CHAPTER 3

第3章 逻辑运算与程序控制

3.1 主要知识点

1. 关系表达式和逻辑表达式的计算及使用

1) 关系表达式

C 语言提供了 6 种关系运算符,如下所示。

(1) > 大于

(2) >= 大于或等于

(3) < 小于

(4) <= 小于或等于

(5) == 等于

(6) != 不等于

关系运算符的优先次序:

(1) 前 4 种关系运算符(>,>=,<,<=)的优先级相同,后两种相同。且前 4 种高于后 2 种。例如,"<"优先于"==",而"<"和">"优先级相同。

(2) 关系运算符的优先级低于算术运算符。

(3) 关系运算符的优先级高于赋值运算符。

用关系运算符将多个表达式(可以是算术表达式或关系表达式、逻辑表达式、赋值表达式、字符表达式)连接起来的式子,称为关系表达式。

关系表达式的运算结果是一个逻辑值,即"真"或"假"。C 语言用 1 来表示逻辑真,用 0 表示逻辑假。例如,a = 3, b = 2, c = 1, 则:

表达式"a > b"的值为"真",表达式的值为 1。

表达式"a > b == c"的值为"真",表达式的值为 1。原因"a > b == c"等效于"(a > b) == c",而 a > b 的值为 1,与 c 相等。

2) 逻辑表达式

C 语言提供了三种逻辑运算符:

(1) && 逻辑与,相当于汉语中的"并且"

(2) ||　　逻辑或,相当于汉语中的“或者”

(3) !　　逻辑非,相当于汉语中的“不是”

“&&”与“||”是双目运算符,要求有两个操作数,如 a&&b。而“!”是一目运算符,如 !a。逻辑与运算符只有当两个操作数都为“真”时,结果才为“真”,而逻辑或运算符则两个操作数中只要有一个为“真”,结果就为“真”;只有当两个操作数同时为“假”时,结果才为“假”。逻辑非相当于取反操作,具体的运算规则如表 3.1 所示。

表 3.1　逻辑运算的真值表

A	B	!A	!B	A&&B	A||B
1	1	0	0	1	1
1	0	0	1	0	1
0	1	1	0	0	1
0	0	1	1	0	0

逻辑运算符的优先次序:

(1) 逻辑运算符的优先级高于关系运算符。

(2) 逻辑非“!”优先级高于逻辑与“&&”,逻辑与“&&”的优先级高于逻辑或“||”。

(3) 逻辑非“!”的优先级高于算术运算符。

(4) 逻辑与“&&”和逻辑或“||”的优先级低于关系运算符。

用逻辑运算符将关系表达式连接起来的式子称为逻辑表达式。逻辑表达式的值是一个逻辑量“真”或“假”,用“1”和“0”表示。但是在判断一个量是否为“真”时,以“0”代表“假”,以非 0 代表“真”。就是将一个非零的数值认作为“真”——非零即真。

2. 单分支 if 语句、双分支 if 语句和多分支 if 语句

单分支 if 语句、双分支 if 语句和多分支 if 语句都属于基本程序控制结构中的选择结构,根据不同的条件来执行不同的程序段。具体的语句形式详见如下。

1) 单分支 if 语句

```
if(表达式 1)   语句 1
```

例如:if(x>y) printf("%d",x);

2) 双分支 if 语句

```
if(表达式 1)   语句 1
else  语句 2
```

例如:if(x>y) printf("%d",x);
    else printf("%d",y);

3) 多分支 if 语句

```
if(表达式 1)  语句 1
else  if(表达式 2)  语句 2
else  if(表达式 2)  语句 3
    …
else  语句 n
```

4) if 语句嵌套的一般形式

```
if(表达式 1)
{
     if(表达式 2)     语句 1
     else   语句 2
}
else if(表达式 3)
{
     if(表达式 4)     语句 3
     else   语句 4
}
else
{
        语句 5
}
```

以上 4 种形式需要注意的如下。

(1) 表达式 1,2,3 可以是任意表达式,如逻辑表达式、关系表达式,也可以是常量、变量。当表达式的值非 0 时代表“真”;0 时代表“假”。

(2) 语句 1,2,3,4,5 可以是复合语句,也可以只有一条语句。提示:在复合语句时注意用{ }括起来。

(3) 嵌套形式中每层的 if 与 else 配对,或用{ }来确定层次关系。即在多个 if-else 嵌套中,一个 else 应与它最近的一个且没有其他 else 配对的 if 组成配对关系。

(4) 最后一点也是最容易错的一点:表达式 1～4 中包含了关系运算符“==”时,容易少输入一个等号,当“=”的左操作数是变量时,原关系表达式就转化为赋值操作。这就改变了初衷,没有达到实际所要的效果。

3. switch 语句

在解决实际问题时,往往需要用到多分支的选择。虽然 if 语句的规则嵌套可以实现多分支的选择,但不够直观简洁,特别是在分支较多的情况下,if 语句的嵌套层次也会更深,从而增加了理解的难度,也不便于修改和扩充。为此,C 语言还提供了一个用于实现多分支选择的 switch 语句,用来解决多分支选择问题。switch 语句形式如下:

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

首先计算表达式的值,然后依次与每一个 case 中常量表达式的值进行比较,一旦发现了某个匹配的值,就执行该 case 后面的语句组直到执行了 break 语句为止。若没有匹配的值则执行 default 后面的语句组。

(1) 表达式：可以是整型、字符型或枚举型等表达式(表达式的值一定为整数)。如果是实型数据，系统会自动将其转换成整型或字符型。

(2) 常量表达式：可以是整数、字符等常量。每个 case 后的常量表达式的值必须互不相同。

(3) 语句组：可以由一条语句或复合语句构成，即使是复合语句也不需要{ }。

(4) 要求 switch 后面的表达式值的类型必须与 case 的常量表达式的类型要相同。

(5) 多个 case 可共用一组执行语句。例如：

```
…
case   'A':
case   'B':
case   'C':
    printf("score>60\n");
    break;
…
```

4. for 循环

循环是计算机解题的一个重要特征。由于计算机运算速度快，最适宜做重复性的工作。当我们在进行程序设计时，总是要把复杂的不易理解的求解过程转换为容易理解的操作的多次重复，从而降低了问题的复杂度，同时也减少了程序书写及输入的工作量。

```
for(表达式 1; 表达式 2; 表达式 3;)
  {
     语句序列;
    }
```

表达式 1：循环初始表达式，用于进入循环体前为循环变量赋初值，由算术、赋值、逻辑和逗号表达式构成。

表达式 2：循环控制表达式，用于控制循环体语句的执行次数，由关系表达式或逻辑表达式构成。

表达式 3：修改循环变量表达式，即每循环一次使得表达式 1 的值就要变化一次。由算术、赋值、逻辑或逗号表达式构成。

首先计算表达式 1 的值，再计算表达式 2 的值，表达式 2 的值为“真”(即非 0 值)，则执行循环体中各语句；然后计算表达式 3 的值，再判断表达式 2 的值，如此重复，直到表达式值为“假”(即 0 值)时，则跳出循环。

下面列举一下 for 循环的几种使用形式。

(1) 省略表达式 1，常用于无需给变量赋初值的情况。

(2) 省略表达式 2，则失去了对循环变量的控制，为此将导致无限循环，除非用 break 控制语句，将在第 8 部分讲解。

(3) 省略表达式 3，则失去了对循环变量的值的修改，为此，在循环体内必须设有替代表达式 3 的功能的语句。

(4) 省略表达式 1、3，这种格式完全等价于后面的 while 语句，即进入循环体前必须有赋初值语句，而且在体内要有修改循环变量的值的语句。

5. while 循环

```
while(表达式)
{
  语句序列;
}
```

先计算表达式的值并判断，若表达式值为“真”(即非 0 值)，则执行循环体中的语句；然后再计算再判断，如此重复，直到表达式值为“假”(即 0 值)时，则跳出循环。

条件表达式，可以是关系表达式，逻辑表达式，赋值表达式等，还可以是常量。

6. do-while 循环

```
do
  {
    语句序列;
  }while(表达式);
```

首先无条件地执行一次循环体中的各条语句，然后再判断表达式的值，若表达式值为“真”(即非 0 值)，则执行循环体中的语句；然后再计算再判断，如此重复，直到表达式值为“假”(即 0 值)时，则跳出循环。

表达式，可以是关系表达式，逻辑表达式，赋值表达式等，还可以是常量。

需要注意的是：while 圆括号后面要以“；”(分号)结束。

7. 循环结构的嵌套及注意事项

一个循环体内又包含另一个完整的循环结构，称为循环的嵌套，内嵌的循环中还可以嵌套循环即为多层循环。上面学过的三种循环(for 循环、while 循环和 do-while 循环)相互可以嵌套。嵌套的原则：不允许交叉。一般用得比较多的是两重循环，即双重循环。一般有如下几种形式。

形式一：

```
for( )
{
    for( )
    {     …     }
}
```

形式二：

```
while( )
{
    for( )
    {     …     }
}
```

形式三：

```
do
{
    …
    for(  )
    {     …     }
}while( );
```

形式四：

```
while(  )
{
    …
    while(  )
    { …}
    …
}
```

形式五：

```
for(  )
{
    …
    while(  )
    {
        …
    }
    …
}
```

无论采用哪种嵌套形式，都要层次清楚，不能出现交叉。外层循环每执行一次，内层循环都需要执行多次。嵌套循环执行的过程是：每进入一次外层循环，内层循环要按照赋初值、判断循环条件、执行内层循环体这三个过程进行，直到内层循环条件不成立；接着顺序执行外层循环体中内层循环后的其他语句，外层循环体执行结束后返回外层循环条件判断，再次循环，直至外层循环条件不成立。

8. 其他流程控制语句

前面我们介绍了三种能够实现循环的语句，它们退出循环的方式通常都是以某个表达式的结果作为判断条件，当其值为零时结束循环。

除了这种正常结束循环的方式外，还可以利用 C 语言提供的专门退出循环的语句：continue、break 和 goto。

1）continue 语句

格式：continue；

功能：结束本次循环，使程序回到循环条件，判断是否可以进入下一次循环。

2）break 语句

格式：break；

功能：循环体中遇见 break 语句，立即结束循环，跳到本层循环体外，执行循环结构后面的语句。

3）goto 语句

格式：goto 标号；

（1）goto 语句为无条件转向语句。goto 语句可以从循环体内跳出循环，尤其在多层循环中，使用 goto 语句可以跳到任意一层循环体内。

（2）标号的命名规则同变量名。

（3）goto 语句不符合结构化程序设计原则，一般不主张使用。

3.2 难点分析

1. 复杂关系表达式和逻辑表达式的计算

要掌握复杂关系表达式和逻辑表达式的计算，首先要牢记关系运算符和逻辑运算符的运算优先级与结合方向。

（1）关系运算符（>，>=，<，<=）的优先级相同，“==”和“!=”这两种优先级相同。且前 4 种高于后 2 种。

（2）逻辑非“!”的优先级高于算术运算符；算术运算符的优先级高于关系运算符；关系运算符的优先级高于逻辑与“&&”；逻辑与“&&”的优先级高于逻辑或“‖”；逻辑或“‖”的优先级高于赋值运算符。

即有如下所示的优先级关系：

当然，算术运算符中的优先级关系相信同学们已经熟记于心了。

在遇到同一优先级的运算符时，按顺序从左至右计算。

而逻辑与和逻辑或这两种运算符还有一个规律就是：在计算逻辑与运算时，若有左运算对象的值为 0，则不再继续计算逻辑与运算，并立即以 0 为逻辑与运算的结果；在计算逻辑或运算时，若有左运算对象的值为 1，则不再继续计算逻辑或运算，并立即以 1 为逻辑或运算的结果。在顺序计算逻辑表达式的过程中，一旦确定了表达式的最终结果，就不再继续计算。

一般双目运算符的结合方向都是自左至右。只有逻辑非，它属于单目运算符，它的结合方向是从右到左，即逻辑非的操作数只有一个，且此操作数在“!”的右侧。

C 语言中：

运算量（操作数）：0 表示“假”，非 0 表示“真”；

运算结果：0 表示“假”，1 表示“真”。

2. 选择结构和循环结构中条件的设置

无论是选择结构还是循环结构，其执行都依赖于条件的合理设置。

一般可以有以下几种形式的循环条件设置。

(1) 利用输入数据作为循环条件。

(2) 利用计数器自增量控制循环。

(3) 利用逻辑值作为循环条件，再通过循环体中添加特定条件下的 break 语句退出循环。

在选择结构中常出现如下两个等价条件：

```
if(x)   ⇔   if(x! = 0)
if(!x)   ⇔   if(x == 0)
```

另外数学中表示某个值处于某区间需要特别注意，比如要表示 a 的值在 10～100 之间，在数学中我们直接用 10＜a＜100 表示，但在 C 语言中这样的表示是错误的，应该表示为 a＞10&&a＜100。

3. for 循环、while 循环和 do-while 循环的比较

三种循环可以相互转换。如果你能正确的将三者进行转换，那说明你真正掌握了这三种循环。

(1) for、while 属当型循环，do-while 循环属直到型循环。

(2) for、while 当条件不成立时循环体可能一次也不执行，而 do-while 循环中循环体至少执行一次。

(3) for 循环常运用于循环次数确定的情况下。而 while 与 do-while 是循环次数不确定时常采用的方法。

(4) 在 for 循环的循环体中无须对循环变量进行修改，其他两种循环则必须在循环体中对循环变量进行修改。

(5) for 循环的初始条件可在表达式 1 中进行设置，其他两种循环则必须在进入循环之前进行设置。

4. break 和 continue 语句的使用

continue 语句只用于循环结构的内部，常与 if 语句联合起来使用，以便在满足条件时提前结束本次循环。

在循环体中 break 语句常与 if 语句搭配使用，并且 break 语句只能用在 switch 语句和循环语句中。break 只能跳出一层循环(或者一层 switch 语句结构)。

5. 避免死循环

避免死循环需要做到以下几点。

(1) 管理好程序中的循环控制变量。

(2) 使循环中的条件有机会为逻辑假。

(3) 在循环体中加入 break,并使它有机会执行。

(4) 在循环体中应该有使循环趋向结束的语句。

3.3 疑难问题解析

一、选择题

1. 若整型变量 a,b,c,d 的值依次为 1,4,3,2。则条件表达式(a < b ? a : b) < (c < d ? c : d)的值是_____。

A) 1　　B) 2　　C) 3　　D) 0

解析:解这道题的前提是掌握了条件表达式与关系运算符的计算规则,首先计算表达式 a < b ? a : b 的值,因 a<b 为真,所以此表达式的值为 a 的值,即 1;再计算表达式 c < d ? c : d 的值,因 c<d 为假,所以 c < d ? c : d 的值为 d 的值,即 2。这样就可以将题目中的条件表达式转为等价的表达式 a<d,由已知条件得 a<d 为真,所以值为 1,所以本题的正确答案为 A。

2. 请从下列表达式中选出当 a 为奇数时,值为 1 的表达式_____。

A) a%2 == 0　　B) a%2

C) a%2 - 1 ! = 0　　D) a/2 * 2-1 == 0

解析:当 a 为奇数时,来计算一下 4 个选项的值。A 选项 a%2==0,由于 a 是奇数,所以 a%2 的值为 1,所以 1==0 这个表达式结果为假,即 A 选项中表达式的值为 0。再看 B 选项,由 A 选项的分析可知 B 选项中表达式的值为 1。到这里应该就可以确定本题答案了,但是为了保险一点,我们还是把 C 选项和 D 选项再确认一下。其中 C 选项中 a%2-1 我们可以由之前的分析得到结果为 0,而 0! =0 为假,所以 C 选项的值为 0。再看 D 选项中关系表达式的左操作数 a/2 * 2-1 不可能为 0,所以关系表达式的值为假。这样本题的答案就非常确定了,为 B。

3. 执行下列程序段后,输出的结果是_____。

```
int i = 5;
while(i = 1)
{   i--;
}
printf("%d", i);
```

A) 5　　B) 0　　C) 1　　D) 无限循环

解析:解这道题时一定要非常仔细,原因在于 while 条件中本来的意图是想让 i 与 1 做是否相等“==”的比较,但是由于少输入了一个等号,就变成了赋值表达式,即第一次进入循环前判断条件时将 1 赋给了 i,这样 while(i = 1){…}就转为 while(i){…},i 为 1 所以条件成立,进而进入循环体,执行“i--;”语句,之后再判断条件,每次的条件是一样,如此重复,为死循环。所以本题的答案为 D。

4. 在 C 语言中,以下 switch-case 语句片段的运行结果是_____。

```
int i = 2;
```

```
switch(i)
   { case 1:
           printf("I'm first!");
     case 2:
           printf("I'm second!");
     case 3:
           printf("I'm third!");
}
```

A）输出结果为："I'm second!"

B）输出结果为："I'm second!"和"I'm third"

C）输出结果为："I'm third!"

D）出现编译错误

解析：由于 i 的值为 2，所以与第二个 case 匹配，首先会输出"I'm second!"，而 printf 语句后没有 break，所以继续执行 case 3 中的语句，输出"I'm third"。所以本题的正确答案是 B。本题代码中 switch-case 语句犯的错误比较常见，初学者往往忘记在需要 break 的 case 中添加 break 语句。

5. 分析如下所示的代码，编译运行后的输出结果是_____。

```
#include <stdio.h>
void main()
{    int a = 10, b = 20; c = 30;
     if(a>b || b>c)
           printf("%d", a);
     else if (a<b && b<c)
           printf("%d", b+c);
     else
           printf("%d", a+b+c);
}
```

A）10　　B)50　　C）60　　D）20

解析：先分析代码结构，主要由 if－else 多分支选择结构构成。if 条件中 a＞b 值为假，b＞c 的值也为假，中间为逻辑运算符逻辑或，所以整个 if 条件为假。再看 else if 条件 a＜b 与 b＜c 两个关系表达式的值都为真，相与，结果为真，所以执行 else if 分支，输出 b＋c 的值。所以本题的正确答案是 B。

6. 编译运行如下代码，其输出结果是_____。

```
#include <stdio.h>
void main()
{  for(int i = 0; i<10; i++)
       {    if(i%2 != 0)
                  continue;
            printf("%d", i);
       }
}
```

A）13579　　B）02468　　C）97531　　D）86420

解析：main()函数中有 for 循环，其中嵌套了一个 if 单分支结构。做这题有一个技巧，