# Input, output, and sorting

In this coding exercise, you will write a program that accepts as input a locale such as `en_US` and a file of lines (such as words or names) in UTF-16 LE format, the native encoding of Microsoft Windows. Your program then will sort the lines alphabetically according to the rules of the locale, and finally write them out in sorted form to another UTF-16 LE file.

## Sample output

```
$ ./iosort1
Enter name of file to sort: cote16.txt
Enter sort locale: es_ES

Line 0, length 8:
ñandú

Line 1, length 6:
coté

Line 2, length 8:
número

Line 3, length 6:
côte

Line 4, length 8:
Namibia

Line 5, length 5:
ñú

Line 6, length 7:
côté

Line 7, length 5:
cote

Opening ICU collation.

Sorting...

Closing ICU collation.
```

```
Writing sorted UTF-16 LE file.
Converting UTF-16 LE file to UTF-8.
Quitting.
```

### Notes

The references in this section contain some good examples of how the same collections of words are sorted in different locales. Also see the Stack Overflow link for a solution to a problem that plagues programmers who use the `icu-config` utility to build their software.

### References

- Sorting and String Comparison - Globalization | Microsoft Docs
- Localization Tips, Part 3: Sorting and Collating – Cliff Meyers
- ICU - International Components for Unicode
- c - Error when linking a program which uses ICU - Stack Overflow

## The Compose Key

Before you begin programming the locale sort exercise, you might want to learn how to enter arbitrary Unicode characters from the keyboard.

### Notes

The references to this section contain some articles about configuring a Compose key to enter Unicode characters on Linux computers. Be careful not to copy the curly quotes used for the `.XCompose` entries from the Samsung article. You can configure Compose keys for Microsoft Windows and Mac OS X as well.

### References

- Compose key - Wikipedia
- Custom Compose Keys on Ubuntu - Samsung Open Source Group Blog
- How can I make IBus not ignore ~/.XCompose? - Ask Ubuntu
- dead keys - How to Get A with Dots in Dvorak of Ubuntu 16.04? - Ask Ubuntu

## Input of filename and sort locale

```
$ ./iosort1
Enter name of file to sort: cote16.txt
Enter sort locale: es_ES
```

### Notes

The sample output in this section shows the user entering a filename and a locale in which to sort it. If you prefer, you can specify these items with command-line options, dialog boxes, and so on instead.

## Reading file into a Unicode string array

```
Line 0, length 8:
ñandú

Line 1, length 6:
coté

Line 2, length 8:
número

Line 3, length 6:
côte

Line 4, length 8:
Namibia

Line 5, length 5:
ñú

Line 6, length 7:
côté
```

### Notes

On Microsoft Windows, reading a UTF-16 LE text file line by line is not difficult. On Linux, it's a little trickier. You might consider converting the file to UTF-8 first, the native Unicode encoding of Linux. After that, you can use ordinary C commands such as `readln()` and `printf()` to read and display the data. Remember, you can convert the encoding of a file with one line of code if you use the `iconv` utility.

If necessary, as you read each string, you can re-convert it to UTF-16, which is the preferred encoding of the ICU utilities.

#### References

- ICU 59.1: ucol.h File Reference
- UTF-8 - ICU User Guide

## Sorting the array by a locale

```
Opening ICU collation.

Sorting...

Closing ICU collation.
```

#### Notes

Before sorting a Unicode text file, you must open a collation that specifies the locale with which to sort the data. You can then use any well-known sort algorithm, such as an insertion sort, to put the data in order. **Warning**: Do not use an ordinary string comparison routine such as `strcmp()` to decide which string comes before another. Your comparison function must use the locale's rules to decide the order of any two strings.

#### References

- Unicode Sorting Test Page
- Collation - ICU User Guide
- Collation Examples - ICU User Guide

## Saving the sorted lines to a file

```
Writing sorted UTF-16 LE file.
Converting UTF-16 LE file to UTF-8.
Quitting.
```

#### Notes

After the lines of the file have been saved in sorted form to a Unicode string array, write them to a UTF-16 LE file. After that, if you are on Linux, you

might want to convert it to UTF-8 (with `iconv` again) so it's easier to see the results by opening them in a text editor.

### References

- ICU 60.1: ustdio.h File Reference