

第2篇 面向过程的程序设计

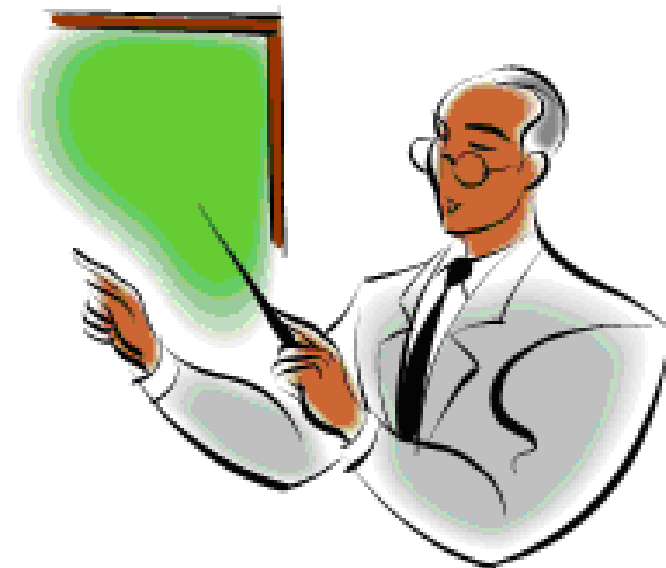
第3章 程序设计初步

第4章 函数与预处理

第5章 数组

第6章 指针

第7章 自定义数据类型



第3章 程序设计初步



3.1 面向过程的程序设计和算法

3.2 C++程序和语句

3.3 赋值语句

3.4 C++的输入与输出

3.5 编写顺序结构的程序

3.6 关系运算和逻辑运算

3.7 选择结构和if语句

3.8 条件运算符和条件表达式

3.9 多分支选择结构和switch语句

3.10 编写选择结构的程序

3.11 循环结构和循环语句

3.12 循环的嵌套

3.13 break语句和continue语句

3.14 编写循环结构的程序

3.1 面向过程的程序设计和算法

3.1.1 算法的概念

面向过程的程序：

- 对数据的描述。在程序中要指定数据的类型和数据的组织形式，即数据结构。
- 对操作的描述。即操作步骤，也就是算法。

程序=算法+数据结构

- 算法:为解决一个问题而采取的方法和步骤。

3.1.2 算法的表示

- 自然语言:

- 流程图:传统的流程图或结构化流程图。用图的形式表示算法,比较形象直观,但修改算法时显得不大方便

- 伪代码:介于自然语言和计算机语言之间的文字和符号来描述算法。

- 用计算机语言表示算法:用一种计算机语言去描述算法,即计算机程序。

3.2 C++程序和语句

```
#include <iostream>    //预处理命令
using namespace std;    //在函数之外的声明部分
int a=3;                //在函数之外的声明部分
int main( )             //函数首部
{   float b;            //函数内的声明部分
    b=4.5;              //执行语句
    cout<<a<<b;         //执行语句
    return 0;           //执行语句
}
```

- 程序应该包括数据描述（由声明语句来实现）和数据操作（由执行语句来实现）
- C++程序中最小的独立单位是语句(statement)。语句一般是用分号结束的(复合语句是以右花括号结束的)。
- C++语句可以分为以下4种：
 - ◆ **声明语句**：对变量(以及其他对象)的定义被认为是一条语句，并且可以出现在函数中的任何行。
 - ◆ **执行语句**：通知计算机完成一定的操作。
 - ◆ **空语句**： ;
 - ◆ **复合语句**：用 { }把一些语句括起来称为复合语句

◆ **执行语句**：通知计算机完成一定的操作。包括：

✓ **控制语句**：完成一定的控制功能。

- | | | |
|---|--------------|-------------------|
| ① | if()~else~ | (条件语句) |
| ② | for()~ | (循环语句) |
| ③ | while()~ | (循环语句) |
| ④ | do~while () | (循环语句) |
| ⑤ | continue | (结束本次循环语句) |
| ⑥ | break | (中止执行switch或循环语句) |
| ⑦ | switch | (多分支选择语句) |
| ⑧ | goto | (转向语句) |
| ⑨ | return | (从函数返回语句) |

◆ 执行语句：通知计算机完成一定的操作。包括：

- ✓ 控制语句：完成一定的控制功能。
- ✓ 函数和流对象调用语句。

例如：

```
sort(x,y,z);      //sort函数调用  
cout<<x<<endl;  //流对象调用语句
```

- ✓ 表达式语句。由一个表达式加一个分号构成一个语句。

```
i=i+1  //是赋值表达式  
i=i+1; //是赋值语句
```


◆空语句: ;

下面是一个空语句:

;
即只有一个分号的语句, 它什么也不做。

◆复合语句: 用 {}把一些语句括起来称为复合语句

如下面是一个复合语句。

```
{  
    z=x+y;  
    if(z>100) z=z-100;  
    cout<<z;  
}
```

3.3 赋值语句

赋值语句是由赋值表达式加上一个分号构成。

- C++中的赋值号“=”是一个运算符，可以写成：`a=b=c=d;`
- 赋值表达式可以包括在其他表达式之中。

例如：`if ((a=b)>0) cout<<"a>0 "<<endl;`

3.4 C++的输入与输出

■有关流对象cin、cout和流运算符的定义等信息是存放在C++的输入输出流库中的，因此如果在程序中使用cin、cout和流运算符，就必须使用预处理命令把头文件stream包含到本文件中：

`#include <iostream>`

■把由cin和流提取运算符“>>”实现输入的语句称为**输入语句**或**cin语句**；

■把由cout和流插入运算符“<<”实现输出的语句称为**输出语句**或**cout语句**。

3.4.1 输入流与输出流的基本操作

cout语句的一般格式为：

cout<<表达式1<<表达式2<<.....<<表达式n;

cin语句的一般格式为：

cin>>变量1>>变量2>>.....>>变量n;

例：

```
cin>>a>>b>>c;
```

```
cin>>a;
```

```
cin>>b;
```

```
cin>>c;
```

例：cout<<"aabbcc"
<<endl;

```
cout<<"aa";
```

```
cout <<"bbcc";
```

```
cout <<endl;
```

3.4.1 输入流与输出流的基本操作

注意：

- 不能用一个插入运算符“<<”插入多个输出项。
- 系统会自动判别输出数据的类型。
- 不能用cin语句把空格字符和回车换行符作为字符输入给字符变量，可用getchar函数。

例：

```
cout<<a,b,c;
```

✗

```
cout<<a+b+c;
```

✓

```
cout<<a<<' '<<b<<endl;
```

例：

```
char c1,c2;int a;float b;
```

```
cin>>c1>>c2>>a>>b;
```

```
2 5 88 123.456
```

```
c1='2',c2='5',a=88,c= 123.456
```

空格做分隔符！

3.4.2 在输入流与输出流中使用控制符

C++提供了在输入输出流中使用的控制符(有的书中称为操纵符), 见书中表3.1。

注意：如果使用了控制符，在程序单位的开头除了要加`iostream`头文件外，还要加`iomanip`头文件。

举例：输出双精度数。

`double a=123.456789012345;`

(1) `cout<<a;` 输出: 123.457

(2) `cout<<setprecision(9)<<a;` 输出: 123.456789

(3) `cout<<setprecision(6);` 恢复默认格式(精度为6)

(4) `cout<< setiosflags(ios :: fixed);` 输出: 123.456789

(5) `cout<<setiosflags(ios :: fixed)<<setprecision(8)<<a;`

输出: 123.45678901

(6) `cout<<setiosflags(ios :: scientific)<<a;` 输出:

1.234568e+02

(7) `cout<<setiosflags(ios :: scientific)<<setprecision(4)<<a;`

输出: 1.2346e02

下面是整数输出的例子：

```
int b=123456; //对b赋初值
```

(1) `cout<<b;` 输出： 123456

(2) `cout<<hex<<b;` 输出： 1e240

(3) `cout<<setiosflags ios::uppercase)<<b;` 输出： 1E240

(4) `cout<<setw(10)<<b<<' '<<b;` 输出： 123456,123456

(5) `cout<<setfill('*')<<setw(10)<<b;` 输出： **** 123456

(6) `cout<<setiosflags ios::showpos)<<b;` 输出： +123456

3.4.3 用getchar和putchar 函数进行字符的输入和输出

1. putchar函数（字符输出函数）

putchar函数的作用是向终端输出一个字符。

例如：putchar(c);

它输出字符变量 c 的值。

例3.2 输出单个字符。

```
#include <iostream>    //或者包含#include <stdio.h>
using namespace std;
int main( )
{char a,b,c;
  a='B';b='O';c='Y';
  putchar(a);putchar(b);putchar(c);putchar('\n');
  putchar(66);putchar(79);putchar(89);putchar(10);
  return 0;
}
```

运行结果为:

BOY

BOY

2. getchar函数（字符输入函数）

此函数的作用是从终端（或系统隐含指定的输入设备）输入一个字符。

getchar函数没有参数，其一般形式为getchar（）

函数的值就是从输入设备得到的字符。

注意：getchar()只能接收一个字符。

例3.3 输入单个字符。

```
#include <iostream>
using namespace std;
int main( )
{ char c;
  c=getchar( ); putchar(c+32); putchar('\n');
  return 0;
}
```

运行结果为:

A ↙

a

3.4.4 用scanf和printf函数进行输入和输出

scanf函数一般格式：

scanf(格式控制，输入表列)；

printf函数的一般格式：

printf(格式控制，输出表列)；

例3.4 用scanf和printf函数进行输入和输出。

```
#include <iostream>
using namespace std;
int main( )
{   int a; float b; char c;
    scanf("%d %c %f",&a,&c,&b); //加地址符&
    printf("a=%d,b=%f,c=%c\n",a,b,c);
    return 0;
}
```

运行情况如下：

12 A 67.98 ✓ (本行为输入，输入的3个数据间以空格分割)
a=12,b=67.980003,c=A(本行为输出)

3.5 编写顺序结构的程序

例 求一元二次方程式 $ax^2+bx+c=0$ 的根。a,b,c的值在运行时由键盘输入，
它们的值满足 $b^2-4ac \geq 0$ 。
根据求 x_1, x_2 的算法。

运行情况如下：

4.5 8.8 2.4 ✓

$x_1 = -0.327612$

$x_2 = -1.17794$

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{ float a,b,c,x1,x2;
  cin>>a>>b>>c;
  x1=(-b+sqrt(b*b-4*a*c))/(2*a);
  x2=(-b-sqrt(b*b-4*a*c))/(2*a);
  cout<<"x1="<<x1<<endl;
  cout<<"x2="<<x2<<endl;
  return 0;
}
```

作业：

P

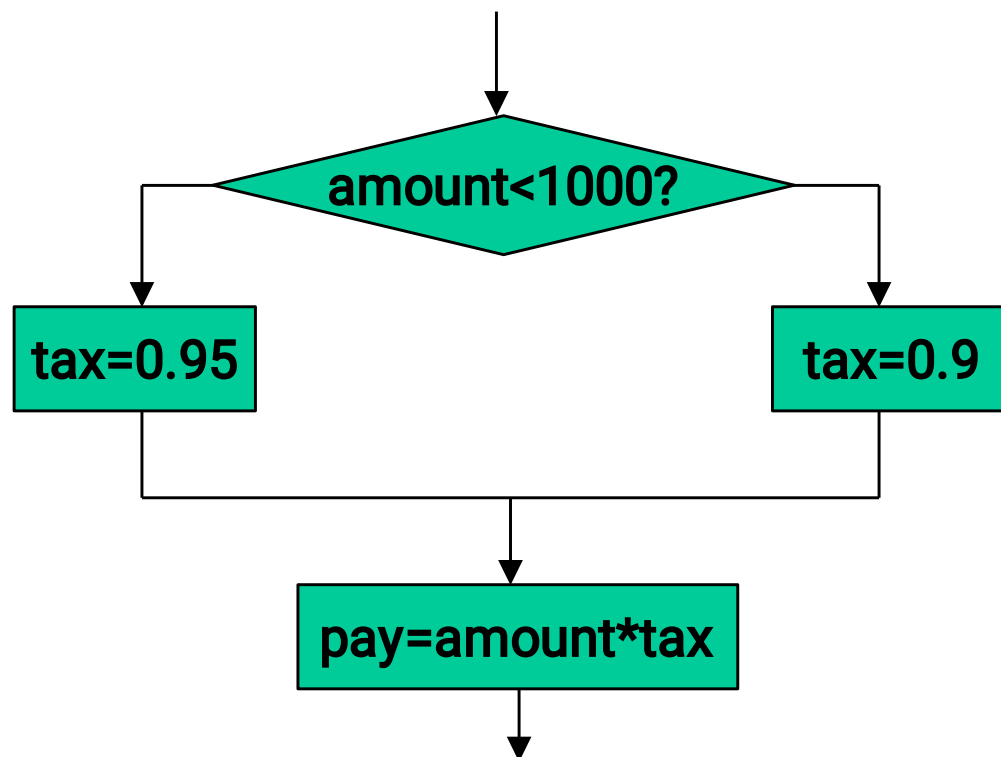
023

3.6 关系运算和逻辑运算

例如，购物在1000元以下的打九五折，1000元及以上的打九折。

C++提供if语句来实现这种条件选择。如

```
if (amount<1000)
    tax=0.95;
else
    tax=0.9;
pay=amount*tax;
```



3.6.1 关系运算和关系表达式

◆ 关系运算符

● 种类: < <= == >= > !=

● 结合方向: 自左向右

● 优先级别:

<	}	优先级7 (高)
<=		
>		
>=		
==	}	优先级8 (低)
!=		

◆ 关系表达式的值是一个逻辑值，即“真”或“假”。

例如：

c>a+b	等效于 c>(a+b)
a>b==c	等效于(a>b)==c
a==b<c	等效于a==(b<c)
a=b>c	等效于a=(b>c)

3.6.2 逻辑常量和逻辑变量

逻辑型常量只有两个，即false(假)和true(真)。

逻辑型变量要用类型标识符bool来定义，它的值只能是true和false之一。

如：bool found,flag=false;

found=true;

flag=123; //赋值后flag的值为true

cout<<flag; //输出为数值1。

3.6.3 逻辑运算和逻辑表达式

◆ 逻辑运算符：

● 种类： ! && ||

● 逻辑运算真值表

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

- 优先级:
- 结合方向:



例如: $a \ \&\& \ b$ 若 a, b 为真, 则 $a \ \&\& \ b$ 为真。
 $a \ || \ b$ 若 a, b 之一为真, 则 $a \ || \ b$ 为真。
 $!a$ 若 a 为真, 则 $!a$ 为假。

例如:

$(a > b) \ \&\& \ (x > y)$

$(a == b) \ || \ (x == y)$

$(!a) \ || \ (a > b)$

可写成 $a > b \ \&\& \ x > y$

可写成 $a == b \ || \ x == y$

可写成 $!a \ || \ a > b$

短路特性：逻辑表达式求解时，并非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

例 `a&&b&&c` //仅在a为真时，才判别b的值；
 //仅在a、b都为真时，才判别 c的值

例 `a||b||c` //仅在a为假时，才判别b的值；
 //仅在a、b都为假时，才判别 c的值

例 `a=1;b=2;c=3;d=4;m=1;n=1;`
 `(m=a>b)&&(n=c>d)` //结果m=0,n=1

例如，要判别某一年(year)是否为闰年。闰年的条件是符合下面两者之一：

- ◆能被4整除，但不能被100整除。
- ◆能被100整除，又能被400整除。

可以用一个逻辑表达式来表示：

$(\text{year \% } 4 == 0 \ \&\& \ \text{year \% } 100 != 0) \ || \ \text{year \% } 400 == 0$

可以加一个“!”用来判别非闰年：

$!((\text{year \% } 4 == 0 \ \&\& \ \text{year \% } 100 != 0) \ || \ \text{year \% } 400 == 0)$

3.7 选择结构和if语句

- if语句（条件选择语句）

- ◆ if语句的三种形式

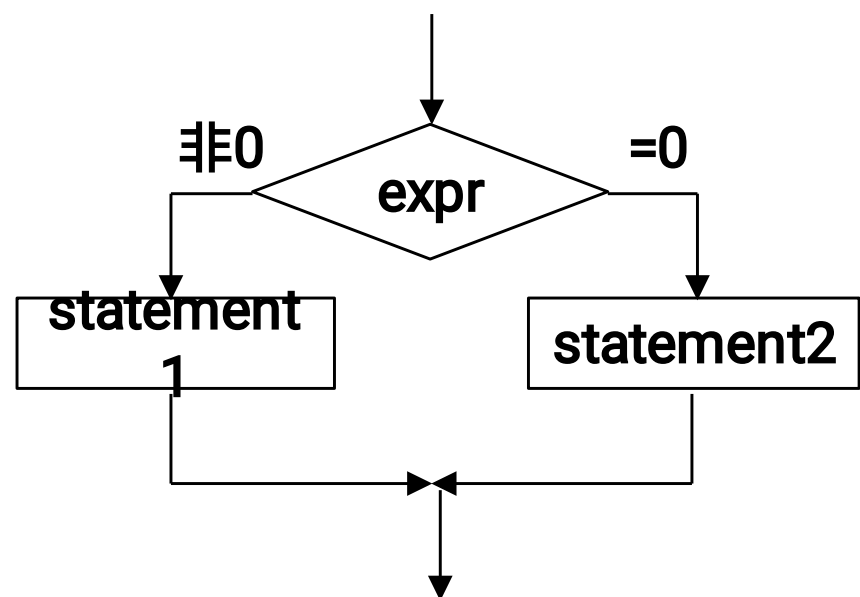
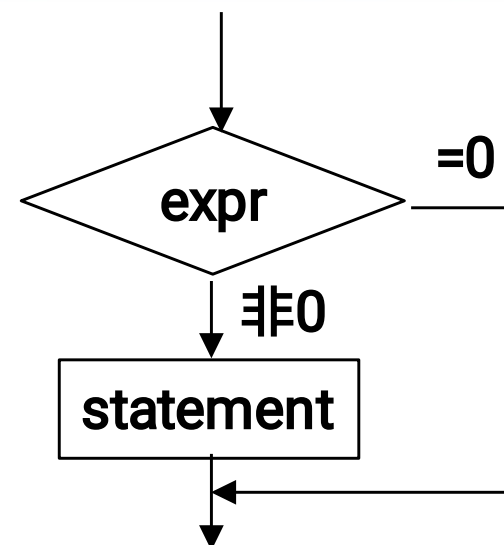
- 形式一：

- ◆ 格式：if (expression)
statement

- 形式二：

- ◆ 格式：if (expression)
statement1
else
statement2

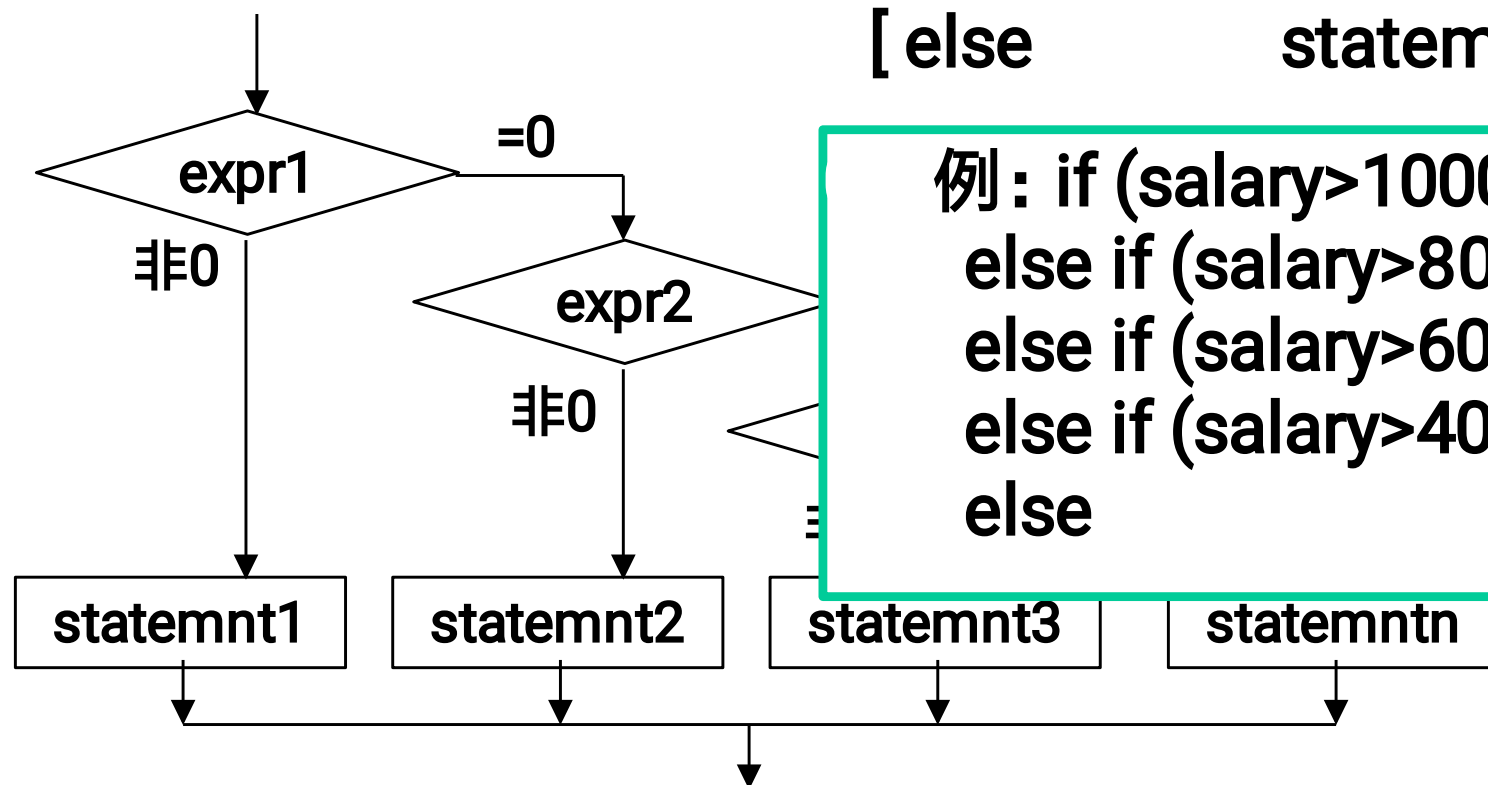
例：
if (x>y)
 max=x;
else
 max=y;



●形式三：

◆格式：

```
if ( expr1 )      statement1
else if (expr2 )  statement2
else if (expr3 )  statement3
.....
[ else           statementn ]
```



例：if (salary>1000) index=0.4;
else if (salary>800) index=0.3;
else if (salary>600) index=0.2;
else if (salary>400) index=0.1;
else index=0;

例3.6 求三角形的面积。

```
#include <iostream>
```

```
#include <cmath> //使用数学函数时要包含头文件cmath
```

```
#include <iomanip> //使用I/O流控制符要包含头文件iomanip
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    double a,b,c;
```

```
    cout<<"please enter a,b,c: ";
```

```
    cin>>a>>b>>c;
```

```
if (a+b>c && b+c>a && c+a>b)
```

```
{ double s,area;
```

```
  s=(a+b+c)/2;
```

```
  area=sqrt(s*(s-a)*(s-b)*(s-c));
```

```
  cout<<setiosflags(ios::fixed)<<setprecision(4);
```

```
  cout<<"area="<<area<<endl;
```

```
}
```

```
else
```

```
  cout<<"it is not a trilateral!"<<endl;
```

```
return 0;
```

```
}
```

有多条语句，
必须有{ }

运行情况如下：

please enter a,b,c: 2.45 3.67

4.89

area=4.3565

3.7.2 if语句嵌套

```
if (expr1)
    if (expr2)
        statement1
    else
        statement2
```

内嵌if

```
if (expr1)
    statement1
else
    if(expr3)
        statement3
    else
        statement4
```

内嵌if

```
if (expr1)
{
    if (expr2)
        statement1
}
else
    statement3
```

内嵌if

```
if (expr1)
    if (expr2)
        statement1
    else
        statement2
else
    if(expr3)
        statement3
    else
        statement4
```

内嵌if 内嵌if

例 输入两数并判断其大小关系

```
#include <iostream>
using namespace std;
int main()
{   int x,y;
    cout<<"Enter integer x,y:";
    cin>>x>>y;
    if(x!=y)
        if(x>y) cout<<"X>Y\n";
        else   cout<<"X<Y\n";
    else
        cout<<"X==Y\n";
}
```

运行:

Enter integer x,y:12,23↵

X<Y

Enter integer x,y:12,6↵

X>Y


Enter integer x,y:12,12↵

X==Y

- **if - else 配对原则：缺省{ }时，else总是和它上面离它最近的未配对的if配对**

```
if(.....)
[
    if(.....)
    [
        if(.....)
        else.....
    ]
    else.....
]
else.....
```

例：if (a==b)
 if(b==c)
 cout<<"a==b==c"<<endl;
 else
 cout<<"a!=b"<<endl;



修改：if (a==b)
 { if(b==c)
 cout<<"a==b==c"<<endl;
 }
 else
 cout<<"a!=b"<<endl;

实现if ~ else 正确配对方法：加{ }

例 考虑下面程序输出结果:

```
#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;
    else
        x=-1;
    cout<<x<<endl;
    return 0;
}
```

结果: -1

3.8 条件运算符与表达式

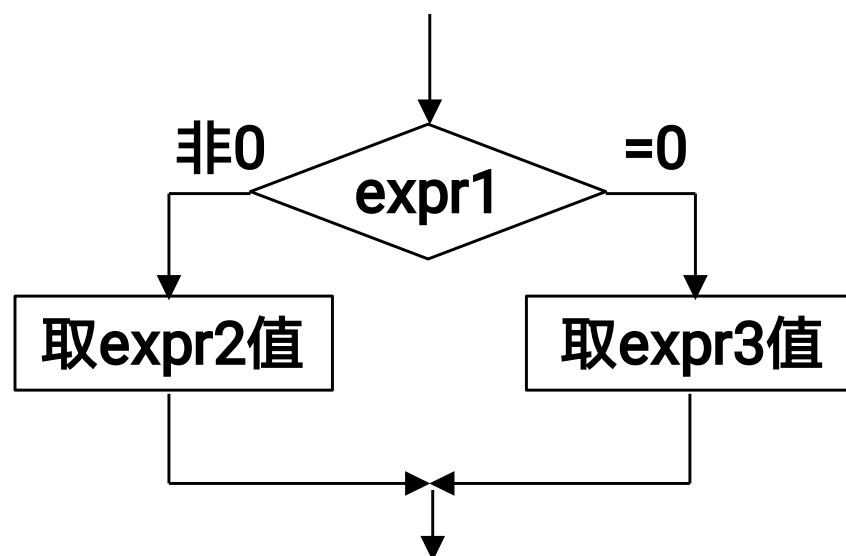
❖ 一般形式: $expr1 ? expr2 : expr3$

❖ 功能: 相当于条件语句, 但不能取代一般if语句

例 if (a>b)
 cout<<a;
else
 cout<<b;



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



例 求 $a+|b|$
cout<<"a+|b|="<<(b>0?a+b:a-b);

3.8 条件运算符与表达式

● 条件运算符可嵌套 如: $x > 0 ? 1 : (x < 0 ? -1 : 0)$

● 优先级: 14

● 结合方向: *自右向左*

如: $a > b ? a : c > d ? c : d \Leftrightarrow a > b ? a : (c > d ? c : d)$

● expr1 、 expr2 、 expr3 类型可不同, 表达式值取较高的类型

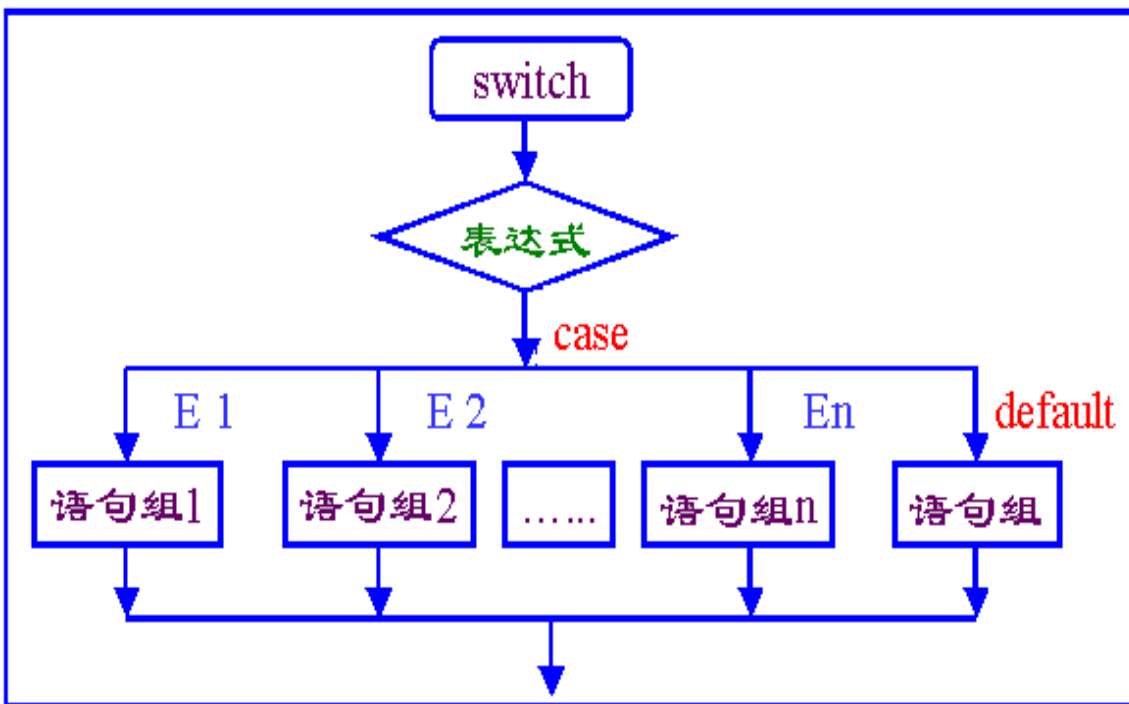
例 $x ? 'a' : 'b'$ // $x=0$, 表达式值为 'b'; $x \neq 0$, 表达式值为 'a'
 $x > y ? 1 : 1.5$ // $x > y$, 值为 1.0; $x < y$, 值为 1.5

例3.7 大写字母转换成小写字母。

```
#include <iostream>
using namespace std;
int main( )
{
    char ch;
    cin>>ch;
    ch=(ch>='A' && ch<='Z')?(ch+32): ch;
    cout<<ch<<endl;
    return 0;
}
```

3.9 switch语句 (开关分支语句)

❖一般形式:



```
switch( 表达式)
{   case   E1:
        语句组 1;
        break;

    case   E2:
        语句组 2;
        break;

    .....

    case   En:
        语句组 n;
        break;
    [default:
        语句组 ;
        break;]
}
```

❖说明:

- E1,E2,...En是**常量表达式**,且值必须互不相同
- 语句标号作用, **必须用break跳出**
- 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;
```

.....

```
例  switch(score)
    {   case 5:  cout<<"Very good!"<<endl;
        case 4:  cout<<"Good!"<<endl;
        case 3:  cout<<"Pass!"<<endl;
        case 2:  cout<<"Fail!"<<endl;
        default : cout<<"data error!"<<endl;
    }
```

运行结果：score为5时，输出：

Very good! Good! Pass! Fail! data error!

```
#include <iostream>
using namespace std;
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<<"a="<<a<<","b="<<b<<endl;
    return 0;
}
```

运行结果：a=2,b=1

3.10 编写选择结构的程序

例3.8 编写程序，判断某一年是否为闰年。

闰年条件：

- 1、能被4整除，但不能被100整除
- 2、能被100整除，又能被400整除

例3.8 编写程序，判断某一年是否为闰年。

```
#include <iostream>
using namespace std;
int main( )
{ int year;
  cout<<"please enter year: "; //输出提示
  cin>>year; //输入年份
  if (year % 4 == 0 && year % 100 != 0) || year % 400 == 0 )
    cout<<year<<" is a leap year "; //若条件为真，则该年为闰年
  else
    cout<<year<<" is not a leap year "; //否则，该年为非闰年
  return 0;
}
```

作业：

P 33-34

3.11 循环结构和循环语句

C++可实现循环的语句：

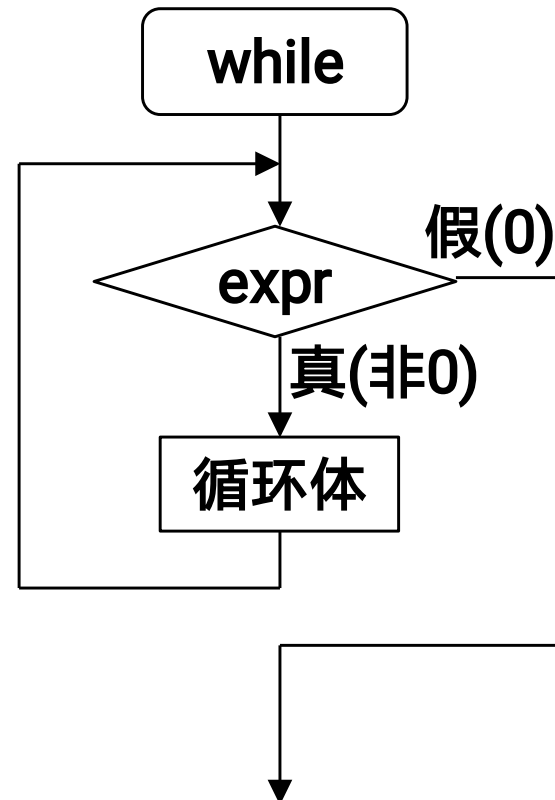
- ✿ 用goto 和 if 构成循环
- ✿ while 语句
- ✿ do ~ while 语句
- ✿ for 语句

● while语句

◆ 一般形式：

```
while(表达式)  
    循环体语句；
```

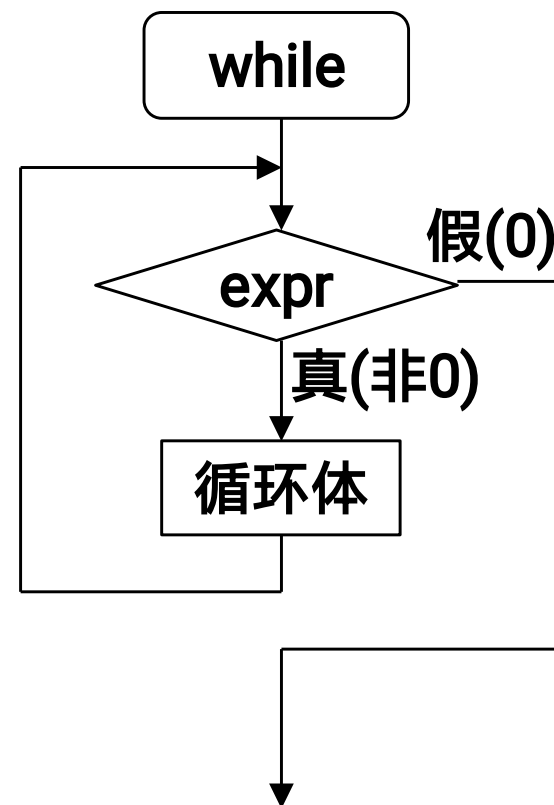
◆ 执行流程：



◆特点：先判断表达式，后执行循环体

◆说明：

- 循环体有可能一次也不执行
- 循环体可为任意类型语句
- 下列情况，退出while循环
 - ◆条件表达式不成立（为零）
 - ◆循环体内遇break, return, goto



例 用while循环求

$$\sum_{n=1}^{100} n$$

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

循环初值

$i=1;$

循环条件

$\text{while}(i \leq 100)$

循环终值

循环变量增值

$i++;$

循环体

$\{ \text{sum}=\text{sum}+\text{i};$
 $\text{i}++;$
 $\}$

$\text{cout}<<\text{sum}<<\text{endl};$

例 显示1~10的平方

```
#include <iostream>
using namespace std;
int main()
{   int i=1;
    while(i<=10)
    {
        cout<<i<<"*" <<i <<"="<<i*i);
        i++;
    }
    return 0;
}
```

运行结果：

1*1=1
2*2=4
3*3=9
4*4=16
5*5=25
6*6=36
7*7=49
8*8=64
9*9=81
10*10=100

do~while语句

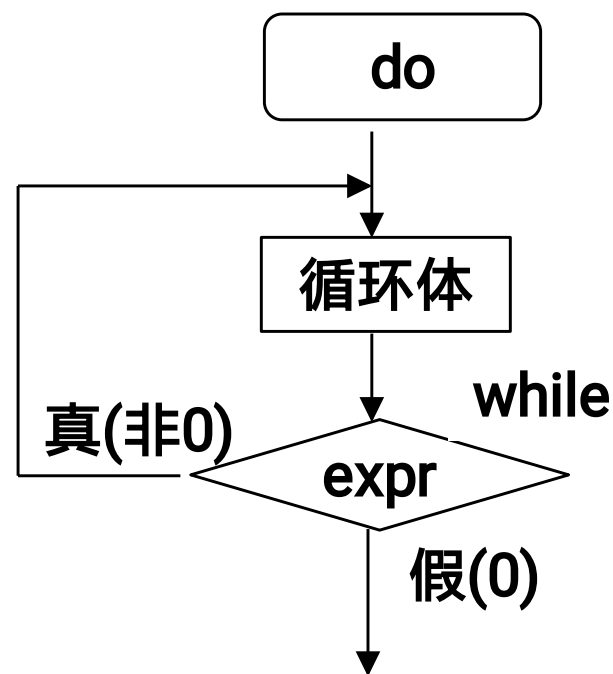
◆一般形式:

do

循环体语句;

while(表达式);

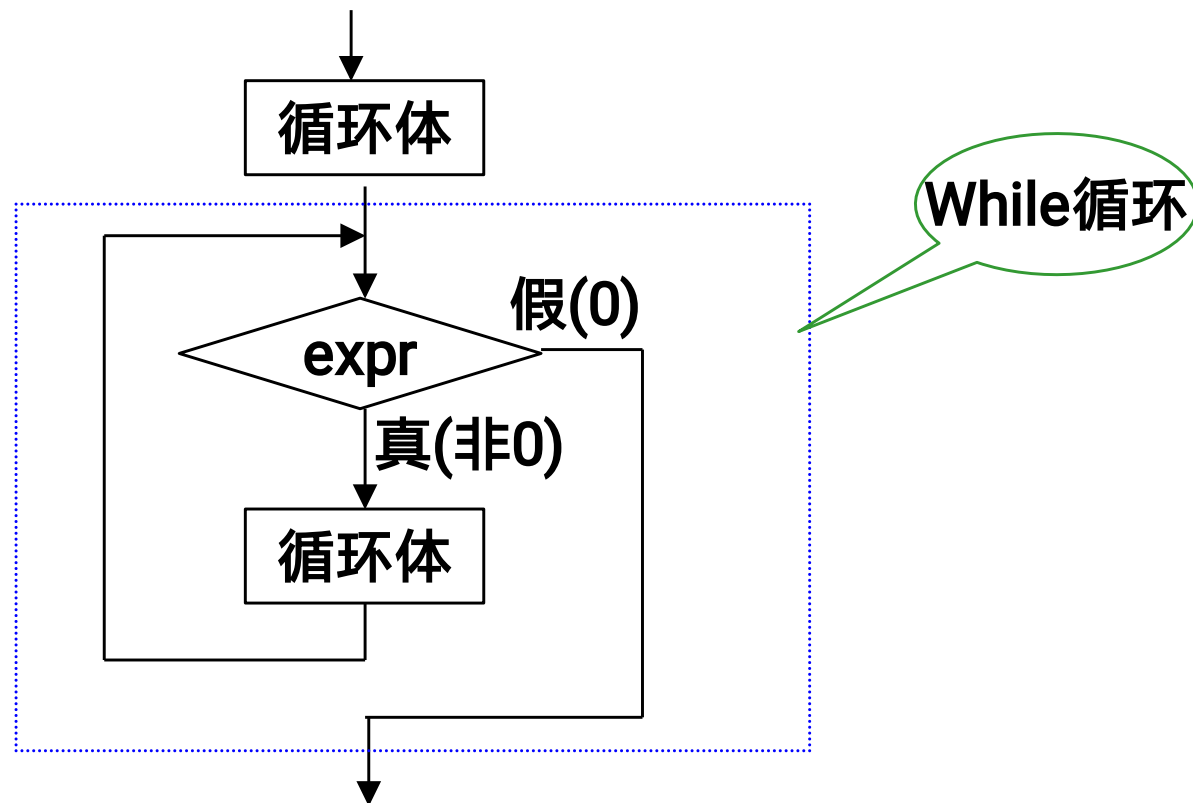
◆执行流程:



◆特点：先执行循环体，后判断表达式

◆说明：

- 至少执行一次循环体
- do~while可转化成while结构



例 用do~while循环求

$$\sum_{n=1}^{100} n$$

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

3.11.4 几种循环的比较

例 while和do~while比较

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

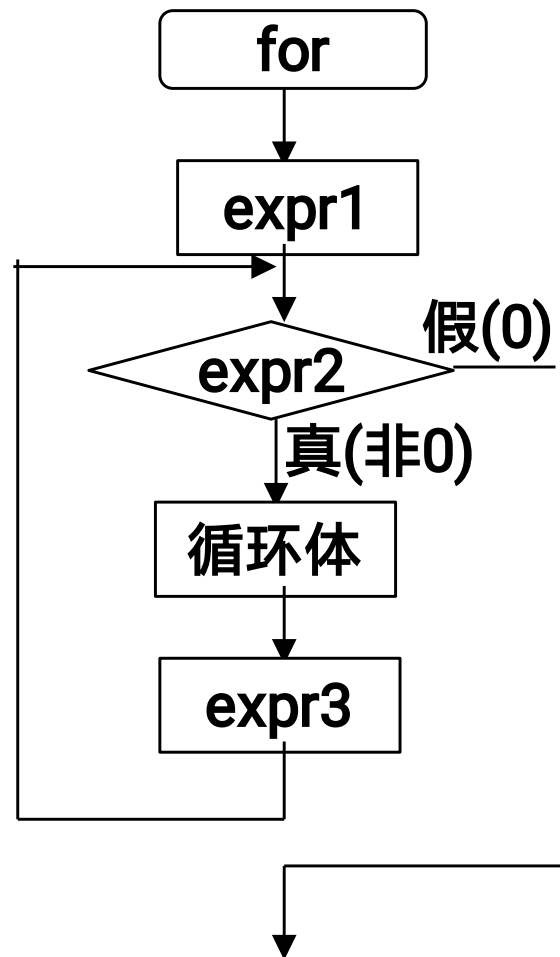
```
#include <iostream>
int main()
{  int i,sum=0;
   cin>>i;
   while(i<=10)
   {  sum+=i;
     i++;
   }
   cout<<sum<<endl;
   return 0;
}
```

● for语句

◆ 一般形式:

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

◆ 执行流程:



◆ for语句一般应用形式:

```
for( 循环变量赋初值; 循环条件; 循环变量增值 )  
{ 循环体语句; }
```

例 用for循环求

$$\sum_{n=1}^{100} n$$

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

◆ for语句一般应用形式:

```
for( 循环变量赋初值; 循环条件; 循环变量增值 )  
{ 循环体语句; }
```

◆ 说明:

- for语句中expr1, expr2, expr3 类型任意, 都可省略, 但分号; 不可省
- 无限循环: for(;;)
- for语句可以转换成while结构

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

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

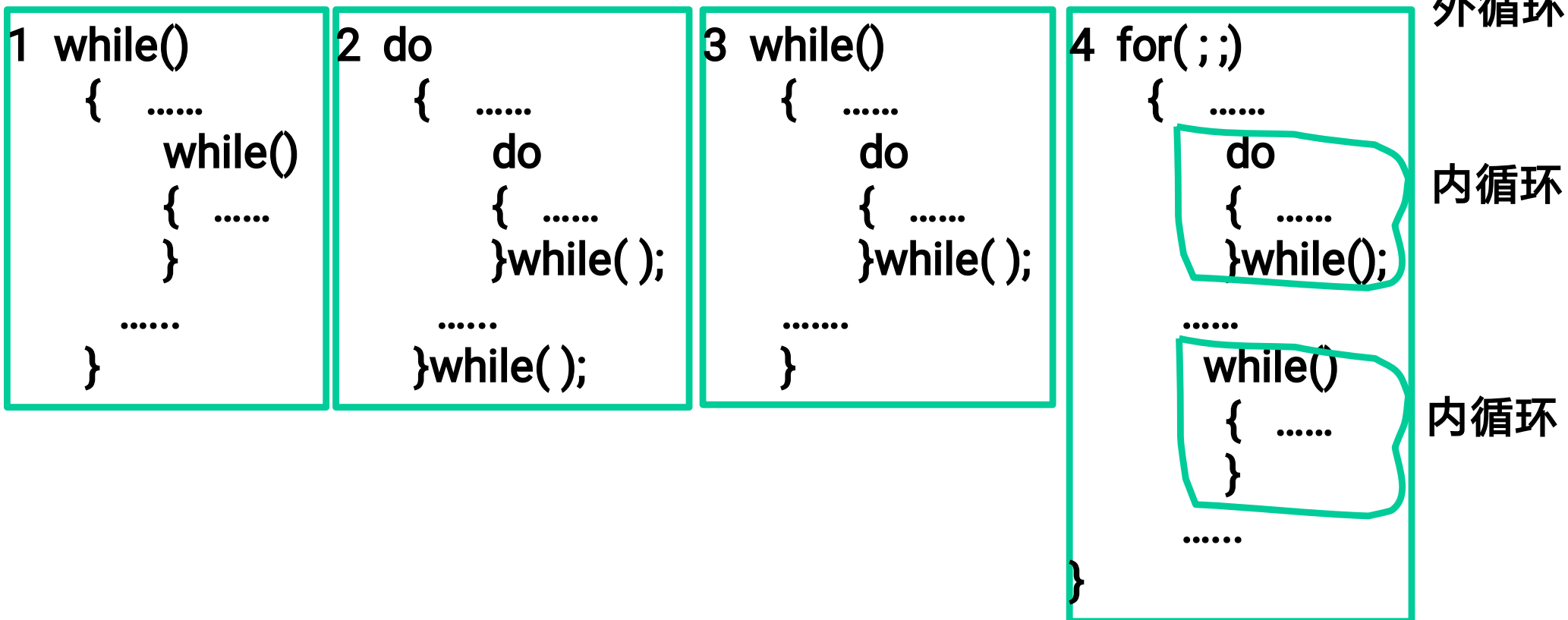
```
例：#include<iostream>
using namespace std;
int main( )
{   int i=0;
    for(;i<10;)
        putchar('a'+(i++));
    return 0;
}
```

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

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

● 循环的嵌套

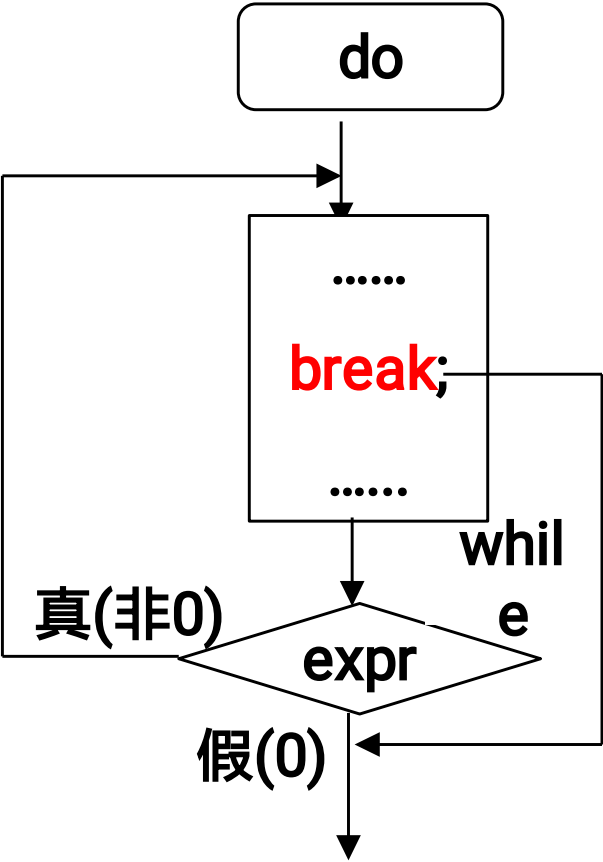
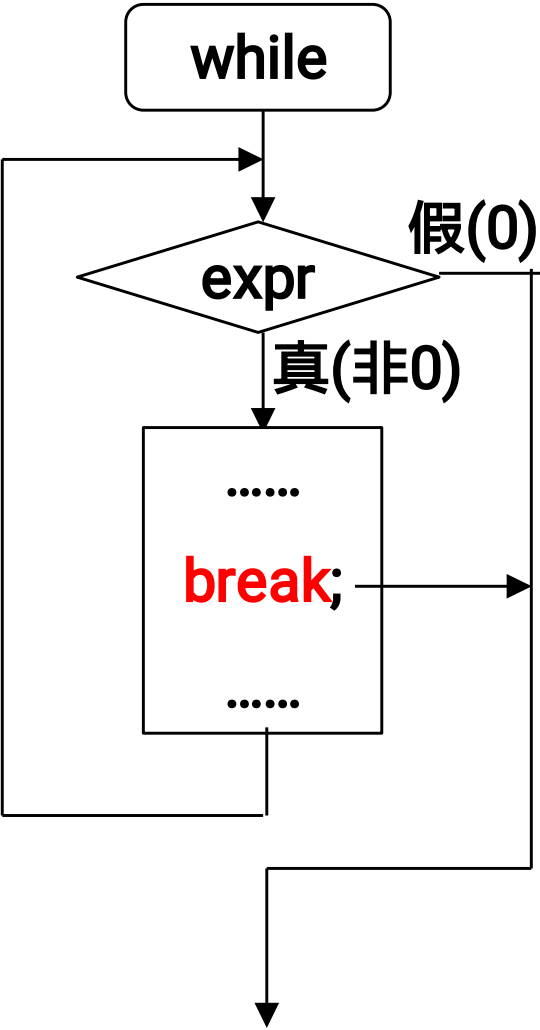
- ◆ 三种循环可互相嵌套,层数不限
- ◆ 外层循环可包含两个以上内循环,但不能相互交叉

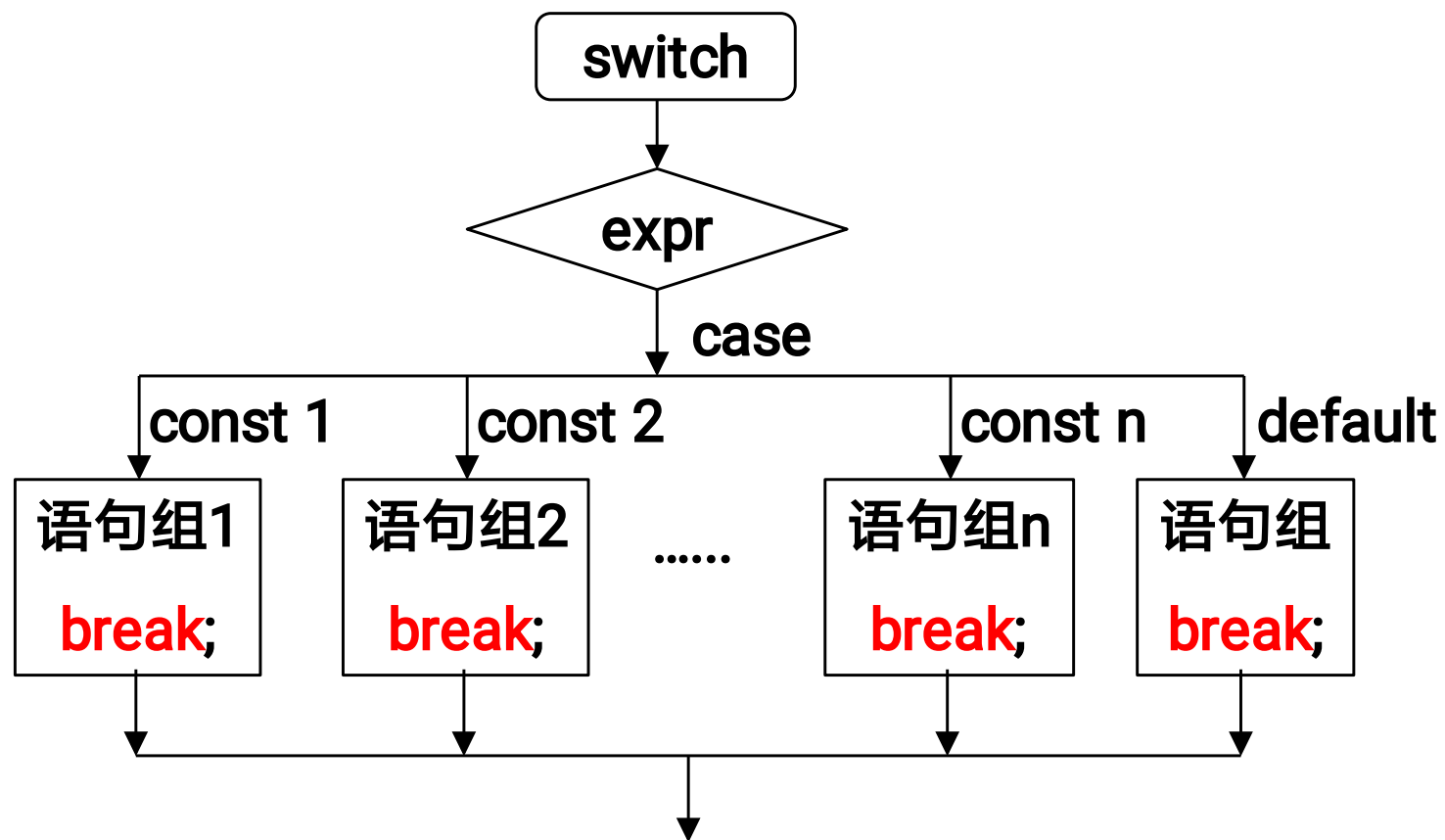
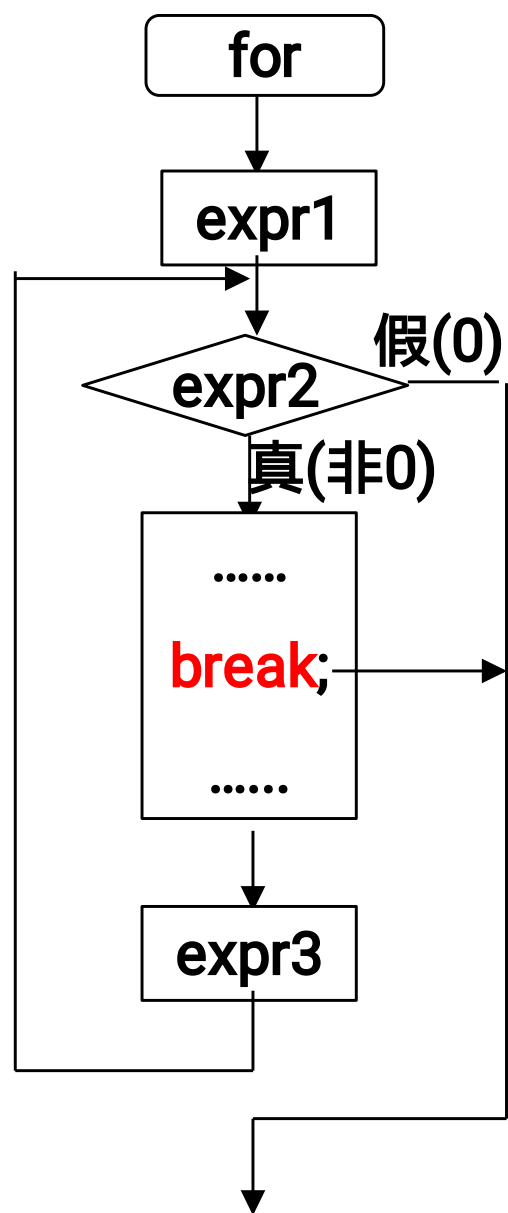


3.13 break语句和continue语句

● break语句

- ◆ 一般格式为: `break;`
- ◆ 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





例 break举例：输出圆面积，面积大于100时停止

```
#define PI 3.14159 //const float PI=3.14159;
#include<iostream>
using namespace std;
int main()
{
    int r;    float area;
    for(r=1;r<=10;r++)
    {
        area=PI*r*r;
        if(area>100) break;
        cout<<"r=" <<r<<"area=" <<area<<endl;
    }
    return 0;
}
```

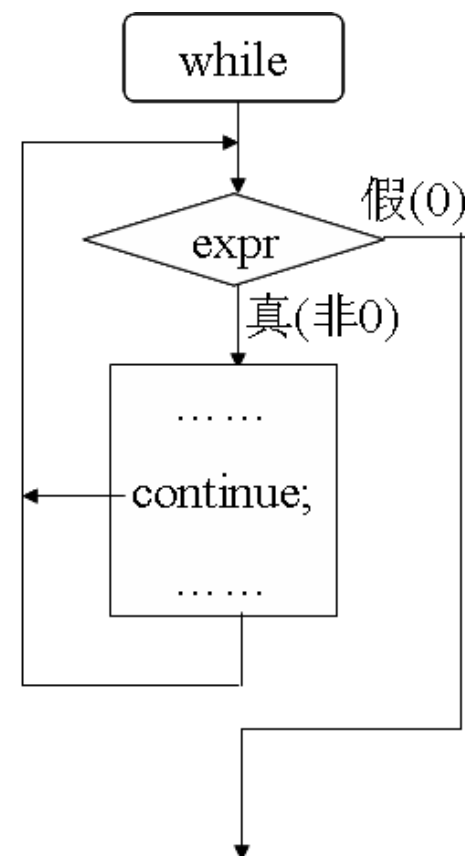
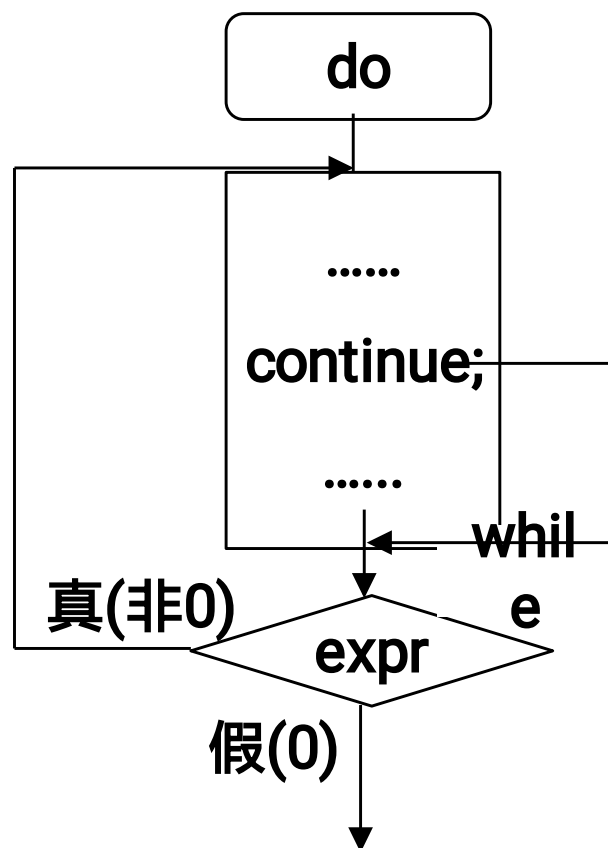
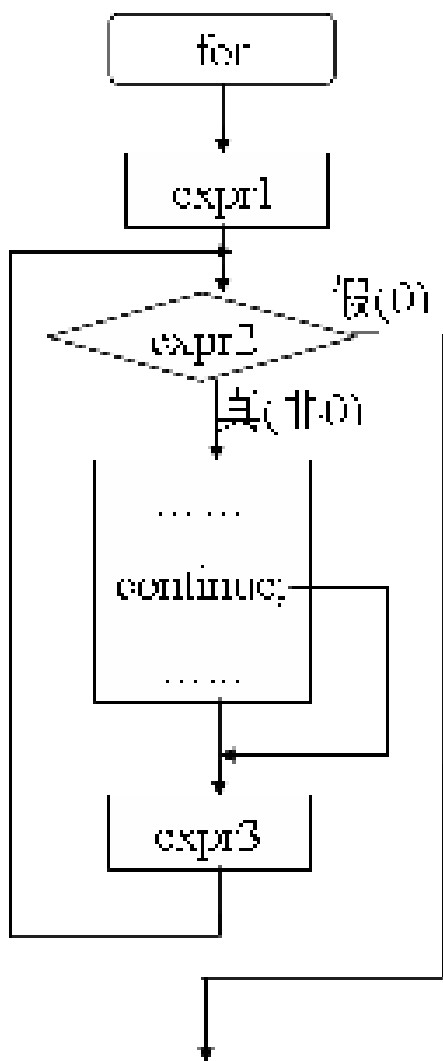
例 break举例：小写字母转换成大写字母,直至输入非字母字符

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

continue语句

◆ 功能：结束本次循环，跳过**循环体中**尚未执行的语句，进行下一次是否执行循环体的判断

◆ 仅用于循环语句中



例 求输入的十个整数中正数的个数及其平均值

```
#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<<" 个正数的和为:"<<sum;
    cout<<"平均值:"<<sum/num;
    return 0;
}
```

3.14 编写循环结构的程序

例用 $\frac{\pi}{4} \approx 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots$ 公式求 π 的近似值，直到最后一项的绝对值小于 10^{-6} 为止

分子: 1, -1, 1, -1...

分母: 1, 3, 5, 7, ...


```
#include<iostream>
#include<iomanip>
#include<cmath>
using namespace std;
int main()
{ int s; double n,t,pi;
  t=1,pi=0,n=1.0,s=1;
  while(fabs(t)>=1e-6)
  { pi=pi+t; n=n+2;
    s=-s; t=s/n;
  }
  pi=pi*4;
  cout<<"pi=" <<setprecision(6)<<pi<<endl;
  return 0;
}
```

t=1,pi=0,n=1.0,s=1	
当 $ t \geq 1e-6$	
	pi=pi+t
	n=n+2
	s=-s
	t=s/n
pi=pi*4	
输出pi	

运行结果：pi= 3.141594

例用 $\frac{\pi}{4} \approx 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots (-1)^{n-1} \frac{1}{2n-1} \dots$ 公式求 π 的近似值。

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{ int i,n; float s=0.0;
  cin>>n;
  for (i=1;i<=n;i++)
    s+=pow(-1.0,i-1)/(2*i-1.0);
  s*=4.0;
  cout<<"pi="<<s<<endl;
  return 0;
}
```

输入的n值越大，就越逼近 π 的精确值。

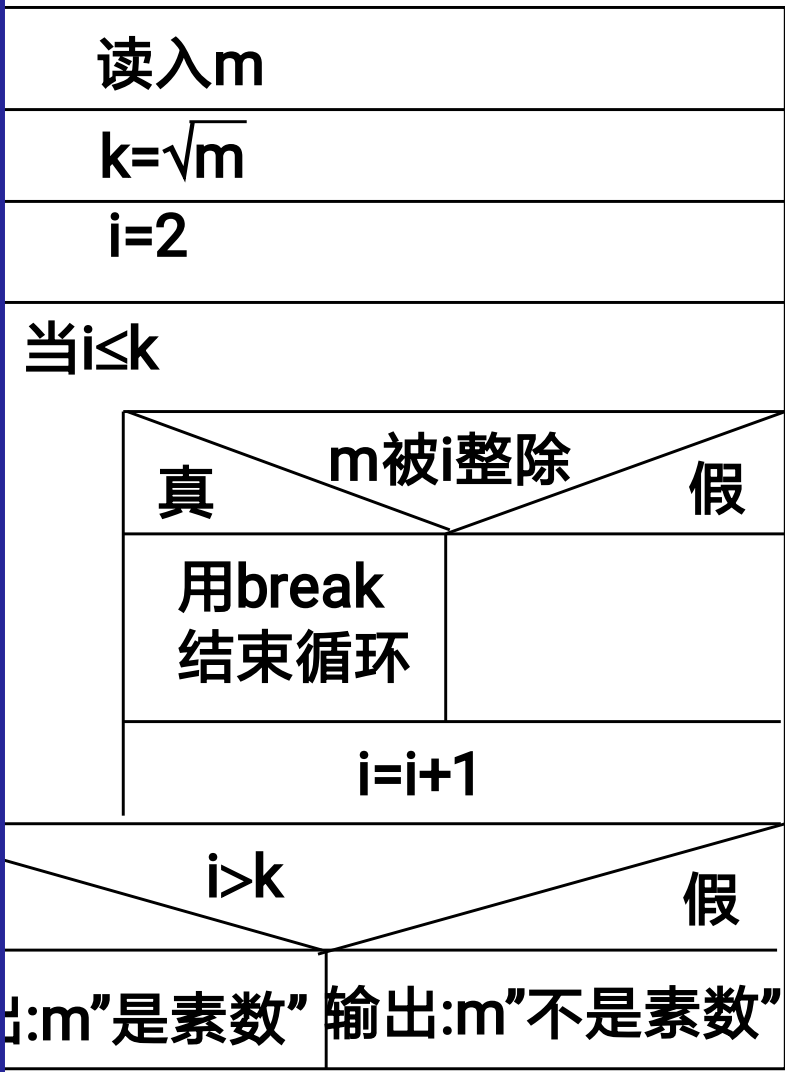
例 求Fibonacci数列: 1, 1, 2, 3, 5, 8,的前40个数

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
    long f1=1,f2=1; int i;
    for(i=1;i<=20;i++)
    {
        cout<<setw(12)<<f1 <<setw(12)<<f2;
        if(i%2==0) cout<<endl;
        f1=f1+f2;  f2=f2+f1;
    }
    return 0;
}
```

1	1	2	3
5	8	13	21
34	55	89	144
233	377	610	987
1597	2584	4181	6765
10946	17711	28657	46368
75025	121393	196418	317811
514229	832040	1346269	2178309
3524578	5702887	9227465	14930352
24157817	39088169	63245986	102334155

例 判断m是否素数

```
#include <cmath>
#include <iostream>
using namespace std;
int main()
{  int m,i,k;
   cin>>m;
   k=sqrt(double(m));
   for(i=2;i<=k;i++)
       if(m%i==0) break;
   if(i>k)
       cout<<m<<"是素数"<<endl;
   else
       cout<<m<<"不是素数"<<endl;
   return 0;
}
```



例 找出100~200之间的素数

```
#include<iostream>
#include<iomanip>
#include<cmath>
using namespace std;
int main()
{   int m,i,k,n=0;
    bool prime;
```

```
    for(m=101;m<=200;m+=2)
    {   prime=true;
        k=int(sqrt(double(m)));
        for(i=2;i<=k;i++)
            if(m%i==0){prime=false;break;}
        if(prime)
            { cout<<setw(5)<<m;n+=1; }
    }
    cout<<endl;
    return 0;
}
```

运行结果：

```
101 103 107 109 113 127 131 137 139 149 151
157 163 167 173 179 181 191 193 197 199
```

例 译密码。将当前字母变成其后的第4个字母。

```
#include<iostream>
using namespace std;
int main()
{ char c;
  while((c=getchar())!='\n')
  { if((c>='a'&&c<='z')||(c>='A'&&c<='Z'))
    { c=c+4;
      if(c>'Z'&&c<='Z'+4||c>'z')c=c-26;
    }/*'Z+4'会控制正常小写字母不-26*/
    cout<<c;
  }
  cout<<endl; return 0;
}
```

运行结果如下：

I am going to Beijing!
M eq ksmrk xs Fimnmrk!

A~Z → 65~90
a~z → 97~122

例：输出1~100之间个位数为6的所有整数

```
#include<iostream>
using namespace std;
int main()
{ int i,j;
  for(i=0; i<=10;i++)
  { j=i*10+6;
    if(j>100)continue;
    cout<<j;
  }
  cout<<endl;
  return 0;
}
```

- 1)此题能否将continue换为break?
- 2)本题若让你自己做，你会如何考虑?

作业：

P 8-15

关于C++的课堂学习，每人必须选且仅选一项：

A.上课完全听不懂！

B.上课大部分内容跟不上老师的进度，课后复习后还是有部分内容不理解。

C.上课少部分内容跟不上老师的进度，课后复习后可理解。

D.上课几乎全部可以跟上老师的进度。

E.上课的进度有点慢，可以进一步加快些进度。

F.上课的进度太慢，学习的内容太少。