**Const\_cast**

* 给指针去除const 修饰符的类型转换 const\_cast<T\*>(ptr);
* 但是例子代码却能过?

<https://www.youtube.com/watch?v=2_-BS9UI0po>

4：37

**Namespace**

* 命名空间是一个作用范围，::操作符是访问这个范围中的成员
* 其实类也是一个作用范围，类中的成员也可以用::操作符访问，比如访问类中的静态成员可以直接用 class name :: static\_var; 进行访问

例:

class xue{

public:

static int number;

};

int xue::number=5;

* Std是c++提供的一个基础命名空间，其中的成员包括常用的基础操作符 比如cout cin

**嵌套命名空间**

* 你自己也可以构造namespace,但需要用::进行调用，也可以进行namespace的嵌套

<https://www.youtube.com/watch?v=T0IrsXLxVEI>

例:

namespace a{

namespace b{

int j;

}

}

int main()

{

a::b:: j;

return 0;

}

**未命名空间**

* 就是没有命名的命名空间，特点是不用::操作符 可以直接调用 其中成员

例:

namespace {

int x;

void display(){

cout<<"x is "<<x<<endl;

}

}

int main()

{

x=10;

display();

return 0;

}

**命名空间改名**

* 命名空间可以被改名

例 给嵌套命名空间改名:

namespace a{

namespace b{

int j;

}

}

namespace one=a;

namespace two=one::b;

int main()

{

two::j;

return 0;

}

**Mutable Class 易变类**

* Mutable 易变修饰符 作用在 const常量修饰的 函数:

class muteclass{

public:

mutable int j; //Mutable修饰

muteclass(){

j=5;

}

void mutefn() const{ //const修饰函数要放在函数名后面

j++; //在const函数中原本不可以访问的对象j在被mutable修饰后变得可以访问

cout<<j;

}

};

int main()

{

const muteclass obj; //const对象一定要对应const函数

obj.mutefn();

return 0;

}

**Virtual inheritance 虚拟继承**

- 在多态章节已学过virtual修饰符是为了继承和多态中 基类和导类重名函数冲突时使用，

- 在继承多态中被virtual修饰意味着，对编译器来说虚拟不存在

- 但virtual同样可以修饰类之间的继承关系:

<https://www.youtube.com/watch?v=7APovvvftQs>