在C#中,同一类中,只要参数声明不同,两个或多个重名的方法构成了重载(与C++比较类似).

C#中提供了特定的隐式(自动)类型转换,只有在所有的重载函数中形参均无法与实参匹配时,才会进行隐式类型转换.

例：程序func\_overload\_test1

// Copyright 2016.刘珅珅

// author：刘珅珅

// 函数重载

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace func\_overload\_test1

{

class Overload

{

public void MyMeth(int x)

{

Console.WriteLine("Inside MyMeth(int): " + x);

}

public void MyMeth(double x)

{

Console.WriteLine("Inside MyMeth(double): " + x);

}

}

class OverloadTest

{

static void Main(string[] args)

{

Overload obj = new Overload();

int i = 10;

double d = 10.1;

obj.MyMeth(i); // MyMeth(int)

obj.MyMeth(d); // MyMeth(double)

byte b = 99;

short s = 10;

float f = 11.5F;

obj.MyMeth(b); // MyMeth(int)

obj.MyMeth(s); // MyMeth(int)

obj.MyMeth(f); // MyMeth(double)

}

}

}

ref和out关键字可以用于重载操作。

例：程序func\_overload\_test2

// Copyright 2016.刘珅珅

// author：刘珅珅

// 函数重载

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace func\_overload\_test2

{

class Overload

{

public void MyMeth(int x)

{

Console.WriteLine("Inside MyMeth(int): " + x);

}

// 只增加了ref关键字可以与上面的函数构成重载

public void MyMeth(ref int x)

{

Console.WriteLine("Inside MyMeth(ref int): " + x);

}

//public void MyMeth(out int x) // error，不能与ref int构成重载

//{

// x = 10;

//}

}

class OverloadTest

{

static void Main(string[] args)

{

Overload obj = new Overload();

int i = 10;

obj.MyMeth(i); // call MyMeth(int)

obj.MyMeth(ref i); // call MyMeth(ref int)

}

}

}

但out int与ref int不能构成重载。