第3章 选择结构程序设计

本章简介

- 关系运算和逻辑运算；
- 用逻辑表达式表示条件；
- 用 if 语句实现选择；
- 用 switch-case 语句实现选择；
- 选择结构的嵌套；
- 复合语句的使用。

大多数程序设计都会遇到选择结构。选择结构是指对给定的条件进行判断，然后根据判断结果来选择执行不同的操作。在 C 语言中，选择结构主要是通过 if 语句和 switch 语句来实现的。其中，选择条件的真假通常是对若干判断条件进行关系运算和逻辑运算的结果。本章首先介绍关系运算和逻辑运算，然后介绍选择结构的不同实现方式。

3.1 逻辑表达式

在选择结构中通常需要测试表达式是“真”还是“假”。例如，对于表达式 a＜b，真值将说明 a 是小于 b 的。在许多编程语言中，类似 a＜b 这样的表达式具有特殊的“布尔”类型或“逻辑”类型，其值为“真”或“假”。而在 C 语言中，诸如 a＜b 这样的比较运算会产生两个整数：0 或 1。0 代表“假”(条件不成立)，1 代表“真”(条件成立)。下面来看一下构成逻辑表达式的运算符。

3.1.1 关系运算

关系运算实际上就是一种“比较运算”，即用关系运算符对两个运算量进行比较，比较它们之间的“大小关系”。例如关系表达式 a＞3 用于判断变量 a 的值是否大于 3。

关系运算符可以用于比较整数和浮点数，也允许比较混合类型的操作数。C 语言的关系运算符和数学上的＜、＞、≤、≥运算符号相对应，只是个别符号的书写形式有所不同，如表 3-1 所示。

表 3-1 C 语言中的关系运算符

符　号	含　义	样　例
<	小于	3<6 运算结果为 1
>	大于	2.7>5.4 运算结果为 0
<=	小于等于	'A'<='B'运算结果为 1
>=	大于等于	'a'>='A'运算结果为 1

字符之间根据对应 ASCII 码值大小进行比较。

【例 3-1】 关系运算示例。求 x=7<3<5 的值。

表达式 x=7<3<5 相当于

$$\begin{aligned} x &= (7 < 3) < 5 \\ &= 0 < 5 \\ &= 1 \end{aligned}$$

因此，x 最后获得的值为 1。

需要注意的是，表达式 x<y<z 在 C 语言中是合法的，但可能并不是用户所期望的含义。这个表达式等价于(x<y)<z，即首先检测 x 是否小于 y，然后用比较后产生的结果(1 或 0)来和 z 进行比较。所以这个表达式并不是测试 y 是否介于 x 和 z 之间。

关系运算的结果为 0 或 1，0 代表条件不成立(假)，1 代表条件成立(真)。

在某些场合下，为了明确运算顺序，增强程序的可读性，最好在需要的地方添加括号。例如 c=(a<=b)表示把关系表达式 a<=b 的结果赋给变量 c，它显然比 c=a<=b 直观明确。

【例 3-2】 判断成绩是否及格。

源程序 ex3_2.c

```
#include<stdio.h>
int main()
{
int grade;
scanf("%d",&grade);
if(grade>=60)                    /*判断成绩是否大于等于 60*/
  printf("Passing Grate\n");     /*若条件成立，则输出及格信息*/
return 0;
}
```

运行结果

```
78
Passing Grate
```

本例通过关系表达式(grade>=60)来判断一个给定的成绩是否大于等于 60，如果条件成立，则输出及格的信息。

关系表达式主要用于选择结构中的条件判断，选择结构的内容将在 3.2 节详细介绍，本例中提前出现了相关的内容。但由于 C 语言的书写比较接近自然语言，因此不会影响对本例的理解。

3.1.2　判等运算

C语言中的判等运算符也属于关系运算符，但它们有着比较特殊的形式，如表3-2所示。

表3-2　C语言中的判等运算符

符　　号	含　　义	样　　例
==	等于	5==6　运算结果为0
!=	不等于	3!=5　运算结果为1

由于一个等号在C语言中已经用来表示赋值运算符了，所以“等于”运算符就用相邻的两个等号“==”来表示。“不等于”运算符也是由两个字符“!”和“=”组成的。

【例3-3】 判等运算示例。已知ch='a'，求x=(ch=='b')的值。

表达式x=(ch=='b')相当于

```
x =('a'=='b')
  =0
```

即，x最后获得的值为0。

本例首先判断表达式(ch=='b')的值，由于ch存储的是字符常量'a'，显然与字符常量'b'不相等，判断结果不成立，为0，因此变量x获得的结果为0。

【例3-4】 简单密码判断。从键盘输入一个数字，与内部设定的密码数字相比较，如果一致则给出OK信息。

源程序 ex3_4.c

```
#include<stdio.h>
int main()
{
  int password=367,guess;
  scanf("%d",&guess);
  if(guess==password)          /*判断输入的数字与内部设定的密码是否一致*/
     printf("OK\n");           /*若一致则输出OK信息*/
  return 0;
}
```

运行结果

```
367
OK
```

本例判断两个数字是否相等，需要使用等于运算符“==”，而不能写成赋值运算符“=”。语句if(guess==password)用来测试guess是否等于password，而如果写成语句if(guess=password)，则是先把password的值赋给guess，然后测试guess是否是一个非零的数值。在这种情况下，很多编译系统不会出现错误提示，但运行结果往往就背离了用户的期望。

不要混淆==(等于)运算符和=(赋值)运算符，这是最容易出现的C编程错误之一。

在第 2 章介绍数据类型时，曾经提到，浮点型数据在存储时会有误差，因此在用关系运算符直接对它们进行比较时，可能会得出错误的结果，下面来看一个简单的例子。

【例 3-5】 浮点数的判等运算。

源程序 ex3_5.c

```
#include <stdio.h>
int main()
{
    double d0 = 0.3;
    double d1 = 0.1;
    double d2 = 0.2;
    printf ("%d\n", d0 == (d1 + d2));
    return 0;
}
```

运行结果

```
0
```

运行结果为 0，说明 d0 与(d1＋d2)不相等，这显然不符合实际情况，这种错误是由浮点数的存储误差造成的。因此，一般应避免对两个浮点数直接进行判等运算，而是采用判断两者的差的绝对值是否小于某个很小的数来实现，例如：

```
x == y  可写成：fabs(x - y)< 1e - 6
```

表示，如果 x 与 y 的差值非常小(小于 10^{-6})，那就可以认为 x 与 y 是相等的。

fabs()是 C 语言标准库中求浮点数绝对值的函数，具体用法见附录 D。

3.1.3 逻辑运算

关系表达式只适用于描述单一的条件，对于较复杂的复合条件就需要将若干个关系表达式连接起来才能描述，如描述“x 大于 0 且不等于 5”，需要将两个关系表达式 x＞0 和 x!＝5 连接起来。

实现多个关系表达式的连接需要用到逻辑运算符，C 语言提供了 3 个逻辑运算符，如表 3-3 所示。其中 &&(逻辑与)和||(逻辑或)是双目运算符，它要求有两个操作数，!（逻辑非)是单目运算符，只要求有一个操作数。

表 3-3 C 语言中的逻辑运算符

符 号	含 义	样 例
!	逻辑非	!(math＜＝60) 如果数学成绩≤60，结果为 0
&&	逻辑与	math＞90&&science＞90 如果数学和科学的成绩都大于 90，结果为 1
||	逻辑或	math＞90||science＞90 数学或科学的成绩只要有一门大于 90，结果为 1

- 逻辑非的用法：！x 为真(当且只当 x 为假)。
- 逻辑与的用法：x&&y 为真(当且只当 x 和 y 都为真)。
- 逻辑或的用法：x||y 为真(当且只当 x 和 y 中至少有一个为真)。

表 3-4 为逻辑运算的真值表，表示当 a 和 b 的值为不同的组合时，各种逻辑运算所得到的值。

表 3-4　逻辑运算的真值表

a	b	! a	! b	a&&b	a||b
0	0	1	1	0	0
0	非 0	1	0	0	1
非 0	0	0	1	0	1
非 0	非 0	0	0	1	1

C 语言把非零数据当作“真”，把零当作“假”看待。

实际上，逻辑运算符两侧的运算对象不但可以是 0 和 1，或者是 0 和非 0 的整数，也可以是任何类型的数据。系统最终以 1 和 0 来判断它们属于“真”还是“假”。表 3-5 给出了几种逻辑运算的样例。

表 3-5　各种逻辑运算示例

逻辑表达式	结果	说　明
! 4	0	4 为非 0，被认作“真”，进行“非”运算，得“假”，用 0 表示
5&&6	1	5 和 6 均为非 0，被认作“真”，“与”运算的结果也为“真”，用 1 表示
'a'&&'b'	1	'a'与'b'的 ASCII 码都不为 0，按“真”处理，“与”的结果也为真，用 1 表示
4||0	1	4 为非 0，“或”运算中只要有一个非 0，结果就是非 0，用 1 表示
! 8||0	0	8 为非 0，“非”运算后！8 的结果为 0，再与 0 进行“或”运算，结果为 0

3.1.4　用逻辑表达式表示条件

要解决编程问题，经常需要将用文字或数学公式描述的条件转换为用 C 语言表示的表达式。许多算法步骤需要检测某个变量的值是否位于指定的取值范围内。例如，假设用 min 表示取值范围的下限，max 表示取值范围的上限(min<max)，如果要表示 x 的取值范围在 min 和 max 之间(含)，则数学上可以用以下表达式来表示：

$$\min \leqslant x \leqslant \max$$

而写成 C 语言的表达式就应该是：

```
min <= x&&x <= max
```

图 3-1 用阴影表示了 x 的取值范围，如果 x 位于该范围内，表达式值为 1，否则表达式值为 0。

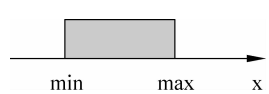

图 3-1　表达式“min<=x&&x<=max”为 1 时 x 的取值范围

表 3-6 给出了一些条件的描述以及对应的 C 语言表达式。

【例 3-6】 判断大写字母。根据 ASCII 码的排列规律，如果某字符处于大写字母'A'以及大写字母'Z'之间，则可以确定它是一个大写字母。

表 3-6 用 C 语言表达式表示条件

条　　件	逻辑表达式
x 和 y 都大于 z	x>z&&y>z
x 位于 u 到 v 之间(含)	u<=x&&x<=v
x 位于 u 到 v 之外	x<u||x>v
x 能被 y 整除,但不能被 z 整除	x%y==0&&x%z!=0
x 能被 y 整除,又能被 z 整除	x%y==0&&x%z==0

源程序 ex3_6.c

```
#include<stdio.h>
int main()
{
    char ch;
    ch=getchar();                  /*从键盘读入一个字符*/
    if(ch>='A'&&ch<='Z')           /*测试字符 ch 是否大写字母*/
      printf("Upper case\n");
    return 0;
}
```

运行结果

```
G
Upper case
```

本例需要确认字符 ch 是否在大写字母'A'和'Z'之间,即是否满足'A'≤ch≤'Z'。在 C 语言中可用“与”运算符将 ch>='A'和 ch<='Z'连接起来,即(ch>='A'&&ch<='Z'),或者('A'<=ch&&ch<='Z')也可以。但切记不能按数学上的习惯直接将公式写成('A'<=ch<='Z')。表 3-7 对这两种写法进行了分析,用不同的测试数据来演示不同的写法对运算过程和运算结果的影响。

表 3-7 条件'A'≤ch≤'z'的 C 表达式解析

逻辑表达式	运算规律	测试数据	说　　明
(正确写法) 'A'<=ch&& ch<='Z'	先计算'A'<=ch,再计算 ch<='Z',最后将两个计算结果进行“与”运算	ch='G'	'A'<=ch 结果为 1,ch<='Z'结果为 1,两个 1 进行“与”运算,结果也为 1。结果正确
		ch='a'	'A'<=ch 结果为 1,ch<='Z'结果为 0,1 和 0 进行“与”运算,结果为 0。结果正确
(错误写法) 'A'<=ch<='Z'	先计算'A'<=ch,再计算该结果是否小于等于 Z,即('A'<=ch)<='Z'	ch='G'	'A'<=ch 结果为 1,1<='Z'结果为 1。结果正确
		ch='a'	'A'<=ch 结果为 1,1<='Z'结果为 1。结果错误

根据表 3-7 的分析可知,判断字符是否大写字母时,如果将表达式误写成'A'<=ch<='Z'的形式了,那么其计算规则就不一样了,当测试的字符为大写字母时,也能误打误撞地出现结果正确的情况,但当测试字符为小写字母时,该表达式也将其判断为大写字母了,这就出现错误了。因此需要正确书写表达式,同时在测试程序时要多测试各种不同数

据，以发现程序中隐藏的问题。

根据前面的分析可知，如果要判断某一字符 ch 是否为小写字母，正确的逻辑表达式应为'a'<=ch&&ch<='z'，如果要判断某一字符是否为数字字母，正确的逻辑表达式为'0'<=ch&&ch<='9'。

3.1.5　短路求值

在逻辑表达式的求解中，并不是所有的逻辑运算符都需要执行，有时只需要执行一部分运算符就可得出逻辑表达式的最后结果，这时就不会继续运算下去，这一现象被称为“短路求值”，&& 和||就是所谓的“短路”运算符。例如：

```
x&&y
```

只有当 x 为真时，才需要判断 y 的值。若 x 为假，就立即得出整个表达式为假，则不需要再判断 y 的值了。又如：

```
x||y
```

只要 x 为真，就立即得出整个表达式为真，就不必判断 y 了。当 x 为假时，才需要判断 y 的值。

【例 3-7】 短路求值分析。

源程序 ex3_7.c

```
#include<stdio.h>
int main()
{
    int a=1,b=2,c=3,d=4;
    int m=1,n=1,t;
    t=(m=a>b)&&(n=c>d);
    printf("t=%d,m=%d,n=%d\n",t,m,n);
    return 0;
}
```

运行结果

```
t=0,m=0,n=1
```

对于逻辑表达式(m=a>b)&&(n=c>d)，首先计算(m=a>b)，由于 a>b 的结果为 0，因此 m=0，这时可立即得出整个逻辑表达式的结果 t 为 0，因此(n=c>d)不被执行，n 的值就不会发生变换，依然保持 1。

短路求值：只要逻辑表达式的值能够确定就停止对表达式求值。

3.2　if 语句

if 语句通过对给定的条件进行分析、比较和判断来完成不同的操作。C 语言的 if 语句有 3 种表现形式：有一个选项的 if 语句(if 语句)；有两个选项的 if 语句(if-else 语句)；有

多个选项的 if 语句(if-else-if 语句)。

3.2.1 一个选项的 if 语句

在前面的例 3-2 中出现了以下语句：

```
if(grade>=60)
  printf("Passing Grate\n");
```

这就是一个选项的 if 语句。其流程图如图 3-2 所示。

一个选项的 if 语句的用法如表 3-8 所示。

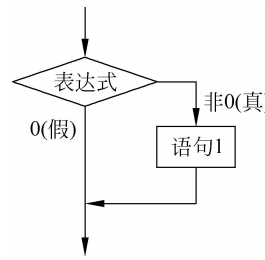

图 3-2 if 语句流程图

表 3-8 一个选项的 if 语句的用法

语 法	样 例	说 明
if(表达式) 语句 1	if(grade>=60) printf("Passing Grate\n");	(1) 首先计算圆括号内表达式的值,若为非 0 值,则执行语句 1,然后脱离本选择结构,继续执行后面的语句；否则绕开语句 1,直接脱离本选择结构去执行后面的语句 (2) 圆括号内的表达式一般是关系表达式或逻辑表达式,但也可以是其他表达式(任意的数值类型表达式)。若表达式值为 0,按“假”处理；若表达式值非 0,按“真”处理。例如： if(1) printf(“That's OK!”); 该 if 语句是合法的,因为表达式的值为 1,非 0,按“真”处理,结果输出“That's OK!”

【例 3-8】 求整数的绝对值。

由于正数和零的绝对值就是其本身,因此不需要处理,而负数的绝对值则可通过类似 x=－x 的形式求取,程序如下。

源程序 ex3_8.c

```
#include<stdio.h>
int main()
{
    int a,b;
    printf("Enter a integer:");
    scanf("%d",&a);
    b=a;            /*将输入的数值保存到变量 b 中*/
    if(b<0)
      b=-b;         /*对 b 进行判断,若是负数,则取反*/
    printf("The %d's absolute value is %d\n",a,b);
    return 0;
}
```

运行结果

```
Enter a integer:-9
The -9's absolute value is 9
```

```
Enter a integer:9
The 9's absolute value is 9
```

对本例分别用一个负整数和一个正整数进行了测试，当输入的整数小于 0 时，关系表达式 b＜0 成立，表达式值为 1，因此执行 b＝－b，改变 b 的符号，使 b 为正值，然后脱离选择结构去执行后面的输出语句；否则，关系表达式 b＜0 不成立，表达式值为 0，因此不执行 b＝－b 操作，直接脱离选择结构去执行输出语句。

从这个例子可以看出，当 b 小于 0 时，需要进行改变符号的操作，而当 b 大于等于 0 时，就不需要进行任何操作。类似这样的，当条件成立时需要执行某种操作，而当条件不成立时不需要进行任何操作的任务求解，可以用一个选项的 if 语句来进行程序的设计。

在 if(表达式)的圆括号后面不能加分号。如：

```
if(a<b);
min=a;
```

语法上没有问题，但由于在 if(a＜b)后面加了分号，使 if 语句到此结束，即 a＜b 时执行空操作，因此代码的意思可能已经和原来的设计相背离了。同样的错误也可能发生在下一章将要介绍的循环语句 for 和 while 中，请予以注意。

3.2.2　两个选项的 if 语句

在例 3-2 中，如果希望当成绩大于等于 60 分时给出合格信息，小于 60 分时也能给出不合格信息，则需要用到两个选项的 if 语句。

【例 3-9】 成绩合格性判断。判断学生成绩是否合格：当分数大于等于 60 分时，输出合格信息，在 60 分以下的，输出不合格信息。

源程序 ex3_9.c

```
#include<stdio.h>
int main()
{
    int grade;
    scanf("%d",&grade);

    if(grade>=60)                  /*判断成绩是否大于等于 60*/
    printf("Passing Grate\n");     /*若条件成立,则输出及格信息*/
    else                           /*否则表示前面表达式里的条件不成立,即成绩小于 60*/
    printf("Failure\n");           /*因此输出不合格信息*/

}
```

运行结果

```
60
Passing Grate
```

```
59
Failure
```

这个例子用到了两个选项的 if 语句，即 if-else 语句，其流程图如图 3-3 所示。

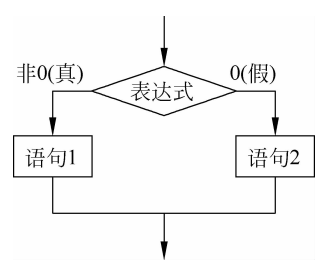

图 3-3　if-else 语句流程图

if-else 语句是选择结构的标准使用形式，其用法如表 3-9 所示。

表 3-9 if-else 语句的用法

语　法	样　例	说　明

if(表达式)
　语句 1
else
　语句 2
	if(grade>=60)
	　printf ("Passing Grate\n");
	else
	　printf("Failure\n);
		(1) 首先计算圆括号内表达式的值,若为非 0 值,则执行语句 1,然后脱离本选择结构,继续执行后面的语句;否则执行语句 2,然后脱离本选择结构去执行后面的语句
		(2) 无论表达式的值为 0 还是非 0,只执行语句 1 和语句 2 中的某一个,不会两个都执行
		(3) else 子句不能单独使用,它前面必须要有 if 配对使用

【例 3-10】 计算以下分段函数的值:

$$y=\begin{cases} x^2-2 & x\geqslant 0 \\ \sqrt{5-x} & x<0 \end{cases}$$

源程序 ex3_10.c

```
#include<stdio.h>
#include<math.h>
int main()
{
  double x,y;
  scanf("%lf",&x);
  if(x>=0)
    y=x*x-2;
  else
    y=sqrt(5-x);
  printf("x=%.2f,y=%.2f\n",x,y);
  return 0;
}
```

运行结果

```
4
x=4.00,y=14.00
```

```
-4
x=-4.00,y=3.00
```

本例中,x 的取值范围有两个区间,x≥0 区间以及 x<0 区间。当 x≥0 时,取值范围如图 3-4 所示,而当 x<0 时,取值范围刚好是表达式 x≥0 的取反,即除去 x≥0 以外的所有区间。因此,在 if-else 语句中,如果表达式 x≥0 不成立,则说明 x 是小于 0 的,就不需要再刻意强调 x<0 这一条件了。

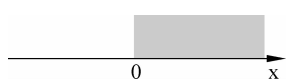

图 3-4　例 3-10 中 x≥0 的取值范围

【例 3-11】 两数求大值。用 if-else 语句求出两数中较大的数值。

源程序 ex3_11.c

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