

## base8 可调用对象类型

① lambda表达式存在的意义 (匿名函数只是为了不起名?)

闭包: 带有上下文(状态的函数) (有些函数的参数列表已经无法改变)

闭包的实现方式: 重载 `operator()`; lambda表达式; `std::bind`

```
vector<string> word = {"Yes", "No", "None"};
```

```
int sz=4;
```

```
lambda: auto wc = find_if(word.begin(), word.end(),  
[sz](const string&a) { return a.size > sz; });
```

```
函数+bind: bool isShorter(const string&a, int sz) {  
    return a.size > sz; }
```

```
using namespace std::placeholders; 使用占位符
```

```
auto fun = std::bind(isShorter, _1, sz);
```

② 可调用对象与function: <<C++Primer>> P51

C++中可调用对象:

a. 函数	b. 函数指针	c. lambda表达式
d. <code>std::bind</code>	e. 重载函数调用运算符的类	

这5种对象类型都不同, 有无可容纳所有可调用对象的容器呢

有的兄弟, 有的  $\Rightarrow$  `std::function`

注: 当 `lambda[]` 无任何捕获时, 且参数列表与目标指针类型兼容  
则可以返回函数指针

```
std::function<int(int, int)>;
```

绑定了两个 `int` 参数, 返回值为 `int` 的所有可调用对象