

FileSorting

Generated by Doxygen 1.8.13

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Class Index	3
2.1	Class List	3
3	Namespace Documentation	5
3.1	FullFileReader Namespace Reference	5
3.1.1	Detailed Description	5
3.1.2	Function Documentation	5
3.1.2.1	changeSlashesToNulles()	5
3.1.2.2	outputInFile()	6
3.1.2.3	readFullFile()	6
3.2	StringSorter Namespace Reference	6
3.2.1	Detailed Description	7
3.2.2	Function Documentation	7
3.2.2.1	comparator()	7
3.2.2.2	freeText()	7
3.2.2.3	outputAscendingFromEndText()	8
3.2.2.4	outputAscendingText()	8
3.2.2.5	outputNormalText()	8
3.2.2.6	sortMyFile()	9
4	Class Documentation	11
4.1	StringSorter::StringToCompare Class Reference	11
4.1.1	Detailed Description	11
4.1.2	Constructor & Destructor Documentation	11
4.1.2.1	StringToCompare()	11
4.1.3	Member Function Documentation	12
4.1.3.1	hasNext()	12
4.1.3.2	isEmpty()	12
4.1.3.3	isNextSpecial()	12
4.1.3.4	next()	13
	Index	15

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

FullFileReader	
Change full file into variable and manipulate with it	5
StringSorter	
Namespace that will Sort text in linux encoding	6

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

StringSorter::StringToCompare	11
---	----

Chapter 3

Namespace Documentation

3.1 FullFileReader Namespace Reference

Change full file into variable and manipulate with it.

Functions

- void [readFullFile](#) (const char *FileName, char **text)
- size_t [changeSlashesToNulles](#) (char *text, size_t **indexes)
- void [outputInFile](#) (size_t *indexes, const char *text, const char *FileName, size_t countOfLines, int typeOfWriting)

3.1.1 Detailed Description

Change full file into variable and manipulate with it.

3.1.2 Function Documentation

3.1.2.1 [changeSlashesToNulles\(\)](#)

```
size_t FullFileReader::changeSlashesToNulles (
    char * text,
    size_t ** indexes )
```

change
to \0

Parameters

<i>text</i>	- variable pointing to text, where to be changed
<i>indexes</i>	- array where elemnts will index to the new tokens

Returns

count of elemnts in indexes

3.1.2.2 outputInFile()

```
void FullFileReader::outputInFile (
    size_t * indexes,
    const char * text,
    const char * FileName,
    size_t countOfLines,
    int typeOfWriting )
```

write text to file with order written in indexes array to the FileName file

Parameters

<i>indexes</i>	- array with order
<i>text</i>	- text where we will read
<i>FileName</i>	- file where to safe file
<i>countOfLines</i>	- count of elements in indexes
<i>typeOfWriting</i>	- parametres to open the file

3.1.2.3 readFullFile()

```
void FullFileReader::readFullFile (
    const char * FileName,
    char ** text )
```

read file to text pointer

Parameters

<i>FileName</i>	- name of the file to be read
<i>text</i>	- variable, that will point to text

3.2 StringSorter Namespace Reference

namespace that will Sort text in linux encoding

Classes

- class [StringToCompare](#)

Functions

- bool [comparator](#) ([StringToCompare](#) &&a, [StringToCompare](#) &&b)
- void [outputAscendingText](#) (size_t *indexes, size_t countOfLines, char *text, const char *DistFileName)
- void [outputAscendingFromEndText](#) (size_t *indexes, size_t countOfLines, char *text, const char *DistFileName)
- void [outputNormalText](#) (size_t *indexes, size_t countOfLines, char *text, const char *DistFileName)
- void [freeText](#) (char *text, size_t *indexes, size_t countOfLines)
- void [sortMyFile](#) (const char *SrcFileName, const char *DistFileName)

3.2.1 Detailed Description

namespace that will Sort text in linux encoding

3.2.2 Function Documentation

3.2.2.1 comparator()

```
bool StringSorter::comparator (
    StringToCompare && a,
    StringToCompare && b )
```

compare two objects

Parameters

<i>a</i>	- first StringToCompare object to be compared
<i>b</i>	- StringToCompare object to be compared

Returns

- true, if object a starting from a.firstElement and iterating with + a.increment is bigger than b starting from b.firstElemnt iterating with + b.increment Empty char* -> at the end else, else

3.2.2.2 freeText()

```
void StringSorter::freeText (
    char * text,
    size_t * indexes,
    size_t countOfLines )
```

free space where text is

Parameters

<i>text</i>	- pointer to first elemt of char
<i>indexes</i>	- pointer to array of indexes, where in left side \0 was put

3.2.2.3 outputAscendingFromEndText()

```
void StringSorter::outputAscendingFromEndText (
    size_t * indexes,
    size_t countOfLines,
    char * text,
    const char * DistFileName )
```

Append to file DistFileName text sorted Ascending starting from the end of line

Parameters

<i>indexes</i>	- array where sorted indexes will be
<i>countOfLines</i>	- count of elements in indexes
<i>text</i>	- full text
<i>DistFileName</i>	- file where text must be printed

3.2.2.4 outputAscendingText()

```
void StringSorter::outputAscendingText (
    size_t * indexes,
    size_t countOfLines,
    char * text,
    const char * DistFileName )
```

Truncate file DistFileName or create it and write there text sorted Ascending

Parameters

<i>indexes</i>	- array where sorted indexes will be
<i>countOfLines</i>	- count of elements in indexes
<i>text</i>	- full text
<i>DistFileName</i>	- file where text must be printed

3.2.2.5 outputNormalText()

```
void StringSorter::outputNormalText (
    size_t * indexes,
```

```
size_t countOfLines,  
char * text,  
const char * DistFileName )
```

Append to file DistFileName full text

Parameters

<i>indexes</i>	- array where sorted indexes will be
<i>countOfLines</i>	- count of elements in indexes
<i>text</i>	- full text
<i>DistFileName</i>	- file where text must be printed

3.2.2.6 sortMyFile()

```
void StringSorter::sortMyFile (  
    const char * SrcFileName,  
    const char * DistFileName )
```

Text read to the variable,
changed to \0 and then outputAscendingText outputAscendingFromEndText outputNormalText are called with appropriate parametres

Parameters

<i>SrcFileName</i>	- file where text must be got
<i>DistFileName</i>	- file where sorted text must be put

Chapter 4

Class Documentation

4.1 StringSorter::StringToCompare Class Reference

```
#include <StringSorter.h>
```

Public Member Functions

- [StringToCompare](#) (const char *data_, int increment_, size_t firstElement_, size_t lastElement_)
- bool [isEmpty](#) ()
- bool [hasNext](#) ()
- bool [isNextSpecial](#) ()
- char [next](#) ()
- void [skipSpecial](#) ()

make currentElement point to element that is going after special symbols

4.1.1 Detailed Description

class that will be used in comparator It will help to iterate through char*

4.1.2 Constructor & Destructor Documentation

4.1.2.1 StringToCompare()

```
StringSorter::StringToCompare::StringToCompare (  
    const char * data_,  
    int increment_,  
    size_t firstElement_,  
    size_t lastElement_ )
```

constructor that will set currentElement = firstElement_

Parameters

<i>data_</i>	- char* to be compared
<i>increment_</i>	- how we iterate through char*
<i>firstElement_</i>	- index of start
<i>lastElement_</i>	- index of end

4.1.3 Member Function Documentation**4.1.3.1 hasNext()**

```
bool StringSorter::StringToCompare::hasNext ( )
```

check if we can further as: currentElement + increment

Returns

true if currentElement + increment != lastElement

4.1.3.2 isEmpty()

```
bool StringSorter::StringToCompare::isEmpty ( )
```

check if given char* is empty

Returns

true - if empty

4.1.3.3 isNextSpecial()

```
bool StringSorter::StringToCompare::isNextSpecial ( )
```

check if next element (currentElement + increment), is not in A..Za..z Call only if [hasNext\(\)](#) - true!!!

Returns

true - if next char is not A..Za..z

4.1.3.4 next()

```
char StringSorter::StringToCompare::next ( )
```

call only if currentElement is adequate

Returns

current char

The documentation for this class was generated from the following files:

- SortString/StringSorter.h
- SortString/StringSorter.cpp

Index

- changeSlashesToNulles
 - FullFileReader, [5](#)
- comparator
 - StringSorter, [7](#)
- freeText
 - StringSorter, [7](#)
- FullFileReader, [5](#)
 - changeSlashesToNulles, [5](#)
 - outputInFile, [6](#)
 - readFullFile, [6](#)
- hasNext
 - StringSorter::StringToCompare, [12](#)
- isEmpty
 - StringSorter::StringToCompare, [12](#)
- isNextSpecial
 - StringSorter::StringToCompare, [12](#)
- next
 - StringSorter::StringToCompare, [12](#)
- outputAscendingFromEndText
 - StringSorter, [8](#)
- outputAscendingText
 - StringSorter, [8](#)
- outputInFile
 - FullFileReader, [6](#)
- outputNormalText
 - StringSorter, [8](#)
- readFullFile
 - FullFileReader, [6](#)
- sortMyFile
 - StringSorter, [9](#)
- StringSorter, [6](#)
 - comparator, [7](#)
 - freeText, [7](#)
 - outputAscendingFromEndText, [8](#)
 - outputAscendingText, [8](#)
 - outputNormalText, [8](#)
 - sortMyFile, [9](#)
- StringSorter::StringToCompare, [11](#)
 - hasNext, [12](#)
 - isEmpty, [12](#)
 - isNextSpecial, [12](#)
 - next, [12](#)
 - StringToCompare, [11](#)
- StringToCompare
 - StringSorter::StringToCompare, [11](#)