

## LineSorter

Kir\_Antipov | 1,127 installs | 2,802 downloads | ★★★★★ (4) | Free

An extension that allows you to sort rows by different criteria

Download

### Overview

### Q & A

### Rating & Review

## LineSorter

An extension that allows you to sort rows by different criteria

Simply select the lines you wanna sort in the editor, then navigate Context Menu => LineSorter and select the sorting mode you need

If you like LineSorter or my other extensions, you can support my projects via PayPal donation:

Donate

Note: PayPal takes a fee of 1.2% + 0.35€ per donation

If you have any questions, don't hesitate to drop me a [mail!](#)

### Menu

- [Description](#)
- [Preview](#)
  - [Example situation](#)
  - [Built-in sorts](#)
    - [Sort lines alphabetically](#)
    - [Sort lines by length](#)
    - [Shuffle](#)
  - [GIF](#)

### Categories

Tools

Coding

Performance

Programming Languages

### Tags

Sort

Coding

Editor

Lines

### Works with

Visual Studio 2017, 2019

### Resources

[Copy ID](#)

### Project Details

[Kir-Antipov/LineSorter](#)

[No Open Issues](#)

[Last Commit: a month ago](#)

[No Pull Requests](#)

### More Info

Version 4.8

Released on 6/17/2018, 12:12:10 AM

Last updated 7/30/2019, 10:20:00 AM

Publisher Kir\_Antipov

- [Custom sorts](#)
  - [Add custom command by built-in tools](#)
  - [Add custom command by myself](#)
- [LineSorter Options](#)
- [Changelog](#)
- [Ideas](#)

[Report](#)[Report Abuse](#)

---

## Example situation

```
using System.Threading.Tasks;
using System.IO;
using System.Collections.Generic;
using Antlr4.Runtime;
using System.Data;
using Microsoft.CodeAnalysis;
using System.Reflection;
using System;
using System.Text;
using System.Drawing;
using System.Numerics;
using System.Linq;
using System.Xml.Linq;
using System.Collections;
using Antlr4;
using Microsoft.CodeAnalysis.Text;
using Antlr4.Runtime.Misc;
```

Now let's sort it)

---

## Sort lines alphabetically

Sort lines alphabetically result (*shortcut*: (CTRL + E) + (CTRL + A)):

```
using Antlr4.Runtime.Misc;
using Antlr4.Runtime;
using Antlr4;
using Microsoft.CodeAnalysis.Text;
using Microsoft.CodeAnalysis;
using System.Collections.Generic;
using System.Collections;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Numerics;
using System.Reflection;
using System.Text;
```

```
using System.Threading.Tasks;  
using System.Xml.Linq;  
using System;
```

### Sort lines alphabetically (desc) result:

```
using System;  
using System.Xml.Linq;  
using System.Threading.Tasks;  
using System.Text;  
using System.Reflection;  
using System.Numerics;  
using System.Linq;  
using System.IO;  
using System.Drawing;  
using System.Data;  
using System.Collections;  
using System.Collections.Generic;  
using Microsoft.CodeAnalysis;  
using Microsoft.CodeAnalysis.Text;  
using Antlr4;  
using Antlr4.Runtime;  
using Antlr4.Runtime.Misc;
```

---

### Sort lines by length

Sort lines by length result (*shortcut*: (CTRL + E) + (CTRL + L)):

```
using Antlr4;  
using System;  
using System.IO;  
using System.Data;  
using System.Linq;  
using System.Text;  
using Antlr4.Runtime;  
using System.Drawing;  
using System.Numerics;  
using System.Xml.Linq;  
using System.Reflection;  
using System.Collections;  
using Antlr4.Runtime.Misc;  
using Microsoft.CodeAnalysis;  
using System.Threading.Tasks;  
using System.Collections.Generic;  
using Microsoft.CodeAnalysis.Text;
```

Sort lines by length (desc) result:

```
using Microsoft.CodeAnalysis.Text;
using System.Collections.Generic;
using System.Threading.Tasks;
using Microsoft.CodeAnalysis;
using Antlr4.Runtime.Misc;
using System.Collections;
using System.Reflection;
using System.Xml.Linq;
using System.Numerics;
using System.Drawing;
using Antlr4.Runtime;
using System.Text;
using System.Linq;
using System.Data;
using System.IO;
using System;
using Antlr4;
```

---

## Shuffle

Shuffle result (*1 in N*):

```
using System.Numerics;
using System.Xml.Linq;
using System.Collections;
using System.Collections.Generic;
using Microsoft.CodeAnalysis.Text;
using Antlr4.Runtime;
using Microsoft.CodeAnalysis;
using System.IO;
using Antlr4.Runtime.Misc;
using System.Drawing;
using System;
using System.Reflection;
using System.Text;
using System.Linq;
using Antlr4;
using System.Threading.Tasks;
using System.Data;
```

---

## Preview

```
1 using System;
2 using System.Xml.Linq;
3 using System.Threading.Tasks;
4 using System.Text;
5 using System.Reflection;
6 using System.Numerics;
7 using System.Linq;
8 using System.IO;
9 using System.Drawing;
10 using System.Data;
11 using System.Collections;
12 using System.Collections.Generic;
13 using Microsoft.CodeAnalysis;
14 using Microsoft.CodeAnalysis.Text;
15 using Antlr4;
16 using Antlr4.Runtime;
17 using Antlr4.Runtime.Misc;
```

## Add custom command

Now users has the opportunity to create **their own** sorting method!

Ok, let's try to do it)

Imagine that you want to sort these lines:

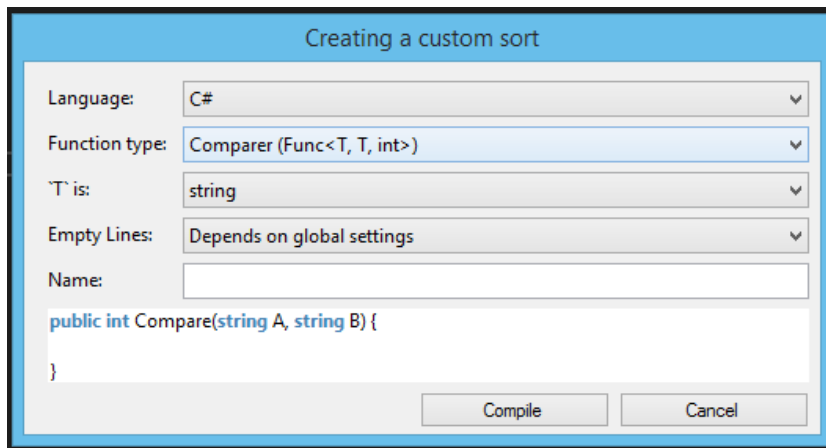
```
-200
3
-7
0
```

```
5
15
String
-1
25
23
Not a number
150
21
Hello, world!
4
53
-123
110
2
```

using next rule: numbers are sorted in ascending order, and all strings follow the numbers and are sorted alphabetically

Navigate Context Menu => LineSorter => Add custom sort

After that the following window will appear:



Creating a custom sort

Language: C#

Function type: Comparer (Func<T, T, int>)

T is: string

Empty Lines: Depends on global settings

Name:

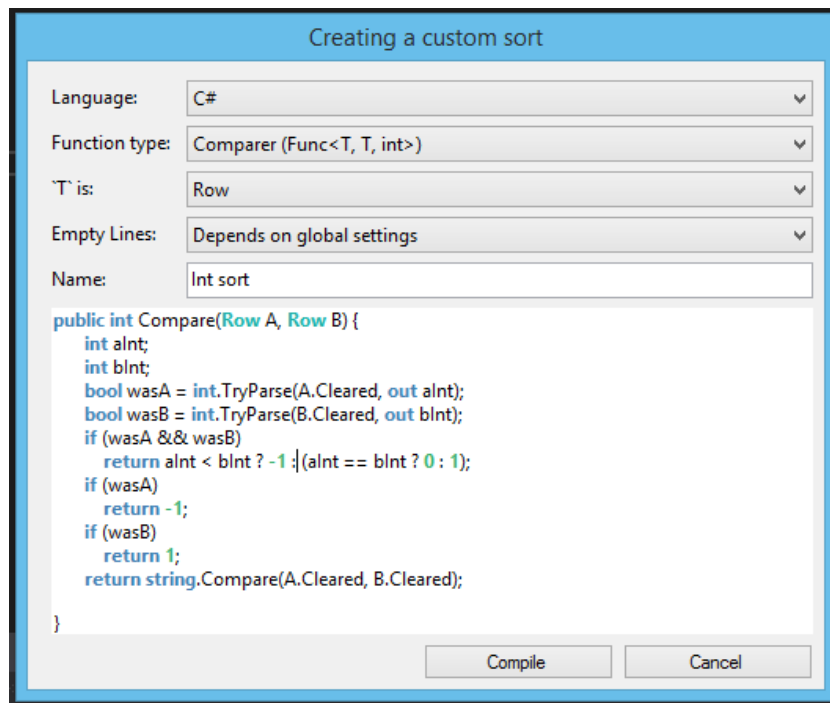
```
public int Compare(string A, string B) {
}
```

Compile Cancel

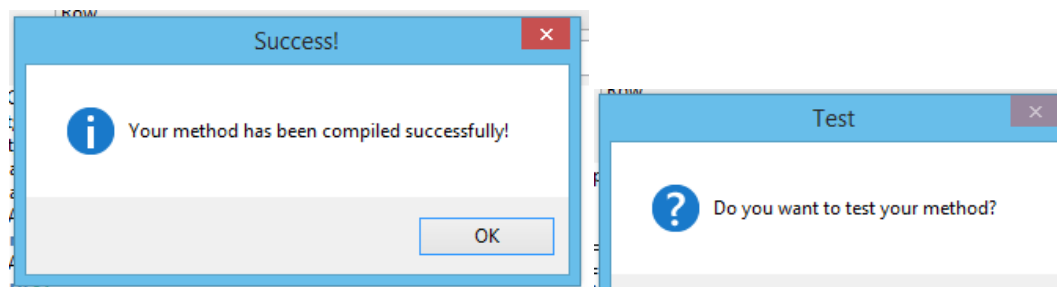
- Using Language combobox you can select language you like (C# or Visual Basic)
- Using Function type combobox you can select implementation you like
- Using 'T' is combobox you can select generic type of collection to be sorted
- Using Empty Lines combobox you can choose the type of processing empty lines

We'll use Comparer function, since we can describe how the 2 elements of the collection relate to each other. Also we'll use Row type ([visit GitHub repo to see its structure](#)), cause lines including the numbers can contain whitespace-symbols we don't need, but they must be returned to the previous place after sorting.

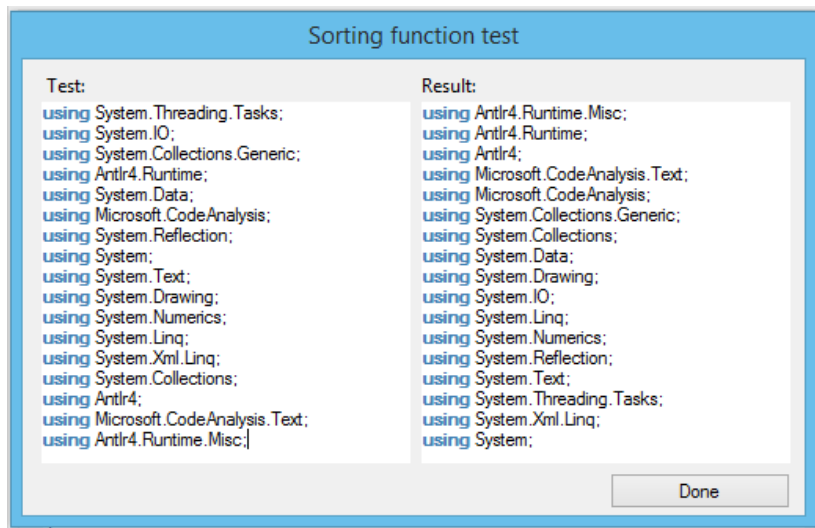
The resulting function will look like this:



Press Compile and you'll see this:

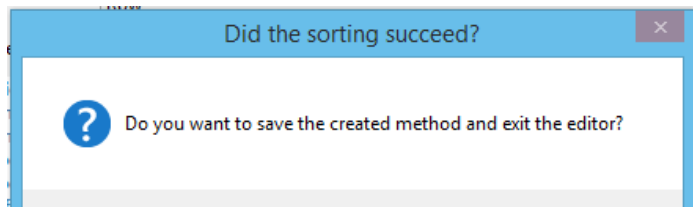


Press Yes to test your method directly in the pop-up window:



(We can change the sample text here, but we'll do it in the VS editor))

Of course you can just skip this step. Anyway you'll see this messagebox:



If you like sorting method you've done and you wanna save it, then press Yes. If you'll press No, you'll be back to code window

Done! Let's try it:



```
1 -200
2 3
3 -7
4 0
5 5
6 15
7 String
8 -1
9 25
10 23
11 Not a number
12 150
13 21
14 Hello, world!
15 4
16 53
17 -123
18 110
19 2
```

We've just created our own sort and applied it in the editor. Successfully!

---

## Add custom command by myself

The method above has some disadvantages:

1. It will be very inconvenient to write a large sorting function
2. You may need some third-party libraries

At the moment LineSorter doesn't have built-in support for adding externally compiled functions, but all this is possible!

Just follow these instructions:

1. Download the current version of [LineSorter.Export.dll](#)
2. Connect it to your project
3. Determine the type of your sorting (it doesn't matter what namespace it will be in)
4. Indicate that it implements the interface [LineSorter.Export.IUserSort](#)
5. Implement the interface:
  1. The Guid property should always return the same unique Guid value (you can generate it using Tools -> Create Guid inside Visual Studio)
  2. The Name property should always return the same human-readable name of your sort
  3. The EmptyLineAction property should return the method of interaction with empty lines
  4. Implement the function Sort as your heart desires)
6. Compile the project
7. Put the resulting dll in the %LocalAppData%\LineSorter\UserSort folder

8. If it has dependent dlls, then put them in the %LocalAppData%\Microsoft\VisualStudio\<your VS version>\Extensions\<LineSorter Path> folder next to LineSorter.dll (<LineSorter Path> is always random, but you can find it if you'll search LineSorter.dll)
9. After launching Visual Studio, you will see your sorting in the list of available

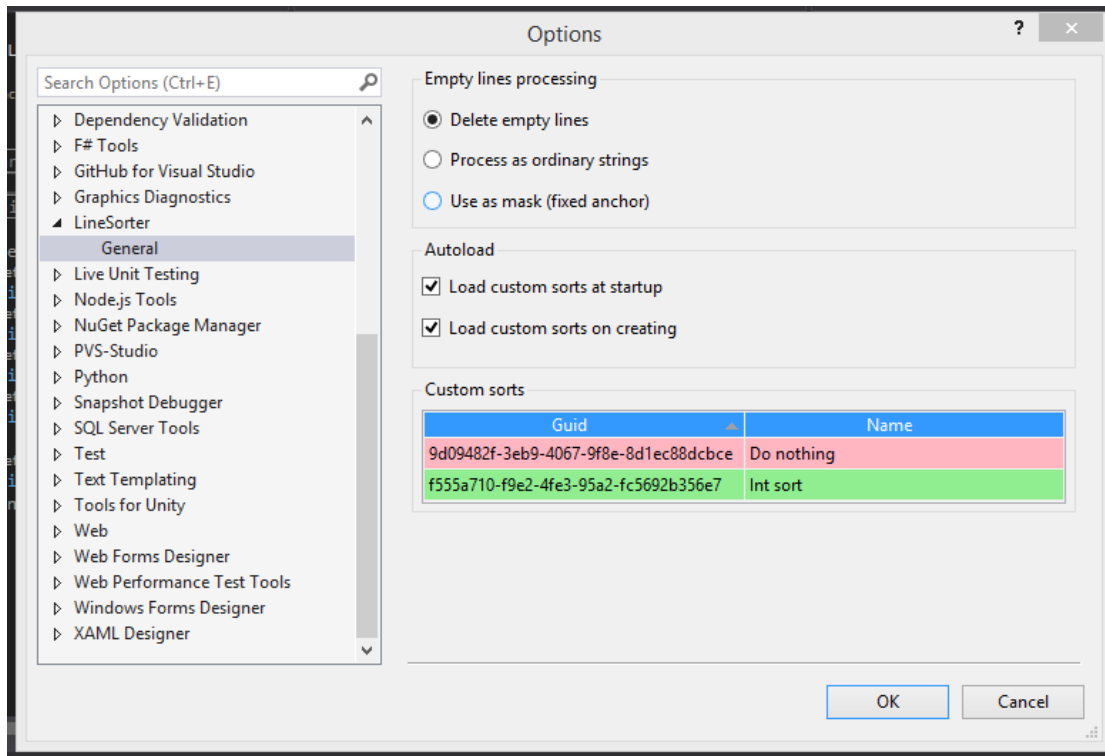
There're some minuses:

- It's not very pleasant to dig into this pile of folders to make such a simple action
- If your sorting has dependencies and you put them next to the extension, as described above, then when you update the extension, all of them will be deleted and you'll have to re-insert them

All this is fixable, so if all of a sudden you need it - open the [issue on GitHub](#) (I just don't want to implement something that nobody needs))

## LineSorter Options

You can manage your sorts in Options => LineSorter:



## Important!

---

Right-click on the line with the sort and you'll be able to turn it **on** or **off** or **delete**)

---

## Changes

---

- 1.0 - Initial release
  - 1.1 - Fixed bug with deletion of a new line from the last line for sorting
  - 2.0 - Localization support and new commands:
    1. Sort lines by length (desc)
    2. Sort lines alphabetically
    3. Sort lines alphabetically (desc)
  - 2.1 - XAML files support
  - 2.5 - New command:
    1. Shuffle
  - 3.0 - Impressive refactoring and  $\infty$  number of new commands, cause
    - Added the ability to create your own sort
  - 3.5 - Few improves:
    - 3 different scenarios for processing empty lines
      1. Remove
      2. Process as ordinary string
      3. Process as mask
    - Fixed critical bug with deletion custom sorts on extension update
  - 4.0 - VS 2019 support ([#1](#)) and some fixes:
    - Different newline representations now available to work with ([#2](#))
    - Custom sorting's windows should now display correctly ([#3](#))
  - 4.0.1 - JSON files support ([#4](#))
  - 4.5 - Some improvements:
    - New cool icon by *Dessader* :3
    - New compilers support (no more limitations when creating custom sorts)
    - Fixed inappropriate create-custom-sort window behavior
    - Ha-ha. From the very first version I forgot about character escaping in regex: `"/.*.*/"`
  - 4.6 - Visual Studio MPF 15.0 reference fixed ([#5](#))
  - 4.7 - Few improves:
    - Improved icon for small sizes
    - Fixed exception due to blocked clipboard ([#6](#))
  - 4.8 - Now all my extensions work on the basis of a single API, which made it possible to simplify some things and repeatedly reduce the weight of extensions!
-

## Ideas

---

1. If we'll refine the idea of adding third-party sortings, then we can even create a kind of store/marketplace of custom sortings from different people based on our GitHub accounts! If you are interested in this idea - create an appropriate [issue](#)

[Contact us](#) [Jobs](#) [Privacy](#) [Terms of use](#) [Trademarks](#)

© 2019 Microsoft