《计算概论A》课程 程序设计部分

C++语言的基本成分 (2)

李　戈

北京大学 信息科学技术学院 软件研究所

2010年10月22日

北京大学

常 量

- 常量
  - 在程序运行过程中，其值保持不变的量
- 字面常量
  - －1，0，123，4.6，－1.23;
- 符号常量
  - 用一个标识符代表一个常量的，称为符号常量

```
#include <iostream>
int main( )
{
        const float PI=3.14159f;
        float  r, area;
        cin>>r;
        area = r * r * PI;
        cout<<“area = ”<<area;
        return 0;
}
```

常量有类型吗?

- 整型常量的后缀
  - n = 10000L; //长整型常量
  - m = -0x88abL; //长整型十六进制常量
  - k = 10000U; //无符号整型常量
  - i = 07777LU; //无符号长整型八进制常量
- 浮点型常量的后缀
  - x = 3.1415F //单精度浮点型常量
  - y = 3.1415L //长双精度浮点型常量
- 说明:
  - 浮点型常量默认为double 型;
  - U, L, F均可以小写;

C++语言中的运算符

- C++语言的运算符范围很宽
  - 求字节数运算符：sizeof
  - 下标运算符　　[ ]
  - 赋值运算符　　=
  - 算术运算符　　+　－　*　／　%
  - 关系运算符　　< > == >= <= !=
  - 逻辑运算符　　！ && ||
  - 条件运算符　　？ ：
  - 逗号运算符　　，
  - 位运算符　　>> ~ | ^ &
  - 指针运算符　　*，&
  - 强制类型转换运算符：(类型)
  - 分量运算符　　. →

赋值表达式

- “=” 赋值运算符
  - 给赋值号左边的变量赋予数值
- 在变量定义的同时可以为变量赋初值。如:
  - **int a=3;**

    相当于:

    int a; a=3;
  - **int a, b, c = 5**

    表示只给 c 赋初值。相当于

    int a, b, c;

    c = 5;
  - **int a = b = c = 5; (Is this right?)**

赋值表达式

- 要点一：两边类型不同
  - 若 = 两边的类型不一致，赋值时要进行类型转换。
  - 不管 = 右边的操作数是什么类型，都要转换为 = 左边变量类型
  - 例如:
    - **int i = 3.56; i 得到的是3**
    - **float f = 23; f 中存储的是23.00000**
    - **float f = double 截取前面7位有效数字**

      例如：**double d = 1234.5678987654321; float f = d;**

      则，**f中存放的是1234.567**

自动类型转换（1）

- 如果=右边的操作数具有较高级别的类型，则在类型转换时，进行截断取舍，可能会损失精度!
  - 例如:
    - int i = 3.56; i 得到的是3
    - float f = 1234.5678987654321

      截取前面7位有效数字

      f中存放的是1234.567

```
(低)---------------------------------------------------------->(高)
char < short < int < unsigned < long < unsigned long < float < double
```

自动类型转换（2）

- 如果=右边的操作数类型级别较低，则在类型转换时，采用补齐方式，不会损失精度。
- 例如：
  - float f = 23; f 中存储的是23.00000
  - int i = 10/3;

    float f = i;

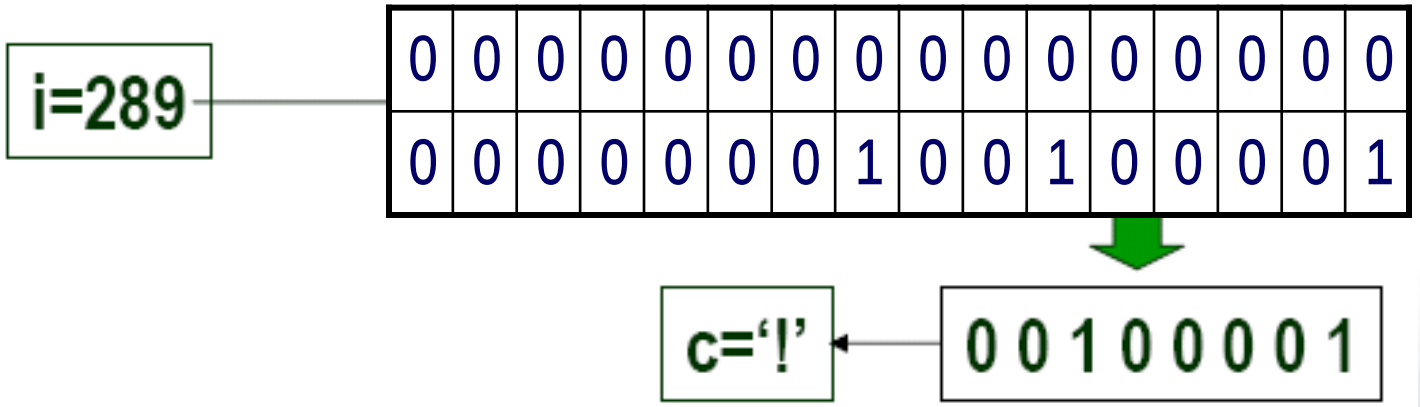
    i 的值为3，而f 的值为3.0，精度没有损失。

```
(低)-------------------------------------------------------->(高)
char < short < int < unsigned < long < unsigned long < float < double
```

赋值表达式

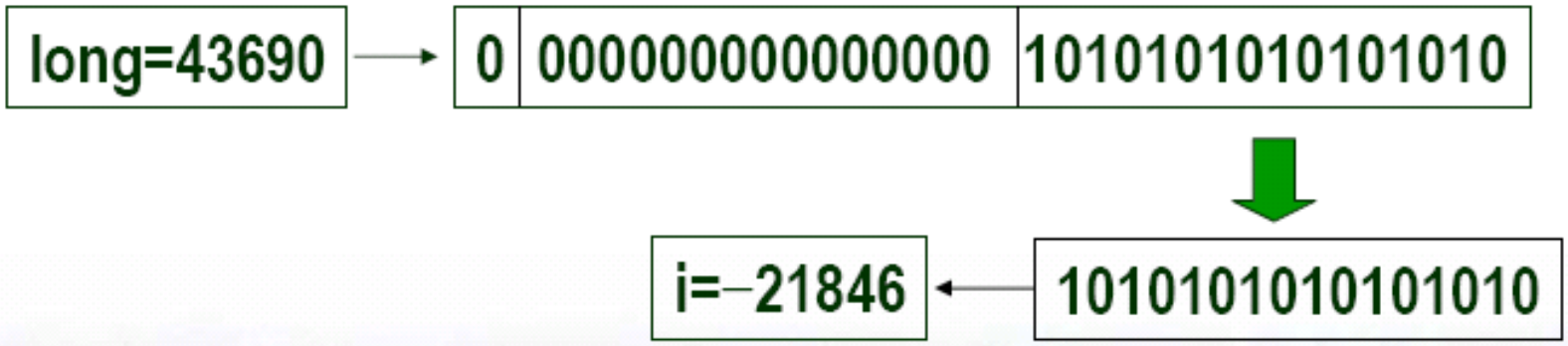
- 要点二：长数赋给短数
  - 截取长数的低n位送给短数
    - 例如：

```
char c = '!';
int i = 289;   c = i;
```



赋值表达式

- 举例:
  - short = long
    - 截取长整型数的低16位送给short字符。如果最高位为0，则得到负数，否则得到正数;



赋值表达式

- 要点三：短数赋给长数
  - 最好处理的情况！是什么就是什么！
  - short int a = -1;　　int b = a;

计算机的处理过程：

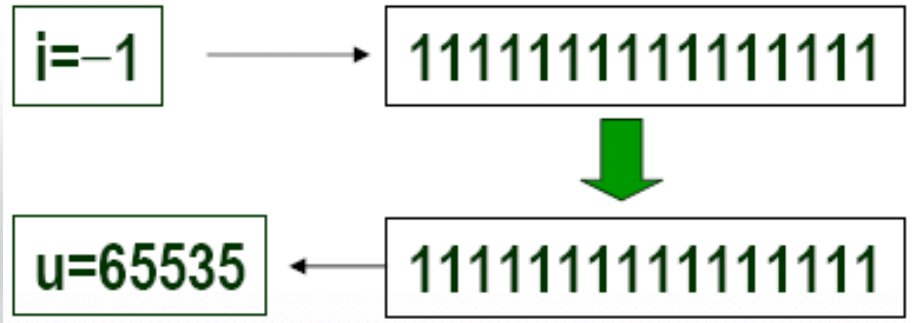
- 若short 型数为无符号数：
  - short 型16位到long型低16位，long型高16位补0;
- 若 short型数为有符号数：
  - short型16位到 long型低16位;
  - 若short型最高位为0，则long型高16位补0;
  - 若short型最高位为1，则long型高16位补1;

赋值表达式

- 要点四：符号位的赋值处理
  - 也很好处理的情况！是什么就是什么！
  - 直接赋值，我才不管符号位是什么！

例如：

- **int = unsigned int**
  - 直接赋值，数字当作符号位。
- **unsigned = int 或long**
  - 直接赋值，符号位当作数字。



赋值运算总结

- 当：两边类型不同
  - 自动完成类型转换，多退少补!
- 当：长数赋给短数
  - 截取长数的低位送给短数!
- 当：短数赋给长数
  - 是什么就是什么!
- 当：符号位的赋值处理
  - 直接赋值，不管符号位是什么!

强制进行的类型转换

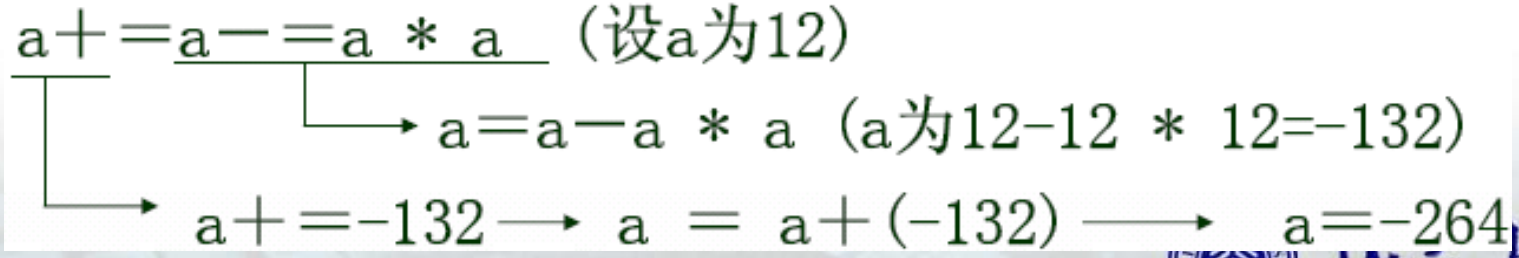
- 强制类型转换的形式
  - (类型名)（表达式）
- 举例
  - 如:(double)a 表示将a转换成double类型
  - (int)(x+y)表示将x+y的值转换成double类型
  - (float)(5/3)表示将5/3的值转换成float类型
- 注意
  - 在强制类型转换后，被转换的量的类型并没有发生变化

赋值表达式

- 复合的赋值运算
  - 在赋值符号前加上其它运算符号则构成复合赋值运算;
  - 例如:
    - a + = 3; 等价于a = a + 3;
    - x * = y + 8; 等价于x = x * (y + 8);
    - x % = 3; 等价于x = x % 3;

赋值表达式

- 连续的赋值运算
  - 自右而左的结合顺序
    - a = (b = 5);　　//a, b均为5
    - a = b = c = 5;　　//a, b, c均为5
      - int a=b=c=5;　　//编译错!
  - a = (b = 4) + (c = 6); a为10,b为4,c为6
- 举例:



算术运算符和算术表达式

- 算术运算符和算术表达式
  - 基本的算术运算 + 、 － 、 *、 ／ 、 %
    - % 是模运算，即，求余运算，必须是整数
      - 如7%4 = 3
- 注意:
  - 整数运算，结果仍为整数
    - 5/3的结果仍是整数，小数部分被忽略
  - 实数与double型运算，结果为double型
    - 5.3/3或5/3.0的结果为double型
  - 舍入的方向随编译器的不同而不同

算术表达式

- 算术运算符的优先级

  ( )

  * / %

  + −

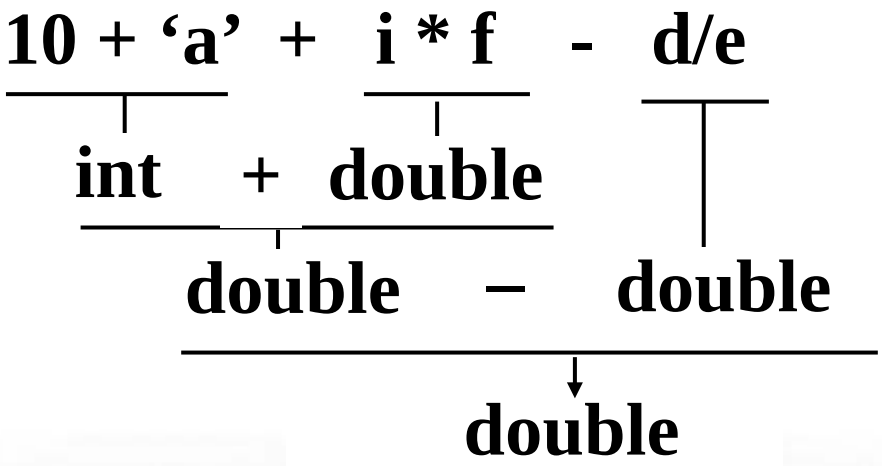
  - 在同一级别中，采取由左至右的结合方向
    - 如：a – b + c 相当于(a – b) + c
    - 如：a % b * c / d 相当于(((a % b)*c))/d
  - 当数据类型非常复杂时
    - 如：10 + ‘a’ + i*f – d/e

      其中 int i; float f; double d; long e;

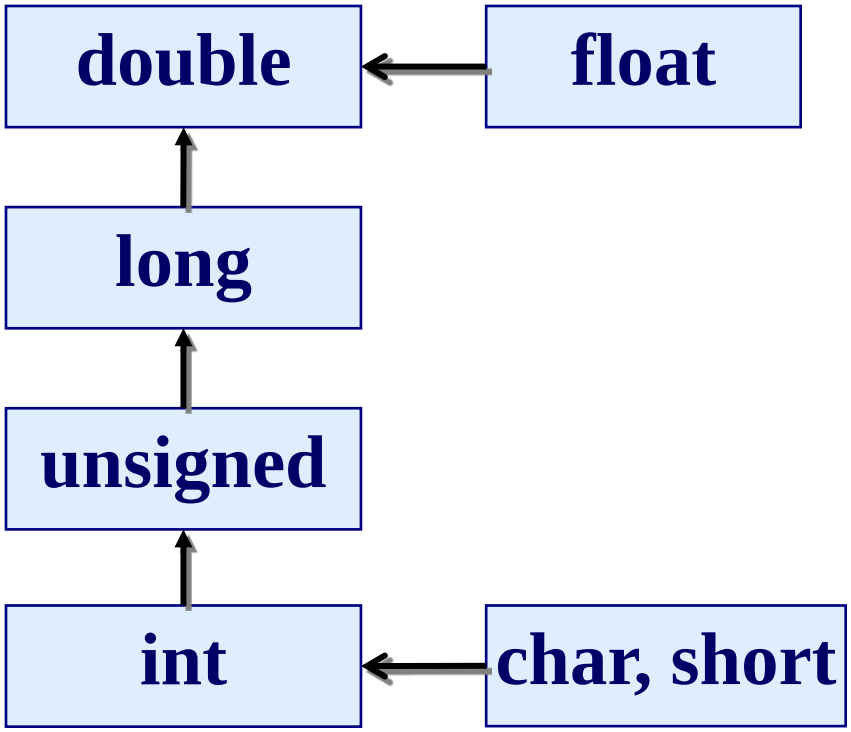
算术表达式

- 算术运算中的类型转换
  - 如： 10 + ‘a’ + i*f – d/e

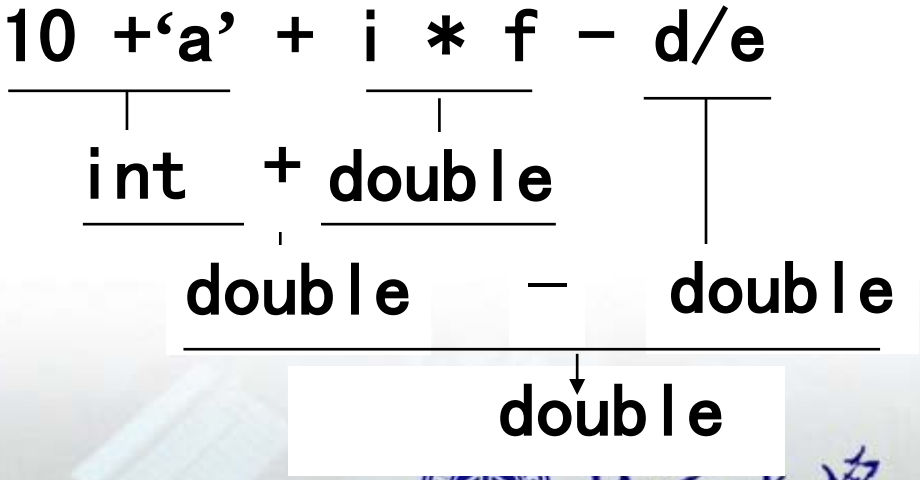
    其中 int i;　float f;　double d;　long e;



各类数值间的混合运算



- 数据转换向着表达能力更强的方向
- 转换总是逐个运算符进行的



算术表达式

- 自增、自减运算符：使变量的值加1或减1
  - ++i，　--i
    - 在使用i之前，先将i的值加（减）1
  - i++，　i--
    - 在使用i之后，再将i的值加（减）1
- 例如：i的值为3，则
  - j=++i;
  - j=i++;
  - cout<<++i;
  - cout<<i++;

自增、自减运算符

- 举例:
  - 设i的值为3; 则

```
cout<<-i++<<endl;
cout<<-++i<<endl;
cout<<(-i)++<<endl;
cout<<++i++<<endl;
```

- 注意:
  - ++和--只能用于变量

自增、自减运算符

```
#include<iostream>
using namespace std;
int main( )
{
   int a1 = 2, b1 = 0, a2 = 2, b2 = 0, c = 0, d = 0;
   cout<<b1<<" "<<b1++<<" "<<endl;
   cout<<++b2<<" "<<b2++<<" "<<endl;
   cout<<a2<<" "<<(a2++)+(++a2)<<" "<<endl;
   cout<<(c=a1++)+(d=a1)<<endl;
   cout<<a1<<" "<<c<<" "<<d<<endl;
   return 0;
}
```

关系运算符的意义

- C++语言提供6种关系运算符:

① < (小于)
② <= (小于或等于)
③ > (大于)
④ >= (大于或等于)

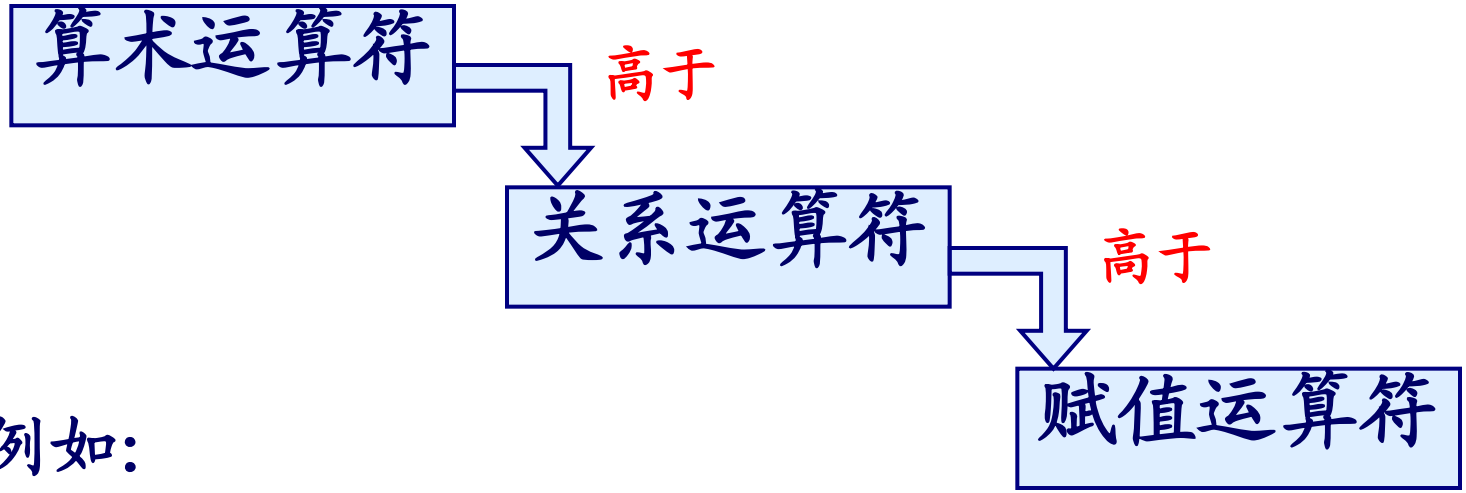
优先级相同 — 高

⑤ == (等于)
⑥ != (不等于)

优先级相同 — 低

运算符的优先级

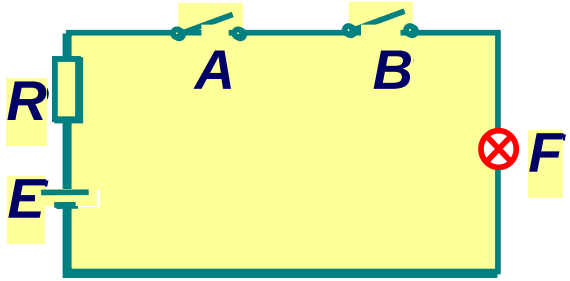
- 关系表达式

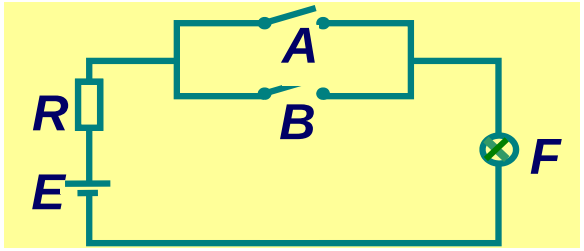


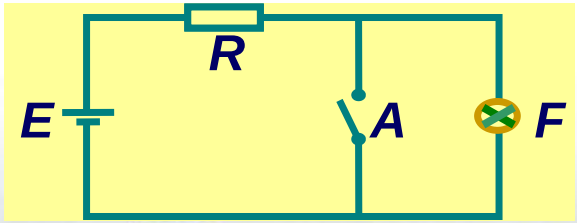
- 例如:
  - 1 + 2 % 3 * 4 > 5 / 6 – 7
  - 1 + 2 % (3 * 4) > (5 / 6 – 7) = = 8
  - X = 1 + 2 % 3 * 4 > 5 / 6 – 7 = = 8
  - 1>2= =3>4

逻辑运算符

- C++语言提供三种逻辑运算符:
  - 逻辑与 &&
  - 逻辑或 ||
  - 逻辑非 !
- 逻辑运算举例如下:
  - 若A、B为真，则F=A&&B为真。
  - 若A、B之一为真，则F=A||B为真。
  - 若A为真，则!A为假。







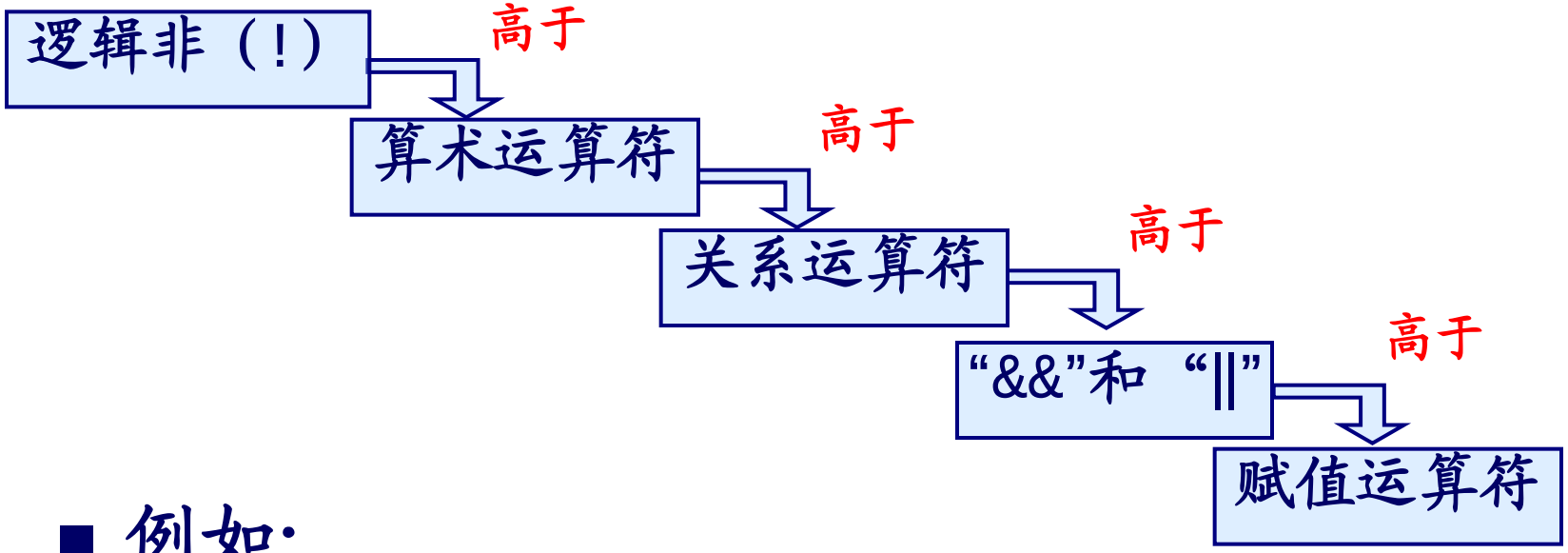
北京大学

逻辑运算符优先级

- 逻辑表达式
  - 一个逻辑表达式中若包含多个逻辑运算符，则按以下的优先次序:
    - ! (非)→&& (与)→|| (或)，即“!”优先级最高。
  - 例如:

```
!a&&b||c;
! a && b || x > y && c;
! a && b || x > y && c + 1;
```

混合运算的优先级



- 例如:
  - a＞b && x＞y可写成(a＞b) && (x＞y)
  - a == b || x == y可写成(a ==b) || (x == y)
  - ! a || a＞b可写成 (! a) || (a＞b)

逻辑表达式的值

- 逻辑表达式的值
  - 计算结果时以数值1代表“真”，以0代表“假”；
  - 判断一个量是否为“真”时，以0代表“假”，以非0代表“真”。即将一个非零的数值认作为“真”。
    - 若 a = 4，则 !a 的值为 0。
    - 若 a = 4，b = 5，则 a && b 的值为1。
    - 若 a = 4，b = 5，则 a || b 的值为1。
    - 若 a = 4，b = 5，则 !a || b 的值为1。
    - 表达式 4 && 0 || 2 的值为1。

示例

- 区分下面的表达式中算术运算量，关系运算量和逻辑运算量
  - 5>3 && 2|| 8<4-!0
  - 5>3是两个数值间的比较，结果为 1
  - 1 && 2是两个非 0 值（逻辑量）间的运算，结果为 1
  - 1||8<4-!0, 根据优先级,先计算! 0 ，结果为 1
  - 1||8<4-1 > 1||8<3>1||0 结果为 1

运算对象的扩展

- 逻辑运算符两侧的运算对象可以是任何类型的数据
  - 如：字符型、实型或指针型等。
  - 系统最终以0和非0来判定它们属于“真”或“假”。例如

    'c' && 'd'

    - ‘c’和‘d’的ASCII值都不为0，按“真”处理。
- 思考题:
  - 5＞3 + ‘a’ % 6 = = ‘b’ && 8＜4 - ! ‘c’

逻辑运算的取舍

- 逻辑表达式求解中，并不总是执行所有的运算
  - 只有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！
    - 对于表达式 a && b && c
      - 只有a为真(非0)时，才需要判别b的值，
      - 只有a和b都为真的情况下才需要判别c的值。
    - 对于表达式 a || b || c
      - 只要a为真(非0)，就不必判断b和c；只有a为假，才判别b；a和b都为假才判别c。

逻辑运算的取舍举例

```
#include<iostream>
using namespace std;
int main( )
{
   int i = 0, a = 1, b = 2, c = 3;
   i = ((++a)||(b=5)||(c=6));
   cout<<i <<" "<< a<<" "<<b<<" "<<c<<endl;
   return 0;
}
```

应用示例

- 问题：要判别某一年year是否闰年。闰年的条件是符合下面二者之一：
  - 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。
- 求解：
  - 可以用如下逻辑表达式来判别year是否为闰年：

```
( year % 4 == 0 && year % 100 ! = 0 ) || year % 400 == 0
```

  - 也可以用如下逻辑表达式判别year是否非闰年：

```
( year % 4 ! = 0 ) || ( year % 100 == 0 && year % 400 ! = 0)
```

逗号运算和逗号表达式

- 用逗号将两个表达式连起来,叫逗号表达式
  - 如3+5，6+8
- 先求表达式1，再求表达式2， 整个表达式的值为表达式2的值
  - a = 3 * 5, a * 4; 先求a的值为15，再求表达式的值为60。
- 逗号表达式的形式可以扩展:
  - 表达式1，表达式2，表达式3，…… 表达式n
- 下式中是否相同？
  - x = (a = 3, 6 * 3);
  - x = a = 3 , 6 * 3; x = 3

位运算

- 位运算
  - 所谓位运算是指进行二进制位的运算。
- C++语言中的位运算符
  - 按位与（&）　　双目运算符
  - 按位或（|）　　双目运算符
  - 按位异或（^）　　双目运算符
  - 取反（~）　　单目运算符
  - 左移（<<）　　单目运算符
  - 右移（>>）　　单目运算符

位运算符（1）

- “按位与”运算符(&)
  - 参加运算的两个数据，按二进位进行“与”运算。如果两个相应的二进位都为1，则该位的结果值为1，否则为0。即

    0＆0＝0；　0＆1＝0；　1＆0＝0；　1＆1＝1;
- 例如：对于表达式：a＝3＆5，有：

```
    3 = 00000011
(&) 5 = 00000101
--------------
    00000001
```

因此，3&5的值得1

位运算符（2）

- “按位或”运算符（|）
  - 两个相应的二进位中只要有一个为1，该位的结果值为1。
  - 即 0|0 = 0； 0|1 = 1； 1|0 = 1； 1|1 = 1。
- 例如：对于表达式：a = 3|5，有：

```
    3 = 00000011
(|) 5 = 00000101
    ----------
    00000111
```

因此，3|5的值得7

位运算符（3）

- “异或”运算符（∧）
  - 异或运算符∧也称XOR运算符。它的规则是若参加运算的两个二进位同号，则结果为0(假)；异号则为1(真)。
  - 即0 ∧ 0 = 0； 0 ∧ 1 = 1； 1 ∧ 0 = 1； 1 ∧1 = 0；

例如：

```
        00111001   (十进制数57，八进制数071)
(∧)     00101010   (十进制数42，八进制数052)
        --------
        00010011   (十进制数19，八进制数023)
```

即071∧052，结果为023(八进制数)。

位运算符（4）

- 取反运算符“~”是一个单目(元)运算符，用来对一个二进制数按位取反，
  - 即将0变1，1变0。
  - 例如：

000000000010101

(~) ↓

111111111101010

因此，~(025)$_8$的值为(177752)$_8$

位运算符（5）

- 左移运算符(<<)
  - 用来将一个数的各二进位全部左移若干位。
  - 高位左移后溢出，舍弃不起作用。
  - 若a=15，即二进制数00001111，左移2位得00111100，即十进制数60

    a = a<<2
  - 左移1位相当于该数乘以2，左移2位相当于该数乘以2^2=4。
    - 只适用于该数左移时被溢出舍弃的高位中不包含1的情况。

位运算符（6）

- 右移运算符(>>)
  - 用来将一个数的各二进位全部右移若干位。
  - 移到右端的低位被舍弃，对无符号数，高位补0。
  - 若a=15，即二进制数00001111，右移2位得00000011，即十进制数3;

$$a = a >> 2$$

  - 当没有非零数位被抛弃时，右移一位相当于除以2，右移n位相当于除以2^n。

关于位运算的几个问题(1)

- 右移运算符号位的处理
  - 对无符号数，右移时左边高位移入0。
  - 对于有符号的值，
    - 若原来符号位为0(该数为正)，则左边移入0;
    - 若符号位原来为1(即负数)，则左边移入0还是1，要取决于所用的计算机系统。
      - 若移入0，称为“逻辑右移”或“简单右移”
      - 若移入1，称为“算术右移”（VC）

关于位运算的几个问题(2)

- 不同长度的数据进行位运算
  - 如果两个数据长度不同进行位运算时，系统会将二者按右端对齐。
    - 如a＆b，而a为int型，b为short型
  - 如何补位
    - 如果b为无符号整数型，则左侧添满0。
    - 如果b为有符号整数型：
      - 如果b为正数，则左侧16位补满0。
      - 如果b为负数，则左端16位补满1。

关于位运算的几个问题(3)

- 位运算赋值运算符
  - 位运算符与赋值运算符可以组成复合赋值运算符如: &=，|=， >>=， <<=， ∧=
    - a&=b 相当于 a=a&b
    - a|=b 相当于 a=a|b
    - a>>=2 相当于 a=a>>2
    - a<<=2 相当于 a=a<<2
    - a∧=b 相当于 a=a∧b

关于位运算的几个问题(4)

- 算符优先级
  - 算数运算符
  - 左移<< 右移>>
  - 关系运算符
  - 按位与&
  - 按位异或^
  - 按位或|
  - 逻辑运算符

高

低

位运算的使用（1）

- “按位与”运算的用途
  - 任意存储单元与0进行“按位与”运算可清零;
  - 取一个数中某些指定位
    - 如有一个整数a，想要其中的低字节。只需将a与 $(377)_8$ 按位与即可。

a	00 10 11 00	10 10 11 00
b	00 00 00 00	11 11 11 11
c	00 00 00 00	10 10 11 00

- “按位或”运算
  - 对一个数据的某些位取定值为1;

位运算的使用（2）

- “异或”运算符的使用

（1）使特定位翻转

- 使01111010低4位翻转（即1变为0，0变为1）可将其与00001111进行∧运算，即：

```
    01111010
(∧) 00001111
------------
    01110101
```

（2）使特定位保持不变

- 与0相∧，保留原值如012∧00=012

```
    00001010
(∧) 00000000
------------
    00001010
```

“异或”运算符示例

- 交换两个值，而不必使用临时变量
  - 假如a=3，b=4。想将a和b的值互换，可以用以下赋值语句实现：

```
a = a ∧ b;
b = b ∧ a;
a = a ∧ b;
```

设：a = 3, b = 4

```
      a=011
(∧)   b=100
      ---------
      a=111 (a∧b的结果，a已变成7)
(∧)   b=100
      ---------
      b=011 (b∧a的结果，b已变成3)
(∧)   a=111
      ---------
      a=100 (a∧b的结果，a变成4)
```

打印二进制数

```
#include <iostream>
using namespace std;
void main()
{
    int i;
    int n = -2147483648;
    unsigned int b = 0x80000000;
    for (i = 0; i < 32; i ++) {
        cout<<n & b ? 1 : 0;
        b >>= 1;
    }
}
```

条件运算符

表达式1? 表达式2: 表达式3

求值规则：如果表达式1的值为真，则以表达式2 的值作为条件表达式的值；否则以表达式2的值作为整个条件表达式的值。

例如：

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

相当于：

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

结构体

- 声明一个名为“学生”的结构体

```
struct student \\结构体的名字为“student”;
 {
      int     id;          \\声明学号为int型;
      char    name[20];  \\声明姓名为字符数组;
      char    sex;         \\声明性别为字符型;
      int     age;         \\声明年龄为整型;
      float   score;     \\声明成绩为实型;
      char    addr[30];  \\声明地址为字符数组
 };          \\注意大括号后的“;”
```

定义结构体类型的变量

- 定义结构体变量

  struct student　　student1, student2;

  （结构体类型名）（结构体变量名）;

  - 对比:

    int a; (struct student 相当于 int)

    float a; (struct student 相当于 float)

结构体变量的引用

- 引用结构体变量中成员的方式为

  结构体变量名.成员名

  - 如：student1.id = 10010;

    student1.birthday.month = 10;

- 不能将一个结构体变量作为一个整体进行输入和输出
  - 不正确的引用：cout<<student1;　　cin>>student1;
- 只能对结构体变量中的各个成员分别进行输入和输出
  - 正确的引用：cin>>student1.id;　cout<<student1.id;

结构体示例

```
int main( )
{
        int day = 7;
        struct date Birthday;
        Birthday.day = 25;
        Birthday.month = 12;
        Birthday.year = 0;
        cout<<day<<endl;
        cout<<Birthday.day<<endl;
        return 0;
}
```

```
struct  date
{
   int    month;
   int    day;
   int    year;
};
```

好好想想，有没有问题?

谢　谢　！

北京大学