基础7 lambda表达式初探

① lambda表达式的底层实现原理

lambda表达式: `size_t sz=0;`

转为↓

```
auto Sizecpr=[sz](const string &s){
return a.size>sz;};
```

可调用类: `class Sizecpr{public:(size_t n):sz(n)}`

本质为匿名函数

```
bool operator()(const string &a) const
{...}

private: size_t sz;}
```

根据捕获方式决定类内成员是常量,引用还是指针

② lambda表达式基础语法 <<现代C++...> P58

`[captures](params) specifiers exceptions → ret{body}`

specifiers 可选限定符: 默认为const (类中的const函数)

可使用mutable,希望可通过函数修改值

exception 可选异常符说明: 可用noexcept指明函数是否抛出异常

→ return 可选返回值类型 大多数情况可以自行推导 但初始化列表时不行

(params)可选参数列表 可使用auto(C++14引入)

```
auto foo=[](auto a){return a;};
int three=foo(3);
```

[captures]捕获列表: ①只能捕获非静态局部变量、可按值、按引用或组合

②捕获发生在lambda表达式定义时,而非使用时

③广义的捕获(C++14后)

非引用捕获实际创建了新变量

所以甚至可使用=进行赋值

至此捕获可传右值[r=std::move(x)]

④特殊的捕获方法[=][&][this][*this]

[this]捕获this指针,可以让我们使用this类型的成员函数或变量

[=]捕获所有局部变量的值,包括this

但只会捕获真正使用了的值

[&]捕获……的引用,……

[*this]可捕获this指向的对象的副本(C++17引入)