C-style cast 举例：

```
int i;
double d;
i = (int) d;
```

上面的代码就是本来为 double 类型的 d，通过(int)d 将其转换成整形值，并将该值赋给整形变量 i (注意 d 本身的值并没有发生改变)。这就是典型的 c-style 类型转换。

在简单的情况下，上面这种类型转换可以很好地工作，但在 C++中往往还是不够的，为此 ANSI-C++新标准定义的四个转换符，即 static_cast、dynamic_cast、reinterpret_cast 和 const_cast。同时在 C++环境中，原先的 C-Style 的类型转换仍旧可以使用。

1) static_cast

用法：static_cast <typeid> (expression)

说明：该运算符把 expression 转换为 typeid 类型，但没有运行时类型检查来确保转换的安全性。

用途：

a) 用于类层次结构中基类和派生类之间指针或者引用的转换。up-casting (把派生类的指针或引用转换成基类的指针或者引用表示)是安全的；down-casting(把基类指针或引用转换成子类的指针或者引用)是不安全的。

b) 用于基本数据类型之间的转换，如把 int 转换成 char，这种转换的安全性也要由开发人员来保证。

c) 可以把空指针转换成目标类型的空指针(null pointer)。

d) 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)dynamic_cast

用法：dynamic_cast <typeid> (expression)

说明：该运算符把 expression 转换成 typeid 类型的对象。typeid 必须是类的指针、类的引用或者 void*。如果 typeid 是类的指针类型，那么 expression 也必须是指针，如果 typeid 是一个引用，那么 expression 也必须是一个引用。一般情况下，dynamic_cast 用于具有多态性的类(即有虚函数的类)的类型转换。

dynamic_cast 依赖于 RTTI 信息，其次，在转换时，dynamic_cast 会检查转换的 source 对象是否真的可以转换成 target 类型，这种检查不是语法上的，而是真实情况的检查。先看 RTTI 相关部分，通常，许多编译器都是通过 vtable 找到对象的 RTTI 信息的，这也就意味着，如果基类没有虚方法，也就无法判断一个基类指针变量所指对象的真实类型，这时候，dynamic_cast 只能用来做安全的转换,例如从派生类指针转换成基类指针。而这种转换其实并不需要 dynamic_cast 参与。也就是说,dynamic_cast 是根据 RTTI 记载的信息来判断类型转换是否合法的。

用途：主要用于类层次之间的 up-casting 和 down-casting，还可以用于类之间的交叉转换。在进行 down-casting 时，dynamic_cast 具有类型检查的功能，比 static_cast 更安全。检测在运行时进行。如果被转换的指针不是一个被请求的有效完整的对象指针，返回值为 NULL。当用于多态类型时，它允许任意的隐式类型转换以及相反过程。不过，与 static_cast 不同，在后一种情况里（注：即隐式转 换的相反过程），dynamic_cast 会检查操作是否有效。也就是说，它会检查转换是否会返回一个被请求的有效的完整对象。

注意：dynamic_cast 不能转换掉 expression 的 const、volitale 或者__unaligned 属性。

3) reinterpret_cast
用法：reinterpret_cast <typeid>(expression)
说明：转换一个指针为其他类型的指针，也允许将一个指针转换为整数类型，反之亦然。这个操作符能够在非相关的类型之间进行转换。操作结果只是简单的从一个指针到别的指针的值的二进制拷贝，在类型之间指向的内容不做任何类型的检查和转换。这是一个强制转换。使用时有很大的风险，慎用之。
注意：reinterpret _cast 不能转换掉 expression 的 const、volitale 或者__unaligned 属性。

4)const_cast
用法：const_cast<typeid>(expression)
说明：这个类型操纵传递对象的 const 属性，或者是设置或者是移除。如：

```
Class C{…}
const C* a = new C;
C* b = const_cast<C*>(a);
```

如果将上面的 const_cast 转换成其他任何其他的转换，编译都不能通过，出错的信心大致如下：
“…cannot convert from 'const class C *' to 'class C *'”。