

Compression task

Generated by Doxygen 1.9.8

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 ClearClosingTagsComp Class Reference	5
3.1.1 Detailed Description	5
3.1.2 Constructor & Destructor Documentation	5
3.1.2.1 ClearClosingTagsComp()	5
3.1.3 Member Function Documentation	6
3.1.3.1 compress()	6
3.2 ClearClosingTagsDec Class Reference	6
3.2.1 Detailed Description	6
3.2.2 Constructor & Destructor Documentation	6
3.2.2.1 ClearClosingTagsDec()	6
3.2.2.2 ~ClearClosingTagsDec()	7
3.2.3 Member Function Documentation	7
3.2.3.1 decompress()	7
3.3 Map Class Reference	8
3.3.1 Detailed Description	8
3.3.2 Constructor & Destructor Documentation	8
3.3.2.1 Map() [1/2]	8
3.3.2.2 Map() [2/2]	8
3.3.2.3 ~Map()	9
3.3.3 Member Function Documentation	9
3.3.3.1 add()	9
3.3.3.2 containKey()	9
3.3.3.3 getKey()	10
3.3.3.4 getSize()	10
3.3.3.5 getValue()	10
3.3.3.6 toString()	11
3.4 MinifyingXML Class Reference	11
3.4.1 Detailed Description	12
3.4.2 Constructor & Destructor Documentation	12
3.4.2.1 MinifyingXML()	12
3.4.3 Member Function Documentation	12
3.4.3.1 getXMLFile()	12
3.4.3.2 isSkipChar()	12
3.4.3.3 minifyString()	13
3.4.3.4 setXMLFile()	13
3.4.4 Friends And Related Symbol Documentation	13

3.4.4.1 skipFromBeginning()	13
3.4.4.2 skipFromEnd()	14
3.5 TagsMapComp Class Reference	14
3.5.1 Detailed Description	15
3.5.2 Constructor & Destructor Documentation	15
3.5.2.1 TagsMapComp()	15
3.5.2.2 ~TagsMapComp()	15
3.5.3 Member Function Documentation	15
3.5.3.1 compress()	15
3.5.3.2 mapTags()	16
3.6 TagsMapDec Class Reference	16
3.6.1 Detailed Description	17
3.6.2 Constructor & Destructor Documentation	17
3.6.2.1 TagsMapDec()	17
3.6.2.2 ~TagsMapDec()	17
3.6.3 Member Function Documentation	17
3.6.3.1 decompress()	17
3.7 Tree Class Reference	18
3.7.1 Detailed Description	18
3.7.2 Constructor & Destructor Documentation	18
3.7.2.1 Tree()	18
3.7.2.2 ~Tree()	19
3.7.3 Member Function Documentation	19
3.7.3.1 getRoot()	19
3.7.3.2 print()	19
3.8 TreeNode Class Reference	19
3.8.1 Detailed Description	20
3.8.2 Constructor & Destructor Documentation	20
3.8.2.1 TreeNode()	20
3.8.2.2 ~TreeNode()	20
3.8.3 Member Function Documentation	20
3.8.3.1 getChild()	20
3.8.3.2 getParent()	21
3.8.3.3 getValue()	21
3.8.3.4 isChild()	21
3.8.4 Friends And Related Symbol Documentation	21
3.8.4.1 Tree	21
4 File Documentation	23
4.1 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Compression/ClearClosing↔ TagsComp.cpp File Reference	23
4.1.1 Detailed Description	23

4.2 ClearClosingTagsComp.cpp	24
4.3 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and- Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Compression/ClearClosing TagsComp.h File Reference	24
4.3.1 Detailed Description	25
4.3.2 Macro Definition Documentation	25
4.3.2.1 CLEAR_CLOSING_TAGS_COMP_H	25
4.4 ClearClosingTagsComp.h	25
4.5 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and- Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Compression/ClearClosing TagsComp_unittest.cpp File Reference	26
4.5.1 Detailed Description	26
4.6 ClearClosingTagsComp_unittest.cpp	26
4.7 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure- and-Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Decompression/Clear ClosingTagsDec.cpp File Reference	27
4.7.1 Detailed Description	27
4.8 ClearClosingTagsDec.cpp	27
4.9 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure- and-Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Decompression/Clear ClosingTagsDec.h File Reference	29
4.9.1 Detailed Description	29
4.9.2 Macro Definition Documentation	30
4.9.2.1 CLEAR_CLOSING_TAGS_DEC_H	30
4.10 ClearClosingTagsDec.h	30
4.11 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure- and-Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Decompression/Clear ClosingTagsDec_unittest.cpp File Reference	30
4.11.1 Detailed Description	30
4.12 ClearClosingTagsDec_unittest.cpp	31
4.13 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and- Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Helper/Tree.cpp File Reference	31
4.13.1 Detailed Description	31
4.14 Tree.cpp	32
4.15 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and- Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Helper/Tree.h File Reference	33
4.15.1 Detailed Description	33
4.15.2 Macro Definition Documentation	34
4.15.2.1 TREE_H	34
4.16 Tree.h	34
4.17 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and- Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Helper/TreeNode.cpp File Ref- erence	34
4.17.1 Detailed Description	35
4.18 TreeNode.cpp	35

4.19 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Helper/TreeNode.h File Reference	35
4.19.1 Detailed Description	36
4.19.2 Macro Definition Documentation	36
4.19.2.1 TREE_NODE_H	36
4.20 TreeNode.h	36
4.21 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/Minifying/MinifyingXML.cpp File Reference	37
4.21.1 Detailed Description	37
4.22 MinifyingXML.cpp	38
4.23 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/Minifying/MinifyingXML.h File Reference	39
4.23.1 Detailed Description	39
4.23.2 Macro Definition Documentation	40
4.23.2.1 MINIFYING_XML_H	40
4.24 MinifyingXML.h	40
4.25 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/Minifying/MinifyingXML_unittest.cpp File Reference	41
4.25.1 Detailed Description	41
4.26 MinifyingXML_unittest.cpp	42
4.27 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Compression/TagsMapComp.cpp File Reference	44
4.27.1 Detailed Description	44
4.28 TagsMapComp.cpp	45
4.29 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Compression/TagsMapComp.h File Reference	46
4.29.1 Detailed Description	47
4.29.2 Macro Definition Documentation	47
4.29.2.1 TAGS_MAP_Comp_H	47
4.30 TagsMapComp.h	47
4.31 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Compression/TagsMapComp_↔ unittest.cpp File Reference	48
4.31.1 Detailed Description	48
4.32 TagsMapComp_unittest.cpp	48
4.33 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Decompression/TagsMapDec.cpp File Reference	50
4.33.1 Detailed Description	50
4.34 TagsMapDec.cpp	51

4.35 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Decompression/TagsMapDec.h File Reference	52
4.35.1 Detailed Description	53
4.35.2 Macro Definition Documentation	53
4.35.2.1 TAGS_MAP_DEC_H	53
4.36 TagsMapDec.h	54
4.37 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Decompression/TagsMapDec_↔ unittest.cpp File Reference	54
4.37.1 Detailed Description	54
4.38 TagsMapDec_unittest.cpp	55
4.39 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Helper/Map.cpp File Reference . . .	56
4.39.1 Detailed Description	56
4.40 Map.cpp	57
4.41 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Helper/Map.h File Reference . . .	58
4.41.1 Detailed Description	58
4.41.2 Macro Definition Documentation	59
4.41.2.1 MAP_H	59
4.42 Map.h	59
4.43 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Helper/Map_unittest.cpp File Refer- ence	59
4.43.1 Detailed Description	60
4.44 Map_unittest.cpp	60
Index	63

Chapter 1

Class Index

1.1 Class List

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

ClearClosingTagsComp	5
ClearClosingTagsDec	6
Map	8
MinifyingXML	11
TagsMapComp	14
TagsMapDec	16
Tree	18
TreeNode	19

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Compression/ ClearClosingTagsComp.cpp The source file of ClearClosingTagsComp class	23
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Compression/ ClearClosingTagsComp.h The header file of ClearClosingTagsComp class	24
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Compression/ ClearClosingTagsComp_unittest.cpp Unit test code for ClearClosingTagsComp class	26
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Decompression/ ClearClosingTagsDec.cpp The source file of ClearClosingTagsDec class	27
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Decompression/ ClearClosingTagsDec.h The header file of ClearClosingTagsDec class	29
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Decompression/ ClearClosingTagsDec_unittest.cpp Unit test code for ClearClosingTagsDec class	30
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Helper/ Tree.cpp A simple Tree DS implementation	31
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Helper/ Tree.h A simple Tree DS implementation	33
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Helper/ TreeNode.cpp A simple Tree Node for the tree DS	34
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Helper/ TreeNode.h A simple Tree Node for the tree DS	35
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/Minifying/ MinifyingXML.cpp The source file of class MinifyingXML	37
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↔ Algorithms-Project/WorkSpace/Compression/XML/Minifying/ MinifyingXML.h Header file of the MinifyingXML class	39

D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and- Algorithms-Project/WorkSpace/Compression/XML/Minifying/MinifyingXML_unittest.cpp	41
Unit test code for MinifyingXML class	
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and- Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Compression/TagsMapComp.cpp	44
The source file of TagsMapComp class	
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and- Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Compression/TagsMapComp.h	46
The header file of TagsMapComp class	
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and- Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Compression/TagsMapComp_unittest.cpp	48
Unit test code for TagsMapComp class	
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and- Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Decompression/TagsMapDec.cpp	50
The header file of TagsMapDec class	
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and- Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Decompression/TagsMapDec.h	52
The header file of TagsMapDec class	
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and- Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Decompression/TagsMapDec_unittest.cpp	54
Unit test code for TagsMapDec class	
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and- Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Helper/Map.cpp	56
The source file of the simple Map	
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and- Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Helper/Map.h	58
The header file of the simple Map	
D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and- Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Helper/Map_unittest.cpp	59
Unit test code for Map class	

Chapter 3

Class Documentation

3.1 ClearClosingTagsComp Class Reference

```
#include <ClearClosingTagsComp.h>
```

Public Member Functions

- [ClearClosingTagsComp](#) (const std::string *xmlFile)
C'tor.
- std::string * [compress](#) ()
This function compresses the XML file.

3.1.1 Detailed Description

Definition at line [32](#) of file [ClearClosingTagsComp.h](#).

3.1.2 Constructor & Destructor Documentation

3.1.2.1 ClearClosingTagsComp()

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

C'tor.

Initializes the XML file.

Parameters

<i>the</i>	XML file without the XML version and encoding line.
------------	---

Definition at line [45](#) of file [ClearClosingTagsComp.h](#).

3.1.3 Member Function Documentation

3.1.3.1 compress()

```
std::string * ClearClosingTagsComp::compress ( )
```

This function compresses the XML file.

Operation summary:

- Find the closing tag.
- Delete the closing tag.

Returns

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

@Warning use only with social network data.

Definition at line 29 of file [ClearClosingTagsComp.cpp](#).

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

- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-
Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Compression/[ClearClosingTagsComp.h](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-
Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Compression/[ClearClosingTagsComp.cpp](#)

3.2 ClearClosingTagsDec Class Reference

```
#include <ClearClosingTagsDec.h>
```

Public Member Functions

- [ClearClosingTagsDec](#) (const std::string *xmlFile)
C'tor that initializes the XML string with provided value, and empty tagsTree, and empty tagsStack.
- [~ClearClosingTagsDec](#) ()
D'tor.
- std::string * [decompress](#) () const
This function decompresses the XML file.

3.2.1 Detailed Description

Definition at line 32 of file [ClearClosingTagsDec.h](#).

3.2.2 Constructor & Destructor Documentation

3.2.2.1 ClearClosingTagsDec()

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

C'tor that initializes the XML string with provided value, and empty tagsTree, and empty tagsStack.

Parameters

<i>the</i>	XML file without the XML version and encoding line.
------------	---

Definition at line 51 of file [ClearClosingTagsDec.h](#).

3.2.2.2 ~ClearClosingTagsDec()

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

D'tor.

Definition at line 57 of file [ClearClosingTagsDec.h](#).

3.2.3 Member Function Documentation

3.2.3.1 decompress()

```
std::string * ClearClosingTagsDec::decompress ( ) const
```

This function decompresses the XML file.

Operation summary:

- collect each opening tag in the stack.
- when reaching a new tag, check from the tree if the current tag (in top of the stack) contains the next tag as a child this don't close the current tag, and add the next tag into the stack.
- if it wasn't a child, add a closing tag for the current tag, then check the same with the next value in the stack.
- At the end add all the remaining tags in the stack in their order.



Returns

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

@Warning use only with social network data.

Definition at line 56 of file [ClearClosingTagsDec.cpp](#).

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

- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and- Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Decompression/[ClearClosingTagsDec.h](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and- Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Decompression/[ClearClosingTagsDec.cpp](#)

3.3 Map Class Reference

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

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.

3.3.1 Detailed Description

Definition at line 20 of file [Map.h](#).

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Map() [1/2]

```
Map::Map ( ) [inline], [explicit]
```

C'tor. Initializes empty map with an empty dynamic array.

Definition at line 38 of file [Map.h](#).

3.3.2.2 Map() [2/2]

```
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.
----------------------	--------------------------

Definition at line 21 of file [Map.cpp](#).

3.3.2.3 ~Map()

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

D'tor.

Warning

It will delete all the keys string assigned to it.

Definition at line 53 of file [Map.h](#).

3.3.3 Member Function Documentation**3.3.3.1 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.

Definition at line 60 of file [Map.cpp](#).

3.3.3.2 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.

Definition at line 89 of file [Map.cpp](#).

3.3.3.3 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.
----------------	---------------------------------------

Definition at line 78 of file [Map.cpp](#).

3.3.3.4 getSize()

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

Returns

the size of the map.

Definition at line 96 of file [Map.h](#).

3.3.3.5 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.

Definition at line 66 of file [Map.cpp](#).

3.3.3.6 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.
----------------	----------------------------

Definition at line 93 of file [Map.cpp](#).

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

- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Helper/[Map.h](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Helper/[Map.cpp](#)

3.4 MinifyingXML Class Reference

```
#include <MinifyingXML.h>
```

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.

3.4.1 Detailed Description

Definition at line 29 of file [MinifyingXML.h](#).

3.4.2 Constructor & Destructor Documentation

3.4.2.1 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>	
----------------	--

Definition at line 45 of file [MinifyingXML.h](#).

3.4.3 Member Function Documentation

3.4.3.1 getXMLFile()

```
const std::string * MinifyingXML::getXMLFile ( ) const [inline]
```

Definition at line 125 of file [MinifyingXML.h](#).

3.4.3.2 isSkipChar()

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

function checks if the char is one the located characters.

Parameters

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

Returns

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

See also

MinifyingXML::charToSkip array.

Definition at line 130 of file [MinifyingXML.cpp](#).

3.4.3.3 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.

Definition at line 29 of file [MinifyingXML.cpp](#).

3.4.3.4 setXMLFile()

```
void MinifyingXML::setXMLFile (
    const std::string * xmlFileNew ) [inline]
```

Definition at line 129 of file [MinifyingXML.h](#).

3.4.4 Friends And Related Symbol Documentation**3.4.4.1 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.
---------------	---

Definition at line 54 of file [MinifyingXML.cpp](#).

3.4.4.2 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.
---------------	--

Definition at line 98 of file [MinifyingXML.cpp](#).

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

- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/Minifying/[MinifyingXML.h](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/Minifying/[MinifyingXML.cpp](#)

3.5 TagsMapComp Class Reference

```
#include <TagsMapComp.h>
```

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.

3.5.1 Detailed Description

Definition at line 38 of file [TagsMapComp.h](#).

3.5.2 Constructor & Destructor Documentation

3.5.2.1 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>the</i>	XML file without the XML version and encoding line.
------------	---

Definition at line 54 of file [TagsMapComp.h](#).

3.5.2.2 ~TagsMapComp()

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

D'tor.

Definition at line 62 of file [TagsMapComp.h](#).

3.5.3 Member Function Documentation

3.5.3.1 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

<code>addMapTable</code>	if true, then a <code><TagMap></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.

Definition at line 68 of file [TagsMapComp.cpp](#).

3.5.3.2 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.

Definition at line 39 of file [TagsMapComp.cpp](#).

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

- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↵ Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Compression/[TagsMapComp.h](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↵ Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Compression/[TagsMapComp.cpp](#)

3.6 TagsMapDec Class Reference

```
#include <TagsMapDec.h>
```


Public Member Functions

- [TagsMapDec](#) (const std::string *xmlFile)
C'tor. -Initialize the [Map](#) with the TagMap block.
- [~TagsMapDec](#) ()
D'tor.
- std::string * [decompress](#) ()
This method decompresses the XML file.

3.6.1 Detailed Description

Definition at line 36 of file [TagsMapDec.h](#).

3.6.2 Constructor & Destructor Documentation

3.6.2.1 TagsMapDec()

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

C'tor. -Initialize the [Map](#) with the TagMap block.

Parameters

<i>the</i>	XML file without the XML version and encoding line.
------------	---

Definition at line 73 of file [TagsMapDec.h](#).

3.6.2.2 ~TagsMapDec()

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

D'tor.

Definition at line 80 of file [TagsMapDec.h](#).

3.6.3 Member Function Documentation

3.6.3.1 decompress()

```
std::string * TagsMapDec::decompress ( )
```

This method decompresses the XML file.

See also

`TagMapComp::compress()` for the functionality.

Returns

the file data after decompression.

Definition at line 77 of file [TagsMapDec.cpp](#).

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

- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Decompression/[TagsMapDec.h](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/TagMapping/Decompression/[TagsMapDec.cpp](#)

3.7 Tree Class Reference

```
#include <Tree.h>
```

Public Member Functions

- [Tree](#) ()
Initialize the [Tree](#) in that arrange for social network system:
- [~Tree](#) ()
D'tor.
- [TreeNode](#) * [getRoot](#) ()
- void [print](#) () const

3.7.1 Detailed Description

Definition at line 28 of file [Tree.h](#).

3.7.2 Constructor & Destructor Documentation

3.7.2.1 [Tree](#)()

```
Tree::Tree ( ) [explicit]
```

Initialize the [Tree](#) in that arrange for social network system:

- users --> {user}
- user --> {id,name,posts,followers}
- posts --> {post}
- post --> {body, topics}
- topics --> {topic}
- followers --> {follower}
- follower --> {id}
- not mentioned: doesn't have a child.

Definition at line 41 of file [Tree.cpp](#).

3.7.2.2 ~Tree()

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

D'tor.

Definition at line 51 of file [Tree.h](#).

3.7.3 Member Function Documentation

3.7.3.1 getRoot()

```
TreeNode * Tree::getRoot ( ) [inline]
```

Definition at line 56 of file [Tree.h](#).

3.7.3.2 print()

```
void Tree::print ( ) const [inline]
```

Definition at line 58 of file [Tree.h](#).

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

- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/Workspace/Compression/XML/ClearClosingTag/Helper/[Tree.h](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/Workspace/Compression/XML/ClearClosingTag/Helper/[Tree.cpp](#)

3.8 TreeNode Class Reference

```
#include <TreeNode.h>
```

Public Member Functions

- [TreeNode](#) (const [TreeNode](#) *parentNode, std::vector< [TreeNode](#) * > *children, std::string *value)
- [~TreeNode](#) ()
- const [TreeNode](#) * [getChild](#) (const std::string *value) const
Returns The child with the assigned value.
- const [TreeNode](#) * [getParent](#) () const
Returns the parent of this node.
- bool [isChild](#) (const std::string *value) const
check if the value is for a child or not.
- std::string [getValue](#) () const

Friends

- class [Tree](#)

3.8.1 Detailed Description

Definition at line 14 of file [TreeNode.h](#).

3.8.2 Constructor & Destructor Documentation

3.8.2.1 [TreeNode\(\)](#)

```
TreeNode::TreeNode (
    const TreeNode * parentNode,
    std::vector< TreeNode * > * children,
    std::string * value ) [inline], [explicit]
```

Definition at line 23 of file [TreeNode.h](#).

3.8.2.2 [~TreeNode\(\)](#)

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

Definition at line 26 of file [TreeNode.h](#).

3.8.3 Member Function Documentation

3.8.3.1 [getChild\(\)](#)

```
const TreeNode * TreeNode::getChild (
    const std::string * value ) const
```

Returns The child with the assigned value.

Parameters

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

Returns

child [TreeNode](#)

Definition at line 12 of file [TreeNode.cpp](#).

3.8.3.2 getParent()

```
const TreeNode * TreeNode::getParent ( ) const [inline]
```

Returns the parent of this node.

Definition at line 45 of file [TreeNode.h](#).

3.8.3.3 getValue()

```
std::string TreeNode::getValue ( ) const [inline]
```

Definition at line 55 of file [TreeNode.h](#).

3.8.3.4 isChild()

```
bool TreeNode::isChild (
    const std::string * value ) const [inline]
```

check if the value is for a child or not.

Parameters

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

Returns

true if found, false otherwise.

Definition at line 52 of file [TreeNode.h](#).

3.8.4 Friends And Related Symbol Documentation

3.8.4.1 Tree

```
friend class Tree [friend]
```

Definition at line 16 of file [TreeNode.h](#).

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

- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↵ Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Helper/[TreeNode.h](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-↵ Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Helper/[TreeNode.cpp](#)

Chapter 4

File Documentation

4.1 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/↵ Compression/XML/ClearClosingTag/Compression/ClearClosing↵ TagsComp.cpp File Reference

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

```
#include "pch.h"
#include "ClearClosingTagsComp.h"
```

4.1.1 Detailed Description

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

- This algorithm of compression is based on deleting all the ending tags to reduce the space of the xmlFile.
- It requires knowing what tags comes after other to know how to decompress the file.

Example:

- File before: <tag0><tag1><tag2>d1</tag2><tag2>d2</tag2></tag1></tag0>
- File after: <tag0><tag1><tag2>d1<tag2>d2

@TODO update the file to work with any type of XML data. Use Trees to record the order of the tags.

@Warning This implementation only works for Social network system, needs an update.

Author

eslam

Date

December 2023

Definition in file [ClearClosingTagsComp.cpp](#).

4.2 ClearClosingTagsComp.cpp

Go to the documentation of this file.

```

00001 /*****
00026 #include "pch.h"
00027 #include "ClearClosingTagsComp.h"
00028
00029 std::string* ClearClosingTagsComp::compress()
00030 {
00031     // to store the result.
00032     std::string* result = new std::string();
00033     //length of the original file.
00034     int length = this->xmlFile->size();
00035
00036
00037     // The max size of the result string is the same of the entered string.
00038     result->reserve(length);
00039
00040     /*
00041     * Loop for all the original string.
00042     * - If the current string is '</'
00043     *     1.skip that tag (increment i till the end of the tag).
00044     *     2.Don't add it to the result string.
00045     * - For other characters, add them to the result.
00046     */
00047     char currentChar = 0;
00048     for (int i = 0; i < length; i++) {
00049         // get current char
00050         currentChar = this->xmlFile->at(i);
00051         if (currentChar == '<' && this->xmlFile->at(i + 1) == '/') {
00052             //skip the closing tag
00053             i = this->xmlFile->find('>', i);
00054             continue;
00055         }
00056         result->append(1, currentChar);
00057     }
00058
00059     // Free the extra allocated memory locations.
00060     result->shrink_to_fit();
00061     return result;
00062 }// compress()

```

4.3 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/↵ Compression/XML/ClearClosingTag/Compression/ClearClosing↵ TagsComp.h File Reference

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

```
#include <string>
```

Classes

- class [ClearClosingTagsComp](#)

Macros

- #define [CLEAR_CLOSING_TAGS_COMP_H](#)

4.3.1 Detailed Description

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

- This algorithm of compression is based on deleting all the ending tags to reduce the space of the xmlFile.
- It requires knowing what tags comes after other to know how to decompress the file.

Example:

- File before: <tag0><tag1><tag2>d1</tag2><tag2>d2</tag2></tag1></tag0>
- File after: <tag0><tag1><tag2>d1<tag2>d2

@TODO update the file to work with any type of XML data. Use Trees to record the order of the tags.

@Warning This implementation only works for Social network system, needs an update.

Author

eslam

Date

December 2023

Definition in file [ClearClosingTagsComp.h](#).

4.3.2 Macro Definition Documentation

4.3.2.1 CLEAR_CLOSING_TAGS_COMP_H

```
#define CLEAR_CLOSING_TAGS_COMP_H
```

Definition at line 28 of file [ClearClosingTagsComp.h](#).

4.4 ClearClosingTagsComp.h

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

```
00001 /*****
00026 #pragma once
00027 #ifndef CLEAR_CLOSING_TAGS_COMP_H
00028 #define CLEAR_CLOSING_TAGS_COMP_H
00029
00030 #include <string>
00031
00032 class ClearClosingTagsComp
00033 {
00034 private:
00035     const std::string* xmlFile;
00036
00037 public:
00045     explicit ClearClosingTagsComp(const std::string* xmlFile) : xmlFile(xmlFile) {}
00046
00058     std::string* compress();
00059 }; // class ClearClosingTagsComp
00060 #endif // !CLEAR_CLOSING_TAGS_COMP_H
```

4.5 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/↵ Compression/XML/ClearClosingTag/Compression/ClearClosingTagsComp_unittest.cpp File Reference

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

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

4.5.1 Detailed Description

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

Author

eslam

Date

December 2023

Definition in file [ClearClosingTagsComp_unittest.cpp](#).

4.6 ClearClosingTagsComp_unittest.cpp

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

```
00001 /*****
00009 #include "pch.h"
00010 #include "gtest/gtest.h"
00011 #include "ClearClosingTagsComp.h"
00012
00013 namespace {
00014     class ClearClosingTagsCompTest : public::testing::Test {
00015     public:
00016         ClearClosingTagsComp* c;
00017         std::string* input;
00018         std::string* output;
00019     protected:
00020         void SetUp() override {
00021             input = new std::string(R"(<users><user><id>1</id><name>Ahmed
Ali</name><posts><post><body>Lorem ipsum dolor sit ametffsjkn &lt;
</body><topics><topic>economy</topic></posts></post></posts><followers><follower><id>2</id></follower></followers></us
00022
00023             output = new std::string(R"(<users><user><id>1<name>Ahmed Ali<posts><post><body>Lorem
ipsum dolor sit ametffsjkn &lt; <topics><topic>economy<followers><follower><id>2)");
00024
00025             c = new ClearClosingTagsComp(input);
00026         }
00027
00028         void TearDown() override {
00029             delete c;
00030             c = nullptr;
00031             delete input;
00032             input = nullptr;
00033             delete output;
00034             output = nullptr;
00035         }
00036     };
00037
00038     TEST_F(ClearClosingTagsCompTest, compressTest) {
00039         std::string* s = c->compress();
00040         EXPECT_EQ(*s, *output);
00041
00042         delete s;
00043         s = nullptr;
00044     }
00045 } // namespace
```

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

```
#include "pch.h"  
#include "ClearClosingTagsDec.h"
```

4.7.1 Detailed Description

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

- The decompression algorithm is based on returning the removed tags from compression.
- It requires knowing what tags comes after other to know how to decompress the file. Example:
- File before: <tag0><tag1><tag2>d1<tag2>d2
- File after: <tag0><tag1><tag2>d1</tag2><tag2>d2</tag2></tag1></tag0>

@TODO update the file to work with any type of XML data. Use Trees to record the order of the tags.

@Warning This implementation only works for Social network system, needs an update.

Author

eslam

Date

December 2023

Definition in file [ClearClosingTagsDec.cpp](#).

4.8 ClearClosingTagsDec.cpp

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

```
00001 /*****  
00025 #include "pch.h"  
00026 #include "ClearClosingTagsDec.h"  
00027  
00028 std::string ClearClosingTagsDec::getTag(int& tagPosition) const  
00029 {  
00030     char currentChar = this->xmlFile->at(tagPosition);  
00031     std::string tag = std::string(1, currentChar);  
00032     while (currentChar != '>') {  
00033         tagPosition++;  
00034         currentChar = this->xmlFile->at(tagPosition);  
00035         tag += currentChar;  
00036     }  
00037     return tag;  
00038 }
```

```

00039
00040 bool ClearClosingTagsDec::needClosingTag(std::string& tag) const
00041 {
00042     if (tagsStack->empty()) {
00043         tagsStack->push(tagsTree->getRoot());
00044         return false;
00045     }
00046     bool isChild = tagsStack->top()->isChild(&tag.substr(1, tag.length() - 2));
00047     if (isChild) {
00048         tagsStack->push(tagsStack->top()->getChild(&tag.substr(1, tag.length() - 2)));
00049         return false;
00050     }
00051     else {
00052         return true;
00053     }
00054 }
00055
00056 std::string* ClearClosingTagsDec::decompress() const
00057 {
00058     // to store the result.
00059     std::string* result = new std::string();
00060     //length of the original file.
00061     int length = this->xmlFile->size();
00062
00063     // Reserve double the space of the original string.
00064     result->reserve(2*length);
00065
00066     /*
00067     * Loop for all the original string.
00068     * - If the current string is '<' collect that tag then:
00069     *     1.If the stack is empty, add the tag's node into the stack.
00070     *     2.Else check the top tag on the stack
00071     *         a.If the current is a child of the top tag:
00072     *             add the current tag to the stack, and to the result string.
00073     *         b.If it's not pop the op tag and append it to the result
00074     *             as a closing tag.
00075     *             Repeat that till the current tag is a child of the top tag,
00076     *             then do the same as in a.
00077     *     3.At the end of the file add all the remaining tags in the stack as
00078     *         a closing tags in order.
00079     * - For other characters, add them to the result.
00080     */
00081     char currentChar = 0;
00082     for (int i = 0; i < length; i++) {
00083         // get current char
00084         currentChar = this->xmlFile->at(i);
00085         if (currentChar == '<') {
00086             //get the tag
00087             std::string tag = getTag(i);
00088             bool addClosingTag = needClosingTag(tag);
00089             while (addClosingTag) {
00090                 const TreeNode* temp = tagsStack->top();
00091                 //get the closing tag
00092                 std::string value = std::string(temp->getValue());
00093                 std::string closingTag("</");
00094                 closingTag.append(value);
00095                 closingTag.append(">");
00096                 result->append(closingTag);
00097
00098                 tagsStack->pop();
00099
00100                 //check the next top of stack
00101                 addClosingTag = needClosingTag(tag);
00102             }
00103             result->append(tag);
00104         }
00105         else {
00106             result->append(1, currentChar);
00107         }
00108     }
00109
00110     //add the remaining tags in the stack
00111     while (!tagsStack->empty()) {
00112         const TreeNode* temp = tagsStack->top();
00113
00114         // get the closing tag and add it to the result
00115         std::string value = std::string(temp->getValue());
00116         std::string closingTag("</");
00117         closingTag.append(value);
00118         closingTag.append(">");
00119
00120         result->append(closingTag);
00121
00122         tagsStack->pop();
00123     }
00124
00125     // Free the extra allocated memory locations.

```

```
00126     result->shrink_to_fit();  
00127     return result;  
00128 }  
00129  
00130
```

4.9 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/↵ Compression/XML/ClearClosingTag/Decompression/ClearClosing↵ TagsDec.h File Reference

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

```
#include "Tree.h"  
#include <stack>
```

Classes

- class [ClearClosingTagsDec](#)

Macros

- #define [CLEAR_CLOSING_TAGS_DEC_H](#)

4.9.1 Detailed Description

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

- The decompression algorithm is based on returning the removed tags from compression.
- It requires knowing what tags comes after other to know how to decompress the file. Example:
- File before: <tag0><tag1><tag2>d1<tag2>d2
- File after: <tag0><tag1><tag2>d1</tag2><tag2>d2</tag2></tag1></tag0>

@TODO update the file to work with any type of XML data. Use Trees to recored the order of the tags.

@Warning This implementation only works for Social network system, needs an update.

Author

eslam

Date

December 2023

Definition in file [ClearClosingTagsDec.h](#).

4.9.2 Macro Definition Documentation

4.9.2.1 CLEAR_CLOSING_TAGS_DEC_H

#define CLEAR_CLOSING_TAGS_DEC_H

Definition at line 27 of file [ClearClosingTagsDec.h](#).

4.10 ClearClosingTagsDec.h

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

```
00001 /*****
00025 #pragma once
00026 #ifndef CLEAR_CLOSING_TAGS_DEC_H
00027 #define CLEAR_CLOSING_TAGS_DEC_H
00028
00029 #include "Tree.h"
00030 #include <stack>
00031
00032 class ClearClosingTagsDec
00033 {
00034 private:
00035     Tree* tagsTree;
00036     std::stack<const TreeNode*>* tagsStack;
00037     const std::string* xmlFile;
00038
00039     //helper methods
00040
00041     std::string getTag(int& tagPosition) const;
00042     bool needClosingTag(std::string& tag) const;
00043
00044 public:
00051     explicit ClearClosingTagsDec(const std::string* xmlFile) : xmlFile(xmlFile),
00052     tagsTree(new Tree()), tagsStack(new std::stack<const TreeNode*>()) {}
00057     ~ClearClosingTagsDec() { delete tagsTree; delete tagsStack; }
00058
00075     std::string* decompress() const;
00076 };
00077
00078 #endif // !CLEAR_CLOSING_TAGS_DEC_H
```

4.11 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/Workspace/↵ Compression/XML/ClearClosingTag/Decompression/ClearClosingTagsDec_unittest.cpp File Reference

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

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

4.11.1 Detailed Description

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

Author

eslam

Date

December 2023

Definition in file [ClearClosingTagsDec_unittest.cpp](#).

4.12 ClearClosingTagsDec_unittest.cpp

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

```

00001  /*****
00009  #include "pch.h"
00010  #include "gtest/gtest.h"
00011  #include "ClearClosingTagsDec.h"
00012
00013  namespace{
00014      class ClearClosingTagsDecTest : public::testing::Test {
00015      public:
00016          ClearClosingTagsDec* c;
00017          std::string* input;
00018          std::string* output;
00019      protected:
00020          void SetUp() override {
00021              output = new std::string(R"(<users><user><id>1</id><name>Ahmed
Ali</name><posts><post><body>Lorem ipsum dolor sit ametffsjkn &alt;
</body><topics><topic>economy</topic></topics></post></posts><followers><follower><id>2</id></follower></followers></us
00022
00023              input = new std::string(R"(<users><user><id>1<name>Ahmed Ali<posts><post><body>Lorem ipsum
dolor sit ametffsjkn &alt; <topics><topic>economy<followers><follower><id>2)");
00024
00025              c = new ClearClosingTagsDec(input);
00026          }
00027
00028          void TearDown() override {
00029              delete c;
00030              c = nullptr;
00031              delete input;
00032              input = nullptr;
00033              delete output;
00034              output = nullptr;
00035          }
00036      };
00037
00038      TEST_F(ClearClosingTagsDecTest, compressTest) {
00039          std::string* s = c->decompress();
00040          EXPECT_EQ(*s, *output);
00041
00042          delete s;
00043          s = nullptr;
00044      }
00045
00046  } // namespace

```

4.13 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/↵ Compression/XML/ClearClosingTag/Helper/Tree.cpp File Reference

A simple [Tree](#) DS implementation.

```

#include "pch.h"
#include "Tree.h"

```

4.13.1 Detailed Description

A simple [Tree](#) DS implementation.

This tree is for arranging social network system tags. it will be in this order:

- users --children--> {user}
- user --> {id,name,posts,followers}

- posts --> {post}
- post --> {body, topics}
- topics --> {topic}
- followers --> {follower}
- follower --> {id}
- not mentioned: doesn't have a child.

Author

eslam

Date

December 2023

Definition in file [Tree.cpp](#).

4.14 Tree.cpp

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

```
00001 /*****
00021 #include "pch.h"
00022 #include "Tree.h"
00023
00024 void Tree::printTreeNode(const TreeNode* node, int depth) const
00025 {
00026     if (node == nullptr) {
00027         return;
00028     }
00029
00030     for (int i = 0; i < depth; ++i) {
00031         std::cout << " ";
00032     }
00033
00034     std::cout << "|-- " << *node->value << std::endl;
00035
00036     for (const TreeNode* child : *node->children) {
00037         printTreeNode(child, depth + 1);
00038     }
00039 }
00040
00041 Tree::Tree() {
00042     // Creating nodes for the social network system tags
00043     root = new TreeNode(nullptr, new std::vector<TreeNode*>(), new std::string("users"));
00044
00045     // Add child nodes for 'users'
00046     root->children->push_back(new TreeNode(root, new std::vector<TreeNode*>(), new
std::string("user")));
00047
00048     // Add child nodes for 'user'
00049     (*root->children)[0]->children->push_back(new TreeNode((*root->children)[0], new
std::vector<TreeNode*>(), new std::string("id")));
00050     (*root->children)[0]->children->push_back(new TreeNode((*root->children)[0], new
std::vector<TreeNode*>(), new std::string("name")));
00051     (*root->children)[0]->children->push_back(new TreeNode((*root->children)[0], new
std::vector<TreeNode*>(), new std::string("posts")));
00052     (*root->children)[0]->children->push_back(new TreeNode((*root->children)[0], new
std::vector<TreeNode*>(), new std::string("followers")));
00053
00054     // Add child nodes for 'posts'
00055     (*(*root->children)[0]->children)[2]->children->push_back(new
TreeNode((*(*root->children)[0]->children)[2], new std::vector<TreeNode*>(), new
std::string("post")));
00056
00057     // Add child nodes for 'post'
```



```

00058     ((*(*root->children)[0]->children)[2]->children)[0]->children->push_back(new
TreeNode ((*(*root->children)[0]->children)[2]->children)[0], new std::vector<TreeNode*>(), new
std::string("body")));
00059     ((*(*root->children)[0]->children)[2]->children)[0]->children->push_back(new
TreeNode ((*(*root->children)[0]->children)[2]->children)[0], new std::vector<TreeNode*>(), new
std::string("topics")));
00060
00061     // Add child nodes for 'topics'
00062     ((*(*root->children)[0]->children)[2]->children)[0]->children->push_back(new
TreeNode ((*(*root->children)[0]->children)[2]->children)[0]->children)[1], new
std::vector<TreeNode*>(), new std::string("topic")));
00063
00064     // Add child nodes for 'followers'
00065     ((*(*root->children)[0]->children)[3]->children->push_back(new
TreeNode ((*(*root->children)[0]->children)[3], new std::vector<TreeNode*>(), new
std::string("follower")));
00066
00067     // Add child nodes for 'follower'
00068     ((*(*root->children)[0]->children)[3]->children)[0]->children->push_back(new
TreeNode ((*(*root->children)[0]->children)[3]->children)[0], new std::vector<TreeNode*>(), new
std::string("id")));
00069 }
00070
00071

```

4.15 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTag/Helper/Tree.h File Reference

A simple [Tree](#) DS implementation.

```

#include "TreeNode.h"
#include <iostream>

```

Classes

- class [Tree](#)

Macros

- #define [TREE_H](#)

4.15.1 Detailed Description

A simple [Tree](#) DS implementation.

This tree is for arranging social network system tags. it will be in this order:

- users --> {user}
- user --> {id,name,posts,followers}
- posts --> {post}
- post --> {body, topics}
- topics --> {topic}
- followers --> {follower}
- follower --> {id}
- not mentioned: doesn't have a child.

Author

eslam

Date

December 2023

Definition in file [Tree.h](#).**4.15.2 Macro Definition Documentation****4.15.2.1 TREE_H**

#define TREE_H

Definition at line 23 of file [Tree.h](#).**4.16 Tree.h**[Go to the documentation of this file.](#)

```

00001 /*****
00021 #pragma once
00022 #ifndef TREE_H
00023 #define TREE_H
00024
00025 #include "TreeNode.h"
00026 #include <iostream>
00027
00028 class Tree
00029 {
00030 private:
00031     TreeNode* root;
00032     //for debugging
00033     void printTreeNode(const TreeNode* node, int depth) const;
00034 public:
00046     explicit Tree();
00051     ~Tree() {
00052         delete root;
00053     }
00054
00055     //getter
00056     TreeNode* getRoot() { return root; }
00057     //for debugging
00058     void print() const {
00059         printTreeNode(root, 0);
00060     }
00061 };
00062
00063 #endif // !TREE_H
00064
00065
00066

```

4.17 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/↵ Compression/XML/ClearClosingTag/Helper/TreeNode.cpp File Reference

A simple [Tree](#) Node for the tree DS.

```

#include "pch.h"
#include "TreeNode.h"

```

4.17.1 Detailed Description

A simple [Tree](#) Node for the tree DS.

Author

eslam

Date

December 2023

Definition in file [TreeNode.cpp](#).

4.18 TreeNode.cpp

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

```
00001 /*****
00009 #include "pch.h"
00010 #include "TreeNode.h"
00011
00012 const TreeNode* TreeNode::getChild(const std::string* value) const
00013 {
00014     for (TreeNode* child : *children) {
00015         if (*child->value == *value) {
00016             return child;
00017         }
00018     }
00019     return nullptr;
00020 }
```

4.19 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/Workspace/↵ Compression/XML/ClearClosingTag/Helper/TreeNode.h File Reference

A simple [Tree](#) Node for the tree DS.

```
#include <string>
#include <vector>
```

Classes

- class [TreeNode](#)

Macros

- #define [TREE_NODE_H](#)

4.19.1 Detailed Description

A simple [Tree](#) Node for the tree DS.

Author

eslam

Date

December 2023

Definition in file [TreeNode.h](#).

4.19.2 Macro Definition Documentation

4.19.2.1 TREE_NODE_H

```
#define TREE_NODE_H
```

Definition at line 10 of file [TreeNode.h](#).

4.20 TreeNode.h

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

```
00001 /*****/
00008 #pragma once
00009 #ifndef TREE_NODE_H
00010 #define TREE_NODE_H
00011
00012 #include <string>
00013 #include <vector>
00014 class TreeNode
00015 {
00016     friend class Tree;
00017 private:
00018     const TreeNode* parentNode;
00019     std::vector<TreeNode*>* children;
00020     std::string* value;
00021
00022 public:
00023     explicit TreeNode(const TreeNode* parentNode, std::vector<TreeNode*>* children, std::string*
value)
00024         : parentNode(parentNode), children(children) , value(value) {}
00025
00026     ~TreeNode() {
00027         for (TreeNode* child : *children) {
00028             delete child;
00029         }
00030         delete children;
00031         delete value;
00032     }
00033
00034     //methods.
00041     const TreeNode* getChild(const std::string* value) const;
00045     const TreeNode* getParent()const { return this->parentNode; }
00052     bool isChild(const std::string* value) const { return getChild(value) != nullptr; }
00053
00054     //getter
00055     std::string getValue() const { return *value; }
00056
00057 };
00058 #endif // !TREE_NODE_H
00059
00060
00061
```

4.21 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/Minifying/MinifyingXML.cpp File Reference

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

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

4.21.1 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>" → "<tagg>value with spaces</tagg>", on other words, the value will be trimmed from spaces before or after it.

Author

eslam

Date

December 2023

Definition in file [MinifyingXML.cpp](#).

4.22 MinifyingXML.cpp

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

```

00001 /*****
00023 #include "pch.h"
00024 #include "MinifyingXML.h"
00025
00026 // char To skip in minifying.
00027 const char MinifyingXML::charToSkip[5] = { ' ', '\n', '\t', '\v', '\f' };
00028
00029 std::string* MinifyingXML::minifyString()
00030 {
00031     // To store the result.
00032     std::string* result = new std::string();
00033     // Length of the original string
00034     int length = this->xmlFile->size();
00035     //
00036     // The max size of the result string is the same of the entered string.
00037     // That happens when the original doesn't contain any extra spaces or
00038     // other charToSkip elements.
00039     //
00040     result->reserve(length);
00041
00042     this->skipFromBeginning(result);
00043     this->skipFromEnd(result);
00044
00045     // Free the extra allocated memory locations.
00046     result->shrink_to_fit();
00047     //return the result.
00048     return result;
00049 }
00050
00051 // TODO: check whether to add the new line too or not if it was in the body value.
00052 // TODO: check whether to delete comments or not.
00053
00054 void MinifyingXML::skipFromBeginning(std::string* result) const
00055 {
00056     // Length of the original string
00057     int length = this->xmlFile->size();
00058     // Flag for skipping spaces.
00059     bool skipSpaces = true;
00060
00061     /*
00062     * Loop for all values starting form 0.
00063     * That will help removing any charToSkip after tags, but it will
00064     * miss the spaces after values and the next tag (starting or ending).
00065     */
00066     // @TODO: check whether to add the new line too or not if it was in the body value.
00067
00068     // To store the value of the current char on this loop.
00069     char currentChar = 0;
00070     for (int i = 0; i < length; i++) {
00071         //get the current element
00072         currentChar = this->xmlFile->at(i);
00073
00074         //check if it was a skip char
00075         if (MinifyingXML::isSkipChar(currentChar)) {
00076             // @TODO if we should add new spaces to, change the condition here.
00077             // If it was a space and skipSpaces is false, add the space to the result string.
00078             if (currentChar == ' ' && !skipSpaces) {
00079                 result->append(1, currentChar);
00080             }
00081         }
00082         else {
00083             // If not add too the result
00084             result->append(1, currentChar);
00085             // If it was a '>' or '<',
00086             // skip the next spaces.
00087             if (currentChar == '<' || currentChar == '>') {
00088                 skipSpaces = true;
00089             }
00090             // else if it was any char, don't skip after it.
00091             else {
00092                 skipSpaces = false;
00093             }
00094         }
00095     }
00096 }
00097
00098 void MinifyingXML::skipFromEnd(std::string* result) const
00099 {
00100     // Length of the original string
00101     int length = result->size();
00102     // Flag for skipping spaces.
00103     bool skipSpaces = true;

```

```

00104
00105     /*
00106     * Loop for all values starting form the end (length - 1).
00107     */
00108     // To store the value of the current char on this loop.
00109     char currentChar = 0;
00110     for (int i = length - 1; i >= 0; i--) {
00111         //get the current element
00112         currentChar = result->at(i);
00113         //if a skip space delete it.
00114         if (currentChar == ' ' && skipSpaces) {
00115             result->erase(i, 1);
00116         }
00117
00118         //if it is a '<', set skip to true.
00119         else if (currentChar == '<') {
00120             skipSpaces = true;
00121         }
00122
00123         // if any other char, set skip to false.
00124         else {
00125             skipSpaces = false;
00126         }
00127     }
00128 }
00129
00130 bool MinifyingXML::isSkipChar(const char c)
00131 {
00132     for (char ch : MinifyingXML::charToSkip) {
00133         if (c == ch) {
00134             return true;
00135         }
00136     }
00137     return false;
00138 }

```

4.23 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/Minifying/MinifyingXML.h File Reference

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

```

#include <string>
#include <stdexcept>

```

Classes

- class [MinifyingXML](#)

Macros

- #define [MINIFYING_XML_H](#)

4.23.1 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., "<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

Definition in file [MinifyingXML.h](#).

4.23.2 Macro Definition Documentation

4.23.2.1 MINIFYING_XML_H

#define MINIFYING_XML_H

Definition at line 24 of file [MinifyingXML.h](#).

4.24 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
```

4.25 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/Minifying/MinifyingXML_unittest.cpp File Reference

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

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

4.25.1 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

Definition in file [MinifyingXML_unittest.cpp](#).

4.26 MinifyingXML_unittest.cpp

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

```

00001  /*****
00010  #include "gtest/gtest.h"
00011  #include "pch.h"
00012  #include "MinifyingXML.h"
00013
00014  namespace {
00015      class MinifyingXML_Test_essentials : public ::testing::Test {
00016      protected:
00017
00018          MinifyingXML* m;
00019
00020          void SetUp() override {
00021              m = nullptr; // Initialize m to nullptr in SetUp
00022              init_m(new std::string(""));
00023          }
00024
00025          void TearDown() override { clearVar(); }
00026
00027      public:
00028          // methods to help with C'tor tests.
00029          void clearVar() {
00030              delete m; // Safe delete, checks if m is nullptr before deletion
00031              m = nullptr; // Reset m to nullptr after deletion
00032          }
00033          void init_m(const std::string* s) {
00034              clearVar();
00035              m = new MinifyingXML(s);
00036          }
00037      }; // class MinifyingXML_Test_essentials
00038
00039      //Getters and C'tor tests.
00040      TEST_F(MinifyingXML_Test_essentials, ConstructorAndGettersTest) {
00041          EXPECT_EQ(*m->getXMLFile(), "");
00042
00043          // Test handling null pointer in initialization
00044          std::string* s = nullptr;
00045          EXPECT_THROW(init_m(s), std::logic_error);
00046
00047          // Initialize with a valid string and verify the state
00048          s = new std::string("this is a new string.");
00049          init_m(s);
00050          EXPECT_EQ(*m->getXMLFile(), "this is a new string.");
00051          // Clean up memory after testing
00052          delete s;
00053          s = nullptr;
00054      }
00055
00056      //Setters Test
00057      TEST_F(MinifyingXML_Test_essentials, SettersTest) {
00058          std::string* s = nullptr;
00059          EXPECT_THROW(m->setXMLFile(s), std::logic_error);
00060
00061          //empty string
00062          s = new std::string("");
00063          m->setXMLFile(s);
00064          EXPECT_EQ(*m->getXMLFile(), "");
00065
00066          delete s;
00067          s = nullptr;
00068
00069          // any string
00070          s = new std::string("this is a new string.");
00071          m->setXMLFile(s);
00072          EXPECT_EQ(*m->getXMLFile(), "this is a new string.");
00073
00074          delete s;
00075          s = nullptr;
00076      }
00077
00078      //isChar test
00079      TEST_F(MinifyingXML_Test_essentials, isSkipCharTest) {
00080          //true cases.
00081          EXPECT_TRUE(MinifyingXML::isSkipChar(' '));
00082          EXPECT_TRUE(MinifyingXML::isSkipChar('\t'));
00083          EXPECT_TRUE(MinifyingXML::isSkipChar('\v'));
00084          EXPECT_TRUE(MinifyingXML::isSkipChar('\n'));
00085          EXPECT_TRUE(MinifyingXML::isSkipChar('\f'));
00086
00087          //some false cases
00088          EXPECT_FALSE(MinifyingXML::isSkipChar('p'));
00089          EXPECT_FALSE(MinifyingXML::isSkipChar('a'));
00090          EXPECT_FALSE(MinifyingXML::isSkipChar('0'));

```

```

00091     EXPECT_FALSE(MinifyingXML::isSkipChar('3'));
00092     EXPECT_FALSE(MinifyingXML::isSkipChar('8'));
00093 }
00094
00095 class MinifyingXML_Test_Functionality : public ::testing::Test {
00096 protected:
00097     const std::string* input1;
00098     const std::string* expectedResult;
00099     const std::string* afterMinifying;
00100     MinifyingXML* m;
00101     void SetUp() override {
00102         input1 = new std::string(R"(      <users>
00103         <user>
00104             <id>      1      </id>
00105             <name>  Ahmed  Ali  </name>
00106             <posts>
00107                 <post>
00108                     <body>  Lorem ipsum dolor sit ametffsjkn</body>
00109                     <topics>
00110                         <topic>      economy</topic>
00111                     </topics>
00112                 </post>
00113             </posts>
00114             <followers>
00115                 <follower>
00116                     <id>2      </id>
00117                 </follower>
00118             </followers>
00119         </user>
00120     </users>      )");
00121
00122         expectedResult = new std::string(R("<users><user><id>1      </id><name>Ahmed  Ali
00123 </name><posts><post><body>Lorem ipsum dolor sit
00124 ametffsjkn</body><topics><topic>economy</topic></topics></post></posts><followers><follower><id>2
00125 </id></follower></followers></user></users>"));
00126
00127         afterMinifying = new std::string(R("<users><user><id>1</id><name>Ahmed
00128 Ali</name><posts><post><body>Lorem ipsum dolor sit
00129 ametffsjkn</body><topics><topic>economy</topic></topics></post></posts><followers><follower><id>2</id></follower></follo
00130
00131         m = new MinifyingXML(input1);
00132     }
00133
00134     void TearDown() override {
00135         delete input1;
00136         delete expectedResult;
00137         delete afterMinifying;
00138     }
00139
00140     input1 = nullptr;
00141     expectedResult = nullptr;
00142     afterMinifying = nullptr;
00143 }
00144
00145 // class MinifyingXML_Test_Functionality
00146
00147 //helper functions test
00148 TEST_F(MinifyingXML_Test_Functionality, skipFromBeginningTest) {
00149     //action
00150     std::string* output = new std::string();
00151     m->skipFromBeginning(output);
00152
00153     //test
00154     EXPECT_EQ(*output, *expectedResult);
00155
00156     //deallocate
00157     delete output;
00158     output = nullptr;
00159 }
00160
00161 TEST_F(MinifyingXML_Test_Functionality, skipFromEndTest) {
00162     //action
00163     std::string* output = new std::string(*expectedResult);
00164     m->skipFromEnd(output);
00165
00166     //test
00167     EXPECT_EQ(*output, *afterMinifying);
00168
00169     //deallocate
00170     delete output;
00171     output = nullptr;
00172 }
00173
00174 TEST_F(MinifyingXML_Test_Functionality, minifyStringTest) {
00175     //action
00176     const std::string* output = m->minifyString();
00177
00178     //test
00179     EXPECT_EQ(*output, *afterMinifying);

```

```

00173
00174         //deallocate
00175         delete output;
00176         output = nullptr;
00177     }
00178 } // namespace

```

4.27 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/↔ Compression/XML/TagMapping/Compression/TagsMapComp.cpp File Reference

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

```

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

```

4.27.1 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

Definition in file [TagsMapComp.cpp](#).

4.28 TagsMapComp.cpp

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

```

00001 /*****
00031 #include "pch.h"
00032 #include "TagsMapComp.h"
00033
00034 //initialize defaultTagMapBlock
00035 const std::string* TagsMapComp::defaultTagMapBlock = new std::string(
00036     "<TagMap>users,user,id,name,posts,post,body,topics,topic,followers,follower</TagMap>"
00037 );
00038
00039 void TagsMapComp::mapTags()
00040 {
00041     std::stringstream ss(*this->xmlFile);
00042     std::string tag;
00043     std::string line;
00044
00045     while (std::getline(ss, line, '<')) {
00046         // Trim leading spaces
00047         line.erase(0, line.find_first_not_of(" \t\n\r"));
00048         // Extract the tag name between '<' and '>'
00049         int pos = line.find('>');
00050         if (pos == -1) {
00051             continue;
00052         }
00053         int start = 0;
00054         //closing tag
00055         if (line.at(0) == '/') {
00056             start = 1;
00057         }
00058         int length = pos - start;
00059         tag = line.substr(start, length);
00060
00061         // If the tag wasn't in the map, add it
00062         if (!map->containKey(&std::string(tag))) {
00063             map->add(new std::string(tag));
00064         }
00065     }
00066 }
00067
00068 std::string* TagsMapComp::compress(bool addMapTable)
00069 {
00070     //to store the result.
00071     std::string* result = new std::string();
00072     //length of the original file.
00073     int length = this->xmlFile->size();
00074
00075     //
00076     // The max size of the result string is the same of the entered string.
00077     // That happens when the original doesn't contain any extra spaces or
00078     // other charToSkip elements.
00079     // // the added 60 is for the mapTable.
00080     //
00081     result->reserve(length + 60);
00082
00083     //add MapTable if required
00084     if (addMapTable) {
00085         std::string* mapTags = map->toString();
00086         result->append(*mapTags);
00087         //result->append(1, '\n');
00088
00089         delete mapTags;
00090         mapTags = nullptr;
00091     }
00092     else {
00093         // Reinitialize the map to the default Tag map for
00094         // social network system.
00095         delete map;
00096         map = new Map(TagsMapComp::defaultTagMapBlock);
00097     }
00098
00099     /*
00100     * Loop for all the original string.
00101     * - If the current string is '<'
00102     *     1.Collect the tag after it.
00103     *     2.Map that tag.
00104     *     3.Add the mapped tag to the result string.
00105     * - For other characters, add them to the result.
00106     */
00107     char currentChar = 0;
00108     for (int i = 0; i < length; i++) {
00109         // get current char
00110         currentChar = this->xmlFile->at(i);
00111

```

```

00112         //current char is '<' --> map the tag.
00113         if (currentChar == '<') {
00114             // increment the counter to get the next char.
00115             i++;
00116             // get the next char
00117             currentChar = this->xmlFile->at(i);
00118
00119             // to know it is an opening or closing tag.
00120             bool openingTag = true;
00121             if (currentChar == '/') {
00122                 openingTag = false;
00123                 // increment the counter to get the next char.
00124                 i++;
00125                 // get the next char
00126                 currentChar = this->xmlFile->at(i);
00127             }
00128
00129             // To store the tag.
00130             std::string tag = std::string();
00131             //loop to get the full tag
00132             while (currentChar != '>') {
00133                 // append it to the tag string
00134                 tag.append(1, currentChar);
00135                 // increment the counter.
00136                 i++;
00137                 // get current char
00138                 currentChar = this->xmlFile->at(i);
00139             }
00140
00141             //map the tag
00142             std::string afterMapping = std::string("<");
00143             if (!openingTag) {
00144                 afterMapping.append("/");
00145             }
00146             afterMapping.append(1, 't');
00147             afterMapping.append(std::to_string(map->getValue(&tag)));
00148             afterMapping.append(1, '>');
00149
00150             //append to the result.
00151             result->append(afterMapping);
00152         } // if current char == '<'
00153
00154         else {
00155             result->append(1, currentChar);
00156         }
00157     }
00158
00159     // Free the extra allocated memory locations.
00160     result->shrink_to_fit();
00161     return result;
00162 } // compress()

```

4.29 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/↵ Compression/XML/TagMapping/Compression/TagsMapComp.h File Reference

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

```

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

```

Classes

- class [TagsMapComp](#)

Macros

- #define [TAGS_MAP_Comp_H](#)

4.29.1 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

Definition in file [TagsMapComp.h](#).

4.29.2 Macro Definition Documentation

4.29.2.1 TAGS_MAP_Comp_H

```
#define TAGS_MAP_Comp_H
```

Definition at line 33 of file [TagsMapComp.h](#).

4.30 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     const static std::string* defaultTagMapBlock;
00043     //Map of tag values.
00044     Map* map;
00045 public:
00054     explicit TagsMapComp(const std::string* xmlFile) : xmlFile(xmlFile),
00055         map(new Map()) {
00056         mapTags();
00057     }
00062     ~TagsMapComp() { delete map; }
00071     void mapTags();
00072
00090     std::string* compress(bool addMapTable = false);
00091 };
00092
00093 #endif // !TAGS_MAP_Comp_H
```

4.31 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/↵ Compression/XML/TagMapping/Compression/TagsMapComp_↵ unittest.cpp File Reference

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

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

4.31.1 Detailed Description

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

Author

eslam

Date

December 2023

Definition in file [TagsMapComp_unittest.cpp](#).

4.32 TagsMapComp_unittest.cpp

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

```
00001 /*****
00009 #include "gtest/gtest.h"
00010 #include "pch.h"
00011 #include "TagsMapComp.h"
00012
00013 namespace {
00014     class TagsMapCompTest : public::testing::Test {
00015     public:
00016         TagsMapComp* t;
00017         std::string* input;
00018         std::string* result;
00019         std::string* resultWithMap;
00020     protected:
00021         void SetUp() {
00022             input = new std::string(R"(<users>
00023         <user>
00024             <id>          1          </id>
00025             <name> Ahmed Ali </name>
00026             <posts>
00027                 <post>
00028                     <body> Lorem ipsum dolor sit ametffsjkn</body>
00029                     <topics>
00030                         <topic>          economy</topic>
00031                     </topics>
00032                 </post>
00033             </posts>
00034             <followers>
00035                 <follower>
00036                     <id>2          </id>
00037                 </follower>
00038             </followers>
00039         </user>
```



```

00040     </users>         )");
00041
00042     t = new TagsMapComp(input);
00043
00044     result = new std::string(R" (         <t0>
00045     <t1>
00046         <t2>         1         </t2>
00047     <t3>  Ahmed  Ali  </t3>
00048     <t4>
00049         <t5>
00050             <t6>  Lorem ipsum dolor sit ametffsjkn</t6>
00051             <t7>
00052                 <t8>         economy</t8>
00053             </t7>
00054         </t5>
00055     </t4>
00056     <t9>
00057         <t10>
00058             <t2>2             </t2>
00059         </t10>
00060     </t9>
00061 </t1>
00062 </t0>         )");
00063
00064     resultWithMap = new
std::string(R"(<TagMap>users,user,id,name,posts,post,body,topics,topic,followers,follower</TagMap>
<t0>
00065     <t1>
00066         <t2>         1         </t2>
00067     <t3>  Ahmed  Ali  </t3>
00068     <t4>
00069         <t5>
00070             <t6>  Lorem ipsum dolor sit ametffsjkn</t6>
00071             <t7>
00072                 <t8>         economy</t8>
00073             </t7>
00074         </t5>
00075     </t4>
00076     <t9>
00077         <t10>
00078             <t2>2             </t2>
00079         </t10>
00080     </t9>
00081 </t1>
00082 </t0>         )");
00083     }
00084
00085     void TearDown() {
00086         delete t;
00087         delete input;
00088         delete result;
00089         delete resultWithMap;
00090         t = nullptr;
00091         input = nullptr;
00092         result = nullptr;
00093         resultWithMap = nullptr;
00094     }
00095 }; // TagsMapCompTest
00096
00097 TEST_F(TagsMapCompTest, noMap) {
00098     std::string* s = t->compress(false);
00099     EXPECT_EQ(*s, *result);
00100
00101     delete s;
00102     s = nullptr;
00103 }
00104
00105 TEST_F(TagsMapCompTest, withMap) {
00106     std::string* s = t->compress(true);
00107     EXPECT_EQ(*s, *resultWithMap);
00108
00109     delete s;
00110     s = nullptr;
00111 }
00112 }

```

4.33 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/↵ Compression/XML/TagMapping/Decompression/TagsMapDec.cpp File Reference

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

```
#include "pch.h"
#include "TagsMapDec.h"
```

4.33.1 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

Definition in file [TagsMapDec.cpp](#).

4.34 TagsMapDec.cpp

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

```

00001  /*****
00028  #include "pch.h"
00029  #include "TagsMapDec.h"
00030
00031  //initialize defaultTagMapBlock
00032  const std::string* TagsMapDec::defaultTagMapBlock = new std::string(
00033      "<TagMap>users,user,id,name,posts,post,body,topics,topic,followers,follower</TagMap>"
00034  );
00035
00036  void TagsMapDec::getMapTags()
00037  {
00038      const std::string* tagMapLine = this->getTagsMapBlock();
00039      this->map = new Map(tagMapLine);
00040      delete tagMapLine;
00041      tagMapLine = nullptr;
00042  }
00043
00044  const std::string* TagsMapDec::getTagsMapBlock()
00045  {
00046      //Minify the file
00047      MinifyingXML* m = new MinifyingXML(this->xmlFile);
00048      std::string* afterMinifying = m->minifyString();
00049      //deallocate m
00050      delete m;
00051      m = nullptr;
00052
00053      //get the position of both the opening and the closing tags
00054      int start = afterMinifying->find("<TagMap>");
00055      int end = afterMinifying->find("</TagMap>");
00056      //if any was not found, then the file is assumed to be for
00057      //social network system --> return the default line
00058      if (start == std::string::npos && end == std::string::npos) {
00059          return TagsMapDec::defaultTagMapBlock;
00060      }
00061
00062      //if tagMap wasn't in the first position, then the file is defected
00063      if (start != 0) {
00064          throw std::runtime_error("Defected file.");
00065      }
00066
00067      //get the line and return it.
00068      const std::string* result = new std::string(
00069          afterMinifying->substr(start, end + 9 - start)
00070      );
00071      //deallocate after minifying string.
00072      delete afterMinifying;
00073      afterMinifying = nullptr;
00074      return result;
00075  } // getTagsMapBlock()
00076
00077  std::string* TagsMapDec::decompress()
00078  {
00079      //to store the result.
00080      std::string* result = new std::string();
00081      //length of the original file.
00082      int length = this->xmlFile->size();
00083      // The max size of the result string is the same of the entered string.
00084      result->reserve(length);
00085
00086      //skip the TagMap block
00087      int i = this->xmlFile->find("</TagMap>");
00088      // if the block is not found, start from the beginning.
00089      if (i == std::string::npos) {
00090          i = 0;
00091      }
00092      else {
00093          i += 9;
00094      }
00095
00096      /*
00097      * Loop for all the original string.
00098      * - If the current string is '<'
00099      *     1. Collect the tag after it.
00100      *     2. Map that tag.
00101      *     3. Add the mapped tag to the result string.
00102      * - For other characters, add them to the result.
00103      */
00104
00105      char currentChar = 0;
00106      for (i; i < length; i++) {
00107          // get current char
00108          currentChar = this->xmlFile->at(i);

```

```

00109
00110 //current char is '<' --> map the tag.
00111 if (currentChar == '<') {
00112     // increment the counter to get the next char.
00113     i++;
00114     // get the next char
00115     currentChar = this->xmlFile->at(i);
00116
00117     // to know it is an opening or closing tag.
00118     bool openingTag = true;
00119     if (currentChar == '/') {
00120         openingTag = false;
00121         // increment the counter to get the next char.
00122         i++;
00123         // get the next char
00124         currentChar = this->xmlFile->at(i);
00125     }
00126
00127     // To store the tag.
00128     std::string tag = std::string();
00129     //loop to get the full tag
00130     while (currentChar != '>') {
00131         // append it to the tag string
00132         tag.append(1, currentChar);
00133         // increment the counter.
00134         i++;
00135         // get current char
00136         currentChar = this->xmlFile->at(i);
00137     }
00138     //get the number from the tag
00139     tag.erase(0, 1);
00140     int value = std::stoi(tag);
00141
00142     //map the tag
00143     std::string afterMapping = std::string("<");
00144     if (!openingTag) {
00145         afterMapping.append("/");
00146     }
00147
00148     afterMapping.append(*map->getKey(value));
00149     afterMapping.append(1, '>');
00150
00151     //append to the result.
00152     result->append(afterMapping);
00153 } // if current char == '<'
00154
00155 else {
00156     result->append(1, currentChar);
00157 }
00158 }
00159
00160 // Free the extra allocated memory locations.
00161 result->shrink_to_fit();
00162 return result;
00163 } // decompress()

```

4.35 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/↵ Compression/XML/TagMapping/Decompression/TagsMapDec.h File Reference

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

```

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

```

Classes

- class [TagsMapDec](#)

- #define [TAGS_MAP_DEC_H](#)

4.35.1 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

Definition in file [TagsMapDec.h](#).

4.35.2 Macro Definition Documentation

4.35.2.1 TAGS_MAP_DEC_H

```
#define TAGS_MAP_DEC_H
```

Definition at line 30 of file [TagsMapDec.h](#).

4.36 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 #include "MinifyingXML.h"
00034 #include <string>
00035
00036 class TagsMapDec
00037 {
00038 private:
00039     const std::string* xmlFile;
00040
00041     const static std::string* defaultTagMapBlock;
00042
00043     //Map of tag values.
00044     Map* map;
00045
00046     //helper methods
00047
00051     void getMapTags();
00065     const std::string* getTagsMapBlock();
00066 public:
00073     explicit TagsMapDec(const std::string* xmlFile) : xmlFile(xmlFile) {
00074         getMapTags();
00075     }
00080     ~TagsMapDec() {
00081         delete map;
00082         map = nullptr;
00083     }
00090     std::string* decompress();
00091 };
00092 #endif // !TAGS_MAP_DEC_H
```

4.37 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/Workspace/↵ Compression/XML/TagMapping/Decompression/TagsMapDec_↵ unittest.cpp File Reference

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

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

4.37.1 Detailed Description

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

Author

eslam

Date

December 2023

Definition in file [TagsMapDec_unittest.cpp](#).

4.38 TagsMapDec_unittest.cpp

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

```

00001  /*****
00009  #include "gtest/gtest.h"
00010  #include "pch.h"
00011  #include "TagsMapDec.h"
00012
00013  namespace {
00014      class TagsMapDecTest : public::testing::Test {
00015      public:
00016          TagsMapDec* t;
00017          std::string* inputWithMap;
00018          std::string* inputWithOutMap;
00019          std::string* result;
00020      protected:
00021          void SetUp() override {
00022              inputWithMap = new
std::string(R"(<TagMap>users,user,id,name,posts,post,body,topics,topic,followers,follower</TagMap>
<t0>
00023              <t1>
00024                  <t2>          1          </t2>
00025                  <t3> Ahmed  Ali </t3>
00026                  <t4>
00027                      <t5>
00028                          <t6> Lorem ipsum dolor sit ametffsjkn</t6>
00029                          <t7>
00030                              <t8>          economy</t8>
00031                              </t7>
00032                          </t5>
00033                      </t4>
00034                  <t9>
00035                      <t10>
00036                          <t2>2          </t2>
00037                      </t10>
00038                  </t9>
00039              </t1>
00040          </t0>          )");
00041
00042              inputWithOutMap = new std::string(R"(<t0>
00043              <t1>
00044                  <t2>          1          </t2>
00045                  <t3> Ahmed  Ali </t3>
00046                  <t4>
00047                      <t5>
00048                          <t6> Lorem ipsum dolor sit ametffsjkn</t6>
00049                          <t7>
00050                              <t8>          economy</t8>
00051                              </t7>
00052                          </t5>
00053                      </t4>
00054                  <t9>
00055                      <t10>
00056                          <t2>2          </t2>
00057                      </t10>
00058                  </t9>
00059              </t1>
00060          </t0>          )");
00061
00062              result = new std::string(R"(<users>
00063              <user>
00064                  <id>          1          </id>
00065                  <name> Ahmed  Ali </name>
00066                  <posts>
00067                      <post>
00068                          <body> Lorem ipsum dolor sit ametffsjkn</body>
00069                          <topics>
00070                              <topic>          economy</topic>
00071                              </topics>
00072                      </post>
00073                  </posts>
00074                  <followers>
00075                      <follower>
00076                          <id>2          </id>
00077                      </follower>
00078                  </followers>
00079              </user>
00080          </users>          )");
00081          }
00082
00083          void TearDown() {
00084              delete t;
00085              t = nullptr;
00086              delete inputWithMap;
00087              inputWithMap = nullptr;

```

```

00088         delete inputWithOutMap;
00089         inputWithOutMap = nullptr;
00090         delete result;
00091         result = nullptr;
00092     }
00093 };
00094
00095 TEST_F(TagsMapDecTest, withMap) {
00096     t = new TagsMapDec(inputWithMap);
00097     std::string* s = t->decompress();
00098     EXPECT_EQ(*s, *result);
00099
00100     delete s;
00101     s = nullptr;
00102 }
00103
00104 TEST_F(TagsMapDecTest, withoutMap) {
00105     t = new TagsMapDec(inputWithOutMap);
00106     std::string* s = t->decompress();
00107     EXPECT_EQ(*s, *result);
00108
00109     delete s;
00110     s = nullptr;
00111 }
00112 }

```

4.39 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/↔ Compression/XML/TagMapping/Helper/Map.cpp File Reference

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

```

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

```

4.39.1 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

Definition in file [Map.cpp](#).

4.40 Map.cpp

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

```

00001 /*****
00012 #include "pch.h"
00013 #include "Map.h"
00014
00015 void Map::trimString(std::string& str)
00016 {
00017     str.erase(0, str.find_first_not_of(' ')); // Remove leading spaces
00018     str.erase(str.find_last_not_of(' ') + 1); // Remove trailing spaces
00019 }
00020
00021 Map::Map(const std::string* tagMapBlock)
00022 {
00023     this->arr = new std::vector<std::string*>();
00024     //clear the spaces of the file.
00025     MinifyingXML* m = new MinifyingXML(tagMapBlock);
00026     std::string* afterMini = m->minifyString();
00027     delete m;
00028     m = nullptr;
00029
00030     // Get the positions of the opening and closing tags to remove them
00031     int openingTagPos = afterMini->find("<TagMap>");
00032     int closingTagPos = afterMini->find("</TagMap>");
00033
00034     //check that the tag is available.
00035     if (openingTagPos == std::string::npos
00036         || closingTagPos == std::string::npos) {
00037         throw std::runtime_error("Defected TagMap block");
00038     }
00039
00040     //erase the tag
00041     afterMini->erase(openingTagPos, 8); // Erase the opening tag "<TagMap>"
00042     afterMini->erase(afterMini->size() - 9, 9); // Erase the closing tag "</TagMap>"
00043
00044     // add the values between ',' into the arr vector.
00045     std::stringstream ss(*afterMini);
00046     std::string* token = new std::string();
00047
00048     while (std::getline(ss, *token, ',')) {
00049         Map::trimString(*token);
00050         this->add(token);
00051         token = new std::string();
00052     }
00053     delete token;
00054     token = nullptr;
00055
00056     delete afterMini;
00057     afterMini = nullptr;
00058 }
00059
00060 int Map::add(std::string* key)
00061 {
00062     arr->push_back(key);
00063     return arr->size() - 1;
00064 }
00065
00066 int Map::getValue(const std::string* key) const
00067 {
00068     int counter = -1;
00069     for (int i = 0; i < arr->size(); i++) {
00070         std::string* k = arr->at(i);
00071         if (*k == *key) {
00072             return i;
00073         }
00074     }
00075     return -1;
00076 }
00077
00078 const std::string* Map::getKey(int value) const
00079 {
00080     if (arr->size() == 0) {
00081         throw std::runtime_error("array out of bound exception");
00082     }
00083     if (value < 0 || value > arr->size() - 1) {
00084         throw std::runtime_error("array out of bound exception");
00085     }
00086     return arr->at(value);
00087 }
00088
00089 bool Map::containKey(const std::string* key) const {
00090     return (this->getValue(key) == -1) ? false : true;
00091 }
00092

```

```

00093 std::string* Map::toString()
00094 {
00095     if (arr->size() == 0) {
00096         throw std::runtime_error("No value are being mapped");
00097     }
00098     std::string* result = new std::string("<TagMap>");
00099     for (std::string* s : *arr) {
00100         result->append(*s);
00101         result->append(",");
00102     }
00103     result->erase(result->size() - 1);
00104     result->append("</TagMap>");
00105     return result;
00106 }

```

4.41 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/Workspace/↵ Compression/XML/TagMapping/Helper/Map.h File Reference

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

```

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

```

Classes

- class [Map](#)

Macros

- #define [MAP_H](#)

4.41.1 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

Definition in file [Map.h](#).

4.41.2 Macro Definition Documentation

4.41.2.1 MAP_H

```
#define MAP_H
```

Definition at line 14 of file [Map.h](#).

4.42 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
```

4.43 D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/Workspace/↵ Compression/XML/TagMapping/Helper/Map_unittest.cpp File Reference

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

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

4.43.1 Detailed Description

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

Author

eslam

Date

December 2023

Definition in file [Map_unittest.cpp](#).

4.44 Map_unittest.cpp

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

```
00001 /*****
00002  *****/
00003 #include "gtest/gtest.h"
00004 #include "pch.h"
00005 #include "Map.h"
00006
00007 namespace {
00008     class Map_Test : public ::testing::Test {
00009     public:
00010         Map* m;
00011         std::string* s0;
00012         std::string* s1;
00013         std::string* s2;
00014         std::string* s3;
00015
00016     protected:
00017         void SetUp() override {
00018             m = new Map();
00019             s0 = new std::string("v0");
00020             s1 = new std::string("v1");
00021             s2 = new std::string("v2");
00022             s3 = new std::string("v3");
00023         }
00024
00025         void add() {
00026             m->add(s0);
00027             m->add(s1);
00028             m->add(s2);
00029             m->add(s3);
00030         }
00031
00032         void TearDown() override {
00033             delete m;
00034             m = nullptr;
00035         }
00036     };
00037
00038     TEST_F(Map_Test, emptyMap) {
00039         EXPECT_EQ(m->getSize(), 0);
00040         std::string* s = new std::string("any");
00041         EXPECT_EQ(m->getValue(s), -1);
00042
00043         EXPECT_THROW(m->getKey(0), std::runtime_error);
00044         EXPECT_THROW(m->getKey(-1), std::runtime_error);
00045         EXPECT_THROW(m->getKey(5), std::runtime_error);
00046
00047         EXPECT_FALSE(m->containKey(s));
00048
00049         EXPECT_THROW(m->toString(), std::runtime_error);
00050
00051         delete s;
00052         s = nullptr;
00053     }
00054
00055     TEST_F(Map_Test, AddToTheMap) {
00056         add();
00057     }
00058 }
```

```

00062     EXPECT_EQ(m->getSize(), 4);
00063
00064     EXPECT_EQ(m->getValue(s0), 0);
00065     EXPECT_EQ(m->getValue(s1), 1);
00066     EXPECT_EQ(m->getValue(s2), 2);
00067     EXPECT_EQ(m->getValue(s3), 3);
00068
00069     EXPECT_THROW(m->getKey(5), std::runtime_error);
00070
00071     EXPECT_EQ(m->getKey(0), s0);
00072     EXPECT_EQ(m->getKey(1), s1);
00073     EXPECT_EQ(m->getKey(2), s2);
00074     EXPECT_EQ(m->getKey(3), s3);
00075
00076     EXPECT_TRUE(m->containKey(s0));
00077     EXPECT_TRUE(m->containKey(s1));
00078     EXPECT_TRUE(m->containKey(s2));
00079     EXPECT_TRUE(m->containKey(s3));
00080
00081     std::string eOutput = "<TagMap>v0,v1,v2,v3</TagMap>";
00082     std::string* output = m->toString();
00083     EXPECT_EQ(*output, eOutput);
00084     delete output;
00085     output = nullptr;
00086 }
00087
00088 class Map_Test2 : public::testing::Test {
00089 public:
00090     Map* m;
00091     std::string* TagMapBlock;
00092     std::string* output;
00093 protected:
00094     void SetUp() override {
00095         TagMapBlock = new std::string(R"(<TagMap>    v0,v1,
00096 v2,    v3
00097 </TagMap>
00098 )");
00099         output = new std::string(R"(<TagMap>v0,v1,v2,v3</TagMap>)");
00100         m = new Map(TagMapBlock);
00101     }
00102     void TearDown() override {
00103         delete m;
00104         m = nullptr;
00105         delete TagMapBlock;
00106         TagMapBlock = nullptr;
00107         delete output;
00108         output = nullptr;
00109     }
00110 }; // Map_Test2
00111
00112 TEST_F(Map_Test2, MapInitConstrucotr) {
00113     EXPECT_EQ(m->getSize(), 4);
00114     EXPECT_EQ(*m->getKey(0), "v0");
00115     EXPECT_EQ(*m->getKey(1), "v1");
00116     EXPECT_EQ(*m->getKey(2), "v2");
00117     EXPECT_EQ(*m->getKey(3), "v3");
00118     std::string* s = m->toString();
00119     EXPECT_EQ(*s, *output);
00120     delete s;
00121     s = nullptr;
00122 }
00123 } // namespace

```


Index

- ~ClearClosingTagsDec
 - ClearClosingTagsDec, [7](#)
- ~Map
 - Map, [9](#)
- ~TagsMapComp
 - TagsMapComp, [15](#)
- ~TagsMapDec
 - TagsMapDec, [17](#)
- ~Tree
 - Tree, [18](#)
- ~TreeNode
 - TreeNode, [20](#)
- add
 - Map, [9](#)
- CLEAR_CLOSING_TAGS_COMP_H
 - ClearClosingTagsComp.h, [25](#)
- CLEAR_CLOSING_TAGS_DEC_H
 - ClearClosingTagsDec.h, [30](#)
- ClearClosingTagsComp, [5](#)
 - ClearClosingTagsComp, [5](#)
 - compress, [6](#)
- ClearClosingTagsComp.h
 - CLEAR_CLOSING_TAGS_COMP_H, [25](#)
- ClearClosingTagsDec, [6](#)
 - ~ClearClosingTagsDec, [7](#)
 - ClearClosingTagsDec, [6](#)
 - decompress, [7](#)
- ClearClosingTagsDec.h
 - CLEAR_CLOSING_TAGS_DEC_H, [30](#)
- compress
 - ClearClosingTagsComp, [6](#)
 - TagsMapComp, [15](#)
- containKey
 - Map, [9](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTagsComp.cpp, [23](#), [24](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTagsDec.h, [24](#), [25](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTagsDec_unittest.cpp, [26](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTagsDec, [27](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTagsComp, [29](#), [30](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTagsComp, [30](#), [31](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTagsComp, [31](#), [32](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTagsComp, [33](#), [34](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTagsComp, [34](#), [35](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/ClearClosingTagsComp, [35](#), [36](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/Minifying/MinifyTagsComp.cpp, [37](#), [38](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/Minifying/MinifyTagsComp, [39](#), [40](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/Minifying/MinifyTagsComp, [41](#), [42](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/TagMapping/TagMappingComp.h, [44](#), [45](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full project repo/CSE331-Data-Structure-and-Algorithms-Project/WorkSpace/Compression/XML/TagMapping/TagMappingComp_unittest.cpp, [46](#), [47](#)
- D:/Engineering/Senior 1/Fall/Data structure/Project/Full

- [TREE_H](#), [34](#)
- [TREE_H](#)
 - [Tree.h](#), [34](#)
- [TREE_NODE_H](#)
 - [TreeNode.h](#), [36](#)
- [TreeNode](#), [19](#)
 - [~TreeNode](#), [20](#)
 - [getChild](#), [20](#)
 - [getParent](#), [20](#)
 - [getValue](#), [21](#)
 - [isChild](#), [21](#)
 - [Tree](#), [21](#)
 - [TreeNode](#), [20](#)
- [TreeNode.h](#)
 - [TREE_NODE_H](#), [36](#)