String类是C#的string类型的基础。

字符串的比较：

.NET Framework以两种常用的方式进行字符串比较：

1. 区分区域性的字符串比较，使用字典顺序和语言特征进行字符串比较。
2. 序数值比较，只使用构成字符串的字符的序数值（二进制值）进行比较。

“==”和“!=”运算符是序数值比较。

例：程序string\_test2

// Copyright 2016.刘珅珅

// author：刘珅珅

// 字符串比较

using *System*;

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

using *System*.*Linq*;

using *System*.*Text*;

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

namespace string\_test2

{

class StringTest

{

static void Main(string[] args)

{

string str1 = "alpha";

string str2 = "Alpha";

// 区分区域性比较

int result = *String*.*Compare*(str1, str2, *StringComparison*.*CurrentCulture*);

*Console*.*Write*("Using a culture-sensitive comparison: ");

if (result < 0)

*Console*.*WriteLine*(str1 + " is less than " + str2);

else if (result > 0)

*Console*.*WriteLine*(str1 + " is greater than " + str2);

else

*Console*.*WriteLine*(str1 + " equals " + str2);

// 序数值比较

result = *String*.*Compare*(str1, str2, *StringComparison*.*Ordinal*);

*Console*.*Write*("Using an ordinal comparison: ");

if (result < 0)

*Console*.*WriteLine*(str1 + " is less than " + str2);

else if (result > 0)

*Console*.*WriteLine*(str1 + " is greater than " + str2);

else

*Console*.*WriteLine*(str1 + " equals " + str2);

}

}

}

输出结果为：

Using a culture-sensitive comparison: alpha is less than Alpha

Using an ordinal comparison: alpha is greater than Alpha

字符串搜索：

// Copyright 2016.刘珅珅

// author：刘珅珅

// 字符串的查找

using *System*;

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

using *System*.*Linq*;

using *System*.*Text*;

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

namespace string\_test3

{

class StringTest

{

static void Main(string[] args)

{

string str = "C# has powerful string handling.";

*Console*.*WriteLine*("str: " + str);

int index = str.*IndexOf*('h');

*Console*.*WriteLine*("Index of first 'h': " + index);

index = str.*LastIndexOf*('h');

*Console*.*WriteLine*("Index of last 'h': " + index);

char[] chrs = { 'a', 'b', 'c'};

index = str.*IndexOfAny*(chrs);

*Console*.*WriteLine*("Index of first 'a', 'b', or 'c': " + index);

if (str.*StartsWith*("C# has"))

*Console*.*WriteLine*("str begins with \"C# has\"");

if (str.*Contains*("power"))

*Console*.*WriteLine*("The sequence power was found.");

}

}

}

输出结果为：

str: C# has powerful string handling.

Index of first 'h': 3

Index of last 'h': 23

Index of first 'a', 'b', or 'c': 4

str begins with "C# has"

The sequence power was found.

拆分和合并字符串：

拆分字符串：

public string[] Split(params char[] separator)

其中separator中包含用于分隔各个子字符串的字符。如果separator为空或执行空字符串，使用空格作为分隔符。

例：程序string\_test4

// Copyright 2016.刘珅珅

// author：刘珅珅

// 拆分和合并字符串

using *System*;

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

using *System*.*Linq*;

using *System*.*Text*;

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

namespace string\_test4

{

class StringTest

{

static void Main(string[] args)

{

string str = "One if by land, two if sea.";

char[] seps = { ' ', '.', ','};

// 拆分字符串

// 拆分后的子字符串会出现空字符串""

string[] parts = str.*Split*(seps);

// 拆分后的子字符串去除了空字符串""

string[] parts1 = str.*Split*(seps, *StringSplitOptions*.*RemoveEmptyEntries*);

*Console*.*WriteLine*("Pieces from split: ");

for (int i = 0; i < parts.*Length*; ++i)

*Console*.*WriteLine*(parts[i]);

*Console*.*WriteLine*();

*Console*.*WriteLine*("remove empty");

for (int j = 0; j < parts1.*Length*; ++j)

*Console*.*WriteLine*(parts1[j]);

*Console*.*WriteLine*();

// 合并字符串

string whole = *String*.*Join*("|", parts1);

*Console*.*WriteLine*("Result of join: ");

*Console*.*WriteLine*(whole);

}

}

}

输出结果为：

Pieces from split:

One

if

by

land

two

if

sea

remove empty

One

if

by

land

two

if

sea

Result of join:

One|if|by|land|two|if|sea

填充和裁剪字符串：

例：程序string\_test5

// Copyright 2016.刘珅珅

// author：刘珅珅

// 裁剪和填充字符串

using *System*;

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

using *System*.*Linq*;

using *System*.*Text*;

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

namespace string\_test5

{

class StringTest

{

static void Main(string[] args)

{

string str = "test";

*Console*.*WriteLine*("Original string: " + str);

// 字符串左边填充N个空格，是字符串总长度为10

// 如果字符串本身的长度已经大于或等于10，

// 则不做任何操作

str = str.*PadLeft*(10);

*Console*.*WriteLine*("|" + str + "|");

// 字符串右边填充N个空格，是字符串总长度为20

str = str.*PadRight*(20);

*Console*.*WriteLine*("|" + str + "|");

// 裁剪字符串

// 裁剪字符串左右两边的空格

str = str.*Trim*();

*Console*.*WriteLine*("|" + str + "|");

// 字符串左边填充N个#，是字符串总长度为10

str = str.*PadLeft*(10, '#');

*Console*.*WriteLine*(str);

// 裁剪字符串左右两边的#

str = str.*Trim*('#');

*Console*.*WriteLine*(str);

}

}

}

输出结果：

Original string: test

| test|

| test |

|test|

######test

test

插入、删除和替换

例：程序string\_test6

// Copyright 2016.刘珅珅

// author：刘珅珅

// 字符串插入、删除和替换

using *System*;

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

using *System*.*Linq*;

using *System*.*Text*;

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

namespace string\_test6

{

class StringTest

{

static void Main(string[] args)

{

string str = "This test";

*Console*.*WriteLine*("Original string: " + str);

// 插入

// 从索引位置5开始插入

str = str.*Insert*(5, "is a ");

*Console*.*WriteLine*(str);

// 替换

// 将所有的is替换成was

str = str.*Replace*("is", "was");

*Console*.*WriteLine*(str);

// 删除

// 从索引位置4开始删除5个字符

str = str.*Remove*(4, 5);

*Console*.*WriteLine*(str);

}

}

}