表达式

2021年10月27日 22:17

1、在类型转换时，小整数类型（如bool、char、short等）通常会被提升成较大的整数类型，主要是int。

2、左值和右值：

一个简单的归纳：当一个对象被用作右值的时候，用的是对象的值（内容）；当对象被用作左值的时候，用的是对象的身份（在内存中的位置）。

一个重要的原则：在需要右值的地方可以用左值来代替，但是不能把右值当成左值（也就是位置）使用。

使用关键字 decltype（参见 2.5.3 节，第 62 页）的时候，左值和右值也有所不同。如果表达式的求值结果是左值，decltype 作用于该表达式（不是变量）得到一个引用类型。举个例子，假定 p 的类型是 int*，因为解引用运算符生成左值，所以 decltype(*p) 的结果是 int&。另一方面，因为取地址运算符生成右值，所以 decltype(&p) 的结果是 int**，也就是说，结果是一个指向整型指针的指针。

3、参与取余运算的运算对象必须是整型。

4、短路求值：逻辑与运算符和逻辑或运算符都是先求左侧运算对象的值再求右侧运算对象的值，当且仅当左侧运算对象无法确定表达式的结果时才会计算右侧运算对象的值。

很多情况下，左侧运算对象是为了确保右侧运算对象求值过程的正确性和安全性。如：index!=s.size()&&!isspace(s[index])

5、后置递增运算符的优先级高于解引用运算符，因此*iter++等价于*(iter++)

6、

表 4.3：位运算符（左结合律）

运算符	功能	用法
~	位求反	~ expr
<<	左移	expr1 << expr2
>>	右移	expr1 >> expr2
&	位与	expr & expr
^	位异或	expr ^ expr
|	位或	expr | expr

7、

C++11 新标准允许我们使用作用域运算符来获取类成员的大小。通常情况下只有通过类的对象才能访问到类的成员，但是 sizeof 运算符无须我们提供一个具体的对象，因为要想知道类成员的大小无须真的获取该成员。

sizeof 运算符的结果部分地依赖于其作用的类型：

- 对 char 或者类型为 char 的表达式执行 sizeof 运算，结果得 1。
- 对引用类型执行 sizeof 运算得到被引用对象所占空间的大小。
- 对指针执行 sizeof 运算得到指针本身所占空间的大小。
- 对解引用指针执行 sizeof 运算得到指针指向的对象所占空间的大小，指针不需有效。
- 对数组执行 sizeof 运算得到整个数组所占空间的大小，等价于对数组中所有的元素各执行一次 sizeof 运算并将所得结果求和。注意，sizeof 运算不会把数组转换成指针来处理。
- 对 string 对象或 vector 对象执行 sizeof 运算只返回该类型固定部分的大小，不会计算对象中的元素占用了多少空间。

因为执行 sizeof 运算能得到整个数组的大小，所以可以用数组的大小除以单个元素的大小得到数组中元素的个数：

8、算术转换的规则定义了一套类型转换的层次，其中运算符的运算对象将转换成最宽的类型。

例如，如果一个运算对象的类型是long double，那么不论另外一个运算对象的类型是什么都会转换成long double。

9、虽然有时不得不使用强制类型转换，但这种方法本质上是非常危险的。所以建议避免强制类型转换。