条款9 优先考虑别名声明而非typedef (基本使用见base3)

① typename关键字：用来说明模板内部的-T标识符代表是某种类型

类型别名

当编译器遇到类似T::xxx这样的代码时，它不认识xxx是T类型成员还是T数据成员，直至实例化才知道，但为了处理好模板，编译器需要命名是否表示一T类型，默认情况下C++语言假定通过作用域运算符访问的命名不是类型，而是数据成员

```
template<typename T>
class MyClass { void foo(){typename T::subType * ptr;}};
```

如果没有typename，subType被假设为非类型成员(比如static或者枚举常量，亦或是内部嵌套或using声明的Public别名)

② 对模板来说，using比typename更好

```
template<typename T>
```

推荐 a. `using myVec = std::vector<T>;` myVec可直接使用无须typename指定类型

```
b. struct myVec2{
       typedef std::vector<T> type;
   }
```

必须带上

由于type在某个myVec2的作用域下非别名

myVec2<T>::type是否为T类型取决于T，所以要用typename声明

③类型萃取器 #include<type_traits>

实现类型查询，类型转换 添加/删除模板的修饰

std::remove_const<T>::type C++11 const T → T

std::remove_const_t<T> C++14

```
template<class T>
using remove_const_t<T> = typename remove_const<T>::type;
```