

## Compression

Generated by Doxygen 1.9.8



<b>1 Class Index</b>	<b>1</b>
1.1 Class List	1
<b>2 File Index</b>	<b>3</b>
2.1 File List	3
<b>3 Class Documentation</b>	<b>5</b>
3.1 ClearClosingTagsComp Class Reference	5
3.2 Map Class Reference	5
3.2.1 Constructor & Destructor Documentation	5
3.2.1.1 Map()	5
3.2.1.2 ~Map()	6
3.2.2 Member Function Documentation	6
3.2.2.1 add()	6
3.2.2.2 containKey()	6
3.2.2.3 getKey()	7
3.2.2.4 getSize()	7
3.2.2.5 getValue()	7
3.2.2.6 toString()	8
3.3 MinifyingXML Class Reference	8
3.3.1 Constructor & Destructor Documentation	8
3.3.1.1 MinifyingXML()	8
3.3.2 Member Function Documentation	9
3.3.2.1 isSkipChar()	9
3.3.2.2 minifyString()	9
3.3.3 Friends And Related Symbol Documentation	10
3.3.3.1 skipFromBeginning()	10
3.3.3.2 skipFromEnd()	10
3.4 TagsMapComp Class Reference	11
3.4.1 Constructor & Destructor Documentation	11
3.4.1.1 TagsMapComp()	11
3.4.2 Member Function Documentation	11
3.4.2.1 compress()	11
3.4.2.2 mapTags()	12
3.5 TagsMapDec Class Reference	12
<b>4 File Documentation</b>	<b>13</b>
4.1 Sample2/ClearClosingTagsComp.h File Reference	13
4.1.1 Detailed Description	13
4.2 ClearClosingTagsComp.h	13
4.3 Sample2/Map.cpp File Reference	14
4.3.1 Detailed Description	14
4.4 Sample2/Map.h File Reference	14

---

4.4.1 Detailed Description . . . . .	14
4.5 Map.h . . . . .	15
4.6 Sample2/Map_unittest.cpp File Reference . . . . .	15
4.6.1 Detailed Description . . . . .	15
4.7 Sample2/MinifyingXML.cpp File Reference . . . . .	16
4.7.1 Detailed Description . . . . .	16
4.8 Sample2/MinifyingXML.h File Reference . . . . .	16
4.8.1 Detailed Description . . . . .	17
4.9 MinifyingXML.h . . . . .	17
4.10 Sample2/MinifyingXML_unittest.cpp File Reference . . . . .	18
4.10.1 Detailed Description . . . . .	18
4.11 pch.h . . . . .	18
4.12 Sample2/TagMapComp_unittest.cpp File Reference . . . . .	19
4.12.1 Detailed Description . . . . .	19
4.13 Sample2/TagsMapComp.cpp File Reference . . . . .	19
4.13.1 Detailed Description . . . . .	19
4.14 Sample2/TagsMapComp.h File Reference . . . . .	20
4.14.1 Detailed Description . . . . .	20
4.15 TagsMapComp.h . . . . .	20
4.16 Sample2/TagsMapDec.h File Reference . . . . .	21
4.16.1 Detailed Description . . . . .	21
4.17 TagsMapDec.h . . . . .	21
<b>Index</b>	<b>23</b>

# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">ClearClosingTagsComp</a>	5
<a href="#">Map</a>	5
<a href="#">MinifyingXML</a>	8
<a href="#">TagsMapComp</a>	11
<a href="#">TagsMapDec</a>	12



## Chapter 2

# File Index

### 2.1 File List

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

Sample2/ <a href="#">ClearClosingTagsComp.h</a> . . . . .	13
Sample2/ <a href="#">Map.cpp</a>	
The source file of the simple <a href="#">Map</a> . . . . .	14
Sample2/ <a href="#">Map.h</a>	
The header file of the simple <a href="#">Map</a> . . . . .	14
Sample2/ <a href="#">Map_unittest.cpp</a>	
Unit test code for <a href="#">Map</a> class . . . . .	15
Sample2/ <a href="#">MinifyingXML.cpp</a>	
The source file of class <a href="#">MinifyingXML</a> . . . . .	16
Sample2/ <a href="#">MinifyingXML.h</a>	
Header file of the <a href="#">MinifyingXML</a> class . . . . .	16
Sample2/ <a href="#">MinifyingXML_unittest.cpp</a>	
Unit test code for <a href="#">MinifyingXML</a> class . . . . .	18
Sample2/ <a href="#">pch.h</a> . . . . .	18
Sample2/ <a href="#">TagMapComp_unittest.cpp</a>	
Unit test code for <a href="#">TagsMapComp</a> class . . . . .	19
Sample2/ <a href="#">TagsMapComp.cpp</a>	
The source file of <a href="#">TagsMapComp</a> class . . . . .	19
Sample2/ <a href="#">TagsMapComp.h</a>	
The header file of <a href="#">TagsMapComp</a> class . . . . .	20
Sample2/ <a href="#">TagsMapDec.h</a>	
The header file of <a href="#">TagsMapDec</a> class . . . . .	21





## Chapter 3

# Class Documentation

### 3.1 ClearClosingTagsComp Class Reference

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

- Sample2/[ClearClosingTagsComp.h](#)

### 3.2 Map Class Reference

#### Public Member Functions

- **Map** ()  
*C'tor. Initializes empty map with an empty dynamic array.*
- **Map** (const std::string \*tagMapBlock)  
*C'tor. Initialize the map from a <TagMap> block.*
- **~Map** ()
- int **add** (std::string \*key)  
*Adds the key to the map.*
- int **getValue** (const std::string \*key) const  
*The value that the key is mapped to.*
- const std::string \* **getKey** (int value) const  
*Get the key from the value that the key was mapped to.*
- bool **containKey** (const std::string \*key) const  
*Checks if the map contains that key.*
- int **getSize** ()
- std::string \* **toString** ()  
*Returns the <TagMap> block so it can be added to the compressed XML file.*

#### 3.2.1 Constructor & Destructor Documentation

##### 3.2.1.1 Map()

```
Map::Map (
    const std::string * tagMapBlock ) [explicit]
```

C'tor. Initialize the map from a <TagMap> block.

The file must start with <TagMap> and ends with </TagMap> otherwise the file is considered defected.

**Parameters**

<i>tagMapBlock.</i>	
---------------------	--

**Exceptions**

<i>runtime_error</i>	if the file is defected.
----------------------	--------------------------

**3.2.1.2 ~Map()**

```
Map::~~Map ( ) [inline]
```

D'tor.

**Warning**

It will delete all the keys string assigned to it.

**3.2.2 Member Function Documentation****3.2.2.1 add()**

```
int Map::add (
    std::string * key )
```

Adds the key to the map.

**Parameters**

<i>key</i>	To add.
------------	---------

**Returns**

The value that the key is mapped to.

**3.2.2.2 containKey()**

```
bool Map::containKey (
    const std::string * key ) const
```

Checks if the map contains that key.

**Parameters**

<i>key.</i>	
-------------	--

**Returns**

true if the key is available in the map, false otherwise.

**3.2.2.3 getKey()**

```
const std::string * Map::getKey (
    int value ) const
```

Get the key from the value that the key was mapped to.

**Parameters**

<i>value</i>	
--------------	--

**Returns**

The key.

**Exceptions**

<i>runtime</i>	error if the value is not in the map.
----------------	---------------------------------------

**3.2.2.4 getSize()**

```
int Map::getSize ( ) [inline]
```

**Returns**

the size of the map.

**3.2.2.5 getValue()**

```
int Map::getValue (
    const std::string * key ) const
```

The value that the key is mapped to.

**Parameters**

<i>key.</i>	
-------------	--

**Returns**

The value if the key is found, -1 otherwise.

### 3.2.2.6 toString()

```
std::string * Map::toString ( )
```

Returns the <TagMap> block so it can be added to the compressed XML file.

#### Exceptions

<i>runtime</i>	error if the map is empty.
----------------	----------------------------

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

- [Sample2/Map.h](#)
- [Sample2/Map.cpp](#)

## 3.3 MinifyingXML Class Reference

### Public Member Functions

- [MinifyingXML](#) (const std::string \*xmlFile)  
*C'tor.*
- std::string \* [minifyString](#) ()  
*This function deletes any spaces and new lines from the XML File.*
- const std::string \* [getXMLFile](#) () const
- void [setXMLFile](#) (const std::string \*xmlFileNew)

### Static Public Member Functions

- static bool [isSkipChar](#) (const char c)  
*function checks if the char is one the located characters.*

### Related Symbols

(Note that these are not member symbols.)

- void [skipFromBeginning](#) (std::string \*result) const  
*This function clears all the skip Chars except some spaces.*
- void [skipFromEnd](#) (std::string \*result) const  
*This function clears the extra spaces left from the prev step.*

## 3.3.1 Constructor & Destructor Documentation

### 3.3.1.1 MinifyingXML()

```
MinifyingXML::MinifyingXML (
    const std::string * xmlFile ) [inline], [explicit]
```

C'tor.

#### See also

D'tor, this class will not deallocate the XML file string.

## Parameters

<i>xmlFile</i>	
----------------	--

### 3.3.2 Member Function Documentation

#### 3.3.2.1 isSkipChar()

```
bool MinifyingXML::isSkipChar (
    const char c ) [static]
```

function checks if the char is one the located characters.

## Parameters

<i>c</i>	-The character to check.
----------	--------------------------

## Returns

- True if it's a skip char.
- False otherwise.

## See also

MinifyingXML::charToSkip array.

#### 3.3.2.2 minifyString()

```
std::string * MinifyingXML::minifyString ( )
```

This function deletes any spaces and new lines from the XML File.

It removes any charToSkip from the file string, except spaces in the tags values (leading and trailing spaces are removed from the value too).

## See also

MinifyingXML::charToSkip array.

## Returns

The result string from minifying function.

### 3.3.3 Friends And Related Symbol Documentation

#### 3.3.3.1 skipFromBeginning()

```
void MinifyingXML::skipFromBeginning (
    std::string * result ) const [related]
```

This function clears all the skip Chars except some spaces.

Operation:

- For all the string, skip (don't add it into the result) all charToSkip elements except spaces.
- For space cases: -> Starting from the beginning, skip any spaces until reaching the first closing tag '>'. -> After the closing tag, skip any spaces until reaching the first non skip value (any chars not in charToSkip array). -> Add all spaces until reaching the next opening tag '<'.

Example: "<name> Ahmed Ali \n </name>" --> "<name>Ahmed Ali </name>"

helper function for: `std::string* MinifyingXML::minifyString()`. be called before `void MinifyingXML::skipFromBeginning(const std::string* result)`.

Parameters

<i>result</i>	an empty string to store the result of this function.
---------------	---

#### 3.3.3.2 skipFromEnd()

```
void MinifyingXML::skipFromEnd (
    std::string * result ) const [related]
```

This function clears the extra spaces left from the prev step.

Operation: Starting from the last element.

- Clear all the spaces before any opening tag '<' till the prev last non skip char. -> If the current element was the opening tag, skip spaces. -> If the current element is any non skip char, stop skipping spaces.

Example: "<name>Ahmed Ali </name>" --> "<name>Ahmed Ali</name>"

- Other skip chars are eliminated from the prev step.

See also

`void MinifyingXML::skipFromBeginning(std::string* result).`

helper function for: `std::string* MinifyingXML::minifyString()`. be called after `void MinifyingXML::skipFromBeginning(std::string* result)`.

## Parameters

<i>result</i>	The result string from the prev step to modify it.
---------------	--

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

- [Sample2/MinifyingXML.h](#)
- [Sample2/MinifyingXML.cpp](#)

## 3.4 TagsMapComp Class Reference

### Public Member Functions

- [TagsMapComp](#) (const std::string \*xmlFile)  
*C'tor.*
- [~TagsMapComp](#) ()  
*D'tor.*
- void [mapTags](#) ()  
*Reads the XML file and create the map with all the tags.*
- std::string \* [compress](#) (bool addMapTable=false)  
*This function compresses the XML file.*

### 3.4.1 Constructor & Destructor Documentation

#### 3.4.1.1 TagsMapComp()

```
TagsMapComp::TagsMapComp (
    const std::string * xmlFile ) [inline], [explicit]
```

C'tor.

Initializes the XML file, reads it and create a map with all the tags.

## Parameters

<i>xmlFile</i>	
----------------	--

### 3.4.2 Member Function Documentation

#### 3.4.2.1 compress()

```
std::string * TagsMapComp::compress (
    bool addMapTable = false )
```

This function compresses the XML file.

Operation:

- If `addMapTable` is true, it adds the `<TagMap>` block in the first line in the string. Will not added otherwise (is false by default).
- It replaces all the tags (closing and opening) with there mapped value in the map, it also adds 't' before the number just for the rules of XML files. Example: `<TagEg> --> <t0>`, `</TagEg> --> </t0>`

#### Parameters

<i>addMapTable</i>	if true, then a <code>&lt;TagMap&gt;</code> block will be added in the 1st line in the result string.
--------------------	---

#### Returns

A string contains the XML file after the compression.

#### Note

The result string doesn't contain XML version and encoding line.

### 3.4.2.2 mapTags()

```
void TagsMapComp::mapTags ( )
```

Reads the XML file and create the map with all the tags.

Explanation:

- Find the next tag.
- If the tag is in the map, do nothing.
- If the tag is not in the map add it.

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

- [Sample2/TagsMapComp.h](#)
- [Sample2/TagsMapComp.cpp](#)

## 3.5 TagsMapDec Class Reference

#### Public Member Functions

- `std::string * decompress ()`

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

- [Sample2/TagsMapDec.h](#)
- [Sample2/TagsMapDec.cpp](#)



# Chapter 4

## File Documentation

### 4.1 Sample2/ClearClosingTagsComp.h File Reference

```
#include <string>
```

#### Classes

- class [ClearClosingTagsComp](#)

#### 4.1.1 Detailed Description

##### Author

eslam

##### Date

December 2023

### 4.2 ClearClosingTagsComp.h

[Go to the documentation of this file.](#)

```
00001 /*****  
00009 #pragma once  
00010 #ifndef CLEAR_CLOSING_TAGS_COMP_H  
00011 #define CLEAR_CLOSING_TAGS_COMP_H  
00012  
00013 #include <string>  
00014  
00015 class ClearClosingTagsComp  
00016 {  
00017 }; // class ClearClosingTagsComp  
00018 #endif // !CLEAR_CLOSING_TAGS_COMP_H
```

## 4.3 Sample2/Map.cpp File Reference

The source file of the simple [Map](#).

```
#include "pch.h"
#include "Map.h"
```

### 4.3.1 Detailed Description

The source file of the simple [Map](#).

This a simple implementation of [Map](#) DS that will help Mapping tags into numbers.

**Author**

eslam

**Date**

December 2023

## 4.4 Sample2/Map.h File Reference

The header file of the simple [Map](#).

```
#include <vector>
#include "MinifyingXML.h"
#include <sstream>
```

**Classes**

- class [Map](#)

### 4.4.1 Detailed Description

The header file of the simple [Map](#).

This a simple implementation of [Map](#) DS that will help Mapping tags into numbers.

**Author**

eslam

**Date**

December 2023

## 4.5 Map.h

[Go to the documentation of this file.](#)

```

00001  /*****
00012  #pragma once
00013  #ifndef MAP_H
00014  #define MAP_H
00015
00016  #include <vector>
00017  #include "MinifyingXML.h"
00018  #include <sstream>
00019
00020  class Map
00021  {
00022  private:
00023      std::vector<std::string*> arr;
00024
00025      //helper method
00031      static void trimString(std::string& str);
00032
00033  public:
00038      explicit Map() :arr(new std::vector<std::string*>()) {}
00048      explicit Map(const std::string* tagMapBlock);
00053      ~Map() {
00054          for (std::string* s : *arr) {
00055              delete s;
00056          }
00057          delete arr;
00058      }
00059
00060      //methods
00061
00068      int add(std::string* key);
00075      int getValue(const std::string* key) const;
00083      const std::string* getKey(int value) const;
00091      bool containKey(const std::string* key) const;
00092
00096      int getSize() { return arr->size(); }
00097
00104      std::string* toString();
00105  };
00106
00107  #endif // !MAP_H

```

## 4.6 Sample2/Map\_unittest.cpp File Reference

Unit test code for [Map](#) class.

```

#include "gtest/gtest.h"
#include "pch.h"
#include "Map.h"

```

### 4.6.1 Detailed Description

Unit test code for [Map](#) class.

#### Author

eslam

#### Date

December 2023

## 4.7 Sample2/MinifyingXML.cpp File Reference

The source file of class [MinifyingXML](#).

```
#include "pch.h"
#include "MinifyingXML.h"
```

### 4.7.1 Detailed Description

The source file of class [MinifyingXML](#).

Minifying is one of the required functions in the data structure and algorithms course's project. Minifying is a way of decreasing the size of the file by deleting all spaces, tabs, new lines.

This class will minify any flawless XML file.

Operation summary:

- Using the array charToSkip.
- All charToSkip (except the space : ' ') will not be added into the result array.
- Spaces will be added to the result string only if it occurred inside the tag's value, not before or after the value. i.e., "<tagg> value with spaces </tagg>" → "apply minifying--> "<tagg>value with spaces</tagg>", on other words, the value will be trimmed from spaces before or after it.

Author

eslam

Date

December 2023

## 4.8 Sample2/MinifyingXML.h File Reference

Header file of the [MinifyingXML](#) class.

```
#include <string>
#include <stdexcept>
```

Classes

- class [MinifyingXML](#)

### 4.8.1 Detailed Description

Header file of the [MinifyingXML](#) class.

Minifying is one of the required functions in the data structure and algorithms course's project. Minifying is a way of decreasing the size of the file by deleting all spaces, tabs, new lines.

This class will minify any flawless XML file.

Operation summary:

- Using the array charToSkip.
- All charToSkip (except the space : ' ') will not be added into the result array.
- Spaces will be added to the result string only if it occurred inside the tag's value, not before or after the value. i.e., "<tag> value with spaces </tag>" → "<tag>value with spaces</tag>", on other words, the value will be trimmed from spaces before or after it.

Author

eslam

Date

December 2023

## 4.9 MinifyingXML.h

[Go to the documentation of this file.](#)

```
00001 /*****
00022 #pragma once
00023 #ifndef MINIFYING_XML_H
00024 #define MINIFYING_XML_H
00025 #include <string>
00026
00027 #include <stdexcept>
00028
00029 class MinifyingXML
00030 {
00031 private:
00032     // the file that needs to be minified.
00033     const std::string* xmlFile;
00034
00035     // char To skip in minifying.
00036     static const char charToSkip[5];
00037
00038 public:
00045     explicit MinifyingXML(const std::string* xmlFile) : xmlFile(xmlFile) {
00046         // check adding a null ptr.
00047         if (xmlFile == nullptr) {
00048             throw std::logic_error("Null pointer exception: Accessing null pointer!");
00049         }
00050     }
00051
00052     //methods
00053
00064     std::string* minifyString();
00065
00075     static bool isSkipChar(const char c);
00076
00077     //helper methods
00078
00097     void skipFromBeginning(std::string* result) const;
00098
00119     void skipFromEnd(std::string* result) const;
```

```

00120
00121 //getters and setters, used for debugging
00122 //getters.
00123
00124 //XML file getter.
00125 const std::string* getXMLFile() const { return this->xmlFile; }
00126
00127 //setters
00128 //XML file setter.
00129 void setXMLFile(const std::string* xmlFileNew) {
00130     // check adding a null ptr.
00131     if (xmlFileNew == nullptr) {
00132         throw std::logic_error("Null pointer exception: Accessing null pointer!");
00133     }
00134     this->xmlFile = xmlFileNew;
00135 }
00136 }; //class MinifyingXML
00137
00138 #endif // !MINIFYING_XML_H

```

## 4.10 Sample2/MinifyingXML\_unittest.cpp File Reference

Unit test code for [MinifyingXML](#) class.

```

#include "gtest/gtest.h"
#include "pch.h"
#include "MinifyingXML.h"

```

### 4.10.1 Detailed Description

Unit test code for [MinifyingXML](#) class.

It includes a test for each member method in the class using gtest framework.

#### Author

eslam

#### Date

December 2023

## 4.11 pch.h

```

00001 //
00002 // pch.h
00003 //
00004
00005 #pragma once
00006
00007 #include "gtest/gtest.h"
00008
00009 int main(int argc, char** argv) {
00010     testing::InitGoogleTest(&argc, argv);
00011
00012     // Run all tests
00013     return RUN_ALL_TESTS();
00014 }

```

## 4.12 Sample2/TagMapComp\_unittest.cpp File Reference

Unit test code for [TagsMapComp](#) class.

```
#include "gtest/gtest.h"
#include "pch.h"
#include "TagsMapComp.h"
```

### 4.12.1 Detailed Description

Unit test code for [TagsMapComp](#) class.

Author

eslam

Date

December 2023

## 4.13 Sample2/TagsMapComp.cpp File Reference

The source file of [TagsMapComp](#) class.

```
#include "pch.h"
#include "TagsMapComp.h"
```

### 4.13.1 Detailed Description

The source file of [TagsMapComp](#) class.

A compression algorithm that maps tags into numbers. By applying this algorithm, the size file decrease, as many characters in tags will be getting red off, so theses char will not repeated over and over again.

To now the mapping values, a `<TagsMap>` block will be added to the start of the XML file.

Example: -> File before: `<tag0><tag1><tag2></tag2><tag2></tag2></tag1></tag0>`

-> File after: `<TagMap>tag0,tag1,tag2<Tag/Map> <t0><t1><t2></t2><t2></t2></t1></t0>`

Note

: `<TagMap>` block is optional, Will not be added to the social network file, is tags are constant there.

: if `<TagMap>` is added, this algorithm will be efficient only if it contains lots of long tags.

all methods in this class assumes that the input file is flawless.

Author

eslam

Date

December 2023

## 4.14 Sample2/TagsMapComp.h File Reference

The header file of [TagsMapComp](#) class.

```
#include <string>
#include "Map.h"
```

### Classes

- class [TagsMapComp](#)

### 4.14.1 Detailed Description

The header file of [TagsMapComp](#) class.

A compression algorithm that maps tags into numbers. By applying this algorithm, the size file decrease, as many characters in tags will be getting red off, so theses char will not repeated over and over again.

To now the mapping values, a `<TagsMap>` block will be added to the start of the XML file.

Example: -> File before: `<tag0><tag1><tag2></tag2><tag2></tag2></tag1></tag0>`

-> File after: `<TagMap>tag0,tag1,tag2<Tag/Map> <t0><t1><t2></t2><t2></t2></t1></t0>`

#### Note

: `<TagMap>` block is optional, Will not be added to the social network file, is tags are constant there.

: if `<TagMap>` is added, this algorithm will be efficient only if it contains lots of long tags.

all methods in this class assumes that the input file is flawless.

#### Author

eslam

#### Date

December 2023

## 4.15 TagsMapComp.h

[Go to the documentation of this file.](#)

```
00001 /*****
00031 #pragma once
00032 #ifndef TAGS_MAP_Comp_H
00033 #define TAGS_MAP_Comp_H
00034
00035 #include <string>
00036 #include "Map.h"
00037
00038 class TagsMapComp
00039 {
00040 private:
00041     const std::string* xmlFile;
00042     //Map of tag values.
00043     Map* map;
00044 public:
00053     explicit TagsMapComp(const std::string* xmlFile) : xmlFile(xmlFile),
00054         map(new Map()) {
00055         mapTags();
00056     }
00061     ~TagsMapComp() { delete map; }
00070     void mapTags();
00071
00089     std::string* compress(bool addMapTable = false);
00090 };
00091
00092 #endif // !TAGS_MAP_Comp_H
```



## 4.16 Sample2/TagsMapDec.h File Reference

The header file of [TagsMapDec](#) class.

```
#include "Map.h"
```

### Classes

- class [TagsMapDec](#)

### 4.16.1 Detailed Description

The header file of [TagsMapDec](#) class.

The decompression algorithm of TagsMap compression algorithm. The decompression will re-map the tags to their original value.

The file might contain a TagsMap tag at the beginning, from that tag we can get the mapping numbers.

If the file doesn't contain this tag, then it will be assumed to be: `<TagMap>users,user,id,name,posts,post,body,topics,topic,followers,fo`  
`TagMap>`. which will be used for social network system only.

Example: -> File before: `<TagMap>tag0,tag1,tag2<Tag/Map> <t0><t1><t2></t2><t2></t2></t1></t0>`

-> File after: `<tag0><tag1><tag2></tag2><tag2></tag2></tag1></tag0>`

See also

[TagsMapComp](#)

Author

eslam

Date

December 2023

## 4.17 TagsMapDec.h

[Go to the documentation of this file.](#)

```
00001 /*****
00028 #pragma once
00029 #ifndef TAGS_MAP_DEC_H
00030 #define TAGS_MAP_DEC_H
00031
00032 #include "Map.h"
00033
00034 class TagsMapDec
00035 {
00036 private:
00037     const std::string* xmlFile;
00038
00039     const static std::string* defaultTagMapBlock;
00040
00041     //Map of tag values.
00042     Map* map;
00043
00044     //helper methods
00045     void getMapTags();
00046     void getTagsMapBlock();
00047 public:
00048     explicit TagsMapDec();
00049     ~TagsMapDec();
00050
00051     std::string* decompress();
00052 };
00053 #endif // !TAGS_MAP_DEC_H
```



# Index

- ~Map
  - Map, [6](#)
- add
  - Map, [6](#)
- ClearClosingTagsComp, [5](#)
- compress
  - TagsMapComp, [11](#)
- containKey
  - Map, [6](#)
- getKey
  - Map, [7](#)
- getSize
  - Map, [7](#)
- getValue
  - Map, [7](#)
- isSkipChar
  - MinifyingXML, [9](#)
- Map, [5](#)
  - ~Map, [6](#)
  - add, [6](#)
  - containKey, [6](#)
  - getKey, [7](#)
  - getSize, [7](#)
  - getValue, [7](#)
  - Map, [5](#)
  - toString, [7](#)
- mapTags
  - TagsMapComp, [12](#)
- MinifyingXML, [8](#)
  - isSkipChar, [9](#)
  - MinifyingXML, [8](#)
  - minifyString, [9](#)
  - skipFromBeginning, [10](#)
  - skipFromEnd, [10](#)
- minifyString
  - MinifyingXML, [9](#)
- Sample2/ClearClosingTagsComp.h, [13](#)
- Sample2/Map.cpp, [14](#)
- Sample2/Map.h, [14](#), [15](#)
- Sample2/Map\_unittest.cpp, [15](#)
- Sample2/MinifyingXML.cpp, [16](#)
- Sample2/MinifyingXML.h, [16](#), [17](#)
- Sample2/MinifyingXML\_unittest.cpp, [18](#)
- Sample2/pch.h, [18](#)
- Sample2/TagMapComp\_unittest.cpp, [19](#)
- Sample2/TagsMapComp.cpp, [19](#)
- Sample2/TagsMapComp.h, [20](#)
- Sample2/TagsMapDec.h, [21](#)
- skipFromBeginning
  - MinifyingXML, [10](#)
- skipFromEnd
  - MinifyingXML, [10](#)
- TagsMapComp, [11](#)
  - compress, [11](#)
  - mapTags, [12](#)
  - TagsMapComp, [11](#)
- TagsMapDec, [12](#)
- toString
  - Map, [7](#)