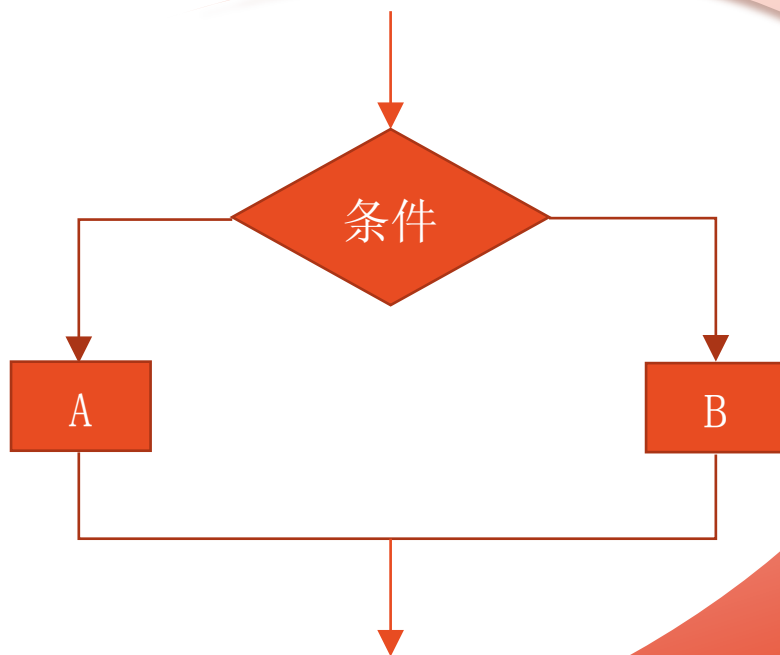


# 第 4 章

## 选择结构程序设计

# 选择结构和条件判断



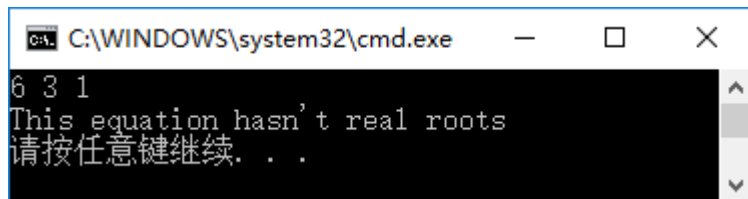
C语言有两种选择语句

- **if**语句，用来实现两个分支的选择结构
- **switch**语句，用来实现多分支的选择结构

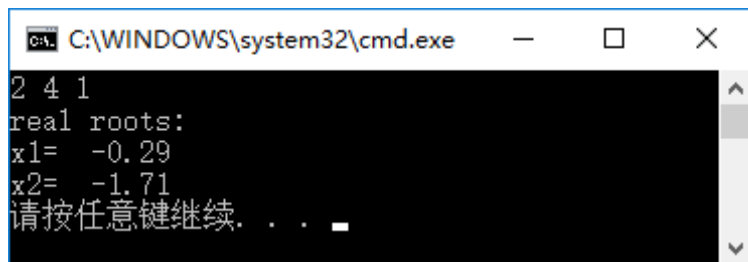
# if语句例题

【例4.1】在例3.5的基础上对程序进行改进。题目要求解得 $ax^2+bx+c=0$ 方程的根。由键盘输入a,b,c。假设a,b,c的值任意，并不保证 $b^2-4ac \geq 0$ 。需要在程序中进行判别，如果 $b^2-4ac \geq 0$ ，就计算并输出方程的两个实根，如果 $b^2-4ac < 0$ ，就输出“此方程无实根”的信息。

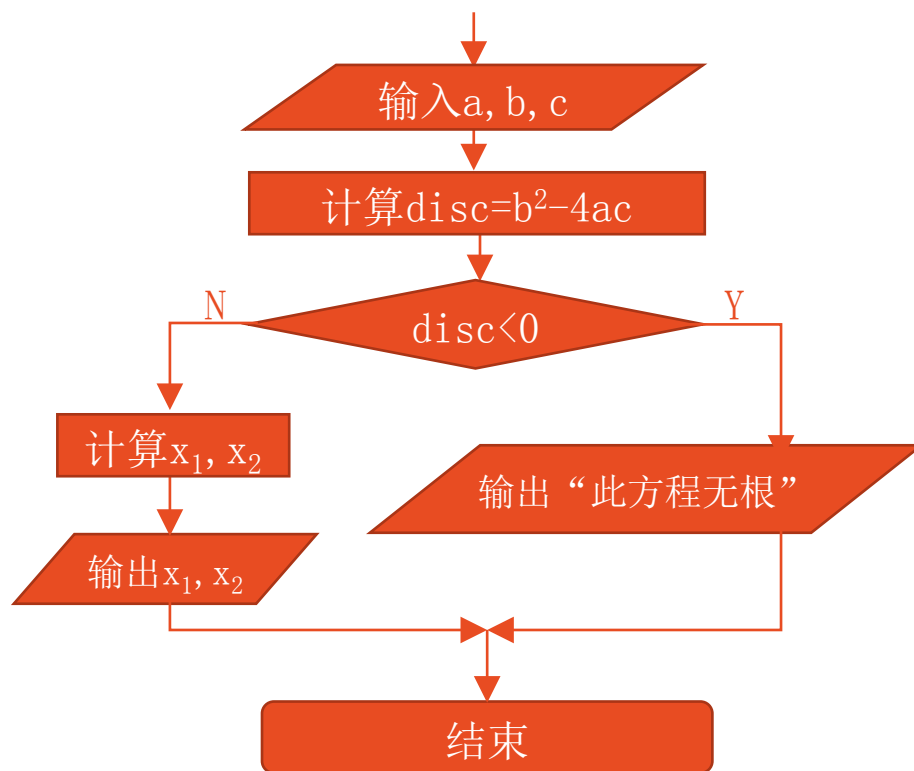
```
#include<stdio.h>
#include<math.h>    //程序中要调用求平方根函数sqrt
int main()
{
    double a, b, c, disc, x1, x2, p, q; //disc是判别式sqrt(b*b-4ac)
    scanf("%lf%lf%lf", &a, &b, &c); //输入双精度浮点型变量的值要用格式声明"%lf"
    disc=b*b-4*a*c;
    if(disc<0)    //若b*b-4ac<0
        printf("This equation hasn't real roots\n"); //输出“此方程无实根”
    else    //b*b-4ac≥0
    {
        p=-b/(2.0*a);
        q=sqrt(disc)/(2.0*a);
        x1=p+q; x2=p-q;    //求出方程的两个根
        printf("real roots:\nx1=%7.2f\nx2=%7.2f\n", x1, x2);    //输出方程的两个根
    }
    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
6 3 1
This equation hasn't real roots
请按任意键继续...
```



```
C:\WINDOWS\system32\cmd.exe
2 4 1
real roots:
x1=  -0.29
x2=  -1.71
请按任意键继续...
```

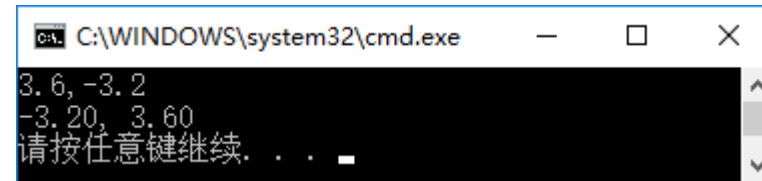


# 用if语句实现选择结构

【例4.2】输入两个实数，按由小到大的顺序输出这两个数。

解题思路：只要做一次比较，然后进行一次交换即可。用if语句实现条件判断。

```
#include <stdio.h>
int main()
{
    float a,b,t;
    scanf("%f,%f",&a,&b);
    if(a>b)
    {
        //将a和b的值互换
        t=a;
        a=b;
        b=t;
    }
    printf("%5.2f,%5.2f\n",a,b);
    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
3.6, -3.2
-3.20, 3.60
请按任意键继续. . . .
```



## 两个变量值的互换

```
a=b; //把变量b的值赋给变量a，a的值等于b的值
b=a; //再把变量a的值赋给变量b，变量b值没有改变
```

因此，为了实现互换，必须借助于第三个变量

# 用if语句实现选择结构

【例4.3】输入3个数 $a$ ， $b$ ， $c$ ，要求按由小到大的顺序输出。

```
#include <stdio.h>
int main()
{
    float a, b, c, t;
    scanf("%f, %f, %f", &a, &b, &c);
    if(a > b)
    {
        t = a;      //借助变量t，实现变量a和变量b互换值
        a = b;
        b = t;
    }
    //互换后，a小于或等于b
    if(a > c)
    {
        t = a;      //借助变量t，实现变量a和变量c互换值
        a = c;
        c = t;
    }
    //互换后，a小于或等于c
    if(b > c)
    {
        t = b;      //借助变量t，实现变量b和变量c互换值
        b = c;
        c = t;
    }
    //互换后，b小于或等于c
    printf("%.2f, %.2f, %.2f\n", a, b, c); //顺序输出a, b, c的值
    return 0;
}
```

## 算法步骤

S1: if  $a > b$ ，将 $a$ 和 $b$ 对换  
(交换后， $a$ 是 $a$ 、 $b$ 中的小者)

S2: if  $a > c$ ，将 $a$ 和 $c$ 对换  
(交换后， $a$ 是 $a$ 、 $c$ 中的小者，  
因此 $a$ 是三者中最小者)

S3: if  $b > c$ ，将 $b$ 和 $c$ 对换  
(交换后， $b$ 是 $b$ 、 $c$ 中的小者，  
也是三者中次小者)

S4: 顺序输出 $a$ ， $b$ ， $c$



在经过第1次互换值后， $a \leq b$ ，经过第2次互换值后 $a \leq c$ ，这样 $a$ 已是三者中最小的(或最小者之一)，但是 $b$ 和 $c$ 谁大还未解决，还需要进行比较和互换。经过第3次互换值后， $a \leq b \leq c$ 。

```
C:\WINDOWS\system32\cmd.exe
3, 7, 1
1.00, 3.00, 7.00
请按任意键继续. . .
```

# if语句的一般形式

```
if (表达式) 语句1  
[ else 语句2 ]
```

“**表达式**”可以是关系表达式、逻辑表达式，甚至是数值表达式

方括号内的部分(即else子句)为可选的，既可以有，也可以没有

**语句1和语句2:**

一个简单的语句，以分号;结束的语句

也可以是一个复合语句{ }

还可以是另一个if语句

## 形式1 没有else子句部分

```
if(表达式) 语句1
```

## 形式2 有else子句部分

```
if (表达式)  
    语句1  
else  
    语句2
```

## 形式3 在else部分又嵌套了多层的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3)      语句3  
  ⋮ ⋮ ⋮              ⋮  
else if(表达式m)      语句m  
else                  语句m+1
```




## 关系运算符和关系表达式

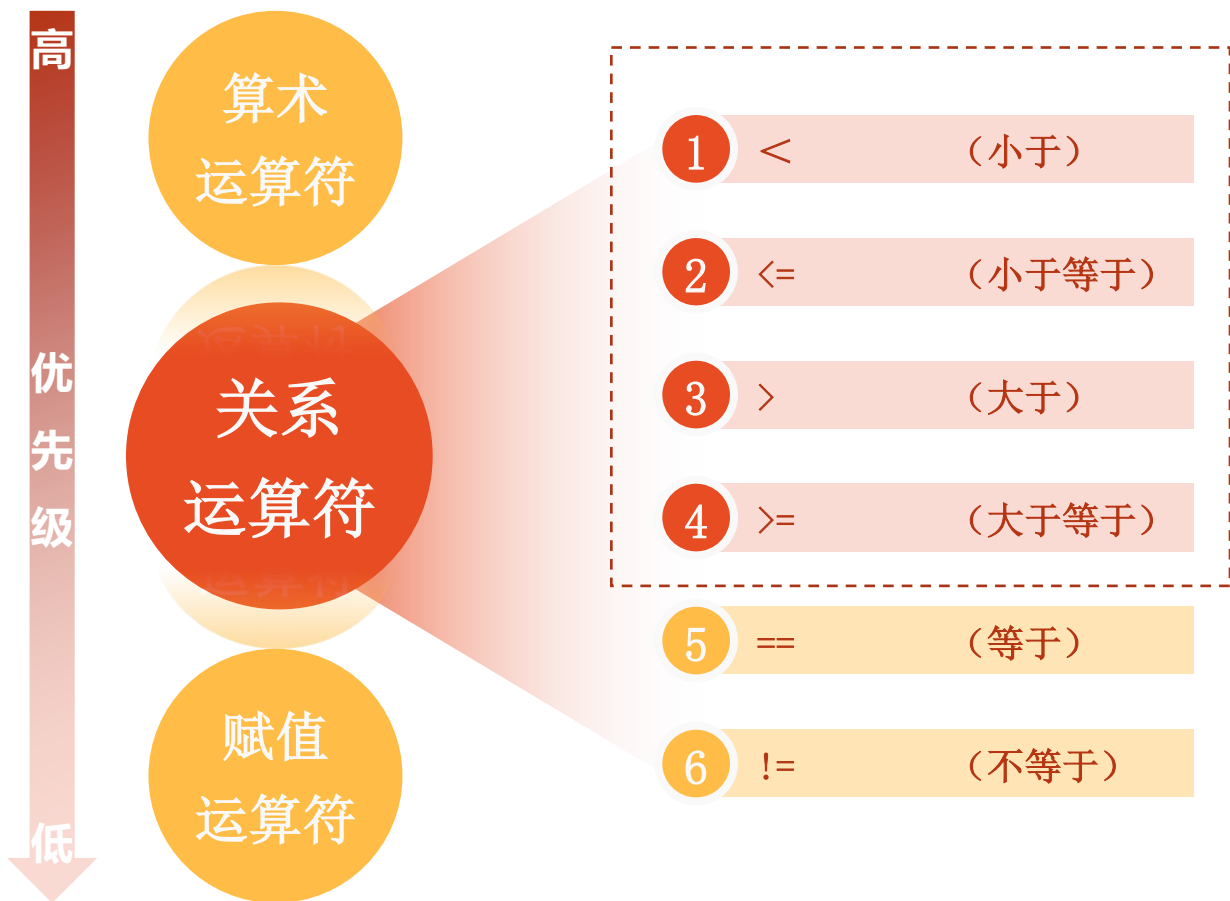
---

在C语言中，比较符(或称比较运算符)称为关系运算符。所谓“关系运算”就是“比较运算”，将两个数值进行比较，判断其比较的结果是否符合给定的条件。

---



# 关系运算符及其优先次序



- 前 4 种关系运算符的优先级别相同，后 2 种也相同。前 4 种高于后 2 种。
- 关系运算符的优先级低于算术运算符。
- 关系运算符的优先级高于赋值运算符。

$c > a + b$  等效于  $c > (a + b)$  (关系运算符的优先级低于算术运算符)

$a > b == c$  等效于  $(a > b) == c$  (大于运算符  $>$  的优先级高于相等运算符  $==$ )

$a == b < c$  等效于  $a == (b < c)$  (小于运算符  $<$  的优先级高于相等运算符  $==$ )

$a = b > c$  等效于  $a = (b > c)$  (关系运算符的优先级高于赋值运算符)

注意：  
= 与 == 不同



# 关系表达式

- 用关系运算符将两个数值或数值表达式连接起来的式子，称为关系表达式。
- 关系表达式的值是一个逻辑值，即“真”或“假”。
- 在C的逻辑运算中，以“1”代表“真”，以“0”代表“假”。

`int a=3, b=2, c=1, d;` 则：

`d = a > b;` 由于`a>b`为真，因此关系表达式`a>b`的值为1，所以赋值后`d`的值为1。

`f = a > b > c;` 则`f`的值为0。因为“`>`”运算符是自左至右的结合方向，先执行“`a>b`”得值为1，再执行关系运算“`1>c`”，得值0，赋给`f`，所以`f`的值为0



# 逻辑运算符和逻辑表达式

---

用逻辑运算符将关系表达式或其他逻辑量连接起来的式子就是逻辑表达式。



# 逻辑运算符及其优先次序

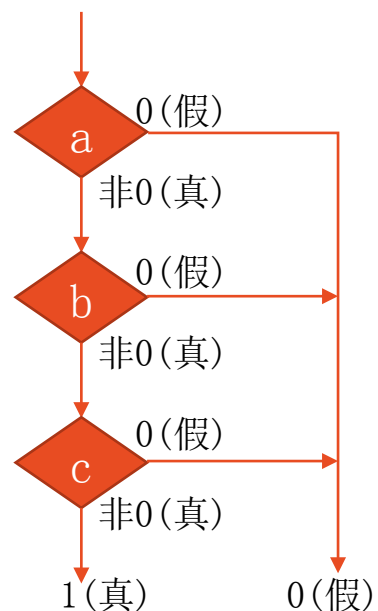
运算符	含义	举例	说明
!	逻辑非 (NOT)	!a	如果a为假, 则!a为真;如果a为真, 则!a为假
&&	逻辑与 (AND)	a && b	如果a和b都为真, 则结果为真, 否则为假
	逻辑或 (OR)	a    b	如果a和b有一个以上为真, 则结果为真, 二者都为假时, 结果为假

- “&&”和“||”是双目运算符, 要求有两个运算对象(操作数); “!”是单目运算符, 只要有一个运算对象
- 优先次序: !(非)→&&(与)→|| (或), 即“!”为三者中最高的; 逻辑运算符中的“&&”和“||”低于关系运算符, “!”高于算术运算符
- 逻辑运算结果不是0就是1, 不可能是其他数值。而在逻辑表达式中作为参加逻辑运算的运算对象可以是0(“假”)或任何非0的数值(按“真”对待)

a	b	!a	!b	a && b	a    b
真 (非0)	真 (非0)	假 (0)	假 (0)	真 (1)	真 (1)
真 (非0)	假 (0)	假 (0)	真 (1)	假 (0)	真 (1)
假 (0)	真 (非0)	真 (1)	假 (0)	假 (0)	真 (1)
假 (0)	假 (0)	真 (1)	真 (1)	假 (0)	假 (0)

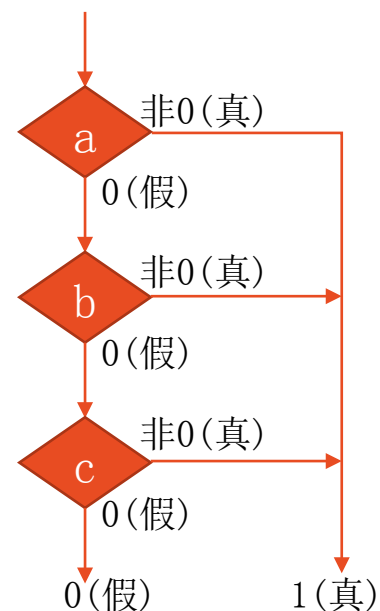
在逻辑表达式的求解中，并不是所有的逻辑运算符都被执行，只是在必须执行下一个逻辑运算符才能求出表达式的解时，才执行该运算符。

- $a \ \&\& \ b \ \&\& \ c$ 。只有a为真(非0)时，才需要判别b的值。只有当a和b都为真时才需要判别c的值。



逻辑  
表达式

- $a \ || \ b \ || \ c$ 。只要a为真(非0)，就不必判断b和c。只有a为假，才判别b。a和b都为假才判别c。



既然关系表达式和逻辑表达式的值是0和1，而且在判断一个量是否为“真”时，以0代表“假”，以非0代表“真”。那么就可以理解为什么在if语句中表达式可以是任何数值表达式。

if (x!=0) 语句1	//括号内的表达式是关系表达式，如果x不等于0，执行语句1
if (x>0 && y>0) 语句2	//表达式是逻辑表达式，如果x和y都大于0，执行语句2
if (x) 语句3	//表达式是变量，如果x不等于0，则条件判断结果为真，执行语句3
if (1) 语句4	//表达式是非0整数，条件判断结果为真，执行语句4
if (0) 语句5	//表达式是整数0，条件判断结果为假，不执行语句5，接着执行下一语句
if(x+3.5) 语句6	//表达式是实数表达式，若x+3.5不等于0，则条件判断结果为真，执行

语句6

## 小例子

判别用year表示的某一年是否闰年，可以用一个逻辑表达式来表示。闰年的条件是符合下面二者之一：①能被4整除，但不能被100整除，如2008。  
②能被400整除，如2000。

```
(year % 4 == 0 && year % 100 != 0) || year % 400 == 0
```

# 条件运算符和条件表达式

```
if (a>b)
    max=a;
else
    max=b;
```



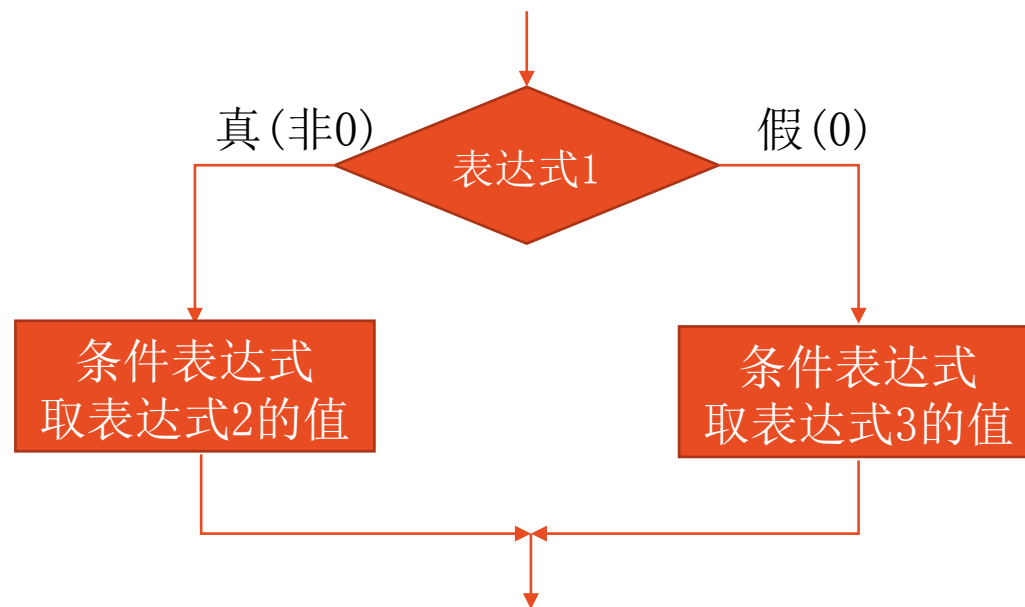
```
max=(a>b) ? a : b; 或
```

```
a>b ? (max=a) : (max=b); //表达式2和表达式3是赋值表达式
```

表达式1 ? 表达式2 : 表达式3

条件运算符由两个符号(?)和(:)组成，必须一起使用。要求有3个操作对象，称为三目(元)运算符，它是C语言中唯一的一个三目运算符。

条件运算符的执行顺序：先求解表达式1，若为非0(真)则求解表达式2，此时表达式2的值就作为整个条件表达式的值。若表达式1的值为0(假)，则求解表达式3，表达式3的值就是整个条件表达式的值。

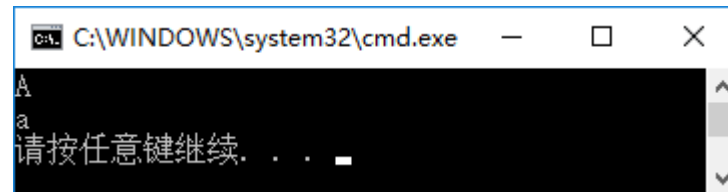


# 条件运算符和条件表达式

【例4.4】输入一个字符，判别它是否为大写字母，如果是，将它转换成小写字母；如果不是，不转换。然后输出最后得到的字符。

解题思路： 用条件表达式来处理，当字母是大写时，转换成小写字母，否则不转换。

```
#include <stdio.h>
int main()
{
    char ch;
    scanf("%c",&ch);
    ch = (ch>='A' &&ch<='Z') ? (ch+32) : ch;
    printf("%c\n",ch);
    return 0;
}
```



条件表达式 “(ch>='A' &&ch<='Z')?(ch+32):ch” 的作用是：如果字符变量ch的值为大写字母，则条件表达式的值为(ch+32)，即相应的小写字母，32是小写字母和大写字母ASCII的差值。如果ch的值不是大写字母，则条件表达式的值为ch，即不进行转换。

注意：C语言表达式值的含义与常规的数学描述的不同。

‘A’ 《 ch 《 ‘Z’  
a = b 与 a == b

# 选择结构的嵌套

```
if()
```

```
    if() 语句1  
    else 语句2
```

} 内嵌if

```
else
```

```
    if() 语句3  
    else 语句4
```

} 内嵌if

**注意**

if与else的配对关系。

else总是与它上面的最近的未配对的if配对。

```
if()  
    if() 语句1  
else  
    if() 语句2  
else 语句3
```

编程者把else写在与第1个if(外层if)同一列上，意图是使else与第1个if对应，但实际上else是与第2个if配对，因为它们相距最近。

如果if与else的数目不一样，为实现程序设计者的思想，可以加花括号来确定配对关系。

```
if()  
{  
    if() 语句1  
}  
else 语句2
```

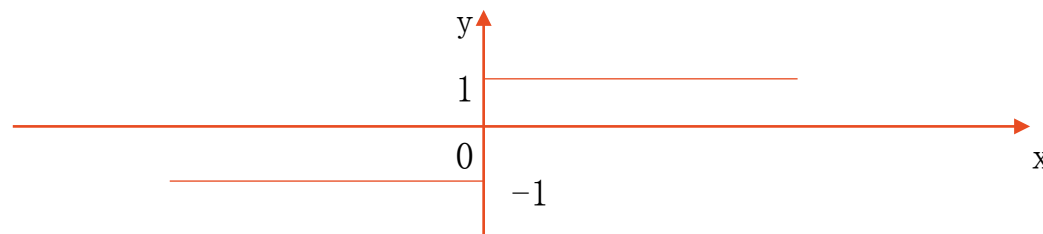
} 内嵌if



# 条件运算符和条件表达式

【例4.5】有一阶跃函数：
$$y = \begin{cases} -1 & (x < 0) \\ 0 & (x = 0) \\ 1 & (x > 0) \end{cases}$$

编一程序,输入一个x值,要求输出相应的y值。



## 算法步骤

先后用3个独立的if语句处理

- S1: 输入x
- S2: 若x<0, 则y = -1
- S3: 若x=0, 则y=0
- S4: 若x>0, 则y=1
- S5: 输出y

```
#include <stdio.h>
int main()
{
    int x, y;
    scanf("%d", &x);
    if(x < 0)
        y = -1;
    else
        if (x == 0) y = 0;
        else y = 1;

    printf("x=%d, y=%d\n", x, y);
    return 0;
}
```

## 算法步骤

用一个嵌套的if语句处理

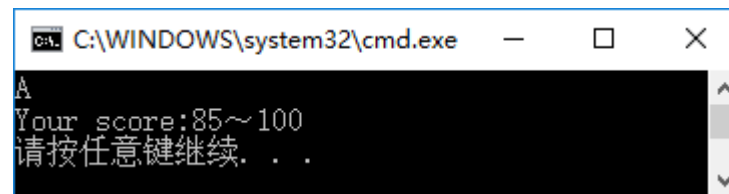
- S1: 输入x
- S2: 若x<0, 则y=-1
- S3: 否则
- S4: 若x=0, 则y=0
- S5: 否则(即x>0), 则y=1
- S6: 输出y

```
#include <stdio.h>
int main()
{
    int x, y;
    scanf("%d", &x);
    if(x>=0) //
        if(x>0) y=1;
        else y=0;
    else y=-1;
    printf("x=%d, y=%d\n", x, y);
    return 0;
}
```

# 用switch语句实现多分支选择结构

【例4.6】要求按照考试成绩的等级输出百分制分数段，A等为85分以上，B等为70～84分，C等为60～69分，D等为60分以下。成绩的等级由键盘输入。

```
#include <stdio.h>
int main()
{
    char grade;
    scanf("%c", &grade);
    printf("Your score:");
    switch(grade)
    {
        case 'A': printf("85~100\n"); break;
        case 'B': printf("70~84\n"); break;
        case 'C': printf("60~69\n"); break;
        case 'D': printf("<60\n"); break;
        default: printf("enter data
error!\n");
    }
    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
A
Your score:85~100
请按任意键继续...
```



等级grade定义为字符变量，从键盘输入一个大写字母，赋给变量grade，switch得到grade的值并把它和各case中给定的值('A'，'B'，'C'，'D'之一)相比较，如果和其中之一相同(称为匹配)，则执行该case后面的语句(即printf语句)。

如果输入的字符与'A'，'B'，'C'，'D'都不相同，就执行default后面的语句，

**注意在每个case后面后的语句中，最后都有一个break语句，它的作用是使流程转到switch语句的末尾(即右花括号处)。**

# 用switch语句实现多分支选择结构

switch(表达式)

{

case 常量1 : 语句1

case 常量2 : 语句2

⋮ ⋮ ⋮

case 常量n : 语句n

default : 语句n+1

}

(1) 括号内的“表达式”，其值的类型应为**整数类型(包括字符型)**。

(2) 花括号内是一个**复合语句**，内包含多个以关键字case开头的语句行和**最多一个以default**开头的行。case后面跟一个**常量(或常量表达式)**，它们和default都是起标号作用，用来标志一个位置。执行switch语句时，先计算switch后面的“表达式”的值，然后将它与各case标号比较，如果与某一个case标号中的常量相同，流程就转到此case标号后面的语句。如果没有与switch表达式相匹配的case常量，流程转去执行default标号后面的语句。

(3) **可以没有default**标号，此时如果没有与switch表达式相匹配的case常量，则不执行任何语句。

(4) 各个case标号出现**次序**不影响执行结果。

(5) 每一个case常量必须互不相同；否则就会出现互相矛盾的现象。

(6) case标号只起标记的作用。在执行switch语句时，根据switch表达式的值找到匹配的入口标号，在执行完一个case标号后面的语句后，就从此标号开始执行下去，不再进行判断。因此，一般情况下，在执行一个case子句后，应当用**break**语句使流程跳出switch结构。最后一个case子句(今为default子句)中可不加break语句。

(7) 在case子句中虽然包含了一个以上执行语句，但可以不必用花括号括起来，会自动顺序执行本case标号后面所有的语句。当然加上花括号也可以。

(8) 多个case标号可以共用一组执行语句。

# 用switch语句实现多分支选择结构

【例4.7】用switch语句处理菜单命令。在许多应用程序中，用菜单对流程进行控制，例如从键盘输入一个'A'或'a'字符，就会执行A操作，输入一个'B'或'b'字符，就会执行B操作。

```
#include <stdio.h>
int main()
{
    void action1(int, int), action2(int, int); //函数声明
    char ch;
    int a = 15, b = 23;
    ch = getchar();
    switch(ch)
    {
        case 'a':
        case 'A': action1(a, b); break; //调用action1函数，执行A操作
        case 'b':
        case 'B': action2(a, b); break; //调用action2函数，执行B操作
        :
        default: putchar(' \a'); //如果输入其他字符，发出警告
    }
    return 0;
}
```

//执行加法的函数

```
void action1(int x, int y)
{
    printf("x+y=%d\n", x+y);
}
```

//执行乘法的函数

```
void action2(int x, int y)
{
    printf("x*y=%d\n", x*y);
}
```

# 选择结构程序综合举例

【例4.8】写一程序，判断某一年是否为闰年。

year被4整除			
真	假		
year被100整除			
真	假		
year被400整除			
真	假	leap=1	leap=0
leap=1	leap=0		
leap == 1			
真	假		
输出“闰年”		输出“非闰年”	

```
C:\WINDOWS\system32\cmd.exe
enter year:2012
2012 is a leap year.
请按任意键继续. . .

C:\WINDOWS\system32\cmd.exe
enter year:2100
2100 is not a leap year.
请按任意键继续. . .
```

```
#include <stdio.h>
int main()
{
    int year, leap;
    printf("enter year:");
    scanf("%d", &year);
    if(year % 4 == 0)
    {
        if(year % 100 == 0)
        {
            if(year % 400 == 0)
                leap=1;
            else
                leap=0;
        }
        else
            leap=1;
    }
    else
        leap=0;
    if(leap)
        printf("%d is ", year);
    else
        printf("%d is not ", year);
    printf("a leap year.\n");
    return 0;
}
```

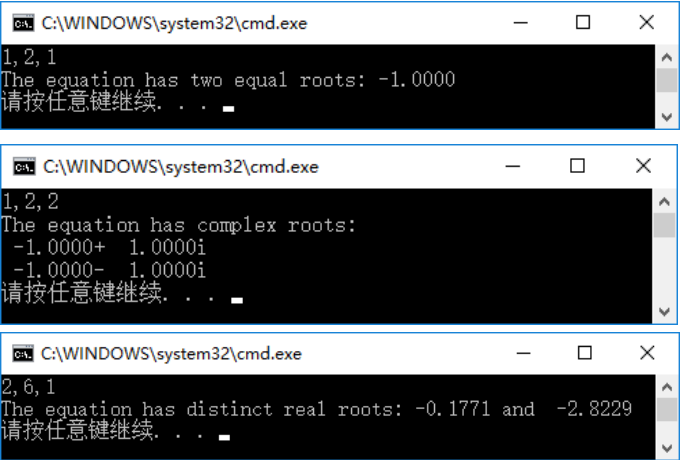
```
if(year%4!=0)
    leap=0;
else if (year%100!=0)
    leap=1;
else if(year%400!=0)
    leap=0;
else
    leap=1;
```

```
if((year%4==0 && year%100!=0)
|| (year%400==0))
    leap=1;
else
    leap=0;
```

# 选择结构程序综合举例

## 【例4.9】求 $ax^2+bx+c=0$ 方程的解。

输入a, b, c			
输出不是 “二次方程”	a=0		
	T	F	
	b <sup>2</sup> -4ac=0		
	T	F	
计算和输出两个相等的实根	b <sup>2</sup> -4ac>0		
	T	F	
	计算和输出两个不等实根		
	计算和输出两个共轭复根		



```
#include <stdio.h>
#include <math.h>
int main()
{
    double a, b, c, disc, x1, x2, realpart, imagpart;
    scanf("%lf, %lf, %lf", &a, &b, &c);
    printf("The equation ");
    if(fabs(a) <= 1e-6)
        printf("is not a quadratic\n");
    else
    {
        disc=b*b-4*a*c;
        if(fabs(disc) <= 1e-6)
            printf("has two equal roots:%8.4f\n", -b/(2*a));
        else
            if(disc>1e-6)
            {
                x1=(-b+sqrt(disc))/(2*a);
                x2=(-b-sqrt(disc))/(2*a);
                printf("has distinct real roots:%8.4f and %8.4f\n", x1, x2);
            }
            else
            {
                realpart=-b/(2*a); //realpart是复根的实部
                imagpart=sqrt(-disc)/(2*a); //imagpart是复根的虚部
                printf("has complex roots:\n");
                printf("%8.4f+%8.4fi\n", realpart, imagpart); //输出一个复数
                printf("%8.4f-%8.4fi\n", realpart, imagpart); //输出另一个复数
            }
    }
    return 0;
}
```

# 选择结构程序综合举例

【例4.10】运输公司对用户计算运输费用。  
路程越远，运费越低。标准如下：

$s < 250$	没有折扣
$250 \leq s < 500$	2%折扣
$500 \leq s < 1000$	5%折扣
$1000 \leq s < 2000$	8%折扣
$2000 \leq s < 3000$	10%折扣
$3000 \leq s$	15%折扣

p: 每吨每千米货物的基本运费

w: 货物重量

s: 运输距离

d: 折扣

f: 总运费

$$f = p * w * s * (1 - d)$$

```
#include <stdio.h>
int main()
{
    int c, s; // c是分类整数
    float p, w, d, f;
    printf("please enter price, weight, discount:"); //提示输入的数据
    scanf("%f, %f, %d", &p, &w, &s); //输入单价、重量、距离
    if (s >= 3000) c = 12; //3000km以上为同一折扣
    else c = s / 250; //3000km以下各段折扣不同，c的值不相同
    switch (c)
    {
        case 0: d = 0; break; //c=0, 代表250km以下, 折扣d=0
        case 1: d = 2; break; //c=1, 代表250~500km以下, 折扣d=2%
        case 2:
        case 3: d = 5; break; //c=2和3, 代表500~1000km, 折扣d=5%
        case 4:
        case 5:
        case 6:
        case 7: d = 8; break; //c=4~7, 代表1000~2000km, 折扣d=8%
        case 8:
        case 9:
        case 10:
        case 11: d = 10; break; //c=8~11, 代表2000~3000km, 折扣d=10%
        case 12: d = 15; break; //c=12, 代表3000km以上, 折扣d=15%
    }
    f = p * w * s * (1 - d / 100); //计算总运费
    printf("freight=%10.2f\n", f); //输出总运费，取两位小数
    return 0;
}
```

# 注意事项小结

1. 关系表达式和逻辑表达式的值为1或0;
2. `if(条件表达式)` 条件表达式可以是关系表达式, 逻辑表达式, 甚至可以是算术表达式; 表达式的值非0即真;
3. `if(a == b)` 与 `if(a = b)` 意义不同;
4. `if(a < c < b)` 与 `if(a < c && c < b)` 含义不同;
5. `else`与最近的`if`匹配, 必要时使用`{ }`;
6. `if { } else { }`; 复合语句必须用`{ }`;
7. `switch(x)` `x`必须是整型或字符型, `int x;` 或 `char x;`
8. `case 'a' :` 表示字符常量
9. 必要时`case`条件终止使用`break`; 否则执行下一条`case`语句。