微博:
@TankyWoo基

新博客:
TankyWoo

Tanky Woo的前博客

C++ 重载(overload)、重写(overrride)、重定义(redefine)总结

昨晚打开论坛，看到有朋友问了一个关于虚函数的问题，因为头太疼了，所以今天中午起床再看。

问题传送门：http://www.cppleyuan.com/viewthread.php?tid=7923

C++的一些特性好久没使用了，导致有些生疏了，所以查了一些资料，回顾了一下。

先来看几个概念：

重载(overload)，重写(override,也称覆盖), 重定义(redefine,也称隐藏)

(PS:第三个我不确定在英文中是否应该称为redefine，如有问题，留言告知，谢谢)

公告

公告:

昵称：Tanky Woo
园龄：4年7个月
荣誉：推荐博客
粉丝：332

一、重载（overload）

指函数名相同，但是它的参数表列个数或顺序，类型不同。但是不能靠返回类型来判断。

（1）相同的范围（在同一个作用域中）；

（2）函数名字相同；

（3）参数不同；

（4）virtual 关键字可有可无。

（5）返回值可以不同；

二、重写（也称为覆盖 override）

是指派生类重新定义基类的虚函数，特征是：

（1）不在同一个作用域（分别位于派生类与基类）；

（2）函数名字相同；

（3）参数相同；

（4）基类函数必须有 virtual 关键字，不能有 static 。

（5）返回值相同（或是协变），否则报错；<—-协变这个概念我也是第一次才知道…

（6）重写函数的访问修饰符可以不同。尽管 virtual 是 private 的，派生类中重写改写为 public,protected 也是可以的

三、重定义（也成隐藏）

（1）不在同一个作用域（分别位于派生类与基类）；

（2）函数名字相同；

（3）返回值可以不同；

（4）参数不同。此时，不论有无 virtual 关键字，基类的函数将被隐藏（注意别与重载以及覆盖混淆）。

（5）参数相同，但是基类函数没有 virtual关键字。此时，基类的函数被隐藏（注意别与覆盖混淆）。

OK,这里给出一个样例代码，是论坛那个问题的代码做了一些修改，方便理解：

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

class Base
{
public:
    virtual void a(int x)    {    cout << "Base::a(int)" << endl;      }
    // overload the Base::a(int) function
    virtual void a(double x) {    cout << "Base::a(double)" << endl;   }
    virtual void b(int x)    {    cout << "Base::b(int)" << endl;      }
    void c(int x)            {    cout << "Base::c(int)" << endl;      }
};

class Derived : public Base
{
public:
    // redefine the Base::a() function
    void a(complex<double> x)   {    cout << "Derived::a(complex)" << endl;      }
    // override the Base::b(int) function
    void b(int x)               {    cout << "Derived::b(int)" << endl;          }
    // redefine the Base::c() function
    void c(int x)               {    cout << "Derived::c(int)" << endl;          }
};

int main()
{
    Base b;
    Derived d;
```

最新评论
我的标签

最新随笔

1. C++ 重载(overload)、重写(overrride)、重定义(redefine)总结
2. 请不要忘记，还有“思考”这个东西
3. 《算法导论》学习总结 — 21.第16章 贪心算法(1) 基础入门1
4. 《算法导论》学习总结 — 20.第15章 动态规划(5) 分析几道DP题
5. 谈一下ACM的入门书籍及方法
6. 《算法导论》学习总结 — 19.第15章 动态规划(4) 案例之LCS
7. 《算法导论》学习总结 — 18.第15章 动态规划(3) 基础入门2
8. 《算法导论》学习总结 — 17.第15章 动态规划(2) 案例之装配线调度
9. 《算法导论》学习总结 — 16.第15章 动态规划(1) 基本入门
10. 《算法导论》学习总结 — 15. 第13章 红黑树(4)

我的标签

01背包(1)
HDOJ(1)
背包(1)
多重背包(1)
母函数(1)
求素数(1)
生成函数(1)
帅选法(1)
水题(1)
完全背包(1)
更多

随笔分类(32)

```
30     Base* pb = new Derived;
31     // ---------------------------------- //
32     b.a(1.0);                              // Base::a(double)
33     d.a(1.0);                              // Derived::a(complex)
34     pb-
>a(1.0);                            // Base::a(double), This is redefine the Base::a() functi
35     // pb->a(complex<double>(1.0, 2.0));   // clear the annotation and have a try
36     // ---------------------------------- //
37     b.b(10);                               // Base::b(int)
38     d.b(10);                               // Derived::b(int)
39     pb-
>b(10);                            // Derived::b(int), This is the virtual function
40     // ---------------------------------- //
41     delete pb;
42
43     return 0;
44 }
```

通过这里可以看出：

1.Base类中的第二个函数a是对第一个的重载

2.Derived类中的函数b是对Base类中函数b的重写，即使用了虚函数特性。

3.Derived类中的函数a是对Base泪中函数a的隐藏，即重定义了。

4.pb指针是一个指向Base类型的指针，但是它实际指向了一个Derived的空间，这里对pd调用函数的处理(多态性)取决于是否重写(虚函数特性)了函数，若没有，则依然调用基类。

5.只有在通过基类指针或基类引用 间接指向派生类类型时多态性才会起作用。

6.因为Base类的函数c没有定义为virtual虚函数，所以Derived类的函数c是对Base::c()的重定义。

C/C++基础(3)
C++标准程序库(5)
算法分析(24)

随笔档案(61)

2012年2月 (1)
2011年9月 (1)
2011年6月 (3)
2011年5月 (9)
2011年4月 (10)
2011年1月 (8)
2010年12月 (8)
2010年11月 (3)
2010年9月 (8)
2010年8月 (7)
2010年7月 (3)

我的独立博客

Tanky Woo的程序人生

这是我的个人独立博客，请大家多多支持。

积分与排名

积分 - 102912
排名 - 1487

最新评论

1. Re:字典树（讲解+模版）

如果用左孩子右兄弟法表示就可以解决储存空间浪费的问题了

--hehe_xixi_haha

2. Re:背包之01背包、完全背包、多重背包详解

感觉写的好详细，但还是要自己慢慢来研究了

网上参考资料：

1.http://sns.linuxpk.com/blog-6583-17085.html

2.http://www.cnblogs.com/xd502djj/archive/2010/09/22/1832912.html

3.http://blog.sina.com.cn/s/blog_6ae7d6b00100pb4v.html

4.http://topic.csdn.net/u/20110227/23/42d93b05-03b1-460b-8521-707117ce5600.html

5.http://www.cnblogs.com/realyan/archive/2011/07/14/2106339.html

6.http://topic.csdn.net/u/20081023/18/122ac3bd-8ad2-4e6e-8624-090f22c82139.html

个人博客原文链接：http://www.wutianqi.com/?p=3171

分类: C/C++基础

Tanky Woo
关注 - 32
粉丝 - 332

荣誉：推荐博客

+加关注

1 0

(请您对文章做出评价)

« 上一篇：请不要忘记，还有“思考”这个东西

posted on 2012-02-08 20:45 Tanky Woo 阅读(5111) 评论(3) 编辑 收藏

评论

#1楼 2012-02-11 21:20 YDuck

我个人觉得从函数的实际调用来看将会更好区分理解：
函数重载情况：同一作用域，寻找适当函数的过程
函数重写情况：比较特殊，用于父类与子类之间的虚函数（虚函数是通过virtual table来实现的，在具体的实现时父类函数的指针会被子类相同的函数指针所覆盖，所以才称为override）
函数重定义情况（个人觉得称为隐藏更恰当）：函数调用时，在两个不同作用域（大的作用域包含小的作用域情况

--tianxia2s

3. Re:字典树（讲解+模版）

在 findTrie当中if(p->v == -1) //字符集中已有串是此串的前缀 return -1;这里有问题吧。比如我字典树当中就有两个单词a和b。当你查找单词ab的时候，肯定会检索到叶子.....

--wzStyle

4. Re:线段树(区间树)Segment Tree

线段树和区间树不一样……见算法导论……

--usafchn

5. Re:C++ 重载(overload)、重写(overrride)、重定义(redefine)总结

#include #include using namespace std;class Base{public: virtual void b(int x = 0) { cout b(); .....

--武右非戈

阅读排行榜

1. 字典树（讲解+模版）(14773)
2. 《算法导论》学习总结 --- 1.前言(14101)
3. 谈一下ACM的入门书籍及方法(10455)
4. C++输入一行字符串的一点小结(9971)
5. 组合博弈知识汇总（算法）(9918)

评论排行榜

1. 一些计算机编程的经典书籍总结（大家一起来补充！）(48)
2. 我的编程学习经历–Tanky Woo(32)
3. 请不要忘记，还有“思考”这个东西(30)
4. 《算法导论》学习总结 --- 1.前言(27)
5. 随机化算法(1) — 随机数(20)

下），在小的作用域中寻找到合适的函数后直接调用，不用再在大的作用域中搜索，故可以称为隐藏

支持(0) 反对(0)

#2楼[楼主] 2012-02-12 14:24 Tanky Woo

@YDuck
嗯，其实重定义是一个很简单的概念，比如在定义变量时，在小范围作用域中隐藏大范围作用域的变量。不过在有了类以及虚函数后，很多人就容易把重写与重定义弄混，其实主要就是一个虚函数这个特性。

支持(0) 反对(0)

#3楼 2014-06-03 12:39 武右非戈

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

class Base
{
public:
    virtual void b(int x = 0)    { cout << "Base::b(int)" << x << endl; }
};

class Derived : public Base
{
public:
    // override the Base::b(int) function
    void b(int x = 5)              { cout << "Derived::b(int)" << x << endl; }
};

int main()
{
    Base* pb = new Derived;
    pb->b();                            // Derived::b(int)0, This is the virtual function

    delete pb;
    cin.get();
```

```
25      return 0;
26  }
```

我想问一下，为什么调用子类的b()函数，但输出的x的值是0？
0应该是父类里的

支持(0) 反对(0)

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。