Huffman

Generated by Doxygen 1.9.6

Chapter 1

README

Program do kompresji/dekompresji plików przy użyciu metody Huffmana.

Usage: huffman -i <input_file> -o <output_file> -t <mode> -s <dictionary_file>

Available parameters: -i <input_file> - path to the input file -o <output_file> - path to the output file -t <mode> - operation mode; k <- compression, d <- decompression -s <dictionary_path> - path to the dictionary file

Projekt na PPK.

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 huffman_node Struct Reference

huffman node structure used later to create a tree of frequencies.

```
#include <huffman.h>
```

Public Attributes

- char symbol
- unsigned int frequency
- huffman_node * **left**
- huffman_node * right

4.1.1 Detailed Description

huffman node structure used later to create a tree of frequencies.

4.1.2 Member Data Documentation

4.1.2.1 frequency

```
unsigned int huffman_node::frequency
```

The frequency of that letter it has been apearing in data

4.1.2.2 right

```
huffman_node * huffman_node::right
```

Pointers to the left (1) and right (0) child of a node.

4.1.2.3 symbol

```
char huffman_node::symbol
```

The symbol, letter that's stored in a node, later it's just a null character

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

- huffman.h

Chapter 5

File Documentation

5.1 args_handler.cpp File Reference

Implements functions defined in args_handler.h.

```
#include <iostream>
#include <map>
#include <vector>
#include "args_handler.h"
#include "error_handler.h"
```

Functions

- bool check_arg (std::string &value, std::string arg)

  Checks if parameters exist.
- bool push_into_error_buffer (std::vector< std::string > &target, std::string &error)

  Pushes errors into an error buffer so if more errors occure in one go the program will list them line-by-line.
- bool arguments_handler (int argc, char ∗argv[ ], std::string &input, std::string &output, std::string &mode, std::string &dictionary)

  Handles the arguments passed to the program.

Variables

- std::map< std::string, std::string > error_messages

5.1.1 Detailed Description

Implements functions defined in args_handler.h.

Author

Michał Czyż

5.1.2 Function Documentation

5.1.2.1 arguments_handler()

```
bool arguments_handler (
             int argc,
             char * argv[],
             std::string & input,
             std::string & output,
             std::string & mode,
             std::string & dictionary )
```

Handles the arguments passed to the program.

Parameters

argc	The number of arguments passed to the program.
argv	An array of pointers to the arguments passed to the program.
input	A reference to a string that will contain the file name of the input.
output	A reference to a string that will contain the file name of the output.
mode	A reference to a string that will contain program mode (compression/decompression).
dictionary	A reference to a string that will contain the file name of the dictionary file.

Returns

Will return true if the arguemtns were successfully handled. Otherwise, it will return false.

5.1.2.2 check_arg()

```
bool check_arg (
             std::string & value,
             std::string arg )
```

Checks if parameters exist.

Parameters

value	Value of an arguement.
arg	Passed argument.

Returns

Returns false if there is a problem i.e. arg is empty or there is no dash.

5.1.2.3 push_into_error_buffer()

```
bool push_into_error_buffer (
             std::vector< std::string > & target,
             std::string & error )
```

Pushes errors into an error buffer so if more errors occure in one go the program will list them line-by-line.

Parameters

target	A reference to the target vector of string holding all errors.
error	A reference to a string holding an error message.

Returns

Will always return false to signal that an error occured.

5.1.3 Variable Documentation

5.1.3.1 error_messages

```
std::map<std::string, std::string> error_messages
```

Initial value:

```
{
    {"not_enough", "Too few arguments.\n"},
    {"input", "-i argument requires <input_file> path.\n"},
    {"output", "-o argument requires <output_file> path.\n"},
    {"mode", "-t argument requires <mode>; k <- compression, d <- decompression.\n"},
    {"dictionary", "-s argument requires <dictionary_file> path.\n"},
}
```

5.2 args_handler.h File Reference

Defines functions needed for parameters recognition and error handling.

```
#include <iostream>
#include <map>
#include <vector>
#include <string>
```

Functions

- bool check_arg (std::string &value, std::string arg)

  Checks if parameters exist.
- bool push_into_error_buffer (std::vector< std::string > &target, std::string &error)

  Pushes errors into an error buffer so if more errors occure in one go the program will list them line-by-line.
- bool arguments_handler (int argc, char $*$argv[ ], std::string &input, std::string &output, std::string &mode, std::string &dictionary)

  Handles the arguments passed to the program.

5.2.1 Detailed Description

Defines functions needed for parameters recognition and error handling.

Author

Michał Czyż

5.2.2 Function Documentation

5.2.2.1 arguments_handler()

```
bool arguments_handler (
            int argc,
            char * argv[],
            std::string & input,
            std::string & output,
            std::string & mode,
            std::string & dictionary )
```

Handles the arguments passed to the program.

Parameters

argc	The number of arguments passed to the program.
argv	An array of pointers to the arguments passed to the program.
input	A reference to a string that will contain the file name of the input.
output	A reference to a string that will contain the file name of the output.
mode	A reference to a string that will contain program mode (compression/decompression).
dictionary	A reference to a string that will contain the file name of the dictionary file.

Returns

Will return true if the arguemtns were successfully handled. Otherwise, it will return false.

5.2.2.2 check_arg()

```
bool check_arg (
            std::string & value,
            std::string arg )
```

Checks if parameters exist.

Parameters

value	Value of an arguement.
arg	Passed argument.

Returns

Returns false if there is a problem i.e. arg is empty or there is no dash.

5.2.2.3 push_into_error_buffer()

```
bool push_into_error_buffer (
            std::vector< std::string > & target,
            std::string & error )
```

Pushes errors into an error buffer so if more errors occure in one go the program will list them line-by-line.

Parameters

target	A reference to the target vector of string holding all errors.
error	A reference to a string holding an error message.

Returns

Will always return false to signal that an error occured.

5.3 args_handler.h

Go to the documentation of this file.

```

#ifndef ARGS_H
#define ARGS_H

#include <iostream>
#include <map>
#include <vector>
#include <string>

bool check_arg(std::string &value, std::string arg);

bool push_into_error_buffer(std::vector<std::string> &target, std::string &error);

bool arguments_handler(int argc, char *argv[], std::string &input, std::string &output, std::string
      &mode, std::string &dictionary);

#endif
```

5.4 error_handler.cpp File Reference

Implements functions that are defined in error_handler.h.

```
#include <iostream>
#include "error_handler.h"
```

Functions

- bool error_handler (std::string program_name, std::string message="")

  Generates an error output in the console and help, with an instruction of how to use a program.

5.4.1 Detailed Description

Implements functions that are defined in error_handler.h.

Author

Michał Czyż

5.4.2 Function Documentation

5.4.2.1 error_handler()

```
bool error_handler (
            std::string program_name,
            std::string message )
```

Generates an error output in the console and help, with an instruction of how to use a program.

Parameters

program_name	A string that holds a name of a program to printout.
message	A string containing an error message to printout.

Returns

Always returns false.

5.5 error_handler.h File Reference

Simple header that holds the general error handler for the program.

```
#include <iostream>
```

Functions

- bool error_handler (std::string program_name, std::string message)

  Generates an error output in the console and help, with an instruction of how to use a program.

5.5.1 Detailed Description

Simple header that holds the general error handler for the program.

Author

Michał Czyż

5.5.2 Function Documentation

5.5.2.1 error_handler()

```
bool error_handler (
             std::string program_name,
             std::string message )
```

Generates an error output in the console and help, with an instruction of how to use a program.

Parameters

program_name	A string that holds a name of a program to printout.
message	A string containing an error message to printout.

Returns

Always returns false.

5.6 error_handler.h

Go to the documentation of this file.

```
#ifndef ERROR_H
#define ERROR_H

#include <iostream>

bool error_handler(std::string program_name, std::string message);

#endif
```

5.7 file_handler.cpp File Reference

Implements functions for interaction with the filesystem defined in file_handler.h.

```
#include <iostream>
#include <map>
#include <sstream>
#include <sys/stat.h>
#include <algorithm>
#include "file_handler.h"
#include "error_handler.h"
```

Functions

- bool read_file (std::string program_name, std::string &data, std::string &file_name, bool binary)

  Reads data from the provided file.
- bool write_file (std::string program_name, std::string &data, std::string &file_name, bool binary)

  Writed data to the provided file.

Variables

- const int **chunk_size** = 8

5.7.1 Detailed Description

Implements functions for interaction with the filesystem defined in file_handler.h.

Author

Michał Czyż

5.7.2 Function Documentation

5.7.2.1 read_file()

```
bool read_file (
            std::string program_name,
            std::string & data,
            std::string & file_name,
            bool binary )
```

Reads data from the provided file.

Parameters

program_name	A string containing the program_name which in case of an error is passed down to the error_handler.
data	A reference to a string that will hold the data read from the file.
file_name	A reference to a string that holds the file name of a desired file.
binary	A method in which the function should read the file (true for binary or false for normal filestream).

Returns

Returns true if operation completed successfully, false if it failed.

5.7.2.2 write_file()

```
bool write_file (
            std::string program_name,
            std::string & data,
            std::string & file_name,
            bool binary )
```

Writed data to the provided file.

Parameters

program_name	A string containing the program_name which in case of an error is passed down to the error_handler.
data	A reference to a string that holds the data.
file_name	A reference to a string that holds the file name of a desired file.
binary	A method in which the function should write into the file (true for binary or false for normal filestream).

Returns

Returns true if operation completed successfully, false if it failed.

5.8 file_handler.h File Reference

Defines functions for interaction with the filesystem.

```
#include <iostream>
#include <fstream>
```

Functions

- bool read_file (std::string program_name, std::string &data, std::string &file_name, bool binary)
  Reads data from the provided file.
- bool write_file (std::string program_name, std::string &data, std::string &file_name, bool binary)
  Writed data to the provided file.

5.8.1 Detailed Description

Defines functions for interaction with the filesystem.

Author

Michał Czyż

5.8.2 Function Documentation

5.8.2.1 read_file()

```
bool read_file (
            std::string program_name,
            std::string & data,
            std::string & file_name,
            bool binary )
```

Reads data from the provided file.

Parameters

program_name	A string containing the program_name which in case of an error is passed down to the error_handler.
data	A reference to a string that will hold the data read from the file.
file_name	A reference to a string that holds the file name of a desired file.
binary	A method in which the function should read the file (true for binary or false for normal filestream).

Returns

Returns true if operation completed successfully, false if it failed.

5.8.2.2 write_file()

```
bool write_file (
            std::string program_name,
            std::string & data,
            std::string & file_name,
            bool binary )
```

Writed data to the provided file.

Parameters

program_name	A string containing the program_name which in case of an error is passed down to the error_handler.
data	A reference to a string that holds the data.
file_name	A reference to a string that holds the file name of a desired file.
binary	A method in which the function should write into the file (true for binary or false for normal filestream).

Returns

Returns true if operation completed successfully, false if it failed.

5.9 file_handler.h

Go to the documentation of this file.

```
#ifndef FILE_HANDLER
#define FILE_HANDLER

#include <iostream>
#include <fstream>

bool read_file(std::string program_name, std::string &data, std::string &file_name, bool binary);

bool write_file(std::string program_name, std::string &data, std::string &file_name, bool binary);

#endif
```

5.10 huffman.cpp File Reference

Implements functions defined in huffman.h, contains the core functionality of this program.

```
#include <iostream>
#include <queue>
#include <map>
#include <sstream>
#include <vector>
#include <string>
#include "huffman.h"
```

Functions

- bool is_leaf (huffman_node ∗node)

  Checks if the provided node has no children. It it the leaf of a tree.
- void compress (huffman_node ∗node, std::string data, std::map< char, std::string > &frequency)

  Compresses data recursively to a binary stream.
- void decompress (std::map< std::string, char > &frequency, std::string &data, std::string &result)

  Decompresses binary stream back to the normal format.
- huffman_node ∗ create_tree (std::map< char, int > &frequency)

  Creates a huffman binary tree out of a map of frequencies.
- std::map< std::string, char > rebuild_tree (std::string &dictionary)

  Rebuilds an inverted (easier to process later) frequency map from a string.
- void clear_tree (huffman_node ∗node)

  Clears a heap of a huffman tree so the program doesn't leak any data.
- std::string escape_char (char character)

  Escapes difficult character that might break something later.
- char unescape_char (std::string character)

  Unescapes characters and prints them in a normal form.

Variables

- auto lowest_frequency

5.10.1 Detailed Description

Implements functions defined in huffman.h, contains the core functionality of this program.

Author

Michał Czyż

5.10.2 Function Documentation

5.10.2.1 clear_tree()

```
void clear_tree (
            huffman_node * node )
```

Clears a heap of a huffman tree so the program doesn't leak any data.

Parameters

node	A pointer to a node that has to be cleared. By default it should be the entire tree.

5.10.2.2 compress()

```
void compress (
            huffman_node * node,
            std::string data,
            std::map< char, std::string > & frequency )
```

Compresses data recursively to a binary stream.

Parameters

node	A pointer to the huffman tree root.
data	A string of data to be compressed.
frequency	A reference to a frequency map.

5.10.2.3 create_tree()

```
huffman_node * create_tree (
            std::map< char, int > & frequency )
```

Creates a huffman binary tree out of a map of frequencies.

Parameters

frequency	A reference to a map of frequencies.

Returns

Returns a pointer to a huffman binary tree.

5.10.2.4 decompress()

```
void decompress (
            std::map< std::string, char > & frequency,
            std::string & data,
            std::string & result )
```

Decompresses binary stream back to the normal format.

Parameters

frequency	A reference to an inverted frequency map.
data	A reference to the string of encoded data.
result	A reference to the string that will contain decomressed data.

5.10.2.5 escape_char()

```
std::string escape_char (
            char character )
```

Escapes difficult character that might break something later.

Parameters

character	A char that holds a normal character to escape i.e '\n'.

Returns

Returns an escaped version of a character.

5.10.2.6 is_leaf()

```
bool is_leaf (
            huffman_node * node )
```

Checks if the provided node has no children. It it the leaf of a tree.

Parameters

node	A pointer to the node that has to be checked.

Returns

Returns true if the node is a leaf node.

5.10.2.7 rebuild_tree()

```
std::map< std::string, char > rebuild_tree (
            std::string & dictionary )
```

Rebuilds an inverted (easier to process later) frequency map from a string.

Parameters

dictionary	A reference to a string that holds a map.

Returns

Returns an inverted frequency map.

5.10.2.8 unescape_char()

```
char unescape_char (
            std::string character )
```

Unescapes characters and prints them in a normal form.

Parameters

character	A string holding an encoded character to escape i.e '\n'.

Returns

Returns a raw, unescaped character.

5.10.3 Variable Documentation

5.10.3.1 lowest_frequency

```
auto lowest_frequency
```

Initial value:

```
= [](huffman_node *left, huffman_node *right)
{
  return left->frequency > right->frequency;
}
```

5.11 huffman.h File Reference

Defines core functions of the program.

```
#include <iostream>
#include <map>
#include <string>
```

Classes

- struct huffman_node

  huffman node structure used later to create a tree of frequencies.

Functions

- bool is_leaf (huffman_node ∗node)

  Checks if the provided node has no children. It it the leaf of a tree.
- void compress (huffman_node ∗node, std::string data, std::map< char, std::string > &frequency)

  Compresses data recursively to a binary stream.
- void decompress (std::map< std::string, char > &frequency, std::string &data, std::string &result)

  Decompresses binary stream back to the normal format.
- huffman_node ∗ create_tree (std::map< char, int > &frequency)

  Creates a huffman binary tree out of a map of frequencies.
- std::map< std::string, char > rebuild_tree (std::string &dictionary)

  Rebuilds an inverted (easier to process later) frequency map from a string.
- void clear_tree (huffman_node ∗node)

  Clears a heap of a huffman tree so the program doesn't leak any data.
- std::string escape_char (char character)

  Escapes difficult character that might break something later.
- char unescape_char (std::string character)

  Unescapes characters and prints them in a normal form.

5.11.1 Detailed Description

Defines core functions of the program.

Author

Michał Czyż

5.11.2 Function Documentation

5.11.2.1 clear_tree()

```
void clear_tree (
            huffman_node * node )
```

Clears a heap of a huffman tree so the program doesn't leak any data.

Parameters

node	A pointer to a node that has to be cleared. By default it should be the entire tree.

5.11.2.2 compress()

```
void compress (
            huffman_node * node,
            std::string data,
            std::map< char, std::string > & frequency )
```

Compresses data recursively to a binary stream.

Parameters

node	A pointer to the huffman tree root.
data	A string of data to be compressed.
frequency	A reference to a frequency map.

5.11.2.3 create_tree()

```
huffman_node * create_tree (
            std::map< char, int > & frequency )
```

Creates a huffman binary tree out of a map of frequencies.

Parameters

frequency	A reference to a map of frequencies.

Returns

Returns a pointer to a huffman binary tree.

5.11.2.4 decompress()

```
void decompress (
            std::map< std::string, char > & frequency,
            std::string & data,
            std::string & result )
```

Decompresses binary stream back to the normal format.

Parameters

frequency	A reference to an inverted frequency map.
data	A reference to the string of encoded data.
result	A reference to the string that will contain decomressed data.

5.11.2.5 escape_char()

```
std::string escape_char (
            char character )
```

Escapes difficult character that might break something later.

Parameters

character	A char that holds a normal character to escape i.e '\n'.

Returns

Returns an escaped version of a character.

5.11.2.6 is_leaf()

```
bool is_leaf (
            huffman_node * node )
```

Checks if the provided node has no children. It it the leaf of a tree.

Parameters

node	A pointer to the node that has to be checked.

Returns

Returns true if the node is a leaf node.

5.11.2.7 rebuild_tree()

```
std::map< std::string, char > rebuild_tree (
            std::string & dictionary )
```

Rebuilds an inverted (easier to process later) frequency map from a string.

Parameters

dictionary	A reference to a string that holds a map.

Returns

Returns an inverted frequency map.

5.11.2.8 unescape_char()

```
char unescape_char (
            std::string character )
```

Unescapes characters and prints them in a normal form.

Parameters

character	A string holding an encoded character to escape i.e '\n'.

Returns

Returns a raw, unescaped character.

5.12 huffman.h

Go to the documentation of this file.
```
00001
```

```
#ifndef HUFFMAN_H
#define HUFFMAN_H

#include <iostream>
#include <map>
#include <string>

struct huffman_node
{
  char symbol;
  unsigned int frequency;
  huffman_node *left, *right;
};

bool is_leaf(huffman_node *node);

void compress(huffman_node *node, std::string data, std::map<char, std::string> &frequency);

void decompress(std::map<std::string, char> &frequency, std::string &data, std::string &result);

huffman_node* create_tree(std::map<char, int> &frequency);

std::map<std::string, char> rebuild_tree(std::string &dictionary);

void clear_tree(huffman_node *node);

std::string escape_char(char character);

char unescape_char(std::string character);

#endif
```

5.13 main.cpp File Reference

Main program file.

```
#include <iostream>
#include <string>
#include <map>
#include <fstream>
#include <sstream>
#include "args_handler.h"
#include "file_handler.h"
#include "error_handler.h"
#include "huffman.h"
```

Functions

- int **main** (int argc, char **argv)

5.13.1 Detailed Description

Main program file.

Author

Michał Czyż

Index