dynamic\_cast、static\_cast、const\_cast、reinterpret\_cast

dynamic\_cast【仅用于指针或者引用】

对于向上类型转换的时候，总是成功并且安全的

对于向下类型转换不都是成功的。对于指针，如果转换不成功，返回空指针。对于引用，如果转换不成功，则会跑出异常(std::bad\_cast)

A B C D菱形的转换：不可以直接从D到A，要选择一条路线，D线转换成B或者C，再转换成A【不过如果菱形用虚继承来实现的话，就可以直接从D到A转换了】

dynamic\_cast转换的依据：

dynamic\_cast来进行类型转换的需要根据虚函数表来判断继承关系（动态多态的实现，就是依托于虚函数表来实现的），这里对虚函数表进行查表的操作，会带来额外的性能损耗

所以基类要有虚函数，基类的析构函数一般是虚函数

static\_cast【普通类型转换也可以使用】

对于向上转换(up)是安全的

对于向下(down)转换总是成功的，并且不提供检查。如果转换不成功，也不会跑出异常。效果和C风格的强制类型转换是一样的，需要编程者保证转换的安全性，既被转换的派生类和目标转换类型之间必须是继承派生关系

const\_cast

如果需要改变cv属性，使用const\_cast【c：const v：volatile】

reinterpret\_cast

在编译期完成，可以转换任何类型的指针，所以极不安全。非极端情况不要使用。

选择方法：

如果是多态应用场景中（非单元测试代码），在基类和派生类进行类型转换（指针或者引用），则优先使用static\_cast，不要使用dynamic\_cast

如果是在单元测试中，需要对类型进行检查（比如测试工厂代码返回的类是否正确）,则可以使用dynamic\_cast

如果是不涉及到cv属性的变化的时候，并且非多态应用场景下，优先使用static\_cast

如果需要改变cv属性，使用const\_cast【c：const v：volatile】

最后选择使用reinterpret\_cast进行转换，或者几种转换组合使用

智能指针的转换就是多个pointer

static\_pointer\_cast、dynamic\_pointer\_cast、const\_pointer\_cast、reinterpret\_pointer\_cast