

饭后温柔


汉堡与老干妈同嚼 有可乐味

博客园 :: 首页 :: 新随笔 :: 联系 :: 订阅 XML :: 管理

posts - 63, comments - 48, trackbacks - 0, articles - 12

 搜索

谷歌搜索

 我的标签

c++ (14)

ogre (12)

编程 (7)

meta programming (5)

3d (5)

模板 (4)

多线程 (3)

网络 (3)

游戏 (3)

haskell (3)

更多

 随笔档案

2016年6月 (3)

2014年10月 (2)

2014年4月 (1)

2014年3月 (2)

2014年2月 (3)

2013年11月 (1)

2013年10月 (4)

2013年8月 (7)

2013年6月 (2)

2013年5月 (1)

2013年4月 (2)

2013年3月 (2)

2013年2月 (4)

2013年1月 (1)

2012年11月 (1)

2012年9月 (3)

2012年7月 (1)

2012年5月 (2)

2012年2月 (2)

2012年1月 (4)

2011年11月 (1)

2011年10月 (4)

2011年9月 (3)

2011年8月 (5)

2011年4月 (1)


2011年3月 (1)

 文章分类

拾人牙慧

星辰大海(2)

学海拾贝(12)

 最新评论

1. Re:坑爹的gltools编译错误解决
至今还没配好环境，我想放弃了
--RunningXie

2. Re:坑爹的gltools编译错误解决
我好几次碰到这个问题了，每次都纠结，都没有解决，好不开心。
--四叶留誓

3. Re:c++11 关于typelist的foreach
第二种办法好,因为第一种办法的实例是typelist里的,如果默认构造函数不可访问就会出错.不过写的似乎有些啰嗦
template class F>static void doTLs()
{ F>().....
--dimeiyi24

4. Re:c++11 function_traits备忘

使用c++11改写loki的TypeList

Posted on 2014-02-16 15:24 饭后温柔 阅读(1939) 评论(7) 编辑 收藏


最近看了C++11的一些特性，最感兴趣的是可变模板参数，自动类型推断和匿名函数。

Loki中的TypeList，是需要递归定义的，并且需要一个NullType作为尾节点。

可变模板参数使得实现TypeList更简洁，更易懂。

以下是我用C++11实现TypeList,其实只用了可变模板参数。

去掉了递归定义，特别是尾节点可直接使用typelist<>,使得整个语义很美。



```
////////////////////////////////////
template<typename... TList>
struct typelist
{
};

typedef typelist<> nulllist;

////////////////////////////////////
template<typename... TList> struct length;

template<typename... TList>
struct length< typelist<TList...> >
{
    enum { value = sizeof...(TList) };
};

////////////////////////////////////
template<typename T, typename... TList> struct push_front;

template<typename T, typename... TList>
struct push_front< T, typelist<TList...> >
{
    typedef typelist<T, TList...> type;
};

////////////////////////////////////
template<typename... TList> struct pop_front;

template<typename T, typename... TList>
struct pop_front< typelist<T, TList...> >
{
    typedef typelist<TList...> type;
};

template<>
struct pop_front< nulllist >
{
    typedef nulllist type;
};

////////////////////////////////////
template<unsigned int N, typename... TList> struct at;

template<unsigned int N, typename T, typename... TList>
struct at< N, typelist<T, TList...> >
{
    typedef typename at< N-1, typelist<TList...> >::type type;
};

template<typename T, typename... TList>
struct at< 0, typelist<T, TList...> >
{
    typedef T type;
};

template<>
struct at< 0, nulllist >
{
    typedef nulllist type;
};

////////////////////////////////////
template<int A, int B>
struct IndexFixer
{
    enum { value = (A == B) ? B : A + 1 };
};
```


可不可以这样呢template struct
function_traits:function_traits {
typedef T class_type;};...

--dimeiyi24

5. Re:ogre 扩展模型描绘轮廓及实现
自阴影明暗

楼主您好，请问我用了您上面的代
码，为什么阴影显示不出来，是不是
所有的实体都要继承base材质？请教
一下您对具体的模型是怎么处理的？

--aaamumuliang


-  阅读排行榜
1. 写给笨人的法线贴图原理(26301)

2. 一个有趣的模拟光照的shader(类似
法线贴图)(8334)

3. windows+apache+mysql+django+mod_...
安装(5816)

4. 法线从object space到eye space的
转换((normal matrix)(3315)

5. unity3d笔记二:基于组件的设计
(3278)


-  评论排行榜
1. 艾尔之光中刀光的ogre实现(15)

2. c++11 tuple实现(8)

3. 使用c++11改写loki的TypeList(7)

4. c++11 function_traits备忘(3)

5. 备忘:c#接口与抽象类(3)

-  推荐排行榜
1. 写给笨人的法线贴图原理(6)

2. 使用c++11改写loki的TypeList(4)

3. windows+apache+mysql+django+mod_...
安装(2)

4. 备忘:c#接口与抽象类(2)

5. c++11模拟boost元占位符
placeholder(2)

```
////////////////////////////////////  
template<typename T, typename... TList> struct indexof;  
  
template<typename T, typename H, typename... TList>  
struct indexof< T, typelist<H, TList...> >  
{  
    enum { value = IndexFixer<indexof<T, typelist<TList...>>::value, -1>::value };  
};  
  
template<typename T, typename... TList>  
struct indexof< T, typelist<T, TList...> >  
{  
    enum { value = 0 };  
};  
  
template<typename T>  
struct indexof< T, nulllist >  
{  
    enum { value = -1 };  
};  
  
////////////////////////////////////  
template<typename A, typename B> struct concat;  
  
template<typename... A, typename... B>  
struct concat<typelist<A...>, typelist<B...> >  
{  
    typedef typelist<A..., B...> type;  
};  
  
template<typename T, typename... TList>  
struct concat<typelist<TList...>, T >  
{  
    typedef typelist<TList..., T> type;  
};  
  
template<typename T, typename... TList>  
struct concat< T, typelist<TList...> >  
{  
    typedef typelist<T, TList...> type;  
};  
  
  
  
////////////////////////////////////  
template<typename T, typename... TList> struct erase;  
  
template<typename T, typename H, typename... TList>  
struct erase<T, typelist<H, TList...> >  
{  
    typedef typename concat<H, typename erase< T, typelist<TList...> >::type>::type type;  
};  
  
template<typename T, typename... TList>  
struct erase<T, typelist<T, TList...> >  
{  
    typedef typelist<TList...> type;  
};  
  
template<typename T>  
struct erase<T, nulllist >  
{  
    typedef nulllist type;  
};  
  
  
  
////////////////////////////////////  
template<typename T, typename... TList> struct erase_all;  
  
template<typename T, typename H, typename... TList>  
struct erase_all<T, typelist<H, TList...> >  
{  
    typedef typename concat<H, typename erase_all< T, typelist<TList...> >::type>::type type;  
};  
  
template<typename T, typename... TList>  
struct erase_all<T, typelist<T, TList...> >  
{  
    typedef typename erase_all< T,typelist<TList...> >::type type;  
};  
  
template<typename T>  
struct erase_all<T, nulllist >  
{  
    typedef nulllist type;  
};  
  
  
  
////////////////////////////////////  
template<typename T, typename...TList> struct no_duplicate;  
  
template<typename T, typename...TList>  
struct no_duplicate< typelist<T, TList...> >  
{  
private:
```

```
        typedef typename no_duplicate< typelist<TList...> >::type inner;
        typedef typename erase<T, inner>::type inner_result;
public:
        typedef typename concat<T, inner_result>::type type;
};

template<>
struct no_duplicate< nulllist >
{
        typedef nulllist type;
};

////////////////////////////////////
template<typename R, typename T, typename...TList> struct replace;

template<typename R, typename T, typename H, typename...TList>
struct replace<R, T, typelist<H, TList...> >
{
        typedef typename concat<H, typename replace<R, T, typelist<TList...>>::type>::type type;
};

template<typename R, typename H, typename...TList>
struct replace<R, H, typelist<H, TList...> >
{
        typedef typename concat<R, typelist<TList...> >::type type;
};

template<typename R, typename T>
struct replace<R, T, nulllist >
{
        typedef nulllist type;
};

////////////////////////////////////
template<typename R, typename T, typename...TList> struct replace_all;

template<typename R, typename T, typename H, typename...TList>
struct replace_all<R, T, typelist<H, TList...> >
{
        typedef typename concat<H, typename replace_all<R, T, typelist<TList...>>::type>::type type;
};

template<typename R, typename H, typename...TList>
struct replace_all<R, H, typelist<H, TList...> >
{
        typedef typename concat<R, typename replace_all<R, H, typelist<TList...>>::type >::type type;
};

template<typename R, typename T>
struct replace_all<R, T, nulllist >
{
        typedef nulllist type;
};


////////////////////////////////////
template<typename T, typename...TList> struct reverse;

template<typename T, typename...TList>
struct reverse<typelist<T, TList...> >
{
        typedef typename concat<typename reverse<typelist<TList...>>::type, T>::type type;
};

template<>
struct reverse< nulllist >
{
        typedef nulllist type;
};
```



例子：

```

#include <iostream>
#include <vector>
#include "TypeList.h"
#include <type_traits>

int main()
{
        typedef typelist<int, float, bool, float> testlist;
        typedef typelist<float, bool> tlist;
        typedef typelist<int, float> hlist;
        typedef typelist<> elist;
        typedef typelist<int> ilist;
        typedef testlist mylist;

        std::cout << "length: " << length<mylist>::value << std::endl;
        bool b;
        b = std::is_same<at<2, mylist>::type, bool>::value;
```

```
std::cout << "is same: " << b << std::endl;
b = std::is_same<push_front<int, elist>::type, ilist>::value;
std::cout << "is same: " << b << std::endl;
std::cout << "indexof : " << indexof<bool, mylist>::value << std::endl;
b = std::is_same<pop_front<typelist<>>::type, pop_front<ilist>::type>::value ;
std::cout << "is same: " << b << std::endl;
b = std::is_same< erase<bool, mylist>::type, typelist<int, float, float>>::value;
std::cout << "is same: " << b << std::endl;
b = std::is_same< no_duplicate<mylist>::type, typelist<int, float, bool>>::value;
std::cout << "is same: " << b << std::endl;
b = std::is_same< replace<double, bool, mylist>::type, typelist<int, float, double, float>>::value;
std::cout << "is same: " << b << std::endl;
b = std::is_same< replace_all<double, float, mylist>::type, typelist<int, double, bool, double>>::value;
std::cout << "is same: " << b << std::endl;
b = std::is_same< reverse<mylist>::type, typelist<float, bool, float, int>>::value;
std::cout << "is same: " << b << std::endl;

//std::cout << "is same: " << CompileTimeCount(int) << std::endl;
//std::cout << "is same: " << CompileTimeCount(int) << std::endl;
return 0;
}
```

标签: [c++](#), [meta programming](#)

好文要顶

关注我

收藏该文

饭后温柔
关注 - 7
粉丝 - 24
[+加关注](#)

4

0

« 上一篇 : [坑爹的gltools编译错误解决](#)
» 下一篇 : [一个基于typelist的typemap](#)

Feedback

#1楼 2014-02-16 20:38 by qicosmos(江南)	很好，给几个测试代码就更好了。 <div>支持(0) 反对(0)</div>
#2楼 2014-02-17 10:04 by john23.net	感谢分享 <div>支持(0) 反对(0)</div>
#3楼 2014-02-17 13:48 by qicosmos(江南)	<p>size和at函数可以改进一点，改得更简单一点：</p> <pre>1 template< typename... TYPES > struct size {}; 2 template< typename... TYPES > struct size< typelist<TYPES...> > : std::integral_constant<int, sizeof...(TYPES)> 3 { 4 //enum { value = sizeof...(TYPES) }; 5 }; 1 template< unsigned int N, typename... TYPES > struct at {}; 2 template< unsigned int N, typename... TYPES > struct at< N, typelist<TYPES...> > 3 { 4 typedef typename std::tuple_element< N, std::tuple<TYPES...> >::type type; 5 }; </pre> <div>支持(0) 反对(0)</div>
#4楼[楼主] 2014-02-17 14:21 by 饭后温柔	@ qicosmos 不希望引入std的tuple。因为没看他里面得实现。tuple似乎可以存值的，不知会不会引入一些问题，typelist应该只是编译期的东西。能一目了然的比较好。 <div>支持(0) 反对(0)</div>
#5楼	

2014-02-22 09:31 by qicosmos(江南)
<div>有了可变参数模板和tuple甚至variant，typelist可以被取代了，我没想到到真正需要typelist的应用场景。</div> <div>我的博客</div> <div>http://www.cnblogs.com/qicosmos/p/3559424.html</div> <div>欢迎加入c++11和boost技术讨论qq群：296561497</div> <div>支持(0) 反对(0)</div>
#6楼[楼主]
2014-02-22 20:54 by 饭后温柔
<div>@ qicosmos</div> <div>写着玩，当练习。typelist是mpl的基础构件。库的设计者会用到。</div> <div>支持(0) 反对(0)</div>
#7楼
2014-12-05 14:04 by RonTang
<div>tuple可以充当typelist使用，但显然tuple的功能更强大，因为无论编译期还是运行期均可使用。楼主说的对，tuple可以保存数值，因此tuple的实现远远比typelist复杂。tuple除了需要保存类型，还需保存具体值。当然tuple的实现采用也是采用变长模板参数，模板特化与偏特化，多重继承等技术。感谢楼主分享</div> <div>支持(0) 反对(0)</div>

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

【推荐】50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库

【推荐】中铁、中石油等大型企业的复杂报表解决方案

【福利】阿里云免费套餐升级，更多产品，更久时长



最新IT新闻:

· 慰抚开发团队成员丧妻之痛：Linux Mint 18.2取名为“Sonya”

· 被诉欺诈：验血公司Theranos与投资者和解、退款并赔偿

· Xbox部门老大：微软旗下工作室应该勇于创新 拓展行业边界

· Bing地图再次“开疆扩土”，新增土耳其、希腊和阿根廷地区地图数据

· 世界上第一本物理教材被拍出79万美元高价

» 更多新闻...



最新知识库文章:

· 唱吧DevOps的落地，微服务CI/CD的范本技术解读

· 程序员，如何从平庸走向理想？

· 我为什么鼓励工程师写blog

· 怎么轻松学习JavaScript

· 如何打好前端游击战

» 更多知识库文章...