**比较字符串（C# 编程指南）**

比较字符串时，产生的结果会是一个字符串大于或小于另一个字符串，或者两个字符串相等。根据执行的是序号比较还是区分区域性比较，确定结果时所依据的规则会有所不同。对特定的任务使用正确类型的比较十分重要。

当需要对两个字符串的值进行比较和排序而不需要考虑语言惯例时，请使用基本的序号比较。基本的序号比较 (**System.StringComparison.Ordinal**) 是区分大小写的，这意味着两个字符串的字符必须完全匹配：“and”不等于“And”或“AND”。常用的变量有 **System.StringComparison.OrdinalIgnoreCase**，它将匹配“and”、“And”和“AND”；还有 **StringComparison.OrdinalIgnoreCase**，它常用于比较文件名、路径名和网络路径，以及其值不随用户计算机的区域设置的更改而变化的任何其他字符串。有关更多信息，请参见[System.StringComparison](https://msdn.microsoft.com/zh-cn/library/system.stringcomparison.aspx)。

区分区域性比较通常用于对终端用户输入的字符串进行比较和排序，因为这些字符串的字符和排序约定可能会根据用户计算机的区域设置的不同而有所不同。即使是包含相同字符的字符串，也可能会根据当前线程的区域性不同而不同。

| **https://wizardforcel.gitbooks.io/msdn-csharp/content/img/note.jpg 注意** |
| --- |
| 在比较字符串时，您使用的方法应该显式指定了要执行的比较类型。这可增强代码的可维护性和可读性。应尽可能使用具有 [StringComparison](https://msdn.microsoft.com/zh-cn/library/system.stringcomparison.aspx) 枚举参数的 [System.String](https://msdn.microsoft.com/zh-cn/library/system.string.aspx) 和 [System.Array](https://msdn.microsoft.com/zh-cn/library/system.array.aspx) 类的方法重载，以便可以指定要执行的比较类型。比较字符串时，最好避免使用 **==** 和 **!=** 运算符。还应该避免使用 [String.CompareTo](https://msdn.microsoft.com/zh-cn/library/system.string.compareto.aspx) 实例方法，因为这些重载没有 [StringComparison](https://msdn.microsoft.com/zh-cn/library/system.stringcomparison.aspx)。 |

下面的示例演示如何正确地比较其值将不随用户计算机的区域设置的改变而变化的字符串。此外，它还演示 C# 中的字符串限定功能。如果程序声明了两个或更多个相同的字符串变量，编译器会将它们存储在同一位置。通过调用 [ReferenceEquals](https://msdn.microsoft.com/zh-cn/library/system.object.referenceequals.aspx) 方法，可以看到这两个字符串实际引用内存中的同一对象。使用 [String.Copy](https://msdn.microsoft.com/zh-cn/library/system.string.copy.aspx) 方法可避免此限定，如下面的示例所示。

// Internal strings that will never be localized.

string root = @"C:\users";

string root2 = @"C:\Users";

// Use the overload of the Equals method that specifies a StringComparison.

// Ordinal is the fastest way to compare two strings.

bool result = root.Equals(root2, StringComparison.Ordinal);

Console.WriteLine("Ordinal comparison: {0} and {1} are {2}", root, root2,

result ? "equal." : "not equal.");

// To ignore case means "user" equals "User". This is the same as using

// String.ToUpperInvariant on each string and then performing an ordinal comparison.

result = root.Equals(root2, StringComparison.OrdinalIgnoreCase);

Console.WriteLine("Ordinal ignore case: {0} and {1} are {2}", root, root2,

result ? "equal." : "not equal.");

// A static method is also available.

bool areEqual = String.Equals(root, root2, StringComparison.Ordinal);

// String interning. Are these really two distinct objects?

string a = "The computer ate my source code.";

string b = "The computer ate my source code.";

// ReferenceEquals returns true if both objects

// point to the same location in memory.

if (String.ReferenceEquals(a, b))

Console.WriteLine("a and b are interned.");

else

Console.WriteLine("a and b are not interned.");

// Use String.Copy method to avoid interning.

string c = String.Copy(a);

if (String.ReferenceEquals(a, c))

Console.WriteLine("a and c are interned.");

else

Console.WriteLine("a and c are not interned.");

// Output:

// Ordinal comparison: C:\users and C:\Users are not equal.

// Ordinal ignore case: C:\users and C:\Users are equal.

// a and b are interned.

// a and c are not interned.

下面的示例演示如何通过首选方式比较字符串，该首选方式使用具有 [StringComparison](https://msdn.microsoft.com/zh-cn/library/system.stringcomparison.aspx) 枚举的 [System.String](https://msdn.microsoft.com/zh-cn/library/system.string.aspx) 方法。请注意，这里没有使用 [String.CompareTo](https://msdn.microsoft.com/zh-cn/library/system.string.compareto.aspx) 实例方法，因为这些重载没有 [StringComparison](https://msdn.microsoft.com/zh-cn/library/system.stringcomparison.aspx)。

// "They dance in the street."

// Linguistically (in Windows), "ss" is equal to

// the German essetz: 'ß' character in both en-US and de-DE cultures.

string first = "Sie tanzen in die Straße.";

string second = "Sie tanzen in die Strasse.";

Console.WriteLine("First sentence is {0}", first);

Console.WriteLine("Second sentence is {0}", second);

// Store CultureInfo for the current culture. Note that the original culture

// can be set and retrieved on the current thread object.

System.Threading.Thread thread = System.Threading.Thread.CurrentThread;

System.Globalization.CultureInfo originalCulture = thread.CurrentCulture;

// Set the culture to en-US.

thread.CurrentCulture = new System.Globalization.CultureInfo("en-US");

// For culture-sensitive comparisons, use the String.Compare

// overload that takes a StringComparison value.

int i = String.Compare(first, second, StringComparison.CurrentCulture);

Console.WriteLine("Comparing in {0} returns {1}.", originalCulture.Name, i);

// Change the current culture to Deutch-Deutchland.

thread.CurrentCulture = new System.Globalization.CultureInfo("de-DE");

i = String.Compare(first, second, StringComparison.CurrentCulture);

Console.WriteLine("Comparing in {0} returns {1}.", thread.CurrentCulture.Name, i);

// For culture-sensitive string equality, use either StringCompare as above

// or the String.Equals overload that takes a StringComparison value.

thread.CurrentCulture = originalCulture;

bool b = String.Equals(first, second, StringComparison.CurrentCulture);

Console.WriteLine("The two strings {0} equal.", b == true ? "are" : "are not");

/\*

\* Output:

First sentence is Sie tanzen in die Straße.

Second sentence is Sie tanzen in die Strasse.

Comparing in en-US returns 0.

Comparing in de-DE returns 0.

The two strings are equal.

\*/

下面的示例演示如何使用具有 [System.StringComparer](https://msdn.microsoft.com/zh-cn/library/system.stringcomparer.aspx) 参数的静态 [Array](https://msdn.microsoft.com/zh-cn/library/system.array.aspx) 方法以区分区域性的方式对数组中的字符串进行排序和搜索。

class SortStringArrays

{

static void Main()

{

string[] lines = new string[]

{

@"c:\public\textfile.txt",

@"c:\public\textFile.TXT",

@"c:\public\Text.txt",

@"c:\public\testfile2.txt"

};

Console.WriteLine("Non-sorted order:");

foreach (string s in lines)

{

Console.WriteLine(" {0}", s);

}

Console.WriteLine("\n\rSorted order:");

// Specify Ordinal to demonstrate the different behavior.

Array.Sort(lines, StringComparer.Ordinal);

foreach (string s in lines)

{

Console.WriteLine(" {0}", s);

}

string searchString = @"c:\public\TEXTFILE.TXT";

Console.WriteLine("Binary search for {0}", searchString);

int result = Array.BinarySearch(lines, searchString, StringComparer.OrdinalIgnoreCase);

ShowWhere<string>(lines, result);

//Console.WriteLine("{0} {1}", result > 0 ? "Found" : "Did not find", searchString);

// Keep the console window open in debug mode.

System.Console.WriteLine("Press any key to exit.");

System.Console.ReadKey();

}

// Displays where the string was found, or, if not found,

// where it would have been located.

private static void ShowWhere<T>(T[] array, int index)

{

if (index < 0)

{

// If the index is negative, it represents the bitwise

// complement of the next larger element in the array.

index = ~index;

Console.Write("Not found. Sorts between: ");

if (index == 0)

Console.Write("beginning of array and ");

else

Console.Write("{0} and ", array[index - 1]);

if (index == array.Length)

Console.WriteLine("end of array.");

else

Console.WriteLine("{0}.", array[index]);

}

else

{

Console.WriteLine("Found at index {0}.", index);

}

}

}

/\*

\* Output:

Non-sorted order:

c:\public\textfile.txt

c:\public\textFile.TXT

c:\public\Text.txt

c:\public\testfile2.txt

Sorted order:

c:\public\Text.txt

c:\public\testfile2.txt

c:\public\textFile.TXT

c:\public\textfile.txt

Binary search for c:\public\TEXTFILE.TXT

Found at index 2.

\*/

当元素或键的类型为 **string** 时，[System.Collections.Hashtable](https://msdn.microsoft.com/zh-cn/library/system.collections.hashtable.aspx)、[System.Collections.Generic.Dictionary<TKey, TValue>](https://msdn.microsoft.com/zh-cn/library/xfhwa508.aspx) 和 [System.Collections.Generic.List<T>](https://msdn.microsoft.com/zh-cn/library/6sh2ey19.aspx) 等集合类的构造函数具有 [System.StringComparer](https://msdn.microsoft.com/zh-cn/library/system.stringcomparer.aspx) 参数。通常，应尽可能使用这些构造函数，并指定 Ordinal 或 OrdinalIgnoreCase。