

## 条款} 理解decltype 模板类型推导

① `decltype` + 变量 所有信息均会保留数组与函数也不会退化

② `decltype` + 表达式 返回表达式结果的类型

左值/右值 { 左值: 得到该类的左值引用  
右值: 得到该类

`int *p = &a; decltype(*p) b = a; *p为表达式, 推出  $\int$`   
`decltype(p) c; 推出  $\int *$`

`decltype` 单独作用于对象, 没有使用对象的表达式属性, 而是获得变量类型, 若想将变量作为表达式, 可以加括号

③ `decltype` 并不会实际计算表达式的值, 编译器会分析表达式并得到类型  
存在的函数不会运行, 仅给出函数返回的类型

④ `decltype` 使用场景:

某些情况想提前知道模板函数返回值

对于 `std::vector<>` 对象, 使用 `operator[]` 通常会返回 `T&`  
但若是 `bool` 时有例外, 因此模板函数的返回值必须是自动推导

C++11 可使用尾置返回类型语法 `auto xxx → decltype(x)`

C++14 可直接使用 `auto` 但 `auto` 作为函数返回值走的是模板类型的推导

因此使用 `decltype(auto)` 保留 `xxx` 的所有修饰

(详情见  
条款二)

## ⑤转发并不完美

就算用上的万能引用与完美转发,由于容器调用了  
访问对象[] (该对象一定为左值), 则返回的永远是左值,  
进而无法保留值类型