3.2 关系运算、逻辑运算与选择结构

3.2.1 关系运算与关系表达式

3.2.2 逻辑常量、逻辑运算与逻辑表达式

3.2.3 选择结构-if语句

3.2.4 选择结构-switch语句

3.2.5 条件运算符与条件表达式

3.2 关系、逻辑运算与选择结构

3.2.1 关系运算与关系表达式

- 关系运算符种类

关系运算符

关系运算符	含义
<	小于
<=	小于或等于
==	等于
>	大于
>=	大于或等于
!=	不等于

将两个值进行比较，结果为“真”（用1表示）或“假”（用0表示）。

例：
3>5 值为假(0)
1!=2 值为真(1)

3.2 关系、逻辑运算与选择结构

3.2.1 关系运算与关系表达式

➢关系表达式

用关系运算符将表达式连接起来的式子

关系表达式的值是逻辑值，即真或假

例：

a+b>c+d

(a==1)>(b==2)

(a=1)>(b=2)

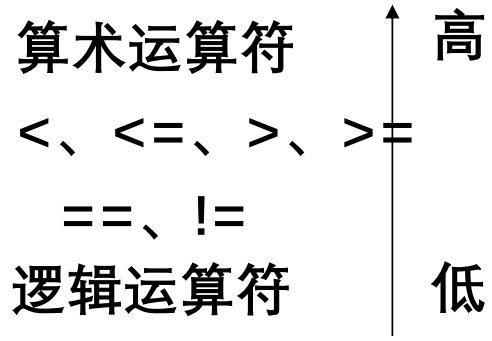
3.2 关系、逻辑运算与选择结构

3.2.1 关系运算与关系表达式

- 优先级

  < 、 <= 、> 、 >=（优先级相同）高于 ==、 !=（优先级相同）

  低于算术运算符，高于赋值运算符



- 结合性

  左结合

3.2 关系、逻辑运算与选择结构

3.2.1 关系运算与关系表达式

例：

(1)x+3>y-2

等价于(x+3)>(y-2)而非x+(3>y)-2

(2)假定a=3,b=2,c=1,f=a>b>c，则f=？ 0

>运算符从左向右结合，先执行a>b得值1，执行1>c,得值0，赋给f

3.2 关系、逻辑运算与选择结构

3.2.1 关系运算与关系表达式

➢几点注意：

✓字符型可以进行大小比较，按ascii码值进行比较

例：'a'<'B'　　值为0

✓一般应避免对两个实数作相等或不等的比较，要考虑误差

例：fabs(a-b)<1e-6

✓ 特别注意区分=和==

例：

musicians==4　　关系表达式 值为1或0

musicians=4　　赋值表达式 值为4

3.2 关系、逻辑运算与选择结构

3.2.2 逻辑常量、逻辑运算与逻辑表达式

3.2.2.1 逻辑常量和逻辑变量

逻辑常量：true/false

逻辑变量：bool 变量名

在内存中占一个字节

true处理为1，false处理为0

例：

```
bool flag=true;
cout<<flag;
```

1

可以用boolalpha设置（noboolalpha取消）显示true/false，而非1/0。

3.2 关系、逻辑运算与选择结构

3.2.2 逻辑常量、逻辑运算与逻辑表达式

3.2.2.1 逻辑常量和逻辑变量

- 逻辑型数据参与运算时，true转换成1，false转换为0

例：
```
int a=100;
bool flag=true;
a=a+flag;
cout<<a;
```
101

- 非逻辑型按“非0为真0为假”转换为逻辑型

例：
```
bool flag=100;
cout<<flag;
```
1

3.2 关系、逻辑运算与选择结构

3.2.2 逻辑常量、逻辑运算与逻辑表达式

3.2.2.2 逻辑运算与逻辑表达式

- **逻辑运算符**

逻辑与 &&、逻辑或 ||、逻辑非！
&& 和 || 是双目运算符，！是单目运算符

逻辑运算真值表

exp1	exp2	!exp1	exp1&&exp2	exp1||exp2
真	真	假	真	真
真	假	假	假	真
假	真	真	假	真
假	假	真	假	假

3.2 关系、逻辑运算与选择结构

3.2.2 逻辑常量、逻辑运算与逻辑表达式

3.2.2.2 逻辑运算与逻辑表达式

对于&&，必须是两个运算对象都为真时，结果才为真；
对于||，只要两个运算对象中有一个为真结果就为真，只有两个都为假时，结果才为假。

例：
5>3&&5>10 结果为假
5>3||5>10 结果为真
若a为真，则!a为假

3.2 关系、逻辑运算与选择结构

3.2.2 逻辑常量、逻辑运算与逻辑表达式

3.2.2.2 逻辑运算与逻辑表达式

➢逻辑表达式

逻辑运算符把逻辑量或表达式连接起来

逻辑运算符两端的表达式，任意类型都可以，以“非0为真0为假”判定真假

逻辑表达式取值为真或假

例：

假设有int x=3,int y=0;，则：

x&&y=　　0

x||y=　　1

!x=　　0

3.2 关系、逻辑运算与选择结构

3.2.2 逻辑常量、逻辑运算与逻辑表达式

3.2.2.2 逻辑运算与逻辑表达式

- 优先级

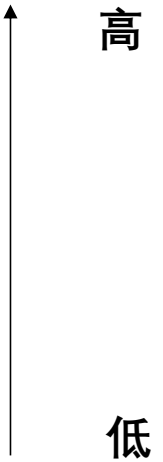
  从高到底： !——> &&——> ||

  !优先级高于算术运算符 &&和||优先级低于关系运算符

- 结合性

  &&、|| 左结合

  ! 右结合



3.2 关系、逻辑运算与选择结构

3.2.2 逻辑常量、逻辑运算与逻辑表达式

3.2.2.2 逻辑运算与逻辑表达式

例：

(1)x>5&&x<10

等价于 (x>5)&&(x<10)

(2)age>30&&age<45||weight>300

等价于 (age>30&&age<40)||(weight>300)

3.2 关系、逻辑运算与选择结构

3.2.2 逻辑常量、逻辑运算与逻辑表达式

3.2.2.2 逻辑运算与逻辑表达式

(3)比较

(age>50||weight>300)&&donation>1000

age>50||weight>300&&donation>1000

(4)比较

!(x>5)　　//x>5时结果为假，x<=5时结果为真，等价于表达式x<=5

!x>5　　//结果总为假，因为!x结果只能为真或假，转换成1或0

3.2 关系、逻辑运算与选择结构

3.2.2 逻辑常量、逻辑运算与逻辑表达式

3.2.2.2 逻辑运算与逻辑表达式

例：根据条件写出逻辑表达式

(1)age在18到34岁之间

表达式：age>17&&age<35

错误的写法：17<age<35　　结果总为真！

(2)判断year是否为闰年，闰年是值能被4整除但不能被100整除，或者能被400整除的年份。

表达式：year%4==0&&year%100!=0||year%400==0

3.2 关系、逻辑运算与选择结构

3.2.2 逻辑常量、逻辑运算与逻辑表达式

3.2.2.2 逻辑运算与逻辑表达式

(3)变量c为字母

表达式：c>='a'&&c<='z'||c>='A'&&c<='Z'

(4)x小于10且|y|>8

表达式：x<10&&y>8||y<-8 ×

表达式：x<10&&(y>8||y<-8) √

3.2 关系、逻辑运算与选择结构

3.2.2 逻辑常量、逻辑运算与逻辑表达式

3.2.2.2 逻辑运算与逻辑表达式

注意：在逻辑表达式的求解中，从左向右进行，在知道答案后立即停止。即：
对于逻辑&&运算，如果左侧表达式为假，则不计算右侧表达式；
对于逻辑||运算，如果左侧表达式为真，则不计算右侧表达式。

3.2 关系、逻辑运算与选择结构

3.2.2 逻辑常量、逻辑运算与逻辑表达式

3.2.2.2 逻辑运算与逻辑表达式

例：

(1)假设有 x=0,则表达式值为?

x!=0&&1.0/x>100.0　　0

(2)分析以下程序段执行后a、b、c、m的值?

```
int a, b, c, m;
a = b = c = 1;
m = a && b-- || ++c;
```

1 0 1 1

3.2 关系、逻辑运算与选择结构

3.2.2 逻辑常量、逻辑运算与逻辑表达式

3.2.2.2 逻辑运算与逻辑表达式

逻辑运算符的另一种表示方式

运算符	另一种表示方法
&&	and
||	or
!	not

3.2 关系、逻辑运算与选择结构

3.2.3 选择结构-if语句

选择结构：根据条件的值来判断程序的走向

两种选择控制语句：if语句和switch语句

3.2.3.1 if语句的三种形式

单分支、双分支、多分支结构

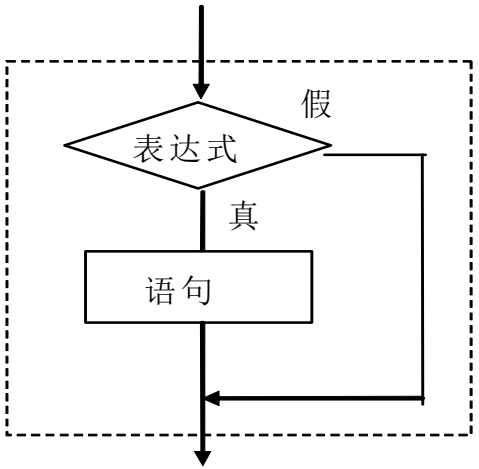
3.2 关系、逻辑运算与选择结构

3.2.3 选择结构-if语句

3.2.3.1 if语句的三种形式

(1)单分支if语句

```
if(表达式)
    语句
```



- ✓当表达式为真时执行语句，否则不执行
- ✓测试条件表达式通常为关系或逻辑表达式，
  实际上可以为任意类型，按“非0为真0为假”判断真假
- ✓语句可以是一条语句，也可以是大括号括起来的语句块
- ✓整个if语句被视为一条语句

3.2 关系、逻辑运算与选择结构

3.2.3 选择结构-if语句

3.2.3.1 if语句的三种形式

例：

```
int a=3;
char ch='0';
```

考虑以下语句执行结果：

```
if(a==5) cout<<"ok";
if(a=5) cout<<"ok";
if(a!=0) cout<<"ok";
if(a) cout<<"ok";
if(!a) cout<<"ok";
if(a==0) cout<<"ok";
if(ch) cout<<"ok";
```

例：

```
#include <iostream>
using namespace std;
int main()
{
    char ch;
    int spaces = 0;
    int total = 0;
    cin.get(ch);
    while (ch != '.')   //循环判断ch是否为'.'字符
    {
        if (ch == ' ')
            ++spaces;
        ++total;
        cin.get(ch);
    }
    cout << spaces << " space, " << total;
    cout << " character total in sentence\n";
    return 0;
}
```

不属于if语句

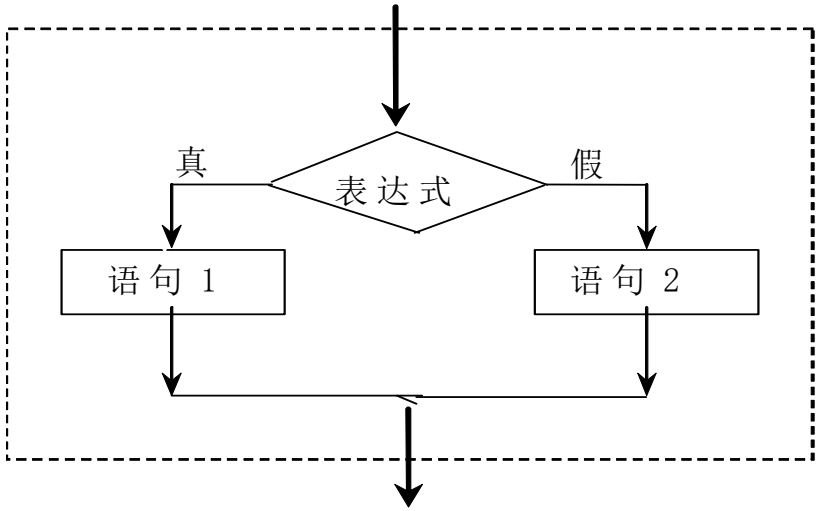
3.2 关系、逻辑运算与选择结构

3.2.3 选择结构-if语句

3.2.3.1 if语句的三种形式

(2)双分支if else语句

```
if(表达式)
    语句1
else
    语句2
```



✓当表达式为真时执行语句1，否则执行语句2

✓测试条件表达式通常为关系或逻辑表达式，

实际上可以为任意类型，按“非0为真0为假”判断真假

✓语句1、2可以是一条语句，也可以是大括号括起来的语句块

✓整个if else结构被视为一条语句

例：

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

例：分析以下程序段：

```
if (ch == 'Z')
    zorro++;
    cout << "Another Zerro candidate\n";
else
    dull++;
    cout << "Not a Zerro candidate\n";
```

错！

```
if (ch == 'Z')
{
    zorro++;
    cout << "Another Zerro candidate\n";
}
else
{
    dull++;
    cout << "Not a Zerro candidate\n";
}
```

应使用复合语句！

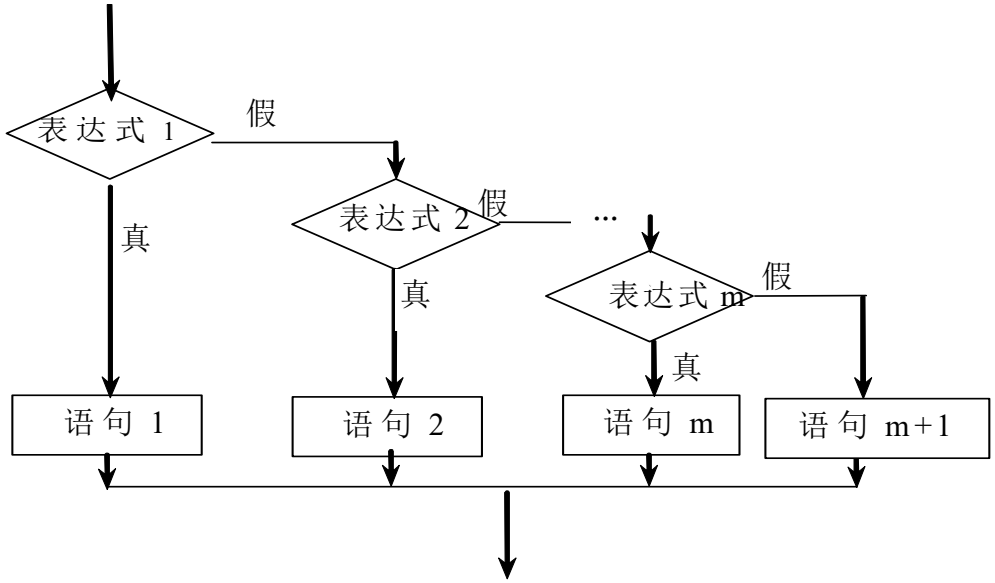
3.2 关系、逻辑运算与选择结构

3.2.3 选择结构-if语句

3.2.3.1 if语句的三种形式

(3)多分支if else 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m)
    语句m
[else
    语句m+1]
```



3.2 关系、逻辑运算与选择结构

3.2.3 选择结构-if语句

3.2.3.1 if语句的三种形式

(3)多分支if else if 语句

- ✓ 表达式1的值为真时，执行语句1；否则判断当表达式2的值为真时执行语句2；依此类推，若表达式的值都为假，则执行语句m+1
- ✓ 测试条件表达式1、2,…,m和语句1、2,…,m+1的用法和要求同前两种形式
- ✓ 整个if else if结构被视为一条语句

例：输入一个0~100的百分制成绩score，输出其对应的五级制成绩“优”(score≥90)、“良”(80≤scorek<90)、“中”(70≤score<80)、“及格”(60≤score<70)和“不及格”(score<60)。

分析以下程序段：

```
if (score >= 90)
    cout << "优" << endl;
else if (score >= 80 && score < 90)
    cout << "良" << endl;
else if (score >= 70 && score < 80)
    cout << "中" << endl;
else if (score >= 60 && score < 70)
    cout << "及格" << endl;
else
    cout << "不及格" << endl;
```

```
if (score >= 90)
    cout << "优" << endl;
else if (score >= 80)
    cout << "良" << endl;
else if (score >= 70)
    cout << "中" << endl;
else if (score >= 60)
    cout << "及格" << endl;
else
    cout << "不及格" << endl;
```

```
if (score >= 90)
    cout << "优" << endl;
else if (80 <= score < 90)
    cout << "良" << endl;
else if (70 <= score < 80)
    cout << "中" << endl;
else if (60 <= score < 70)
    cout << "及格" << endl;
else
    cout << "不及格" << endl;
```

错！

```
if (score >= 60)
    cout << "及格";
else if (score >= 70)
    cout << "中";
else if (score >= 80)
    cout << "良";
else if (score >= 90)
    cout << "优";
else
    cout << "不及格";
```

错！

3.2 关系、逻辑运算与选择结构

3.2.3 选择结构-if语句

3.2.3.2 if语句的嵌套

3.2 关系、逻辑运算与选择结构

3.2.3 选择结构-if语句

3.2.3.2 if语句的嵌套

✓if或else后面的执行语句又是if语句，称为if语句的嵌套

例：

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

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

3.2 关系、逻辑运算与选择结构

3.2.3 选择结构-if语句

3.2.3.2 if语句的嵌套

例：

```
if (ch == 'A')
    a_grade++;
else
    if (ch == 'B')
        b_grade++;
    else
        soso++;
```

3.2 关系、逻辑运算与选择结构

3.2.3 选择结构-if语句

3.2.3.2 if语句的嵌套

例：

```
if (ch == 'A')
      a_grade++;
else
      if (ch == 'B')
            b_grade++;
      else
            soso++;
```

```
if (ch == 'A')
      a_grade++;
else if (ch == 'B')
      b_grade++;
else
      soso++;
```

3.2 关系、逻辑运算与选择结构

3.2.3 选择结构-if语句

3.2.3.2 if语句的嵌套

例：

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

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

3.2 关系、逻辑运算与选择结构

3.2.3 选择结构-if语句

3.2.3.2 if语句的嵌套

例：

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

X

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

3.2 关系、逻辑运算与选择结构

3.2.3 选择结构-if语句

3.2.3.2 if语句的嵌套

例：

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

X

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

✓ 应注意if else的配对关系

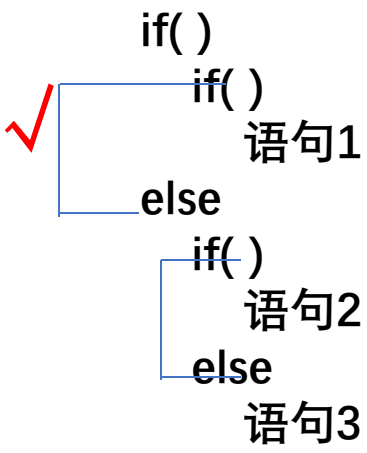
else总是与它上面最近的未曾配对的if配对

用{}可改变优先级，从而改变配对次序

3.2 关系、逻辑运算与选择结构

3.2.3 选择结构-if语句

3.2.3.2 if语句的嵌套



正确的缩进形式：

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

3.2 关系、逻辑运算与选择结构

3.2.3 选择结构-if语句

3.2.3.2 if语句的嵌套

加{}改变配对次序

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

3.2 关系、逻辑运算与选择结构

3.2.4 选择结构-switch语句

3.2 关系、逻辑运算与选择结构

3.2.4 选择结构-switch语句

形式：

```
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
}
```

执行流程：计算switch后括号内表达式的值，与case后面的常量表达式比较，与某个常量表达式的值相等，则执行该常量表达式后相应语句序列，若找不到相匹配的常量表达式，则执行default后面的语句序列。

3.2 关系、逻辑运算与选择结构

3.2.4 选择结构-switch语句

说明：

- ✓switch后括号内的表达式结果必须为整数值(包括字符型)。
- ✓case后的表达式必须为整型常量表达式，最常见的是int或char常量，且互不相同。
- ✓为保证多个选项的正确实现，通常在各语句序列最后使用break语句跳出switch结构，若无，则继续执行后面的语句序列。
- ✓default选项是可选的，如果被省略，又没有其他匹配选项，程序跳到switch后面的语句。
- ✓语句序列可以是一条或多条语句，不必用{}括起来。
- ✓各个case的出现顺序可以任意，每个case都有break的情况下，case的次序不影响执行结果。
- ✓多个case可以共用语句序列。

3.2 关系、逻辑运算与选择结构

3.2.4 选择结构-switch语句

例：

```
char ch;
cin >> ch;
switch (ch)
{
    case 'A':    a_grade++;
                 break;
    case 'B':b_grade++;
                 break;
    case 'C':    c_grade++;
                 break;
    case 'D':    d_grade++;
                 break;
    default:     cout << "输入有误" << endl;
}
```

3.2 关系、逻辑运算与选择结构

3.2.4 选择结构-switch语句

例：

```
char ch;
cin >> ch;
switch (ch)
{
    case 'A':    a_grade++;
                 break;
    case 'B':b_grade++;
                 break;
    case 'C':    c_grade++;
                 break;
    case 'D':    d_grade++;
                 break;
    default:     cout << "输入有误" << endl;
}
```

若省去break，会怎样?

例：已知百分制成绩score，使用switch语句显示对应五级制成绩。

```
#include <iostream>
using namespace std;
int main()
{
    int score;
    cout << "请输入百分制成绩(0~100)：";
    cin >> score;
    switch (score /10)
    {
        case 10:
        case 9:cout << "优" << endl; break;
        case 8:cout << "良" << endl; break;
        case 7:cout << "中" << endl; break;
        case 6:cout << "及格" << endl; break;
        default:cout << "不及格" << endl;
    }
    return 0;
}
```

与后一个case共用语句序列

例：已知百分制成绩score，使用switch语句显示对应五级制成绩。

```
#include <iostream>
using namespace std;
int main()
{
    int score;
    cout << "请输入百分制成绩(0~100)：";
    cin >> score;
    switch (score /10)
    {
        case 10:
        case 9:cout << "优" << endl; break;
        case 8:cout << "良" << endl; break;
        case 7:cout << "中" << endl; break;
        case 6:cout << "及格" << endl; break;
        default:cout << "不及格" << endl;
    }
    return 0;
}
```

与后一个case共用语句序列

若score 定义成float型，怎么改？

例：已知百分制成绩score，使用switch语句显示对应五级制成绩。

```
#include <iostream>
using namespace std;
int main()
{
    int score;
    cout << "请输入百分制成绩(0~100)：";
    cin >> score;
    switch (score /10)
    {
        case 10:
        case 9:cout << "优" << endl; break;
        case 8:cout << "良" << endl; break;
        case 7:cout << "中" << endl; break;
        case 6:cout << "及格" << endl; break;
        default:cout << "不及格" << endl;
    }
    return 0;
}
```

与后一个case共用语句序列

若score 定义成float型，怎么改?

int(score/10)或int(score)/10

3.2 关系、逻辑运算与选择结构

3.2.4 选择结构-switch语句

switch语句和多分支if语句的比较

3.2 关系、逻辑运算与选择结构

3.2.4 选择结构-switch语句

switch语句和多分支if语句的比较

✓多分支if语句更通用，如果选项涉及取值范围、浮点测试或变量比较等，则应使用多分支if语句。

例：

```
if (age > 17 && age < 35)
      index = 0;
else if (age >= 35 && age < 50)
      index = 1;
else if (age >= 50 && age < 65)
      index = 2;
else
      index = 3;
```

3.2 关系、逻辑运算与选择结构

3.2.4 选择结构-switch语句

switch语句和多分支if语句的比较

✓多分支if语句更通用，如果选项涉及取值范围、浮点测试或变量比较等，则应使用多分支if语句。

例：

```
if (age > 17 && age < 35)
      index = 0;
else if (age >= 35 && age < 50)
      index = 1;
else if (age >= 50 && age < 65)
      index = 2;
else
      index = 3;
```

✓如果所有的选项都可以使用整型常量来标识，二者都可用，选项较多时，应尽量使用switch语句。

3.2 关系、逻辑运算与选择结构

3.2.5 条件运算符与条件表达式

3.2 关系、逻辑运算与选择结构

3.2.5 条件运算符与条件表达式

条件运算符：？：

形式：表达式1？表达式2：表达式3

✓唯一的一个三目运算符

✓执行顺序：如果表达式1为真，则求解表达式2，并将表达式2的值作为整个表达式的值，否则求解表达式3的值，并将表达式3的值作为整个表达式的值。

✓优先级：高于赋值运算符，低于算术、关系、逻辑运算符

✓结合性：右结合

3.2 关系、逻辑运算与选择结构

3.2.5 条件运算符与条件表达式

例：

```
c = a > b ? a : b;
```

3.2 关系、逻辑运算与选择结构

3.2.5 条件运算符与条件表达式

例：

```
c = a > b ? a : b;
```

等价于：

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

3.2 关系、逻辑运算与选择结构

3.2.5 条件运算符与条件表达式

例：

```
c = a > b ? a : b;
```

等价于：

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

例：

```
a > b ? (max = a, min = b) : (max = b, min = a);
```

3.2 关系、逻辑运算与选择结构

3.2.5 条件运算符与条件表达式

条件表达式嵌套：条件表达式嵌套在另一个条件表达式中

例：

函数：$f(x)=\begin{cases}1, & x>0\\ 0, & x=0\\ -1, & x<0\end{cases}$

如何采用条件表达式表示?

3.2 关系、逻辑运算与选择结构

3.2.5 条件运算符与条件表达式

条件表达式嵌套：条件表达式嵌套在另一个条件表达式中

例：

函数：$f(x)=\begin{cases}1, & x>0\\ 0, & x=0\\ -1, & x<0\end{cases}$

如何采用条件表达式表示?

```
f = (x > 0) ? 1 : (x == 0) ? 0 : -1;
```

等价于

```
f = (x > 0) ? 1 : ((x == 0) ? 0 : -1);
```

3.2 关系、逻辑运算与选择结构

3.2.5 条件运算符与条件表达式

条件表达式嵌套：条件表达式嵌套在另一个条件表达式中

例：

函数：$f(x)=\begin{cases}1, & x>0\\ 0, & x=0\\ -1, & x<0\end{cases}$

如何采用条件表达式表示?

```
f = (x > 0) ? 1 : (x == 0) ? 0 : -1;
```

等价于

```
f = (x > 0) ? 1 : ((x == 0) ? 0 : -1);
```

代码复杂时更推荐使用if else语句。

3.3 循环结构

3.3.1 if+goto 循环

3.3.2 while 循环

3.3.3 do while 循环

3.2.4 for 循环

3.2.5 循环嵌套、循环与分支嵌套

3.2.6 break和continue 语句

3.3 循环结构

3.3.1 if+goto 循环

3.3 循环结构

3.3.1 if+goto 循环

- goto语句

无条件转向语句，形式：

goto 语句标号;

语句标号：语句

其中语句标号命名规则符合标识符命名规则

3.3 循环结构

3.3.1 if+goto 循环

➢goto语句

无条件转向语句，形式：

goto 标号;

标号：语句

其中语句标号命名规则符合标识符命名规则

例：

```
char ch;
cin >> ch;
if (ch == 'P')
      goto paris;
cout << ...
...
parsi:cout << "You've just arrived at Paris.\n";
```

3.3 循环结构

3.3.1 if+goto 循环

- if语句+goto语句构成循环

3.3 循环结构

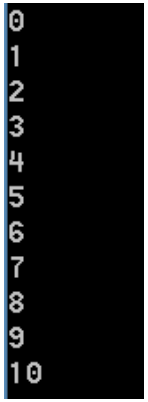
3.3.1 if+goto 循环

➢if语句+goto语句构成循环

例：

```
#include <iostream>
using namespace std;
int main()
{
	int i = 0;
	re: cout << i++ << endl;
	if (i <= 10)
		goto re;
	cout << endl;
	return 0;
}
```

运行结果：



3.3 循环结构

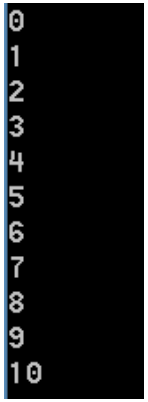
3.3.1 if+goto 循环

➢if语句+goto语句构成循环

例：

```
#include <iostream>
using namespace std;
int main()
{
	int i = 0;
	re: cout << i++ << endl;
	if (i <= 10)
		goto re;
	cout << endl;
	return 0;
}
```

运行结果：



结构化程序设计方法主张限制使用goto语句
滥用goto语句将使程序流程变得无规律，可读性差

3.3 循环结构

3.3.2 while 循环

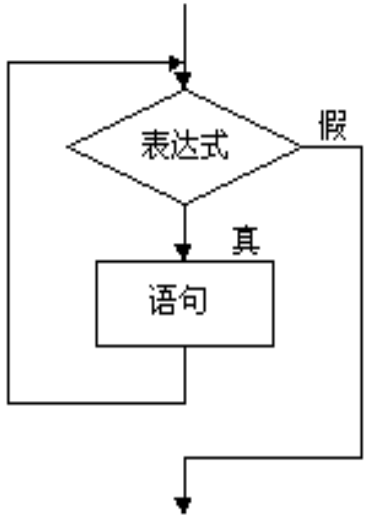
3.3 循环结构

3.3.2 while 循环

循环条件

```
while (表达式)
    语句
```

循环体



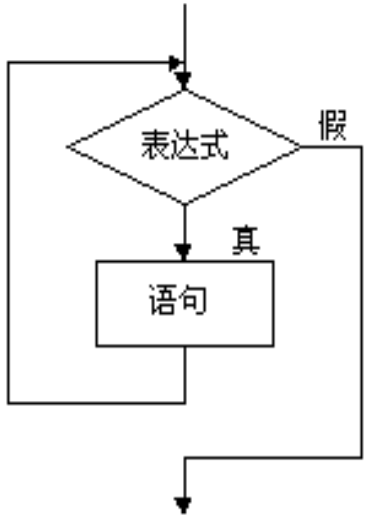
3.3 循环结构

3.3.2 while 循环

循环条件

```
while (表达式)
    语句
```

循环体



- ✓循环条件为真时执行循环体，否则退出循环
- ✓表达式可以是任意类型，按“非0为真0为假”判断真假
- ✓循环体语句从语法上要求是一条语句，如果包含一个以上的语句，要采用复合语句
- ✓循环体中应包含影响循环条件的操作，否则死循环
- ✓入口条件循环，即先判断后执行，如果循环条件一开始为假，则不会执行循环体

3.3 循环结构

3.3.2 while 循环

✓ 设计循环的几条指导原则：
指定循环终止的条件
在首次测试之前初始化条件
在条件被再次测试之前更新条件

3.3 循环结构

3.3.2 while 循环

✓ 设计循环的几条指导原则：
指定循环终止的条件
在首次测试之前初始化条件
在条件被再次测试之前更新条件

例：延时循环

```
long wait = 0;
while (wait < 10000)
        wait++;
```

3.3 循环结构

3.3.2 while 循环

✓ 设计循环的几条指导原则：
指定循环终止的条件
在首次测试之前初始化条件
在条件被再次测试之前更新条件

例：延时循环

```
long wait = 0;
while (wait < 10000)
        wait++;
```

```
#include <iostream>
#include <ctime>
using namespace std;
int main()
{
    cout << "Enter the delay time, in seonds:";
    float secs;
    cin >> secs;
    clock_t delay= secs * CLOCKS_PER_SEC;
    cout << "starting \a\n";
    clock_t start = clock();
    while (clock() - start < delay)
        ;       //空语句
    cout << "done \a\n";
    return 0;
}
```

3.3 循环结构

3.3.2 while 循环

例：求1+2+…+100的和。

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

3.3 循环结构

3.3.2 while 循环

例：求1+2+…+100的和。

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

```
while (i <= 100)
    sum = sum + i;
    i++;
```

```
while (i <= 100);
{
    sum = sum + i;
    i++;
}
```

3.3 循环结构

3.3.2 while 循环

例：求1+2+…+100的和。

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

```
while (i <= 100)
    sum = sum + i;
    i++;
```

错！少{}

```
while (i <= 100);
{
    sum = sum + i;
    i++;
}
```

错！多;

3.3 循环结构

3.3.2 while 循环

例：求1+2+…+100的和。

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

等价于：

```
while (i <= 100)
    sum += i++;
```

```
while (i <= 100)
    sum = sum + i;
    i++;
```

错！少{}

```
while (i <= 100);
{
    sum = sum + i;
    i++;
}
```

错！多;

3.3 循环结构

3.3.2 while 循环

例：用公式$\frac{\pi}{4} \approx 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$求$\pi$的值，
直到最后一项的绝对值小于10^{-7}为止。

3.3 循环结构

3.3.2 while 循环

例：用公式$\frac{\pi}{4} \approx 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$求$\pi$的值，直到最后一项的绝对值小于$10^{-7}$为止。

```
和及通项赋初值
while(通项尚未足够小)
{
    和=和+通项;
    通项=f(通项);
}
```

3.3 循环结构

3.3.2 while 循环

例：用公式$\frac{\pi}{4} \approx 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$求$\pi$的值，直到最后一项的绝对值小于$10^{-7}$为止。

```
和及通项赋初值
while(通项尚未足够小)
{
    和=和+通项;
    通项=f(通项);
}
```

```
#include <iostream>
#include <cmath>  //fabs()函数所需头文件
using namespace std;
int main()
{
        int s = 1;
        double n = 1, t = 1, pi = 0;
        while (fabs(t) >= 1e-7)
        {
                pi = pi + t;
                n = n + 2;
                s = -s;
                t = s / n;
        }
        pi = pi * 4;
        cout << "pi=" << pi << endl;
        return 0;
}
```

3.3 循环结构

3.3.3 do while 循环

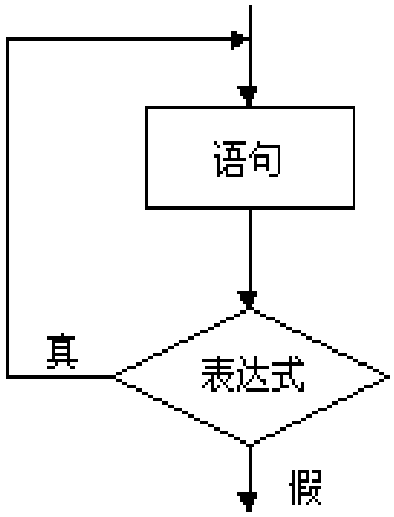
3.3 循环结构

3.3.3 do while 循环

```
do
    语句
while (表达式);
```

循环体

循环条件



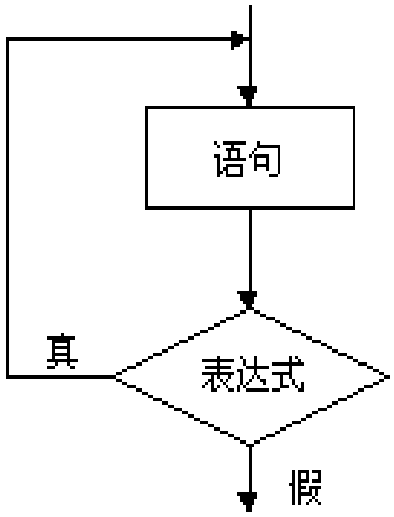
3.3 循环结构

3.3.3 do while 循环

```
do
    语句
while (表达式);
```

循环体

循环条件



- ✓ 先执行循环体，然后判定测试表达式，如果表达式为真（非0）进入新一轮的执行和测试，如果为假（0），则循环终止
- ✓ 循环体语句是一条语句或用{}括起来的复合语句
- ✓ 循环体中应包含影响循环条件的操作，否则死循环
- ✓ 出口条件循环，循环体至少执行一次

3.3 循环结构

3.3.3 do while 循环

例：

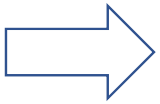
```
int n;
do
{
      cin >> n;
} while (n != 7);
cout << "Yes, 7 is my favorite.\n";
```

3.3 循环结构

3.3.3 do while 循环

例：

```
int n;
do
{
      cin >> n;
} while (n != 7);
cout << "Yes, 7 is my favorite.\n";
```



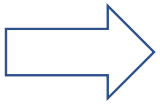
```
int n;
cin >> n;
while (n != 7)
      cin >> n;
cout << "Yes, 7 is my favorite.\n";
```

3.3 循环结构

3.3.3 do while 循环

例：

```
int n;
do
{
      cin >> n;
} while (n != 7);
cout << "Yes, 7 is my favorite.\n";
```



```
int n;
cin >> n;
while (n != 7)
      cin >> n;
cout << "Yes, 7 is my favorite.\n";
```

```
int n;
while (cin >> n, n != 7);
cout << "Yes, 7 is my favorite.\n";
```

3.3 循环结构

3.3.3 do while 循环

例：求1+2+…+100的和。

```
int sum = 0, i = 1;
do
{
      sum = sum + i;
      i++;
} while (i <= 100);
```

3.3 循环结构

3.3.4 for 循环

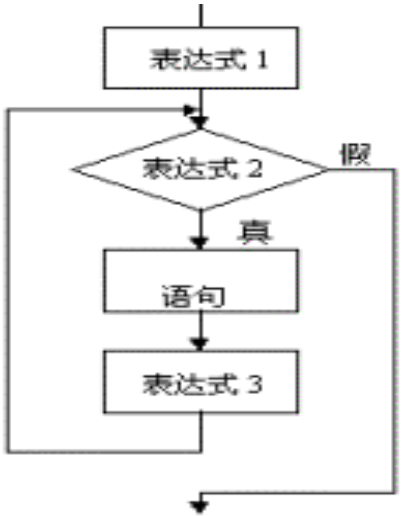
3.3 循环结构

3.3.4 for 循环

循环条件

```
for (表达式1;表达式2;表达式3)
    语句
```

循环体



✓ 执行流程：

（1）求解表达式1

（2）求解表达式2，若为真（非0），则执行循环体；若为假（0），则结束循环

（3）求解表达式3

（4）转回（2）继续执行

✓入口条件循环，循环体有可能一次也不执行

✓循环体如果包含一条以上的语句，要用复合语句

3.3 循环结构

3.3.4 for 循环

（1）基本形式

for (初值表达式;测试表达式;更新表达式)
    语句

例：

```
int sum = 0, i;
for (i = 1; i <= 100; i++)
      sum = sum + i;
```

3.3 循环结构

3.3.4 for 循环

（2）扩展使用

✓for语句基本形式中的3个表达式任何一个都可以省略

3.3 循环结构

3.3.4 for 循环

（2）扩展使用

✓for语句基本形式中的3个表达式任何一个都可以省略

✓如果省略表达式1，应在for语句前给循环变量赋初值

例：

```
int sum = 0, i = 1;
for (; i <= 100; i++)
      sum = sum + i;
```

3.3 循环结构

3.3.4 for 循环

（2）扩展使用

✓for语句基本形式中的3个表达式任何一个都可以省略

✓如果省略表达式1，应在for语句前给循环变量赋初值

例：

```
int sum = 0, i = 1;
for (; i <= 100; i++)
      sum = sum + i;
```

✓ 如果省略表达式2，循环条件将为真，循环无休止地运行下去，在循环体内必须有语句负责退出循环

例：

```
int sum = 0, i;
for (i = 1; ; i++)
      sum = sum + i;
```

死循环!

```
int sum = 0, i;
for (i = 1; ; i++)
{
      if (i > 100)
            break;   //结束循环
      sum = sum + i;
}
```

3.3 循环结构

3.3.4 for 循环

✓ 如果省略表达式3，应在条件部分或者循环体改变循环变量的值

例：

```
int sum = 0, i;
for (i = 1; i <= 100;)
{
        sum = sum + i;
        i++;
}
```

3.3 循环结构

3.3.4 for 循环

✓ 如果省略表达式3，应在条件部分或者循环体改变循环变量的值

例：

```
int sum = 0, i;
for (i = 1; i <= 100;)
{
        sum = sum + i;
        i++;
}
```

✓ 如果表达式1和3省略，完全等价于while语句形式

例：

```
int sum = 0, i=1;
for (; i <= 100;)
{
        sum = sum + i;
        i++;
}
```

```
int sum = 0, i=1;
while(i <= 100)
{
        sum = sum + i;
        i++;
}
```

3.3 循环结构

3.3.4 for 循环

✓ 三个表达式可全省，但分号不可省

例：

```
for (;;)
    语句
```

相当于

```
while(1)
    语句
```

3.3 循环结构

3.3.4 for 循环

✓ 三个表达式可全省，但分号不可省

例：

```
for (;;)
    语句
```

相当于

```
while(1)
    语句
```

✓表达式1和表达式3可以是一个简单的表达式，也可以是逗号表达式

例：

```
int sum, i;
for (i = 1, sum = 0; i <= 100; i++)
    sum = sum + i;
```

```
int sum, i;
for (i = 1, sum = 0; i <= 100; sum = sum + i, i++)
    ; //空语句
```

3.3 循环结构

3.3.4 for 循环

- ✓ for循环中可使用初始化语句声明一个局部变量

例：

```
for (int i = 0; i < 5; i++)
      cout << "C++ knows loops.\n";
cout << i << endl;   //报错，i只存在于for语句中
```

3.3 循环结构

3.3.4 for 循环

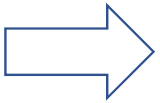
（3）for 与while语句的互换性

3.3 循环结构

3.3.4 for 循环

（3）for 与while语句的互换性

```
for (表达式1;表达式2;表达式3)
    语句
```



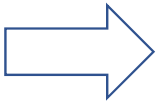
```
表达式1;
while（表达式2）
{
    语句
    表达式3;
}
```

3.3 循环结构

3.3.4 for 循环

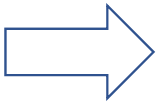
（3）for 与while语句的互换性

```
for (表达式1;表达式2;表达式3)
    语句
```



```
表达式1;
while（表达式2）
{
    语句
    表达式3;
}
```

```
while (表达式)
    语句
```



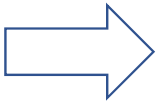
```
for (;表达式2;)
    语句
```

3.3 循环结构

3.3.4 for 循环

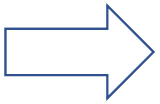
（3）for 与while语句的互换性

```
for (表达式1;表达式2;表达式3)
    语句
```



```
表达式1;
while（表达式2）
{
    语句
    表达式3;
}
```

```
while (表达式)
    语句
```



```
for (;表达式2;)
    语句
```

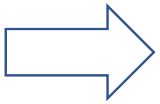
区别：

- ✓ for循环中可省略循环条件，将认为条件为真
- ✓ for循环中可使用初始化语句声明一个局部变量
- ✓ 循环次数已知的情况下更适合用for语句

```
int I = 0;
for (;;)
{
	I++;
	...
	if (30 == I)break;
}
```

```
int I = 0;
for (;; I++)
{
	if (30 == I)break;
	...
}
```

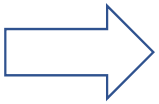
```
int I = 0;
for (;;)
{
      I++;
      ...
      if (30 == I)break;
}
```



```
int I = 0;
do
{
      I++;
      ...
} while (30 > I);
```

更好！

```
int I = 0;
for (;; I++)
{
      if (30 == I)break;
      ...
}
```



```
int I = 0;
while (I < 30)
{
      ...
      I++;
}
```

更好！

要编写清晰、容易理解的代码！

3.3 循环结构

3.3.5 循环嵌套、循环分支嵌套

3.3 循环结构

3.3.5 循环嵌套、循环分支嵌套

各种循环语句、分支语句之间可以相互嵌套。

例：

```
for (i = 1; i <= 3; i++)
        for (j = 1; j <= 3; j++)
                cout << i * j << ' ';
```

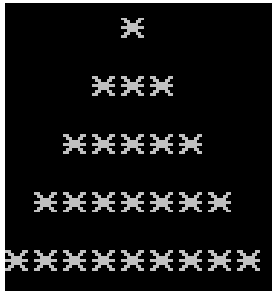
```
1 2 3 2 4 6 3 6 9
```

外层循环执行一次,内层循环执行一遍

3.3 循环结构

3.3.5 循环嵌套、循环分支嵌套

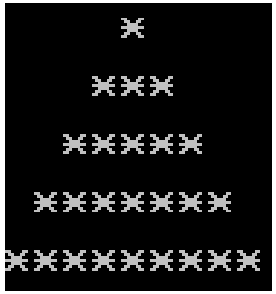
例：输出如图示图形



3.3 循环结构

3.3.5 循环嵌套、循环分支嵌套

例：输出如图示图形

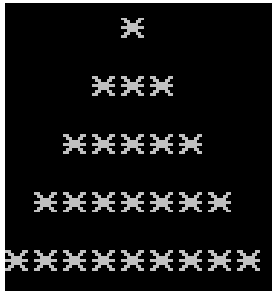


```
for (i = 0; i < 5; i++)          //控制行数
{
    for (j = 0; j < 4-i; j++)    //控制每行起始位置
        cout << ' ';
    for (k = 0; k < 2 * i + 1; k++) //控制列数
        cout << '*';
    cout << endl;
}
```

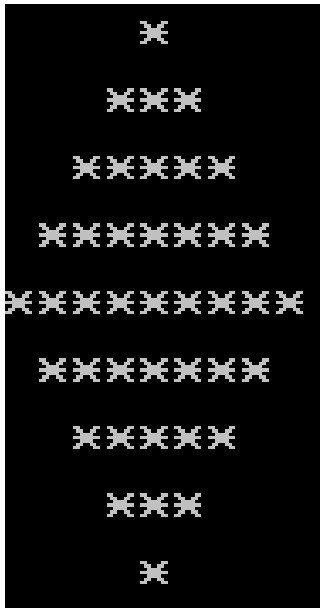
3.3 循环结构

3.3.5 循环嵌套、循环分支嵌套

例：输出如图示图形



```
for (i = 0; i < 5; i++)          //控制行数
{
    for (j = 0; j < 4-i; j++)    //控制每行起始位置
        cout << ' ';
    for (k = 0; k < 2 * i + 1; k++) //控制列数
        cout << '*';
    cout << endl;
}
```



如何实现？

3.3 循环结构

3.3.5 循环嵌套、循环分支嵌套

例：将输入的以'#'结束的一串字符中的字母进行大小写转换并输出。

```
char ch;
while ((ch = getchar()) != '#')
{
      if (ch >= 'a' && ch <= 'z')
            ch = ch - 32;
      else if (ch >= 'A' && ch <= 'Z')
            ch = ch + 32;
      cout << ch;
}
```

3.3 循环结构

3.3.5 循环嵌套、循环分支嵌套

例：将输入的以'#'结束的一串字符中的字母进行大小写转换并输出。

如果写成if会怎么样?

```
char ch;
while ((ch = getchar()) != '#')
{
    if (ch >= 'a' && ch <= 'z')
        ch = ch - 32;
    else if (ch >= 'A' && ch <= 'Z')
        ch = ch + 32;
    cout << ch;
}
```

3.3 循环结构

3.3.6 break和continue语句

break语句：可以用于switch语句或循环语句中，使程序跳到switch或循环后面的语句处执行。
continue：用于循环语句中，让程序跳过循环体中continue后面的语句，并开始新一轮循环。

3.3 循环结构

3.3.6 break和continue语句

break语句：可以用于switch语句或循环语句中，使程序跳到switch或循环后面的语句处执行。

continue：用于循环语句中，让程序跳过循环体中continue后面的语句，并开始新一轮循环。

例：

```
while (cin.get(ch))
{
      statement1;
      if (ch == '\n')
            continue;
      statement2;
}
statements3;
```

```
while (cin.get(ch))
{
      statement1;
      if (ch == '\n')
            break;
      statement2;
}
statements3;
```

3.3 循环结构

3.3.6 break和continue语句

break语句：可以用于switch语句或循环语句中，使程序跳到switch或循环后面的语句处执行。

continue：用于循环语句中，让程序跳过循环体中continue后面的语句，并开始新一轮循环。

例：

```
while (cin.get(ch))
{
      statement1;
      if (ch == '\n')
            continue;
      statement2;
}
statements3;
```

```
while (cin.get(ch))
{
      statement1;
      if (ch == '\n')
            break;
      statement2;
}
statements3;
```

注意：在for循环中，continue语句使程序直接跳到表达式3处，然后再跳到表达式2。

3.3 循环结构

3.3.6 break和continue语句

例：

```
char ch;
int spaces=0;
while (cin.get(ch))
{
    if (ch == '.')      //如果是'.',退出循环
        break;
    if (ch != ' ')
        continue;       //如果不是' '，跳过spaces++;语句
    spaces++;
}
cout << spaces << " spaces\n";
```

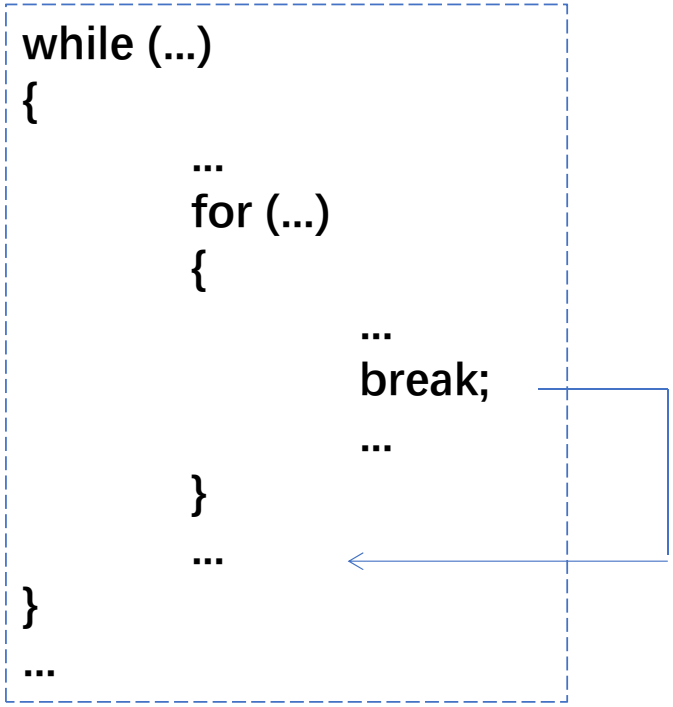
3.3 循环结构

3.3.6 break和continue语句

✓break出现在多层循环中，仅跳出本层循环

例：

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

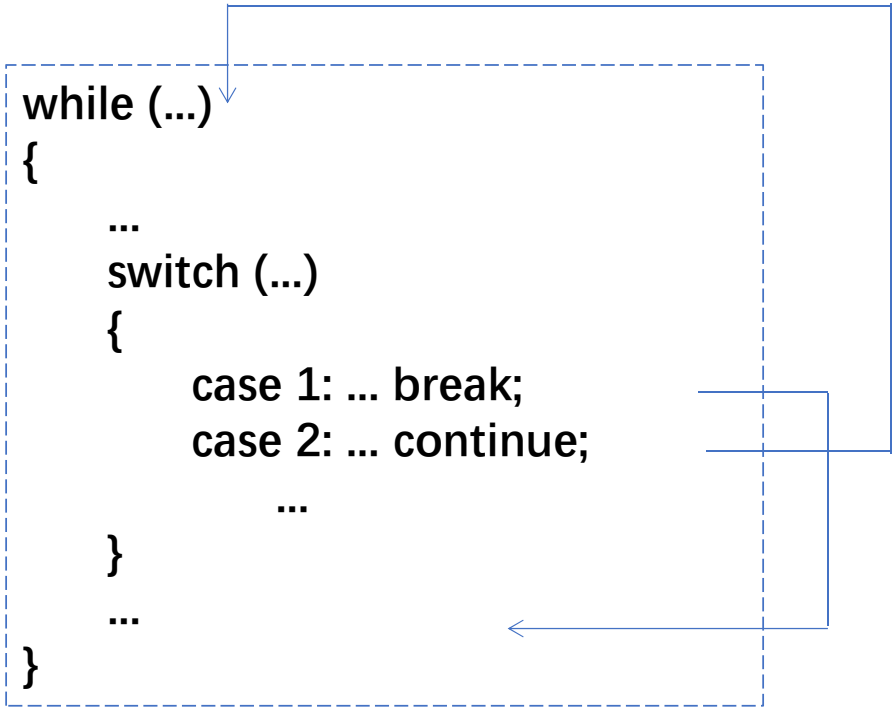


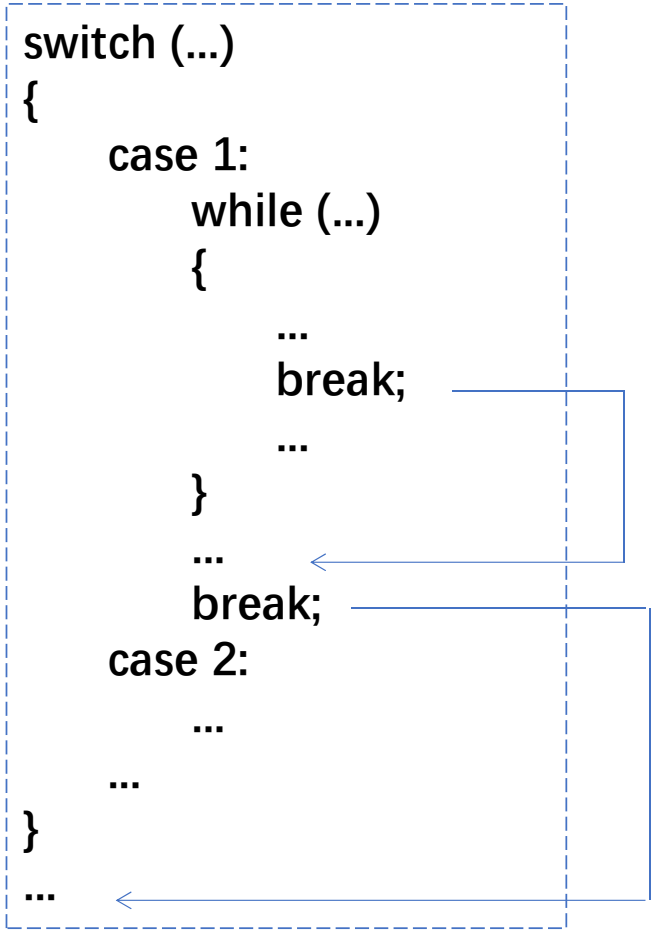
3.3 循环结构

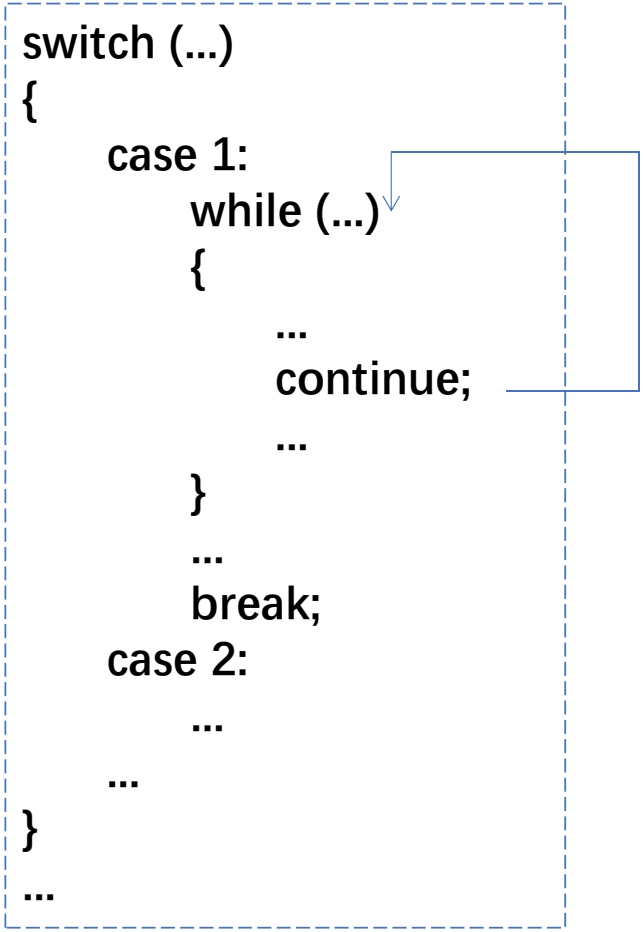
3.3.6 break和continue语句

- ✓ 当循环语句和switch语句嵌套时,break跳转到其所在的循环或switch语句后，continue仅对循环有效

例：




switch (…)
{
case 1:
while (…)
{
…
break;
…
}
…
break;
case 2:
…
…
}
…


switch (…)
{
case 1:
while (…)
{
…
continue;
…
}
…
break;
case 2:
…
…
}
…

例：求100～200之间的素数，
并以每行10个显示。

素数判别算法：
若m不能被2~$\sqrt{m}$ 中任一整数整除，则m为素数

```
k = int(sqrt(m));
for (i = 2; i <= k; i++)
      if (m % i == 0)
            break;
if(...)
      ...
```

例：求100～200之间的素数，并以每行10个显示。

素数判别算法：

若m不能被2~$\sqrt{m}$ 中任一整数整除，则m为素数

```
k = int(sqrt(m));
for (i = 2; i <= k; i++)
    if (m % i == 0)
        break;
if(...)
    ...
```

```
#include <iostream>
#include <iomanip>  //setw()函数需要的头文件
using namespace std;
int main()
{
    int m, k, i, n = 0;
    bool prime;
    for (m = 101; m <= 200; m = m + 2)
    {
        prime = true;
        k = int(sqrt(m));
        for (i = 2; i <= k; i++)
            if (m % i == 0)
            {
                prime = false;
                break;
            }
        if (prime)
        {
            cout << setw(5) << m;
            n = n + 1;
            if (n % 10 == 0) cout << endl;
        }
    }
    return 0;
}
```

例：求100～200之间的素数，并以每行10个显示。

素数判别算法：

若m不能被2~$\sqrt{m}$ 中任一整数整除，则m为素数

```
k = int(sqrt(m));
for (i = 2; i <= k; i++)
    if (m % i == 0)
        break;
if(...)
    ...
```

```
#include <iostream>
#include <iomanip>  //setw()函数需要的头文件
using namespace std;
int main()
{
    int m, k, i, n = 0;
    bool prime;
    for (m = 101; m <= 200; m = m + 2)
    {
        prime = true;
        k = int(sqrt(m));
        for (i = 2; i <= k; i++)
            if (m % i == 0)
            {
                prime = false;
                break;
            }
        if (prime)
        {
            cout << setw(5) << m;
            n = n + 1;
            if (n % 10 == 0) cout << endl;
        }
    }
    return 0;
}
```

不用prime变量怎么实现？

能放在复合语句外吗？