C++中的每个表达式都是右值（读作“are-value”）或左值（读作“ell-value”）。这些名称继承自C，最初有一个简单的助记目的：左值可以位于赋值的左侧，而右值不能

在C++中，区别就不那么简单了。在C++中，左值表达式产生一个对象或函数。但是，某些左值，如const对象，可能不是赋值的左操作数。此外，有些表达式产生对象，但将它们作为右值而不是左值返回。粗略地说，当我们使用一个对象作为右值时，我们使用对象的值（它的内容）。当我们使用一个对象作为左值时，我们使用对象的标识（它在内存中的位置）。

赋值需要一个（非常量）左值作为其左操作数，并将其左操作数作为左值生成。

地址操作符（第2.3.2节）需要一个左值操作数，并返回一个指向其操作数的指针作为右值

内置的解引用和下标运算符（第53页，第2.3.2节和第116页，第3.5.2节）以及迭代器解引用和字符串以及向量下标运算符（第106页，第3.4.1节，第93页，第3.2.3节和第102页，第3.3.3节）都产生左值。

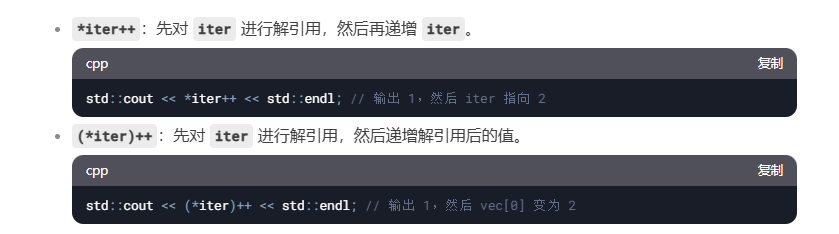
内置的迭代器递增和递减操作符（第1.4.1节和第3.4.1节）需要左值操作数，前缀版本（也就是我们到目前为止使用的版本）也产生左值。

图形用户界面, 文本, 电子邮件

描述已自动生成



AND（&&）OR（||）（？：）（，）

1. 如果有疑问，请将表达式用圆括号括起来，以强制进行程序逻辑所需的分组。2.如果你改变了一个操作数的值，不要在同一表达式的其他地方使用这个操作数。
2. 
3. 文本, 信件

   描述已自动生成

关系运算符接受算术或指针类型的操作数;逻辑运算符接受任何可以转换为bool的类型的操作数。这些运算符都返回bool类型的值。值为零的算术和指针操作数为false;所有其他值为true。这些运算符的操作数是右值，结果是右值。

除非另有说明，算术运算符可以应用于任何算术类型（第2.1.1节）或任何可以转换为算术类型的类型。这些运算符的操作数和结果都是右值。

如果我们想测试一个算术或指针对象的真值，最直接的方法是使用该值作为条件：

在比较中使用布尔值true和false作为操作数通常是个坏主意。这些文字只能用于与bool类型的对象进行比较。

赋值和初始化是不一样的。

赋值运算符的左操作数必须是可修改的左值

赋值的结果是它的左操作数，它是一个左值。结果的类型是左操作数的类型。如果左操作数和右操作数的类型不同，则右操作数转换为左操作数的类型

如果左操作数是内置类型，初始化器列表最多只能包含一个值，并且该值不需要收缩转换

分配具有低优先级

由于赋值运算符的优先级低于关系运算符，因此在条件中的赋值运算符周围通常需要括号。

例外的是，当我们使用复合赋值时，左操作数只计算一次。如果我们使用普通赋值，该操作数被求值两次：一次在右边的表达式中，另一次作为左手的操作数。

这种形式增加（或减少）其操作数，并产生已更改的对象作为其结果。后缀运算符递增（或递减）操作数，但产生原始未更改值的副本作为其结果：

仅在必要时使用后缀运算符

auto pbeg = v.begin();

// print elements up to the first negative value

while (pbeg != v.end() && \*beg >= 0)

cout << \*pbeg++ << endl; // print the current value and advance pbeg

文本, 信件

描述已自动生成

点运算符从类类型的对象中获取成员;箭头被定义为ptr-> par是（\*ptr）的同义词。

so that ptr->mem is a synonym for (\*ptr).mem:

箭头运算符需要一个指针操作数并产生一个左值。如果从中获取成员的对象是左值，则点运算符产生左值;否则结果是右值。

如果两个表达式都是左值或它们转换为公共左值类型，则条件运算符的结果是左值。否则，结果为右值。

嵌套的条件语句很快变得不可读。最好不要超过两三个。

由于无法保证如何处理符号位，因此我们强烈建议使用带位运算符的无符号类型。

文本

描述已自动生成

将操作数转换为一个公共类型，而不是尝试将两种不同类型的值相加。这些转换是自动执行的，不需要程序员的干预，有时甚至不需要程序员的知识。因此，它们被称为隐式转换

在以下情况下，编译器会自动转换操作数：·

在大多数表达式中，小于int的整数类型的值会首先提升为适当的较大整数类型。

·在条件中，非bool表达式被转换为bool。·

在初始化中，初始化式被转换为变量的类型;

在赋值中，右边的操作数被转换为左边的类型。

在具有混合类型的操作数的算术和关系表达式中，类型被转换为公共类型。

正如我们将在第6章中看到的，转换也发生在函数调用期间。

数组到指针的转换：在大多数表达式中，当我们使用数组时，数组会自动转换为指向该数组中第一个元素的指针：

虽然有时候是必要的，但强制转换本质上是危险的构造。

reinterpret\_cast本质上依赖于机器。安全地使用reinterpret\_cast需要完全理解所涉及的类型以及编译器如何实现强制转换的细节。

旧式强制转换比命名强制转换更不明显。因为他们很容易被忽视，所以更难追踪到一个流氓演员。