

# C11 新特性之function

本人能力有限，请多包涵



讲述人：李智博  
老师



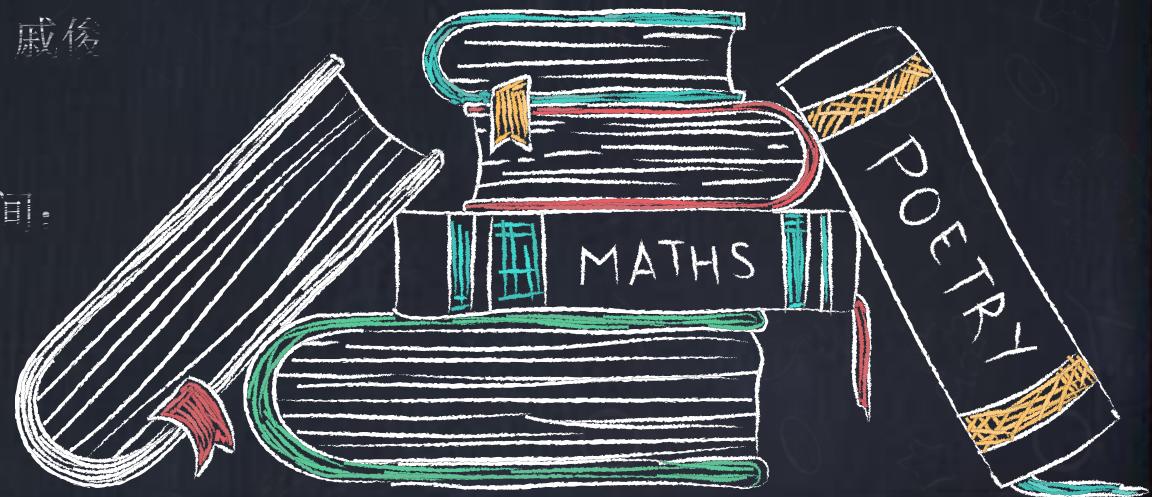
武汉科锐CR\_31  
2018-9-19



指导：戚俊



时间：



# 目 录



A function简介及语法



B function应用实例



C bind简介及语法



D bind应用实例



E 总结新特性的优点



# A. Function简介及语法

※ function简介

※ c11 新特性前后对比

※ function语法定义



# function简介

C11 新特性的前后对比

类模版std::function是一种通用、多态的函数封装。

std::function可以对任何可以调用的实体进行封装

程序设计，特别是程序库设计时，经常需要涉及到回调，如果针对每种不同的可调用对象单独进行声明类型，代码将会非常散乱，也不灵活。

std::function 对象是对C++中现有的可调用实体的一种**类型安全**的包裹

也就是说，通过std::function对C++中各种可调用实体的封装，形成一个新的可调用的std::function对象；让我们不再纠结那么多的可调用实体。一切变的简单粗暴。

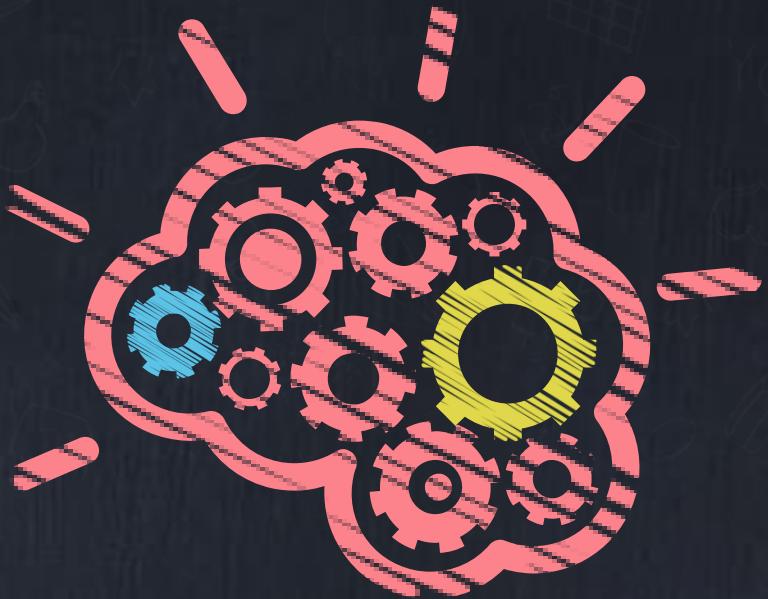


# function语法

简单介绍function定义

## function的头文件

如果想使用function必须  
包含#include <functional>



声明一个function时，需要给出所包装的函数对象的返回值类型和各个参数的类型。

## 声明示例

声明一个返回值为int，  
参数为两个int的 可调用 对象类型

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



# Function应用实例

※ 实例简介

※ 使用示例



# Function应用实例

## 各种实例简介

### 普通函数

这种函数定义比较简单，一般声明在一个文件开头

### 仿函数

又叫函数对象，是使用类来模拟函数调用行为，我们只要重载一个类的operator()方法，即可像调用一个函数一样调用类。这种方式用得比较少

### 类成员函数

在一个类class中定义的函数一般称为类的方法，分为成员方法和静态方法，区别是成员方法的参数列表中隐含着类this指针。（bind）

### 类静态函数

在一个类class中定义的函数一般称为类的方法，分为成员方法和静态方法，区别是成员方法的参数列表中隐含着类this指针。

# Function使用示例

## 五种示例

```
1 #include <functional>
2 #include <iostream>
3 using namespace std;
4
5 std::function< int(int)> Functional;
6
7 // 普通函数
8 int TestFunc(int a)
9 {
10     return a;
11 }
12
13 // Lambda 表达式
14 auto lambda = [](int a)->int{ return a; };
15
16 // 仿函数(functor)
17 class Functor
18 {
19 public:
20     int operator()(int a)
21     {
22         return a;
23     }
24 };
25
26 // 1. 类成员函数
27 // 2. 类静态函数
28 class TestClass
29 {
30 public:
31     int ClassMember(int a) { return a; }
32     static int StaticMember(int a) { return a; }
33 };
34
```

```
35 int main()
36 {
37     // 普通函数
38     Functional = TestFunc;
39     int result = Functional(10);
40     cout << "普通函数: "<< result << endl;
41
42     // Lambda 表达式
43     Functional = lambda;
44     result = Functional(20);
45     cout << "Lambda表达式: "<< result << endl;
46
47     // 仿函数
48     Functor testFunctor;
49     Functional = testFunctor;
50     result = Functional(30);
51     cout << "仿函数: "<< result << endl;
52
53     // 类成员函数
54     TestClass testObj;
55     Functional = std::bind(&TestClass::ClassMember, testObj, std::placeholders::_1);
56     result = Functional(40);
57     cout << "类成员函数: "<< result << endl;
58
59     // 类静态函数
60     Functional = TestClass::StaticMember;
61     result = Functional(50);
62     cout << "类静态函数: "<< result << endl;
63
64     return 0;
65 }
```

# Cbind简介及语法

※ bind前世

※ bind简介

※ bind语法



# I bind的前世今生

C98下的bind

## C98下的bind

C++98中，有两个函数bind1st和bind2nd，它们分别用来绑定functor的第一个和第二个参数，都只能绑定一个参数。

1

2

3

4

## Boost库

shared\_ptr等智能指针，  
bind及function

## C++STL库

C++STL很大一部分由  
Boost库扩充

## C++11下的bind

C++98提供的这些特性  
已经由于C++11的到来  
而过时，由于各种限制，  
我们经常使用bind而非  
bind1st和bind2nd

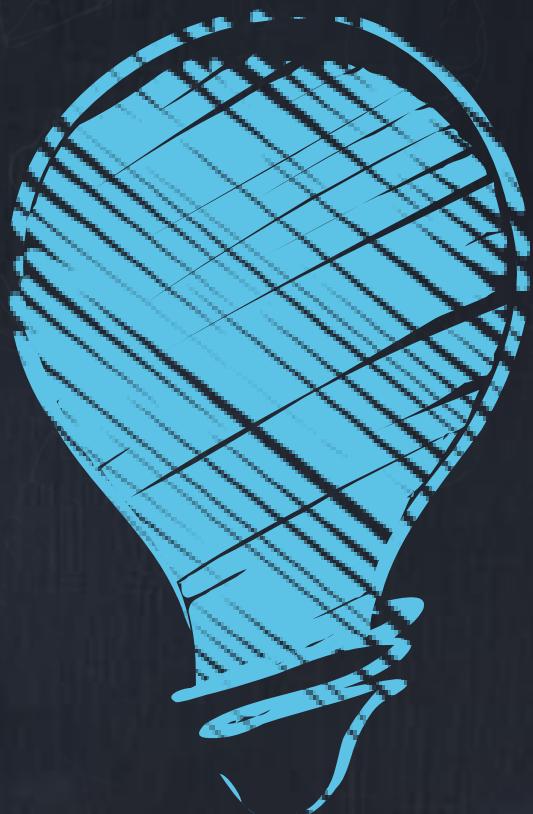
# Ibind简介

Std::bind绑定器

bind是这样一种机制，它可以预先把指定可调用实体的某些参数绑定到已有的变量，产生一个**新的可调用的实体**

bind就是函数适配器

函数适配器是用来让一个函数对象表现出另外一种类型的函数对象的特征。



绑定之后的结果可以使用 std::function进行保存，并延迟调用到任何需要的时候。一般来讲，它主要有两大作用：

- (1) 将可调用对象与其参数一起绑定成为一个仿函数
- (2) 将多元可调用对象转换成1元或是(n-1)元调用对象，既只是绑定部分参数

# I bind语法定义

语法定义及举例

## bind定义

```
auto newCallable = bind(callable,  
                        arg_list);
```

## 示例

```
void fun(int x, int y, int z)  
{ cout<<x<<" "<<y<<" "<<z<<endl; }  
auto f2 = std::bind(fun,  
placeholders::_1, placeholders::_2,  
3);  
//表示绑定函数 fun 的第三个参数为 3,  
而 fun 的第一, 二个参数分别有调用 f2  
的第一, 二个参数指定  
f2(1, 2); //print:1 2 3
```



# bind使用示例

※ 三个小例子



# bind使用示例

## 三个简单小例子

```
#include <iostream>
#include <functional>
using namespace std;

int TestFunc(int a, char c, float f)
{
    cout << a << endl;
    cout << c << endl;
    cout << f << endl;

    return a;
}

int main()
{
    auto bindFunc1 = bind(TestFunc, std::placeholders::_1, 'A', 100.1);
    bindFunc1(10);

    cout << "=====\n";

    auto bindFunc2 = bind(TestFunc, std::placeholders::_2, std::placeholders::_1, 100.1);
    bindFunc2('B', 10);

    cout << "=====\n";

    auto bindFunc3 = bind(TestFunc, std::placeholders::_2, std::placeholders::_3, std::placeholders::_1);
    bindFunc3(100.1, 30, 'C');

    return 0;
}
```



从左面的代码可以看到，  
bind能够在绑定时候  
同时绑定一部分参数，  
未提供的参数则使用占  
位符表示，然后在运行  
时传入实际的参数值。



# C11 新特性的优点

※ 安全性

※ 便捷性

※ 兼容性



# |新特性优点

安全性 便捷性 兼容性



## 安全性

`std::function`对象是对C++中现有的可调用实体的一种类型安全的包裹



## 便捷性

通过`std::function`对C++中各种可调用实体的封装，形成一个新的可调用的`std::function`对象



## 兼容性

对于使用C回调机制的程序库来说，C++的`std::function<>`能兼容传统C函数指针，使用`std::function<>`代替函数指针，并不会影响旧有客户端程序的编码方式



# 感谢您的聆听

wwwwww



本人能力有限，请多包涵



讲述人：李智博  
老师



武汉科锐CR\_31  
2018-9-19



指导：戚俊



时间：

