FileSorting

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Index

1.1 Namespace List

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Namespace Documentation

3.1 FullFileReader Namespace Reference

Change full file into variable and manupulate 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 typeOf↩ Writing)

3.1.1 Detailed Description

Change full file into variable and manupulate 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

text	- variable pointing to text, where to be changed
indexes	- 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

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

3.1.2.3 readFullFile()

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

read file to text pointer

Parameters

FileName	- name of the file to be read
text	- 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 *DistFile↩ Name)
- 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

a	- first StringToCompare object to be compared
b	- 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

text	- pointer to first elemt of char
indexes	- 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

indexes	- array where sorted indexes will be
countOfLines	- count of elements in indexes
text	- full text
DistFileName	- 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

indexes	- array where sorted indexes will be
countOfLines	- count of elements in indexes
text	- full text
DistFileName	- 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

indexes	- array where sorted indexes will be
countOfLines	- count of elements in indexes
text	- full text
DistFileName	- 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

SrcFileName	- file where text must be got
DistFileName	- 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 interate 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

*data_*	- char∗ to be compared
increment↩_	- how we iterate through char∗
*first↩ Element_*	- index of start
*last↩ Element_*	- 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