Alexis Rico and Jordi Romero

P001

Generated by Doxygen 1.8.8

Thu Apr 28 2016 11:17:47

Contents

Chapter 1

Gestor de textos i cites | Alexis Rico and Jordi Romero

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 files with brief descriptions:

Chapter 4

Class Documentation

4.1 Author Class Reference

Data model for Authors (autors) from Library.

Public Member Functions

- Author ()

  Creates an empty Author.
- Author (string name)

  Creates an empty Author.
- ~Author ()

  Destructs the implicit Author.
- void incrementBookCount (int value)
- void incrementLineCount (int value)
- void incrementWordCount (int value)
- void addBook (string title)
- void addQuote ()
- void deleteBook (string title)
- void deleteQuote (string id)
- void printInformation ()

  Prints information about the Author.
- void printBooks ()

  Prints information about the Books.
- void printQuotes ()

  Prints information about the Quotes.

Private Attributes

- string authorName
- int totalBooks
- int totalLines
- int totalWords
- set< string > authorBooks
- set< string > authorQuotes

4.1.1 Detailed Description

Data model for Authors (autors) from Library.

Definition at line 21 of file Author.hh.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Author::Author ( )

Creates an empty Author.

Precondition

True

Postcondition

Returns an implicit empty author

4.1.2.2 Author::Author (string *name*)

Creates an empty Author.

Parameters

name	= Name of the Author

Precondition

True

Postcondition

Returns an implicit author with name

4.1.2.3 Author::~Author ( )

Destructs the implicit Author.

Precondition

An implicit Author

Postcondition

Deletes the implicit author

4.1.3 Member Function Documentation

4.1.3.1 void Author::incrementBookCount (int *value*)

4.1.3.2 void Author::incrementLineCount (int *value*)

4.1.3.3 void Author::incrementWordCount (int *value*)

4.1.3.4 void Author::addBook (string *title*)

4.1.3.5 void Author::addQuote ( )

4.1.3.6 void Author::deleteBook (string *title*)

4.1.3.7 void Author::deleteQuote (string *id*)

4.1.3.8 void Author::printInformation ( )

Prints information about the Author.

Precondition

An implicit Author

Postcondition

True

4.1.3.9 void Author::printBooks ( )

Prints information about the Books.

Precondition

An implicit Author

Postcondition

True

4.1.3.10 void Author::printQuotes ( )

Prints information about the Quotes.

Precondition

An implicit Author

Postcondition

True

4.1.4 Member Data Documentation

4.1.4.1 string Author::authorName [private]

Definition at line 25 of file Author.hh.

4.1.4.2 int Author::totalBooks [private]

Definition at line 28 of file Author.hh.

4.1.4.3 int Author::totalLines [private]

Definition at line 29 of file Author.hh.

4.1.4.4 int Author::totalWords [private]

Definition at line 30 of file Author.hh.

4.1.4.5 set<string> Author::authorBooks [private]

Definition at line 32 of file Author.hh.

4.1.4.6 set<string> Author::authorQuotes [private]

Definition at line 36 of file Author.hh.

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

- Author.hh

4.2 Book Class Reference

Data model for Books (textos) from Library.

Classes

- struct frequencyComparator

Public Member Functions

- Book ()
  Creates an empty Book.
- Book (string title, string author)
  Creates a Book with title and author.
- ~Book ()
  Destructs the implicit Book.
- void readBookContent ()
  Reads the content of the implicit Book.
- string getBookTitle () const
  Returns the title of the implicit Book.
- string getAuthorName ()
  Returns the name of the author of the implicit Book.
- int getBookLines ()
  Returns the number of lines of the implicit Book.
- int getBookWords ()
  Returns the number of words of the implicit Book.
- void replaceWords (string oldWord, string newWord)
  Replaces one word for another word in the implicit Book.
- bool findWord (string word)
  Finds if a word is on the content of the implicit Book.

- void generateFrequencyTable ()

  Generates the FrequencyTable Vector.
- void printInformation ()

  Prints the information of the implicit book.
- void printAllLines ()

  Prints all lines of the implicit Book, from 1 to bookContent.size()
- void printLines (string query)

  Prints the lines by logical expression match of the implicit Book.
- void printSelectLines (int start, int end)

  Prints lines from [start - 1] to [end - 1].
- void printFrequencyTable ()

  Prints all words of the content of the implicit book.

Public Attributes

- set< string > Book:getBookQuotes()

  Returns the book quotes.

Private Attributes

- string authorName
- string bookTitle
- vector< string > bookContent
- map< int, set< string > > wordDictionary
- map< string, int > wordFrequencyMap
- vector< pair< string, int > > wordFrequencyVector
- int bookWords
- set< string > bookQuotes

4.2.1 Detailed Description

Data model for Books (textos) from Library.

Definition at line 23 of file Book.hh.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Book::Book ( )

Creates an empty Book.

Precondition

True

Postcondition

Returns an implicit empty book

4.2.2.2 Book::Book (string *title*, string *author*)

Creates a Book with title and author.

Parameters

title	= Book title
author	= Book author

Precondition

True

Postcondition

Returns an implicit book with title and the author

4.2.2.3 Book::~Book ( )

Destructs the implicit Book.

Precondition

An implicit Book

Postcondition

Deletes the implicit book

4.2.3 Member Function Documentation

4.2.3.1 void Book::readBookContent ( )

Reads the content of the implicit Book.

Precondition

An implicit Book

Postcondition

The content of the implicit Book

4.2.3.2 string Book::getBookTitle ( ) const

Returns the title of the implicit Book.

Precondition

An implicit Book

Postcondition

The title of the implicit book

4.2.3.3 string Book::getAuthorName ( )

Returns the name of the author of the implicit Book.

Precondition

An implicit Book

Postcondition

The name of the author of the implicit book

4.2.3.4 int Book::getBookLines ( )

Returns the number of lines of the implicit Book.

Precondition

An implicit Book

Postcondition

The number of the lines of the implicit book

4.2.3.5 int Book::getBookWords ( )

Returns the number of words of the implicit Book.

Precondition

An implicit Book

Postcondition

The words of lines of the implicit book

4.2.3.6 void Book::replaceWords (string *oldWord*, string *newWord*)

Replaces one word for another word in the implicit Book.

Parameters

oldWord	= Word (old)
newWord	= Word (new)

Precondition

An implicit Book, the old word that we replaces and the new word

Postcondition

The implicit book with the old word replaced for the new word

4.2.3.7 bool Book::findWord (string *word*)

Finds if a word is on the content of the implicit Book.

Parameters

word	= Word to find on the book

Precondition

The wordDictionary of the implicit Book and a word that we want to find

Postcondition

Returns true if the word is on the content of the implicit book

4.2.3.8 void Book::generateFrequencyTable ( )

Generates the FrequencyTable Vector.

Precondition

An implicit Book

Postcondition

Generates the frequency table ordered

4.2.3.9 void Book::printInformation ( )

Prints the information of the implicit book.

Precondition

An implicit Book

Postcondition

Prints the title and author of the book

4.2.3.10 void Book::printAllLines ( )

Prints all lines of the implicit Book, from 1 to bookContent.size()

Precondition

An implicit Book

Postcondition

Prints all lines of the content of the implicit book with its number in increasingly ordered for the number

4.2.3.11 void Book::printLines (string *query*)

Prints the lines by logical expression match of the implicit Book.

Parameters

query	= Query to find the lines to print

Precondition

An implicit Book and logical expression match

Postcondition

Prints the number of the line and the line of the implicit book that keep the logical expression

4.2.3.12 void Book::printSelectLines (int *start*, int *end*)

Prints lines from [start - 1] to [end - 1].

Parameters

start	= Start line
end	= end line

Precondition

An implicit Book, and the range

Postcondition

Prints the lines of the range and its number of the implicit book

4.2.3.13 void Book::printFrequencyTable ( )

Prints all words of the content of the implicit book.

Precondition

An implicit Book

Postcondition

Prints all words of the content and its frequencies in decreasingly ordered by frequencies

4.2.4 Member Data Documentation

4.2.4.1 string Book::authorName [private]

Definition at line 27 of file Book.hh.

4.2.4.2 string Book::bookTitle [private]

Definition at line 29 of file Book.hh.

4.2.4.3 vector<string> Book::bookContent [private]

Definition at line 31 of file Book.hh.

4.2.4.4 map<int, set<string> > Book::wordDictionary [private]

Definition at line 36 of file Book.hh.

4.2.4.5 map<string, int> Book::wordFrequencyMap [private]

Definition at line 41 of file Book.hh.

4.2.4.6 vector<pair<string, int> > Book::wordFrequencyVector [private]

Definition at line 42 of file Book.hh.

4.2.4.7 int Book::bookWords [private]

Definition at line 45 of file Book.hh.

4.2.4.8 set<string> Book::bookQuotes [private]

Definition at line 47 of file Book.hh.

4.2.4.9 set<string> Book::Book

Returns the book quotes.

Precondition

An implicit Book

Postcondition

Returns the bookQuotes set

Definition at line 151 of file Book.hh.

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

- Book.hh

4.3 Book::frequencyComparator Struct Reference

Public Member Functions

- bool operator() (const pair< string, int > &a, const pair< string, int > &b)

4.3.1 Detailed Description

Definition at line 51 of file Book.hh.

4.3.2 Member Function Documentation

4.3.2.1 bool Book::frequencyComparator::operator() (const pair< string, int > & *a*, const pair< string, int > & *b*)

Definition at line 52 of file Book.hh.

```
52                                                                                  {
53             // Special case: Same frequency
54             if (a.second == b.second) {
55                 // Special case: Same length
56                 if (a.first.length() == b.first.length()) {
57                     // Make a lower case copy of the strings and compare them
58                     string aString = a.first;
59                     string bString = b.first;
60                     // Inv: They have the same length
61                     for (int i = 0; i < aString.length(); ++i) {
62                         aString[i] = tolower(aString[i]);
63                         bString[i] = tolower(bString[i]);
64                     }
65                     return aString < bString;
66                 }
67                 // Base case: Order by length in asc order
68                 return a.first.length() < b.first.length();
69             }
70             // Base case: Order by frequency in desc order
71             return a.second > b.second;
72         }
```

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

- Book.hh

4.4 Library Class Reference

Main storage of all the Books, Authors and Quotes.

Public Member Functions

- Library ()

  Creates an empty Library.
- ~Library ()

  Destructs the implicit Library.
- void readBook (string title, string authorName)

  Inserts a new book to the bookCollection.
- bool isBookSelected ()

  Returns whether we have a chosen book or not.
- void selectBook (string query)

  Selects a book from the collection.
- void deleteBook ()

  Deletes a book from the collection.
- void replaceWordsOnBook (string input)

  Replaces a word with anopther one in a book from the collection.
- void deleteQuote (string id)

  Deletes a quote from the library.
- Book getBook ()

  Returns the selected book.
- Quote getQuote (string id)

  Returns a quote from its ID.
- void printAuthors ()

Prints information about the Authors.

- void printBooks ()

  Prints information about the Books.

- void printQuotes ()

  Prints information about the Quotes.

- void printBooksByAuthor (string author)

  Prints information about the Books (by Author)

- void printQuotesByAuthor (string author)

  Prints information about the Quotes (by Author)

Private Attributes

- map< string, Book > bookCollection
- map< string, Author > authorCollection
- map< string, Quote > quoteCollection
- map< string, Book >::iterator currentBook

4.4.1 Detailed Description

Main storage of all the Books, Authors and Quotes.

Definition at line 25 of file Library.hh.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Library::Library ( )

Creates an empty Library.

Precondition

True

Postcondition

Returns an implicit empty library

4.4.2.2 Library::~Library ( )

Destructs the implicit Library.

Precondition

An implicit Library

Postcondition

Deletes the implicit library

4.4.3 Member Function Documentation

4.4.3.1 void Library::readBook (string *title*, string *authorName*)

Inserts a new book to the bookCollection.

Parameters

title	= The book title
author	= The book author

Precondition

An implicit Library

Postcondition

The implicit library with a new book (identified by title and author)

4.4.3.2 bool Library::isBookSelected ( )

Returns whether we have a chosen book or not.

Precondition

An implicit Library

Postcondition

True

4.4.3.3 void Library::selectBook (string *query*)

Selects a book from the collection.

Parameters

query	= Query to select a book

Precondition

An implicit Library

Postcondition

currentBook iterator has the desired book

4.4.3.4 void Library::deleteBook ( )

Deletes a book from the collection.

Precondition

An implicit Library and a selected book

Postcondition

The implicit library without the previous currentBook

4.4.3.5 void Library::replaceWordsOnBook (string *input*)

Replaces a word with anopther one in a book from the collection.

Parameters

input	= Pattern from the IO channel to replace words

Precondition

An implicit Library and a selected book

Postcondition

The implicit library with the replaced words on currentBook

4.4.3.6 void Library::deleteQuote (string *id*)

Deletes a quote from the library.

Parameters

id	= Quote ID

Precondition

An implicit Library and a quote with that id

Postcondition

The implicit library without the referenced quote

4.4.3.7 Book Library::getBook ( )

Returns the selected book.

Precondition

An implicit Library and a selected book

Postcondition

Returns currentBook

4.4.3.8 Quote Library::getQuote (string *id*)

Returns a quote from its ID.

Parameters

id	= Quote ID

Precondition

An implicit Library and a quote with that id

Postcondition

Returns the quote referenced by id

4.4.3.9 void Library::printAuthors ( )

Prints information about the Authors.

Precondition

An implicit Library

Postcondition

True

4.4.3.10 void Library::printBooks ( )

Prints information about the Books.

Precondition

An implicit Library

Postcondition

True

4.4.3.11 void Library::printQuotes ( )

Prints information about the Quotes.

Precondition

An implicit Library

Postcondition

True

4.4.3.12 void Library::printBooksByAuthor (string *author*)

Prints information about the Books (by Author)

Parameters

author	= Author name

Precondition

An implicit Library

Postcondition

True

4.4.3.13 void Library::printQuotesByAuthor (string *author*)

Prints information about the Quotes (by Author)

Parameters

author	= Author name

Precondition

An implicit Library

Postcondition

True

4.4.4 Member Data Documentation

4.4.4.1 map<string, Book> Library::bookCollection [private]

Definition at line 29 of file Library.hh.

4.4.4.2 map<string, Author> Library::authorCollection [private]

Definition at line 31 of file Library.hh.

4.4.4.3 map<string, Quote> Library::quoteCollection [private]

Definition at line 33 of file Library.hh.

4.4.4.4 map<string, Book>::iterator Library::currentBook [private]

Definition at line 35 of file Library.hh.

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

- Library.hh

4.5 Quote Class Reference

Data model for Quotes (cites) from Books.

Public Member Functions

- Quote ()
  Creates an empty Quote.
- Quote (string reference, string author, string book)
  Creates an empty Quote.
- ~Quote ()
  Destructs the implicit Quote.
- void printInformation ()
  Print all the information of the implicit Quote.
- void setQuoteLines (int start, int end)
  Update the start/end line values.
- void setContent (vector< string > content)
  Update the quote content.

Private Attributes

- string quoteReference
- string quoteAuthor
- string quoteBook
- int quoteStart
- int quoteEnd
- vector< string > quoteContent

4.5.1 Detailed Description

Data model for Quotes (cites) from Books.

Definition at line 21 of file Quote.hh.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Quote::Quote ( )

Creates an empty Quote.

Precondition

True

Postcondition

Returns an implicit empty quote

4.5.2.2 Quote::Quote (string *reference,* string *author,* string *book*)

Creates an empty Quote.

Parameters

reference	= The reference ID
author	= The Quote Author
book	= The book from the Quote

Precondition

True

Postcondition

Returns an implicit quote with the reference, author and book

4.5.2.3 Quote::~Quote ( )

Destructs the implicit Quote.

Precondition

An implicit Quote

Postcondition

Deletes the implicit quote

4.5.3 Member Function Documentation

4.5.3.1 void Quote::printInformation ( )

Print all the information of the implicit Quote.

Precondition

An implicit Quote

Postcondition

Prints all the quote information

4.5.3.2 void Quote::setQuoteLines (int *start*, int *end*)

Update the start/end line values.

Parameters

start	= Start line
end	= End line

Precondition

An implicit Quote

Postcondition

Updates the Quote start/end lines

4.5.3.3 void Quote::setContent (vector< string > *content*)

Update the quote content.

Parameters

content	= Content of the Quote

Precondition

An implicit Quote

Postcondition

Updates the Quote content

4.5.4 Member Data Documentation

4.5.4.1 string Quote::quoteReference [private]

Definition at line 24 of file Quote.hh.

4.5.4.2 string Quote::quoteAuthor [private]

Definition at line 26 of file Quote.hh.

4.5.4.3 string Quote::quoteBook [private]

Definition at line 28 of file Quote.hh.

4.5.4.4 int Quote::quoteStart [private]

Definition at line 30 of file Quote.hh.

4.5.4.5 int Quote::quoteEnd [private]

Definition at line 30 of file Quote.hh.

4.5.4.6 vector<string> Quote::quoteContent [private]

Definition at line 32 of file Quote.hh.

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

- Quote.hh

Chapter 5

File Documentation

5.1 Author.hh File Reference

Data model that hosts information about an Author.

Classes

- class Author

  Data model for Authors (autors) from Library.

5.1.1 Detailed Description

Data model that hosts information about an Author.

Definition in file Author.hh.

5.2 Book.hh File Reference

Data model that hosts information about a Book.

Classes

- class Book

  Data model for Books (textos) from Library.
- struct Book::frequencyComparator

5.2.1 Detailed Description

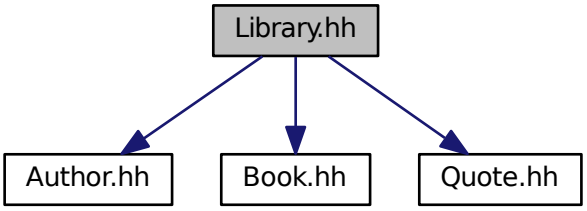
Data model that hosts information about a Book.

Definition in file Book.hh.

5.3 Library.hh File Reference

Main structure of our Library with several dependant data childs.

Include dependency graph for Library.hh:



Classes

- class Library

  Main storage of all the Books, Authors and Quotes.

5.3.1 Detailed Description

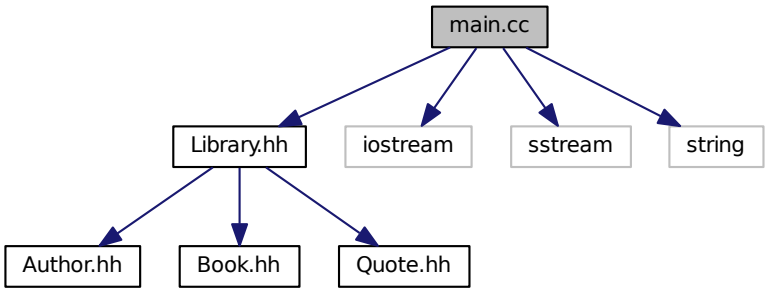
Main structure of our Library with several dependant data childs.

Definition in file Library.hh.

5.4 main.cc File Reference

Main program that handles the I/O operations with the end user.

Include dependency graph for main.cc:



Functions

- bool startsWith (string input, string compare)
- void readActions (Library &library)

- int main ()

  Main Procedure of main.cc.

Variables

- const string BOOK_INSERT = "afegir text"
- const string BOOK_DELETE = "eliminar text"
- const string BOOK_SELECT = "triar text"
- const string BOOK_REPLACE_WORD = "substitueix"
- const string QUOTE_INSERT = "afegir cita"
- const string QUOTE_DELETE = "eliminar cita"
- const string QUERY_AUTHORS = "tots autors ?"
- const string QUERY_BOOKS_ALL = "tots textos ?"
- const string QUERY_BOOKS_BY_AUTHOR = "textos autor"
- const string QUERY_CURRENT_AUTHOR = "autor ?"
- const string QUERY_CURRENT_CONTENT = "contingut ?"
- const string QUERY_CURRENT_INFO = "info ?"
- const string QUERY_CURRENT_EXPRESION = "frases"
- const string QUERY_CURRENT_LINES = "nombre de frases ?"
- const string QUERY_CURRENT_WORDS = "nombre de paraules ?"
- const string QUERY_CURRENT_FREQUENCY = "taula de frequencies ?"
- const string QUERY_CURRENT_QUOTES = "cites ?"
- const string QUERY_QUOTES_ALL = "totes cites ?"
- const string QUERY_QUOTES_BY_AUTHOR = "cites autor"
- const string QUERY_QUOTE_INFO = "info cita"
- const string QUIT = "sortir"

5.4.1 Detailed Description

Main program that handles the I/O operations with the end user.

Definition in file main.cc.

5.4.2 Function Documentation

5.4.2.1 bool startsWith (string *input*, string *compare*)

Definition at line 44 of file main.cc.

```
44                                                {
45     return input.substr(0, compare.length()) == compare;
46 }
```

5.4.2.2 void readActions (Library & *library*)

Definition at line 48 of file main.cc.

```
48                                    {
49     string input;
50     while (getline(cin, input)) {
51         cout << input << endl;
52         if (startsWith(input, BOOK_INSERT)) {
53             string title, author;
54             title = input.erase(input.length() - 1, 1).substr(input.find_first_of("\"") + 1);
55             getline(cin, input);
56             author = input.erase(input.length() - 1, 1).substr(input.find_first_of("\"") + 1);
57             library.readBook(title, author);
```

```
58          } else if (startsWith(input, BOOK_DELETE)) {
59              library.deleteBook();
60          } else if (startsWith(input, BOOK_SELECT)) {
61              library.selectBook(input.erase(input.length() - 1, 1).substr(input.find_first_of("{")
      + 1));
62          } else if (startsWith(input, BOOK_REPLACE_WORD)) {
63              library.replaceWordsOnBook(input);
64          } else if (startsWith(input, QUOTE_INSERT)) {
65              // TODO
66          } else if (startsWith(input, QUOTE_DELETE)) {
67              string reference;
68              // TODO: Substring reference from input
69              library.deleteQuote(reference);
70          } else if (startsWith(input, QUERY_AUTHORS)) {
71              library.printAuthors();
72          } else if (startsWith(input, QUERY_BOOKS_ALL)) {
73              library.printBooks();
74          } else if (startsWith(input, QUERY_BOOKS_BY_AUTHOR)) {
75        string authorName;
76              // TODO: Substring author
77        library.printBooksByAuthor(authorName);
78          } else if (startsWith(input, QUERY_CURRENT_AUTHOR)) {
79              if (library.isBookSelected())
80                  cout << library.getBook().getAuthorName() << endl;
81              else cout << "error" << endl;
82          } else if (startsWith(input, QUERY_CURRENT_CONTENT)) {
83              if (library.isBookSelected())
84                  library.getBook().printAllLines();
85              else cout << "error" << endl;
86          } else if (startsWith(input, QUERY_CURRENT_INFO)) {
87              // TODO
88          } else if (startsWith(input, QUERY_CURRENT_EXPRESION)) {
89              // TODO
90          } else if (startsWith(input, QUERY_CURRENT_LINES)) {
91              if (library.isBookSelected())
92                  cout << library.getBook().getBookLines() << endl;
93              else cout << "error" << endl;
94          } else if (startsWith(input, QUERY_CURRENT_WORDS)) {
95              if (library.isBookSelected())
96                  cout << library.getBook().getBookWords() << endl;
97              else cout << "error" << endl;
98          } else if (startsWith(input, QUERY_CURRENT_FREQUENCY)) {
99              if (library.isBookSelected())
100                 library.getBook().printFrequencyTable();
101             else cout << "error" << endl;
102         } else if (startsWith(input, QUERY_CURRENT_QUOTES)) {
103             // TODO
104         } else if (startsWith(input, QUERY_QUOTES_ALL)) {
105             library.printQuotes();
106         } else if (startsWith(input, QUERY_QUOTES_BY_AUTHOR)) {
107             string author;
108             // TODO: Substring author from input
109             library.printQuotesByAuthor(author);
110         } else if (startsWith(input, QUERY_QUOTE_INFO)) {
111             string reference;
112             // TODO: Substring reference from input
113             library.getQuote(reference).printQuoteInformation();
114         } else if (startsWith(input, QUIT)) {
115             return;
116         }
117     }
118 }
```

5.4.2.3 int main ( )

Main Procedure of main.cc.

Definition at line 121 of file main.cc.

```
121             {
122     Library mainLibrary;
123     readActions(mainLibrary);
124 }
```

5.4.3 Variable Documentation

5.4.3.1 const string BOOK_INSERT = "afegir text"

Definition at line 17 of file main.cc.

5.4.3.2 const string BOOK_DELETE = "eliminar text"

Definition at line 18 of file main.cc.

5.4.3.3 const string BOOK_SELECT = "triar text"

Definition at line 19 of file main.cc.

5.4.3.4 const string BOOK_REPLACE_WORD = "substitueix"

Definition at line 20 of file main.cc.

5.4.3.5 const string QUOTE_INSERT = "afegir cita"

Definition at line 22 of file main.cc.

5.4.3.6 const string QUOTE_DELETE = "eliminar cita"

Definition at line 23 of file main.cc.

5.4.3.7 const string QUERY_AUTHORS = "tots autors ?"

Definition at line 25 of file main.cc.

5.4.3.8 const string QUERY_BOOKS_ALL = "tots textos ?"

Definition at line 26 of file main.cc.

5.4.3.9 const string QUERY_BOOKS_BY_AUTHOR = "textos autor"

Definition at line 27 of file main.cc.

5.4.3.10 const string QUERY_CURRENT_AUTHOR = "autor ?"

Definition at line 29 of file main.cc.

5.4.3.11 const string QUERY_CURRENT_CONTENT = "contingut ?"

Definition at line 30 of file main.cc.

5.4.3.12 const string QUERY_CURRENT_INFO = "info ?"

Definition at line 31 of file main.cc.

5.4.3.13 const string QUERY_CURRENT_EXPRESION = "frases"

Definition at line 32 of file main.cc.

5.4.3.14 const string QUERY_CURRENT_LINES = "nombre de frases ?"

Definition at line 33 of file main.cc.

5.4.3.15 const string QUERY_CURRENT_WORDS = "nombre de paraules ?"

Definition at line 34 of file main.cc.

5.4.3.16 const string QUERY_CURRENT_FREQUENCY = "taula de frequencies ?"

Definition at line 35 of file main.cc.

5.4.3.17 const string QUERY_CURRENT_QUOTES = "cites ?"

Definition at line 36 of file main.cc.

5.4.3.18 const string QUERY_QUOTES_ALL = "totes cites ?"

Definition at line 38 of file main.cc.

5.4.3.19 const string QUERY_QUOTES_BY_AUTHOR = "cites autor"

Definition at line 39 of file main.cc.

5.4.3.20 const string QUERY_QUOTE_INFO = "info cita"

Definition at line 40 of file main.cc.

5.4.3.21 const string QUIT = "sortir"

Definition at line 42 of file main.cc.

5.5 Quote.hh File Reference

Data model that hosts information about a Quote.

Classes

- class Quote

  Data model for Quotes (cites) from Books.

5.5.1 Detailed Description

Data model that hosts information about a Quote.

Definition in file Quote.hh.

Index