表达式

COMP250205：计算机程序设计

李昊

hao.li@xjtu.edu.cn

西安交通大学计算机学院

表达式（expression）

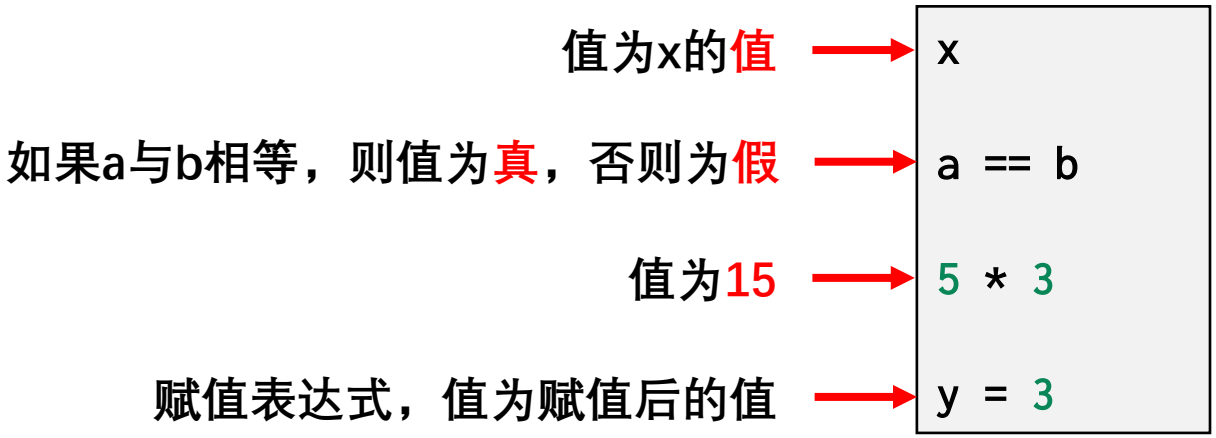
由一个或多个运算对象（operand）组成

运算对象：变量或常量

一个运算对象组成：单个变量或常量

多个运算对象组成：使用运算符组合多个变量或常量

对表达式求值将得到一个结果



运算符

一元运算符：作用在一个运算对象上

```
c++
```

二元运算符：作用在两个运算对象上

```
5 * 3
```

```
a > b
```

```
a && b
```

三元运算符：作用在三个运算对象上

```
a > b ? 3 : 5
```

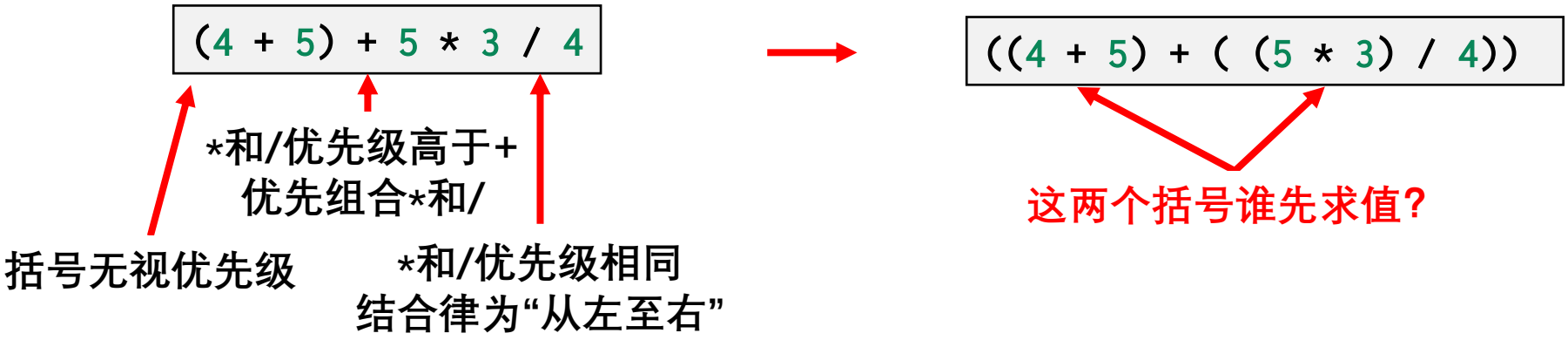
如果a>b为真，则值为3，否则为5

复合表达式与优先级

复合表达式的求值

1. 优先级高的操作符优先组合
2. 优先级相同的操作符按照结合律组合
3. 括号无视优先级和结合律

仅影响组合方式，不影响求值顺序



算术运算符

+，-，*，/，%

与四则运算、取模基本对应，但需考虑

- 整型的范围问题
- 浮点数的精度问题
- 数据类型的隐式转换问题

整型数不是整数，浮点型数不是实数

$x * x \geq 0$?

算术运算符 - QUIZ

下面的表达式中添加括号，说明其求值过程

```
12 / 3 * 4 + 5 * 15 + 24 % 4 / 2
```

```
((((12 / 3) * 4) + (5 * 15)) + ((24 % 4) / 2))
```

写出一条表达式确定一个整数是奇数还是偶数

```
(n % 2) != 0
```

按位运算符

在二进制上进行移位和逻辑操作

左移和右移：<< 和 >>

逻辑运算符：按位非（~），按位或（|），按位与（&），按位异或（^）

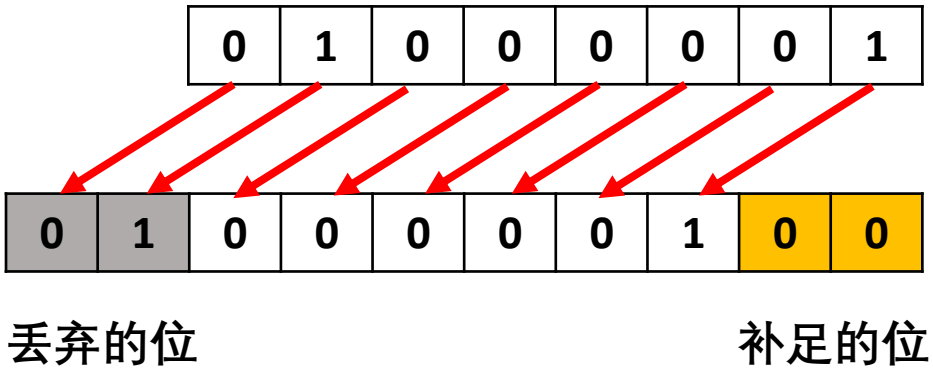
使用按位运算符 - 1

移位运算符

按位移动，按照数据类型保留/丢弃

```
char a = 'A';
a = a << 2;
```

将a左移两位

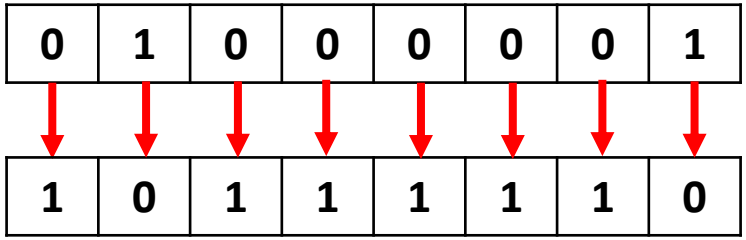


使用按位运算符 - 2

逻辑按位运算符

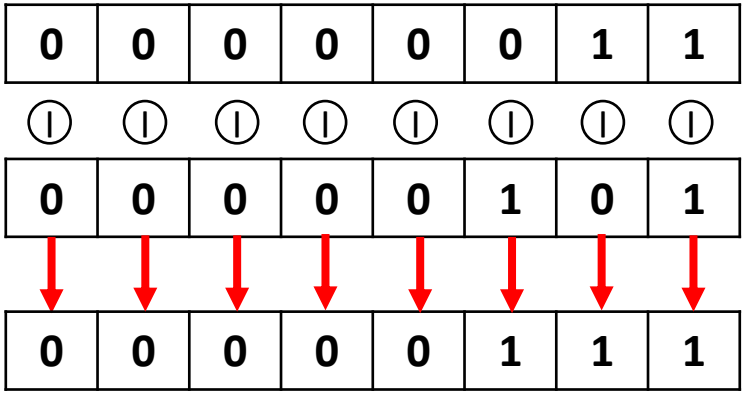
```
unsigned char x = 'A';
x = ~x;
```

对x求反




```
unsigned char x = 3;
unsigned char y = 5;
unsigned char z = x | y;
```

按位或



使用按位运算符 - 3

为什么需要按位运算符

高级语言中的最小存储单位是字节

当需要检查、修改字节内部的值时，需要按位运算

硬件控制、网络协议、操作系统内核

打开位 → `lottabits = lottabits | bits;`

切换位 → `lottabits = lottabits ^ bits;`

关闭位 → `lottabits = lottabits & ~bits;`

测试位 → `if (lottabits & bits) {}`

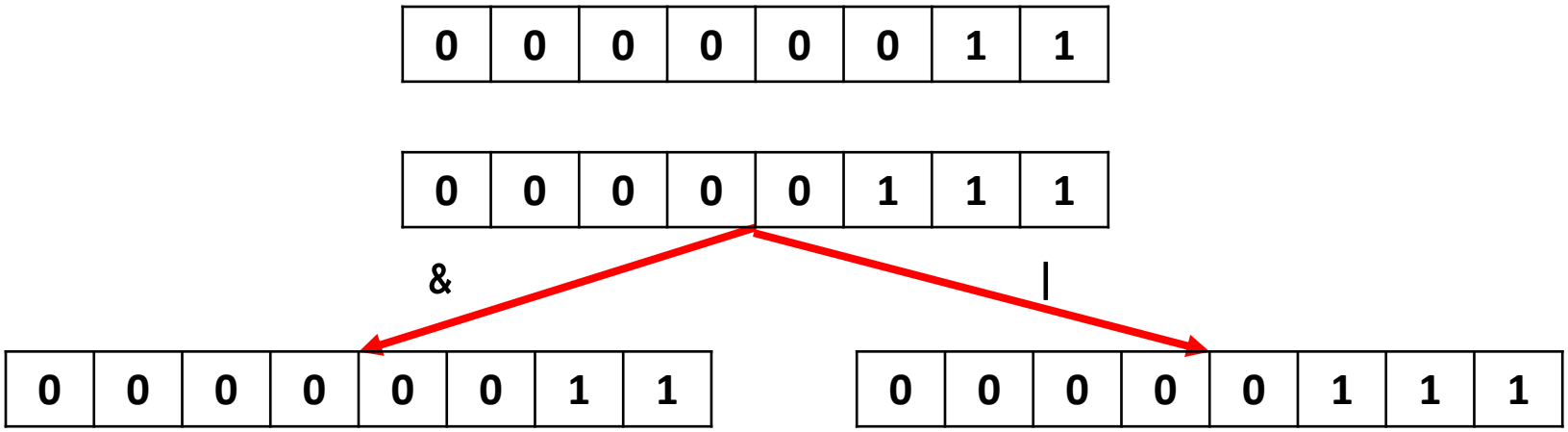
按位运算符 - QUIZ

下列表达式的值是什么?

```
unsigned long u1 = 3, u2 = 7;
```

u1 & u2

u1 | u2



逻辑运算符 - 1

对表达式进行逻辑复合，常用在分支和循环条件

- 逻辑或：||
- 逻辑与：&&
- 逻辑非：!

逻辑操作与位操作的不同

- 逻辑操作：对布尔值进行操作
- 位操作：对整数中的比特进行操作

逻辑运算符 - 2

逻辑或：两个表达式至少有一个为真，则为真

真 → `5 == 5 || 5 == 9`
真 → `5 > 3 || 5 > 10`
假 → `5 > 8 || 5 < 2`

逻辑与：两个表达式至少有一个为假，则为假

真 → `5 == 5 && 4 == 4`
假 → `5 == 3 && 4 == 4`
假 → `5 > 8 && 5 < 2`

逻辑非：对表达式的真值取反

假 → `!(5 > 3)`
若x>5，则为假；否则为真 → `!(x > 5)`

使用逻辑表达式 – 真值

在C++中，对真值的判定不止布尔型

非零整数，为真 →

```
int x = 1;
if (x) { cout << x << endl;}
```

零浮点数，为假 →

```
float y = 0.0;
if (y) { cout << x << endl; }
```

所有非零的表达式均为真

赋值表达式的值为0，为假 →

```
int x = 1;
if (x = 0) { cout << x << endl;}
```

使用逻辑表达式 – 短路 - 1

短路： 如果当前表达式的值已经可以判定整个表达式的值，则不需进一步判定后续表达式的值

减少运算量，加快程序速度

一个成年人基因检测正常（A），或高于150厘米（B），可以排除矮小症

对每个人都要执行一次基因检测 → `if (A || B) {}`

仅需对身高低于150厘米的人进行检测 → `if (B || A) {}`

使用逻辑表达式 – 短路 - 2

短路： 如果当前表达式的值已经可以判定整个表达式的值，则不需进一步判定后续表达式的值

利用短路精简逻辑表达

接受输入整型b，如果100/b大于1，则输出b

```
if (100/b > 1) {
    cout << b;
}
```

b == 0?

```
if (b != 0) {
    if (100/b > 1) {
        cout << b;
    }
}
```

双重if

```
if (b && 100/b > 1) {
    cout << b;
}
```

b为0时，第二个表达式不会被计算

逻辑运算符 - QUIZ

书写一条表达式用于测试3个值a、b、c的关系，要求a大于b，b大于c

```
a > b > c
```

如果a>b，则实际在判断1>c；否则实际在判断0>c

```
a > b && b > c
```

要求a>b和b>c同时成立

为while循环写一个条件，使其从标准输入中读取整数，直到遇到42为止

cin函数的返回值是是否读取成功

```
int n;
while (...) {}
```

```
int n;
while (cin >> n && n != 42) {}
```

赋值运算符

执行赋值（擦除并覆盖）操作，返回左侧对象

左值（lvalue）：可以放在赋值运算符左边的值

可以获取该值的地址：具备持久的内存位置

右值（rvalue）：只能放在赋值运算符右边的值

没有持久存储的临时对象，通常是字面量/表达式结果

均为初始化而非赋值

```
int i = 0, j = 0, k = 0;
const int ci = i;
```

数字常量是右值

算术表达式是右值

ci是左值，但不可修改

```
1024 = k;
i + j = k;
ci = k;
```

使用赋值运算符 - 1

赋值运算符满足右结合律

先执行jval=0，再执行ival = jval

```
int ival, jval;
ival = jval = 0;
```

赋值运算符优先级较低

必须添加括号，确保赋值操作先进行

```
int fahr = 0;
int upper = 300, step = 20;

while (fahr <= upper) {
    // calculation
    fahr = fahr + step;
}
```

```
int fahr = -20;
int upper = 300, step = 20;

while ((fahr = fahr + step) <= upper)
{
    // calculation
}
```

使用赋值运算符 - 2

赋值运算符和相等运算符

```
int i = 0, j = 1;
if (i = j) {}
if (i == j) {}
```

复合赋值运算符

```
int sum = 0;
for (int val = 1; val <= 10; val += 1) {
    sum += val;
}
```

```
a = a op b;
a op= b;
```

赋值运算符 - QUIZ

下列语句执行完毕后，变量的值是多少

先执行i=3.5，截断为3
再执行d=3，自动类型转换，d为3.0

先执行d=3.5，值为3.5
再执行i=3.5，截断为3

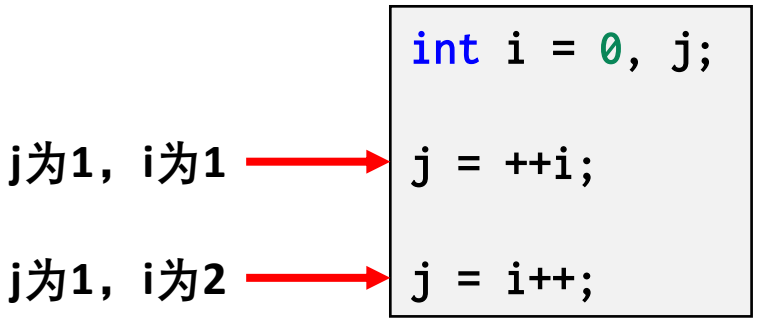
```
int i;
double d;

d = i = 3.5;

i = d = 3.5;
```

递增/递减运算符

一元运算符，把运算对象加1/减1

前置版本：将改变后的对象作为左值返回

后置版本：将改变前的对象作为右值返回



建议使用前置版本，语义更加明确

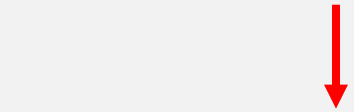
逗号运算符

简单连接两个表达式，从左到右求值

最终求值结果是右侧表达式的值

```
int main()
{
    const int max = 10;
    int cnt = max;
    for (int ix = 0; ix != max; ++ix, --cnt) {
        cout << "No." << ix << ": " << cnt << endl;
    }
    return 0;
}
```

每次循环后执行两个表达式



内容总结

表达式的优先级和结合律：

- 决定了操作对象和运算符的结合方式
- 不能决定求值顺序

基本运算符

- 算术运算符
- 逻辑运算符
- 位运算符
- 赋值运算符
- 递增递减运算符