C#中属性的定义：

type name {

get {

}

set {

}

}

属性如果同时定义了get存取器和set存取器，则为可读写属性，如果只定义了get存取器，则为只读属性；如果只定义了set存储区，则为只写属性。

例：程序property\_test1

// Copyright 2016.刘珅珅

// author：刘珅珅

// 属性测试

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace property\_test1

{

class SimpleProperty

{

int prop;

int[] array;

int length;

public SimpleProperty(int size)

{

array = new int[size];

length = size;

}

// 读写属性

public int MyProp

{

get

{

return prop;

}

set

{

if (value >= 0)

{

prop = value;

}

}

}

public int Length

{

get

{

return length;

}

}

}

class PropertyTest

{

static void Main(string[] args)

{

SimpleProperty obj = new SimpleProperty(5);

obj.MyProp = 100;

Console.WriteLine("Value of MyProp: " + obj.MyProp);

Console.WriteLine("Value of Length: " + obj.Length);

// obj.Length = 10; // error，Length是只读属性

}

}

}

属性的限制：

1. 属性不能重载；
2. 属性不定义存储位置，所以不能以ref或out参数的形式传递给方法；
3. 在get存取器中不应该（不强制实施的规则）改变底层变量的状态。

在索引器和属性中，可以使用get和set存取器。默认情况下，set和get存取器的可访问性与它们所属的索引器和属性相同。

可以为set和get存取器设置自身的访问修饰符，在任何情况下，存取器的访问修饰符应该比起属性或索引器的访问规范更严格的限制。

自动实现属性：一般的属性，在底层都有一个备份字段，而自动实现属性没有备份字段，而且必须同时定义get和set存取器。

例：程序property\_test2

// Copyright 2016.刘珅珅

// author：刘珅珅

// 自动实现属性

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace property\_test2

{

class FailSoftArray

{

int[] array;

public FailSoftArray(int size)

{

array = new int[size];

Length = size;

}

public int this[int index]

{

get

{

if (Ok(index))

{

Error = false;

return array[index];

}

else

{

Error = true;

return 0;

}

}

set

{

if (Ok(index))

{

array[index] = value;

Error = false;

}

else

{

Error = true;

}

}

}

// 自动实现属性

public int Length

{

get;

private set;

}

public bool Error

{

get;

private set;

}

private bool Ok(int index)

{

if (index >= 0 && index < Length)

{

return true;

}

return false;

}

}

class AutoPropertyTest

{

static void Main(string[] args)

{

FailSoftArray fs = new FailSoftArray(5);

for (int i = 0; i < fs.Length + 1; ++i)

{

fs[i] = i + 1;

if (fs.Error)

{

Console.WriteLine("Error with index " + i);

}

}

}

}

}