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；按 F10 键单步执行程序的方法。
- 贯穿实例的部分程序(在主菜单中输入选项，并根据选项显示相应信息)。

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。

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　求 3 个数中最大值；单步执行程序。

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。

贯穿实例——成绩管理程序(2)。

(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，经常根据条件成立与否来确定要执行哪条语句，这些条件一般用关系表达式和逻辑表达式来表示。下面详细介绍构成这两种表达式的运算及相应的表达式。

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	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 的值是否大于或等于 0
＜	小于	6	x＜0	x 的值是否小于 0
＜＝	小于等于	6	x＜＝0	x 的值是否小于或等于 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

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。x＞0 和 x＝＝0 都是关系表达式。关系表达式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。

表 3.2　关系表达式举例

x 的值	举　例	关系表达式的判断结果	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

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)。

(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 3.2 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	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)

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

逻辑表达式由逻辑运算符连接 C 语言表达式而构成的表达式,如 x>=0 && x<=2、x<-5 || x>5 等都是逻辑表达式,逻辑表达式的结果只有“真”“假”两种情况。

【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>
int main(void)
{   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);
    return 0;
}
```

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 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
||	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>
int main(void)
{   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
    return 0;
}
```

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,如果第一个表达式为“假”,系统就不再对第二个表达式做运算,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,如果第一个运算量为“真”,系统也不再对第二个运算量做运算,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

前面两章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执行不同的语句，这时需要设计分支结构程序来实现，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。本节将分别介绍 if 语句和 switch 语句，在 3.6.2 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>
int main(void)
{   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);
    return 0;
}
```

第一次运行结果：

```
Input a,b:2 5
a=2,b=5
```

（a 和 b 的值不变）

第二次运行结果：

```
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，所以在得到第一次运行结果后还不能断定程序是否正确，必须通过第二批数据的测试。也就是说，如果程序中有多个分支的情况，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”)，后一个表达式的值为 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 子句用花括号括起来了，这是因为 if 子句语法上要求一条语句，而用花括号括起来的多条语句(称为复合语句)，在语法上当作一条语句。当 if 子句只包含一条语句时，一对花括号可以省略。

(4) 第一种 if 语句的执行过程是：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)。

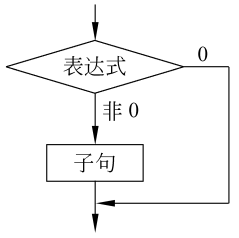


图 3.1　不带 else 的 if 语句执行过程

【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。

$$y=\begin{cases} x-2, & x<0 \\ 1, & x=0 \\ 2x, & 0<x\leqslant 3 \\ \dfrac{x}{3}, & x>3 \end{cases}$$

【解】 编程点拨：

本例题需要根据 x 的 4 个不同取值范围找出相应的 C 语言表达式，然后使用 4 条 if 语句实现相应的功能。程序如下：

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

```
{   int x=0;
    double y=0.0;

    printf("Input x:");
    scanf("%d",&x);
    if(x<0)                  //如果 x<0,执行 y=x-2; 否则跳过此语句
        y=x-2;
    if(x==0)                 //如果 x=0,执行 y=1.0; 否则跳过此语句
        y=1.0;
    if(x>0 && x<=3)          //如果 0<x≤3,执行 y=2.0*x; 否则跳过此语句
        y=2.0*x;
    if(x>3)                  //如果 x>3,执行 y=x/3.0; 否则跳过此语句
        y=x/3.0;
    printf("x=%d,y=%lf\n",x,y);
    return 0;
}
```

第一次运行结果：

```
Input x:-5
x=-5,y=-7.000000
```

第二次运行结果：

```
Input x:0
x=0,y=1.000000
```

第三次运行结果：

```
Input x:2
x=2,y=4.000000
```

第四次运行结果：

```
Input x:8
x=8,y=2.666667
```

程序说明：

(1) 运行程序时，通过输入 4 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>
int main(void)
{   int x=0;

    printf("Input x:");  scanf("%d",&x);
    if(x<0)   printf("x=%d,y=%lf\n",x,x-2);
    if(x==0)   printf("x=%d,y=%lf\n",x,1.0);
    if(x>0 && x<=3)   printf("x=%d,y=%lf\n",x,2.0*x);
    if(x>3)   printf("x=%d,y=%lf\n",x,x/3.0);
```

```
    return 0;
}
```

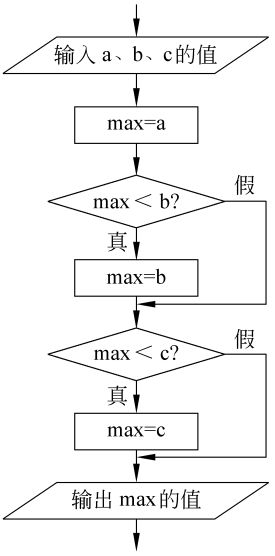


图 3.2　例 3.6 的流程图

【例 3.6】 输入 3 个整数，输出其中最大数。

【解】 编程点拨：

先举一个例题。擂台赛：谁赢谁上奖台；只有 1 个人时，他自然就站在奖台上；来了第二个人，他需要跟奖台上的人进行比赛，谁赢谁上奖台；又来了第三个人，他又需要跟奖台上的人进行比赛，谁赢谁上奖台；以此类推，最终站在奖台上的一定是冠军。

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

```
#include <stdio.h>
int main(void)
{   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);
    return 0;
}
```

第一次运行结果：

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

第二次运行结果：

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

第三次运行结果：

```
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 为其变量名。

【讨论题 3.1】 在 4 个数中如何找最大数？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，并修改语法错误。当程序中没有语法错误，但运行结果仍不正确时，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。单步执行界面如图 3.3 所示（假设输入的 3 个整数为 3、8、7）。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“名称”栏中输入某表达式，立即在“值”栏中显示相应的值。通过运行界面可随时观察运行结果。

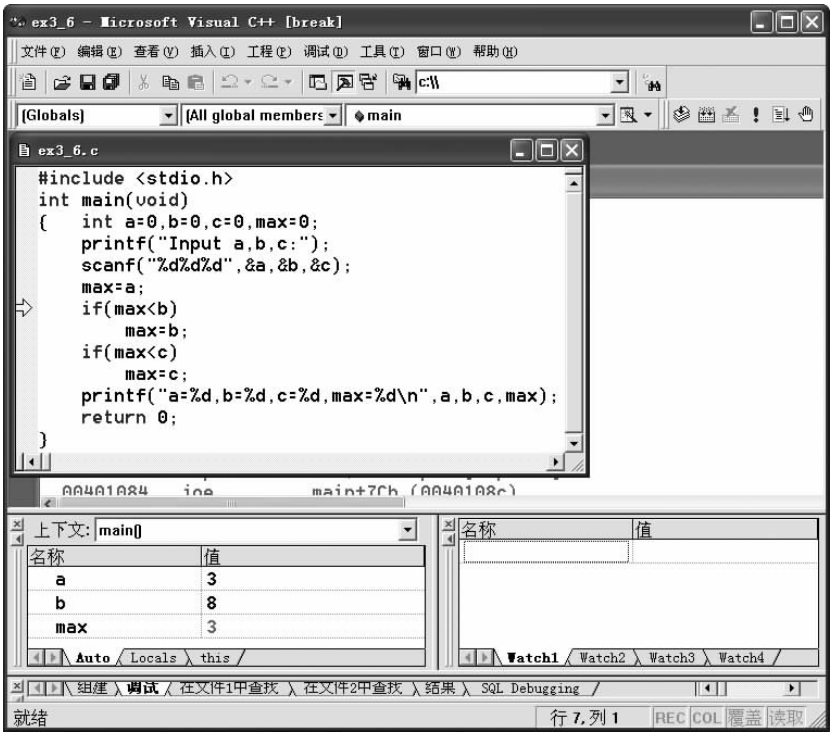


图 3.3　在 Visual C++ 6.0 中单步执行情况

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;
```

在单步执行的过程中，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 中的最大者）。

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

若要停止调式，选择“调试”|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，然后输出。

【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) 第二个数(前两个数中大者)与最后一个比大小。如果最后一个比第二个小，则这两个数交换，否则不交换，这时最后面的数是3个数中最大值(冒第一个泡7，如图3.4(b)所示)。

(3) 前两个数进行比较。如果后面的数比前面的小，则这两个数交换，否则不交换。这时中间的数次大(冒第二个泡6，如图3.4(c)所示)，自然第一个数最小。

这种算法被称为冒泡法。其程序如下：

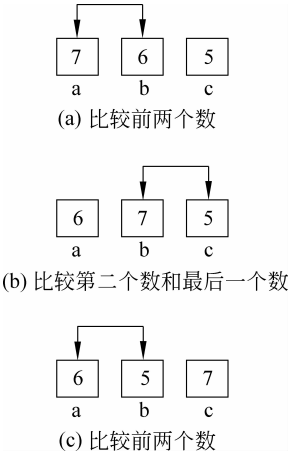


图3.4 3个数排序过程

```
#include <stdio.h>
int main(void)
{   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);
    return 0;
}
```

运行结果：

```
Input a,b,c:7 6 5
Before: a=7,b=6,c=5
After: a=5,b=6,c=7
```

程序说明：

(1) if子句语法上要求一条语句，如果语句多于一条，就用花括号把它们括起来。用花括号括起来的多条语句称为复合语句，复合语句在语法上当作一条语句。

(2) if语句“if(a>b) {temp=a; a=b; b=temp;}”是一条语句。如果将此语句写成“if(a>b) {temp=a; a=b; b=temp;};”即在花括号后面加一个分号，则变成两条语句：一条if语句和一条空语句，仅由分号构成的语句称为空语句。如果将上面if语句写成“if(a>b); {temp=a; a=b; b=temp;}”也变成两条语句：一条if语句(if子句为空语句)和一个复合语句。其作用是：如果a>b，则先执行空语句，再执行复合语句；否则，直接执

行复合语句。由于空语句不产生任何效果，不管 a 的值是否比 b 的值大，都要执行复合语句，两种效果相同。这导致与原 if 语句作用不同。

(3) 本程序只给出一种情况的运行结果，请读者自行测试其他各种情况，并建议用单步执行的方法观察程序中 a、b、c 的变化情况。

【讨论题 3.2】 在本题中将 3 个数改成 4 个数，程序应如何修改？若改成 20(或更多)个数，此方法方便吗？

2. 带 else 的 if 语句

【例 3.8】 输入一个整数，如果是偶数，则输出 Even number；如果是奇数，则输出 Odd number。

【解】 程序如下：

```
#include <stdio.h>
int main(void)
{   int a=0;
    printf("Input a:");    scanf("%d",&a);
    if(a%2==0)                          //如果 a 的值是偶数
        printf("Even number\n");        //输出 Even number
    else                                //否则
        printf("Odd number\n");         //输出 Odd number
    return 0;
}
```

第一次运行结果：

```
Input a:16
Even number
```

第二次运行结果：

```
Input a:5
Odd number
```

程序说明：

如果 a 的值是偶数，则表达式 a%2==0 的值为 1，是非 0 值，因此执行语句“printf("Even number\n");”，如果 a 的值是奇数，则表达式 a%2==0 的值为 0，因此执行语句“printf("Odd number\n");”。

if-else 的语句形式如下：

```
if(表达式)
{   if 子句   }
else
{   else 子句   }
```

说明：

(1) if 和 else 都是关键字，else 必须与 if 配对使用。if 子句和 else 子句都用花括号

括起来当作一条语句，当 if 子句或 else 子句只包含一条语句时，可将对应的花括号省略。

(2) if-else 语句的执行过程是：先计算表达式的值，若表达式的结果为非零值，则执行 if 子句，否则执行 else 子句(如图 3.5 所示)。

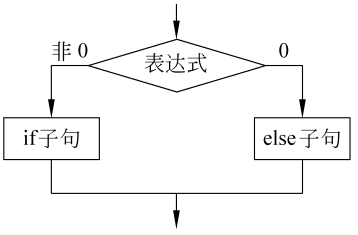


图 3.5　带 else 的 if 语句执行过程

(3) 使用 if 语句时，不要随意加分号，否则会产生语法错误或导致语句作用不同。例如，若将本例中的 if 语句写成

```
if(a%2==0);                          //多加了分号,所以 if 语句到此结束
    printf("Even number\n");         //此语句变成了 if 语句的下一条语句
else                                 //没有配对的 if
    printf("Odd number\n");
```

则导致语法错误；如果改写成

```
if(a%2==0)
    printf("Even number\n");
else;                                //多加了分号,所以 if 语句到此结束
printf("Odd number\n");              //此语句变成了 if 语句的下一条语句
```

则虽然没有语法错误，但语句作用不同。

3.3.2　if 语句的嵌套

在 if 子句或 else 子句中还可以包含另一个 if 语句，这样的 if 语句称为嵌套的 if 语句。

【例 3.9】 编写含有嵌套 if 语句的程序。

【解】 程序如下：

```
#include <stdio.h>
int main(void)
{    int a=3,b=3;

     if(a>5)
     if(a<10) a++;                   内嵌 if 语句    外层 if 语句
     else a--;
     if(b>5)                         内嵌 if 语句    外层 if 语句
     { if(b<10) b++; }
     else b--;
     printf("a=%d,b=%d\n",a,b);
     return 0;
}
```

运行结果：

```
a=3,b=2
```

程序说明：

(1) 本程序由于书写不规范，给阅读和分析程序带来了困难。C 语法规定，在分支结构中，else 总是与前面最近的不带 else 的 if 相结合。根据这个原则，程序中的第一个 else 应该与 if(a<10)配对，作为一个完整的 if-else 语句内嵌在外层 if(a>5)中。第二个 else 应该与 if(b>5) 配对，原因是语句“if(b<10) b++;”在用一对花括号括起来之后，已作为 if(b>5)的子句内嵌在其中。

(2) 为了增强程序的可读性，应将本程序中内嵌 if 语句往右缩进几个格。

上面的例题都是在 if 子句中内嵌 if 语句，当内嵌层数较多时，else 与 if 的配对关系往往不好确定，如果在 else 子句中内嵌 if 语句，配对关系比较明确。下面举例说明，请读者比较。

【例 3.10】 编写求下面分段函数值的程序，其中 x 的值从键盘输入。

$$y=\begin{cases}0, & x<0\\ x^3+5, & 0\leqslant x<10\\ 2x^2-x-6, & 10\leqslant x<20\\ x^2+1, & 20\leqslant x<30\\ x+3 & x\geqslant 30\end{cases}$$

【解】 当 x 小于 0 时，通过关系式“y=0”计算函数值，否则(即 x≥0)，还要判断 x 是否小于 10，若满足，说明 x 在[0,10)之内，因此用关系式 y= x^3+5 计算函数值，若不满足(即 x≥10)，则再判断 x 是否小于 20，以此类推，在每次的判断中只要满足条件，就选对应的关系式计算，不满足时继续判断。所以本程序属于 else 子句包含另一个 if 语句的情况，应采用嵌套的 if 语句，其程序如下：

```
#include <stdio.h>
int main(void)
{   double x=0,y=0;
    printf("Input data:");     scanf("%lf",&x);
    if(x<0)  y=0;
    else
        if(x<10) y=x*x*x+5;
        else
            if(x<20) y=2*x*x-x-6;
            else
                if(x<30) y=x*x+1;
                else y=x+3;
    printf("x=%lf,y=%lf\n",x,y);
    return 0;
}
```

调试本程序时，应至少验证具有代表性的 5 个数据，例如，当输入 0、5、15、25、35 时，相应的函数值应分别为 5.000000、130.000000、429.000000、626.000000、38.000000。

程序说明：

(1) 程序中 if 语句只在 else 子句中不断包含另外 if 语句，其好处是 if 与 else 的配对关系一目了然，不容易出错。因此，建议读者尽量使用 else 子句中包含 if 语句的形式。本例中的 if 语句常用下面的简单形式表示：

```
if(x<0)  y=0;
else if(x<10) y=x*x*x+5;
else if(x<20) y=2*x*x-x-6;
else if(x<30) y=x*x+1;
else y=x+3;
```

(2) 程序中第一个 else 隐含了 x≥0，所以在其后 if 的判断表达式中不必写成“x>=0 && x<10”，后面 if 的情况也相同。

3.6.1 节中将更详细地介绍嵌套 if 语句。

3.4 switch 语句

前面介绍的 if 语句只有两个分支，如果要用 if 语句解决多分支问题，只能使用嵌套 if 语句，如果分支很多，则嵌套层次多，编程时容易出错，阅读程序也会混淆。用 switch 语句可以方便地解决多分支问题。

【例 3.11】 switch 语句的示例。从键盘输入一个整数放在 a 中，当输入的值为 1 时屏幕上显示 A，输入 2 时显示 B，输入 3 时显示 C，当输入其他整数时显示 D。

【解】 程序如下：

```
#include <stdio.h>
int main(void)
{   int a=0;
    scanf("%d",&a);
    switch(a)                                          //根据 a 的值选择分支
    {   case  1:  printf("A\n");  break;               //a 的值为 1 时
        case  2:  printf("B\n");  break;               //a 的值为 2 时
        case  3:  printf("C\n");  break;               //a 的值为 3 时
        default:  printf("D\n");  break;               //a 的值为 1、2、3 以外的数时
    }
    return 0;
}
```

分别输入 1、2、3、4 时，程序运行结果分别是：

程序说明：

(1) switch、case 和 default 都是关键字，不能作为标识符。

(2) 当 a 的值为 1 或 2 或 3 时，执行相应 case 后面的语句，遇到 break 语句，立即结束 switch 语句的执行。如果 a 的值为 1、2、3 以外的整数，则执行 default 后面的语句。

switch 语句的一般形式为：

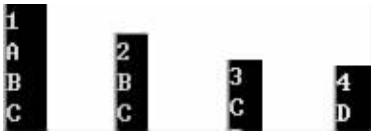
```
switch (表达式)
{   case  常量表达式 1:  语句组 1   break;
    case  常量表达式 2:  语句组 2   break;
          ⋮
    case  常量表达式 n:  语句组 n   break;
    default:    语句组 n+1  break;
}
```

说明：

(1) switch 后面的表达式和常量表达式 k(=1,2,…,n)可以是整型或字符型。

(2) case 后面不能出现变量或含变量的表达式，只能出现常量表达式，而且每个常量表达式的值不能相等。

(3) switch 语句的执行过程是先用表达式的值逐个与常量表达式的值比较，在找到值相等的常量表达式时，执行其后的语句组，否则执行 default 后面的语句组。在执行 case 或 default 后面的语句组时，遇到 break 语句，则退出 switch 语句体，否则继续执行其下面的语句。例如，执行下面程序段时分别输入 1、2、3、4，则程序运行结果分别是：




```
scanf("%d",&a);
switch(a)
{   case  1:  printf("A\n");
    case  2:  printf("B\n");
    case  3:  printf("C\n");  break;
    default:  printf("D\n");
}
```

(4) 在 switch 语句体中可以没有 default 分支，这时如果找不到对应的 case 分支，那么流程将不进入 switch 语句。例 3.11 的流程图参见图 3.6。

switch 语句常用于处理键盘命令，例如，3.5 节中贯穿实例的菜单选择程序。

在解决实际问题时，经常希望将某一限定范围作为 case 后的常量表达式，这时应先将该范围转换成整数或字符，然后再通过 switch 语句进行处理，请看例 3.12。

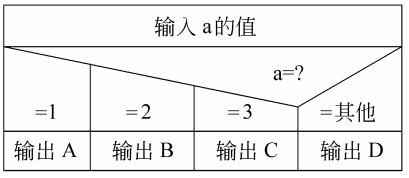


图 3.6　switch 语句的执行过程

【例 3.12】 输入一个百分制成绩，输出成绩等级 A、B、C、D、E。输入的数据在 90～100 分为 A，80～89 分为 B，70～79 分为 C，60～69 分为 D，0～59 分为 E，否则显示出错信息。

【解】 编程点拨：

本程序属于多分支问题，因此用 switch 语句解决，但因为要将每个分数段作为一个分支处理，所以必须把分数段转换成某一分支。可以用表达式“成绩/10”进行转换，例如，对于 80～89 之间的成绩，通过“成绩/10”的计算值都是 8，即将一个分数段转化为一个整数。用 10 除是因为分数段的间隔是 10。程序如下：

```
#include <stdio.h>
int main(void)
{   int score=0,temp=0;
    printf("Input score:");
    scanf("%d",&score);
    if(score<0 || score>100)
        printf("Error! \n");
    else
    {   temp=score/10;                  //将范围转换成整数
        switch(temp)
        {   case  10 :                  //可以没有语句组
            case   9 :  printf("A\n");  break;
            case   8 :  printf("B\n");  break;
            case   7 :  printf("C\n");  break;
            case   6 :  printf("D\n");  break;
            default :  printf("E\n");  break;
        }
    }
    return 0;
}
```

第一次运行结果：

```
Input score:89
B
```

第二次运行结果：

```
Input score:105
Error!
```

其他情况请读者自行尝试。

程序说明：

(1) 本程序在 else 子句中包含了 switch 语句，这在 C 语言中是允许的。

(2) case 后面可以没有语句组。如在本程序中，case 10 和 case 9 的情况，都需要执行相同的语句组，这时 case 10 后面的语句组可以省略。

(3) 希望读者利用本例题中的程序,体会程序的书写格式。

switch 语句的进一步讨论参见 3.6.1 节。

【讨论题 3.3】 如果将例 3.12 的功能改为:输入的数据在 85～100 分为 A,70～84 分为 B,55～69 分为 C,40～54 分为 D,0～39 分为 E,否则显示出错信息,那么应怎样改写程序?

3.5 贯穿实例——成绩管理程序(2)

成绩管理程序之二:完善 2.4 节的贯穿实例,即编写程序,在主菜单中输入选项,并根据输入的选项显示信息,若输入 1,则显示“选择了 1”;若输入 0～7 之外的选项,则显示“非法选项”。

【解】 编程点拨:

本程序中要求根据不同的输入选项显示不同的信息,这是一个多分支结构的问题,可以用嵌套的 if-else 语句实现,也可以用 switch 语句实现。本程序使用 switch 语句层次更清晰、更容易阅读,因此本程序用 switch 语句实现。程序流程图如图 3.7 所示。

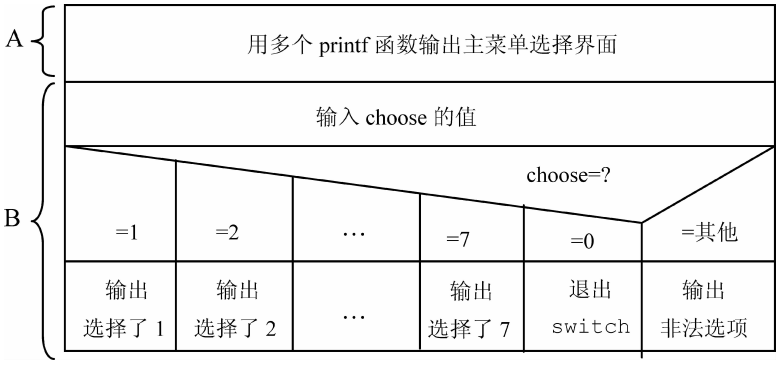


图 3.7 选择主菜单选项的 N-S 图

程序如下:

```
#include <stdio.h>
int main(void)
{   char choose='\0';
    printf("      |~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\n");
    printf("      |     请输入选项编号(0-7)      |\n");
    printf("      |~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|\n");
    printf("      |          1:输入              |\n");
    printf("      |          2:显示              |\n");
    printf("      |          3:查找              |\n");
    printf("      |          4:最值              |\n");
    printf("      |          5:插入              |\n");
```

```
    printf("      |          6: 删除             |\n");
    printf("      |          7: 排序             |\n");
    printf("      |          0: 退出             |\n");
    printf("      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");
    printf("              ");
    choose=getchar();
    switch(choose)
    {   case '1': printf("          选择了 1\n");  break;
        case '2': printf("          选择了 2\n");  break;
        case '3': printf("          选择了 3\n");  break;
        case '4': printf("          选择了 4\n");  break;
        case '5': printf("          选择了 5\n");  break;
        case '6': printf("          选择了 6\n");  break;
        case '7': printf("          选择了 7\n");  break;
        case '0': break;
        default :printf("            %c 为非法选项! \n",choose);
    }
    return 0;
}
```

运行结果：

3.6 提 高 部 分

3.6.1 if 语句和 switch 语句的进一步讨论

在 3.3 节和 3.4 节中，已经介绍了 if、switch 语句的一般形式及使用方法，由于这两个语句的形式灵活，在使用时容易误操作，下面再举一些例题。

【例 3.13】 分析下面程序的执行过程。

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

```
    scanf("%d%d",&a,&b);
    if(a>b)                 //外层 if 子句是另一个 if 语句
        if(b>0)                                          ┐
            x=a+1;          ——内嵌 if 子句               │ 外层 if 子句
        else                                             │
            x=a-1;          ——内嵌 else 子句             ┘
    else                    //外层 else 子句也是另一个 if 语句
        if(a>0)                                          ┐
            x=b+1;          ——内嵌 if 子句               │ 外层 else 子句
        else                                             │
            x=b-1;          ——内嵌 else 子句             ┘
    printf("a=%d,b=%d,x=%d\n",a,b,x);
    return 0;
}
```

【解】 第一次运行结果：

```
7 5
a=7,b=5,x=8
```

第二次运行结果：

```
7 -5
a=7,b=-5,x=6
```

第三次运行结果：

```
1 8
a=1,b=8,x=9
```

第四次运行结果：

```
-1 8
a=-1,b=8,x=7
```

本程序是属于 if 语句的嵌套，外层 if 语句中的 if 子句和 else 子句都是另一个 if 语句。嵌套 if 语句的一般形式如下：

```
if(表达式 1)
{   …                                          ┐
    if(表达式 2) { 内嵌 if 子句 }               │
    else { 内嵌 else 子句 }                     │ 外层 if 子句
    …                                          │
}                                              ┘
else
{   …                                          ┐
    if(表达式 3) { 内嵌 if 子句 }               │
    else { 内嵌 else 子句 }                     │ 外层 else 子句
    …                                          │
}                                              ┘
```