My Project

Generated by Doxygen 1.10.0

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 CommandLineInput Struct Reference

Structure for handling command-line input parameters.

`#include <ComandLineInput.h>`

Public Member Functions

- CommandLineInput (int argc, char ∗argv[ ])

  Constructor for CommandLineInput, parses command-line arguments.

Public Attributes

- std::string **inputFileName**

  Input file name.
- std::string **outputFileName**

  Output file name.
- bool **encoding**

  Flag indicating encoding operation.
- bool **decoding**

  Flag indicating decoding operation.

3.1.1 Detailed Description

Structure for handling command-line input parameters.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 CommandLineInput()

```
CommandLineInput::CommandLineInput (
            int argc,
            char * argv[] )  [inline]
```

Constructor for CommandLineInput, parses command-line arguments.

Parameters

argc	Number of command-line arguments.
argv	Array of command-line argument strings.

Exceptions

std::exception	if there are missing or invalid options.

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

- ComandLineInput.h

3.2 Dictionary Class Reference

Represents the Huffman coding dictionary.

`#include <Dictionary.h>`

Public Member Functions

- void **sort** ()

  Sorts the dictionary based on the comparator and removes elements with Quantity equal to 0.
- Dictionary (int)

  Constructor for Dictionary with specified size.
- **Dictionary** ()

  Default constructor for Dictionary with size 255.
- ∼**Dictionary** ()

  Destructor for Dictionary.
- char decode (std::string &code)

  Decodes a given Huffman code to a character.

Public Attributes

- std::vector< DictionaryElement > **tab**

  Vector representing the dictionary.

3.2.1 Detailed Description

Represents the Huffman coding dictionary.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Dictionary()

```
Dictionary::Dictionary (
            int i )
```

Constructor for Dictionary with specified size.

Parameters

i	Size of the dictionary.

3.2.3 Member Function Documentation

3.2.3.1 decode()

```
char Dictionary::decode (
            std::string & code )
```

Decodes a given Huffman code to a character.

Parameters

code	Huffman code to decode.

Returns

Decoded character, or -1 if not found.

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

- Dictionary.h
- Dictionary.cpp

3.3 DictionaryElement Struct Reference

Represents an element in the Huffman coding dictionary.

```
#include <Dictionary.h>
```

Public Member Functions

- **DictionaryElement** ()
  Default constructor for DictionaryElement.
- DictionaryElement (std::pair< std::string, char >)
  Constructor for DictionaryElement with a pair of code and character.
- DictionaryElement (std::string &, char)
  Constructor for DictionaryElement with specified code and character.
- ~**DictionaryElement** ()
  Destructor for DictionaryElement.
- int operator++ (int)
  Post-increment operator for Quantity.
- void setCode (const std::string &newCode)
  Appends a given code to the existing code of the dictionary element.

Public Attributes

- std::string **code**

  Huffman code.
- int **Quantity**

  Frequency or quantity of the character.
- char **Character**

  Character represented by the code.

3.3.1 Detailed Description

Represents an element in the Huffman coding dictionary.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 DictionaryElement() [1/2]

```
DictionaryElement::DictionaryElement (
            std::pair< std::string, char > p )
```

Constructor for DictionaryElement with a pair of code and character.

Parameters

p	Pair of code and character.

3.3.2.2 DictionaryElement() [2/2]

```
DictionaryElement::DictionaryElement (
            std::string & c,
            char ch )
```

Constructor for DictionaryElement with specified code and character.

Parameters

c	Code for the dictionary element.
ch	Character for the dictionary element.

3.3.3 Member Function Documentation

3.3.3.1 operator++()

```
int DictionaryElement::operator++ (
            int  )  [inline]
```

Post-increment operator for Quantity.

Parameters

Unused	parameter (int).

Returns

The previous value of Quantity.

3.3.3.2 setCode()

```
void DictionaryElement::setCode (
            const std::string & code )
```

Appends a given code to the existing code of the dictionary element.

Parameters

newCode	Code to append.
code	Code to append.

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

- Dictionary.h
- Dictionary.cpp

3.4 Engiene Struct Reference

Represents the Huffman coding/decoding engine.

```
#include <huffmanEngiene.h>
```

Public Member Functions

- Engiene (std::string in, std::string out, bool encode)

  Constructor for Engiene.
- ~Engiene ()

  Destructor for Engiene.
- void **countChars** ()

  Counts the occurrences of characters in the input file and builds the frequency table.
- void SetElement (int id, std::string &code)

  Sets the character code for a specific dictionary element.
- void **GetDictionary** ()

  Retrieves the dictionary from the input file.
- void **Decode** ()

  Decodes the input file using the Huffman coding.
- void **writeDictionary** ()

  Writes the Huffman dictionary to the output file.
- void **code** ()

  Encodes the input file using Huffman coding.

Public Attributes

- FileIO * **file**

  Pointer to FileIO object.
- Dictionary * **dictionary**

  Pointer to Dictionary object.

3.4.1 Detailed Description

Represents the Huffman coding/decoding engine.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Engiene()

```
Engiene::Engiene (
            std::string in,
            std::string out,
            bool encode )
```

Constructor for Engiene.

Constructor for HuffmanEngiene.

Parameters

in	Input file name.
out	Output file name.
encode	Flag indicating encoding operation.

3.4.2.2 ~Engiene()

```
Engiene::~Engiene ( )
```

Destructor for Engiene.

Destructor for HuffmanEngiene.

3.4.3 Member Function Documentation

3.4.3.1 SetElement()

```
void Engiene::SetElement (
            int id,
            std::string & code )
```

Sets the character code for a specific dictionary element.

Parameters

id	Index of the dictionary element.
code	New character code to set.

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

- huffmanEngiene.h
- huffmanEngiene.cpp

3.5 FileIO Class Reference

Public Member Functions

- **FileIO** (std::string &input, std::string &output)
- char **getChar** ()
- std::pair< std::string, int > **getDicElement** ()
- bool **getCode** (std::string *)
- void **resetFile** (std::string &)
- void **wr** (std::string &)
- void **wr** (char)

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

- FileIO.h
- FileIO.cpp

3.6 graph Class Reference

Represents the Huffman tree for encoding/decoding.

`#include <Graph.h>`

Public Member Functions

- graph (std::vector< DictionaryElement > &)

  Constructor for graph with a given vector of DictionaryElement.
- ~**graph** ()

  Destructor for graph.
- void **updateDictionary** ()

  Updates the dictionary with Huffman codes after tree initialization.
- void req (Node *node, std::string *code)

  Recursively traverses the Huffman tree and assigns Huffman codes to characters.

Public Attributes

- std::vector< DictionaryElement > * **dic**

  Pointer to the dictionary.

3.6.1 Detailed Description

Represents the Huffman tree for encoding/decoding.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 graph()

```
graph::graph (
            std::vector< DictionaryElement > & tab )
```

Constructor for graph with a given vector of DictionaryElement.

Parameters

tab	Vector of DictionaryElement representing character frequencies.

3.6.3 Member Function Documentation

3.6.3.1 req()

```
void graph::req (
            Node * node,
            std::string * code )
```

Recursively traverses the Huffman tree and assigns Huffman codes to characters.

Parameters

node	Current node in the traversal.
code	Huffman code built during traversal.

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

- Graph.h
- Graph.cpp

3.7 Node Struct Reference

Represents a node in the Huffman tree.

`#include <Graph.h>`

Public Member Functions

- **Node** ()

  Default constructor for Node.
- Node (int value, char)

  Constructor for Node with specified value and character.
- ∼**Node** ()

  Destructor for Node.

Public Attributes

- int **value**

  Value or frequency of the node.
- Node ∗ **leftChild**

  Pointer to the left child node.
- Node ∗ **rightChild**

  Pointer to the right child node.
- char **c**

  Character represented by the node.

3.7.1 Detailed Description

Represents a node in the Huffman tree.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 Node()

```
Node::Node (
            int value,
            char c )
```

Constructor for Node with specified value and character.

Parameters

value	Value or frequency of the node.
c	Character represented by the node.

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

- Graph.h
- Graph.cpp

Chapter 4

File Documentation

4.1 ComandLineInput.h

```
00001
00006 #pragma once
00007
00008 #include <exception>
00009 #include <string>
00010
00015 struct CommandLineInput {
00016     std::string inputFileName;
00017     std::string outputFileName;
00018     bool encoding;
00019     bool decoding;
00020
00027     CommandLineInput(int argc, char* argv[]);
00028 };
00029
00036 inline CommandLineInput::CommandLineInput(int argc, char* argv[]) {
00037     encoding = false;
00038     decoding = false;
00039     if (argc < 2) {
00040         throw std::exception("Usage: -i input_file -o output_file -c for coding and/or -d for
      decoding");
00041     }
00042
00043     // Processing command-line arguments
00044     for (int i = 1; i < argc; ++i) {
00045         if (std::string(argv[i]) == "-i") {
00046             if (i + 1 < argc) {
00047                 inputFileName = argv[i + 1];
00048                 ++i;
00049             }
00050             else {
00051                 throw std::exception("Error: -i option requires the input file name.");
00052             }
00053         }
00054         else if (std::string(argv[i]) == "-o") {
00055             if (i + 1 < argc) {
00056                 outputFileName = argv[i + 1];
00057                 ++i;
00058             }
00059             else {
00060                 throw std::exception("Error: -o option requires the output file name.");
00061             }
00062         }
00063         else if (std::string(argv[i]) == "-c") {
00064             encoding = true;
00065             ++i;
00066         }
00067         else if (std::string(argv[i]) == "-d") {
00068             decoding = true;
00069             ++i;
00070         }
00071     }
00072
00073     // Check if all required options are provided
00074     if (inputFileName.empty() || outputFileName.empty() || (!decoding && !encoding)) {
00075         throw std::exception("Error: Missing required options.");
00076     }
00077     else {
```

```
00078         if (decoding && encoding) {
00079             throw std::exception("Error: You can't encode and decode at the same time.");
00080         }
00081     }
00082 }
```

4.2 Dictionary.h

```
00001 #pragma once
00002
00003 #include <vector>
00004 #include <string>
00005 #include <algorithm>
00006
00011 struct DictionaryElement
00012 {
00013     std::string code;
00014     int Quantity;
00015     char Character;
00016
00020     DictionaryElement();
00021
00026     DictionaryElement(std::pair<std::string, char>);
00027
00033     DictionaryElement(std::string&, char);
00034
00038     ~DictionaryElement();
00039
00045     int operator++(int)
00046     {
00047         int temp = Quantity;
00048         Quantity++;
00049         return temp;
00050     }
00051
00056     void setCode(const std::string& newCode);
00057 };
00058
00063 class Dictionary
00064 {
00065 private:
00072     static bool comparator(const DictionaryElement& a, const DictionaryElement& b) {
00073         return a.Quantity > b.Quantity;
00074     }
00075
00076 public:
00080     void sort();
00081
00082     std::vector<DictionaryElement> tab;
00083
00088     Dictionary(int);
00089
00093     Dictionary();
00094
00098     ~Dictionary();
00099
00105     char decode(std::string& code);
00106 };
```

4.3 FileIO.h File Reference

Declaration of the FileIO class for file input/output operations.

```
#include <fstream>
#include <vector>
#include <string>
#include <cstdlib>
#include <bitset>
```

Classes

- class FileIO

4.3.1 Detailed Description

Declaration of the FileIO class for file input/output operations.

4.4 FileIO.h

Go to the documentation of this file.

```
00001
00006 #pragma once
00007 #include <fstream>
00008 #include <vector>
00009 #include <string>
00010 #include <cstdlib>
00011 #include <bitset>
00012
00013
00014 class FileIO
00015 {
00016 public:
00017
00018     FileIO(std::string& input, std::string& output);
00019
00020
00021     ~FileIO();
00022
00023
00024     char getChar();
00025     std::pair<std::string, int>  getDicElement();
00026     bool getCode(std::string*);
00027     void resetFile(std::string&);
00028     void wr(std::string&);
00029     void wr(char);
00030
00031 private:
00032     std::ifstream inputFile;
00033     std::ofstream outputFile;
00034 };
```

4.5 Graph.h

```
00001 #pragma once
00002 #include "Dictionary.h"
00003 #include <vector>
00004 #include <string>
00005 #include <queue>
00006
00011 struct Node
00012 {
00016     Node();
00017
00023     Node(int value, char);
00024
00028     ~Node();
00029
00030     int value;
00031     Node* leftChild;
00032     Node* rightChild;
00033     char c;
00034 };
00035
00040 class graph
00041 {
00042 public:
00047     graph(std::vector<DictionaryElement>&);
00048
00052     ~graph();
00053
00057     void updateDictionary();
00058
00059     std::vector<DictionaryElement>* dic;
00060
00066     void req(Node* node, std::string* code);
00067
00068 private:
00075     static bool comparator(const Node& a, const Node& b) {
```

```
00076         return a.value > b.value;
00077     }
00078
00079     Node firstNode;
00080     std::vector<Node> nodeTab;
00081
00085     void initialize();
00086 };
```

4.6 huffmanEngiene.cpp File Reference

Implementation of the HuffmanEngiene class for Huffman coding/decoding.

```
#include "HuffmanEngiene.h"
```

4.6.1 Detailed Description

Implementation of the HuffmanEngiene class for Huffman coding/decoding.

4.7 huffmanEngiene.h

```
00001 #pragma once
00002 #include "FileIO.h"
00003 #include "Dictionary.h"
00004 #include "Graph.h"
00005 #include <string>
00006 #include <vector>
00007
00012 struct Engiene
00013 {
00020     Engiene(std::string in, std::string out, bool encode);
00021
00025     ~Engiene();
00026
00027     FileIO* file;
00028     Dictionary* dictionary;
00029
00033     void countChars();
00034
00040     void SetElement(int id, std::string& code);
00041
00045     void GetDictionary();
00046
00050     void Decode();
00051
00055     void writeDictionary();
00056
00060     void code();
00061 };
```

Index