Compression

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 documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 ClearClosingTagsComp Class Reference

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

- Sample2/ClearClosingTagsComp.h

3.2 Map Class Reference

Public Member Functions

- **Map** ()

  C'tor. Initializes empty map with an empty dynamic array.
- Map (const std::string ∗tagMapBlock)

  C'tor. Initialize the map from a <TagMap> block.
- ∼Map ()
- int add (std::string ∗key)

  Adds the key to the map.
- int getValue (const std::string ∗key) const

  The value that the key is mapped to.
- const std::string ∗ getKey (int value) const

  Get the key from the value that the key was mapped to.
- bool containKey (const std::string ∗key) const

  Checks if the map contains that key.
- int getSize ()
- std::string ∗ toString ()

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

3.2.1 Constructor & Destructor Documentation

3.2.1.1 Map()

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

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

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

Parameters

tagMapBlock.	

Exceptions

runtime_error	if the file is defected.

3.2.1.2 ∼Map()

```
Map::∼Map ( )  [inline]
```

D'tor.

Warning

It will delete all the keys string assigned to it.

3.2.2 Member Function Documentation

3.2.2.1 add()

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

Adds the key to the map.

Parameters

key	To add.

Returns

The value that the key is mapped to.

3.2.2.2 containKey()

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

Checks if the map contains that key.

Parameters

key.	

Returns

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

3.2.2.3 getKey()

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

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

Parameters

value	

Returns

The key.

Exceptions

runtime	error if the value is not in the map.

3.2.2.4 getSize()

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

Returns

the size of the map.

3.2.2.5 getValue()

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

The value that the key is mapped to.

Parameters

key.	

Returns

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

3.2.2.6 toString()

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

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

Exceptions

runtime	error if the map is empty.

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

- Sample2/Map.h
- Sample2/Map.cpp

3.3 MinifyingXML Class Reference

Public Member Functions

- MinifyingXML (const std::string *xmlFile)

  C'tor.
- std::string * minifyString ()

  This function deletes any spaces and new lines from the XML File.
- const std::string * **getXMLFile** () const
- void **setXMLFile** (const std::string *xmlFileNew)

Static Public Member Functions

- static bool isSkipChar (const char c)

  function checks if the char is one the located characters.

Related Symbols

(Note that these are not member symbols.)

- void skipFromBeginning (std::string *result) const

  This function clears all the skip Chars except some spaces.
- void skipFromEnd (std::string *result) const

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

3.3.1 Constructor & Destructor Documentation

3.3.1.1 MinifyingXML()

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

C'tor.

See also

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

Parameters

xmlFile	

3.3.2 Member Function Documentation

3.3.2.1 isSkipChar()

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

function checks if the char is one the located characters.

Parameters

c	-The character to check.

Returns

- True if it's a skip char.

  - False otherwise.

See also

MinifyingXML::charToSkip array.

3.3.2.2 minifyString()

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

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

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

See also

MinifyingXML::charToSkip array.

Returns

The result string from minifying function.

3.3.3 Friends And Related Symbol Documentation

3.3.3.1 skipFromBeginning()

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

This function clears all the skip Chars except some spaces.

Operation:

- For all the string, skip (don't add it into the result) all charToskip elements except spaces.
- For space cases: -> Starting from the beginning, skip any spaces until reaching the first closing tag '>'. -> After the closing 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.

3.3.3.2 skipFromEnd()

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

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

Operation: Starting from the last element.

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

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

- Other skip chars are eliminated from the prev step.

See also

void MinifyingXML::skipFromBeginning(std::string∗ result).

helper function for: std::string∗ MinifyingXML::minifyString(). be called after void MinifyingXML::skipFrom↵Beginning(std::string∗ result).

Parameters

result	The result string from the prev step to modify it.

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

- Sample2/MinifyingXML.h
- Sample2/MinifyingXML.cpp

3.4 TagsMapComp Class Reference

Public Member Functions

- TagsMapComp (const std::string ∗xmlFile)

  C'tor.
- ∼**TagsMapComp** ()

  D'tor.
- void mapTags ()

  Reads the XML file and create the map with all the tags.
- std::string ∗ compress (bool addMapTable=false)

  This function compresses the XML file.

3.4.1 Constructor & Destructor Documentation

3.4.1.1 TagsMapComp()

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

C'tor.

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

Parameters

xmlFile	

3.4.2 Member Function Documentation

3.4.2.1 compress()

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

This function compresses the XML file.

Operation:

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

Parameters

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.

3.4.2.2 mapTags()

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

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

Explanation:

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

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

- Sample2/TagsMapComp.h
- Sample2/TagsMapComp.cpp

3.5 TagsMapDec Class Reference

Public Member Functions

- std::string ∗ **decompress** ()

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

- Sample2/TagsMapDec.h
- Sample2/TagsMapDec.cpp

Chapter 4

File Documentation

4.1 Sample2/ClearClosingTagsComp.h File Reference

```
#include <string>
```

Classes

- class ClearClosingTagsComp

4.1.1 Detailed Description

Author

eslam

Date

December 2023

4.2 ClearClosingTagsComp.h

Go to the documentation of this file.

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

#include <string>

class ClearClosingTagsComp
{
}; // class ClearClosingTagsComp
#endif // !CLEAR_CLOSING_TAGS_COMP_H
```

4.3 Sample2/Map.cpp File Reference

The source file of the simple Map.

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

4.3.1 Detailed Description

The source file of the simple Map.

This a simple implementation of Map DS that will help Mapping tags into numbers.

Author

eslam

Date

December 2023

4.4 Sample2/Map.h File Reference

The header file of the simple Map.

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

Classes

- class Map

4.4.1 Detailed Description

The header file of the simple Map.

This a simple implementation of Map DS that will help Mapping tags into numbers.

Author

eslam

Date

December 2023

4.5 Map.h

Go to the documentation of this file.

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

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

class Map
{
private:
    std::vector<std::string*>* arr;

    //helper method
    static void trimString(std::string& str);

public:
    explicit Map() :arr(new std::vector<std::string*>()) {}
    explicit Map(const std::string* tagMapBlock);
    ~Map() {
        for (std::string* s : *arr) {
            delete s;
        }
        delete arr;
    }

    //methods

    int add(std::string* key);
    int getValue(const std::string* key) const;
    const std::string* getKey(int value) const;
    bool containKey(const std::string* key) const;

    int getSize() { return arr->size(); }

    std::string* toString();
};

#endif // !MAP_H
```

4.6 Sample2/Map_unittest.cpp File Reference

Unit test code for Map class.

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

4.6.1 Detailed Description

Unit test code for Map class.

Author

eslam

Date

December 2023

4.7 Sample2/MinifyingXML.cpp File Reference

The source file of class MinifyingXML.

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

4.7.1 Detailed Description

The source file of class MinifyingXML.

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

This class will minify any flawless XML file.

Operation summary:

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

Author

eslam

Date

December 2023

4.8 Sample2/MinifyingXML.h File Reference

Header file of the MinifyingXML class.

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

Classes

- class MinifyingXML

4.8.1 Detailed Description

Header file of the MinifyingXML class.

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

This class will minify any flawless XML file.

Operation summary:

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

Author

eslam

Date

December 2023

4.9 MinifyingXML.h

Go to the documentation of this file.

```
/*******************************************************************/
#pragma once
#ifndef MINIFYING_XML_H
#define MINIFYING_XML_H
#include <string>

#include <stdexcept>

class MinifyingXML
{
private:
    // the file that neads to be minified.
    const std::string* xmlFile;

    // char To skip in minifying.
    static const char charToSkip[5];

public:
    explicit MinifyingXML(const std::string* xmlFile) : xmlFile(xmlFile) {
        // check adding a null ptr.
        if (xmlFile == nullptr) {
            throw std::logic_error("Null pointer exception: Accessing null pointer!");
        }
    }

    //methods

    std::string* minifyString();

    static bool isSkipChar(const char c);

    //helper methods

    void skipFromBeginning(std::string* result)const;

    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
```

4.10 Sample2/MinifyingXML_unittest.cpp File Reference

Unit test code for MinifyingXML class.

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

4.10.1 Detailed Description

Unit test code for MinifyingXML class.

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

Author

eslam

Date

December 2023

4.11 pch.h

```
//
// pch.h
//

#pragma once

#include "gtest/gtest.h"

int main(int argc, char** argv) {
    testing::InitGoogleTest(&argc, argv);

    // Run all tests
    return RUN_ALL_TESTS();
}
```

4.12 Sample2/TagMapComp_unittest.cpp File Reference

Unit test code for TagsMapComp class.

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

4.12.1 Detailed Description

Unit test code for TagsMapComp class.

Author

eslam

Date

December 2023

4.13 Sample2/TagsMapComp.cpp File Reference

The source file of TagsMapComp class.

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

4.13.1 Detailed Description

The source file of TagsMapComp class.

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

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

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

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

Note

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

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

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

Author

eslam

Date

December 2023

4.14 Sample2/TagsMapComp.h File Reference

The header file of TagsMapComp class.

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

Classes

- class TagsMapComp

4.14.1 Detailed Description

The header file of TagsMapComp class.

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

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

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

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

Note

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

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

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

Author

eslam

Date

December 2023

4.15 TagsMapComp.h

Go to the documentation of this file.

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

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

class TagsMapComp
{
private:
    const std::string* xmlFile;
    //Map of tag values.
    Map* map;
public:
    explicit TagsMapComp(const std::string* xmlFile) : xmlFile(xmlFile),
        map(new Map()) {
        mapTags();
    }
    ~TagsMapComp() { delete map; }
    void mapTags();

    std::string* compress(bool addMapTable = false);
};

#endif // !TAGS_MAP_Comp_H
```

4.16 Sample2/TagsMapDec.h File Reference

The header file of TagsMapDec class.

`#include "Map.h"`

Classes

- class TagsMapDec

4.16.1 Detailed Description

The header file of TagsMapDec class.

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

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

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

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

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

See also

TagsMapComp

Author

eslam

Date

December 2023

4.17 TagsMapDec.h

Go to the documentation of this file.

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

#include "Map.h"

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

    const static std::string* defaultTagMapBlock;

    //Map of tag values.
    Map* map;

    //helper methods
    void getMapTags();
    void getTagsMapBlock();
public:
    explicit TagsMapDec();
    ~TagsMapDec();

    std::string* decompress();
};
#endif // !TAGS_MAP_DEC_H
```

Index