写法

参数=>表达式

p=>p-1;

(p,j)=>{ return p-j; }

Lambda表达式其实就相当于匿名函数，函数参数及返回值得类型通过定义的委托来确定

Lambda与委托

delegate int GuangChaoshi(int a,int b);

static void Main(string[] args)

{

//p, z的类型通过GuangChaoshi 委托的int a,int b来确定

GuangChaoshi gwl = (p, z) =>

{

int zuidixiaofei = 10;

if (p < zuidixiaofei)

{

//返回类型通过GuangChaoshi 委托的返回类型确定

return 100;

}

else

{

return z - p - 10;

}

};

Console.WriteLine(gwl(10,100) + ""); //打印80，z对应参数b，p对应参数a

Console.ReadKey();

}

Lambda与泛型委托

static void Main(string[] args)

{

//Func<int, int, bool>是一个以定义好的泛型委托Func<T1, T2, TResult>

Func<int, int, bool> gwl = (p, j) =>

{

if (p + j == 10)

{

return true;

}

return false;

};

Console.WriteLine(gwl(5,5) + ""); //打印‘True’，z对应参数b，p对应参数a

Console.ReadKey();

}

例：

//定义委托Func<int, int> abcd1

int a(Func<int, int> abcd1)

{

//调用委托

return abcd1(1);

}

public void main(){

//Lambda表达式p => p + 1，p与返回值的类型有Func<int, int> abcd1确定

a(p => p + 1);

}