在C#中，值类型默认的传递方式为值传递，即实参将副本传递给形参，形参的改变不影响实参。如果希望改变值类型变量的传递行为，可以使用ref和out来实现。

ref关键字

例：程序ref\_out\_test1

// Copyright 2016.刘珅珅

// author：刘珅珅

// ref和out改变值类型的传递

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ref\_out\_test1

{

class RefTest

{

public void Sqr(ref int i)

{

i = i \* i;

}

public void Swap(ref int a, ref int b)

{

a = a + b;

b = a - b;

a = a - b;

}

}

class RefOutTest

{

static void Main(string[] args)

{

RefTest obj = new RefTest();

int a = 10;

Console.WriteLine("a before call: " + a); // 10

obj.Sqr(ref a);

Console.WriteLine("a after call: " + a); // 100

int m = 10;

int n = 6;

Console.WriteLine("m before call: " + m); // 10

Console.WriteLine("n before call: " + n); // 6

obj.Swap(ref m, ref n);

Console.WriteLine("m after call: " + m); // 6

Console.WriteLine("n after call: " + n); // 10

}

}

}

ref参数必须先初始化。

out参数可以从方法接收值，其不需要初始化。在方法终止前必须给out参数赋值。

例：程序ref\_out\_test1

// Copyright 2016.刘珅珅

// author：刘珅珅

// ref和out改变值类型的传递

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ref\_out\_test1

{

class RefTest

{

public void Sqr(ref int i)

{

i = i \* i;

}

public void Swap(ref int a, ref int b)

{

a = a + b;

b = a - b;

a = a - b;

}

}

class OutTest

{

public int GetParts(double src, out double frac)

{

int whole;

whole = (int)src;

frac = src - whole;

return whole;

}

}

class RefOutTest

{

static void Main(string[] args)

{

RefTest obj = new RefTest();

int a = 10;

Console.WriteLine("a before call: " + a); // 10

// a必须初始化

obj.Sqr(ref a);

Console.WriteLine("a after call: " + a); // 100

int m = 10;

int n = 6;

Console.WriteLine("m before call: " + m); // 10

Console.WriteLine("n before call: " + n); // 6

obj.Swap(ref m, ref n);

Console.WriteLine("m after call: " + m); // 6

Console.WriteLine("n after call: " + n); // 10

double val = 3.56;

double frac;

OutTest obj1 = new OutTest();

obj1.GetParts(val, out frac);

Console.WriteLine("Fractional part is " + frac); // 0.56

}

}

}

对于引用类型，也可以使用ref和out关键字。使用ref和out关键字修饰引用时，允许函数改变引用指向的对象，类似用C++中的二重指针。

例：程序ref\_out\_test2

// Copyright 2015.

// author：刘珅珅

// ref和out修饰引用类型参数

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ref\_out\_test2

{

class RefSwap

{

int a, b;

public RefSwap(int i, int j)

{

a = i;

b = j;

}

public void Show()

{

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

}

public void Swap(RefSwap obj1, RefSwap obj2)

{

RefSwap temp;

temp = obj1;

obj1 = obj2;

obj2 = temp;

}

public void SwapRef(ref RefSwap obj1, ref RefSwap obj2)

{

RefSwap temp;

temp = obj1;

obj1 = obj2;

obj2 = temp;

}

}

class RefOutTest

{

static void Main(string[] args)

{

RefSwap x = new RefSwap(1, 2);

RefSwap y = new RefSwap(3, 4);

Console.Write("x before call: "); // 1, 2

x.Show();

Console.Write("y before call: "); // 3, 4

y.Show();

Console.WriteLine();

// not ref

x.Swap(x, y);

Console.Write("not ref x after call: "); // 1, 2

x.Show();

Console.Write("not ref y after call: "); // 3, 4

y.Show();

Console.WriteLine();

// ref

x.SwapRef(ref x, ref y);

Console.Write("ref x after call: "); // 3, 4

x.Show();

Console.Write("ref y after call: "); // 1, 2

y.Show();

}

}

}