x、回顾

Chapter	M
Name	xingzp

- 智能指针类是一个组合类，旨在管理动态分配的内存，并确保在智能指针对象超出作用域时删除内存。
- 复制语义允许复制我们的类。这主要通过复制构造函数和复制赋值操作符来完成。
- 移动语义意味着类将转移对象的所有权，而不是进行复制。这主要是通过移动构造函数和移动赋值操作符完成的。
- 已弃用auto_ptr，应该避免使用。
- r值引用是设计为用r值初始化的引用。使用双&号创建r值引用。编写接受r值引用形参的函数是可以的，但您几乎不应该返回r值引用。
- 如果我们构造一个对象或进行赋值，其中实参是一个l值，我们唯一能做的合理的事情就是复制这个l值。我们不能假设改变l值是安全的，因为它可能在稍后的程序中再次被使用。如果我们有一个表达式" a = b "，我们就不会合理地期望b以任何方式改变。
- 然而，如果我们构造一个对象或做一个赋值，其中参数是r-value，那么我们知道r-value只是某种类型的临时对象。我们不需要复制它(这可能很昂贵)，而是可以简单地将它的资源(这很便宜)转移到我们正在构造或分配的对象。这样做是安全的，

因为临时变量无论如何都会在表达式结束时被销毁，所以我们知道它永远不会再被使用!

- 通过删除复制构造函数和复制赋值操作符，可以使用delete关键字禁用所创建类的复制语义。
- std::move允许将l值作为r值处理。当我们想在l值上调用移动语义而不是copy语义时，这很有用。
- 使用std::move_if_noexcept时，如果对象具有noexcept移动构造函数，则返回可移动的r值，否则将返回可复制的l值。我们可以将noexcept说明符与std::move_if_noexcept一起使用，以便仅在存在强异常保证时使用移动语义(否则使用复制语义)。
- std::unique_ptr是您应该可能使用的智能指针类。它管理单一的不可共享资源。应该优先使用std::make_unique()(在c++ 14中)来创建新的std::unique_ptr。unique_ptr禁用复制语义。
- shared_ptr是当您需要多个对象访问同一资源时使用的智能指针类。直到管理该资源的最后一个std::shared_ptr被销毁，该资源才会被销毁。应该优先使用std::make_shared()来创建新的std::shared_ptr。对于std::shared_ptr，应该使用复制语义来创建指向相同对象的额外std::shared_ptr。
- 当你需要一个或多个对象来查看和访问由std::shared_ptr管理的资源时，可以使用智能指针类std::weak_ptr，但与std::shared_ptr不同的是，在决定是否应该销毁资源时，std::weak_ptr不用被考虑。