在C#中，没有拷贝构造函数，也不允许赋值运算符=重载。用户可以定义一个与C++类似的拷贝构造函数，但它与其他的构造函数没有什么不同。

例：程序copy\_test1

// Copyright 2016.刘珅珅

// author：刘珅珅

// 拷贝对象

using *System*;

using *System*.*Collections*.*Generic*;

using *System*.*Linq*;

using *System*.*Text*;

using *System*.*Threading*.*Tasks*;

namespace copy\_test1

{

public class MyClass

{

public int n;

public MyClass(int k)

{

*Console*.*WriteLine*("MyClass(int)");

n = k;

}

public MyClass(MyClass rhs)

{

*Console*.*WriteLine*("MyClass(MyClass)");

n = rhs.n;

}

}

class CopyTest

{

static void Main(string[] args)

{

MyClass obj = new MyClass(5);

MyClass obj1 = new MyClass(obj);

MyClass copy = obj; // 不会调用构造函数

}

}

}

输出结果为：

MyClass(int)

MyClass(MyClass)

通过实现ICloneable接口，可以实现对象的拷贝。需要实现object Clone()接口函数。

深拷贝：副本和源对象完全独立，如果源对象包含对另一个对象O的引用，进行深拷贝

例：程序copy\_test2

// Copyright 2016.刘珅珅

// author：刘珅珅

// 类的拷贝

using *System*;

using *System*.*Collections*.*Generic*;

using *System*.*Linq*;

using *System*.*Text*;

using *System*.*Threading*.*Tasks*;

namespace copy\_test2

{

class X

{

public int a;

public X(int x)

{

*Console*.*WriteLine*("construct X.");

a = x;

}

}

// 深拷贝

class MyDeepCopyClass : *ICloneable*

{

public X obj;

public int b;

public MyDeepCopyClass(int x, int y)

{

obj = new X(x);

b = y;

}

public void Show(string name)

{

*Console*.*Write*(name + " values are ");

*Console*.*WriteLine*("obj.a : {0}, b: {1}", obj.a, b);

}

public object Clone()

{

MyDeepCopyClass temp = new MyDeepCopyClass(obj.a, b);

return temp;

}

}

// 浅拷贝

class MyCopyClass : *ICloneable*

{

public X obj;

public int b;

public MyCopyClass(int x, int y)

{

obj = new X(x);

b = y;

}

public void Show(string name)

{

*Console*.*Write*(name + " values are ");

*Console*.*WriteLine*("obj.a : {0}, b: {1}", obj.a, b);

}

public object Clone()

{

MyCopyClass temp = (MyCopyClass) *MemberwiseClone*();

return temp;

}

}

class CopyTest

{

static void Main(string[] args)

{

*Console*.*WriteLine*("Deep copy.");

MyDeepCopyClass obj1 = new MyDeepCopyClass(10, 20);

obj1.Show("obj1");

// 深拷贝

MyDeepCopyClass obj2 = (MyDeepCopyClass)obj1.Clone();

obj2.Show("obj2");

obj1.obj.a = 99;

obj1.b = 88;

*Console*.*WriteLine*();

obj1.Show("obj1");

obj2.Show("obj2");

*Console*.*WriteLine*();

*Console*.*WriteLine*("Copy.");

MyCopyClass obj3 = new MyCopyClass(30, 40);

obj3.Show("obj3");

// 浅拷贝

MyCopyClass obj4 = (MyCopyClass)obj3.Clone();

obj4.Show("obj4");

obj3.obj.a = 111;

obj3.b = 222;

*Console*.*WriteLine*();

obj3.Show("obj3");

obj4.Show("obj4");

}

}

}

输出结果：

Deep copy.

construct X.

obj1 values are obj.a : 10, b: 20

construct X.

obj2 values are obj.a : 10, b: 20

obj1 values are obj.a : 99, b: 88

obj2 values are obj.a : 10, b: 20

Copy.

construct X.

obj3 values are obj.a : 30, b: 40

obj4 values are obj.a : 30, b: 40

obj3 values are obj.a : 111, b: 222

obj4 values are obj.a : 111, b: 40

从输出可以看出，在深拷贝类MyDeepCopyClass中包含一个类X的引用，在拷贝过程中，会创建这个类X的副本，用来创建MyDeepCopyClass的副本，类X的构造函数执行了两次。改变原始对象obj1中的类X的值，不会影响到副本对象obj2中的类X对象。

在浅拷贝类MyCopyClass中包含一个类X的引用，在Clone()函数中调用系统的MemberwiseClone()，这是一个浅拷贝函数，原始对象obj3中的类X的引用和拷贝对象obj4中的类X都引用同一个类X对象。类X的构造函数也只执行了一次。所以改变原始对象obj3中的类X的值，会影响拷贝对象obj4中的类X的值，对于MyCopyClass中的字段b的值，obj3改变b的值不会影响到obj4的值，因为b为值类型，不通过引用访问。