C++多态

2016年11月5日 15:08

1.静态多态：在编译时根据函数参数的类型、个数、函数返回值不同来判断调用哪个函数。

2.动态多态：运行时，必须是有继承关系，函数参数一样时。使用虚函数，在同函数名前面加上 virtual关键字：（virtual double Area();）解决父类指针调用不了子类多态函数问题。

注：

- 析构函数只执行父类析构函数解决方法：用虚析构函数，父类和子类的析构函数前加virtual关键字
- 虚继承方式，即在public前加virtual关键字（class Worker::virtual public person）解决多继承类继承的重复

virtual在函数中的使用限制

1.修饰的函数必须是某个类的成员函数，不能是全局函数。
2.不能修饰静态函数（静态成员变量虽然在类中,但它并不是随对象的建立而分配空间的,也不是随对象的撤销而释）：static void circle（）；
3。不能修饰内联函数（inline int eat（）；内联函数）
4.不能修饰构造函数

```
class Shape
{
public:
   virtual double calcArea ()              //虚函数
    {return 0;}
   virtual double calcPerimeter () = 0;  //纯虚函数
   ......
};
```

抽象类：带有纯虚函数的类

❖ 接口类：

仅含纯虚函数的类，没有数据成员

不需要.cpp文件，不需要写构造函数这些

❖ RTTI:运行时类型识别

```
void doSomething(Flyable *obj)
{
   obj->takeoff();
   cout << typeid(*obj).name() << endl;
   if(typeid(*obj) == typeid(Bird))
   {
        Bird *bird = dynamic_cast<Bird *>(obj);
        bird->foraging();
   }
   obj->land();
```

❖ dynamic_cast

将一个基类对象指针（或引用）cast到继承类指针，dynamic_cast会根据基类指针是否真正指向继承类指针来做相应处理

dynamic_cast注意事项：

- 只能应用于指针和引用的转换
- 要转换的类型中必须包含虚函数
- 转换成功返回子类的地址，失败返回NULL

typeid注意事项：

- type_id返回一个type_info对象的引用
- 如果想通过基类的指针获得派生类的数据类型，基类必须带有虚函数
- 只能获取对象的实际类型

❖ 捕获异常处理：

```
void test()
{
    throw IndexException();
}

int main(void)
{
    try
    {
        test();
    }
    catch (IndexException &e)
    {
        e.printException();
    }
```