My Project

Generated by Doxygen 1.9.8

Chapter 1

Test List

Member TEST (DetailInfoTest, EncodeFunctionWorks)

DetailInfoTest.EncodeFunctionWorks

Member TEST (DetailInfoTest, EncodeWithParametersWorks)

DetailInfoTest.EncodeWithParametersWorks

Member TEST (DetailInfoTest, DecodeFunctionWorks)

DetailInfoTest.DecodeFunctionWorks

Member TEST (DetailInfoTest, DecodeFunctionWorks2)

DetailInfoTest.DecodeFunctionWorks2

Member TEST (DetailInfoTest, DecodeFunctionWorks3)

DetailInfoTest.DecodeFunctionWorks3

Member TEST (DetailInfoTest, DecodeFunctionInvalidInput)

DetailInfoTest.DecodeFunctionInvalidInput

Member TEST (DetailInfoTest, DecodeFunctionInvalidInput2)

DetailInfoTest.DecodeFunctionInvalidInput2

Member TEST (DetailInfoTest, DecodeFunctionInvalidInput3)

DetailInfoTest.DecodeFunctionInvalidInput3

Member TEST (DetailInfoTest, DefaultConstructorWorks)

DetailInfoTest.DefaultConstructorWorks

Member TEST (DetailInfoTest, ParameterizedConstructorWorks)

DetailInfoTest.ParameterizedConstructorWorks

Member TEST (DetailInfoTest, ConstructorWithEncodedStringWorks)

DetailInfoTest.ConstructorWithEncodedStringWorks

Member TEST (DetailInfoTest, PrintFunctionWorks)

DetailInfoTest.PrintFunctionWorks

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 Detail_info Class Reference

Class representing detailed information with encoding and decoding capabilities.

`#include <detail.hpp>`

Public Member Functions

- string encode ()
  Encodes the detail into a JSON-like string.
- string encode (const string &id, const string &name, std::size_t count)
  Encodes the provided detail information into a JSON-like string.
- void decode (const string &str)
  Decodes a JSON-like string and extracts the detail information.
- void decode (const char *str)
  Decodes a C-style string and extracts the detail information.
- void decode (const char *str, std::size_t size)
  Decodes a C-style string of specified size and extracts the detail information.
- void **print** ()
  Prints the detail information to the standard output.
- Detail_info (const string &id, const string &name, std::size_t count)
  Constructs a Detail_info object with the provided values.
- Detail_info (const string &str)
  Constructs a Detail_info object by decoding the provided string.
- **Detail_info** ()
  Default constructor for Detail_info. Initializes with empty values.

4.1.1 Detailed Description

Class representing detailed information with encoding and decoding capabilities.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Detail_info() [1/2]

```
Detail_info::Detail_info (
            const string & id,
            const string & name,
            std::size_t count )
```

Constructs a Detail_info object with the provided values.

Parameters

id	The ID of the detail.
name	The name of the detail.
count	The count of the detail.

4.1.2.2 Detail_info() [2/2]

```
Detail_info::Detail_info (
            const string & str )
```

Constructs a Detail_info object by decoding the provided string.

Parameters

str	A string in the format {'id':'<id>', 'name':'<name>', 'count':<count>}.

4.1.3 Member Function Documentation

4.1.3.1 decode() [1/3]

```
void Detail_info::decode (
            const char * str )
```

Decodes a C-style string and extracts the detail information.

Parameters

str	A C-style string containing the detail information.

4.1.3.2 decode() [2/3]

```
void Detail_info::decode (
            const char * str,
            std::size_t size )
```

Decodes a C-style string of specified size and extracts the detail information.

Parameters

str	A C-style string containing the detail information.
size	The size of the string.

4.1.3.3 decode() [3/3]

```
void Detail_info::decode (
            const string & str )
```

Decodes a JSON-like string and extracts the detail information.

Parameters

str	A string in the format {'id':'<id>', 'name':'<name>', 'count':<count>}.

Exceptions

errors::BAD_JSON	if the string is not in the expected format.

4.1.3.4 encode() [1/2]

```
string Detail_info::encode ( )
```

Encodes the detail into a JSON-like string.

Encodes the detail information into a JSON-like string.

Returns

A string in the format {'id':'<id>', 'name':'<name>', 'count':<count>}.

4.1.3.5 encode() [2/2]

```
string Detail_info::encode (
            const string & id,
            const string & name,
            std::size_t count )
```

Encodes the provided detail information into a JSON-like string.

Parameters

id	The ID of the detail.
name	The name of the detail.
count	The count of the detail.

Returns

A string in the format {'id':'<id>', 'name':'<name>', 'count':<count>}.

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

- /home/farguss/files/sem3/oop/1/detail.hpp
- /home/farguss/files/sem3/oop/1/detail.cpp

Chapter 5

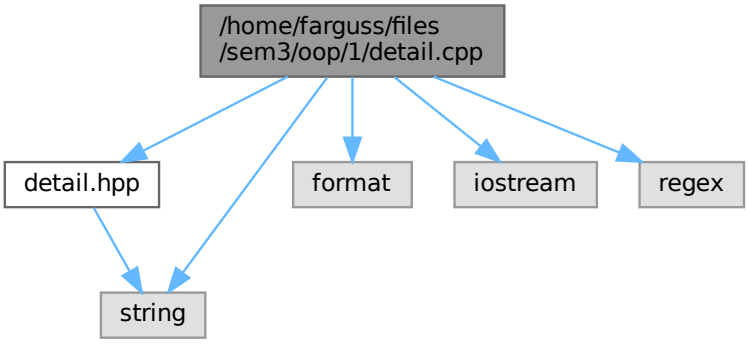
File Documentation

5.1 /home/farguss/files/sem3/oop/1/detail.cpp File Reference

Implementation of the Detail_info class and its associated methods.

```
#include "detail.hpp"
#include <format>
#include <iostream>
#include <regex>
#include <string>
```

Include dependency graph for detail.cpp:



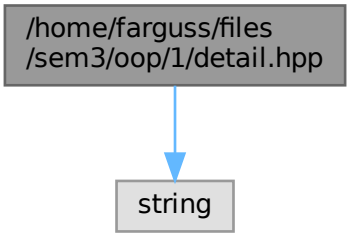
5.1.1 Detailed Description

Implementation of the Detail_info class and its associated methods.

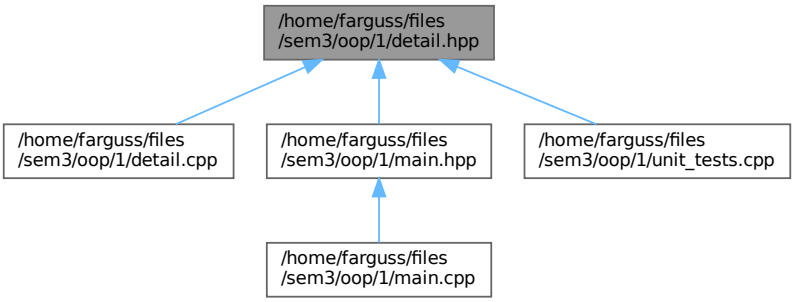
5.2 /home/farguss/files/sem3/oop/1/detail.hpp File Reference

Header file for the Detail_info class and associated functions.

`#include <string>`
Include dependency graph for detail.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class Detail_info

  Class representing detailed information with encoding and decoding capabilities.

Typedefs

- typedef enum _errors **errors**

  Error codes for Detail_info.

Enumerations

- enum _errors { BAD_JSON }

  Error codes for Detail_info.

5.2.1 Detailed Description

Header file for the Detail_info class and associated functions.

5.2.2 Enumeration Type Documentation

5.2.2.1 _errors

```
enum _errors
```

Error codes for Detail_info.

Enumerator

BAD_JSON	Error thrown when decoding a malformed JSON string.

5.3 /home/farguss/files/sem3/oop/1/detail.hpp

Go to the documentation of this file.

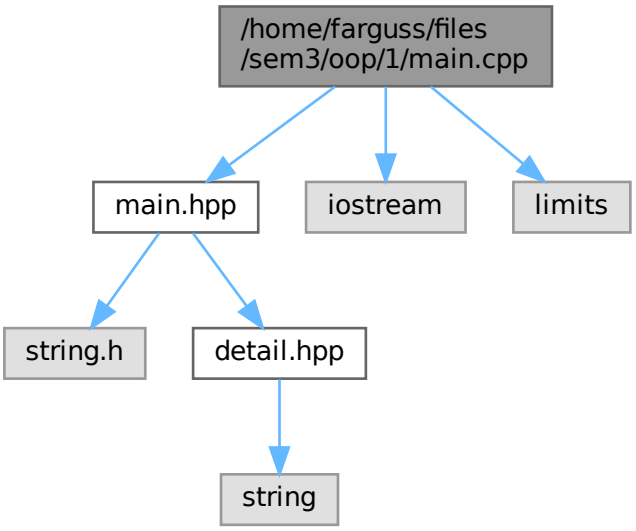
```
00001
00006 #ifndef LAB1_DETAIL_HPP
00007 #define LAB1_DETAIL_HPP
00008
00009 #include <string>
00010
00014 typedef enum _errors {
00015     BAD_JSON
00016 } errors;
00017
00018 using std::string;
00019
00024 class Detail_info {
00025   private:
00026     string id;
00027     string name;
00028     std::size_t count;
00029
00036     void init(const string& id, const string& name, std::size_t count);
00037
00038   public:
00043     string encode();
00044
00052     string encode(const string& id, const string& name, std::size_t count);
00053
00059     void decode(const string& str);
00060
00065     void decode(const char* str);
00066
00072     void decode(const char* str, std::size_t size);
00073
00077     void print();
00078
00085     Detail_info(const string& id, const string& name, std::size_t count);
00086
00091     Detail_info(const string& str);
00092
00096     Detail_info();
00097 };
00098
00099 #endif // LAB1_DETAIL_HPP
```

5.4 /home/farguss/files/sem3/oop/1/main.cpp File Reference

Main entry point for the Detail_info encoding and decoding application.

```
#include "main.hpp"
#include <iostream>
#include <limits>
```

Include dependency graph for main.cpp:



Macros

- #define **PROMPT** "(d) - decode\n(e) - encode\n"

Functions

- int main ()

  Main function. Provides a prompt to either encode or decode a Detail_info object.
- void decode (Detail_info &detail)

  Decodes the input string into a Detail_info object and prints the result.
- void encode (Detail_info &detail)

  Encodes user input into a JSON-like string and prints it.
- string get_str (const char *prompt)

  Prompts the user to input a string value.
- template<typename T >
  T get_num (const char *prompt, T min, T max)

  Prompts the user to input a numeric value within a specified range.

5.4.1 Detailed Description

Main entry point for the Detail_info encoding and decoding application.

5.4.2 Function Documentation

5.4.2.1 decode()

```
void decode (
            Detail_info & detail )
```

Decodes the input string into a Detail_info object and prints the result.

Parameters

detail	A reference to the Detail_info object to be populated.

Exceptions

std::runtime_error	if input fails or decoding is unsuccessful.

5.4.2.2 encode()

```
void encode (
            Detail_info & detail )
```

Encodes user input into a JSON-like string and prints it.

Parameters

detail	A reference to the Detail_info object to be encoded.

Exceptions

std::runtime_error	if input fails or encoding is unsuccessful.

5.4.2.3 get_num()

```
template<typename T >
T get_num (
            const char * prompt,
            T min,
            T max )
```

Prompts the user to input a numeric value within a specified range.

Template Parameters

T	The type of the numeric value (e.g., int, float, std::size_t).

Parameters

prompt	The prompt to be displayed to the user.
min	The minimum acceptable value.
max	The maximum acceptable value.

Returns

The user input as a numeric value of type T.

Exceptions

std::runtime_error	if the input fails or the value is out of range.

5.4.2.4 get_str()

```
string get_str (
            const char * prompt )
```

Prompts the user to input a string value.

Parameters

prompt	The prompt to be displayed to the user.

Returns

The user input as a string.

Exceptions

std::runtime_error	if the input fails (EOF or error).

5.4.2.5 main()

```
int main ( )
```

Main function. Provides a prompt to either encode or decode a Detail_info object.

Returns

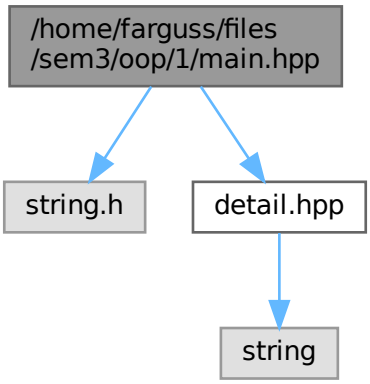
Returns 0 on success, 1 on failure.

5.5 /home/farguss/files/sem3/oop/1/main.hpp File Reference

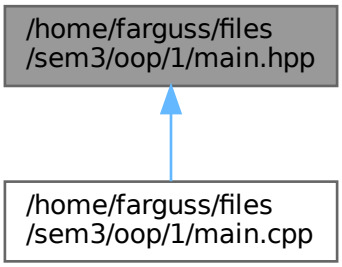
Header file for the main application functions.

```
#include <string.h>
#include "detail.hpp"
```

Include dependency graph for main.hpp:



This graph shows which files directly or indirectly include this file:



Functions

- string get_str (const char *prompt)

  Prompts the user to input a string value.

- template<typename T >
  T get_num (const char *prompt, T min=std::numeric_limits< T >::lowest(), T max=std::numeric_limits< T >::max())

  Prompts the user to input a numeric value within a specified range.

- void decode (Detail_info &detail)

  Decodes the input string into a Detail_info object and prints the result.
- void encode (Detail_info &detail)

  Encodes user input into a JSON-like string and prints it.

5.5.1 Detailed Description

Header file for the main application functions.

5.5.2 Function Documentation

5.5.2.1 decode()

```
void decode (
            Detail_info & detail )
```

Decodes the input string into a Detail_info object and prints the result.

Parameters

detail	A reference to the Detail_info object to be populated.

Exceptions

std::runtime_error	if input fails or decoding is unsuccessful.

5.5.2.2 encode()

```
void encode (
            Detail_info & detail )
```

Encodes user input into a JSON-like string and prints it.

Parameters

detail	A reference to the Detail_info object to be encoded.

Exceptions

std::runtime_error	if input fails or encoding is unsuccessful.

5.5.2.3 get_num()

```
template<typename T >
T get_num (
```

```
            const char * prompt,
            T min,
            T max )
```

Prompts the user to input a numeric value within a specified range.

Template Parameters

T	The type of the numeric value (e.g., int, float, std::size_t).

Parameters

prompt	The prompt to be displayed to the user.
min	The minimum acceptable value (default is lowest possible value).
max	The maximum acceptable value (default is highest possible value).

Returns

The user input as a numeric value of type T.

Exceptions

std::runtime_error	if the input fails or the value is out of range.

Template Parameters

T	The type of the numeric value (e.g., int, float, std::size_t).

Parameters

prompt	The prompt to be displayed to the user.
min	The minimum acceptable value.
max	The maximum acceptable value.

Returns

The user input as a numeric value of type T.

Exceptions

std::runtime_error	if the input fails or the value is out of range.

5.5.2.4 get_str()

```
string get_str (
            const char * prompt )
```

Prompts the user to input a string value.

Parameters

prompt	The prompt to be displayed to the user.

Returns

The user input as a string.

Exceptions

std::runtime_error	if the input fails (EOF or error).

5.6 /home/farguss/files/sem3/oop/1/main.hpp

Go to the documentation of this file.

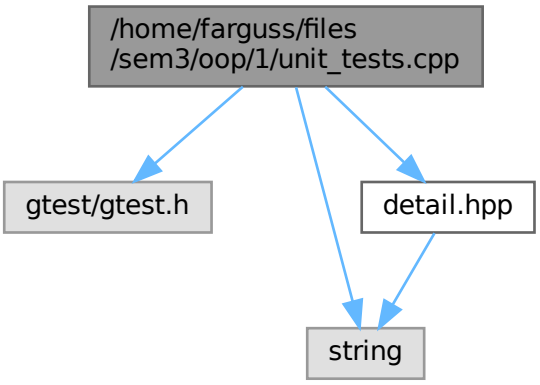
```
00001
00006 #ifndef LAB1_MAIN_HPP
00007 #define LAB1_MAIN_HPP
00008
00009 #include <string.h>
00010 #include "detail.hpp"
00011
00018 string get_str(const char* prompt);
00019
00029 template <typename T>
00030 T get_num(const char* prompt, T min = std::numeric_limits<T>::lowest(), T max =
      std::numeric_limits<T>::max());
00031
00037 void decode(Detail_info& detail);
00038
00044 void encode(Detail_info& detail);
00045
00046 #endif // LAB1_MAIN_HPP
```

5.7 /home/farguss/files/sem3/oop/1/unit_tests.cpp File Reference

Unit tests for the Detail_info class using Google Test framework.

```
#include <gtest/gtest.h>
#include <string>
#include "detail.hpp"
```

Include dependency graph for unit_tests.cpp:



Functions

- TEST (DetailInfoTest, EncodeFunctionWorks)

  Tests the encode() function without parameters.
- TEST (DetailInfoTest, EncodeWithParametersWorks)

  Tests the encode() function with parameters.
- TEST (DetailInfoTest, DecodeFunctionWorks)

  Tests the decode() function with valid input std::string.
- TEST (DetailInfoTest, DecodeFunctionWorks2)

  Tests the decode() function with valid input const char∗.
- TEST (DetailInfoTest, DecodeFunctionWorks3)

  Tests the decode() function with valid input const char∗ and size.
- TEST (DetailInfoTest, DecodeFunctionInvalidInput)

  Tests the decode() function with invalid input std::string.
- TEST (DetailInfoTest, DecodeFunctionInvalidInput2)

  Tests the decode() function with invalid input const char∗.
- TEST (DetailInfoTest, DecodeFunctionInvalidInput3)

  Tests the decode() function with invalid input const char∗ and size.
- TEST (DetailInfoTest, DefaultConstructorWorks)

  Tests the default constructor.
- TEST (DetailInfoTest, ParameterizedConstructorWorks)

  Tests the constructor with parameters.
- TEST (DetailInfoTest, ConstructorWithEncodedStringWorks)

  Tests the constructor with encoded string input.
- TEST (DetailInfoTest, PrintFunctionWorks)

  Tests the print() function.

5.7.1 Detailed Description

Unit tests for the Detail_info class using Google Test framework.

5.7.2 Function Documentation

5.7.2.1 TEST() [1/12]

```
TEST (
            DetailInfoTest ,
            ConstructorWithEncodedStringWorks  )
```

Tests the constructor with encoded string input.

Test DetailInfoTest.ConstructorWithEncodedStringWorks

5.7.2.2 TEST() [2/12]

```
TEST (
            DetailInfoTest ,
            DecodeFunctionInvalidInput  )
```

Tests the decode() function with invalid input std::string.

Test DetailInfoTest.DecodeFunctionInvalidInput

Exceptions

errors	

5.7.2.3 TEST() [3/12]

```
TEST (
            DetailInfoTest ,
            DecodeFunctionInvalidInput2  )
```

Tests the decode() function with invalid input const char∗.

Test DetailInfoTest.DecodeFunctionInvalidInput2

Exceptions

errors	

5.7.2.4 TEST() [4/12]

```
TEST (
            DetailInfoTest ,
            DecodeFunctionInvalidInput3  )
```

Tests the decode() function with invalid input const char* and size.

Test DetailInfoTest.DecodeFunctionInvalidInput3

Exceptions

errors	

5.7.2.5 TEST() [5/12]

```
TEST (
            DetailInfoTest ,
            DecodeFunctionWorks  )
```

Tests the decode() function with valid input std::string.

Test DetailInfoTest.DecodeFunctionWorks

5.7.2.6 TEST() [6/12]

```
TEST (
            DetailInfoTest ,
            DecodeFunctionWorks2  )
```

Tests the decode() function with valid input const char*.

Test DetailInfoTest.DecodeFunctionWorks2

5.7.2.7 TEST() [7/12]

```
TEST (
            DetailInfoTest ,
            DecodeFunctionWorks3  )
```

Tests the decode() function with valid input const char* and size.

Test DetailInfoTest.DecodeFunctionWorks3

5.7.2.8 TEST() [8/12]

```
TEST (
            DetailInfoTest ,
            DefaultConstructorWorks  )
```

Tests the default constructor.

Test DetailInfoTest.DefaultConstructorWorks

5.7.2.9 TEST() [9/12]

```
TEST (
            DetailInfoTest ,
            EncodeFunctionWorks  )
```

Tests the encode() function without parameters.

Test DetailInfoTest.EncodeFunctionWorks

5.7.2.10 TEST() [10/12]

```
TEST (
            DetailInfoTest ,
            EncodeWithParametersWorks  )
```

Tests the encode() function with parameters.

Test DetailInfoTest.EncodeWithParametersWorks

5.7.2.11 TEST() [11/12]

```
TEST (
            DetailInfoTest ,
            ParameterizedConstructorWorks  )
```

Tests the constructor with parameters.

Test DetailInfoTest.ParameterizedConstructorWorks

5.7.2.12 TEST() [12/12]

```
TEST (
            DetailInfoTest ,
            PrintFunctionWorks  )
```

Tests the print() function.

Test DetailInfoTest.PrintFunctionWorks

Index