* const\_cast

1. **原型：**

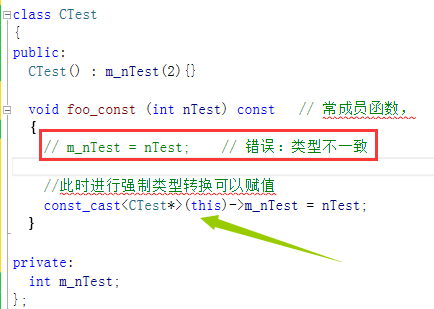
**const\_cast<type-id>expression**

1. **特点：**

**2.1 用于去除或者添加上const属性**

**2.2 cosnt\_cast是四种类型转换符中唯一可以对常量进行操作的转换符**

1. **应用语境：**
   1. **类中常成员函数里通过const属性的添加对普通类成员数据进行修改：**



**此时常成员函数：foo\_const (int nTest) const 中的this指针是一个指向const类型对象的const指针，所以当要修改类成员数据时需要经过强制类型转换**

**3.2 常量指针被转化成非常量指针，转换后指针指向原来的变量**

1. **对于基本数据类型：**

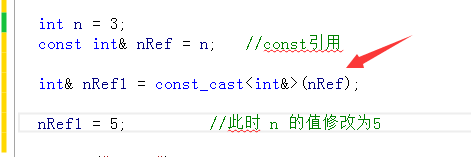


1. **同理，常量对象指针转换为非常量对象指针也是如此：**

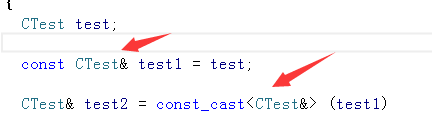


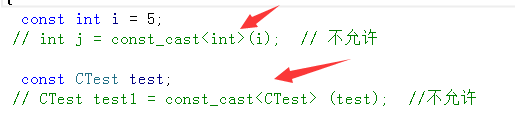
**3.3 常量引用转为非常量引用**

1. **基本数据类型：**



1. **常量对象引用转为非常量对象引用：**



**3.4 常量对象或者是基本数据类型不允许转化为非常量对象，只能通过指针和引用来修改：**

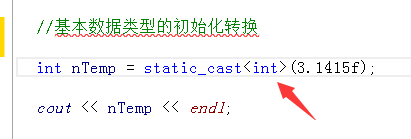
* static\_cast

1. **原型：**

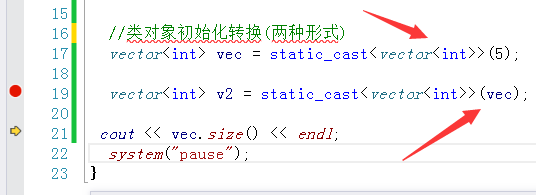
**static\_cast<type-id> expression**

1. **特点：**
   1. **执行非多态的转换操作，用于代替C中通常的转换操作**
   2. **没有运行时类型检查来保证转换的安全性。**
   3. **用于常见的隐式转换、强转**
2. **应用语境：**
   1. **运用1：**

**用于基本数据类型之间的转换**，**如把int转换成char，把int转换成float等。这种转换的安全隐患由程序员自己把控。**



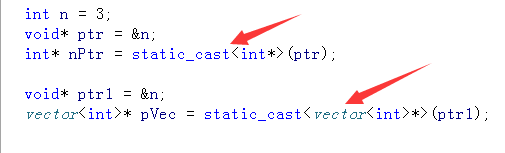
**类对象的初始化转换也是一样，安全性由自己把控：**



* 1. **运用2：**

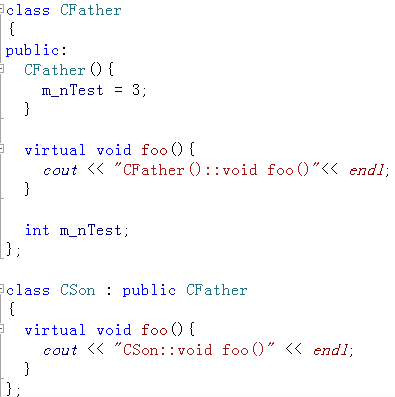
**void\* 到任何类型的强转：空指针转换为任何目标类型的指针**

**这里目标类型可以是对象、其他基本数据类型的指针,如下所示：**



* 1. **运用3：**

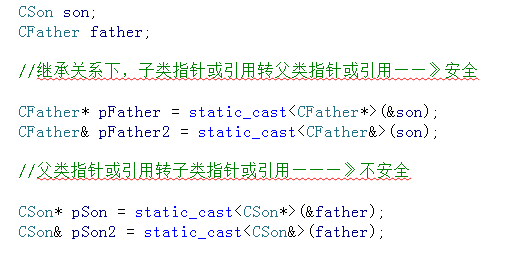
**用于类层次结构中基类和子类之间指针或引用的转换**:



**进行上行转换时：（把子类的指针或引用转换成基类表示）是安全的；**

**而进行下行转换（把父类指针或引用转换成子类指针或引用）时，由于没有动态类型检查，所以是不安全的；**

**如下所示：**



* dynamic\_cast

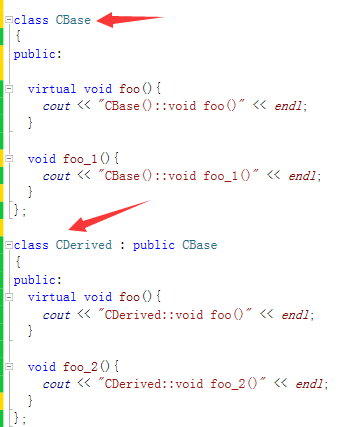
1. **原型：**

**dynamic\_cast<type-id> expression**

1. **特点:**
   1. **只有在派生类之间转换时才使用dynamic\_cast，type-id必须是类指针，类引用或者void\***
   2. **dynamic\_cast在运行时检查（一个基类指针是否实际指向的是派生类对象），如果类型不一致、转型不成功则返回一个空指针**
   3. **基类必须要有虚函数，因为dynamic\_cast是运行时类型检查，需要运行时类型信息，而这个信息是存储在类的虚函数表中，只有一个类定义了虚函数，才会有虚函数表。**
   4. **dynamic\_cast依赖于RTTI信息（RTTI是运行阶段类型识别，旨在为程序在运行阶段确定对象类型提供一种标准方式）**
2. **应用场景：**
   1. **应用场景1：**

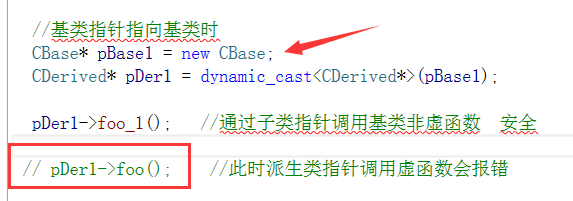
**下行转换，将基类指针或引用转换为派生类指针或引用**

**源码如下：**

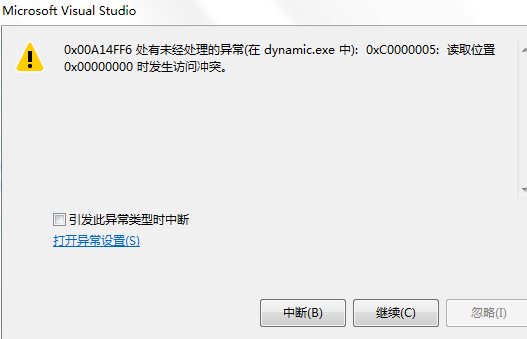


**如上图, 将基类的指针或引用转换成派生类的指针或引用：**

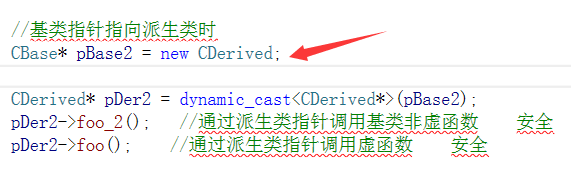
1. **当基类指针指向基类时，用派生类指针调用虚函数会报错，因为此时转换后返回的指针是空指针**



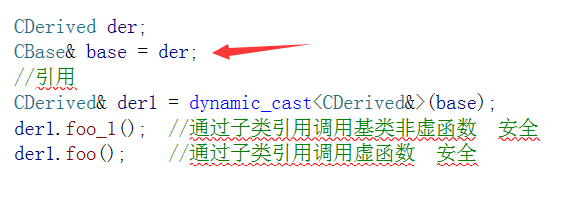
**此时调用虚函数之后程序会报错：**



1. **当基类指针指向派生类时，用派生类的指针调用虚函数时是安全的**



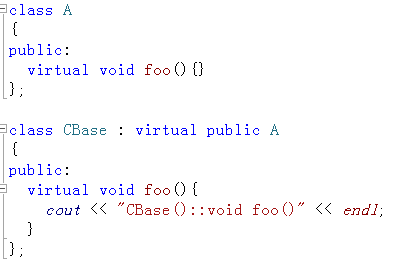
1. **当基类引用指向派生类时，用派生类引用调用虚函数是安全的：**

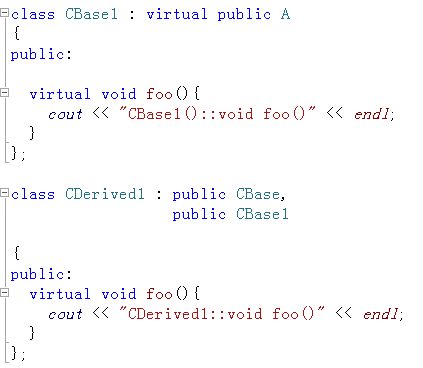


* 1. **应用场景2：**

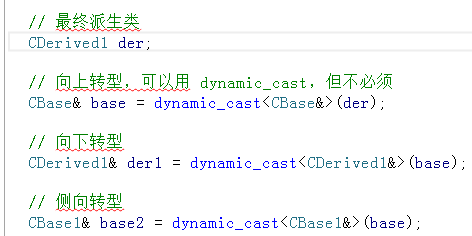
**多重继承、菱形继承情况下的 下行转换、上行转换、侧向转换**

**源码如下：**





**此时用dynamic\_cast来进行几种如下转换：**



* reinterpret\_cast

1. **原型：**

**reinterpret\_cast<type-id>expression**

1. **特点：**

**2.1 编译器在编译期处理**

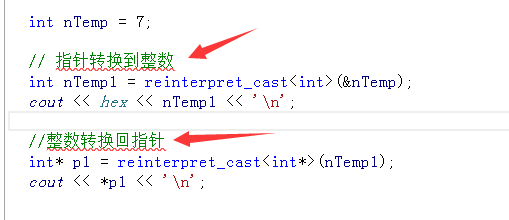
**2.2 该类型转换具有高危性**

**2.3 reinterpret\_cast是从底层对数据进行重新解释，依赖具体的平台，**

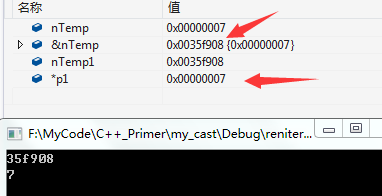
**可移植性差**

1. **应用语境：**
   1. **运用1：**

**指针与整型之间互转：**



**运行结果显示：**

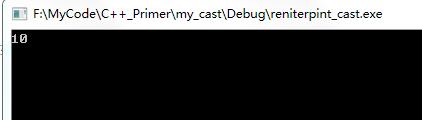


* 1. **运用2：**

**函数转为函数指针并转回：**

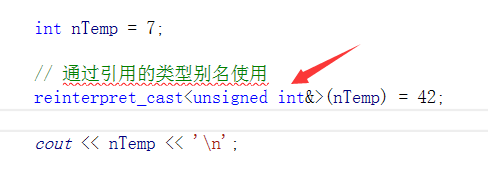


**程序运行结果：**



* 1. **运用3：**

**通过引用的类型别名使用：**



**运行结果如下：**

