

控制结构

Wang Houfeng

EECS, PKU

wanghf@pku.edu.cn

内容

➤ 分支结构

- 循环（重复）结构

程序的三种基本控制结构

- 三种基本控制结构

(理论上证明, 程序仅需包含3种基本控制)

- 顺序结构

- 分支 (选择)

- 重复 (循环)



C. Bohm & G. Jacopini, "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules," Communications of the ACM, vol9(5) May 1966, pp 366-371.

顺序结构例子

```
#include <iostream>
using namespace std;
int main()
{   int a,b,sum;
    a=10;
    b=24;
    sum=a+b;
    cout<<"sum="<<sum<<endl;
    return 0;
}
```

按书写的顺序执行语句



分支结构: 改变执行顺序

- 求解问题时, 可能面临多情况的选择, 需要决定选择哪种情况
- 例: 求解一元二次方程:

$$ax^2 + bx + c = 0$$

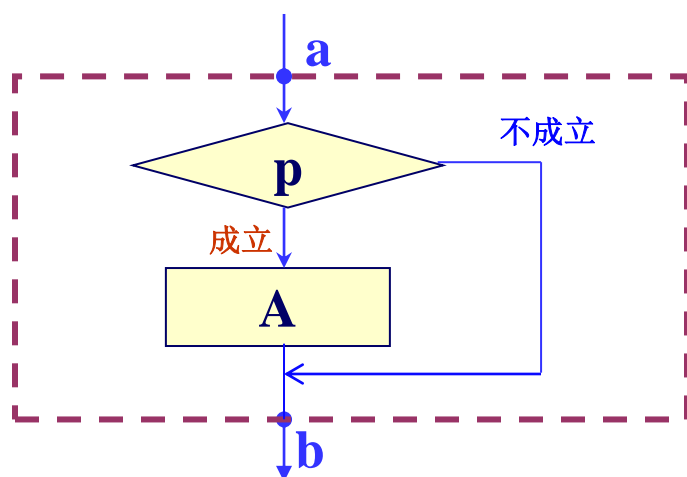
当 $\Delta = b^2 - 4ac \geq 0$ 时, 有实数解.

判别式的不同取值 (如 $\Delta \geq 0$) 决定不同求解策略

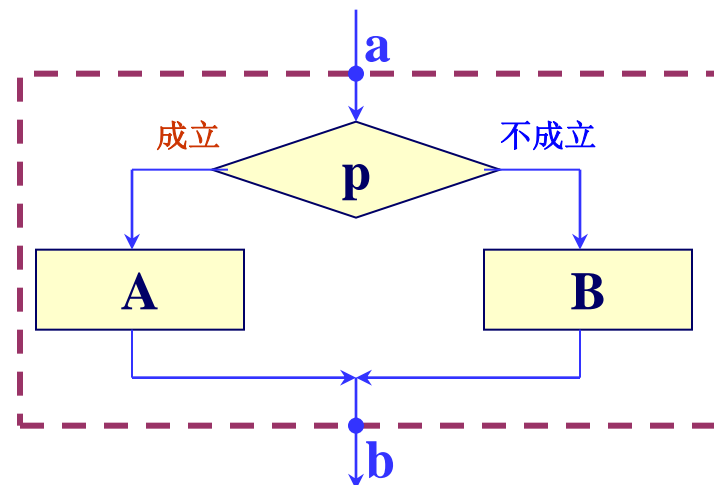
如何判断条件

- 需要检验 $\Delta = b^2 - 4ac$ 是否在实数范围内可以开平方根（条件判断）？
- 条件判断的基本方法：真假值检测
- C/C++语言真假表示：**0** -> 假；**非0** -> 真
- 产生真假值运算
 - 关系运算
 - 逻辑运算
 - 一般值可以作为真假判断：**0 / 非0**

简单的分支：二分支



选择结构1：跳过



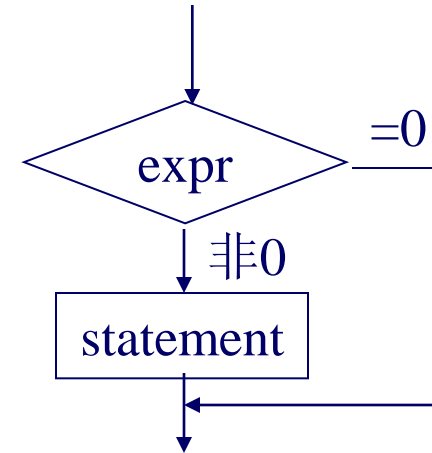
选择结构2

- 二分支结构就是根据条件值(即产生真/假值的表达式)，形成两条路径

if 语句形式之一

— 语句形式一：

» 格式：if (expression)
statement



- 思考：用max表示整数a和b中的较大者

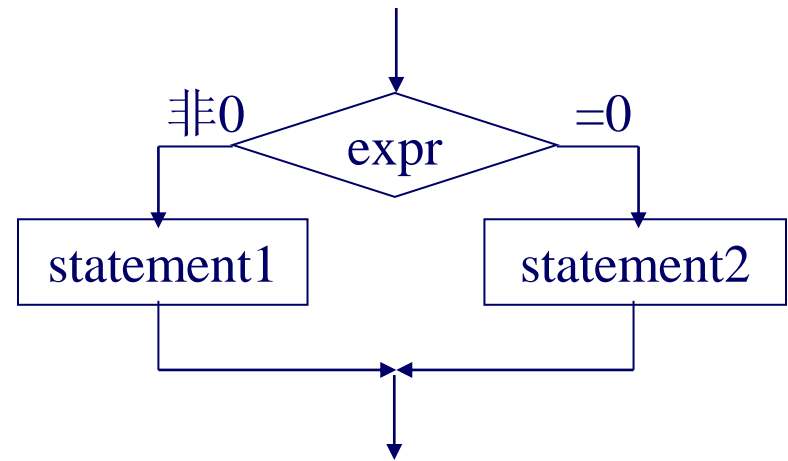


```
max = b;  
if (a > b)  
    max = a;
```


if 语句形式之二

— 语句形式二：

» 格式：if (expression)
 statement-1
else
 statement-2



● 思考：用max表示整数a和b中的较大者



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

比较

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

一元二次方程求解

- **问题**：求一元二次方程为 $ax^2+bx+c=0$ 的

- **已知**： $\Delta = b^2 - 4ac$

- **令**： $p = \frac{-b}{2a}$ $q = \frac{\sqrt{|\Delta|}}{2a}$

- **如果** , $\Delta \geq 0$ $x_1 = p + q$ $x_2 = p - q$

- **否则**： $x_1 = p + qi$ $x_2 = p - qi$

```

#include <iostream>

#include <cmath> //或#include <math.h>, 但不提倡
using namespace std;

int main()
{
    float a, b, c, disc, x1, x2, p, q;    //声明变量
    cin>>a>>b>>c; //输入系数和常量
    disc = b * b - 4 * a * c;
    p = -b / (2 * a); q = f表示实型sqrt(fabs(disc)) / (2 * a);
    if (disc<0)
        cout<< p<<'+'<<q<<"i, " << p<<'+'<<q<<"i"<<endl;
    else
        cout<< p+q<<', ' << p-q<<endl;
    return 0;
}

```

$$p = \frac{-b}{2a}$$

$$q = \frac{\sqrt{|\Delta|}}{2a}$$

再看 if 语句

◆ if 后面的表达式类型任意 (数值: 非0 则真!)

$\text{if}(x) \Leftrightarrow \text{if}(x \neq 0)$

$\text{if}(!x) \Leftrightarrow \text{if}(x = 0)$

◆ 语句可以是复合语句

分析如下程序, 指出错误, 说明程序的功能

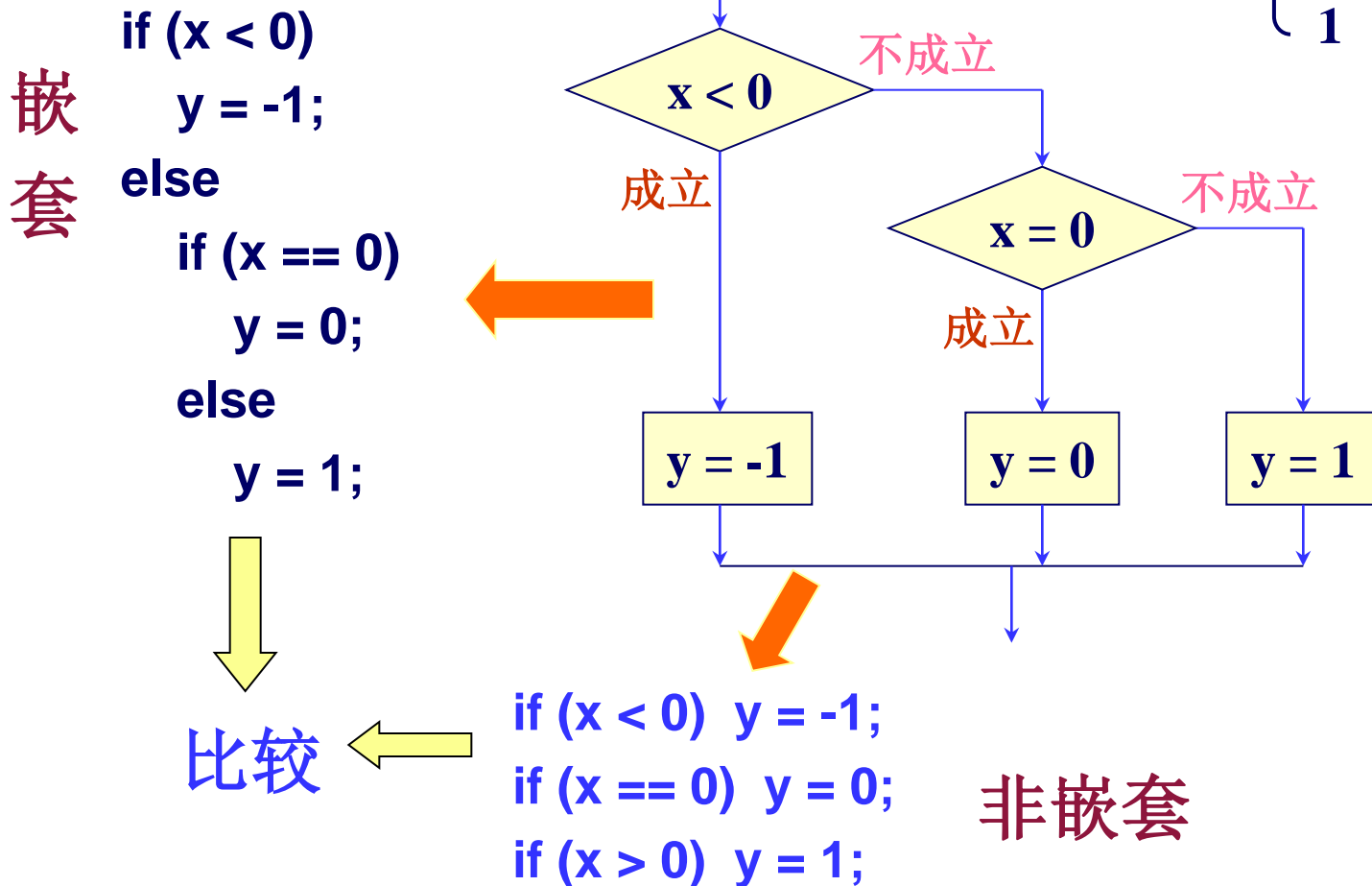
```
#include <iostream>
using namespace std;
int main()
{   int x, y, temp;
    cin >> x >> y;
    if(x > y)
    {   temp = y; y = x; x = temp; }
    else
    {   x++; y++; }
    cout << x << y << endl;
    return 0;
}
```



作为整体!
(多于1条语句)

if语句及嵌套

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



if语句嵌套的一般形式

| | | | |
|------------|--------|------------|--------|
| if (expr1) | | if (expr1) | |
| if (expr2) | | if (expr2) | } 内嵌if |
| statement1 | | statement1 | |
| else | | else | |
| statement2 | } 内嵌if | statement3 | |

如何判断 **else** 的配对??

| | | | |
|------------|--------|-----------------------|--------|
| if (expr1) | | if (expr1) | |
| statement1 | | if (expr2) statement1 | } 内嵌if |
| else | | else statement2 | |
| if(expr3) | | else | |
| statement3 | } 内嵌if | if(expr3) statement3 | } 内嵌if |
| else | | statement4 | |
| statement4 | | else statement4 | |

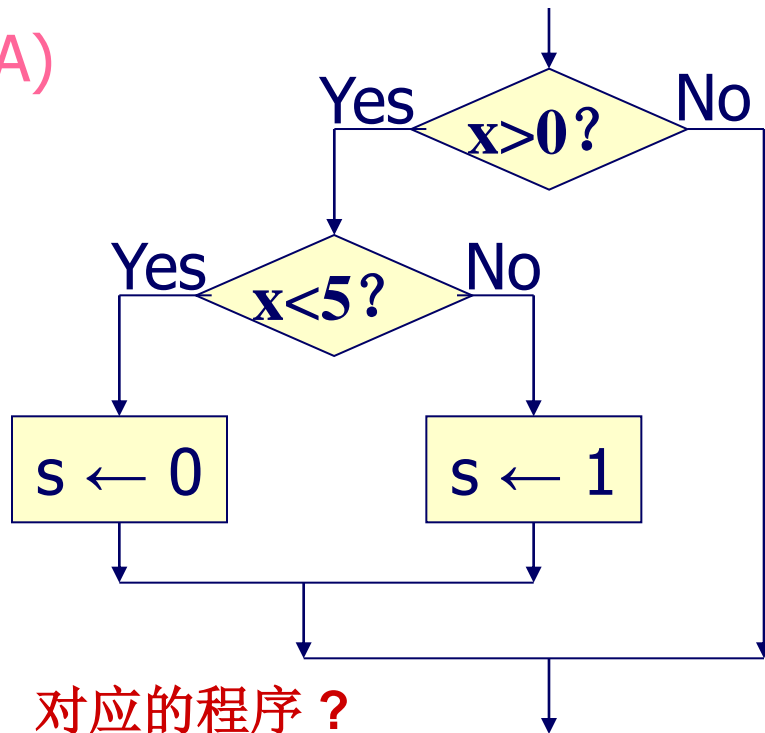
if ~ else 配对原则：缺省{ }时，else总是和它上面最近的未配对的 if 配对

```
    if(.....)
    {
        if(.....)
        {
            if(.....)
            {
                else.....
            }
            else.....
        }
        else.....
    }
```

书写程序时，养成匹配对齐，标识配对的良好习惯

else 的配对

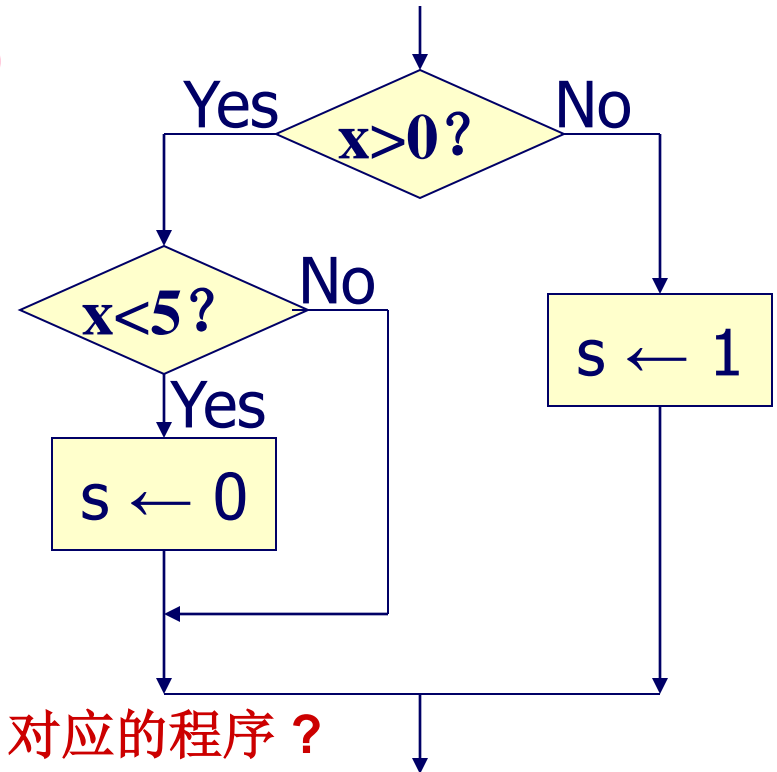
(A)



对应的程序 ?

```
if (x > 0) {  
    if (x < 5)  
        s = 0;  
    else  
        s = 1;  
}
```

(B)



对应的程序 ?

```
if (x > 0)  
{ if (x < 5)  
    s = 0; }  
else  
    s = 1;
```

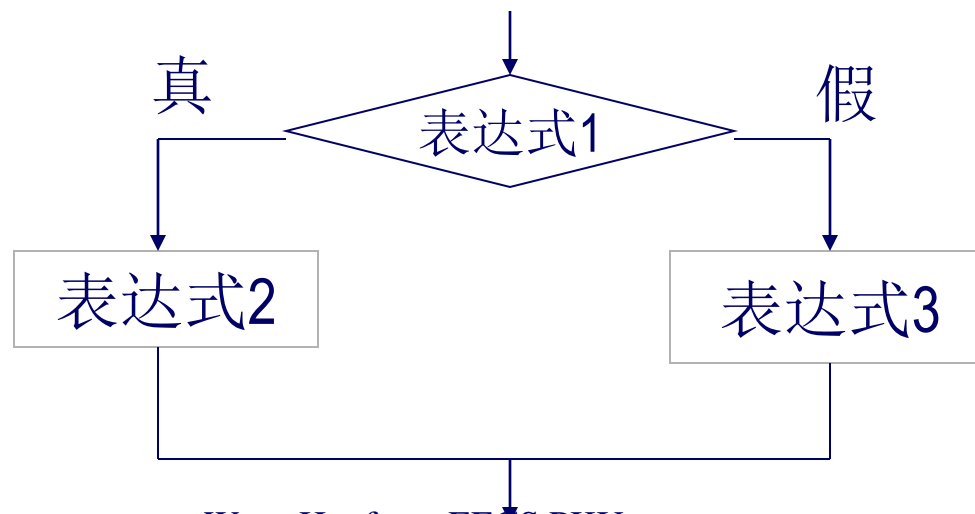
<-等价->

分析下面程序输出结果:

```
#include <iostream>
using namespace std;
int main()
{   int x=100,a=10,b=20;
    int v1=5,v2=0;
    if(a<b)
        if(b!=15)
            if(!v1)
                x=1;
            else
                if(v2) x=10;
    x=x-1;
    cout<<x<<endl;
    return 0;
}
```

条件表达式（条件运算符）

- 条件运算符是**三目**运算符；
- 格式为：
表达式1? 表达式2: 表达式3
例子： $m = (a > b ? a : b)$
问：该例子的功能？
- 图示：



例子

- 一个例子:

```
if(a>b) cout<<a<<endl;
```

```
else cout<<b<<endl;
```

如何取代为条件表达式?

```
cout<< (a>b ? a:b)<<endl;
```

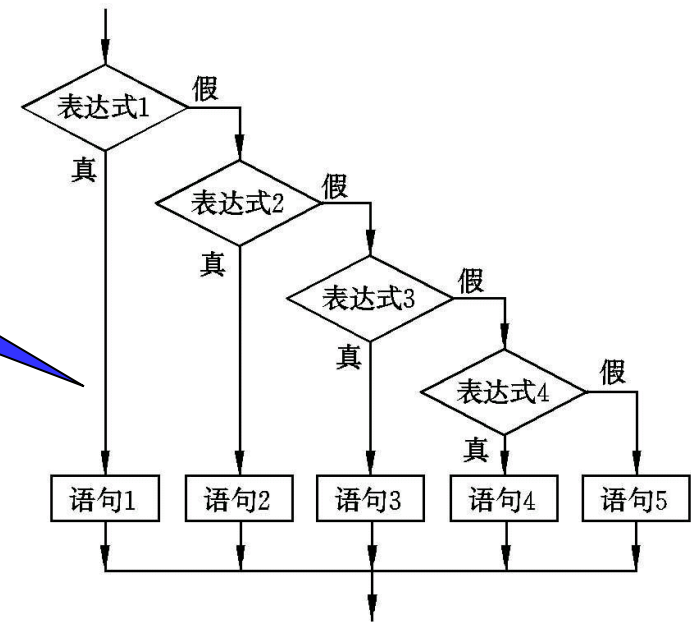
复杂嵌套与if 语句实现

问题——成绩转换：输入一个百分制的成绩(整数)，将其转换为等级分制输出

● 规则如下：

- 90 ~ 100 : A
- 80 ~ 89 : B
- 70 ~ 79 : C
- 60 ~ 69 : D
- 低于60 : F

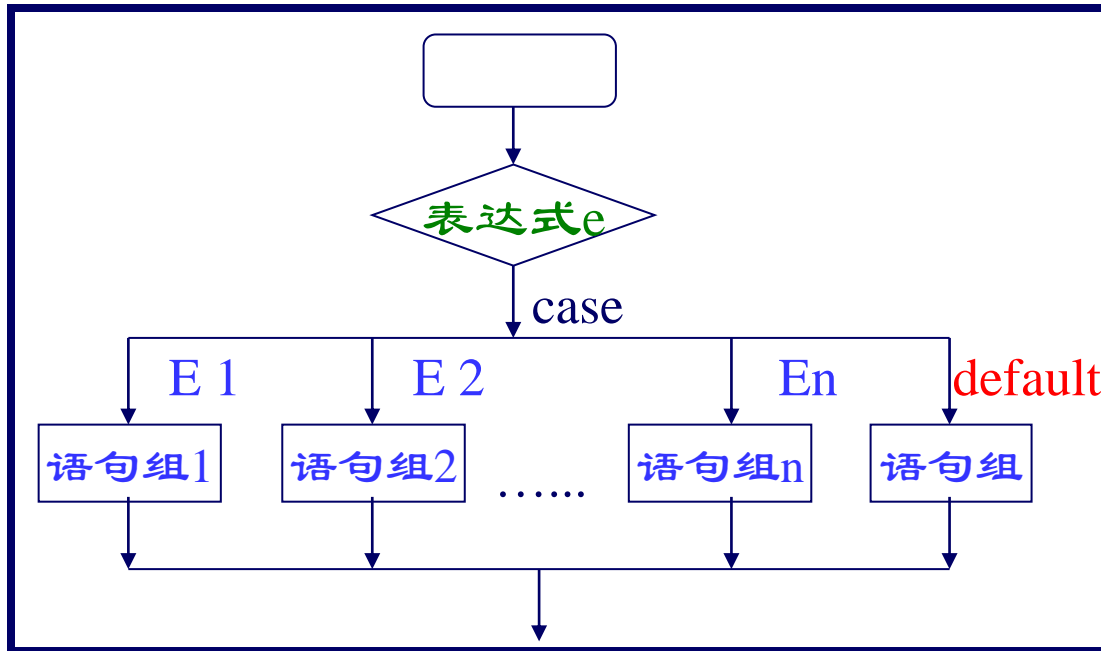
嵌套层次太多



```
if(n>=90) cout<<" grade = A "<<endl;
else if(n>=80) cout<<" grade = B "<<endl;
    else if(n>=70) cout<<" grade = C "<<endl;
        else if(n>=60) cout<<" grade = D "<<endl;
            else cout<<" grade = F "<<endl;
```

多分支

❖一般形式：



```
switch( e)
{   case   E1:
      语句组 1;
```

```
    case   E2:
      语句组 2;
```

.....

```
    case   En:
      语句组 n;
```

```
    [default:
      语句组 ;
      break;]
```

```
}
```

❖说明：

- E1, E2, ...En是**整型或字符型常量表达式**, 且值必须互不相同, 表示执行的条件
- case后可包含多个可执行语句, 且不必加{ }
- 多个case可共用一组执行语句

如：

```
case 'A':
```

```
case 'B':
```

```
case 'C':
```

```
    cout<<"score>60"<<endl;  
    break;
```

```
.....
```

用switch实现成绩转换

```
#include <iostream>
using namespace std;

int main()
{   int score;
    cin>>score;
    switch (score / 10)
    { case 10:
      case 9: cout<<" grade = A "<<endl; break;
      case 8: cout<<" grade = B "<<endl; break;
      case 7: cout<<" grade = C "<<endl; break;
      case 6: cout<<" grade = D "<<endl; break;
      default: cout<<" grade = F "<<endl;
    }
}
```

- 规则如下：
 - 90 ~ 100 : A
 - 80 ~ 89 : B
 - 70 ~ 79 : C
 - 60 ~ 69 : D
 - 低于60 : F

没有这些break会怎样？

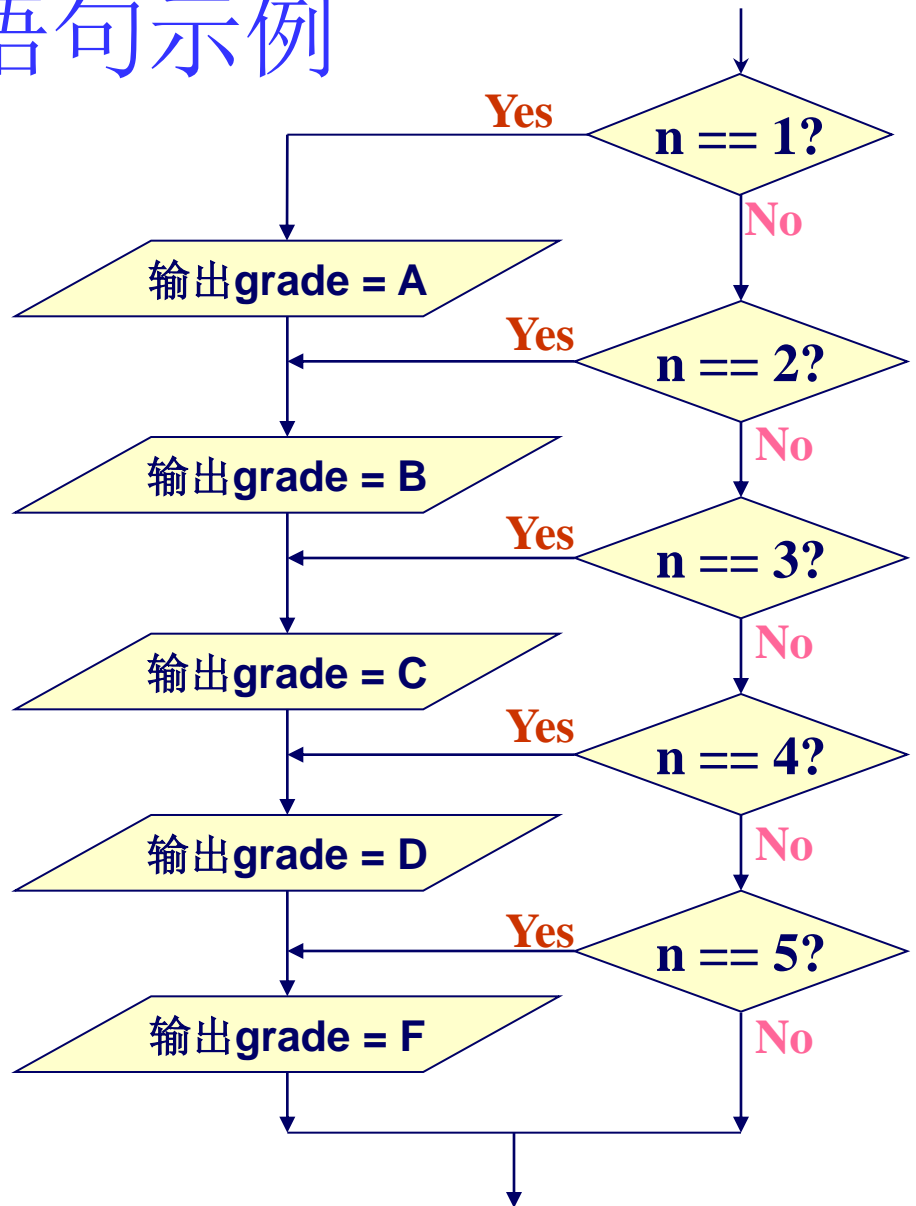
缺少break的switch语句示例

switch (n)

```
{ case 1: printf(" grade = A\n ");  
  case 2: printf(" grade = B\n ");  
  case 3: printf(" grade = C\n ");  
  case 4: printf(" grade = D\n ");  
  case 5: printf(" grade = F\n ");  
}
```

如果 $n=1$, 输出什么?

$n=3$ 呢?




```
int main()
{   int x=1,y=0,a=0,b=0;
    switch(x)
    {   case 1:
            switch(y)
            {   case 0:  a++; break;
                case 1:  b++; break;
            }
        case 2:  a++;b++; break;
        case 3:  a++;b++;
    }
    cout<<"\na="<<a<<" ,b="<<b<<endl;
    return 0;
}
```

Switch 可以嵌套

注意:

- switch 可以嵌套
- default 可有可无
- **break** 作用范围

问题:

如何使外层的
case 1: 执行后,
不执行 case 2:

运行后, 输出的结果是:

a=2,b=1

假设不用 if、switch语句呢？思考！

用顺序结构解决如下问题（如何实现？）：

输入一名学生的百分制成绩（0~100），以等级制输出,每10分为一个级别（A:100； B:90~99； C: 80~89； D:70~79； E: 60~69； F:50 ~ 59； ...），

注意：不能使用if 语句和switch

...

int score;

cin>>score;

cout<<(10-score/10)+'A';

内容

➤ 分支结构

➤ 循环（重复）结构

为什么需要循环

- 例子：计算 $1+2+3+4+5$

用S表示每次相加的结果

- 步骤如下：

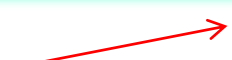
- 步骤1：先计算 $0+1$ ，得到 $S \leftarrow 0+1$ ；
- 步骤2：将步骤1得到的结果加上2，得到 $S \leftarrow 1+2=3$
- 步骤3：将步骤2得到的结果加上3，得到 $S \leftarrow 3+3=6$
- 步骤4：将步骤3得到的结果加上4，得到 $S \leftarrow 6+4=10$
- 步骤5：将步骤4得到的结果加上5，得到 $S \leftarrow 10+5=15$ ，

最终计算结果为15

按照同样的思路可写出计算 $1+2+\dots N (N > 100)$ 的步骤——通用性

怎样重复

步骤1: $S \leftarrow 0; I \leftarrow 1;$
步骤2: 若 I 大于 N , 则转向步骤6; 否则, 处理步骤3;
步骤3: $S \leftarrow S + I;$
步骤4: $I \leftarrow I + 1;$
步骤5: 转向步骤2; (改变方向—重复)
步骤6: S 的值就是计算结果, 运算结束。



- C/C++ 语言实现重复的两种方法
 - 用 **goto** 和 **if** 构成循环
 - 结构化循环
 - while 语句
 - do ~ while 语句
 - for 语句

goto 循环

- goto构成的循环格式:

```
goto 语句标号;  
.....  
标号: 语句;
```

❖关于标号的说明:

- 不能用整数作标号, 标号的命名规则同标识符一样
- 当一个语句前加上一个标号, 并带上冒号, 该语句便称为**带标号的语句**, 可以作为 goto 语句转移的目标
- 标号出现在goto所在函数内, 且唯一**

例子：求 $1+2+3+\dots+100$

```
#include <iostream>
using namespace std;
int main()
{   int i,sum=0;
    i=1;
loop: if(i<=100)
    {   sum=sum+i;
        i++;
        goto loop;
    }
    cout<<sum<<endl;
    return 0;
}
```

循环条件

goto 循环需要与 if 结合

循环初值

循环变量增值

循环终值

循环体

例： 从键盘输入一组整数，以0结束输入，求数据和

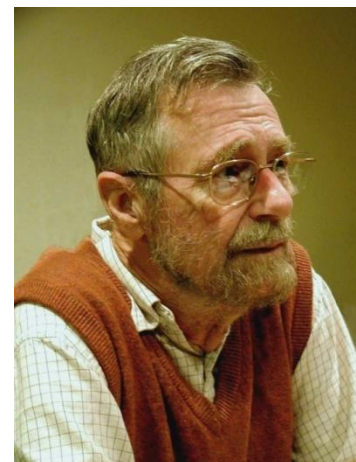
```
#include <iostream>
using namespace std;
int main()
{
    int number,sum=0;
    read_loop: cin>>number;
    if(!number) goto print_sum;
    sum+=number;
    goto read_loop;
    print_sum: cout<<"The total sum is "<<sum<<endl;
    return 0;
}
```

goto 循环需要与 if 结合

goto 很复杂， 限制使用！！

关于goto语句的讨论

- 著名的荷兰教授 **E. W. Dijkstra**
 - 1965年，IFIP（International Federation for Information Processing）会议上，Dijkstra提出“goto语句可以从高级语言中取消”，“一个程序的质量与程序中所含的 goto语句的数量成反比”。
 - 但是，Dijkstra讲话的影响很小，当时人们正广泛地使用FORTRAN，而 goto语句则是FORTRAN的支柱。



Algol60设计实现者之一
goto有害论提出者
信号量理论提出者
最短路径算法提出者
THE操作系统设计者
程序正确性证明推动者

关于goto语句的讨论

- 60年代末至70年代，关于goto语句的争论非常激烈
- 正方：从高级语言中去掉goto语句
 - 包含goto语句的程序难以阅读，难以查错
 - 去掉goto语句后，可以直接从程序结构上反映程序的运行过程。使程序的结构清晰、便于阅读，便于查错，而且也有利于程序正确性证明
- 反方：goto语句无害，应该保留
 - goto语句使用起来比较灵活，而且有些情形能够提高程序的效率
 - 如果一味强调删除goto语句，有些情形反而会使程序过于复杂，增加一些不必要的计算量

关于goto语句的讨论

- Donald E. Knuth
- 1974年，D.E.Knuth对于goto语句的争论作了全面的公正的评述：
 - 不加限制地使用goto语句，特别是使用来回跳的goto语句，会使程序的结构难于理解，这种情形应该尽量避免使用goto语句
 - 为了提高程序的效率，同时又不破坏程序的良好结构，有控制地使用一些goto语句是有必要的
- “有些情形，主张废除转向语句，有些情形我主张引进转向语句。”



尽量避免使用goto语句

- 结构化的程序设计技术：限制goto语句的使用，因为滥用goto语句，将会导致程序结构混乱、可读性差。
- 现代高级程序设计语言都会提供专门表达循环的**结构化语句**

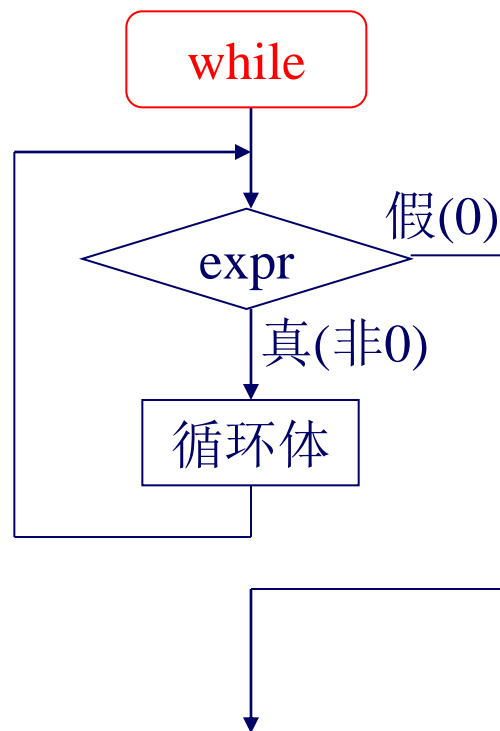
结构化循环while 语句

★while语句

❖一般形式:

```
while(表达式)  
    循环体语句;
```

❖执行流程:



例子： $1+2+\dots+100$

```
#include <iostream>
using namespace std;
int main()
{ int i,sum=0;
  i=1;
  while(i<=100)
  { sum=sum+i;
    i++;
  }
  cout<<sum<<endl;
  return 0;
}
```

循环初值

循环条件

循环变量增值

循环终值

循环体

另一个例子

小红10岁,父亲33岁,问多少年之后,父亲的年龄是小红的二倍?

```
#include<iostream>
using namespace std;
int main()
{
    int ageOfHong = 10, ageOfFather = 33, count = 0;
    while (2 * ageOfHong != ageOfFather)
    {
        ageOfHong++;
        ageOfFather++;
        count++;
    }
    cout << count<<endl;
    return 0;
}
```

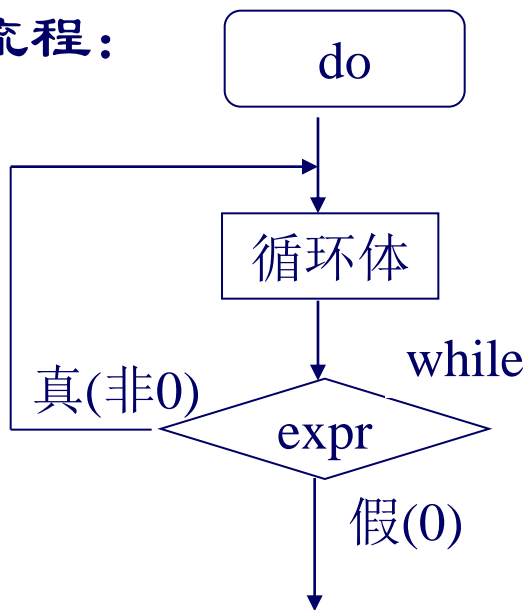

结构化循环 — do~while 循环

★ do~while语句

❖ 一般形式：

```
do  
    循环体语句;  
while(表达式);
```

❖ 执行流程：



while和do~while比较

```
#include <iostream>
using namespace std;
int main()
{  int i,sum=0;
  cin>>i;
  do
  {  sum+=i;
    i++;
  }while(i<=10);
  cout<<sum<<endl;
  return 0;
}
```

先执行循环体，再判断

等价吗？

```
#include <iostream>
using namespace std;
int main()
{  int i,sum=0;
  cin>>i;
  while(i<=10)
  {  sum+=i;
    i++;
  }
  cout<<sum<<endl;
  return 0;
}
```

先判断，再执行循环体

如何计算无符号整数的逆序？

```
#include<iostream>
using namespace std;
int main()
{
    unsigned int num;
    int count = 0;
    cout << "Please enter an integer." << endl;
    cin >> num;
    do {
        cout << num % 10;
        num = num / 10;
        count++;
    } while (num != 0);
    cout << count << "digits" << endl;
    return 0;
}
```

变量的含义？

```
Please enter an integer.
12345
543215digits
```

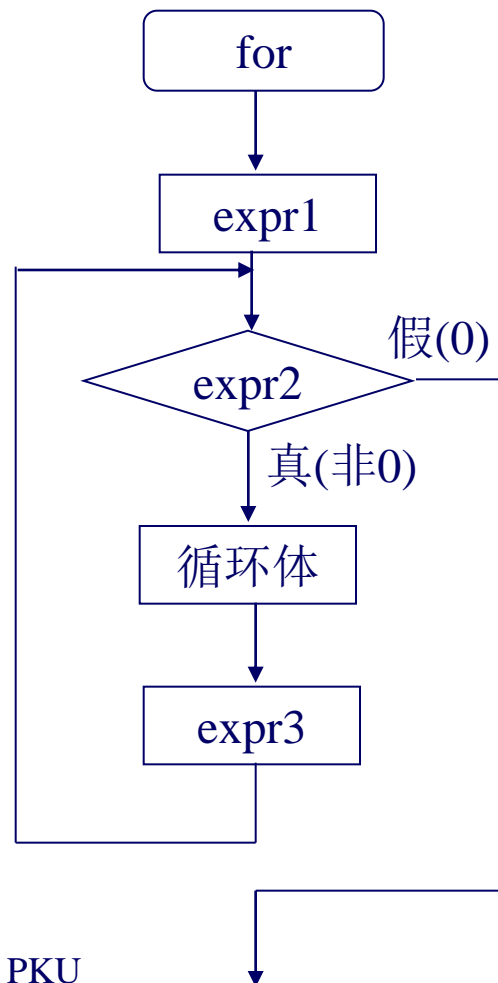
计数循环 — for 循环

★for语句

❖一般形式：

```
for( expr1 ; expr2 ; expr3 )  
    循环体语句；
```

❖执行流程：



for语句的解释

```
for(循环变量初值化; 循环终止条件; 循环变量增值)
{
    循环体语句;
}
```

for 转化为 while

```
#include <iostream>
using namespace std;
int main()
{
    int i,sum=0;
    for( i=1; i<=100; i++)
    {
        sum+=1;
        cout<<sum<<endl;
        return 0;
    }
}
```

初始

循环体

终止

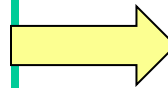
增量

```
expr1;
while(expr2)
{
    循环体;
    expr3;
}
```

for 循环的进一步说明

- for语句中expr1, expr2, expr3都可省略，但分号；不可省

```
例：#include<iostream>
using namespace std;
int main( )
{   int i=0;
    for(i=0;i<10;i++)
        cout<<(char) i+'a';
    return 0;
}
```



```
例：#include<iostream>
using namespace std;
int main( )
{   int i=0;
    for(;i<10;i++)
        cout<<(char) i+'a';
    return 0;
}
```

运行结果 abcdefghij

几种循环语句的比较

- `for` 语句主要用于计数循环；
- `While / do-while` 主要用于条件循环（计数并不明确），但两者的区别在于条件检测的时机。
- `goto` 语句尽可能少使用。

多重循环

- ❖ 三种循环可互相嵌套, 层数不限
- ❖ 外层循环可包含多个平行内循环, 但不能相互交叉
- ❖ 嵌套循环的执行流程

```
while()
{
    .....
    while()
    {
        .....
    }
    .....
}
```

```
do
{
    .....
    do
    {
        .....
    }while( );
    .....
}while( );
```

```
while()
{
    .....
    do
    {
        .....
    }while( );
    .....
}
```

```
for( ; ; )
{
    .....
    do
    {
        .....
    }while( );
    .....
    while()
    {
        .....
    }
    .....
}
```

外循环

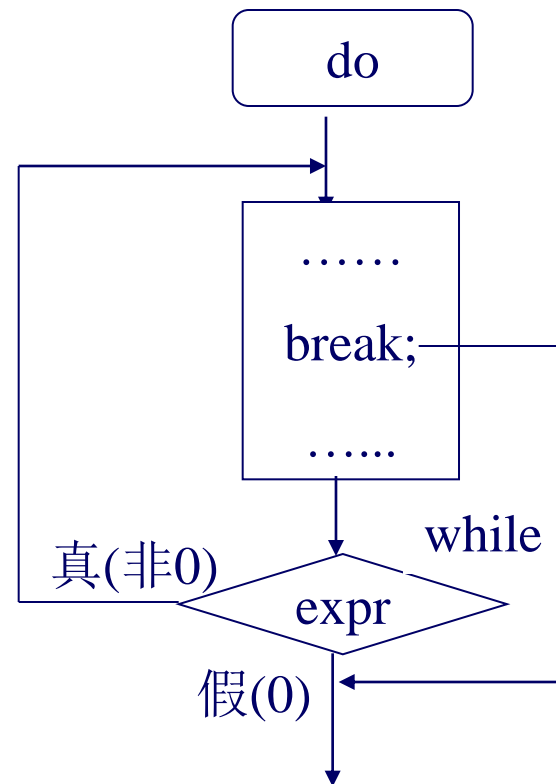
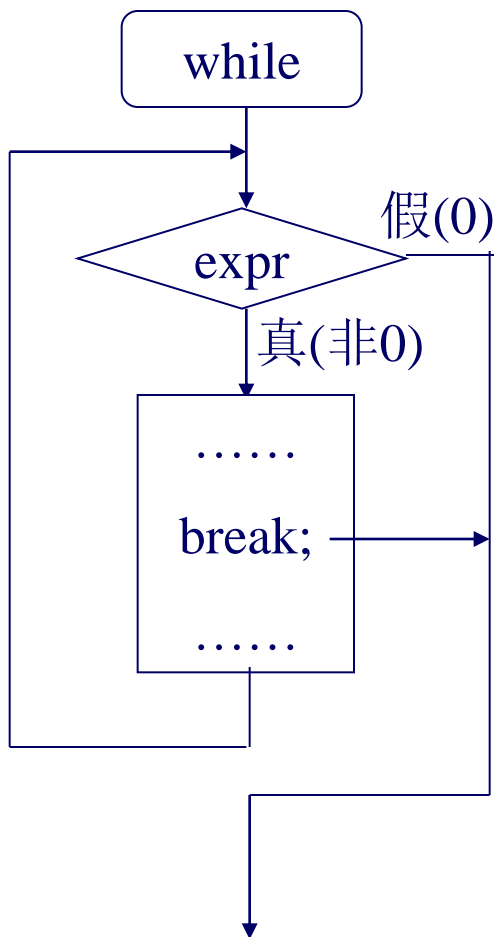
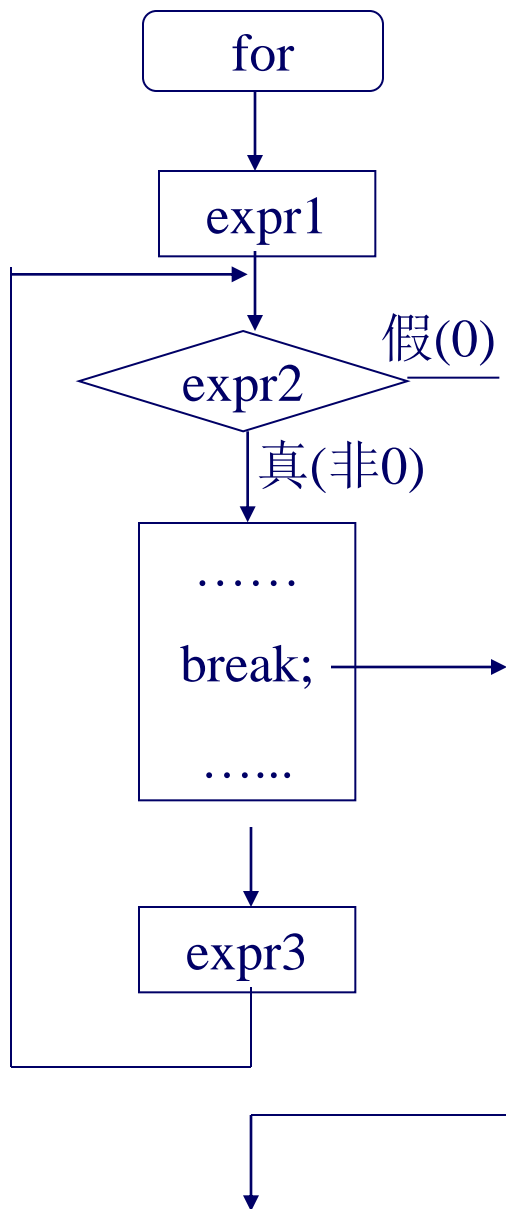
内循环

内循环

循环中的 break 控制

- 功能：在**循环语句**和**switch语句**中,终止并跳出循环体或开关体
- 说明：
 - break只能终止并跳出**最近一层**的结构
 - break不能用于循环语句和switch语句之外的任何其它语句之中

跳出所在的最近循环！



分析程序

程序的功能？

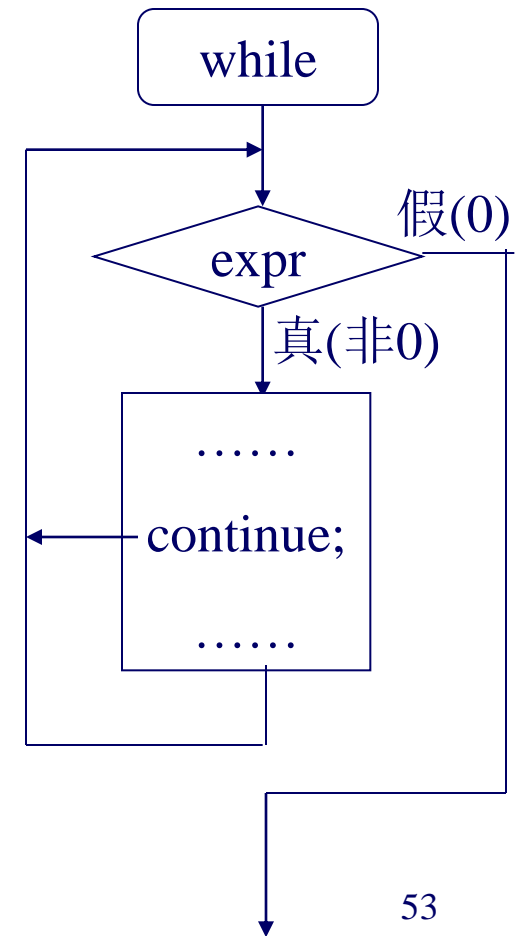
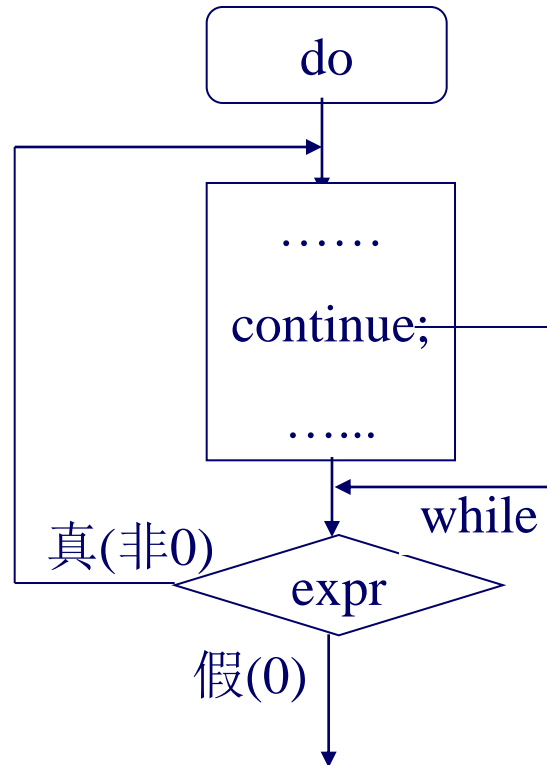
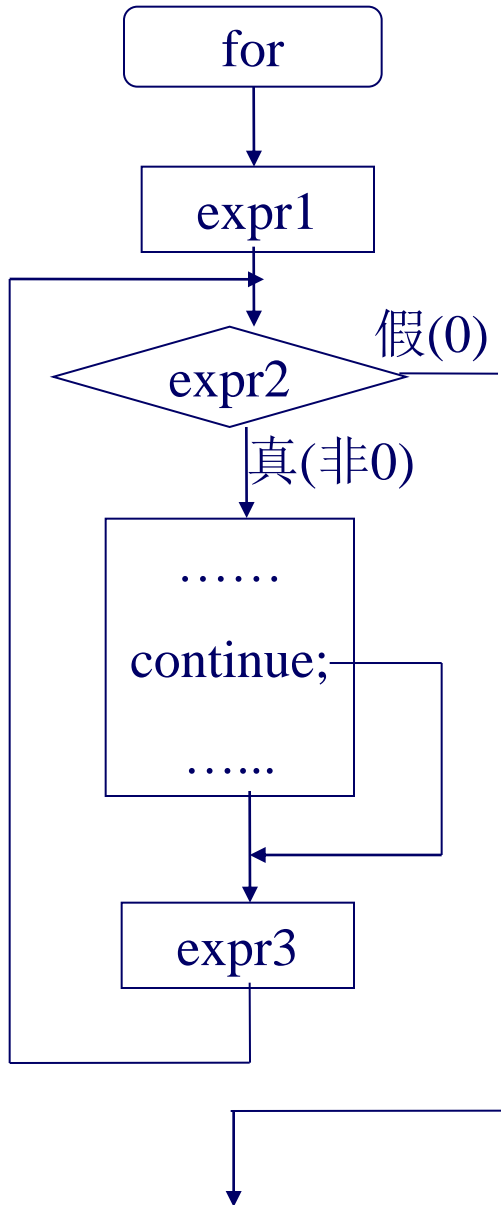
```
#include <iostream>
using namespace std;
int main()
{
    int i,j;
    char c;
    while(1)
    {    cin>>c;
        if(c>='a' && c<='z')
            cout<<c-'a'+'A'<<endl;
        else
            break;
    }
    return 0;
}
```

continue 语句

- ❖ 功能：结束本次循环，跳过所在循环中尚未执行的语句，进入同一层循环的下一轮（检测条件）
- ❖ 主要用于循环语句中

continue 语句

❖ 功能：结束本次循环，跳过尚未执行的语句，进入下一轮（检测条件）



例：输入**10**个整数，计算正数个数及平均值

```
#include <iostream>
using namespace std;
int main()
{   int i,num=0,a;
    float sum=0;
    for(i=0;i<10;i++)
    {   cin>>a;
        if(a<=0) continue;
        num++;
        sum+=a;
    }
    cout<<num<<"plus integer's sum :"<<sum)<<endl;
    cout<<"Mean value:"<<sum/num<<endl;
}
```