第3章 分支结构程序设计

本章将介绍的内容

基础部分：

- 关系运算符、逻辑运算符及其表达式。
- 实现分支结构的 if 语句和 switch 语句。
- 分支结构的流程图和调试程序的方法(参见例 3.6)。
- 贯穿实例 A 的部分程序。

提高部分：

- 进一步学习 if 语句和 switch 语句。
- 条件运算符及其表达式。
- 贯穿实例 B 的部分程序。

各例题的知识要点

例 3.1　关系表达式。

例 3.2　逻辑表达式。

例 3.3　特殊的逻辑表达式。

例 3.4　不带 else 的 if 语句。

例 3.5　用不带 else 的 if 语句求分段函数值。

例 3.6　输出 3 个数中的最大数，单步执行程序的方法。

例 3.7　3 个数的冒泡排序法。

例 3.8　带 else 的 if 语句。

例 3.9　嵌套 if 语句的概念。

例 3.10　用嵌套 if 语句计算分段函数值。

例 3.11　switch 语句的概念。

例 3.12　switch 语句的应用：将成绩分为 5 个等级。

(以下为提高部分例题)

例 3.13　if 子句和 else 子句中均包含另一个 if 语句的嵌套 if 语句。

例 3.14　else 与 if 的配对。

例 3.15　在 switch 语句中 break 语句的作用和 default 的位置。

例 3.16　嵌套的 switch 语句。

例 3.17　条件运算符的使用。

3.1　关系运算符和关系表达式

在分支结构程序设计中，经常使用的运算符是关系运算符和逻辑运算符。下面先介绍关系运算符。

3.1.1　关系运算符

C 语言中提供的关系运算符共有 6 种，如表 3.1 所示。

表 3.1　关系运算符

运算符	含义	优先级的数值	举 例	例题含义
＞	大于	6	x＞0	x 的值是否大于 0
＞＝	大于等于	6	x＞＝0	x 的值是否大于等于 0
＜	小于	6	x＜0	x 的值是否小于 0
＜＝	小于等于	6	x＜＝0	x 的值是否小于等于 0
＝＝	等于	7	x＝＝0	x 的值是否等于 0
!＝	不等于	7	x!＝0	x 的值是否不等于 0

关系运算符＞、＞＝、＜、＜＝的优先级高于＝＝、!＝，结合方向是自左至右。关系运算符隐含“是否”的含义，例如，“x＞0”隐含 x 的值是否大于 0。

3.1.2　关系表达式

x＞0 和 x＝＝0 都是关系表达式。用关系运算符把两个 C 语言表达式连接起来的表达式称为关系表达式。关系运算符的判断结果只有两种可能：“真”或“假”。当关系成立时结果为“真”，否则结果为“假”。

【例 3.1】 假设表 3.2 中 a、b、x 为整型变量，y 为单精度型变量，请观察输出结果。

【解】 请参见表 3.2 的输出结果。

说明：

(1) 当关系表达式的判断结果为“真”时，关系表达式的值为 1，当判断结果为“假”时，关系表达式的值为 0，即关系表达式的值只能是整数 0 或 1。关系表达式能参加数值计算，例如，表达式(5＞3)＋7 的值为 8(即 1＋7)。

表 3.2　关系表达式举例

x 的值	举　例	关系表达式的判断	关系表达式的值	输出
x=1	printf("%d",x>0);	x>0 为“真”	x>0 的值为 1	1
x=1	a=x==0; printf("%d",a);	x==0 为“假”	x==0 的值为 0	0

x=3	a=x>0;
	b=x<5;
	printf("%d",a==b);
		x>0 为“真”
		x<5 为“真”
		a==b 为“真”
			x>0 的值为 1
			x<5 的值为 1
			a==b 的值为 1
				1

x=1	printf("%d", 0<=x<=2);	0<=x<=2 为“真”	0<=x<=2 的值为 1	1
x=-3	printf("%d", 0<=x<=2);	0<=x<=2 为“真”	0<=x<=2 的值为 1	1
x=3	a=(x<=5)+2; printf("%d",a);	x<=5 为“真”	x<=5 的值为 1	3
y=45.3219	printf("%d", y==45.3219);	y==45.3219 为“假”	y==45.3219 值为 0	0

(2) 关系运算符的结合方向为自左至右。计算表达式 0<=x<=2 的值时,先计算表达式 0<=x 的值,不管 x 取什么值,表达式 0<=x 的值只能是 0 或 1,表达式 0<=x<=2相当于 0<=2 或 1<=2,且这两个表达式的值都是 1,因此表达式 0<=x<=2 的值永远是 1,这说明表达式 0<=x<=2 不能代表 x 的取值范围 0≤x≤2,那么,应如何表示这范围呢? 读者在下一节的学习中将得到答案。

(3) 存放在内存中的实型数总是有误差。当把 45.3219 存放在单精度型变量 y 中时,y 的实际值就会变为 45.3219 *(* 代表若干个不确定的数字),关系表达式 y==45.3219 总为假,也就是其值永远为 0,与预期结果不相符,所以应当避免使用判断“实型数==实型数”这样的关系表达式,可以采用 y−45.3219<10^{-6}等形式代替 y==45.3219。

3.2　逻辑运算符和逻辑表达式

3.2.1　逻辑运算符

C 语言中提供的逻辑运算符共有 3 种,如表 3.3 所示。

表 3.3　逻辑运算符

运算符	含义	优先级的数值	结合方向	举　例
&&	逻辑与	11	自左至右	x>=0 && x<=2
||	逻辑或	12	自左至右	x<-3 || x>3
!	逻辑非	2	自右至左	!(x>3)

请注意，逻辑运算符的逻辑量(即运算量)可以是任意一个合法的表达式(可以是常量或变量)。运算符 && 和 || 的结合方向是自左至右，而！的结合方向是自右至左。要输入运算符 ||，只要输入两次反斜杠(\)的上档键即可。

3.2.2 逻辑表达式

【例 3.2】 编写一个含有逻辑表达式的程序。

【解】 程序如下：

```
#include <stdio.h>
main( )
{    int x=1;

     printf("%d ",x>=0 && x<=2);          //x 满足 0≤x≤2,输出 1
     x=5;
     printf("%d ",x>=0 && x<=2);          //x 不满足 0≤x≤2,输出 0
     printf("%d ",x<-3||x>3);             //x 不满足 x<-3 但满足 x>3,输出 1
     x=0;
     printf("%d ",x<-3||x>3);             //x 不满足 x<-3 也不满足 x>3,输出 0
     printf("%d ",!x);                    //x 的值为 0,输出 1
     x=5;
     printf("%d ",!x);                    //x 的值为非 0,输出 0
     printf("%d ",3 && 'A');              //两个运算量为非 0 数,输出 1
     printf("%d ",(x=2)||0);              //第 1 个运算量为非 0 数,输出 1
     printf("x=%d\n",x);
}
```

运行结果：

```
1 0 1 0 1 0 1 1 x=2
```

程序说明：

(1) 在 C 语言中，任何一个非零值都表示“真”，零表示“假”。例如，'A'、5>=0 和 x=2 都表示“真”，因为 3 项的值分别为非零数 65、1、2(均非零)。

(2) 由程序的运行结果可以看出，逻辑表达式的值也只能是 1 或 0。当逻辑运算的结果为“真”时值为 1，为“假”时值为 0。

逻辑运算的规则如表 3.4 所示，表中 a 和 b 代表运算量，它们可以是任意表达式。

表 3.4 逻辑运算的规则

运算符	表达式	举　例	逻辑运算的规则	表达式的值
&&	a && b	(5>1) && 3	两个运算量都为真，结果为真	1
	a && b	(5>1) && (3==2)	至少有一个运算量为假，结果为假	0

续表

运算符	表达式	举　例	逻辑运算的规则	表达式的值
||	a || b	(2==3) || (5>3)	至少有一个运算量为真,结果为真	1
	a || b	0 || (3>5)	两个运算量都为假,结果为假	0
!	!a	!(x=0)	运算量为假,结果为真	1
	!a	!5	运算量为真,结果为假	0

【例 3.3】 编写一个含有特殊逻辑表达式的程序。

【解】 程序如下:

```
#include <stdio.h>
main( )
{    int a=1,b=0;

     printf("%d ",0&&(a=2));          //0 为"假",不执行 a=2
     printf("a=%d ",a);               //a 的值仍为 1
     printf("%d ",5&&(a=2));          //5 为非 0 数,是"真",要执行 a=2
     printf("a=%d ",a);               //a 的值为 2
     b=1;
     printf("%d ",5||(b=2));          //5 为非 0 数,是"真",不执行 b=2
     printf("b=%d ",b);               //b 的值仍为 1
     printf("%d ",0||(b=2));          //0 为"假",要执行 b=2
     printf("b=%d\n",b);              //b 的值为 2
}
```

运行结果:

```
0  a=1  1  a=2  1  b=1  1  b=2
```

程序说明:

程序中可以看到,表达式 a=2 和 b=2 有时不被处理。在逻辑表达式的求解过程中,并不是所有的表达式都被运算。例如,进行"逻辑与"运算时,如果第 1 个表达式为"假",系统就不再对第 2 个表达式做运算,因为此时已经可以确定逻辑表达式的值为 0;进行"逻辑或"运算时,如果第 1 个运算量为"真",系统也不再对第 2 个运算量做运算,因为此时已经可以确定逻辑表达式的值为 1。

算术运算符、赋值运算符、关系运算符、逻辑运算符参加运算的先后顺序是:

逻辑非运算符→算术运算符→关系运算符→逻辑与运算符→逻辑或运算符→赋值运算符

3.3　if 语 句

到目前为止,所介绍的程序都属于顺序结构,顺序结构程序中的所有语句都将被执行一次。但是在实际应用中,常常需要根据不同情况选择不同的执行语句,这时需要设计分

支结构，例如，学生成绩不低于 60 分就算通过，否则按不通过处理。在 C 语言中，通常用 if 语句、switch 语句或条件表达式解决分支结构问题。本节将介绍 if 语句和 switch 语句，在 3.5.2 节中介绍条件表达式。分支结构逻辑性较强，使用时有一定的难度，希望读者认真学习本章介绍的所有内容，并自己动手编写程序，加强上机实践。

3.3.1 if 语句的一般形式

if 语句有两种形式。

1. 不带 else 的 if 语句

【例 3.4】 任意输入两个整数存放在变量 a、b 中，输出时保证 a 中的值不比 b 中的值大。编写程序实现其功能。

【解】 编程点拨：

为了输出时保证 a 中的值不比 b 中的值大，当 a＞b 时，需要交换 a 和 b。程序如下：

```
#include <stdio.h>
main( )
{    int a=0,b=0,t=0;

     printf("Input a,b:");   scanf("%d%d",&a,&b);
     if(a>b)                      //如果 a 的值比 b 的值大，交换 a 和 b 的值
     {    t=a;
          a=b;
          b=t;
     }
     printf("a=%d,b=%d\n",a,b);
}
```

第 1 次运行结果：

```
Input a,b:2 5
a=2,b=5
```
（a 和 b 的值不变）

第 2 次运行结果：

```
Input a,b:5 2
a=2,b=5
```
（a 和 b 的值被交换）

程序说明：

（1）从以上两次的运行情况可以看出，3 条语句“t＝a；a＝b；b＝t；”是 if 的子句，当表达式 a＞b 为“真”时，要执行“t＝a；a＝b；b＝t；”，为“假”时，不执行。

（2）由于 3 条语句“t＝a；a＝b；b＝t；”是 if 语句的一部分，因此要求像本程序中那样，书写时缩进几个格，以此体现它们之间的隶属关系，提高程序的可读性，这一点对初学者十分重要。

（3）由于本题有两种可能，所以在得到第 1 次运行结果后还不能断定程序是否正确，必须通过第 2 批数据的测试。这也就是说如果程序中有多个分支情况，必须对每一分支

进行数据测试，才能确保程序无误。

不带 else 的 if 语句形式如下：

```
if(表达式)
{  子句  }
```

说明：

(1) if 是关键字，if 后面的表达式可以是任意合法表达式。例如，x＞5、x＝＝5、x＝5、x＋5、x、5 等。不管是何种表达式，都要先计算该表达式的值，再根据其结果的非零或零来判断表达式是“真”还是“假”。

(2) x＝＝0 和 x＝0 是两个不同的表达式，当 x 的值为 0 时，前一个表达式的值为 1(表示“真”)，而后一表达式的值为 0(表示“假”)，因此编写程序时一定要分清是用“＝＝”还是用“＝”。

(3) 在 if 语句格式中将 if 子句用大括号括起来了，这是因为 if 子句语法上要求一条语句，而用大括号括起来的多条语句(称为复合语句)，在语法上当做一条语句。当 if 子句只包含一条语句时，一对大括号可以省略。

(4) 这种类型的 if 语句执行过程是：先计算表达式的值，若表达式的计算结果为非零数，则执行 if 子句，否则跳过 if 子句(如图 3.1 所示)。

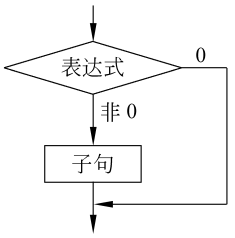

图 3.1　不带 else 的 if 语句执行过程

【例 3.5】 编写输出如下分段函数值的程序，要求整数 x 的值从键盘输入。

$$y=\begin{cases} x+1 & (x\leqslant 0)\\ 1 & (0<x\leqslant 3)\\ x & (x>3)\end{cases}$$

【解】 编程点拨：

本例题需要根据 x 的 3 个不同取值范围找出相应的函数关系式，可以使用 3 个 if 语句。程序如下：

```
#include <stdio.h>
main( )
{    int x=0,y=0;

     printf("Input x:"); scanf("%d",&x);
     if(x<=0)                    //如果 x≤0,执行 y=x+1; 否则跳过此语句
         y=x+1;
     if(x>0 && x<=3)             //如果 0<x≤3,执行 y=1; 否则跳过此语句
         y=1;
     if(x>3)                     //如果 x>3,执行 y=x; 否则跳过此语句
         y=x;
     printf("x=%d,y=%d\n", x,y);
}
```

第 1 次运行结果：

```
Input x:5
x=5,y=5
```

第 2 次运行结果：

```
Input x:-3
x=-3,y=-2
```

第 3 次运行结果：

```
Input x:2
x=2,y=1
```

程序说明：

(1) 本程序通过 3 类数据分别对每一种情况进行了验证。

(2) 上面的程序可改写为如下形式，但此时没把计算所得的函数值存起来：

```
#include <stdio.h>
main( )
{    int x=0;

     printf("Input x:"); scanf("%d",&x);
     if(x<=0)  printf("x=%d,y=%d\n",x,x+1);
     if(x>0 && x<=3) printf("x=%d,y=%d\n",x,1);
     if(x>3)  printf("x=%d,y=%d\n",x,x);
}
```

【例 3.6】 输入 3 个整数，输出其中最大数。

【解】 编程点拨：

先看一个例题。假设有 3 个苹果，为了找出其中最大的，首先用手中拿的第 1 个苹果与第 2 个进行比较。如果第 2 个苹果比手中的大，则放弃手中的苹果，拿起第 2 个苹果，否则不拿第 2 个，这时手中的苹果是前两个苹果中的大者。接着将手中的苹果和第 3 个做比较，如果第 3 个苹果比手中的大，放弃手中的拿起第 3 个，否则不拿，这时手中的苹果已经是 3 个苹果中的最大者。

本题的算法与上面思路类似，其算法用图 3.2 表示，其中 max 相当于拿苹果的手，a、b、c 相当于 3 个苹果，最后 max 中存放 3 个数中的最大值。程序如下：

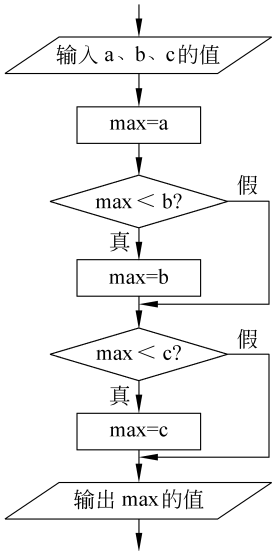

图 3.2　例 3.6 的流程图

```
#include <stdio.h>
main( )
{    int a=0,b=0,c=0,max=0;

     printf("Input a,b,c:"); scanf("%d%d%d",&a,&b,&c);
     max=a;                  //max 内存放 a 的值
     if(max<b) max=b;        //max 内存放 a、b 中较大的值
```

```
    if(max<c) max=c;                    //max 内存放 a、b、c 中最大的值
    printf("a=%d,b=%d,c=%d,max=%d\n",a,b,c,max);
}
```

第 1 次运行结果：

```
Input a,b,c:3 5 7
a=3,b=5,c=7,max=7
```

第 2 次运行结果：

```
Input a,b,c:9 5 7
a=9,b=5,c=7,max=9
```

第 3 次运行结果：

```
Input a,b,c:3 8 7
a=3,b=8,c=7,max=8
```

程序说明：

max 是为存放最大值开辟的变量，变量名可以按照其命名规则随意起名，在本例中为了达到见名知意的效果，选择 max 做其变量名。

【讨论题 3.1】 在 4 个数中找最大数如何解决？在 100 个或更多的数中用同样的方法找最大数方便吗？

下面以例 3.6 为例介绍用单步执行的方法测试、调试程序的方法。

首先进行编译和连接，并修改所发现的错误。当程序中已没有语法错误，但运行结果不正确时，可以用下面介绍的单步执行方法进行调试。

使用 F10 键可以按程序的执行顺序逐行执行（注意，不一定是一条语句），每按一次 F10 键，系统执行一行程序。假设输入的三个数为 3，8，7，则单步执行界面如图 3.3 所示。在执行过程中各变量的变化情况显示在下部窗口中，在其右侧小窗口的“Name”栏中输入某表达式，立即在“Value”栏中显示相应的值。通过运行界面可随时观察运行结果。

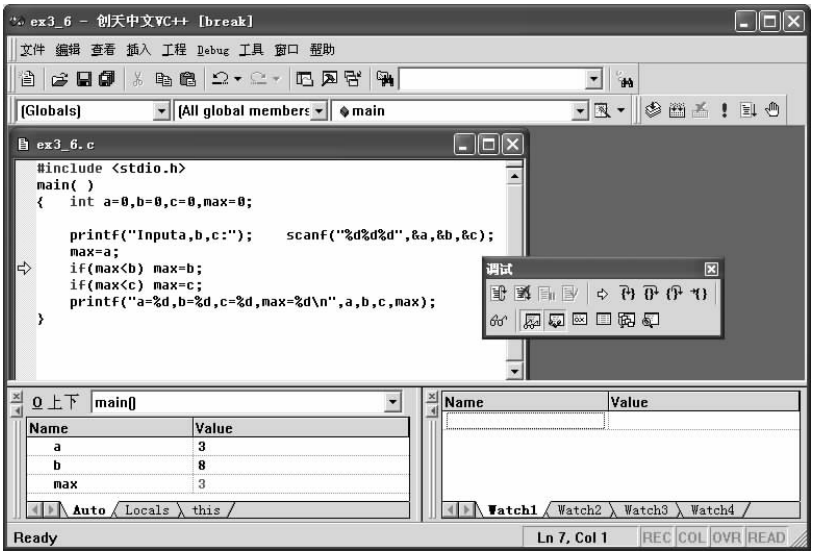

图 3.3 在 VC++ 中单步执行情况

为了有效地观察 if 语句的执行过程，可以将两个 if 语句改写成以下形式：

```
if(max<b)
    max=b;
if(max<c)
    max=c;
```

单步执行的过程中，黄色箭头会在“max＝b;”处停留，说明“max＜b”的判断结果为“真”，max 的值变为 8(即 3 和 8 中较大者)，再按 F10 键时，由于“max＜c”的判断结果为“假”，黄色箭头跳过“max＝c;”，max 的值还是 8(3、8、7 中最大者)。

单步执行也可用 F11 键，但 F11 键和 F10 键的功能有一定的区别(参见第 7 章)。

若要停止调试，执行 Debug|Stop Debugging 菜单项。

【例 3.7】 输入 3 个不同的整数，分别存放在 a、b、c 中，再把这 3 个数按从小到大的顺序重新放入 a、b、c 后输出。

【解】 编程点拨：

3 个数从小到大顺序排列的算法类似于 3 个小孩由矮到高排队。本题可以按以下步骤进行：

(1) 比较前两个数。如果后面的数比前面的小，两个数交换，否则不交换，如图 3.4(a)所示。

(2) 第 2 个数(前两个数中大者)与最后一个比大小。如果最后一个比第 2 个小，这两个数交换，否则不交换，这时最后面的数是 3 个数中最大值(冒第 1 个泡 7)，如图 3.4(b)所示。

(3) 前两数进行比较。如果后面的数比前面的小，这两个数交换，否则不交换。这时中间的数次大(冒第 2 个泡 6)，如图 3.4(c)所示，自然第 1 个数最小。

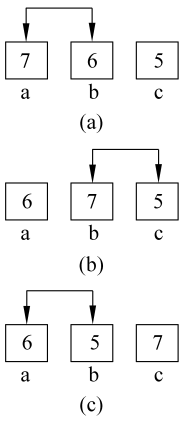

图 3.4　3 个数排序过程

这种算法被称为冒泡法。其程序如下：

```
#include <stdio.h>
main( )
{    int a=0,b=0,c=0,temp=0;

     printf("Input a,b,c:");          scanf("%d%d%d",&a,&b,&c);
     printf("Before: a=%d,b=%d,c=%d\n",a,b,c);
     if(a>b)                          //执行 if 语句后,b 内存放 a 和 b 中较大数
     {  temp=a; a=b; b=temp; }
     if(b>c)                          //执行 if 语句后,c 内存放 3 数中最大数
     {  temp=b; b=c; c=temp; }
     if(a>b)                          //执行 if 语句后,b 内存放 3 数中次大数
     {  temp=a; a=b; b=temp; }
     printf("After: a=%d,b=%d,c=%d\n",a,b,c);
}
```