4.1 基础

左值和右值

- 当一个对象被用作右值时，用的是对象的值（内容）
- 当一个对象被用作左值时，用的是对象的身份

4.2 算术运算符

- % : 取余或取模，负责两个整数相除所得余数

```
// m%n不等于0，则他的符号与m相同
-21 % -8 = -5;
21 & -5 = 1
```

4.3 逻辑和关系运算符

```
// 逻辑与 && 和逻辑或 || ，都是短路求值，先判断左侧再判断右侧
// 对于&& 左侧往往是为了右侧的正确性和安全性

// 相等性测试与布尔字面值
if(val); // 如果val是任意非0值，条件为真，负数也为真
if(!val); // 如果val是0，条件为真
```

4.4 赋值运算符

- = 满足左右结合律，即满足 `ival = jval = 0;` ，但要求对象类型相同或可转换
- = 优先级低，复合表达式中加 ()

4.5 递增递减运算符

- ++i将对象本身作为左侧返回
- i++将对象原始值的副本作为右值返回
- 为减少开销，不必要不用后置递增，后置递增在while循环中用得较多

```
// pbeg本身和他的递增版本都用得到
while(pbeg!=v.end() && *pbeg >=0){
    cout<<*pbeg++<<endl;
}


// 关于递增运算符的一种未定义的用法
while(beg!=s.end()&&!isspace(*beg)){
    *beg = toupper(*beg++); //注意两边的beg ，这种赋值语句是未定义的
```

4.8 位运算符（作用于整型，将对象看成是二进制集合）

运算符	功能	用法
~	位求反	~expr

运算符	功能	用法
<<	左移	expr1 << expr2
>>	右移	expr1 >> expr2
&	位与	expr & expr（都为1则为1）
^	位异或	expr ^ expr (不一样为1，相同为0)
|	位或	expr | expr (有一个为1，则为1)

位运算符仅用于处理无符号类型

```
// 位运算符优先级高于关系、赋值、条件运算符，低于算术运算符
cout<<42+10; // 正确
cout<(10<42); // 正确
cout<<10<42; // 错误
```

4.9 sizeof运算符

- `sizeof(type)` 或者 `sizeof expr` 返回expr类型的大小
- 所得值是size_t类型的常量表达式，可以为数组赋维度大小
- 特别的对数组执行sizeof得到整个数组的大小，所以要获取数组ia的元素数量可用表达式：
  `sizeof(ia)/sizeof(*ia)`

4.10 逗号运算符：？

4.11 类型转换

4.11.1 算术转换

- 整型提升，小整形转换成大整形，如：`double+int`，int会转换成double
- 类似的，bool,char,unsigned char,short/unsigned short 会转换成int

4.11.2 其他隐式类型转换

- 数组转换成指向首元素的指针
- 指针的转换
- 指针或算术类型转换成布尔类型
- 转换成常量（指针或引用指向const T）

```
int i;
const int &j = i;
const int *p = &i;
* 类类型 定义的转换
```cpp
string s = "a value"; // 字符串字面值转换成string
while(cin>>s); // IO类转换成bool，读入成功则为真
```

4.11.3 显式转换/强制类型转换（危险形为，最好不用）

- 命名的强制类型转换：`cast_name<type>(expression);`
  - `type` 是要转换的目标类型
```

- cast_name 指定要执行的是哪种转换
- expression 是要转换的类型