C++中的类型转换

1．static\_cast<int>(98.87);

用法：static\_cast < type-id > ( expression )

该运算符把expression转换为type-id类型，但没有运行时类型检查来保证转换的安全性。它主要有如下几种用法：

1）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（把子类的指针或引用转换成基类表示）是安全的；

　　进行下行转换（把基类指针或引用转换成子类表示）时，由于没有动态类型检查，所以是不安全的。

2）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enum。这种转换的安全性也要开发人员来保证。

3）把空指针转换成目标类型的空指针。

4）把任何类型的表达式转换成void类型。

　　注意：static\_cast不能转换掉expression的const、volitale、或者\_\_unaligned属性。

2．const\_cast<int\*>(p);

去掉常量指针的常量属性

3．reinterpret\_cast<int>(98.87);

数据以二进制存在形式的重新解释，reinterpret\_cast 只能在指针之间转换。

4．dynamic\_cast<int>(98.87);

dynamic\_cast主要用于类层次间的上行转换和下行转换，还可以用于类之间的交叉转换。

在类层次间进行上行转换时，dynamic\_cast和static\_cast的效果是一样的；

在进行下行转换时，dynamic\_cast具有类型检查的功能，比static\_cast更安全。

用法：dynamic\_cast < type-id > ( expression )

该运算符把expression转换成type-id类型的对象。Type-id必须是类的指针、类的引用或者void \*；

如果type-id是类指针类型，那么expression也必须是一个指针，如果type-id是一个引用，那么expression也必须是一个引用。

下行转换时，基类要有虚函数，否则会编译出错；static\_cast则没有这个限制。

？平行转换会变为空指针