



## 第3章 分支选择结构

选择结构是通过对条件的判断来选择执行不同的程序语句。

在C语言中是用if语句或switch语句来构成选择结构的。if语句一般适用于两路选择，也可以通过嵌套形式来实现多路选择。switch语句能方便地实现多路选择。



## ● 结构化程序设计的三种基本结构？

- 顺序结构
- 选择结构
- 循环结构



# 主要内容

- 1 关系运算符和逻辑运算符
- 2 if语句
- 3 switch语句

# 谁会得到面试通知

- 例题要求：
- 假设一家大型药厂面试求职者，满足某些教育条件的求职者可得到面试机会。
- 满足如下条件的求职者会接到面试通知：
  - 25岁以上，化学专业毕业生，但不是毕业于北京大学
  - 北京大学化学专业毕业生
  - 28岁以下，清华大学经济学专业毕业生
  - 25岁以上，北京大学非化学专业毕业生

# 谁会得到面试通知

## ● 分析与设计

药厂选择面试候选人的条件和对应的表达式如下表所示。

条件	表达式（毕业院校：1、清华大学 2、北京大学 3、其他；专业：1、化学 2、经济3、其他）
25岁以上，化学专业毕业生，但不是毕业于北京大学	<code>age&gt;25&amp;&amp;subject==1&amp;&amp;college!=2</code>
北京大学化学专业毕业生	<code>college==2&amp;&amp;subject==1</code>
28岁以下，清华大学经济学专业毕业生	<code>college==1&amp;&amp;subject==2&amp;&amp;age&lt;28</code>
25岁以上，北京大学非化学专业毕业生	<code>college==2&amp;&amp;age&gt;25&amp;&amp;subject!=1</code>



# 谁会得到面试通知

## ● 编码实现:

```
if (age>25&&subject==1&&college!=2)
    interview = 1;
if (college==2 &&subject ==1)
    interview = 1;
if (college==1&&subject==2&&age<28)
    interview = 1;
if (college==2&&age>25&&subject!=1)
    interview = 1;
if (interview)          /* interview=1, 能够取得面试机会 */
    printf("\n\n恭喜你得到面试机会!");
else                    /* interview=0, 没有面试机会 */
    printf("\n\n抱歉, 你没有面试机会!");
}
```

# 关系运算符

- 关系表达式是用关系运算符和括号将运算对象（常量、变量、函数等）连接起来的式子。
- 关系运算符共有以下**六种**：
  - $<$ 、 $<=$ 、 $>$ 、 $>=$ 、
  - $==$ 、 $!=$
- **关系运算符优先级低于算术运算符，高于赋值运算符**

- 例如：
- $a > b + c$  等价于  $a > (b + c)$
- $a > b == c$  等价于  $(a > b) == c$
- $a = b > c$  等价于  $a = (b > c)$

# 关系运算符

- 注意:

- 1) 进行比较时**一定要用双等号 (==)**，单个等号是赋值运算符，不要混淆;
- 2) 不能将浮点变量用 == 或者 != 与任何数字比较;

例如:

```
float x=3.26;  
if (x==0.0)
```

设法转换成“>=”或 “<=”

```
const float EPSINON =  
0.00001  
float x=3.26;  
if (x>=EPSINON)
```





# 逻辑运算符和逻辑表达式

- 逻辑表达式是用逻辑运算符和括号将运算对象连接起来的式子，它的值反应了逻辑运算的结果。C语言提供的逻辑运算符有以下3种：

■ &&          (逻辑与)

■ ||          (逻辑或)

■ !          (逻辑非)

优先级：

逻辑非>逻辑与>逻辑或

运算规则：

a && b      当a和b都为真时，结果为真；否则结果为假。

a || b      如果a和b之一为真，则结果为真；

如果a和b都为假，结果为假。

! a      如果a为真，结果为假；如果a为假，结果为真。

# 逻辑运算符和逻辑表达式

## 逻辑运算的真值表

a	b	!a	!b	a&&b	a  b
真	真	假	假	真	真
真	假	假	真	假	真
假	真	真	假	假	真
假	假	真	真	假	假

**逻辑真：值为1**

**逻辑假：值为0**

# 各种运算符优先级

! (非)  
算术运算符  
关系运算符  
&&  
||  
赋值运算符

高

低

例如:

$x < y \&\& m < n$   
等效于  $(x < y) \&\& (m < n)$

$a = b \parallel c = d$   
等效于  $(a = b) \parallel (c = d)$

$!a \&\& b > c$   
等效于  $(!a) \&\& (b > c)$

$5 > 3 \&\& 4 \parallel 8 < 4$   
等效于  $((5 > 3) \&\& 4) \parallel (8 < 4)$



# 逻辑运算符和逻辑表达式

例：写出下面各逻辑表达式的值。

设  $a=4$ ,  $b=5$ ,  $c=5$ 。

- $a+b < c \&\& b == c$       值为0
- $a || b + c \&\& b - c$       值为1
- $!(a > b) \&\& !c || 1$       值为1
- $!(x = a) \&\& (y = b) \&\& 0$       值为0
- $!(a + b) + c - 1 \&\& b + c / 2$       值为1

优先级：

逻辑非 > 逻辑与 > 逻辑或

# 逻辑运算符和逻辑表达式

- 逻辑表达式中**任何非零的数值都被认作为“真”**。
- 在逻辑表达式的求解中，并不是所有的逻辑运算符都会被执行到。
  - ① `a&&b&&c`：只有a为真时，才需要判断b的值，只有a和b都为真时，才需要判断c的值。
  - ② `a||b||c`：只要a为真，就不必判断b和c的值，只有a为假，才需判断b。a和b都为假才判断c。

即当左值可以决定整个表达式的值时，就不再求右边表达式的值，因此对于与(`&&`)运算来说，左值为0，就不再继续后面的运算；对于或(`||`)运算来说，左值为1，就不再继续后面的运算，这样可以提高运行速度。



# 逻辑运算符和逻辑表达式

● 例:  $(m=a>b)\&\&(n=c>d)$

- 当  $a=1, b=2, c=3, d=4$ ,  $m$  和  $n$  的原值为 1 时, 由于 “ $a>b$ ” 的值为 0, 因此  $m=0$ , 即可判断出表达式  $(m=a>b)\&\&(n=c>d)$  的值为 0, 而不必再求 “ $n=c>d$ ” 的值, 因此  $n$  的值不是 0 而仍保持原值 1。

# 条件运算符和条件表达式

- 条件表达式是由**条件运算符(? :)**把3个表达式连接起来的式子，其形式为：
  - 表达式1? 表达式2: 表达式3
- 求解过程：先判断表达式1的值是否为真（非0），若为真，则求解表达式2，表达式2的值就是整个条件表达式的值；若为假（0），则求解表达式3，表达式3的值就是整个条件表达式的值。

```
maxValue = ( a > b ) ? a : b;
```

Equivalent to:

```
if ( a > b )
```

```
    maxValue = a;
```

```
else
```

```
    maxValue = b;
```

# 条件运算符和条件表达式

例: 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。

```
#include "stdio.h"
void main ( )
{
    char ch;
    printf("请输入一个字符: ");
    scanf("%c",&ch);    /*输入一个字符*/
    ch=(ch>=' A' &&ch<=' Z' ) ? (ch+32): ch;
    printf("%c\n",ch);/*输出最后得到的字母*/
}
```

# 主要内容

- 1 关系运算符和逻辑运算符
- 2 if语句
- 3 switch语句

# if语句

- if语句也称为条件语句，用于实现程序的选择结构。if语句通过判断给定的条件（一般是关系表达式或逻辑表达式）是否成立来控制执行不同的程序语句，完成相应的功能。
- if语句有3种语法形式，构成了3种选择结构。
  - 简单选择结构
  - 二路选择结构
  - 多路选择结构



# 简单选择结构

- 语句形式为：

**if (表达式)**  
**语句;**

- 例如：

**if(x > y)**

**printf( "%d" ,x);**

- 执行过程：如果表达式的值为真（非0值），则执行语句；如果表达式的值为假（0值），则跳过该语句继续执行后续程序。其过程可表示为图3-1。

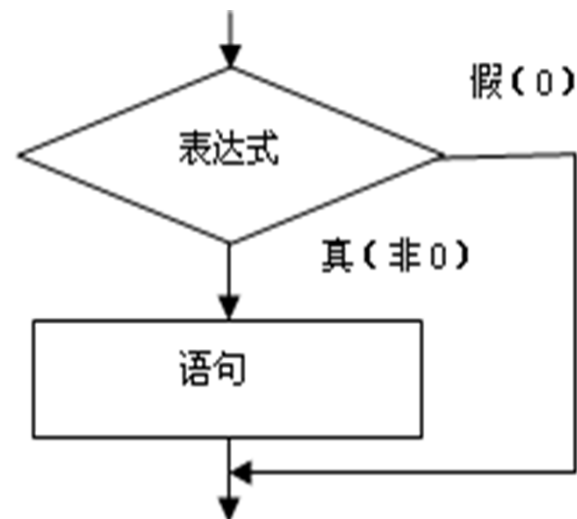


图 3-1 简单选择结构

# 简单选择结构

例：输入三个数  $a, b, c$ , 要求按由大到小的顺序输出。

分析：

该程序假定处理后， $a$ 中放三个数中的最大值， $b$ 中放三个数中的次大值， $c$ 中放三个数中的最小值。

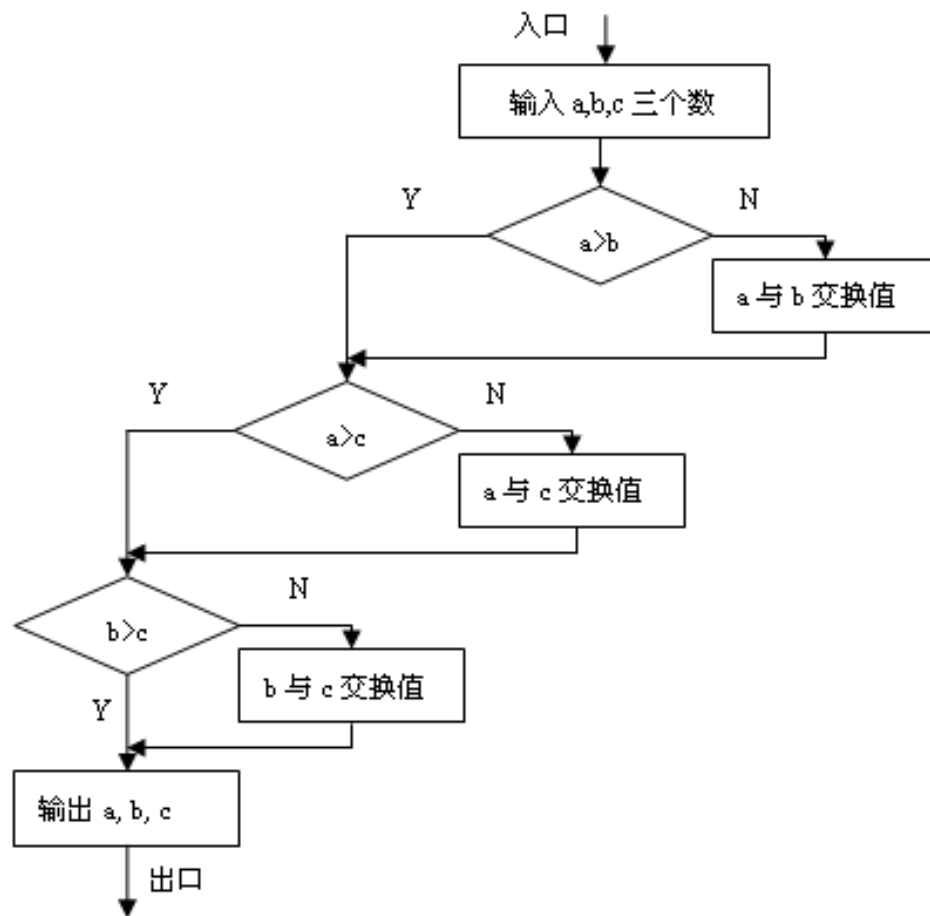


图 3-2 例 3\_5 程序流程图

# 简单选择结构

**if(a<b)**

**{t=a;a=b;b=t;} /\*a,b两个变量值进行交换\*/**

**if(a<c)**

**{t=a;a=c;c=t;} /\*a,c两个变量值进行交换\*/**

**if(b<c)**

**{t=b;b=c;c=t;} /\*b,c两个变量值进行交换\*/**

**当选择结构的执行语句超过一句时，一定要用一对大括号括起来。**

# 二路选择结构

- 语句形式为：  
if(表达式)  
    语句1;  
else  
    语句2;

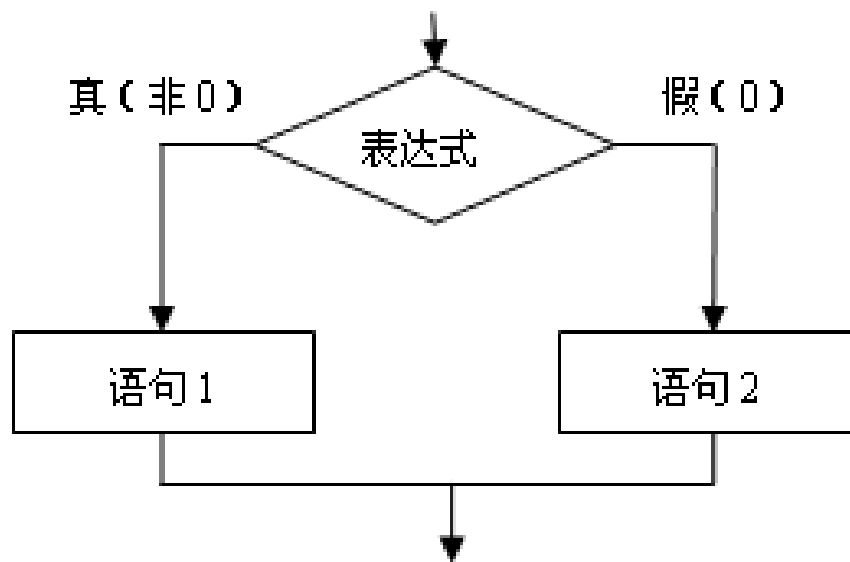


图 3-3 二路选择结构

- 执行过程：  
如果表达式的值为真（非0值），则执行语句1，否则执行语句2。其过程可表示为图3-3。

## 二路选择结构

例：从键盘上输入年号，判断这一年是否为闰年，若是闰年则输出“Y”，否则输出“N”。

分析：闰年的判断依据是：若某年号能被4整除但不能被100整除，则这一年是闰年；或者这一年能被400整除也是闰年。判断闰年的逻辑表达式为：

- $(year \% 4 == 0 \& \& year \% 100 != 0) || year \% 400 == 0$

```
if((year%4==0&&year%100!=0)||year%400==0)
    p=1;        /*是闰年，将标志变量p置为1*/
else
    p=0;        /*不是闰年，将标志变量p置为0*/
if(p) printf("Y \n");    /*变量p的值为真输出Y*/
else printf("N \n");    /*变量p的值为假输出N*/
```



# 多路选择结构

- 语句形式为:

```
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n;
```

执行过程：依次判断表达式的值，当出现某个值为真时，则执行其对应的语句，然后跳到整个if语句之外继续执行程序。如果所有的表达式值均为假，则执行语句n。然后继续执行后续程序。

# 多路选择结构

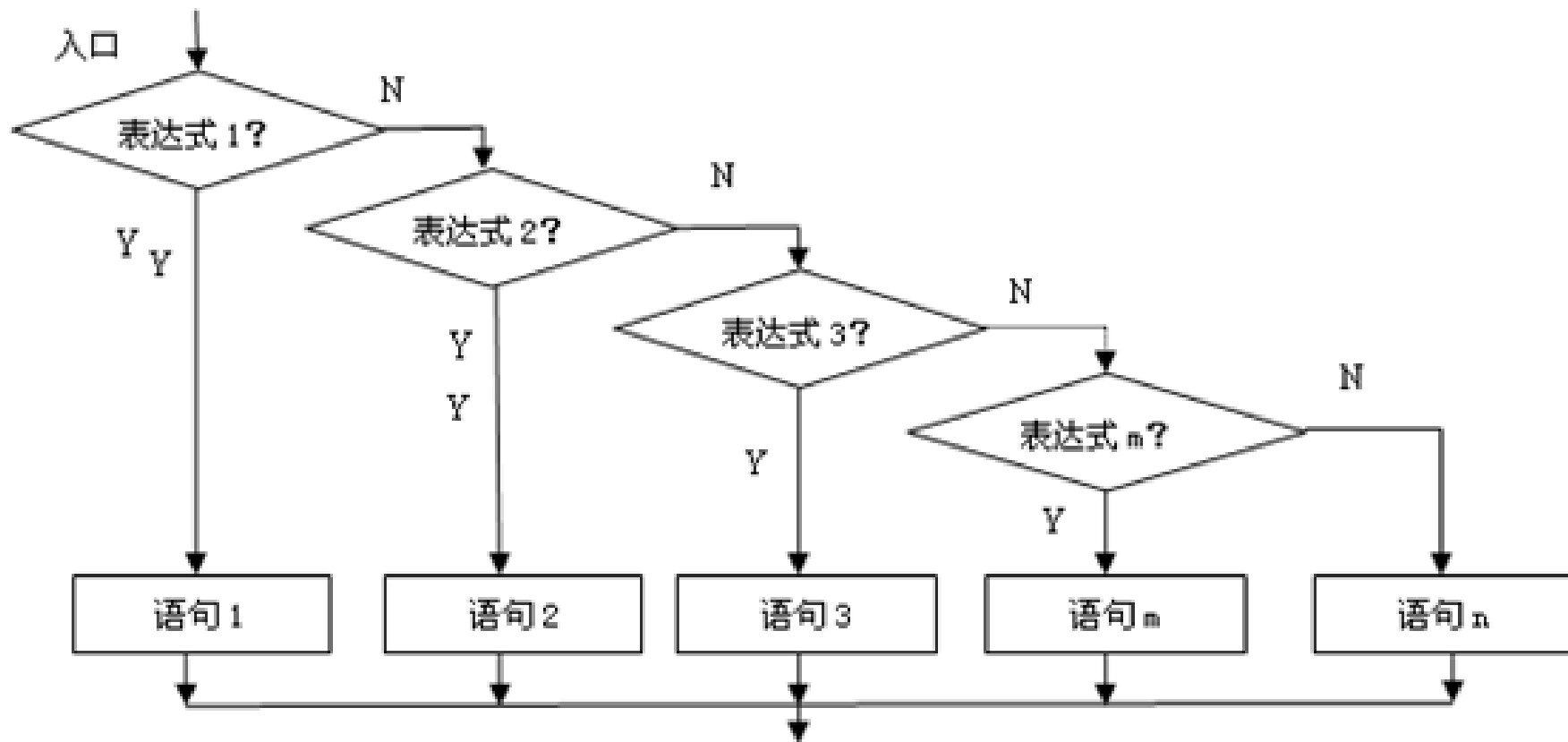


图 3-4 多路选择结构

# 多路选择结构

例：从键盘上输入应纳税所得额，计算并输出应纳个人所得税。

分析：我国最新个人所得税计算公式如下：

应纳个人所得税额 = (应纳税所得额 - 扣除标准) × 适用税率 - 速算扣除数

按照我国2011年9月1日开始实行的新个人所得税扣除标准为3500元/月，超过3500元的收入所得部分按分段税率扣除，超过3500元的收入所得部分的分段使用税率规定如下：

- (1) 不超过1500元的部分，税率3%，速算扣除数为0
- (2) 超过1500元至4500元的部分，税率10%，速算扣除数为105
- (3) 超过4500元至9000元的部分，税率20%，速算扣除数为555
- (4) 超过9000元至35000元的部分，税率25%，速算扣除数为1005
- (5) 超过35000元至55000元的部分，税率30%，速算扣除数为2755
- (6) 超过55000元至80000元的部分，税率35%，速算扣除数为5505
- (7) 超过80000元的部分，税率45%，速算扣除数为13505

# 多路选择结构

```
if(x<=1500.0)          /*应上税部分的金额≤1500*/  
    yngrsdsse=x*0.03;  
else if(x<=4500)       /*1500<应上税部分的金额≤4500*/  
    yngrsdsse=x*0.1-105;  
else if(x<=9000)       /*4500<应上税部分的金额≤9000*/  
    yngrsdsse=x*0.20-555;  
else if(x<=35000)      /*9000<应上税部分的金额≤35000*/  
    yngrsdsse=x*0.25-1005;  
else if(x<=55000)      /*35000<应上税部分的金额≤55000*/  
    yngrsdsse=x*0.30-2755;  
else if(x<=80000)      /*55000<应上税部分的金额≤80000*/  
    yngrsdsse=x*0.35-5505;  
else if(x>80000)       /*应上税部分的金额>80000*/  
    yngrsdsse=x*0.45-13505;
```

# 多路选择结构

- 多路选择结构为**多选一**的结构。
- 如果当所有条件均不成立时，也不需要完成任何操作，则**可以省略else子句**。
- 对if语句的几点说明：
  - 3种形式的if语句中，在if后面都有表达式，**一般为逻辑表达式或关系表达式**。
  - 第二、第三种形式的if语句中，**在每个else前面有一个分号，整个语句结束处有一个分号**。
  - 在if和else后面可以只含有一个内嵌的操作语句，也可以有多个操作语句，此时**用花括号将几个语句括起来成为一个复合语句**。



# if语句的嵌套

- 在if语句中又包含一个或多个if语句就称为if语句的嵌套。其一般形式为：

```
if ()  
    if ()  语句1  
    else  语句2  
else  
    if ()  语句3  
    else  语句4
```

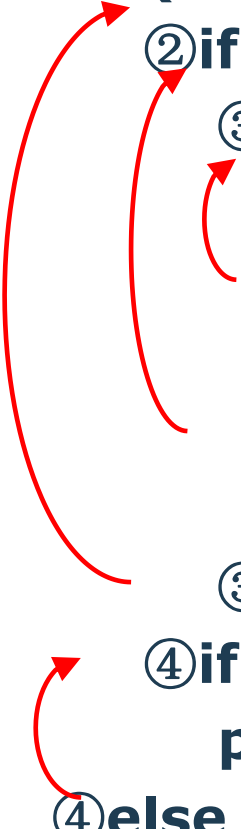
if和else的配对原则为：

else总是与它上面的，最近的，同一复合语句中的，未配对的if语句配对。

# if语句的嵌套

- 分析以下程序段的配对关系？

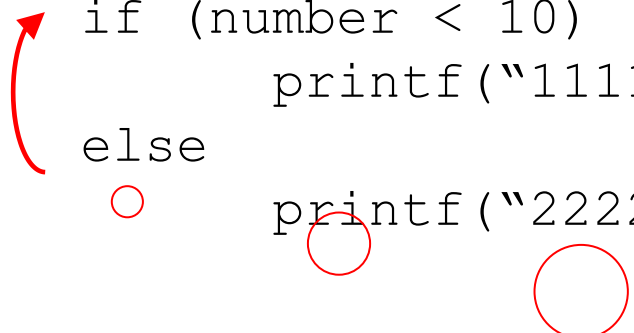
```
①if(score<80)
    ②if(score<70)
        ③if(score<60)
            printf("E\n");
        ①else
            printf("D\n");
        ②else
            printf("C\n");
    ③else
        ④if(score<90)
            printf("B\n");
        ④else
            printf("A\n");
```



注意：配对时将从第一个else开始向上寻找与其配对的if语句。

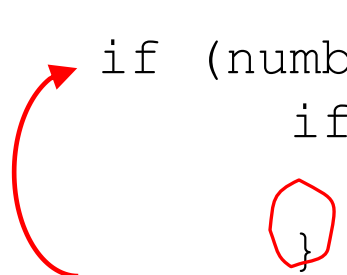
# if语句的嵌套

```
if (number > 5)
    if (number < 10)
        printf("1111\n");
    else
        printf("2222\n");
```



Rule: an else goes with the **most recent if**, unless braces indicate otherwise

```
if (number > 5) {
    if (number < 10)
        printf("1111\n");
}
else
    printf("2222\n");
```



# 实战演练:计算分段水价

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

```
    double x, y;
    printf("Enter x:");
    scanf("%f", &x);
```

```
    if (?) {
```

```
        y = 0;
```

```
    }
```

```
    else if (?) {
```

```
        y = 4 * x / 3;
```

```
    }
```

```
    else {
```

```
        y = 2.5 * x - 10.5;
```

```
    }
```

```
    printf("f(%.2f) = %.2f\n", x, y);
```

```
    return 0;
```



$$y = f(x) = \begin{cases} 0 & x < 0 \\ \frac{4x}{3} & 0 \leq x \leq 15 \\ 2.5x - 10.5 & x > 15 \end{cases}$$

Enter x: **-0.5**

f(-0.50) = 0.00

Enter x: **9.5**

f(9.50) = 12.67

Enter x: **21.3**

f(21.30) = 42.75

# 实战演练: 简单计算器

```
/* Program to evaluate simple expressions of the form
number operator number */
#include <stdio.h>
int main (void) {
    float value1, value2;
    char operator;
    printf ("Type in your expression.\n");
    scanf ("%f %c %f", &value1, &operator, &value2);
    if ( operator == '+' )
        printf (".2f\n", value1 + value2);
    else if ( operator == '-' )
        printf (".2f\n", value1 - value2);
    else if ( operator == '*' )
        printf (".2f\n", value1 * value2);
    else if ( operator == '/' )
        printf (".2f\n", value1 / value2);
    else printf ("Unknown operator.\n");
    return 0;
}
```

# 主要内容

- 1 关系运算符和逻辑运算符
- 2 if语句
- 3 switch语句



# switch语句

- 在某抽奖活动中幸运数字为1、2、3，如果参与者猜测出其中一个幸运数字则显示出可能赢得的奖品（选择1可以赢得一辆车，选择2可以赢得一台彩电，选择3可以赢得一台电脑），如果猜测这三个数字之外的数字则显示“没有奖品给你”。
- 分析与设计  
该题目可以用之前介绍的if多路选择结构完成，**但用switch语句更加简洁。**

# switch语句

## ● 例：选择幸运数字

```
switch(choice)           /*根据选择的数字显示相应的信息*/
{
    case 1: printf(“恭喜你得到一辆车!”); break;
    case 2: printf(“恭喜你得到一台彩电!”); break;
    case 3: printf(“恭喜你得到一台电脑!”); break;
    case 4:
    case 5:
    case 6:
    case 7:
    case 8:
    case 9: printf(“很遗憾，没有奖品给你”); break;
    default: printf(“请输入1—9中的数字!”);
}
```

# switch语句

**switch**语句的一般形式:

**switch**(表达式)

{

**case** 常量表达式1: 语句1;  
                                  **break;**

**case** 常量表达式2: 语句2;  
                                  **break;**

...

**case** 常量表达式n: 语句n;  
                                  **break;**

**default:**语句n+1;

}

执行过程: 首先计算switch语句后的表达式的值, 再依次与1到n个常量表达式的值进行比较, 当表达式的值与某个case后的常量表达式的值相等时, 则执行该case后的语句, 然后执行break语句跳出switch结构。如果所有常量表达式的值都不等于switch中表达式的值, 则执行default后的语句。

# switch语句

switch语句执行过程如图3-7所示。

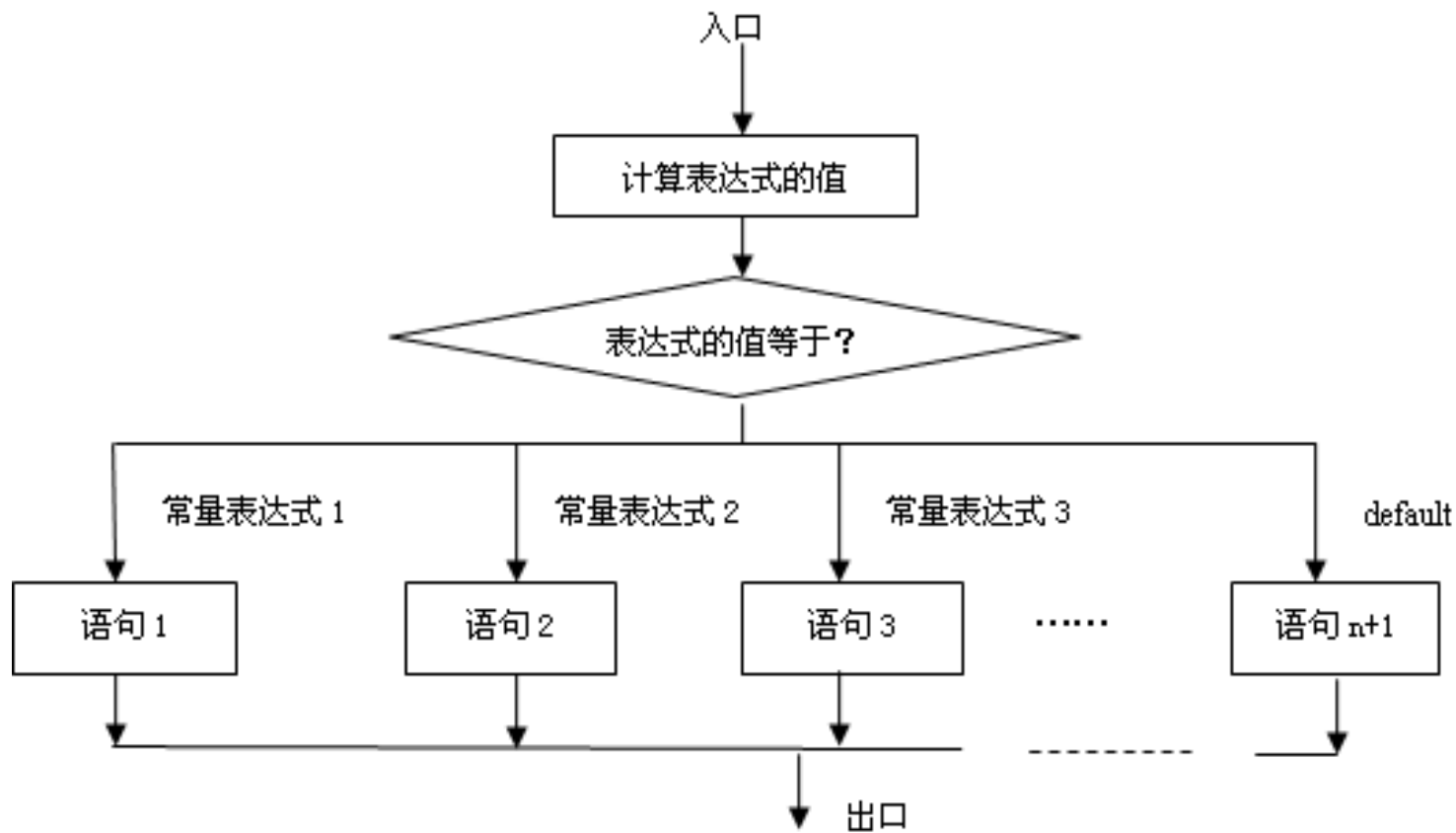


图 3-7 switch 结构流程图

# switch语句

例: case后无break语句的例子(落空)

```
#include "stdio.h"
```

```
void main()
```

```
{
```

```
    int c=8;
```

```
    switch (c<10?1:c<25?2:c<35?3:4) {
```

```
        case 1:
```

```
            printf("%d°C 有点冷\n ",c);
```

```
        case 2:
```

```
            printf("%d°C 正合适\n ",c);
```

```
        case 3:
```

```
            printf("%d°C 有点热\n ",c);
```

```
        default:
```

```
            printf("%d°C 太热了\n ",c);
```

```
    }
```

```
}
```

```
8°C 有点冷
8°C 正合适
8°C 有点热
8°C 太热了
```

```
-----
Process exited after 0.0341 second
请按任意键继续. . .
```

注意: 通常在每一个case中都应使用break语句提供一个出口, 使流程跳出switch语句。

若没有使用break语句则第一个满足条件的case后面的所有语句都会被执行, 这种情况叫做落空。

# switch语句的一些说明

- (1) case后面的常量表达式的值应该与switch后面的表达式的值**类型一致**，都**必须为整型或字符型**，**不允许为浮点型**。
- (2) 同一个switch语句中所有case后面的**常量表达式的值都必须互不相同**，否则会出现互相矛盾的现象。
- (3) 当switch表达式的值与某一个case子句中的常量表达式的值相匹配时，就执行此case子句中的内嵌语句，若所有的case子句中的常量表达式的值都不能与switch表达式的值匹配，就执行default子句的内嵌语句。
- (4) 各个case和default子句的先后顺序可以变动，而不会影响程序执行结果，但要注意：**如果default子句前置，后边要加break语句结果才正确**，只有最后的分支语句可以不加break而不影响结果。





# switch语句的一些说明

(5) **多个case可以共用一组执行语句**，例如：

case 4:

case 5:

case 6:

case 7:

case 8:

case 9: printf(“很遗憾，没有奖品给你”);

当输入4, 5, 6, 7, 8, 9时都在屏幕上显示“很遗憾，没有奖品给你”。

(6) 每个case后面**可以是一个语句，也可以是多个语句，还可以没有语句**。当是多个语句时**可不用花括号括起来**。

# switch语句的一些说明

Break can miss !

Statement list on a case can miss !

```
switch (operator)
{
    ...
    case '*':
    case 'x':
        printf (".2f\n", value1 * value2);
        break;
    ...
}
```

No two case values can be the same!



# 实战演练: 计算成绩

// 根据分数给出分段成绩

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

```
int main() {
```

```
    int score, grade;
```

```
    printf("Input a score(0~100): ");    scanf("%d", &score);
```

```
    grade = score/10;
```

```
    switch (grade) {
```

```
        case 10:
```

```
        case 9: printf("grade=A\n"); break;
```

```
        case 8: printf("grade=B\n"); break;
```

```
        case 7: printf("grade=C\n"); break;
```

```
        case 6: printf("grade=D\n"); break;
```

```
        case 5:
```

```
        case 4:
```

```
        case 3:
```

```
        case 2:
```

```
        case 1:
```

```
        case 0: printf("grade=E\n"); break;
```

```
        default: printf("The score is out of range!\n");
```

```
    }  
}
```

# 实战演练: 简单计算器 (switch-case)

```
# include <stdio.h>
int main(void)
{ char operator; float value1, value2;
  printf("Type in your expression: ");
  scanf("%f%c%f", &value1, &operator, &value2);
  switch(operator){
    case '+':
      printf("=%.2f\n", value1+value2);
      break;
    case '-':
      printf("=%.2f\n", value1-value2);
      break;
    case '*':
      printf("=%.2f\n", value1*value2);
      break;
    case '/':
      printf("=%.2f\n", value1/value2);
      break;
    default:
      printf("Unknown operator\n");
      break;
  }
  return 0;
```



# switch和if-else的比较

- switch结构和if-else if-else多路选择结构都能实现多分支选择结构，但**两者各有优势**。
- if-else比switch的条件控制更强大一些：if-else可以依照各种逻辑运算的结果进行流程控制，而 switch只能进行 == 判断，并且只能是整数判断。
- switch的结构比if-else更清晰。

注意：两者都要尽量避免用得过多、过长，尤其不要嵌套得太多，它们会大大增加程序的分支，使逻辑关系显得混乱，不易维护，易出错。



# 关系逻辑运算 补充

1) 假设 $a=1$ ,  $b=2$ ,  $c=3$ ,  $d=4$ ,  $m=1$ ,  $n=1$ , 执行表达式  $(m=a>b) \&\& (n=c>d)$  后 $m$ 和 $n$ 的值各为多少?

分析:  $a>b$ 的值为0  $\rightarrow (m=a>b)$  值为0  $\rightarrow$  表达式  $(m=a>b) \&\& (n=c>d)$  的值为0, 运算就此结束, 即不再进行  $(n=c>d)$  的运算。因此结果是:  $m$ 为0,  $n$ 为1。

2) 假设 $a=5$ 、 $b=10$ 、 $c=15$ 、 $d=20$ , 求下列逻辑表达式的结果。

$a<c \&\& c<b$ 、 $a==b || c!=d$

$!8+3 \&\& 0$ :

由原式  $\rightarrow 0+3 \&\& 0 \rightarrow 1 \&\& 0 \rightarrow 0$ 。

$!a || (a>b)+1 \&\& c<d$ :

由原式  $\rightarrow !5 || (a>b)+1 \&\& c<d$

$\rightarrow 0 || (a>b)+1 \&\& c<d$

$\rightarrow 0 || 0+1 \&\& c<d \rightarrow 0 || 1 \&\& c<d$

$\rightarrow 0 || 1 \&\& 15<20 \rightarrow 0 || 1 \&\& 1 \rightarrow 0 || 1 \rightarrow 1$



# scanf()补充

- 输入数据中没有字符型时，在两个输入数据间可以用空格、回车键、Tab间隔；
- 输入多个字符时，**这些字符间不能有间隔**。如果使用了间隔符(如空格或回车' \n' )，由于它本身也是字符，该间隔符就被作为输入字符。

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

```
int main(void)
```

```
{
```

```
    char ch1, ch2, ch3;
```

```
    scanf("%c%c%c", &ch1, &ch2, &ch3);
```

```
    printf("%c%c%c%c%c", ch1, '#', ch2, '#', ch3);
```

```
    return 0;
```

```
}
```

AbC

A#b#C

A bC

A# #b





**Thank you**

