ue4的智能指针很多，这些智能的存在，让开发者方便做资源，内存、对象的管理

1 持有非UObject对象的智能指针

TSharedPtr，TSharedRef，TWeakPtr，TSharedFromThis，TUniquePtr，TUniqueObj，TPimplPtr，TNonNullPtr，TOptional

2 持有UObject的智能指针

TStrongObjectPtr，TWeakObjectPtr，FSoftObjectPtr，TSoftObjectPtr，TSoftClassPtr，FSoftObjectPath，FLazyObjectPtr，TPersistentObjectPtr，FGCObject，FGCObjectScopeGuard，TGCObjectsScopeGuard，TWeakInterfacePtr

TSharedPtr（抄shared\_ptr的）

一个TSharedPtr占16字节，64位一个指针占8字节

一、TSharedPtr结构

ObjectType\* object;//我们所持有引用的Object

SharedPointerInternals::FSharedReferencer< Mode > SharedReferenceCount;//这个内部其实也就是一个指针，占8字节

template< ESPMode Mode >

class FSharedReferencer

{

     FReferenceControllerBase\* ReferenceController;

}

为什么不直接用STL呢？

std非全平台可用，并且ue4的智能指针可以无缝兼容UE4的容器。

成员变量

ObjectType\* Object; //原始指针

SharedPointerInternals::FSharedReferencer< Mode > SharedReferenceCount;  //做引用计数的控制

TSharedRef

与TSharedPtr唯一的区别是TSharedRef不能为空，TSharedPtr可以转换成TSharedRef，但是必须是有效的指针，否则会触发 Check引起崩溃

TWeakPtr

TShared和TWeakPtr，如果指针是互相转换得到的，那么拿着的ReferenceController都是同一个，WeakPtr没有重载->和.这样的指针运算符，所以不能直接用，要从TWeakPtr转换成TSharedPtr才能用

TSharedFromThis

类似于C++的std::enable\_shared\_from\_this

std::enable\_shared\_from\_this

std::enable\_shared\_from\_this 能让其一个对象（假设其名为 t ，且已被一个 std::shared\_ptr 对象 pt 管理）安全地生成其他额外的 std::shared\_ptr 实例（假设名为 pt1, pt2, … ） ，它们与 pt 共享对象 t 的所有权。

struct Bad

{

*std*::*shared\_ptr*<Bad> getptr()

       {

              return *std*::*shared\_ptr*<Bad>(this);

       }

};

//每个shared\_ptr都会认为自己是对象的唯一拥有者。

//会导致两次析构，第二次析构，指针指向的空间已经被析构

*std*::*shared\_ptr*<Bad> bp1 = *std*::*make\_shared*<Bad>();

*std*::*shared\_ptr*<Bad> bp2 = bp1->getptr();

*std*::*cout* << "bp2\_use\_count()=" << bp2.*use\_count*() << "\n";

使用原因：

需要在类对象的内部中获得一个指向当前对象的shared\_ptr对象