关于C++中的虚拟继承的一些总结

1.为什么要引入虚拟继承

虚拟继承是多重继承中特有的概念。虚拟基类是为解决多重继承而出现的。如:类D继承自类B1、B2，而类B1、B2都继承自类A，因此在类D中两次出现类A中的变量和函数。为了节省内存空间，可以将B1、B2对A的继承定义为虚拟继承，而A就成了虚拟基类。实现的代码如下：

```
class A

class B1:public virtual A;

class B2:public virtual A;

class D:public B1,public B2;
```

虚拟继承在一般的应用中很少用到，所以也往往被忽视，这也主要是因为在C++中，多重继承是不推荐的，也并不常用，而一旦离开了多重继承，虚拟继承就完全失去了存在的必要因为这样只会降低效率和占用更多的空间。

2.引入虚继承和直接继承会有什么区别呢

由于有了间接性和共享性两个特征，所以决定了虚继承体系下的对象在访问时必然会在时间和空间上与一般情况有较大不同。

2.1时间：在通过继承类对象访问虚基类对象中的成员（包括数据成员和函数成员）时，都必须通过某种间接引用来完成，这样会增加引用寻址时间（就和虚函数一样），其实就是调整this指针以指向虚基类对象，只不过这个调整是运行时间接完成的。

2.2空间：由于共享所以不必要在对象内存中保存多份虚基类子对象的拷贝，这样较之多继承节省空间。虚拟继承与普通继承不同的是，虚拟继承可以防止出现diamond继承时，一个派生类中同时出现了两个基类的子对象。也就是说，为了保证这一点，在虚拟继承情况下，基类子对象的布局是不同于普通继承的。因此，它需要多出一个指向基类子对象的指针。

3.笔试，面试中常考的C++虚拟继承的知识点

```
第一种情况：
class a
{
    virtual void func();
};
class b:public virtual a
{
    virtual void foo();
};
```

```
第二种情况：
class a
{
    virtual void func();
};
class b :public a
{
    virtual void foo();
};
```

```
第三种情况
class a
{
    virtual void func();
    char x;
};
class b:public virtual a
{
    virtual void foo();
};
```

```
第四种情况：
class a
{
    virtual void func();
    char x;
};
class b:public a
{
    virtual void foo();
};
```

如果对这四种情况分别求sizeof(a)，sizeof(b)。结果是什么样的呢？下面是输出结果：（在vc6.0中运行）

第一种：4，12

第二种：4，4

第三种：8，16

第四种：8，8

想想这是为什么呢？

因为每个存在虚函数的类都要有一个4字节的指针指向自己的虚函数表，所以每种情况的类a所占的字节数应该是没有什么问题的，那么类b的字节数怎么算呢？看“第一种”和“第三种”情况采用的是虚继承，那么这时候就要有这样的一个指针vptr_b_a，这个指针叫虚类指针，也是四个字节；还要包括类a的字节数，所以类b的字节数就求出来了。而“第二种”和“第四种”情况则不包括vptr_b_a这个指针，这回应该木有问题了吧。

4.c++重载、覆盖、隐藏的区别和执行方式

既然说到了继承的问题，那么不妨讨论一下经常提到的重载，覆盖和隐藏

4.1成员函数被重载的特征

（1）相同的范围（在同一个类中）；
（2）函数名字相同；
（3）参数不同；
（4）virtual 关键字可有可无。

4.2“覆盖”是指派生类函数覆盖基类函数，特征是：

（1）不同的范围（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。

4.3“隐藏”是指派生类的函数屏蔽了与其同名的基类函数，特征是：

（1）如果派生类的函数与基类的函数同名，但是参数不同，此时，不论有无virtual关键字，基类的函数将被隐藏（注意别与重载混淆）。
（2）如果派生类的函数与基类的函数同名，但是参数相同，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）。

小结：说白了就是如果派生类和基类的函数名和参数都相同，属于覆盖，这是可以理解的吧，完全一样当然要覆盖了；如果只是函数名相同，参数并不相同，则属于隐藏。

4.4 三种情况怎么执行：

4.4.1 重载：看参数。

4.4.2 隐藏：用什么就调用什么。

4.4.3 覆盖：调用派生类。

学习中的一点总结，欢迎拍砖哦^^

分类: C++

标签: C++中的虚继承

3. Re:设计模式之中介者模式（Mediator）
易懂！
--街角_祝福
4. Re:设计模式之建造者模式（Builder）
讲得不错！
--街角_祝福

VincentCZW
关注 - 8
粉丝 - 164
+加关注

11 0

(请您对文章做出评价)

« 上一篇：关于C++中的友元函数的总结
» 下一篇：电梯调度算法

posted on 2012-06-05 23:29 VincentCZW 阅读(14505) 评论(13) 编辑 收藏

Comments

#1楼
rhinoceros
Posted @ 2012-06-06 11:14
4.3(2)应该是相同

支持(0) 反对(0)

#2楼[楼主]
VincentCZW
Posted @ 2012-06-06 12:31
@rhinoceros
嗯，对的，写错了，谢谢帮我指出来哦，要不还得影响其他看的人。
呵呵……

支持(0) 反对(0)

#3楼
油炸西瓜
Posted @ 2012-10-04 15:35
博主总结得很好，赞。
只不过，我用G + +运行上面的四个例子，发现结果跟楼主的很不一样啊。
如果笔试考到这类题真是要歇菜了，还得清楚各种编译器对virtual继承的实现方式。。。

支持(0) 反对(0)

#4楼
bizhu
Posted @ 2012-10-05 21:27
“看“第一种”和“第三种”情况采用的是虚继承，那么这时候就要有这样的一个指针vptr_b_a，这个指针叫虚类指针，也是四个字

节；还要包括类a的字节数，所以类b的字节数就求出来了”不是只多了四个字节吗？第一种为什么b是12呀，而不是4+4= 8，第二种也一样，为什么不是8+4 = 12,？

支持(2) 反对(0)

#5楼
听松馆少主
Posted @ 2014-05-04 14:34
刚刚昨晚实验。。。 64位linux gcc
8,8
8,8
16,24
16,16

支持(0) 反对(0)

#6楼
听松馆少主
Posted @ 2014-05-04 14:54
知道原因了，对齐的原因

支持(0) 反对(0)

#7楼
月冷风和霜
Posted @ 2014-05-06 11:05
很赞

支持(0) 反对(0)

#8楼
灿子
Posted @ 2014-05-22 21:06
为什么在我机器上测得的第一种跟第三种情况都不一样呢？
第一种：4 4
第三种：8 12

支持(0) 反对(0)

#9楼
拨浪鼓儿jay
Posted @ 2014-08-28 10:25
大可不必纠结得到的结果和博主的不一样，不同操作系统，不同位数，不同编译器结果基本都不一样，博主的可以通过vs 提供的d1reportSingleClassLayout[classname]去查看内存结构，一般基类都没问题（64位系统指针占8个字节），对于子类：
第一个：vfptr(b:foo)+vbptr+vfptr(a:func)=12
第二个：vfptr(a:func, b:foo)=4
第三个：vfptr(b:foo)+vbptr+vfptr(a:func)+x(对齐为四个字节)=16

第四个：vfptr(a:func, b:foo)+x(对齐为四个字节)=8
这里没有类之间的函数覆盖，如果有会更麻烦一点。

支持(0) 反对(0)

#10楼
subFire
Posted @ 2014-10-15 11:09
@拨浪鼓儿jay
正解，确是如此！

支持(0) 反对(0)

#11楼
subFire
Posted @ 2014-10-15 11:11
可以参考 http://blog.csdn.net/wxc1987821/article/details/5958325

支持(0) 反对(0)

#12楼
wangjie&xq
Posted @ 2014-10-31 15:54
@拨浪鼓儿jay
你好，我是个C++初学者，我想问下在第一种情况里，虚继承之后，基类的虚函数表和子类的虚函数表是独立的两个表，所以要用两个指针；而普通的继承，子类的虚函数表会和基类的虚函数表合并成一个表，所以用一个指针。所以才出现第一种情况12字节，第二种4字节吗？

支持(0) 反对(0)

#13楼
追风筝的小蜗牛
Posted @ 2014-12-27 23:01
@以上，其实这个问题，归根到底是虚继承和继承的真实差异问题。
虚继承：子类has虚基类，所以带来的问题是虚基类必须先于子类构造，虚基类要有独立性，如拥有虚函数表指针。
一般继承：子类is基类，属于关系，虚函数表自然可以共享。
ps：覆盖的问题也是虚基类为了优化子类存储空间的一种编译器解决方法尝试。

支持(0) 反对(0)

» 更多知识库文章...