# 5.1　关系运算符和逻辑运算符

## 一、关系运算符和关系表达式

### 1.　关系运算符

(1) 关系运算符的分类

C语言为用户提供了6个关系运算符：<(小于)、<=(小于等于)、>(大于)、>=(大于等于)、==(关系相等)和!=(关系不等)。

(2) 关系运算符的优先级

关系运算符内部优先级是<、<=、>、>=的优先级相等，且优先级高于==和!=。

在C语言所有运算符中，关系运算符的优先级要低于算术运算符和位移运算符。

(3) 关系运算符的结合性

所有的关系运算符的结合性都是从左至右。

### 2.　关系表达式的基本形式

有值表达式1 op 有值表达式2，其中op代表6个关系运算符。关系表达式的结果要么为0(操作数不满足关系运算)，要么为1(操作数满足关系运算)。

假设有以下变量定义：int a=1,b=-3,c=2,i=13;

以下都是合法的关系表达式：a>b、i<=10、a+b<b+c、(a=c)>(b=5)、a==5、a+b==2009、a!=b+2、b*b>=4*a*c等。

### 3.　关系运算符的使用

(1) 带有关系运算符的复杂表达式计算

**【例5.1】**　当a=3，b=2，c=1时，表达式f=a>b>c的值是_____。

**【例5.2】**　下面关系表达式中结果为假的是_____。

A) 0!=1　　B) 2<=8

C) (a=2*2)==2　　D) y=(2+2)==4

说明：在C语言中用一切非0数(**往往是1**)表示逻辑真，0表示逻辑假。

**【课内思考题5.1】**　当a=5,b=4,c=2时，表达式a>b!=c的值是_____。

(2) 关系运算中的隐式类型转换

**【例5.3】**　分析下面程序的输出结果，并说明原因。

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

```
{
int i=-10;
unsigned int j=1;
printf("i<j: %d\n",i<j);
}
```

(3) 怎样比较两个浮点数是否相等

有些浮点数在计算机内部是无法精确表示的，这主要是由于浮点数在计算机内部特殊的存储格式引起的，例如：

```
#include <stdio.h>
main()
{
double a=123456789.9*9,b=1111111109.1;
double c=0.1+0.2;
printf("c==0.3   %d\n",c==0.3);
printf("a==b       %d\n",a==b);
}
```

【例5.4】　比较两个浮点数是否相等的正确方法

```
#include <stdio.h>
#include <math.h>
#define YUZHI 1e-6
main()
{
double a=123456789.9*9,b=1111111109.1;
double c=0.1+0.2;
printf("a=%.9f,b=%.9f\n",a,b);
printf("a==b       %d\n",a==b);
printf("a==b       %d\n",fabs(a-b)<=YUZHI);
printf("c==0.3   %d\n",c==0.3);
printf("c==0.3   %d\n",fabs(c-0.3)<=YUZHI);
}
```

说明：应该避免直接比较两个浮点数是否相等，而是通过看两个浮点数差的绝对值是否小于某个很小的数来达到判断两个浮点数是否相等。

## 二、逻辑运算符和逻辑表达式

### 1.　逻辑运算符

(1) 逻辑运算符的分类

C语言为用户提供了3个逻辑运算符：!(逻辑非)、&&(逻辑与)和||(逻辑或)。

(2) 逻辑运算符的优先级

其中!(逻辑非)是单目运算符，在C语言所有运算符中优先级排第二(所有的单目运算符优先级都是第二)。&&(逻辑与)的优先级高于||(逻辑或)，||的优先级高于?:(条件运算符)。

(3) 逻辑运算符的结合性

其中!(逻辑非)的结合性是从右至左(所有的单目运算符的结合性都是从右至左)，&&和||的结合性是从左至右。

## 2. 逻辑表达式的基本形式

!有值表达式1，有值表达式1 && 有值表达式2，有值表达式1 || 有值表达式2。逻辑表达式的结果见下表：

| 有值表达式1 | 有值表达式2 | !有值表达式1 | 有值表达式1&&<br>有值表达式2 | 有值表达式1\|\|<br>有值表达式2 |
|---|---|---|---|---|
| 非0 | 非0 | 0 | 1 | 1 |
| 非0 | 0 | 0 | 0 | 1 |
| 0 | 非0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 |

说明：实际编程时，总是用逻辑运算符去连接关系表达式。

## 3. 逻辑运算符的使用

(1) 带有逻辑运算符的复杂表达式计算

【例5.5】 以下程序的运行结果是_____。

```
#include <stdio.h>
main()
{
int a,b,d=241;
a=d/100%9;
b=(-1)&&(-1);
printf("%d,%d",a,b);
}
```

A) 6,1　　B) 2,1　　C) 6,0

D) 2,0

**【课内思考题5.2】** 若a=2,b=4，则表达式!(x=a)||(y=b)&&0的值是_____。

**【课内思考题5.3】** int a=3,b=4,c=5;,则以下表达式!(a+b)+c-1&&b+c/2的值为_____。

(2) 短路计算

对于逻辑表达式“表达式1 && 表达式2 && … && 表达式n”而言，若在计算过程中发现表达式m的值为0，根据&&的特点，整个逻辑表达式的结果肯定为0，为提高计算效率，编译器不会计算从表达式m+1开始的后序表达式了，称为**短路计算**。

同样对于“表达式1 || 表达式2 ||…|| 表达式n”而言，只要发现某个表达式的值非0，肯定计算结果的同时也没有必要计算后面这些表达式的值。

短路计算在编程中有重要用途，能够为其他运算提供**卫士**：(number!=0) && (1/number>0.5)，由于在做除法时，除数不能为0，所以在计算1/number是否大于0.5之前，先判断变量number的值是不是0。

**【例5.6】** 设有说明语句：int a=1,b=2,c=3,d=4,m=2,n=2;，则执行(m=a>b) && (n=c>d)后n的值为____2____。

**【例5.7】** 执行以下语句后a的值为____0____，b的值为____6____。

```
int a=5,b=6,w=1,x=2,y=3,z=4;
(a=w>x) && (b=y>z);
```

## 四、用关系或逻辑表达式表示条件

要按照语义并使用关系或逻辑表达式去描述条件，且当条件成立时，表达式值为1；当条件不成立时，表达式值为0。

**【例5.8】** 设y为int变量，请写出描述“y是奇数”的表达式_____。

分析：首先，用来描述条件的表达式中必然包含变量y。其次，这个带有变量y的表达式的值只有两种结果：假如y的值是奇数的话，该表达式的计算结果为1；假如y是偶数的话，该表达式的计算结果为0。最后，按照语义来构造表达式——奇数的特点是被2除余1(也可认为不能被2整除)，不难得到：(y%2)==1(由%的优先级高于!=，所以也可写成y%2==1)。请思考还有没有别的写法？

**【课内思考题5.4】** 假设x是一个int变量，怎样用表达式表示条件“x被3整除，同时被5除余3”？

**【课内思考题5.5】** 假设x是一个double变量，怎样描述条件”2<x<3”(也即写一个表达式能判断x∈(2,3)中，如果x∈(2,3)，表达式值为1；如果x不在(2,3)，则表达式值为0)？

**【例5.9】** 若x、y、z均为int变量，则描述“x或y中有一个小于z”的表达式是_____。

**【例5.10】** 判别某年year是否闰年。闰年的条件是符合下面二者之一：能被4整除，但不能被100整除；能被400整除。可以用一个逻辑表达式来表示：

((year%4==0) && (year%100!=0)) || (year%400==0)，如果熟悉运算符的优先级，则表达式进一步简化为year%4==0 && year%100!=0 || year%400==0。

**【课内思考题5.6】** 判断char型变量ch是否为大写字母的正确表达式是_____无_____。

A) 'A'<=ch<='Z'　　B) (ch>='A') & (ch<='Z')

C) (ch>='0') && (ch<='9')　　D) (ch>='A') AND (ch<='Z')

## 5.2　用if语句和if-else语句实现选择控制流程

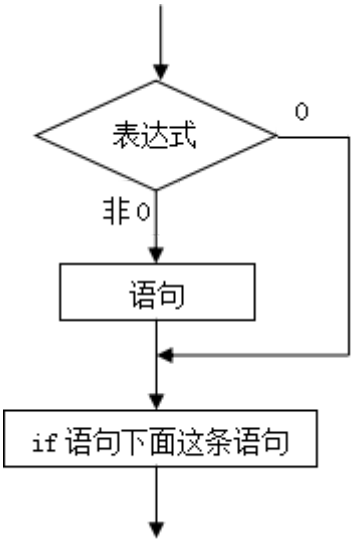
### 一、if语句

#### 1.　基本语法格式

if(表达式)

语句

说明：




(1) if语句的执行过程是先计算表达式的值，如果非0就执行包含在if中的语句；如果为0就不执行该语句。如下图所示：

(2) if语句中的表达式可以是任意的有值表达式，实际编程时往往使用**关系表达式**或将多个关系表达式用逻辑运算符连接起来的**逻辑表达式**。

(3) if中的语句可以是除了声明语句以外的其他语句，例如各种流程控制语句(if、if-else、switch、while、do-while和for)、流程转移语句(break、continue、return和goto)、表达式语句(包括函数调用语句)、空语句和复合语句。

2. 应用示例

【例5.11】 分析以下程序的执行结果。

```
#include <stdio.h>
main()
{
int i=1;
if(i<=0) printf("%d\n",i++);
printf("%d",--i);
}
```

【例5.13】 从键盘输入三个整数，计算其中的最大者并将其输出。

```
#include <stdio.h>
main()
{
int a,b,c;
int max;
scanf("%d,%d,%d",&a,&b,&c);
max=a; /*先假定变量a的值最大*/
if(max<b) max=b; /*比较变量max值和变量b值的大小,如果b大,就将b赋给max*/
if(max<c) max=c;  /*比较变量max值和变量c值的大小,如果c大,就将c赋给max*/
printf("the max number in %d,%d,%d, is %d\n",a,b,c,max);
}
```

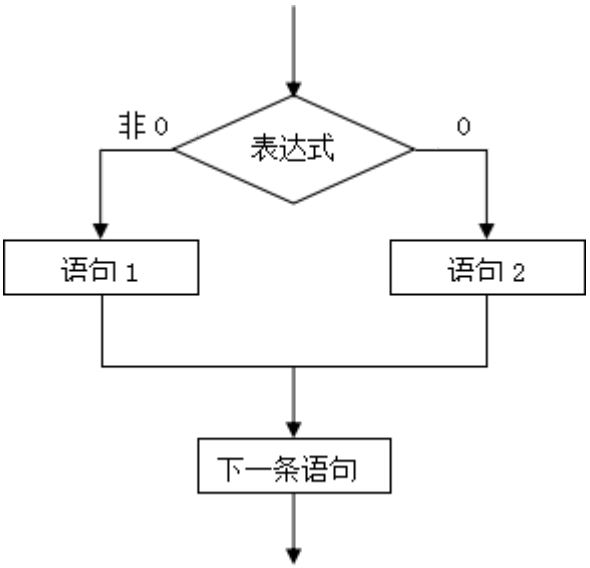
## 二、if-else语句

### 1. 基本语法格式

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

说明：

(1) if-else语句是由if子句和else子句组成的。具体执行过程是先计算表达式的值，如果非0就执行语句1；如果为0就执行语句2。如下图所示：




(2) 表达式和语句的说明同if语句。

## 2. 应用示例

【例5.13】 分析以下程序的执行结果。

```
#include <stdio.h>
main()
{
int a=10,b=50,c=30;
if(a=b+c) printf("*******\n");
else printf("$$$$$$$\n");
}
```

【例5.14】 输入两个实数，按代数值由小到大的次序输出这两个数。

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

```
scanf("%lf,%lf",&a,&b);
if(a>b)
    {
t=a;
a=b;
b=t;
        }
printf("%lf,%lf",a,b);
}
```

**【课内思考题5.7】** 如果将上面的if的复合语句的花括号去掉，根据输入值，结果如何？

本例也可改用if-else语句实现：

```
#include <stdio.h>
main()
{
double a,b;
scanf("%lf,%lf",&a,&b);
if(a>b)  printf("%lf,%lf",a,b);
        else printf("%lf,%lf",b,a);
}
```

**【例5.15】** 以下不正确的if语句形式是_____。

```
A) if(x>y && x!=y);
B) if(x==y) x+=y;
C) if(x!=y) scanf("%d",&x); else scanf("%d",&y);
D) if(x<y)x++;
    y++;
```

分析：选项A中的if后面是一个空语句(仅仅由;构成的语句)。空语句语法主要用在以下场合：**程序的某些地方不需要执行任何操作，但是由于语法的定义，这些地方必须要有语句，在这种情况下就不得不使用空语句。**

## 三、嵌套的if-else语句

### 1. if子句后面嵌套if语句或者if-else语句

有四种基本形式：

(1) if(表达式1)
        if(表达式2) 语句

(2) if(表达式1)
　　if(表达式2) 语句1
　　else 语句2
(3) if(表达式1)
　　if(表达式2) 语句1
　else 语句2
(4) if(表达式1)
　　if(表达式2) 语句1
　else 语句2
　　else 语句3

上面只给出一次嵌套的四种情况(其实是三种，(2)和(3)是等价的)，事实上if语句和if-else语句可以连续嵌套。

从(2)和(3)中不难发现，当if子句后面嵌套if语句或者if-else语句时，就会产生else子句究竟和哪个if子句配对的问题。为了解决表达上的歧义，C语言语法规定：在嵌套的if语句或者if-else语句中，else子句总是与**之前离它最近**的if子句配对。根据这条规则，那么(2)和(3)的语句结构就是相同了。

**【例5.16】** 以下程序的输出是_____。

```
#include <stdio.h>
main()
{
    int x=2,y=-1,z=2;
    if(x<y)
    if(y<0) z=0;
    else z+=1;
    printf("%d\n",z);
}
```

A) 3　　B) 2　　C) 1　　D) 0

**2. else子句后面嵌套if语句或者if-else语句**

有两种基本形式：

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

(2) if(表达式)

语句1

else

if(表达式2) 语句2

else 语句3

当if-else语句的else子句后面连续嵌套if-else语句时，就会出现一种非常有用编程结构——else-if结构：

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

说明：

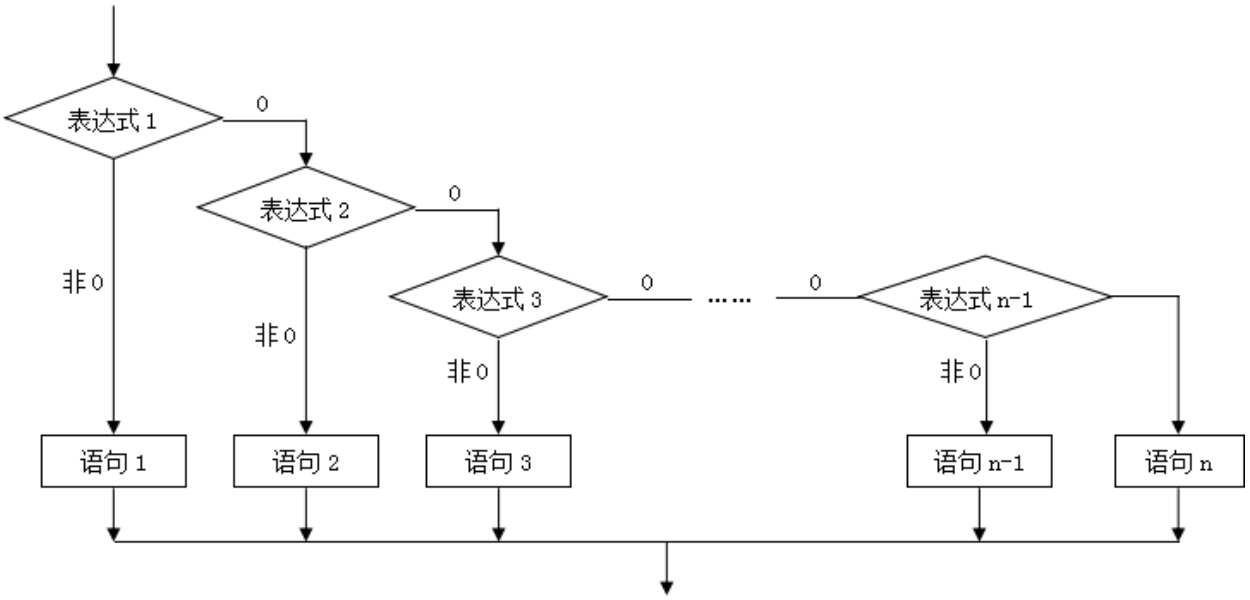
(1) 很多书认为else-if是 C语言中的一种选择语句，这种语法观点是错误的。确切地将else-if是一种编程结构，在多路条件判断时非常有用。if-else语句只能对一个条件进行一次判断，而else-if能对多个条件进行多次判断。

(2) 上述else-if结构是根据程序缩进的原则来书写的，一旦嵌套层次过多，就会造成程序段向右下倾斜。由于空白符(空格、换行和制表符等)会被编译器完全忽略，所以将每个else子句左对齐后就得到更为美观的形式：

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

(3) 最后一个else子句后面可以嵌套一个if语句。

(4) else-if结构的含义就是从上到下依次检测每个if子句中的表达式是否为0，执行第




一个表达式非0的if子句后面的语句，如下图所示。

【例5.17】　　成绩>=90分的同学成绩用A表示，80-90分的用B表示，70-80分的用C表示，60-70分的用D表示，60分以下输出FAIL!(本题改编自C程序100例之15)

```
#include <stdio.h>
main()
{
int  score;
printf("please input the score:\n");
scanf("%d",&score);
if(score>=90) printf("A\n");
else
        if(score>=80) printf("B\n");
        else
            if(score>=70) printf("C\n");
                else
                if(score>=60) printf("D\n");
```

```
                    else printf("FAIL!\n");
}
```

## 四、条件运算符

对一些简单的选择问题，完全可以由条件运算符(?:)实现。条件运算符是C语言中唯一的一个三目运算符，使用时必须连接三个表达式。条件表达式的基本形式为：

有值表达式1?有值表达式2:有值表达式3，执行过程如下：先计算有值表达式1的值，如果非0，就执行表达式2；如果为0就执行表达式3。如果表达式2和表达式3是有值表达式的话，那么整个条件表达式的值或者是表达式2的值或者是表达式3的值(视表达式1的值是否为0)。

?:的优先级高于赋值运算符(=)和逗号运算符(,)，排倒数第三，结合性是从右往左的。

**【例5.19】** 用条件运算符计算两个整型变量的较为大者并输出。

```
#include <stdio.h>
main()
{
int a,b;
int max;
scanf("%d,%d",&a,&b);
max=a>b?a:b;
printf("%d\n",max);
}
```

**【例5.20】** 分析以下程序程序的运行结果。

```
#include <stdio.h>
main()
{
int a;
scanf("%d",&a);
printf("%s",(a%2!=0)?"no":"yes");
}
```

**【课内思考题5.8】** 仿照本例程序的技法，实现以下问题：输入鸡蛋数n，当输入的鸡蛋数n=1时，输出“I have 1 egg.”；当输入的鸡蛋数n>1时(例如n=2009)，输出“I have 2009 eggs.”

**【课后作业3】**

(1) 以下程序的输出是_____。

```
#include <stdio.h>
main()
{
int a=5,b=8,c=3,max;
max=a;
if(c>b)
    if(c>a)  max=c;
    else
        if(b>a)  max=b;
printf("max=%d\n",max);
}
```

(2) 【同济大学99】阅读下面的程序并写出程序执行结果：

```
main()
{
int x,y,z,w;
z=(x=-1)?(y=-1,y+=x+5):(x=7,y=3);
w=y*'a'/4;
printf("%d %d %d %c\n",x,y,z,w);
}
```

## 5.3　用switch语句实现选择控制流程

### 一、switch语句

#### 1.　引例

当处理多个选项的时候，else-if结构有些笨重，为此C语言向用户提供了switch语句。为了对两者作一比较，特举例如下：

【例5.21】　从键盘输入一个星期的某一天，然后输出其对应的英文单词。

```
#include <stdio.h>
main()
{
int day;
scanf("%d",&day);
if(day==1)
      printf("Monday\n");
else if(day==2)
      printf("Tuesday\n");
```

```
else if(day==3)
      printf("Wednesday\n");
else if(day==4)
      printf("Thursday\n");
else if(day==5)
      printf("Friday\n");
else if(day==6)
      printf("Saturday");
else if(day==7)
      printf("Sunday\n");
else
      printf("the day between 1～7!\n");
}
```

说明：本例采用else-if结构对day值进行反复判断，一旦判断次数过多就会造成程序冗长。如改用switch语句实现，程序就能做到简洁易读。

## 2. 基本语法格式

```
switch(表达式)
{
        case 常量表达式1：语句序列1
        case 常量表达式2：语句序列2;
        ……
        case 常量表达式n：语句序列n
        [default：                语句序列n+1]
}
```

说明：

(1) switch语句的执行过程是依次比较每个case子句的常量表达式的值和switch圆括号里的表达式值是否相等，假设常量表达式k的值与switch圆括号里的表达式的值相等，就执行语句序列k，然后执行语句序列k+1直到最后一组语句序列。如果所有的常量表达式的值都不等于switch后面圆括号里的表达式值，且存在default子句(default子句是可选的)，就执行default子句后面的语句序列；不存在default子句，则结束对switch语句的执行。

(2) switch语句圆括号里的表达式的结果类型只能是整数类型、字符类型和枚举类型，case子句中的常量表达式(表达式中的操作数都是常量)的结果类型要与之匹配。

(3) 所有case子句中的常量表达式值都不重复。

改用switch语句实现例5.21：

```
#include <stdio.h>
main()
{
int day;
scanf("%d",&day);
        switch(day)
{
case 1:printf("Monday\n");
        case 2:printf("Tuesday\n");
case 3:printf("Wednesday\n");
case 4:printf("Thursday\n");
case 5:printf("Friday\n");
case 6:printf("Saturday\n");
    case 7:printf("Sunday\n");
        default:printf("the day between 1～7!\n");
}
}
```

说明：

(1) 在每个case子句的语句序列最后要安排一条break语句，目的是退出当前的case子句

避免执行后续case子句中的语句序列。

(2) case子句和default子句的出现次序是任意的，不会影响程序的结果。

(3) 当把default子句提前书写时，勿忘加在语句序列的最后写上break语句。

【例5.22】 以下程序的输出结果是_____。

```
#include <stdio.h>
main()
{
int x,y;
x=2;
switch(x)
{
case 1:
case 2:
case 3:
case 4:printf("x<5\n");
```

```
case 5:printf("x=5\n");
default:printf("The value of x is unknown.\n");
}
}
```

【例5.23】　计算某年某月天数。

```
#include <stdio.h>
main()
{
int year,month,days;
scanf("%d-%d",&year,&month);/*注意年月输入格式2009-4*/
        switch(month)
{
case 1:days=31;break;
case 2:if(year%4==0 && year%100!=0 || year%400==0)
                              days=29;
       else days=28;
       break;
case 3:days=31;break;
case 4:days=30;break;
case 5:days=31;break;
case 6:days=30;break;
case 7:days=31;break;
case 8:days=31;break;
case 9:days=30;break;
case 10:days=31;break;
case 11:days=30;break;
case 12:days=31;break;
}
printf("%d.%d has %d days\n",year,month,days);
}
```

【课内思考题5.9】　从简化语句的角度对上述程序进行优化。

*【例5.24】　输入某年某月某日，判断这一天是这一年的第几天？(本题改编自C程序100例之4)

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

```
main()
{
int year,month,day;
int sumdays=0;
printf("please input the date,for an example 2009.4.14:\n");
scanf("%d.%d.%d",&year,&month,&day);/输入格式为：2009.4.13*/
        switch(1)
{
case 1: if(month==1) break;sumdays=sumdays+31;
case 2: if(month==2) break;
        if(year%4==0 && year%100!=0 || year%400==0)
      sumdays=sumdays+29;
        else sumdays+=28;
case 3: if(month==3) break;sumdays+=31;
case 4: if(month==4) break;sumdays+=30;
    case 5: if(month==5) break;sumdays+=31;
case 6: if(month==6) break;sumdays+=30;
case 7: if(month==7) break;sumdays+=31;
case 8: if(month==8) break;sumdays+=31;
        case 9: if(month==9) break;sumdays+=30;
case 10:if(month==10) break;sumdays+=31;
case 11:if(month==11) break;sumdays+=30;
}
sumdays+=day;
printf("%d.%d.%d is the %dth day\n",year,month,day,sumdays);}
```

说明：本例中的switch语句在实现时，创造性地将break语句作为if语句的内嵌语句，有条件地退出case子句，体现了用C语言语法的灵活性和自由性。

【例5.25】　以下程序的输出结果是_____。

```
#include <stdio.h>
main()
{
int a=2,b=7,c=5;
switch(a>0)
{
```

```
      case 1:switch(b<0)
    {
                          case 1:printf("@");break;
  case 2:printf("!");break;
    }
      case 0:switch(c==5)
    {
                          case 0:printf("*");break;
  case 1:printf("#");break;
  default:printf("#");break;
    }
      default:printf("&");
}
printf("\n");
}
```

说明：本例给出了在switch语句的case子句中嵌套switch语句的例子。

【例5.26】 有一分段函数：

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

用switch语句编一程序输入一个x的值，输出y的值。

分析：本例说明switch语句也可以实现一般的选择结构。

```
#include <stdio.h>
main()
{
```

```
double x,y;
scanf("%lf",&x);
switch(x>=0.0)
{
case 0:y=-1.0;break;
case 1:switch(x>0.0)
               {
                        case 0:y=0.0;break;
                        case 1:y=1.0;
               }
}
printf("y=%.1f\n",y);
}
```