Lambda表达式

表达式Lambda

(param-list)=>expr

count => count + 2 // count的值增加2

例：程序lambda\_test1

// Copyright 2016.刘珅珅

// author：刘珅珅

// 表达式Lambda

using *System*;

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

using *System*.*Linq*;

using *System*.*Text*;

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

namespace lambda\_test1

{

delegate int Incr(int v);

delegate bool IsEven(int v);

class LambdaTest

{

static void Main(string[] args)

{

// 表达式Lambda与委托

Incr incr = count => count + 2;

int x = -10;

while (x <= 0)

{

x = incr(x);

}

IsEven is\_even = n => n % 2 == 0;

for (int i = 0; i <= 10; ++i)

{

if (is\_even(i))

{

*Console*.*WriteLine*(i + " is even.");

}

}

}

}

}

首先声明一个委托，然后将Lambda表达式赋给委托。编译器根据委托的类型来推断Lambda表达式参数的类型和表达式的返回类型。因此，Lambda表达式参数类型和返回类型必须和委托类型一致。

如果一个委托返回类型为void，可以使用语句Lambda。

语句Lambda：

例：程序lambda\_test2

// Copyright 2016.刘珅珅

// author：刘珅珅

// 语句Lambda

using *System*;

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

using *System*.*Linq*;

using *System*.*Text*;

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

namespace lambda\_test2

{

class LambdaTest

{

delegate void IntOp(int end);

static void Main(string[] args)

{

// 语句Lambda

IntOp fact = n => {

int r = 1;

for (int i = 1; i <= n; ++i)

{

r = i \* r;

}

*Console*.*WriteLine*("The factorial of {0} is {1}", n, r);

};

fact(5);

}

}

}