dynamic类型：

dynamic类型与object类型类似，可以用于引用任何类型的对象。

Object类与dynamic的主要不同在于，所有与dynamic类型相关的类型检查都延迟到运行时执行（而object类则在编译时进行类型检查）。

使用dynamic类型，如果运行时该对象的实际类型不支持之前声明的操作，就会出现异常。

dynamic类型在使用反射、访问COM对象或者与动态语言互操作时非常有用。

在生成dll时，一定要注意，dll中的类的访问类型要为public，否则无法在其它程序集中访问类的方法。

使用dynamic类型时，要在工程中添加Microsoft.CSharp引用。

例：程序dynamic\_test1

// Copyright 2016.刘珅珅

// author：刘珅珅

// dynamic用于反射

using System;

using System.Collections.Generic;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Threading.Tasks;

namespace dynamic\_test1

{

class DynamicTest

{

static void Main(string[] args)

{

Assembly asm = Assembly.LoadFrom("dll\_test1.dll");

Type[] all = asm.GetTypes();

int i;

for ( i = 0; i < all.Length; ++i)

if (all[i].Name == "MyClass")

break;

if (i == all.Length)

{

Console.WriteLine("MyClass not found in assembly.");

return;

}

Type t = all[i];

// 获取默认构造函数

ConstructorInfo[] ci = t.GetConstructors();

int j;

for (j = 0; j < ci.Length; ++j)

if (ci[j].GetParameters().Length == 0)

break;

if (j == ci.Length)

{

Console.WriteLine("Default constructor not found.");

return;

}

// 创建一个动态的MyClass对象

dynamic obj = ci[j].Invoke(null);

// 通过dynamic对象调用MyClass类的方法

obj.IsEven(15);

}

}

}