句的执行过程

【例 5-1】 输入一个实数，求它的绝对值。

```
#include<stdio.h>
int main(int argc,char *argv[])
{
    double x;
    printf("请输入一个实数：");
    scanf("%lf",&x);
    if(x<0) x=-x;
    printf("该数的绝对值是：%g\n",x);
    return 0;
}
```

在 x 小于 0 的情况下 x 取反，而 x 大于或等于 0 并不取反，这样就可以得到 x 的绝对值。该程序的一次运行结果如图 5-2 所示。

【例 5-2】 输入 3 个实数，按从小到大的顺序输出。

图 5-2　例 5-1 的一次运行结果

```
#include<stdio.h>
int main(int argc,char *argv[])
{
    double a,b,c,t;
    printf("请输入 3 个实数：");
    scanf("%lf%lf%lf",&a,&b,&c);  //输入数据
    if(a>b)    //比较 a,b, 让 a 取 a、b 的最小值
    {  t=a;  a=b;  b=t; }
    if(a>c)    //比较 a,c, 让 a 取 a、c 的最小值, 则 a 是 a、b、c 的最小值
    {  t=a;  a=c;  c=t;  }
    if(b>c)    //比较 b,c, 让 c 取 b、c 的最大值, 则 c 是 a、b、c 的最大值
    {  t=b;  b=c;  c=t;  }
    printf("这三个数按从小到大的顺序排序后是：%g,%g,%g\n",a,b,c);
    return 0;
}
```

一次运行结果如图 5-3 所示。

图 5-3　例 5-2 的一次运行结果

将例 5-2 改为输出三个数中的最大值或者最小值，程序应该怎么实现？

2. 含 else 子句的 if 语句

1）语句形式

含 else 子句的 if 语句也称为双分支选择语句，它的语句形式如下：

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

其中：

（1）if、else 都是 C 语言的关键字，不能用作标识符。

（2）“语句组 1”称为 if 子句，“语句组 2”称为 else 子句。若语句组仅由一条语句构成，也可不使用复合语句形式（即去掉大括号）。

（3）else 是 if 语句的一部分，它总是与它前面最近的未匹配的 if 语句进行匹配，不能单独出现。

例如：

```
else  printf("***");       //错误语句, else 必须与 if 配对使用, 不能单独使用
```

2）执行过程

当 if 表达式的值不等于 0（逻辑真）时，执行语句组 1，然后执行 if 语句后面的语句；否则，执行语句组 2，接着执行 if 语句后面的语句。if…else 语句的执行过程如图 5-4 所示。

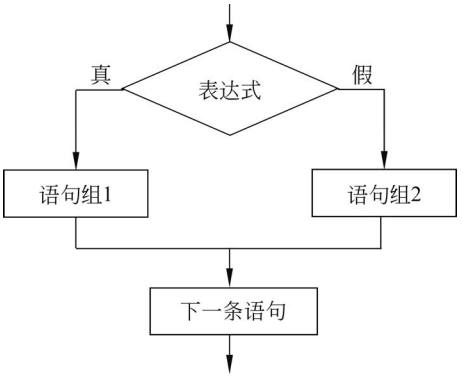


图 5-4　if…else 语句的执行过程

【例 5-3】 输入一个整数，判断它是奇数还是偶数。

```
#include<stdio.h>
int main(int argc,char *argv[])
{
    int x;
    printf("请输入一个整数：");
    scanf("%d",&x);
    if(x%2==0)       //能被 2 整除的是偶数
       printf("%d 是偶数\n",x);
    else             //不能被 2 整除的是奇数
       printf("%d 是奇数\n",x);
    return 0;
}
```

运行结果如图 5-5 所示。

```
请输入一个整数：37
37是奇数
```

（a）输入奇数的运行结果

```
请输入一个整数：-124
-124是偶数
```

（b）输入偶数的运行结果

图 5-5　例 5-3 的运行结果

【例 5-4】 输入一个年份，判断它是平年还是闰年。

```
#include<stdio.h>
int main(int argc,char *argv[])
{
    int year;
    printf("请输入一个年份：");
    scanf("%d",&year);
    //能被 4 整除但是不能被 100 整除或者能够被 400 整除的是闰年
    if(((year%4==0) && (year%100!=0)) || (year%400==0))
        printf("%d 是闰年\n", year);
```

```
    else
        printf("%d 是平年\n", year);
    return 0;
}
```

运行结果如图 5-6 所示。

（a）输入闰年的运行结果

（b）输入平年的运行结果

图 5-6　例 5-4 的运行结果

目前闰年的规则是：四年一闰，百年不闰，四百年再闰，这是由地球绕太阳运行的周期决定的。

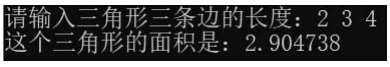
本例中 if 后面的括号中的其他括号根据运算符的优先级都可以省略，但是根据程序设计“清晰第一、效率第二”的原则还是建议使用括号，这样可以提高程序的可读性。

【例 5-5】 求三角形的面积。

```
#include<stdio.h>
#include<math.h>                              //sqrt 函数所在的头文件
int main(int argc,char *argv[])
{
    double a,b,c,area,s;
    printf("请输入三角形三条边的长度：");
    scanf("%lf%lf%lf",&a,&b,&c);
    if(a+b>c && b+c>a && a+c>b)               //三角形的条件
    {
        //下面两行用来求三角形的面积
        s=0.5*(a+b+c);
        area=sqrt(s*(s-a)*(s-b)*(s-c));    // sqrt 为开平方函数
        printf("这个三角形的面积是：%lf\n",area);
    }
    else
       printf("这不是一个三角形，不能计算面积!\n");
    return 0;
}
```

运行结果如图 5-7 所示。

（a）a、b、c 不能构成三角形的运行结果



（b）a、b、c 能构成三角形的运行结果

图 5-7　例 5-5 的运行结果

5.1.2　嵌套的 if 语句

在数学中经常会用到分段函数，例如 $y=\begin{cases} x-12 & x<6 \\ 3x-1 & 6\leqslant x<15 \\ 5x+9 & x\geqslant 15 \end{cases}$。

如何用程序来实现分段函数的计算呢？这显然是一个多分支的情况，可以使用 if…else 的嵌套语句来解决。

if 和 else 子句中可以是任意合法的 C 语言语句，因此也可以是 if 语句，通常称为嵌套的 if 语句。内嵌的 if 语句可以嵌套在 if 子句中，也可以嵌套在 else 子句中。

1. 在 if 语句中嵌套

语句的一般形式如下：

```
if(表达式 1)
{
    if(表达式 2)
    {
        语句组 1;
    }
    [else
    {
        语句组 2;
    }]
}
else
{
    语句组 3;
}
```

执行过程：当表达式 1 的值为真（非零）时，执行内嵌的 if…else 语句；当表达式 1 的值为假（零）时，执行语句组 3。

【例 5-6】 编写程序，完成求分段函数 $y=\begin{cases} x-12 & x<6 \\ 3x-1 & 6\leqslant x<15 \\ 5x+9 & x\geqslant 15 \end{cases}$ 的值。

```
#include<stdio.h>
int main(int argc,char *argv[])
{
    int x, y;
    printf("请输入自变量 x：");
    scanf("%d", &x);
    if(x<15)
    {
        if(x<6)
        {
            y = x - 12;
        }
        else
```

```
        {
            y = 3*x - 1;
        }
    }
    else
    {
        y = 5*x + 9;
    }
    printf("x = %d, y = %d\n", x, y);
    return 0;
}
```

运行结果如图 5-8 所示。

```
请输入自变量x: 5
x = 5, y = -7
```

（a）x<6 的运行结果

```
请输入自变量x: 6
x = 6, y = 17
```

（b）6≤x<15 的运行结果

```
请输入自变量x: 15
x = 15, y = 84
```

（c）x≥15 的运行结果

图 5-8　例 5-6 的运行结果

2. 在 else 中嵌套 if 语句

语句形式如下：

```
if(表达式 1)
{
    语句组 1;
}
else
{
    if(表达式 2)
    {
        语句组 2;
    }
    [else{ 语句组 3; }]
}
```

或写为

```
if(表达式 1){ 语句组 1; }
else if(表达式 2)  { 语句组 2; }
   [else{ 语句组 3; }]
```

可以看出，在 else 中嵌套 if 语句程序的结构更清晰，因此建议在使用嵌套的 if 语句时，尽量把内嵌的 if 语句嵌套在 else 子句中。

C 语言程序书写格式比较自由，但是过于“自由”的程序书写格式，往往可读性不高。为了提高程序的可读性，一般程序在书写时都采用按层缩进的方式，每层缩进 4 个字符。

本书例程都采用这种方式。

3. 多层嵌套

语句形式如下：

```
if(表达式 1）
{
    语句组 1;
}
else
    if(表达式 2）
    {
        语句组 2;
    }
    else
                …
                    if(表达式 n)
                    {
                        语句组 n;
                    }
                    else
                    {
                        语句组 n+1;
                    }
```

这就形成了阶梯形的嵌套 if 语句，此语句可用以下形式表示，使得读起来层次分明又不占太多的篇幅。

```
if(表达式 1)
{   语句组 1;    }
else if(表达式 2)
{   语句组 2;    }
     …
elseif(表达式 n)
{   语句组 n;    }
else
{   语句组 n+1;  }
```

阶梯形嵌套 if 语句的执行过程如下：从上向下逐一对 if 后的表达式进行检测，当某一个表达式的值为真时，就执行它后面的语句组，阶梯形中的其他语句组就不执行。如果所

有表达式的值都为 0，则执行最后的 else 中的语句组 n+1，此时，如果程序中最内层的 if 语句没有 else 子句，即没有最后的那个 else 子句，那么就不进行任何操作。语句流程如图 5-9 所示。

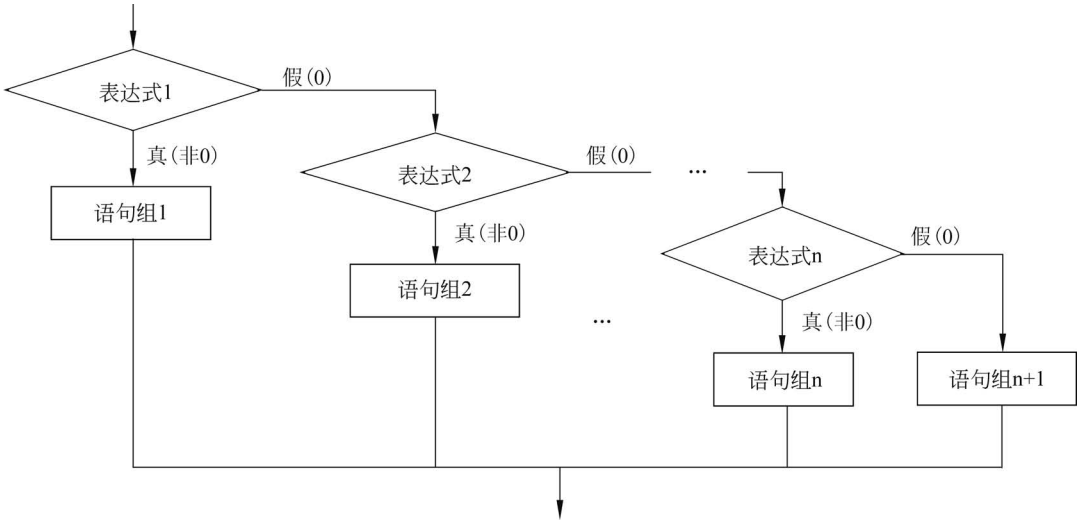


图 5-9　阶梯形嵌套 if 语句的执行过程

注意：

（1）当 if 语句中出现多个 if 与 else 时，要特别注意它们之间的匹配关系，否则就可能导致程序逻辑错误。else 与 if 的匹配原则是“就近一致原则”，即 else 总是与在它上面、距它最近且尚未匹配的 if 配对。为明确匹配关系，避免匹配错误，建议将内嵌的 if 语句，一律用大括号括起来。

（2）if 语句允许嵌套，但嵌套的层数不宜太多。在实际编程时，应适当控制嵌套层数，一般 2～3 层为宜，层数太多会导致可读性变差，匹配关系也容易出错。

（3）嵌套的 if 语句适用于数据范围是一个全集而且该全集可以分解成若干互不相交的子集的情况（见例 5-7）。如果各子集求并集的结果只是全集的一部分则不适合使用嵌套的 if 语句，而适合使用并列的 if 语句（见例 5-8）。

【例 5-7】 将例 5-6 改为使用在 else 中嵌套 if 语句来实现分段函数的求值。

```
#include<stdio.h>
int main(int argc,char *argv[])
{
    int x, y;
    printf("请输入自变量 x：");
    scanf("%d", &x);
    if(x < 6)
    {   y = x - 12;
    }
    else if(x < 15)
    {  y = 3*x - 1;
    }
    else
```

```
    {  y = 5*x + 9;
    }
    printf("x = %d, y = %d\n", x, y);
    return 0;
}
```

【例 5-8】编写程序，计算考查课的最终成绩，其规则是：平时成绩大于或等于 90 且小于或等于 100 则最终成绩为“优秀”，平时成绩大于或等于 80 且小于 90 则最终成绩为“良好”，平时成绩大于或等于 70 且小于 80 则最终成绩为“中等”，平时成绩大于或等于 60 且小于 70 则最终成绩为“及格”，平时成绩大于或等于 0 且小于 60 则最终成绩为“不及格”。

分析：该题目中所有成绩范围求并集后的结果是大于或等于 0 且小于或等于 100，这是整数的一个区间，并不是整数全集，因此不适合使用嵌套的 if 语句（因为当 if 取某一个成绩区间时，该 if 所对应的 else 会包含不合理的数据区域，例如成绩小于 0 或者成绩大于 100），这样的问题适合使用并列的 if 语句编写程序。示例代码如下：

```
#include<stdio.h>
int main(int argc,char *argv[])
{
    float score;  //成绩可以是实数, 如 84.5
    printf("请输入一个学生考查课的平时成绩：");
    scanf("%f", &score);
    if(score>=90 && score<=100) printf("该生该考查课的最终成绩为优秀\n");
    if(score>=80 && score<90) printf("该生该考查课的最终成绩为良好\n ");
    if(score>=70 && score<80) printf("该生该考查课的最终成绩为中等\n ");
    if(score>=60 && score<70) printf("该生该考查课的最终成绩为及格\n ");
    if(score>=0 && score<60) printf("该生该考查课的最终成绩为不及格\n ");
    if(score>100 || score<0) printf("该成绩无效\n ");
    return 0;
}
```

运行结果如图 5-10 所示。

```
请输入一个学生考查课的平时成绩：95
该生该考查课的最终成绩为优秀
```

（a）一个有效成绩的运行结果

```
请输入一个学生考查课的平时成绩：120
该成绩无效
```

（b）一个无效成绩的运行结果

图 5-10　例 5-8 的运行结果

5.2 switch 语句和 break 语句构成的选择结构

除了嵌套的 if 语句或并列的 if 语句外，C 语言还提供了 switch 语句作为多分支选择语句，但是 switch 语句的使用范围有限，它只能判断整型数据、字符型数据或枚举型数据，而这几种数据实质上都是整型数据。