个人资料

coderchenjingui

访问： 138483次
积分： 2409
等级： BLOG 5
排名： 第7262名

原创： 94篇
转载： 11篇
译文： 0篇
评论： 48条

文章搜索

Markdown那么好，还不来试试 中国云计算大会最新议题 5月问答又送C币咯！ Hadoop实战高手速成宝典

std::function与std::bind 函数指针

分类： cpp 2013-01-22 20:37 25357人阅读 评论(5) 收藏 举报

function模板类和bind模板函数，使用它们可以实现类似函数指针的功能，但却却比函数指针更加灵活，特别是函数指向类 的非静态成员函数时。
std::function可以绑定到全局函数/类静态成员函数(类静态成员函数与全局函数没有区别),如果要绑定到类的非静态成员函数，则需要使用std::bind。

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

typedef std::function<void ()> fp;
void g_fun()
{
    cout<<"g_fun()"<<endl;
}
class A
{
public:
    static void A_fun_static()
    {
        cout<<"A_fun_static()"<<endl;
    }
    void A_fun()
    {
        cout<<"A_fun()"<<endl;
    }
    void A_fun_int(int i)
    {
        cout<<"A_fun_int() "<<i<<endl;
    }

    //非静态类成员，因为含有this指针，所以需要使用bind
```

```
27.     void init()
28.     {
29.         fp fp1=std::bind(&A::A_fun,this);
30.         fp1();
31.     }
32.
33.     void init2()
34.     {
35.         typedef std::function<void (int)> fpi;
36.         //对于参数要使用占位符 std::placeholders::_1
37.         fpi f=std::bind(&A::A_fun_int,this,std::placeholders::_1);
38.         f(5);
39.     }
40. };
41. int main()
42. {
43.     //绑定到全局函数
44.     fp f2=fp(&g_fun);
45.     f2();
46.
47.     //绑定到类静态成员函数
48.     fp f1=fp(&A::A_fun_static);
49.     f1();
50.
51.     A().init();
52.     A().init2();
53.     return 0;
54. }
```

同时，std::bind绑定到虚函数时会表现出多态行为。

```
[cpp]
01. #include <iostream>
02. #include <functional>
03. using namespace std;
04.
05. typedef std::function<void ()> fp;
06.
07. class A
08. {
09. public:
10.     virtual void f()
11.     {
12.         cout<<"A::f()"<<endl;
13.     }
14.
15.     void init()
16.     {
17.         //std::bind可以表现出多态行为
18.         fp f=std::bind(&A::f,this);
19.         f();
20.     }
```

```
};
class B:public A
{
public:
    virtual void f()
    {
        cout<<"B::f()"<<endl;
    }
};
int main()
{
    A* pa=new B;
    pa->init();

    return 0;
}
```


上一篇 警惕C++令人恼怒的解析

下一篇 用empty()代替size()==0

主题推荐 function namespace 指针 iostream 多态

猜你在找

有效使用 Lambda 表达式和 stdfunction

Cocos2d-x 3D开发六使用cocoStudio创建一个骨骼动画

抄袭事件判决书

手把手实现红黑树

Unity 3D--摄像机平滑跟随方法一

【精品课程】JavaScript for Qt Quick(QML)

【精品课程】Qt基础与Qt on Android入门

【精品课程】深入浅出Java的反射

【精品课程】太空大战游戏实战课程

【精品课程】Part 1：基础语言-Cocos2d-x手机游戏开发必备C++语言基础

准备好了么？跳吧 ！

更多职位尽在 CSDN JOB

iOS，Android，HTML5 程序员 我要跳槽

上海胜因软件技术有限公司 | 8-10K/月

iOS，Android，HTML5 程序员 我要跳槽

上海普望企业管理有限公司 | 8-10K/月

delphi工程师 我要跳槽

深圳硕软技术有限公司 | 10-20K/月

iOS，Android，HTML5 储备程序员 我要跳槽

上海普望企业管理有限公司 | 4-6K/月

推荐文章

* 2015博文大赛

*CSDN Markdown简明教程-基本使用

*CSDN Markdown简明教程-快速上手

*CSDN Markdown如何绘制UML图

*CSDN Markdown使用LaTeX编写数学公式

*CSDN Markdown扩展语法

最新评论

用empty()代替size()==0
3x3只眼: 其实个人觉得gcc对list.size()的这个折衷实在不讨好，大多数人用list的时候，用size...

信号槽传递非Qt库类型参数时，
chenfengyiranlalala: 感谢楼主的分享 评论也很有用 感谢两位

由Qt中qApp想到的(这是单例模式
henry_gyr: nice

Qt4.8.2和QtCreator，以及VS20(
LBD1744764093: 幸亏你有这个

编译器的概述
asd8532: @QQ575787460:thank you very much

编译器的概述
coderchenjingui: @u012840458:怎么总是发两条呢，csdn做的有点烂啊，我记得之前在论坛上总是会发出去两条。...

编译器的概述
asd8532: @QQ575787460:晚上再来

编译器的概述
coderchenjingui: @u012840458:你的list_reverse_print怎么写的，我写的是这样的：// "p...

编译器的概述
asd8532: @QQ575787460:我还以为没发送成功又打了一次

编译器的概述
asd8532: @QQ575787460:输出push 654566566这样的未定义的空间 来自于亮瞎狗眼的99...

查看评论

2楼 Tangbzh 2014-07-29 15:31发表

感谢分享

1楼 MrSimp1e 2014-05-04 13:29发表

function绑定类的静态函数也是可以的。例如

```
//
class View{

public:
void onClick(int x, int y)
{
cout<<"X : "<<x<<", Y : "<<y<<endl;
}
} ;

// 定义function类型, 三个参数
function<void (View*, int, int)> clickCallback ;

//
int main(int argc, const char * argv[])
{
View button ;
// 指向成员函数
clickCallback = &View::onClick ;
// 进行调用
clickCallback(&button, 10, 123);
return 0;
}
```

Re: liuzhihan209 2014-12-04 23:10发表

回复MrSimp1e：大哥呢，没试验，没发言权啊，你的这个程序在VS 2013 上是会报错的，你确定能直接这样绑定非静态成员.....自己去试试吧，难道是我错了？？？？

Re: 宁采臣 2014-12-08 18:59发表

回复liuzhihan209：改成以下的就可以了：

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

```
class View{

public:
void onClick(int x, int y)
{
cout << "X : " << x << ", Y : " << y << endl;
}
};

// 定义function类型, 三个参数
function<void(View, int, int)> clickCallback;

//
int main(int argc, const char * argv[])
{
View button;
// 指向成员函数
clickCallback = &View::onClick;
// 进行调用
clickCallback(button, 10, 123);
return 0;
}
```

去掉“&”取地址操作符，这样就可以了。

Re: 宁采臣 2014-12-08 18:57发表

回复liuzhihan209：我这里也不行。他的想法是对的，也就是添加一个this指针。但是，语法上是编译过不去的。

发表评论

用 户 名： zyp2524153

评论内容：

提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack VPN Spark ERP IE10
Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery BI HTML5 Spring Apache .NET API HTML SDK IIS
Fedora XML LBS Unity Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack FTC
coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo Compuware 大数据 aptech Perl
Tornado Ruby Hibernate ThinkPHP HBase Pure Solr Angular Cloud Foundry Redis Scala Django Bootstrap

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持

京 ICP 证 070598 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved