StringTools

1.0.2

Generated by Doxygen 1.9.1

1 StringTools 1

1.1 Index 1

1.2 Overview 1

1.3 Namespace features 2

1.4 Main function features 2

1.5 Installation 2

1.6 Namespace Usage 2

1.6.1 Length Calculation 2

1.6.2 Concatenation 2

1.6.3 Substring Extraction 3

1.6.4 Insertion 3

1.6.5 Deletion 3

1.6.6 Searching 3

1.6.7 Replacement 3

1.7 Main function Usage 3

1.7.1 Menu Options 4

1.7.2 Example Usage 4

1.7.3 Input Handling 4

1.7.4 String Operations 5

1.8 Full Documentation 5

1.9 License 5

2 Namespace Index 7

2.1 Namespace List 7

3 Class Index 9

3.1 Class List 9

4 File Index 11

4.1 File List 11

5 Namespace Documentation 13

5.1 strTools Namespace Reference 13

5.1.1 Detailed Description 13

5.1.2 Function Documentation 13

5.1.2.1 concatStr() 14

5.1.2.2 delSubStr() 15

5.1.2.3 findSubStr() 16

5.1.2.4 insertStr() 16

5.1.2.5 replaceStr() 17

5.1.2.6 subStr() 18

5.2 strUtil Namespace Reference 18

5.2.1 Detailed Description 19

5.2.2 Function Documentation 19

5.2.2.1 clearScr() 19
5.2.2.2 isCapturedValueInvalid() 19
5.2.2.3 toLower() [1/2] 20
5.2.2.4 toLower() [2/2] 20
5.2.2.5 toUpper() [1/2] 21
5.2.2.6 toUpper() [2/2] 21
5.2.2.7 userInputHandler() 22

6 Class Documentation **23**
6.1 __StrLogger Class Reference 23
6.1.1 Constructor & Destructor Documentation 23
6.1.1.1 __StrLogger() 23
6.1.1.2 ∼__StrLogger() 23
6.1.2 Member Function Documentation 24
6.1.2.1 log() 24
6.1.2.2 loggerStatus() 24
6.1.2.3 setLogFile() 24
6.1.2.4 toggleLogger() 25
6.2 __StrUtilHelper Class Reference 25
6.2.1 Member Function Documentation 25
6.2.1.1 checkInvalidCharPtr() 25
6.2.1.2 checkLogicErrors() 26
6.2.1.3 ignoreCapturedValue() 26
6.2.1.4 makeSmartPtr() 28
6.2.1.5 toSomething() 28

7 File Documentation **31**
7.1 main.cpp File Reference 31
7.2 strtools.hh File Reference 31
7.2.1 Detailed Description 32
7.3 strutil.hh File Reference 32
7.3.1 Detailed Description 33
7.4 strutilhelper.hh File Reference 34
7.4.1 Detailed Description 34

Index **35**

Chapter 1

StringTools

The `strTools` namespace provides a set of tools for manipulating C-style strings. These functions are designed to simplify common string operations, such as concatenation, substring extraction, insertion, deletion, searching, and replacement. The library ensures proper memory management using `unique_ptr<char[]>`.

1.1 Index

- StringTools
  - Index
  - Overview
  - Namespace features
  - Main function features
  - Installation
  - Namespace Usage
    * Length Calculation
    * Concatenation
    * Substring Extraction
    * Insertion
    * Deletion
    * Searching
    * Replacement
  - Main function Usage
    * Menu Options
    * Example Usage
    * Input Handling
    * String Operations
  - Full Documentation
  - License

1.2 Overview

This project provides a string manipulation library to easily handle strings. It includes functions for calculating the length of a string, concatenating strings, searching for a substring, and generating substrings. The `main.cpp` file demonstrates examples of how to use this library through a simple console menu.

1.3 Namespace features

- **Length Calculation:** Calculate the length of a C-string.
- **Concatenation:** Concatenate two C-strings into a new dynamically allocated string.
- **Substring Extraction:** Extract a substring from a C-string.
- **Insertion:** Insert one C-string into another at a specified position.
- **Deletion:** Remove a substring from a C-string.
- **Searching:** Find the first occurrence of a substring within a C-string.
- **Replacement:** Replace the first occurrence of a substring with another substring.

1.4 Main function features

- **Calculate the Length of a String:** Enter a string and get its length.
- **Concatenate Strings:** Enter three strings and concatenate them.
- **Search for a Substring:** Enter a string and a substring to find its position within the string.
- **Generate a Substring:** Enter a string and generate a random substring from it.
- **Exit:** Exit the program.

1.5 Installation

To use the `strTools` & `strUtil` library, include the `strtools.hh` header in your C++ project:
```
#include "src/.hxx"
```

or:
```
#include "src/strlogger.hh"
#include "src/strtools.hh"
#include "src/strutil.hh"
#include "src/strutilhelper.hh"
```

Ensure that your project is set up to find the header file in its include path.

You can also try running the test program using:
```
g++ -std=c++17 -I src main.cpp -o main.exe && ./main.exe
```

1.6 Namespace Usage

1.6.1 Length Calculation

Calculate the length of a C-string.
```
const char* myString = "Hello, World!";
uint64_t length = strTools::len(myString);  // length will be 13
```

1.6.2 Concatenation

Concatenate two C-strings into a new unique_ptr<char[ ]>.

```
const char* str1 = "Hello, ";
const char* str2 = "World!";
auto result = strTools::concatStr(str1, str2);
// result will contain "Hello, World!"
```

1.6.3 Substring Extraction

Extract a substring from a C-string.

```
const char* myString = "Hello, World!";
auto sub = strTools::subStr(myString, 7, 5);
// sub will contain "World"
```

1.6.4 Insertion

Insert one C-string into another at a specified position.

```
const char* str1 = "Hello, World!";
const char* str2 = "Beautiful ";
auto result = strTools::insertStr(str1, str2, 8);
// result will contain "Hello, Beautiful World!"
```

1.6.5 Deletion

Remove a substring from a C-string.

```
const char* myString = "Hello, World!";
auto result = strTools::delSubStr(myString, 7, 6);
// result will contain "Hello, !"
```

1.6.6 Searching

Find the first occurrence of a substring within a C-string.

```
const char* myString = "Hello, World!";
int64_t index = strTools::findSubStr(myString, "World");
// index will be 7
```

1.6.7 Replacement

Replace the first occurrence of a substring with another substring.

```
const char* myString = "Hello, World!";
const char* sub1 = "World";
const char* sub2 = "Universe";
auto result = strTools::replaceStr(myString, sub1, sub2);
// result will contain "Hello, Universe!"
```

1.7 Main function Usage

NOTE: The `main` function requires C++20. If you are using C++17, this section will not compile.

1.7.1 Menu Options

1. **Calculate the Length of a String:**
    - Prompts the user to enter a string.
    - Calculates the length using `strTools::len`.
2. **Concatenate Three Strings:**
    - Prompts the user to enter three strings.
    - Concatenates them using `strTools::concatStr`.
    - Displays the concatenated result.
3. **Search for a Substring:**
    - Prompts the user to enter a string and a substring.
    - Searches for the substring using `strTools::findSubStr`.
    - Extracts the substring using `strTools::subStr`.
    - Displays the result or an error message if the substring is not found.
4. **Generate a Substring from a String:**
    - Prompts the user to enter a string.
    - Generates random start and end indices.
    - Extracts a substring using `strTools::subStr`.
    - Displays the extracted substring.
5. **Exit:**
    - Exits the program.

1.7.2 Example Usage

1. Run the program.
2. Select an option by entering a number (0-4).
3. Follow the prompts to perform the desired operation.
4. View the result or error message.
5. Repeat until you choose to exit (option 0).

1.7.3 Input Handling

The program uses the `helpers` namespace to manage invalid inputs, out-of-bounds values, and user input for different operations:

- **Invalid Input:** If the input is invalid (non-numeric), an error message is shown.
- **Out-of-Bounds Input:** If the input is not within the range [0, 4], an error message is shown.
- **User Input Handling:** Manages input from the user, including handling exit commands and input overflow.

1.7.4 String Operations

The `strtools` namespace provides the following functions for string manipulation:

- ∗∗Length Calculation (`strTools::len`):∗∗ Calculates the length of a string.
- ∗∗Concatenation (`strTools::concatStr`):∗∗ Concatenates multiple strings.
- ∗∗Substring Search (`strTools::findSubStr`):∗∗ Finds the position of a substring within a string.
- ∗∗Substring Extraction (`strTools::subStr`):∗∗ Extracts a substring from a string based on start and end indices.

1.8 Full Documentation

For more detailed documentation on the code, including function descriptions and usage, refer to the Doxygen documentation available `here`.

1.9 License

The `strTools` library is licensed under the GNU General Public License v3.0. For more details, see `license`.

Chapter 2

Namespace Index

2.1 Namespace List

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

strTools
String manipulation tools 13
strUtil
Utility functions for input handling and console management 18

Chapter 3

Class Index

3.1 Class List

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

__StrLogger 23
__StrUtilHelper 25

Chapter 4

File Index

4.1 File List

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

.hxx . . . . **??**
foo.h . . . . **??**
main.cpp
Examples on how to use the strtools.hh header 31
strlogger.hh . . . . **??**
strtools.hh
String manipulation tools 31
strutil.hh
Utilities for input handling and console management 32
strutilhelper.hh
Utilities for the strutil namespace 34

Chapter 5

Namespace Documentation

5.1 strTools Namespace Reference

String manipulation tools.

Functions

- uniqueStr concatStr (const char ∗s1, const char ∗s2) noexcept

    Concatenates two C-strings into a new unique_ptr<char[]>.
- uniqueStr subStr (const char ∗s, const uint64_t i, uint64_t j)

    Extracts a substring from a string.
- uniqueStr insertStr (const char ∗s1, const char ∗s2, const uint64_t i)

    Inserts one string into another at the specified position.
- uniqueStr delSubStr (const char ∗s, const uint64_t i, const uint64_t j)

    Removes a substring from a string.
- int64_t findSubStr (const char ∗s, const char ∗find)

    Finds the first occurrence of a substring within a string.
- uniqueStr replaceStr (const char ∗s, const char ∗sub1, const char ∗sub2)

    Replaces the first occurrence of a substring with another substring.

5.1.1 Detailed Description

String manipulation tools.

This namespace provides a set of functions for various string operations, including length calculation, concatenation, substring extraction, insertion, deletion, finding substrings, and replacement of substrings. These functions use C-style strings and return results in `uniqueStr` to ensure proper memory management.

5.1.2 Function Documentation

5.1.2.1 concatStr()

```
uniqueStr strTools::concatStr (
            const char * s1,
            const char * s2 )  [noexcept]
```

Concatenates two C-strings into a new unique_ptr<char[ ]>.

This function takes two C-strings and concatenates them into a new dynamically allocated string managed by `uniqueStr`.

Parameters

s1	The first source C-string.
s2	The second source C-string.

Returns

A unique_ptr<char[ ]> containing the concatenated string.

Note

Example usage:
```
const char* str1 = "Hello, ";
const char* str2 = "World!";
auto result = strTools::concatStr(str1, str2);
// result will contain "Hello, World!"
```

5.1.2.2 delSubStr()

```
uniqueStr strTools::delSubStr (
            const char * s,
            const uint64_t i,
            const uint64_t j )
```

Removes a substring from a string.

This function removes a substring from the source string `s` starting at position `i` and having length `j`. The resulting string is returned as a `uniqueStr`.

Parameters

s	The source C-string.
i	The starting position of the substring to be removed.
j	The length of the substring to be removed.

Returns

A unique_ptr<char[ ]> containing the resulting string.

Exceptions

std::out_of_range	if indices are out of bounds.

Note

Example usage:
```
const char* myString = "Hello, World!";
auto result = strTools::delSubStr(myString, 7, 6);
// result will contain "Hello, !"
```

5.1.2.3 findSubStr()

```
int64_t strTools::findSubStr (
            const char * s,
            const char * find )
```

Finds the first occurrence of a substring within a string.

This function searches for the first occurrence of the substring `find` within the source string `s`. It returns the index of the first occurrence, or `INT64_MAX` if the substring is not found.

Parameters

s	The source C-string.
find	The substring to find.

Returns

The index of the first occurrence of the substring, or INT64_MAX if not found.

Note

Example usage:

```
const char* myString = "Hello, World!";
int64_t index = strTools::findSubStr(myString, "World");
// index will be 7
```

5.1.2.4 insertStr()

```
uniqueStr strTools::insertStr (
            const char * s1,
            const char * s2,
            const uint64_t i )
```

Inserts one string into another at the specified position.

This function inserts the source string `s2` into the destination string `s1` at the specified position `i`. The resulting string is returned as a `uniqueStr`.

Parameters

s1	The destination C-string.
s2	The source C-string to be inserted.
i	The position at which to insert s2 into s1.

Returns

A unique_ptr<char[ ]> containing the resulting string.

Exceptions

std::out_of_range	if the position is out of bounds.

Note

Example usage:
```
const char* str1 = "Hello, World!";
const char* str2 = "Beautiful ";
auto result = strTools::insertStr(str1, str2, 8);
// result will contain "Hello, Beautiful World!"
```

5.1.2.5 replaceStr()

```
uniqueStr strTools::replaceStr (
            const char * s,
            const char * sub1,
            const char * sub2 )
```

Replaces the first occurrence of a substring with another substring.

This function replaces the first occurrence of the substring `sub1` in the source string `s` with the substring `sub2`. The resulting string is returned as a `uniqueStr`.

Parameters

s	The source C-string.
sub1	The substring to be replaced.
sub2	The substring to replace with.

Returns

A unique_ptr<char[ ]> containing the resulting string.

Note

Example usage:
```
const char* myString = "Hello, World!";
const char* sub1 = "World";
const char* sub2 = "Universe";
auto result = strTools::replaceStr(myString, sub1, sub2);
// result will contain "Hello, Universe!"
```

5.1.2.6 subStr()

```
uniqueStr strTools::subStr (
             const char * s,
             const uint64_t i,
             uint64_t j )
```

Extracts a substring from a string.

This function extracts a substring from a given C-string starting at position `i` and having `j` characters. The extracted substring is returned as a `uniqueStr`.

Parameters

s	The source C-string.
i	Position of the first character to include (index 0 = first character).
j	Number of characters to extract from i.

Returns

A unique_ptr<char[ ]> containing the extracted substring.

Exceptions

std::out_of_range	if indices are out of bounds.

Note

Example usage:
```
const char* myString = "Hello, World!";
auto sub = strTools::subStr(myString, 7, 5);
// sub will contain "World"
```

5.2 strUtil Namespace Reference

Utility functions for input handling and console management.

Functions

- void clearScr () noexcept

  "Clears" the console screen.
- void toLower (char *src)

  Converts a string to lowercase (in-place).
- void toUpper (char *src)

  Converts a string to uppercase (in-place).
- uniqueStr toLower (const char *src)

  Converts a string to lowercase.
- uniqueStr toUpper (const char *src)

  Converts a string to uppercase.
- bool isCapturedValueInvalid (char value='\n', bool force=false)

  Checks if the captured value from standard input is invalid.
- bool userInputHandler (char *input, const uint64_t size)

  Handles user input, checks for exit command, and handles overflow.

5.2.1 Detailed Description

Utility functions for input handling and console management.

(former name: helpers)

This namespace provides a set of utility functions for handling standard input errors, checking bounds of values, clearing the console screen, and managing user input with overflow and exit command handling.

5.2.2 Function Documentation

5.2.2.1 clearScr()

```
void strUtil::clearScr ( )  [noexcept]
```

"Clears" the console screen.

This function sends escape sequences to the console to clear the screen and move the cursor to the top-left corner. This functionality is platform-specific and might not work on all terminals.

Note

Example usage:
```
strUtil::clearScr();
cout « "Screen ’cleared’.\n";
```

5.2.2.2 isCapturedValueInvalid()

```
bool strUtil::isCapturedValueInvalid (
             char value = '\n',
             bool force = false )
```

Checks if the captured value from standard input is invalid.

This function checks if the last input operation on `cin` failed (e.g., due to non-numeric input). If it fails, it performs the following:

- Clears the error flags from `cin` to allow further input.
- Ignores the remaining invalid input in the stream up to the next newline.

Parameters

value	An optional character to ignore. Default is an escape character (`\n`).
force	Force ignoring input (discards valid input).

Returns

`true` if the captured value was invalid, `false` otherwise.

Note

Example usage:

```
int value;
std::cin » value;
if (strUtil::isCapturedValueInvalid()) {
    std::cout « "Invalid input. Please enter a numeric value.\n";
}
```

5.2.2.3 toLower() [1/2]

```
void strUtil::toLower (
            char * src )
```

Converts a string to lowercase (in-place).

This function modifies the input string by converting all uppercase characters to lowercase. It iterates over each character and applies the `tolower` function.

Parameters

str	The input string to be modified.

Note

Modifies the original string.

Example usage:

```
char myString[] = "Hello, World!";
toLower(myString); // myString will be "hello, world!"
```

5.2.2.4 toLower() [2/2]

```
uniqueStr strUtil::toLower (
            const char * src )
```

Converts a string to lowercase.

This function creates a new string that is a lowercase version of the input string. It uses the `tolower` function to convert each character to lowercase.

Parameters

str	The input string to be converted.

Returns

A new string with all characters converted to lowercase.

Note

Example usage:

```
const char* myString = "Hello, World!";
auto lowerString = toLower(myString); // lowerString will be "hello, world!"
```

5.2.2.5 toUpper() [1/2]

```
void strUtil::toUpper (
             char * src )
```

Converts a string to uppercase (in-place).

This function modifies the input string by converting all lowercase characters to uppercase. It iterates over each character and applies the `toupper` function.

Parameters

str	The input string to be modified.

Note

Modifies the original string.

Example usage:

```
char myString[] = "Hello, World!";
toUpper(myString); // myString will be "HELLO, WORLD!"
```

5.2.2.6 toUpper() [2/2]

```
uniqueStr strUtil::toUpper (
             const char * src )
```

Converts a string to uppercase.

This function creates a new string that is an uppercase version of the input string. It uses the `toupper` function to convert each character to uppercase.

Parameters

str	The input string to be converted.

Returns

A new string with all characters converted to uppercase.

Note

Example usage:

```
const char* myString = "Hello, World!";
auto upperString = toUpper(myString); // upperString will be "HELLO, WORLD!"
```

5.2.2.7 userInputHandler()

```
bool strUtil::userInputHandler (
            char * input,
            const uint64_t size )
```

Handles user input, checks for exit command, and handles overflow.

This function reads user input from the standard input stream until a newline character is encountered or the buffer is full. It checks if the user input is the exit command "/exit". If the input exceeds the buffer size, it truncates the input and clears the remaining characters in the input stream.

Parameters

input	Pointer to the character array where the input will be stored.
size	The size of the input buffer.

Returns

`true` if the input is the exit command "/exit".

`false` otherwise.

Note

Example usage:

```
const int32_t bufferSize = 100;
char input[bufferSize];
bool exit = strUtil::userInputHandler(input, bufferSize);
if (exit) {
    cout « "Exit command received.\n";
} else {
    cout « "You entered: " « input « "\n";
}
```

Chapter 6

Class Documentation

6.1 __StrLogger Class Reference

Public Member Functions

- __StrLogger ()

  Constructs the logger.
- ∼__StrLogger ()

  Destructs the logger.
- void toggleLogger () noexcept

  Toggles the logger state.
- bool loggerStatus () const noexcept

  Gets the logger status.
- void setLogFile (const string &filename) noexcept

  Sets the log file.
- void log (__StrToolsLogLvl level, const string &message)

  Logs a message.

6.1.1 Constructor & Destructor Documentation

6.1.1.1 __StrLogger()

```
__StrLogger::__StrLogger ( )  [inline]
```

Constructs the logger.

This constructor initializes the logger with file closed and logger disabled.

6.1.1.2 ∼__StrLogger()

```
__StrLogger::∼__StrLogger ( )  [inline]
```

Destructs the logger.

This destructor ensures that if the log file is open, it is properly flushed and closed.

6.1.2 Member Function Documentation

6.1.2.1 log()

```
void __StrLogger::log (
            __StrToolsLogLvl level,
            const string & message )  [inline]
```

Logs a message.

This function logs a message with the given log level. It formats the message with a timestamp and the log level, then writes it to the log file and the terminal.

Parameters

level	The log level of the message.
message	The message to log.

6.1.2.2 loggerStatus()

```
bool __StrLogger::loggerStatus ( ) const  [inline], [noexcept]
```

Gets the logger status.

This function returns the current status of the logger.

Returns

True if the logger is enabled, false otherwise.

6.1.2.3 setLogFile()

```
void __StrLogger::setLogFile (
            const string & filename )  [inline], [noexcept]
```

Sets the log file.

This function sets the log file to the provided filename. If a log file is already open, it closes it before opening the new file.

Parameters

filename	The name of the file to log to.

6.1.2.4 toggleLogger()

```
void __StrLogger::toggleLogger ( )  [inline], [noexcept]
```

Toggles the logger state.

This function enables or disables the logger.

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

- strlogger.hh

6.2 __StrUtilHelper Class Reference

Public Member Functions

- void ignoreCapturedValue (char s, bool doClear=true) noexcept

  Ignores invalid input from standard input.
- void checkLogicErrors (bool rule, const string &msg)

  Checks for invalid inputs and throws an exception if the rule is violated.
- void toSomething (char ∗s, int(∗f)(int))

  Converts a string to something (in-place).
- bool checkInvalidCharPtr (const char ∗s, const string &from) noexcept

  Checks for a null character pointer and throws an exception if it is invalid.
- template<class T >
  T makeSmartPtr (const char ∗src) noexcept

  Creates a smart pointer from a C-string.

6.2.1 Member Function Documentation

6.2.1.1 checkInvalidCharPtr()

```
bool __StrUtilHelper::checkInvalidCharPtr (
             const char * s,
             const string & from )  [inline], [noexcept]
```

Checks for a null character pointer and throws an exception if it is invalid.

This function checks if the provided character pointer is `nullptr`. If it is, an `std::invalid_argument` exception is thrown with a specified message. It is useful for validating character pointers before performing operations on them.

Parameters

c	The character pointer to be checked.

Exceptions

std::invalid_argument	if the character pointer is `nullptr`.

Note

Example usage:
```
char* myString = nullptr;
checkInvalidCharPtr(myString); // Throws an exception with the message.
```

6.2.1.2 checkLogicErrors()

```
void __StrUtilHelper::checkLogicErrors (
             bool rule,
             const string & msg )  [inline]
```

Checks for invalid inputs and throws an exception if the rule is violated.

This function evaluates a given condition (rule) and throws a `std::out_of_range` exception with a specified message if the condition is true. It is commonly used to enforce constraints and validate inputs within other functions.

Parameters

rule	The condition to be checked. If this condition evaluates to true, an exception is thrown.
msg	The message to be included in the exception if the rule is violated.

Exceptions

std::out_of_range	if the rule is true.

Note

Example usage:
```
checkLogicErrors(index < arraySize, "Index out of range");
```

6.2.1.3 ignoreCapturedValue()

```
void __StrUtilHelper::ignoreCapturedValue (
             char s,
             bool doClear = true )  [inline], [noexcept]
```

Ignores invalid input from standard input.

This function clears the error flags from `cin` and ignores remaining invalid input up to the next newline. It is typically used after a failed input operation.

Parameters

s	Character to ignore.
doClear	Clears the error state if `true`.

Note

Example usage:
```
int value;
std::cin » value;
ignoreCapturedValue('\n');
```

6.2.1.4 makeSmartPtr()

```
template<class T >
T __StrUtilHelper::makeSmartPtr (
             const char * src )  [inline], [noexcept]
```

Creates a smart pointer from a C-string.

This template function takes a C-string and creates a smart pointer of the specified type, managing the memory allocation and copying of the string.

Template Parameters

T	The type of the smart pointer to create.

Parameters

src	The source C-string to copy.

Returns

A smart pointer of type T containing the copied string.

Note

Example usage:
```
const char* source = "Example";
auto result = makeSmartPtr<std::unique_ptr<char[]»(source);
// result will contain "Example"
```

6.2.1.5 toSomething()

```
void __StrUtilHelper::toSomething (
             char * s,
             int(*)(int) f )  [inline]
```

Converts a string to something (in-place).

This function modifies the input string by converting all characters into something.

Parameters

str	The input string to be modified.
f	Function to do something with the string (e.g., `tolower` or `toupper`).

Note

Modifies the original string.

Example usage:

```
std::string myString = "Hello, World!";
toSomething(myString); // `myString` will be something, I don't know
```

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

- strutilhelper.hh

Chapter 7

File Documentation

7.1 main.cpp File Reference

Examples on how to use the strtools.hh header.

```
#include "src/.hxx"
#include <array>
#include <cstdint>
#include <iostream>
#include <istream>
#include <memory>
#include <new>
#include <random>
#include <string>
#include <string.h>
```

Include dependency graph for main.cpp:

7.2 strtools.hh File Reference

String manipulation tools.

```
#include "strlogger.hh"
#include "strutil.hh"
#include "strutilhelper.hh"
#include <cstdint>
#include <cstring>
#include <memory>
#include <string>
#include <string.h>
```

Include dependency graph for strtools.hh: This graph shows which files directly or indirectly include this file:

Namespaces

- strTools

  String manipulation tools.

Functions

- uniqueStr strTools::concatStr (const char ∗s1, const char ∗s2) noexcept

    Concatenates two C-strings into a new unique_ptr<char[ ]>.
- uniqueStr strTools::subStr (const char ∗s, const uint64_t i, uint64_t j)

    Extracts a substring from a string.
- uniqueStr strTools::insertStr (const char ∗s1, const char ∗s2, const uint64_t i)

    Inserts one string into another at the specified position.
- uniqueStr strTools::delSubStr (const char ∗s, const uint64_t i, const uint64_t j)

    Removes a substring from a string.
- int64_t strTools::findSubStr (const char ∗s, const char ∗find)

    Finds the first occurrence of a substring within a string.
- uniqueStr strTools::replaceStr (const char ∗s, const char ∗sub1, const char ∗sub2)

    Replaces the first occurrence of a substring with another substring.

7.2.1 Detailed Description

String manipulation tools.

Author

Ian Hylton

Version

1.0.2

Date

2024-08-02

Copyright

Copyright (c) zperk 2024

7.3 strutil.hh File Reference

Utilities for input handling and console management.

```
#include "strlogger.hh"
#include "strutilhelper.hh"
#include <cctype>
#include <cstdint>
#include <iostream>
#include <memory>
#include <string>
#include <string.h>
```

Include dependency graph for strutil.hh: This graph shows which files directly or indirectly include this file:

Namespaces

- strUtil

  Utility functions for input handling and console management.

Macros

- #define **uniqueStr** std::unique_ptr<char[ ]>
- #define **sharedStr** std::shared_ptr<char[ ]>

Functions

- void strUtil::clearScr () noexcept

  "Clears" the console screen.
- void strUtil::toLower (char ∗src)

  Converts a string to lowercase (in-place).
- void strUtil::toUpper (char ∗src)

  Converts a string to uppercase (in-place).
- uniqueStr strUtil::toLower (const char ∗src)

  Converts a string to lowercase.
- uniqueStr strUtil::toUpper (const char ∗src)

  Converts a string to uppercase.
- bool strUtil::isCapturedValueInvalid (char value='\n', bool force=false)

  Checks if the captured value from standard input is invalid.
- bool strUtil::userInputHandler (char ∗input, const uint64_t size)

  Handles user input, checks for exit command, and handles overflow.

7.3.1 Detailed Description

Utilities for input handling and console management.

(former helpers.hh)

Author

Ian Hylton

Version

1.0.4

Date

2024-08-02

Copyright

Copyright (c) zperk 2024

7.4 strutilhelper.hh File Reference

Utilities for the strutil namespace.

```
#include "strlogger.hh"
#include <cstring>
#include <iosfwd>
#include <iostream>
#include <limits>
#include <stdexcept>
#include <string>
```
Include dependency graph for strutilhelper.hh: This graph shows which files directly or indirectly include this file:

Classes

- class __StrUtilHelper

Variables

- class __StrUtilHelper **__StrUtilExtra**

7.4.1 Detailed Description

Utilities for the strutil namespace.

Author

Ian Hylton

Version

1.0.0

Date

2024-07-31

Copyright

Copyright (c) zperk 2024

Index

__StrLogger, 23
    __StrLogger, 23
    ∼__StrLogger, 23
    log, 24
    loggerStatus, 24
    setLogFile, 24
    toggleLogger, 25
__StrUtilHelper, 25
    checkInvalidCharPtr, 25
    checkLogicErrors, 26
    ignoreCapturedValue, 26
    makeSmartPtr, 28
    toSomething, 28
∼__StrLogger
    __StrLogger, 23

checkInvalidCharPtr
    __StrUtilHelper, 25
checkLogicErrors
    __StrUtilHelper, 26
clearScr
    strUtil, 19
concatStr
    strTools, 13

delSubStr
    strTools, 15

findSubStr
    strTools, 16

ignoreCapturedValue
    __StrUtilHelper, 26
insertStr
    strTools, 16
isCapturedValueInvalid
    strUtil, 19

log
    __StrLogger, 24
loggerStatus
    __StrLogger, 24

main.cpp, 31
makeSmartPtr
    __StrUtilHelper, 28

replaceStr
    strTools, 17

setLogFile
    __StrLogger, 24
strTools, 13
    concatStr, 13
    delSubStr, 15
    findSubStr, 16
    insertStr, 16
    replaceStr, 17
    subStr, 17
strtools.hh, 31
strUtil, 18
    clearScr, 19
    isCapturedValueInvalid, 19
    toLower, 20
    toUpper, 21
    userInputHandler, 22
strutil.hh, 32
strutilhelper.hh, 34
subStr
    strTools, 17

toggleLogger
    __StrLogger, 25
toLower
    strUtil, 20
toSomething
    __StrUtilHelper, 28
toUpper
    strUtil, 21

userInputHandler
    strUtil, 22