第 5 章　控制语句：分支和跳转

5.1　本章内容与要求

本章介绍以下内容：

- C 语言的选择语句：if 单分支、双分支和多分支选择结构、switch 选择分支结构。
- 逻辑运算符、逻辑表达式，以及条件运算符和条件表达式。

本章首先介绍 if 单分支、双分支和多分支选择结构的一般形式、流程图和使用方法，然后介绍逻辑运算符、逻辑表达式，以及条件运算符和条件表达式优先级和结合性，最后介绍 switch 语句的一般形式和使用注意事项。

5.2　if 选择分支结构

5.2.1　为什么使用选择分支结构

现实生活中，往往需要进行判断和选择。例如，判断一个一元二次方程是否有实数根，根据学生的考试分数确定学生的等级，用户登录验证等。如果用户输入的账号和密码与系统预存的信息匹配，则允许用户登录并跳转到相应的用户界面；如果不匹配，则会提示用户输入错误，允许用户重新输入或采取锁定账户等其他安全措施。因此，C 语言设计了选择分支结构。

C 语言的选择分支语句有两种：if 语句和 switch 语句。switch 语句专门处理多分支语句，本节先介绍 if 语句的使用和案例，5.5 节介绍多分支语句和 switch 语句。

案例 5.1　假设 a＞0，判断方程 $ax^2+bx+c=0$ 是否有实根，若有，请输出实根。

```
#include<stdio.h>
#include<math.h>

int main() {
    double a, b, c, disc, x1, x2, p, q;

    scanf("%lf %lf %lf",&a,&b,&c);
    disc = b * b-4 * a * c;

    if(disc<0)
```

```
        printf("This equation has no real roots.\n");

    else
      {
         p = -b/(2.0 * a);
         q = sqrt(disc)/(2.0 * a);
         x1 = p + q;
         x2 = p - q;
         printf("Real roots are:x1 = %f, x2 = %f.\n", x1,x2);
      }

    return 0;

}
```

案例分析：

如从键盘输入 3 6.5 2.2，则输出为

```
Real roots are:x1 = -0.419799, x2 = -1.746868.
```

案例 5.1 中，变量 a、b、c 分别表示一元二次方程的 3 个系数，x1、x2 表示方程的两个根，disc、p、q 为计算过程中的中间变量。a、b、c 通过 scanf()函数从键盘获得，sqrt()函数为求平方根函数，函数定义包含在头文件<math.h>中。if else 语句为 if 语句的双分支结构，用于判断 disc 的值是否小于 0，即方程是否有实根。5.2.2 节将详细介绍 if 语句的具体使用方法。

5.2.2 if 选择分支语句

if 语句是 C 语言选择分支结构中的重要语句，包括单分支、双分支和多分支选择结构。单分支表示只有一个选择分支，双分支表示有两个选择分支，多分支表示有多个选择分支。

if 单分支选择语句的流程图如图 5.1 所示，一般形式为：

```
if(表达式)
   语句
```

其中，语句可以是简单语句，也可以是复合语句，复合语句需要用花括号括起来。当表达式的值为真(非 0)时，执行语句；当条件表达式的值为假时，不做任何操作。

if 双分支选择语句的流程图如图 5.2 所示，一般形式为：

```
if(表达式)
   语句 1
else
   语句 2
```

其中，语句 1 和语句 2 可以是简单语句，也可以是复合语句，复合语句需要用花括号括起来。当表达式的值为真(非 0)时，执行语句 1；当条件表达式的值为假(0)时，执行语句 2。

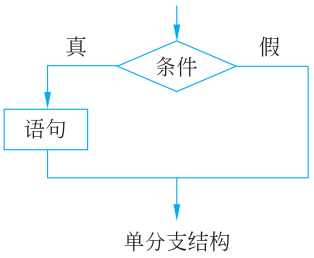

图 5.1　if 单分支选择语句的流程图

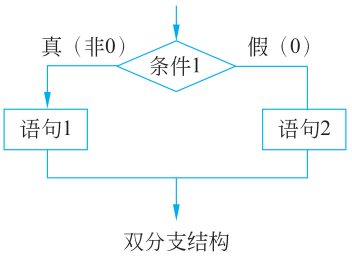

图 5.2　if 双分支选择语句的流程图

案例 5.2　从键盘获取字符，若输入字符为字母，则加 1 输出。采用单分支结构实现。

```
#include <stdio.h>
#include <ctype.h>
int main(void)
{
    char ch;

    while ((ch = getchar()) != '\n')
    {
        if (isalpha(ch))
            putchar(ch + 1);
    }

    return 0;
}
```

案例 5.3　从键盘获取字符，若输入字符为字母，则将其下一个字母输出。采用双分支结构实现。

```
#include <stdio.h>
#include <ctype.h>
int main(void)
{
    char ch;

    while ((ch = getchar()) != '\n')
    {
        if (isalpha(ch))
            putchar(ch + 1);
        else
            putchar(ch);

    }

    return 0;
}
```

案例分析：

上面两个案例中，通过 getchar()函数从键盘接收字符，利用 putchar()函数将字符显示在显示器上。getchar()的特点是可以接收空白字符，而 scanf()函数在接收字符串时遇到空白即停止读取。while 循环负责对输入的所有字符进行处理，直到遇到换行符“\n”结束循环。在循环体中，对接收到的每一个字符进行判断。使用 isalpha(ch)函数来检查当前读取的字符 ch 是否是一个字母(无论是大写还是小写)。isalpha()函数是＜ctype.h＞头文件提供的一个函数，用于字符分类，如果参数是字母('A'到'Z'或'a'到'z')，则返回非零值(真)，否则返回 0(假)。单分支语句只包括一个 if 语句，双分支语句由 if else 实现。输出字母的下一个字母由 ch＋1 实现。由于字母在 ASCII 码表中是连续排列的(大写字母'A'到'Z'的 ASCII 码值为从 65 到 90，小写字母'a'到'z'的 ASCII 码值为从 97 到 122)，因此将字母的 ASCII 码值加 1 相当于将其“转换”为字母表中的下一个字母。注意，这里没有对字母表边界(如'z'加 1 后变为'{')进行特殊处理，这可能会导致非预期的字符输出。

除了 isalpha()函数，头文件＜ctype.h＞中还包括多个字符处理函数，如表 5.1 所示。

表 5.1　头文件＜ctype.h＞中的字符处理函数

函数名	如果是下列参数，返回值为真
isalpha()	字母
isalnum()	字母或数字
isblank()	标准的空白字符(空格、水平制表符或换行符)或其他本地化指定为空白的字符
isctrl()	控制字符，如 Ctrl＋C
isdigit()	数字
isgraph()	除空格之外的任意可打印字符
islower()	小写字母
isprint()	可打印字符
ispunct()	标点符号(除空格或字母数字以外的任意可打印字符)
isspace()	空白字符(空格、换行符、换页符、回车符、垂直制表符、水平制表符或其他本地化定义的字符)
isupper()	大写字母
isxdigit()	十六进制数字符

除了单分支和双分支选择结构，if 语句还有多分支选择结构，其流程图如图 5.3 所示，一般形式为：

```
if(条件 1) 语句 1
else if(条件 2) 语句 2
else if(条件 3) 语句 3
…
else if(条件 n) 语句 n
else 语句 n+1
```

多分支结构的执行过程为：当条件 1 为真时，执行语句 1；当条件 1 为假时判断条件 2；

当条件 2 为真时,执行语句 2;当条件 2 为假时判断条件 3,以此类推;如果一直到条件 n 都不成立,就执行语句 n+1。

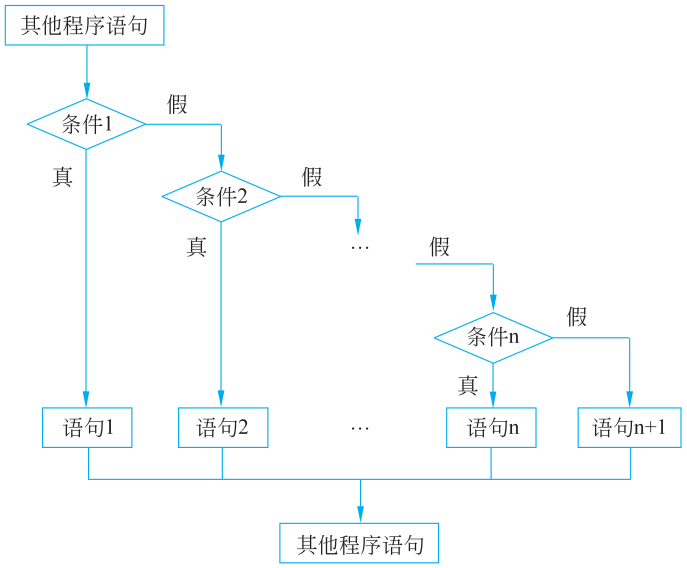

图 5.3　if 多分支语句的流程图

案例 5.4　实现简单的算术计算器,采用多分支结构实现。

```
#include<stdio.h>
int main()
{
    double v1, v2,result;
    char op;
    printf("input expression: ");
    scanf("%lf%c%lf", &v1, &op, &v2);
    printf("%.6lf %c %.6lf\n", v1, op, v2);
    if(op=='+')
        result=v1+v2;
    else if(op=='-')
        result=v1-v2;
    else if(op=='*')
        result=v1*v2;
    else if(op=='/')
        result=v1/v2;
    else
        printf("operator error !\n");
    printf("%.6lf\n",result);
    return 0;
}
```

案例分析:

案例 5.4 中,变量 v1、v2 表示两个操作数,result 表示算术运算结果,op 表示算术运算。

v1、v2 和 op 通过 scanf()函数从键盘接收。案例 5.3 采用了 if、else if、else 的多重选择分支结构，分别判断算术运算是否为加、减、乘、除中的一种。但此处需要注意，在除法操作中没有考虑除数是否为 0 这一问题。算法还有欠缺，需要进一步完善。

5.3 逻辑运算符和逻辑表达式

案例 5.5 判断输入的年份是否为闰年。

```
#include<stdio.h>
int main()
{
    int year;
    printf("请输入一个年份:");
    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;
}
```

案例分析：

根据格里高利历(公历)，闰年的规则为如果年份能被 4 整除但不能被 100 整除或者如果年份能被 400 整除，即满足两个条件中的任意一个为闰年。案例 5.4 中，if 后面的表达式描述了闰年的条件，“&&”“||”是逻辑运算符。C 语言提供了 3 种逻辑运算符，分别为!、&& 和||，分别表示逻辑非、逻辑与和逻辑或。“&&”和“||”为双目运算符，需要两个操作数。“!”为单目运算符，需要一个操作数。

由逻辑运算符构成的表达式叫作逻辑表达式。逻辑表达式的计算结果为逻辑值，即“真”或“假”。在 C 中表示逻辑运算结果，1 代表“真”，0 代表“假”。判断一个量是否为“真”时，非 0 为“真”；0 为“假”。表 5.2 展示了逻辑运算的真值表。

表 5.2 逻辑运算的真值表

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

逻辑运算符的优先级：！>算术运算符>关系运算符> &&和||>赋值运算符。

结合性：&&和||从左向右算，即左结合；！为单目运算符，即右结合。

例如，若 a 的值为 19，b 的值为 7，则！a 的值为 0，a&&b 的值为 1，！a&&b 的值为 0。

逻辑表达式的求值顺序是从左往右,但并不是所有的逻辑运算符都会被执行,只有在必须执行下一个逻辑运算符才能求出表达式的解时,才执行该运算符。即在逻辑表达式的求值过程中,一旦能确定逻辑表达式的值,就不再逐步求值。

假设“int a=0,b=2,c=1;”,求下列表达式的值及各变量的值:

(1) a && b++ && --c;

(2) a || b-- || c++。

表达式(1)的值为 0,各变量的值分别为 a=0,b=2,c=1。a 的值为 0,与任何表达式相与的结果均为 0,因此,表达式 1 的值确定为 0,后续的逻辑运算不再执行,a、b、c 保持原值不变。

表达式(2)的值为 1,各变量的值分别为 a=0,b=1,c=1。表达式(2)中,第一个逻辑运算符为“||”,则先计算表达式 a || b--的值。b--为后缀模式,所以先计算 a || b,b 的值再自减 1 变为 1。第二个逻辑运算符同样为“||”,a || b--表达式的值为 1,与任何表达式相关的结果均为 1,所以无须再进行后续的逻辑运算,c 保持原值不变。

5.4　条件运算符和条件表达式

条件运算符“?”“:”是 C 语言中唯一的三目运算符。一般形式为:

```
表达式 1? 表达式 2: 表达式 3
```

条件运算符相当于一个双分支选择结构,执行过程是:判断表达式 1 的值,如果为真,则执行表达式 2;如果为假,则执行表达式 3。

由条件运算符构成的表达式称为条件表达式,条件表达式的值为表达式 2 或表达式 3 的值。

案例 5.6　打印两个整数中较小的数。

```
#include<stdio.h>
int main()
{
    int num1 = 0, num2 = 0;
    scanf("%d %d", &num1,&num2);
    if (num1<num2)
        printf("%d 为较小的整数", num1);
    else
        printf("%d 为较小的整数", num2);
    return 0;
}
```

条件运算符的实现代码如下:

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

```
{
    int num1 = 0, num2 = 0,num = 0;
    scanf("%d %d", &num1,&num2);
    num = num1<num2 ? num1 : num2;
    printf("%d 为较小的整数", num);
    return 0;
}
```

案例分析：

案例 5.6 中，num1＜num2 ? num1 ：num2 的作用等效于 if else 双分支结构。用条件表达式处理简单的选择结构可以使程序更加简洁。

条件运算符的优先级高于赋值、逗号运算符，低于其他运算符。例如：

```
M < n ? x : a + 3  等效于(m < n) ? x :(a + 3)
```

当一个表达式中出现多个条件运算符时，结合方向为自右至左，即应该将位于最右边的问号与离它最近的冒号配对，并按这一原则正确区分各条件运算符的运算对象。

例如：

```
W < x ? x + w : x < y ? x : y  等效于 w < x ? x + w : ( x < y ? x : y)
```

5.5 嵌套分支选择结构

在 if 语句中又包含一个或多个 if 语句称为 if 语句的嵌套。一般形式为：

```
if(表达式)
    内嵌 if 语句
else
    内嵌 if 语句
```

“内嵌 if 语句”可以为 if 语句 3 种基本形式(单分支、双分支、多分支)中的任意一种。if 语句的嵌套形式中，可能会出现多个 if 和多个 else 重叠的情况，这时要特别注意 if 和 else 的配对问题。C 语言规定：else 总是与它前面最近且还没有配对的 if 配对，称为就近原则。

案例 5.7 分析下列代码中 if 和 else 的配对情况。

```
#include<stdio.h>
int main()
{
    int num1 = 0, num2 = 0, num3 = 0, num = 0;
    scanf("%d %d %d", &num1,&num2,&num3);
    if(num1<num2)
    if(num1<num3)
        num = 10;
```

```
    else
        num = 20;
    printf("num = %d.",num);

    return 0;
}
```

案例分析：

当输入 5 2 7 时，程序的运行结果为：

```
num = 0
```

根据就近匹配原则，案例中的 else 与第二个 if 配对，因为 num1 的值大于 num2，所以程序直接跳转到 printf()函数处打印。案例中的配对关系不够清晰，为了使配对关系更加明确，可以用花括号和缩进来确定配对关系，即可以改写为以下代码：

```
#include<stdio.h>
int main()
{
    int num1 = 0, num2 = 0, num3 = 0, num = 0;
    scanf("%d %d %d", &num1,&num2,&num3);
    if(num1<num2)
    {
        if(num1<num3)
            num = 10;
    }
    else
        num = 20;

    printf("num = %d.",num);
    return 0;
}
```

此时，当输入 5 2 7 时，程序的运行结果为 num = 20。

5.6　switch 语句

if 语句可以处理多分支语句，但如果分支较多，则嵌套的 if 语句层数就会变多，程序会变得冗长，导致可读性降低。当针对整数进行分类处理时，用 switch 多分支选择结构要比 if-else 清晰得多。

switch 语句的一般形式为：

```
switch ( 整型表达式 )
{
  case 常量 1:
  语句 1;                                    //← 可选
  break;                                     //← 可选
```

```
  case 常量 2:
  语句 2;                                    //← 可选
  break;                                     //← 可选
  default :                                  //← 可选
  语句 3;                                    //← 可选
  break;                                     //← 可选
}
```

案例 5.8 用 switch 语句实现案例 5.3。

```
#include<stdio.h>
int main()
{
    double v1, v2,result;
    char op;
    printf("input expression: ");
    scanf("%lf%c%lf", &v1, &op, &v2);
    printf("%.6lf%c%.6lf\n", v1, op, v2);
    switch(op)
    {
      case '+':
          result=v1+v2; break;
      case '-':
          result=v1-v2; break;
      case '*':
          result=v1*v2; break;
      case '/':
          result=v1/v2; break;
      default:
          printf("operator error!\n");
          result=0.0;
    }

    printf("%.6lf\n",result);
    return 0;
}
```

案例分析：

switch 语句的执行顺序是：判断紧跟 switch 关键字的表达式的值，当表达式的值与某一个 case 后面的常量表达式的值相等时，就执行此 case 后面的语句；如果遇到 break 语句，就结束整个 switch 语句；若所有的 case 中的常量表达式的值都没有与表达式的值匹配的，就执行 default 后面的语句。

使用 switch 语句时，要注意以下几点。

(1) switch 关键字后括号内的表达式为任意符合 C 语言语法规则的表达式，但其值只能是整数类型，其中包括字符型和枚举型类型。