在C++中，创建派生类对象时，如果基类的构造函数中调用虚函数，则肯定调用基类的函数，原因是执行基类的构造函数时，派生类对象还未创建。

在C#中，情况有所不同。

// Copyright 2016.刘珅珅

// author：刘珅珅

// 在构造函数中调用虚函数

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace inherit\_class\_test5

{

class Base

{

public Base()

{

Show();

}

public virtual void Show()

{

Console.WriteLine("Show Base");

}

}

class Derived : Base

{

public Derived()

{

}

public override void Show()

{

Console.WriteLine("Show Derived");

}

}

class InheritClassTest

{

static void Main()

{

Derived obj = new Derived();

}

}

}

输出结果为：

Show Derived

原因是：在基类构造函数中调用虚方法时，C#会查找派生类中是否重写了Show()方法，如果重写了就会调用派生类的方法。

即使在派生类的重写方法中改变了派生类的私有成员变量，仍然为实现上述的调用。

例：程序inherit\_class\_test6

// Copyright 2016.刘珅珅

// author：刘珅珅

// 在基类构造函数中调用虚函数

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace inherit\_class\_test6

{

class Base

{

private int b;

public Base()

{

Show();

}

public virtual void Show()

{

b = 3;

Console.WriteLine("Show Base " + b);

}

}

class Derived : Base

{

public int d;

public Derived()

{

d = 5;

}

public override void Show()

{

d = 4;

Console.WriteLine("Show Derived " + d);

}

}

//public class Base

//{

// public Base()

// {

// System.Console.WriteLine("Base.Base");

// ABitDangerousCall();

// }

// public virtual void ABitDangerousCall()

// {

// System.Console.WriteLine("Base.ABitDangerousCall");

// }

// private class Inner

// {

// public Inner()

// {

// System.Console.WriteLine("Base.Inner.Inner");

// }

// }

// private Inner inner = new Inner();

//}

//class Derived : Base

//{

// public Derived()

// {

// System.Console.WriteLine("Derived.Derived");

// ctorInitializedMember = 5;

// }

// // ctorInitializedMember is default initialized to zero before the constructor initializes it.

// private int ctorInitializedMember;

// private int derivedInt = 6;

// public override void ABitDangerousCall()

// {

// System.Console.WriteLine(String.Format("Derived.ABitDangerousCallctorInitializedMember={0} derivedInt={1}", ctorInitializedMember, derivedInt));

// }

// private class Inner

// {

// public Inner()

// {

// System.Console.WriteLine("Derived.Inner.Inner");

// }

// }

// private Inner inner = new Inner();

//}

class InheritClassTest

{

static void Main(string[] args)

{

Derived d = new Derived();

Console.WriteLine(d.d); // 5

}

}

}

输出结果为：

Show Derived 4

5