Callable Object：可被调用的对象，可以被某种方式调用其某些函数的对象，可以是：

1. 一个函数。
2. 指向成员函数的指针，当你通过对象调用它，该对象被传递为第一个实参（必须是reference或pointer），其它实参则一一对应成员函数的参数。
3. 一个函数对象（该对应拥有operator ()，对函数运算符重载）
4. 一个lambda，严格来说也是一个函数对象。

例：程序stl\_test12

// 可被调用的对象

// 1 函数

void Func(int x,int y)

{

*cout*<<"callable function : "<<"x: "<<x<<": y : "<<y<<*endl*;

}

// 可被调用的对象

// 2 lambda

auto lambda=[](int x,int y)

{

*cout*<<"callable lambda : "<<"x: "<<x<<": y : "<<y<<*endl*;

};

// 可被调用的对象

// 3 函数对象

// 4 指向成员函数的指针

classTest{

public:

// 仿函数

void operator ()(int x,int y) const

{

*cout*<<"callable funtion object : "<<"x: "<<x<<": y : "<<y<<*endl*;

}

voidMemberFunc(int x,int y)

{

*cout*<<"callable member function : "<<"x: "<<x<<": y : "<<y<<*endl*;

}

};

int *main*(int argc,char\* argv[])

{

Test t;

*shared\_ptr*<Test> shared\_pointer(new Test);

//std::bind() 绑定可调用对象传递参数

auto b1= *bind*(Func,77,33);

b1();// calls: Func(77, 33)

*bind*(lambda,77,33)();// calls: lambda(77, 33)

*bind*(Test(),77,33)();// calls: Test::operator() (77, 33)

// calls: t.MemberFunc(77, 33)

*bind*(&Test::MemberFunc,t,77,33)();

// calls: shared\_pointer->MemberFunc(77, 33)

*bind*(&Test::MemberFunc,shared\_pointer,77,33)();

return0;

}

输出为：

callable function : x: 77: y : 33

callable lambda : x: 77: y : 33

callable funtion object : x: 77: y : 33

callable member function : x: 77: y : 33

callable member function : x: 77: y : 33