Compression task

Generated by Doxygen 1.9.8

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

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

the	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

the	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

tagMapBlock.	

Exceptions

runtime_error	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

key	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

key.	

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

value	

Returns

The key.

Exceptions

runtime	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

key.	

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

runtime	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

xmlFile	

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

c	-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 tab, 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::skipFrom↩ End(const std::string∗ result).

Parameters

result	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::skipFrom←Beginning(std::string∗ result).

Parameters

result	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

the	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

addMapTable	if true, then a <TagMap> 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

the	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 –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.

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

value	

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

value	

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 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 ClearClosingTagsComp.cpp.

4.2 ClearClosingTagsComp.cpp

Go to the documentation of this file.

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

std::string* ClearClosingTagsComp::compress()
{
    // to store the result.
    std::string* result = new std::string();
    //length of the original file.
    int length = this->xmlFile->size();


    // The max size of the result string is the same of the entered string.
    result->reserve(length);

    /*
    * Loop for all the original string.
    * - If the current string is '</'
    *       1.skip that tag (increment i till the end of the tag).
    *       2.Don't add it to the result string.
    * - For other characters, add them to the result.
    */
    char currentChar = 0;
    for (int i = 0; i < length; i++) {
        // get current char
        currentChar = this->xmlFile->at(i);
        if (currentChar == '<' && this->xmlFile->at(i + 1) == '/') {
            //skip the closing tag
            i = this->xmlFile->find('>', i);
            continue;
        }
        result->append(1, currentChar);
    }

    // Free the extra allocated memory locations.
    result->shrink_to_fit();
    return result;
}// 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 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 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/ClearClosing↩ TagsComp_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.

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

namespace {
    class ClearClosingTagsCompTest : public::testing::Test {
    public:
        ClearClosingTagsComp* c;
        std::string* input;
        std::string* output;
    protected:
        void SetUp() override {
            input = 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></use

            output = 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)");

            c = new ClearClosingTagsComp(input);
        }

        void TearDown() override {
            delete c;
            c = nullptr;
            delete input;
            input = nullptr;
            delete output;
            output = nullptr;
        }
    };

    TEST_F(ClearClosingTagsCompTest, compressTest) {
        std::string* s = c->compress();
        EXPECT_EQ(*s, *output);

        delete s;
        s = nullptr;
    }
}// namespace
```

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

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 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.cpp.

4.8 ClearClosingTagsDec.cpp

Go to the documentation of this file.

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

std::string ClearClosingTagsDec::getTag(int& tagPosition) const
{
    char currentChar = this->xmlFile->at(tagPosition);
    std::string tag = std::string(1, currentChar);
    while (currentChar != '>') {
        tagPosition++;
        currentChar = this->xmlFile->at(tagPosition);
        tag += currentChar;
    }
    return tag;
}
```

```

bool ClearClosingTagsDec::needClosingTag(std::string& tag) const
{
    if (tagsStack->empty()) {
        tagsStack->push(tagsTree->getRoot());
        return false;
    }
    bool isChild = tagsStack->top()->isChild(&tag.substr(1, tag.length() - 2));
    if (isChild) {
        tagsStack->push(tagsStack->top()->getChild(&tag.substr(1, tag.length() - 2)));
        return false;
    }
    else {
        return true;
    }
}

std::string* ClearClosingTagsDec::decompress() const
{
    // to store the result.
    std::string* result = new std::string();
    //length of the original file.
    int length = this->xmlFile->size();

    // Reserve double the space of the original string.
    result->reserve(2*length);

    /*
    * Loop for all the original string.
    * - If the current string is '<' collect that tag then:
    *       1.If the stack is empty, add the tag's node into the stack.
    *       2.else check the top tag on the stack
    *           a.If the current is a child of the top tag:
    *             add the current tag to the stack, and to the result string.
    *           b.if it's not pop the op tag and append it to the result
    *             as a closing tag.
    *             Repeat that till the current tag is a child of the top tag,
    *             then do the same as in a.
    *       3.At the end of the file add all the remaining tags in the stack as
    *         a closing tags in order.
    * - For other characters, add them to the result.
    */
    char currentChar = 0;
    for (int i = 0; i < length; i++) {
        // get current char
        currentChar = this->xmlFile->at(i);
        if (currentChar == '<') {
            //get the tag
            std::string tag = getTag(i);
            bool addClosingTag = needClosingTag(tag);
            while (addClosingTag) {
                const TreeNode* temp = tagsStack->top();
                //get the closing tag
                std::string value = std::string(temp->getValue());
                std::string closingTag("</");
                closingTag.append(value);
                closingTag.append(">");
                result->append(closingTag);

                tagsStack->pop();

                //check the next top of stack
                addClosingTag = needClosingTag(tag);
            }
            result->append(tag);
        }
        else {
            result->append(1, currentChar);
        }
    }

    //add the remaining tags in the stack
    while (!tagsStack->empty()) {
        const TreeNode* temp = tagsStack->top();

        // get the closing tag and add it to the result
        std::string value = std::string(temp->getValue());
        std::string closingTag("</");
        closingTag.append(value);
        closingTag.append(">");

        result->append(closingTag);

        tagsStack->pop();
    }

    // Free the extra allocated memory locations.
    result->shrink_to_fit();
    return result;
}


```

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.

```
/*******************************************************************/
#pragma once
#ifndef CLEAR_CLOSING_TAGS_DEC_H
#define CLEAR_CLOSING_TAGS_DEC_H

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

class ClearClosingTagsDec
{
private:
    Tree* tagsTree;
    std::stack<const TreeNode*>* tagsStack;
    const std::string* xmlFile;

    //helper methods

    std::string getTag(int& tagPosition) const;
    bool needClosingTag(std::string& tag) const;

public:
    explicit ClearClosingTagsDec(const std::string* xmlFile) : xmlFile(xmlFile),
    tagsTree(new Tree()), tagsStack(new std::stack<const TreeNode*>()) {}
    ~ClearClosingTagsDec() { delete tagsTree; delete tagsStack; }

    std::string* decompress() const;
};

#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/ClearClosing↩ TagsDec_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.

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

namespace{
    class ClearClosingTagsDecTest : public::testing::Test {
    public:
        ClearClosingTagsDec* c;
        std::string* input;
        std::string* output;
    protected:
        void SetUp() override {
            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></use

            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)");

            c = new ClearClosingTagsDec(input);
        }

        void TearDown() override {
            delete c;
            c = nullptr;
            delete input;
            input = nullptr;
            delete output;
            output = nullptr;
        }
    };

    TEST_F(ClearClosingTagsDecTest, compressTest) {
        std::string* s = c->decompress();
        EXPECT_EQ(*s, *output);

        delete s;
        s = nullptr;
    }

} // 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.

```
/*******************************************************************/
#include "pch.h"
#include "Tree.h"

void Tree::printTreeNode(const TreeNode* node, int depth) const
{
    if (node == nullptr) {
        return;
    }

    for (int i = 0; i < depth; ++i) {
        std::cout « "  ";
    }

    std::cout « "|-- " « *node->value « std::endl;

    for (const TreeNode* child : *node->children) {
        printTreeNode(child, depth + 1);
    }
}

Tree::Tree() {
    // Creating nodes for the social network system tags
    root = new TreeNode(nullptr, new std::vector<TreeNode*>(), new std::string("users"));

    // Add child nodes for 'users'
    root->children->push_back(new TreeNode(root, new std::vector<TreeNode*>(), new
std::string("user")));

    // Add child nodes for 'user'
    (*root->children)[0]->children->push_back(new TreeNode((*root->children)[0], new
std::vector<TreeNode*>(), new std::string("id")));
    (*root->children)[0]->children->push_back(new TreeNode((*root->children)[0], new
std::vector<TreeNode*>(), new std::string("name")));
    (*root->children)[0]->children->push_back(new TreeNode((*root->children)[0], new
std::vector<TreeNode*>(), new std::string("posts")));
    (*root->children)[0]->children->push_back(new TreeNode((*root->children)[0], new
std::vector<TreeNode*>(), new std::string("followers")));

    // Add child nodes for 'posts'
    (*(*root->children)[0]->children)[2]->children->push_back(new
TreeNode((*(*root->children)[0]->children)[2], new std::vector<TreeNode*>(), new
std::string("post")));

    // Add child nodes for 'post'
    (*(*(*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")));
    (*(*(*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")));

    // Add child nodes for 'topics'
    (*(*(*(*root->children)[0]->children)[2]->children)[0]->children)[1]->children->push_back(new
TreeNode((*(*(*(*root->children)[0]->children)[2]->children)[0]->children)[1], new
std::vector<TreeNode*>(), new std::string("topic")));

    // Add child nodes for 'followers'
    (*(*root->children)[0]->children)[3]->children->push_back(new
TreeNode((*(*root->children)[0]->children)[3], new std::vector<TreeNode*>(), new
std::string("follower")));

    // Add child nodes for 'follower'
    (*(*(*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")));
}


```

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 –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.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.

```
/*******************************************************************/
#include "pch.h"
#include "TreeNode.h"

const TreeNode* TreeNode::getChild(const std::string* value) const
{
    for (TreeNode* child : *children) {
        if (*child->value == *value) {
            return child;
        }
    }
    return nullptr;
}
```

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>" –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.cpp.

4.22 MinifyingXML.cpp

Go to the documentation of this file.

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

 // char To skip in minifying.
const char MinifyingXML::charToSkip[5] = { ' ', '\n', '\t','\v','\f' };

std::string* MinifyingXML::minifyString()
{
    // To store the result.
    std::string* result = new std::string();
    // Length of the original string
    int length = this->xmlFile->size();
    //
    // The max size of the result string is the same of the entered string.
    // That happens when the original doesn't contain any extra spaces or
    // other charToSkip elements.
    //
    result->reserve(length);

    this->skipFromBeginning(result);
    this->skipFromEnd(result);

    // Free the extra allocated memory locations.
    result->shrink_to_fit();
    //return the result.
    return result;
}

// TODO: check whether to add the new line too or not if it was in the body value.
// TODO: check whether to delete comments or not.

void MinifyingXML::skipFromBeginning(std::string* result) const
{
    // Length of the original string
    int length = this->xmlFile->size();
    // Flag for skipping spaces.
    bool skipSpaces = true;

    /*
    * Loop for all values starting form 0.
    * That will help removing any charToSkip after tags, but it will
    * miss the spaces after values and the next tag (starting or ending).
    */
    // @TODO: check whether to add the new line too or not if it was in the body value.

    // To store the value of the current char on this loop.
    char currentChar = 0;
    for (int i = 0; i < length; i++) {
        //get the current element
        currentChar = this->xmlFile->at(i);

        //check if it was a skip char
        if (MinifyingXML::isSkipChar(currentChar)) {
            // @TODO if we should  add new spaces to, change the condition here.
            // If it was a space and skipSpaces is false, add the space to the result string.
            if (currentChar == ' ' && !skipSpaces) {
                result->append(1, currentChar);
            }
        }
        else {
            // If not add too the result
            result->append(1, currentChar);
            // If it was a '>' or  '<',
            // skip the next spaces.
            if (currentChar == '<' || currentChar == '>') {
                skipSpaces = true;
            }
            // else if it was any char, don't skip after it.
            else {
                skipSpaces = false;
            }
        }
    }
}

void MinifyingXML::skipFromEnd(std::string* result) const
{
    // Length of the original string
    int length = result->size();
    // Flag for skipping spaces.
    bool skipSpaces = true;

    /*
    * Loop for all values starting form the end (length - 1).
    */
    // To store the value of the current char on this loop.
    char currentChar = 0;
    for (int i = length - 1; i >= 0; i--) {
        //get the current element
        currentChar = result->at(i);
        //if a skip space delete it.
        if (currentChar == ' ' && skipSpaces) {
            result->erase(i, 1);
        }

        //if it is a '<', set skip to true.
        else if (currentChar == '<') {
            skipSpaces = true;
        }

        // if any other char, set skip to false.
        else {
            skipSpaces = false;
        }
    }
}

bool MinifyingXML::isSkipChar(const char c)
{
    for (char ch : MinifyingXML::charToSkip) {
        if (c == ch) {
            return true;
        }
    }
    return false;
}
```

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 neads 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 
```

```
    void skipFromEnd(std::string* result) const;

    //getters and setters, used for debugging
    //getters.

    //XML file getter.
    const std::string* getXMLFile() const { return this->xmlFile; }

    //setters
    //XML file setter.
    void setXMLFile(const std::string* xmlFileNew) {
        // check adding a null ptr.
        if (xmlFileNew == nullptr) {
            throw std::logic_error("Null pointer exception: Accessing null pointer!");
        }
        this->xmlFile = xmlFileNew;
    }
}; //class MinifyingXML

#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.

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

namespace {
    class MinifyingXML_Test_essintials : public ::testing::Test {
    protected:

        MinifyingXML* m;

        void SetUp() override {
            m = nullptr;  // Initialize m to nullptr in SetUp
            init_m(new std::string(""));
        }

        void TearDown()override { clearVar(); }

    public:
        // methods to help with C'tor tests.
        void clearVar() {
            delete m;  // Safe delete, checks if m is nullptr before deletion
            m = nullptr;  // Reset m to nullptr after deletion
        }
        void init_m(const std::string* s) {
            clearVar();
            m = new MinifyingXML(s);
        }
    }; // class MinifyingXML_Test_essintials

    //Getters and C'tor tests.
    TEST_F(MinifyingXML_Test_essintials, ConstructorAndGettersTest) {
        EXPECT_EQ(*m->getXMLFile(), "");

        // Test handling null pointer in initialization
        std::string* s = nullptr;
        EXPECT_THROW(init_m(s), std::logic_error);

        // Initialize with a valid string and verify the state
        s = new std::string("this is a new string.");
        init_m(s);
        EXPECT_EQ(*m->getXMLFile(), "this is a new string.");
        // Clean up memory after testing
        delete s;
        s = nullptr;
    }

    //Setters Test
    TEST_F(MinifyingXML_Test_essintials, SettersTest) {
        std::string* s = nullptr;
        EXPECT_THROW(m->setXMLFile(s), std::logic_error);

        //empty string
        s = new std::string("");
        m->setXMLFile(s);
        EXPECT_EQ(*m->getXMLFile(), "");

        delete s;
        s = nullptr;

        // any string
        s = new std::string("this is a new string.");
        m->setXMLFile(s);
        EXPECT_EQ(*m->getXMLFile(), "this is a new string.");

        delete s;
        s = nullptr;
    }

    //isChar test
    TEST_F(MinifyingXML_Test_essintials, isSkipCharTest) {
        //true cases.
        EXPECT_TRUE(MinifyingXML::isSkipChar(' '));
        EXPECT_TRUE(MinifyingXML::isSkipChar('\t'));
        EXPECT_TRUE(MinifyingXML::isSkipChar('\v'));
        EXPECT_TRUE(MinifyingXML::isSkipChar('\n'));
        EXPECT_TRUE(MinifyingXML::isSkipChar('\f'));

        //some false cases
        EXPECT_FALSE(MinifyingXML::isSkipChar('p'));
        EXPECT_FALSE(MinifyingXML::isSkipChar('a'));
        EXPECT_FALSE(MinifyingXML::isSkipChar('0'));
        EXPECT_FALSE(MinifyingXML::isSkipChar('3'));
        EXPECT_FALSE(MinifyingXML::isSkipChar('8'));
    }

    class MinifyingXML_Test_Functionality : public ::testing::Test {
    protected:
        const std::string* input1;
        const std::string* expectedResult;
        const std::string* afterMinifying;
        MinifyingXML* m;
        void SetUp() override {
            input1 = new std::string(R"(      <users>
        <user>
                  <id>        1       </id>
            <name>  Ahmed  Ali  </name>
            <posts>
                <post>
                    <body>  Lorem ipsum dolor sit ametffsjkn</body>
                    <topics>
                        <topic>     economy</topic>
                    </topics>
                </post>
            </posts>
            <followers>
                <follower>
                    <id>2           </id>
                </follower>
            </followers>
        </user>
    </users>     )");

            expectedResult = new std::string(R"(<users><user><id>1       </id><name>Ahmed  Ali
</name><posts><post><body>Lorem ipsum dolor sit
ametffsjkn</body><topics><topic>economy</topic></topics></post></posts><followers><follower><id>2
</id></follower></followers></user></users>)");

            afterMinifying = new std::string(R"(<users><user><id>1</id><name>Ahmed
Ali</name><posts><post><body>Lorem ipsum dolor sit
ametffsjkn</body><topics><topic>economy</topic></topics></post></posts><followers><follower><id>2</id></follower></follo

            m = new MinifyingXML(input1);
        }

        void TearDown() override {
            delete input1;
            delete expectedResult;
            delete afterMinifying;

            input1 = nullptr;
            expectedResult = nullptr;
            afterMinifying = nullptr;
        }
    }; // class MinifyingXML_Test_Functionality

    //helper functions test
    TEST_F(MinifyingXML_Test_Functionality, skipFromBeginningTest) {
        //action
        std::string* output = new std::string();
        m->skipFromBeginning(output);

        //test
        EXPECT_EQ(*output, *expectedResult);

        //deallocate
        delete output;
        output = nullptr;
    }

    TEST_F(MinifyingXML_Test_Functionality, skipFromEndTest) {
        //action
        std::string* output = new std::string(*expectedResult);
        m->skipFromEnd(output);

        //test
        EXPECT_EQ(*output, *afterMinifying);

        //deallocate
        delete output;
        output = nullptr;
    }

    TEST_F(MinifyingXML_Test_Functionality, minifyStringTest) {
        //action
        const std::string* output = m->minifyString();

        //test
        EXPECT_EQ(*output, *afterMinifying);

        //deallocate
        delete output;
        output = nullptr;
    }
} // 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.

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

 //initialize defaultTagMapBlock
const std::string* TagsMapComp::defualtTagMapBlock = new std::string(
    "<TagMap>users,user,id,name,posts,post,body,topics,topic,followers,follower</TagMap>"
);

void TagsMapComp::mapTags()
{
    std::stringstream ss(*this->xmlFile);
    std::string tag;
    std::string line;

    while (std::getline(ss, line, '<')) {
        // Trim leading spaces
        line.erase(0, line.find_first_not_of(" \t\n\r"));
        // Extract the tag name between '<' and '>'
        int pos = line.find('>');
        if (pos == -1) {
            continue;
        }
        int start = 0;
        //closing tag
        if (line.at(0) == '/') {
            start = 1;
        }
        int length = pos - start;
        tag = line.substr(start, length);

        // If the tag wasn't in the map, add it
        if (!map->containKey(&std::string(tag))) {
            map->add(new std::string(tag));
        }
    }
}

std::string* TagsMapComp::compress(bool addMapTable)
{
    //to store the result.
    std::string* result = new std::string();
    //length of the original file.
    int length = this->xmlFile->size();

    //
    // The max size of the result string is the same of the entered string.
    // That happens when the original doesn't contain any extra spaces or
    // other charToSkip elements.
    // // the added 60 is for the mapTable.
    //
    result->reserve(length + 60);

    //add MapTable if required
    if (addMapTable) {
        std::string* mapTags = map->toString();
        result->append(*mapTags);
        //result->append(1, '\n');

        delete mapTags;
        mapTags = nullptr;
    }
    else {
        //  Reinitialize the map to the default Tag map for
        //  social network system.
        delete map;
        map = new Map(TagsMapComp::defualtTagMapBlock);
    }

    /*
    * Loop for all the original string.
    * - If the current string is '<'
    *       1.Collect the tag after it.
    *       2.Map that tag.
    *       3.Add the mapped tag to the result string.
    * - For other characters, add them to the result.
    */
    char currentChar = 0;
    for (int i = 0; i < length; i++) {
        // get current char
        currentChar = this->xmlFile->at(i);

        //current char is '<' --> map the tag.
        if (currentChar == '<') {
            // increment the counter to get the next char.
            i++;
            // get the next char
            currentChar = this->xmlFile->at(i);

            // to know it is an opening or closing tag.
            bool openingTag = true;
            if (currentChar == '/') {
                openingTag = false;
                // increment the counter to get the next char.
                i++;
                // get the next char
                currentChar = this->xmlFile->at(i);
            }

            // To store the tag.
            std::string tag = std::string();
            //loop to get the full tag
            while (currentChar != '>') {
                // append it to the tag string
                tag.append(1, currentChar);
                // increment the counter.
                i++;
                // get current char
                currentChar = this->xmlFile->at(i);
            }

            //map the tag
            std::string afterMaping = std::string("<");
            if (!openingTag) {
                afterMaping.append("/");
            }
            afterMaping.append(1, 't');
            afterMaping.append(std::to_string(map->getValue(&tag)));
            afterMaping.append(1, '>');

            //append to the result.
            result->append(afterMaping);
        } // if current char == '<'

        else {
            result->append(1, currentChar);
        }
    }

    // Free the extra allocated memory locations.
    result->shrink_to_fit();
    return result;
}// 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* defualtTagMapBlock;
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.

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

namespace {
    class TagsMapCompTest : public::testing::Test {
    public:
        TagsMapComp* t;
        std::string* input;
        std::string* result;
        std::string* resultWithMap;
    protected:
        void SetUp() {
            input = new std::string(R"(      <users>
        <user>
                  <id>        1       </id>
            <name>  Ahmed  Ali  </name>
            <posts>
                <post>
                    <body>  Lorem ipsum dolor sit ametffsjkn</body>
                    <topics>
                        <topic>     economy</topic>
                    </topics>
                </post>
            </posts>
            <followers>
                <follower>
                    <id>2           </id>
                </follower>
            </followers>
        </user>
    </users>     )");

            t = new TagsMapComp(input);

            result = new std::string(R"(      <t0>
        <t1>
                  <t2>        1       </t2>
            <t3>  Ahmed  Ali  </t3>
            <t4>
                <t5>
                    <t6>  Lorem ipsum dolor sit ametffsjkn</t6>
                    <t7>
                        <t8>     economy</t8>
                    </t7>
                </t5>
            </t4>
            <t9>
                <t10>
                    <t2>2           </t2>
                </t10>
            </t9>
        </t1>
    </t0>     )");

            resultWithMap = new std::string(R"(<TagMap>users,user,id,name,posts,post,body,topics,topic,followers,follower</TagMap> <t0>
        <t1>
                  <t2>        1       </t2>
            <t3>  Ahmed  Ali  </t3>
            <t4>
                <t5>
                    <t6>  Lorem ipsum dolor sit ametffsjkn</t6>
                    <t7>
                        <t8>     economy</t8>
                    </t7>
                </t5>
            </t4>
            <t9>
                <t10>
                    <t2>2           </t2>
                </t10>
            </t9>
        </t1>
    </t0>     )");
        }

        void TearDown() {
            delete t;
            delete input;
            delete result;
            delete resultWithMap;
            t = nullptr;
            input = nullptr;
            result = nullptr;
            resultWithMap = nullptr;
        }
    }; // TagsMapCompTest

    TEST_F(TagsMapCompTest, noMap) {
        std::string* s = t->compress(false);
        EXPECT_EQ(*s, *result);

        delete s;
        s = nullptr;
    }

    TEST_F(TagsMapCompTest, withMap) {
        std::string* s = t->compress(true);
        EXPECT_EQ(*s, *resultWithMap);

        delete s;
        s = nullptr;
    }
}
```

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.

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

 //initialize defaultTagMapBlock
const std::string* TagsMapDec::defualtTagMapBlock = new std::string(
    "<TagMap>users,user,id,name,posts,post,body,topics,topic,followers,follower</TagMap>"
);

void TagsMapDec::getMapTags()
{
    const std::string* tagMapLine = this->getTagsMapBlock();
    this->map = new Map(tagMapLine);
    delete tagMapLine;
    tagMapLine = nullptr;
}

const std::string* TagsMapDec::getTagsMapBlock()
{
    //Minify the file
    MinifyingXML* m = new MinifyingXML(this->xmlFile);
    std::string* afterMinifying = m->minifyString();
    //deallocate m
    delete m;
    m = nullptr;

    //get the position of both the opening and the closing tags
    int start = afterMinifying->find("<TagMap>");
    int end = afterMinifying->find("</TagMap>");
    //if any was not found, then the file is assumed to be for
    //social network system --> return the default line
    if (start == std::string::npos && end == std::string::npos) {
        return TagsMapDec::defualtTagMapBlock;
    }

    //if tagMap wasn't in the first position, then the file is defected
    if (start != 0) {
        throw std::runtime_error("Defected file.");
    }

    //get the line and return it.
    const std::string* result = new std::string(
        afterMinifying->substr(start, end + 9 - start)
    );
    //deallocate after minifying string.
    delete afterMinifying;
    afterMinifying = nullptr;
    return result;
}// getTagsMapBlock()

std::string* TagsMapDec::decompress()
{
    //to store the result.
    std::string* result = new std::string();
    //length of the original file.
    int length = this->xmlFile->size();
    // The max size of the result string is the same of the entered string.
    result->reserve(length);

    //skip the TagMap block
    int i = this->xmlFile->find("</TagMap>");
    // if the block is not found, start from the beginning.
    if (i == std::string::npos) {
        i = 0;
    }
    else {
        i += 9;
    }

    /*
    * Loop for all the original string.
    * - If the current string is '<'
    *       1.Collect the tag after it.
    *       2.Map that tag.
    *       3.Add the mapped tag to the result string.
    * - For other characters, add them to the result.
    */

    char currentChar = 0;
    for (i; i < length; i++) {
        // get current char
        currentChar = this->xmlFile->at(i);

        //current char is '<' --> map the tag.
        if (currentChar == '<') {
            // increment the counter to get the next char.
            i++;
            // get the next char
            currentChar = this->xmlFile->at(i);

            // to know it is an opening or closing tag.
            bool openingTag = true;
            if (currentChar == '/') {
                openingTag = false;
                // increment the counter to get the next char.
                i++;
                // get the next char
                currentChar = this->xmlFile->at(i);
            }

            // To store the tag.
            std::string tag = std::string();
            //loop to get the full tag
            while (currentChar != '>') {
                // append it to the tag string
                tag.append(1, currentChar);
                // increment the counter.
                i++;
                // get current char
                currentChar = this->xmlFile->at(i);
            }
            //get the number from the tag
            tag.erase(0, 1);
            int value = std::stoi(tag);

            //map the tag
            std::string afterMaping = std::string("<");
            if (!openingTag) {
                afterMaping.append("/");
            }

            afterMaping.append(*map->getKey(value));
            afterMaping.append(1, '>');

            //append to the result.
            result->append(afterMaping);
        } // if current char == '<'

        else {
            result->append(1, currentChar);
        }
    }

    // Free the extra allocated memory locations.
    result->shrink_to_fit();
    return result;
}// 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

Macros

- #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.

```
/*********************************************************************/
#pragma once
#ifndef TAGS_MAP_DEC_H
#define TAGS_MAP_DEC_H

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

class TagsMapDec
{
private:
    const std::string* xmlFile;

    const static std::string* defualtTagMapBlock;

    //Map of tag values.
    Map* map;

    //helper methods

    void getMapTags();
    const std::string* getTagsMapBlock();
public:
    explicit TagsMapDec(const std::string* xmlFile) : xmlFile(xmlFile) {
        getMapTags();
    }
    ~TagsMapDec() {
        delete map;
        map = nullptr;
    }
    std::string* decompress();
};
#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.

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

namespace {
    class TagsMapDecTest : public::testing::Test {
    public:
        TagsMapDec* t;
        std::string* inputWithMap;
        std::string* inputWithOutMap;
        std::string* result;
    protected:
        void SetUp() override {
            inputWithMap = new
std::string(R"(<TagMap>users,user,id,name,posts,post,body,topics,topic,followers,follower</TagMap>
<t0>
        <t1>
                  <t2>        1       </t2>
            <t3>  Ahmed  Ali  </t3>
            <t4>
                <t5>
                    <t6>  Lorem ipsum dolor sit ametffsjkn</t6>
                    <t7>
                        <t8>     economy</t8>
                    </t7>
                </t5>
            </t4>
            <t9>
                <t10>
                    <t2>2           </t2>
                </t10>
            </t9>
        </t1>
    </t0>     )");

            inputWithOutMap = new std::string(R"(     <t0>
        <t1>
                  <t2>        1       </t2>
            <t3>  Ahmed  Ali  </t3>
            <t4>
                <t5>
                    <t6>  Lorem ipsum dolor sit ametffsjkn</t6>
                    <t7>
                        <t8>     economy</t8>
                    </t7>
                </t5>
            </t4>
            <t9>
                <t10>
                    <t2>2           </t2>
                </t10>
            </t9>
        </t1>
    </t0>     )");

            result = new std::string(R"(      <users>
        <user>
                  <id>        1       </id>
            <name>  Ahmed  Ali  </name>
            <posts>
                <post>
                    <body>  Lorem ipsum dolor sit ametffsjkn</body>
                    <topics>
                        <topic>     economy</topic>
                    </topics>
                </post>
            </posts>
            <followers>
                <follower>
                    <id>2           </id>
                </follower>
            </followers>
        </user>
    </users>     )");
        }

        void TearDown() {
            delete t;
            t = nullptr;
            delete inputWithMap;
            inputWithMap = nullptr;
            delete inputWithOutMap;
            inputWithOutMap = nullptr;
            delete result;
            result = nullptr;
        }
    };

    TEST_F(TagsMapDecTest, withMap) {
        t = new TagsMapDec(inputWithMap);
        std::string* s = t->decompress();
        EXPECT_EQ(*s, *result);

        delete s;
        s = nullptr;
    }

    TEST_F(TagsMapDecTest, withOutMap) {
        t = new TagsMapDec(inputWithOutMap);
        std::string* s = t->decompress();
        EXPECT_EQ(*s, *result);

        delete s;
        s = nullptr;
    }
}
```

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.

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

void Map::trimString(std::string& str)
{
    str.erase(0, str.find_first_not_of(' ')); // Remove leading spaces
    str.erase(str.find_last_not_of(' ') + 1); // Remove trailing spaces
}

Map::Map(const std::string* tagMapBlock)
{
    this->arr = new std::vector<std::string*>();
    //clear the spaces of the file.
    MinifyingXML* m = new MinifyingXML(tagMapBlock);
    std::string* afterMini = m->minifyString();
    delete m;
    m = nullptr;

    // Get the positions of the opening and closing tags to remove them
    int openingTagPos = afterMini->find("<TagMap>");
    int closingTagPos = afterMini->find("</TagMap>");

    //check that the tag is available.
    if (openingTagPos == std::string::npos
        || closingTagPos == std::string::npos) {
        throw std::runtime_error("Defected TagMAp block");
    }

    //erase the tag
    afterMini->erase(openingTagPos, 8); // Erase the opening tag "<TagMap>"
    afterMini->erase(afterMini->size() - 9, 9); // Erase the closing tag "</TagMap>"

    // add the values between ',' into the arr vector.
    std::stringstream ss(*afterMini);
    std::string* token = new std::string();

    while (std::getline(ss, *token, ',')) {
        Map::trimString(*token);
        this->add(token);
        token = new std::string();
    }
    delete token;
    token = nullptr;

    delete afterMini;
    afterMini = nullptr;
}

int Map::add(std::string* key)
{
    arr->push_back(key);
    return arr->size() - 1;
}

int Map::getValue(const std::string* key) const
{
    int counter = -1;
    for (int i = 0; i < arr->size(); i++) {
        std::string* k = arr->at(i);
        if (*k == *key) {
            return i;
        }
    }
    return -1;
}

const std::string* Map::getKey(int value) const
{
    if (arr->size() == 0) {
        throw std::runtime_error("array out of bound exception");
    }
    if (value < 0 || value> arr->size() - 1) {
        throw std::runtime_error("array out of bound exception");
    }
    return arr->at(value);
}

bool Map::containKey(const std::string* key) const {
    return (this->getValue(key) == -1) ? false : true;
}

```

```
std::string* Map::toString()
{
    if (arr->size() == 0) {
        throw std::runtime_error("No value are being mapped");
    }
    std::string* result = new std::string("<TagMap>");
    for (std::string* s : *arr) {
        result->append(*s);
        result->append(",");
    }
    result->erase(result->size() - 1);
    result->append("</TagMap>");
    return result;
}
```

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.

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

namespace {
    class Map_Test : public ::testing::Test {
    public:
        Map* m;
        std::string* s0;
        std::string* s1;
        std::string* s2;
        std::string* s3;

    protected:
        void SetUp() override {
            m = new Map();
            s0 = new std::string("v0");
            s1 = new std::string("v1");
            s2 = new std::string("v2");
            s3 = new std::string("v3");
        }

        void add() {
            m->add(s0);
            m->add(s1);
            m->add(s2);
            m->add(s3);
        }
        void TearDown() override {
            delete m;
            m = nullptr;
        }
    };

    TEST_F(Map_Test, emptyMap) {
        EXPECT_EQ(m->getSize(), 0);
        std::string* s = new std::string("any");
        EXPECT_EQ(m->getValue(s), -1);

        EXPECT_THROW(m->getKey(0), std::runtime_error);
        EXPECT_THROW(m->getKey(-1), std::runtime_error);
        EXPECT_THROW(m->getKey(5), std::runtime_error);

        EXPECT_FALSE(m->containKey(s));

        EXPECT_THROW(m->toString(), std::runtime_error);

        delete s;
        s = nullptr;
    }

    TEST_F(Map_Test, AddToTheMap) {
        add();

        EXPECT_EQ(m->getSize(), 4);

        EXPECT_EQ(m->getValue(s0), 0);
        EXPECT_EQ(m->getValue(s1), 1);
        EXPECT_EQ(m->getValue(s2), 2);
        EXPECT_EQ(m->getValue(s3), 3);

        EXPECT_THROW(m->getKey(5), std::runtime_error);

        EXPECT_EQ(m->getKey(0), s0);
        EXPECT_EQ(m->getKey(1), s1);
        EXPECT_EQ(m->getKey(2), s2);
        EXPECT_EQ(m->getKey(3), s3);

        EXPECT_TRUE(m->containKey(s0));
        EXPECT_TRUE(m->containKey(s1));
        EXPECT_TRUE(m->containKey(s2));
        EXPECT_TRUE(m->containKey(s3));

        std::string eOutput = "<TagMap>v0,v1,v2,v3</TagMap>";
        std::string* output = m->toString();
        EXPECT_EQ(*output, eOutput);
        delete output;
        output = nullptr;
    }

    class Map_Test2 : public::testing::Test {
    public:
        Map* m;
        std::string* TagMapBlock;
        std::string* output;
    protected:
        void SetUp() override {
            TagMapBlock = new std::string(R"(     <TagMap>   v0,v1,
       v2,   v3
       </TagMap>
       )");
            output = new std::string(R"(<TagMap>v0,v1,v2,v3</TagMap>)");
            m = new Map(TagMapBlock);
        }
        void TearDown() override {
            delete m;
            m = nullptr;
            delete TagMapBlock;
            TagMapBlock = nullptr;
            delete output;
            output = nullptr;
        }
    }; // Map_Test2

    TEST_F(Map_Test2, MapInitConstrucotr) {
        EXPECT_EQ(m->getSize(), 4);
        EXPECT_EQ(*m->getKey(0), "v0");
        EXPECT_EQ(*m->getKey(1), "v1");
        EXPECT_EQ(*m->getKey(2), "v2");
        EXPECT_EQ(*m->getKey(3), "v3");
        std::string* s = m->toString();
        EXPECT_EQ(*s, *output);
        delete s;
        s = nullptr;
    }
} // namespace
```

Index