

# Re: Solution for sorting an array alpha-numerically

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.general/2005-07/msg00659.html>

---

- *From:* "Lacey" <[Lacey@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:Lacey@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Wed, 13 Jul 2005 12:51:01 -0700
- 

Hi Nick,

You and I are on two different topics. You are talking about lexicographic sorting (like a dictionary) and I am talking about alphanumeric sorting. .NET Framework does not support alphanumeric sorting. (I'll prove it. Keep breathing, Nick! Grab some coffee!)

Let's start with a definition for "alphanumeric sort": a re-ordering of data so that numeric and alphabetic data sort as separate groups. This is unlike a "lexicographic sort" which re-orders data like a dictionary.

I looked through the articles you suggested. In all the examples, the author either:

1. Compared strings that are completely numeric (for example: (int)"11" compared to (int)"2")
2. Compared strings that are completely alphabetic (for example: "John" compared to "Adam")
3. Or, displayed the same problem I solved:  
<http://aspnet.4guysfromrolla.com/demos/ListArticles.aspx> (See how "dgExample2.aspx" is listed after "dgExample19.aspx"?)

None of these articles address the problem of alphanumeric sorting with mixed strings like "22Beers" versus "3Beers".

For example, try this with the CompareTo method you suggested:

```
int intCompare = "22Beers".CompareTo("3Beers");  
PROBLEM: The result is -1, meaning that "22Beers" < "3Beers". If we are doing an alphabetic comparison, this is correct. But if we are doing an alphanumeric comparison, this is incorrect because the number 22 is not less than the number 3.
```

Let's look at a simple example with an array:

1. Create a web page with three labels (lblOriginal, lblLexo, lblAlphaNum)
2. In the Page\_Load:  

```
string[] arrOriginal = {"a3", "a11", "a2", "a1", "a11", "a22"};  
for (int i = 0; i < arrOriginal.Length; i++) lblOriginal.Text += (arrOriginal[i].ToString() + "<BR>");
```

## Re: Solution for sorting an array alpha-numerically

```
string[] arrLexo = {"a3", "a111", "a2", "a1", "a11", "a22"};  
Array.Sort(arrLexo);  
for (int j = 0; j < arrLexo.Length; j++) lblLexo.Text +=  
(arrLexo[j].ToString() + "<BR>");
```

```
string[] arrAlphaNum = {"a3", "a111", "a2", "a1", "a11", "a22"};  
Array.Sort(arrAlphaNum, new AlphaNumCompare());  
for (int k = 0; k < arrAlphaNum.Length; k++) lblAlphaNum.Text +=  
(arrAlphaNum[k].ToString() + "<BR>");
```

3. To the same project as the web page, add the class AlphaNumCompare() which implements IComparer (provided in my previous posting).

Run the page, and you'll see the problem.

My original strings:

a3  
a111  
a2  
a1  
a11  
a22

Microsoft .NET sort:

a1  
a11  
a111  
a2  
a22  
a3

Custom alphanumeric sort:

a1  
a2  
a3  
a11  
a22  
a111

See how a11 is before a2 in the Lexographic sort? This is a common problem – getting string arrays to sort alphanumerically. I found a lot of postings from programmers also looking for solutions. For example, if I have an array of chapter titles for a book, my array needs to sort as:

1.1 Introduction  
2.0 Procurement  
....  
2.9 Department-Specific Accounting  
2.10 Delivery Addresses  
3.0 Forms

You may want to check out this article from Microsoft which says "The .NET

## Re: Solution for sorting an array alpha-numerically

Framework supports word [culture-sensitive comparison of strings], string [similar to a word sort, except that there are no special cases], and ordinal sort [compares the numeric value of each character; for example, a = 65] rules." Note that it does not have any mention of comparing strings with both alphabetic and numeric character and treating these character types as separate groups:

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfstringclassstopic.asp>)

I hope Microsoft soon includes support for alphanumeric sorting of strings.

Lacey

"Nick Malik [Microsoft]" wrote:

- > Fascinating.
- >
- > You make the following statement:
- >> .NET does not support alphanumeric sort of arrays.
- >
- > This is simply not true. For your example, you simply have to implement
- > IComparable for your complex type, you don't need to implement a string
- > comparison. That is built in.
- >
- > In other words, you could replace the line:
- >> return CompareAlphaNum(a, b);
- >
- > with
- > return a.CompareTo(b);
- >
- > and delete the entire "CompareAlphaNum" routine.
- >
- >
- > Note that the CompareAlphaNum routine below doesn't use the same collation
- > order that .Net uses. Perhaps you meant that .Net doesn't sort a capital
- > 'Z' in front of a lowercase 'a'. If that is what you meant, you are right.
- > However, it is patently false to say that .Net doesn't support alphanumeric
- > sorting.
- >
- > Some reference articles for you to read:
- > <http://msdn.microsoft.com/library/en-us/cpref/html/frlrfstringarrayclasssorttopic3.asp>
- > <http://odetocode.com/Articles/203.aspx>
- > <http://www.devx.com/dotnet/Article/21089>
- > <http://aspnet.4guysfromrolla.com/articles/060403-1.aspx>
- >
- > ---
- > ---- Nick Malik [Microsoft]
- > MCSD, CFPS, Certified Scrummaster
- > <http://blogs.msdn.com/nickmalik>
- >
- > Disclaimer: Opinions expressed in this forum are my own, and not
- > representative of my employer.

Re: Solution for sorting an array alpha-numerically

## Re: Solution for sorting an array alpha-numerically

```
> I do not answer questions on behalf of my employer. I'm just a
> programmer helping programmers.
> --
> "Lacey" <Lacey@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message
> news:1190A7FA-8A5A-41F2-82EA-5E45D1C257AA@xxxxxxxxxxxxxxxx
>> .NET does not support alphanumeric sort of arrays. (I hope that changes
>> soon!) Meanwhile, here is a solution.
>>
>> You could use this in your listbox by sorting the items in an array and
>> then
>> adding the array items from 0 to length to your listbox.
>>
>> 1. Create a custom comparer class that implements IComparer:
>> using System;
>> using System.IO;
>>
>> namespace Test1CSharp
>> {
>> /// <summary>
>> /// Lacey Orr
>> /// 29 June 2005
>> /// Alpha-numeric sorting solution.
>> /// </summary>
>>
>> public class AlphaNumCompare : System.Collections.IComparer
>> {
>> public int Compare(Object a1, Object b1)
>> {
>> //In my case, I compared Directory objects. So I took
>> out
>> // the filenames / foldernames from the parameter
>> objects and
>> // passed those to the sort.
>>
>> //The string variables to compare
>> string a = "";
>> string b = "";
>>
>> //Is a1 a FileInfo?
>> if (a1.GetType() == System.Type.GetType("FileInfo"))
>> a = ((FileInfo)a1).Name;
>> else
>> a = a1.ToString();
>>
>> //Is b1 a FileInfo?
>> if (b1.GetType() == System.Type.GetType("FileInfo"))
>> b = ((FileInfo)b1).Name;
>> else
>> b = b1.ToString();
>>
>> return CompareAlphaNum(a, b);
```

Re: Solution for sorting an array alpha-numerically

## Re: Solution for sorting an array alpha-numerically

```
>> }
>>
>> // CompareAlphaNum: Does an alphabetic sort.
>> private static int CompareAlphaNum (string a, string b)
>> {
>> //Do a quick check for empty strings. If one
>> string
>> is empty, then we
>> // can get out without doing any work.
>>
>> if (a.Length == 0 && b.Length > 0)
>> return -1;
>> else if (a.Length > 0 && b.Length == 0)
>> return 1;
>> else if (a.Length == 0 && b.Length == 0)
>> return 0;
>>
>> //The order of chars – make this however you want.
>> string strNums = "0123456789";
>> string strSortOrder = "
>>
>> .!#$%&'()*+,-./:;<=>?@[^_{ }~0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
>>
>> //Variables for comparing
>> bool aSmaller = true;
>> bool isFound = false;
>> int intIndex = 0;
>> // intLength determines the number of times to
>> loop.
>> We will loop
>> // until we hit the end of the shorter string –
>> a
>> or b.
>> int intLength = (a.Length < b.Length? a.Length:
>> b.Length);
>> string strNumA = "";
>> string strNumB = "";
>> int numA = 0;
>> int numB = 0;
>> int j = 0;
>> int k = 0;
>>
>> //Do the compare while we are not at the end of
>> either string and haven't found
>> // the result.
>> while (!isFound && intIndex < intLength)
>> {
>> // if we are dealing with numbers, then sort
>> the numbers numerically
>> if (strNums.IndexOf(a[intIndex]) > -1 &&
>> strNums.IndexOf(b[intIndex]) > -1)
```

## Re: Solution for sorting an array alpha-numerically

```
>> {
>> //Get all the numbers in string A until
>> we hit a non-number
>> j = intIndex;
>> while (j < a.Length &&
>> strNums.IndexOf(a[j]) > -1)
>> {
>> strNumA += a[j].ToString();
>> j++;
>> }
>> //Get all the numbers in string B until
>> we hit a non-number
>> k = intIndex;
>> while (k < b.Length &&
>> strNums.IndexOf(b[k]) > -1)
>> {
>> strNumB += b[k].ToString();
>> k++;
>> }
>>
>> numA = Convert.ToInt32(strNumA);
>> numB = Convert.ToInt32(strNumB);
>>
>> if (numA < numB) // a is before b in
>> sort order; a < b
>> return -1;
>> else if (numA > numB) // b is before a
>> in sort order; a > b
>> return 1;
>> else if (numA == numB)
>> {
>> //The numbers are the same.
>> Remove the number part from the strings
>> // and compare the remainder
>> of
>> the string.
>> return
>> CompareAlphaNum(a.Substring(strNumA.Length, a.Length-strNumA.Length),
>> b.Substring(strNumB.Length, b.Length-strNumB.Length));
>> }
>> }
>> else
>> {
>> if (strSortOrder.IndexOf(b[intIndex]) <
>> strSortOrder.IndexOf(a[intIndex]))
>> {
>> // If string a < b in a sort,
>> then
>> we're done
>> aSmaller = false;
>> isFound = true;
```



## Re: Solution for sorting an array alpha-numerically

```
>>
>> private void Page_Load(object sender, System.EventArgs e)
>> {
>> //Get files in dir
>> String strDir = MapPath("~/./");
>> DirectoryInfo curDir = new DirectoryInfo(strDir);
>> FileInfo [] fiArray = curDir.GetFiles();
>> string [] strFilenames = new string[fiArray.Length];
>> for (int j = 0; j < fiArray.Length; j++)
>> {
>> strFilenames[j] = fiArray[j].Name;
>> }
>>
>> // Sort files
>> Array.Sort(strFilenames, new AlphaNumCompare());
>>
>> //Display files
>> StringBuilder sbFiles = new StringBuilder();
>> for (int k = 0; k < strFilenames.Length; k++)
>> {
>> sbFiles.Append(strFilenames[k] + "<BR>");
>> }
>> lblDir.Text = sbFiles.ToString();
>> }
>>
>> Lacey
>>
>> "Federico G. Babelis" wrote:
>>
>>> Hi All:
>>>
>>> I have this line of code, but the syntax check in VB.NET 2003 and also in
>>> VB.NET 2005 Beta 2 shows as unknown:
>>>
>>> Dim local4 As Byte
>>>
>>> Fixed(local4 = AddressOf dest(offset))
>>>
>>> CType(local4, Short) = CType(src, Short)
>>>
>>> Return
>>>
>>> End Fixed
>>>
>>> What is the "FIXED and END FIXED" and how the syntax error can be "fixed"
>>> ???
>>>
>>> Thx,
>>>
>>> Federico
>>>
```

>>>  
>>>  
>  
>  
>  
.

---

• **Follow-Ups:**

- ◆ **[Re: Solution for sorting an array alpha-numerically](#)**  
    ◇ From: Ross Presser
- ◆ **[Re: Solution for sorting an array alpha-numerically](#)**  
    ◇ From: Nick Malik [Microsoft]

• **References:**

- ◆ **[New functions in .NET 2.0 ???](#)**  
    ◇ From: Federico G. Babelis
- ◆ **[Solution for sorting an array alpha-numerically](#)**  
    ◇ From: Lacey
- ◆ **[Re: Solution for sorting an array alpha-numerically](#)**  
    ◇ From: Nick Malik [Microsoft]

- Prev by Date: **[Re: Public variable QueryString issue](#)**
- Next by Date: **[Re: Stopping error message in VB .NET](#)**
- Previous by thread: **[Re: Solution for sorting an array alpha-numerically](#)**
- Next by thread: **[Re: Solution for sorting an array alpha-numerically](#)**
- Index(es):
  - ◆ **[Date](#)**
  - ◆ **[Thread](#)**