CS124 Lab1 - Dictionary Program

Generated by Doxygen 1.8.8

Sat Sep 9 2017 05:49:52

Contents

1 Specification

This program has a built in dictionary for english that can be translated to tagalog. It has a user input in which the user may add the translation of a word that is not within the dictionary.

2 Analysis

As the user runs the program, it will ask the user to type in a word that he/she would like to translate into tagalog. As they type in their input of the word, it is then sent through loops which categorize if it is in the dictionary or not; if it is in the dictionary, it's translation is outputted to the user. However, if it is not in the dictionary, it will ask for the user input of translation for the word they want to add into the dictionary.

3 Design

The program will run an executable file called "lab" which is a bunch of files that are compiled into one runnable file. When "lab" is executed, it tells the user how many words and vectors are in the file called "dict.dat", which is the dictionary. The dictionary is loaded into the program and is asking the user which word would he/she would want to translate into tagalog. If the word is not in the dictionary, he/she is asked if they would like to add the word into the dictionary with the translation. If they do not, they may exit and continue more translations, but if they want to add a translation, they can add the word and then the translation into the dictionary which is 'dict.dat'

4 Test

```
English - Tagalog Dictionary
.....................................
 (8 word)

 (8 size of vector)

Enter a word or 'q' to quit ==> hello
kumusta

Enter a word or 'q' to quit ==> no
 -- not in the dictionary.

Would you like to add translation? (y/n) y
Enter the word you want to put in the dictionary
no
Enter the translation
hinde
Enter a word or 'q' to quit ==> no
hinde

Enter a word or 'q' to quit ==> q
debian@debian:~/lab1$
```

5 Class Index

5.1 Class List

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

6 File Index

6.1 File List

Here is a list of all files with brief descriptions:

7 Class Documentation

7.1 ENTRY Struct Reference

`#include <lab.h>`

Public Attributes

- string word
- string translation

7.1.1 Member Data Documentation

7.1.1.1 string ENTRY::translation

7.1.1.2 string ENTRY::word

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

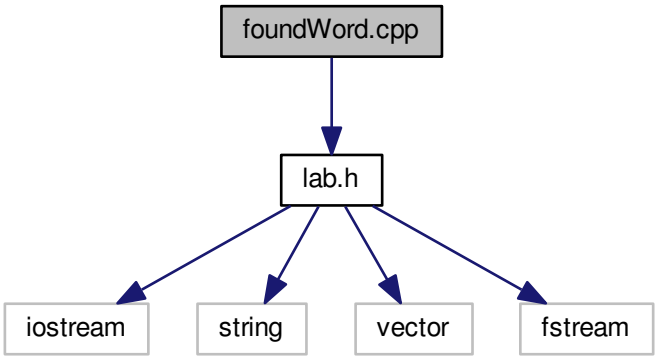
- lab.h

8 File Documentation

8.1 foundWord.cpp File Reference

```
#include "lab.h"
```

Include dependency graph for foundWord.cpp:



Functions

- bool foundWord (const vector< ENTRY > &dict, const string &word, string &translation)

8.1.1 Function Documentation

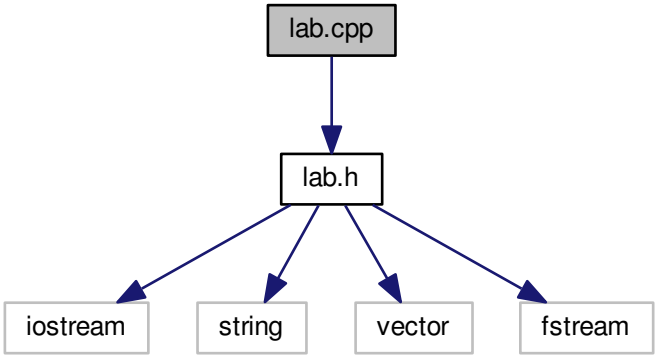
8.1.1.1 bool foundWord (const vector< ENTRY > & *dict*, const string & *word*, string & *translation*)

```
19 {
20     bool found = false; //if the word is not found, it will return false
21     //integer for the for loop and len is for the length of the vector
22     int i, len = dict.size();
23     /*for loop which is needed to find the word in dictionary
24      which in turn gives the translation of the word*/
25     for (i = 0; !found && i < len; i++) {
26         if (dict[i].word == word){
27             translation = dict[i].translation;
28             found = true;
29         }//end of if
30     } //end of for
31     return found; //returns the translation to user for it to be output
32 } //end of bool
```

8.2 lab.cpp File Reference

```
#include "lab.h"
```

Include dependency graph for lab.cpp:



Functions

- ostream & operator<< (ostream &o, ENTRY &e)
- istream & operator>> (istream &i, ENTRY &e)

8.2.1 Function Documentation

8.2.1.1 ostream& operator<< (ostream & *o,* ENTRY & *e*)

```
19 {
20     o << e.word << "\t";
21     o << e.translation << endl;
22     return o;
23 }
```

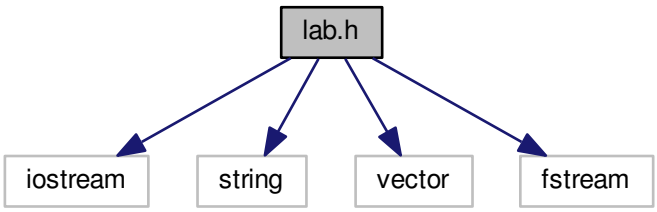
8.2.1.2 istream& operator>> (istream & *i,* ENTRY & *e*)

```
26 {
27     i >> e.word >> e.translation;
28 }
```

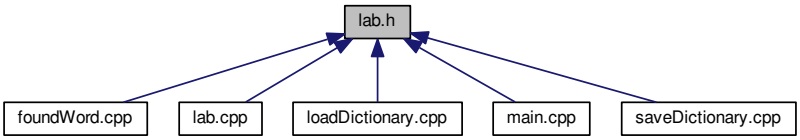
8.3 lab.h File Reference

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

Include dependency graph for lab.h:



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



Classes

- struct ENTRY

Functions

- bool loadDictionary (string fileName, vector< ENTRY > &dict)
- bool saveDictionary (string fileName, vector< ENTRY > &dict)
- ostream & operator<< (ostream &o, const ENTRY &e)
- istream & operator>> (istream &i, const ENTRY &e)
- bool foundWord (const vector< ENTRY > &dict, const string &word, string &stranslation)

8.3.1 Function Documentation

8.3.1.1 bool foundWord (const vector< ENTRY > & *dict,* const string & *word,* string & *stranslation*)

```
19 {
20     bool found = false; //if the word is not found, it will return false
21     //integer for the for loop and len is for the length of the vector
22     int i, len = dict.size();
23     /*for loop which is needed to find the word in dictionary
24      which in turn gives the translation of the word*/
25     for (i = 0; !found && i < len; i++) {
26         if (dict[i].word == word){
27             translation = dict[i].translation;
```

```
28             found = true;
29         }//end of if
30     } //end of for
31     return found; //returns the translation to user for it to be output
32 } //end of bool
```

8.3.1.2 bool loadDictionary (string *fileName,* vector< ENTRY > & *dict*)

```
20 {   //integer needed for counting how many words are in dictionary
21     int cnt = 0;
22     ENTRY e; //to access the vector ENTRY which contain word and translation
23     ifstream inpFile(fileName.c_str());
24     if (!inpFile) return false; //if file does not open, it will return false
25     string title; //string to contain title
26     getline(inpFile, title); //to get the title from the text file
27     cout << title << endl; //to print the title of the program
28     cout << "....................................." << endl;
29     while (inpFile >> e.word >> e.translation) {
30         dict.push_back(e);
31         inpFile.ignore(80, '\n'); //skip the rest of the line
32         cnt++;
33     }//end of while for inpFile
34
35     cout << " (" << cnt << " words)\n\n"; //prints out the number of words
36     //prints out the number of vectors used
37     cout << " (" << dict.size() << " size of vector)\n\n";
38
39     return true;
40 } //end of bool loadDictionary
```

8.3.1.3 ostream& operator<< (ostream & *o,* const ENTRY & *e*)

8.3.1.4 istream& operator>> (istream & *i,* const ENTRY & *e*)

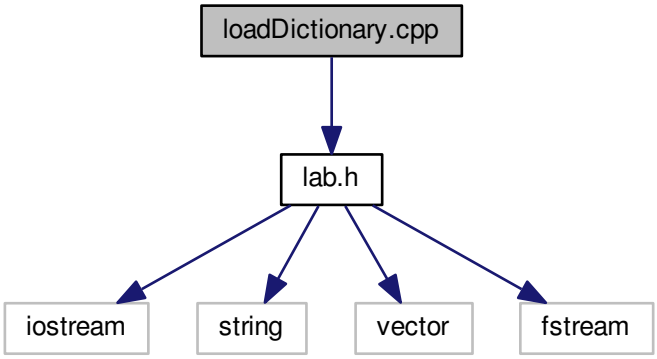
8.3.1.5 bool saveDictionary (string *fileName,* vector< ENTRY > & *dict*)

```
20 {
21     string input; //to get user's input
22     ENTRY e; //to access the vector ENTRY
23     fstream saveFile; //object to access text file 'dict.dat'
24     saveFile.open(fileName.c_str(), ios::app); //to open text file
25     //to tell the user the word is not in the dictionary
26     cout << e.word << " -- not in the dictionary.\n\n";
27     cout << "Would you like to add translation? (y/n) ";
28     cin >> input; //taking in user's input
29     /*if loop that if the user wants to add their word and translation
30       into the dictionary, they must type 'y', if not, anything else
31       to continue the code.
32     */
33             if(input == "y")
34                 {
35                     ENTRY e;
36                     cout << "Enter the word you want to put in the dictionary " << endl;
37                     cin >> e.word; //take in user input for word
38                     cout << "Enter the translation: " << endl;
39                     cin >> e.translation; //take in user input for translation
40                     dict.push_back(e); // enters user input to last element of vector
41                     //saves user's input into the text file
42                     saveFile << e.word << " " << e.translation << endl;
43
44                 }
45 }
```

8.4 loadDictionary.cpp File Reference

```
#include "lab.h"
```

Include dependency graph for loadDictionary.cpp:



Functions

- bool loadDictionary (string fileName, vector< ENTRY > &dict)

8.4.1 Function Documentation

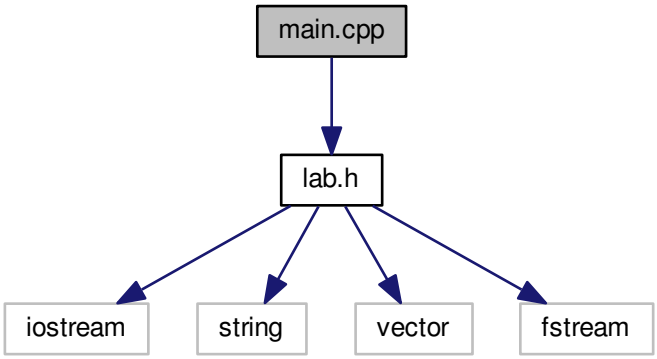
8.4.1.1 bool loadDictionary (string *fileName,* vector< ENTRY > & *dict*)

```
20 {   //integer needed for counting how many words are in dictionary
21     int cnt = 0;
22     ENTRY e; //to access the vector ENTRY which contain word and translation
23     ifstream inpFile(fileName.c_str());
24     if (!inpFile) return false; //if file does not open, it will return false
25     string title; //string to contain title
26     getline(inpFile, title); //to get the title from the text file
27     cout << title << endl; //to print the title of the program
28     cout << "....................................." << endl;
29     while (inpFile >> e.word >> e.translation) {
30         dict.push_back(e);
31         inpFile.ignore(80, '\n'); //skip the rest of the line
32         cnt++;
33     }//end of while for inpFile
34 
35     cout << " (" << cnt << " words)\n\n"; //prints out the number of words
36     //prints out the number of vectors used
37     cout << " (" << dict.size() << " size of vector)\n\n";
38 
39     return true;
40 } //end of bool loadDictionary
```

8.5 main.cpp File Reference

```
#include "lab.h"
```

Include dependency graph for main.cpp:



Functions

- int main ()

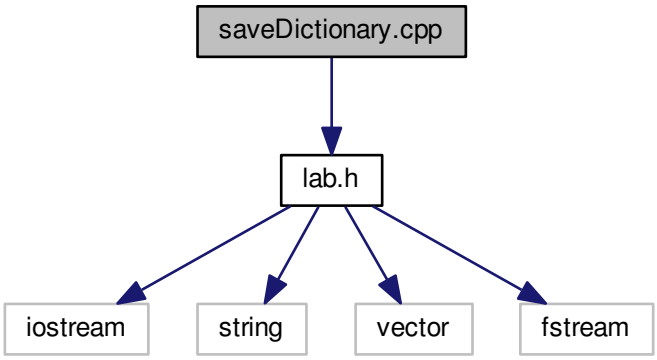
8.5.1 Function Documentation

8.5.1.1 int main ( )

```
18          {
19      vector<ENTRY> dict; //vector object to be able to use ENTRY
20      string word, translation; //strings needed for user input and output
21      bool ok, quit; //loads dictionary file
22
23      ok = loadDictionary("dict.dat", dict); //object that uses function loadDictionary
24      //if loop that is an error catcher to tell user the dictionary cannot be loaded
25      if (!ok){
26          cout << "*** Cannot load dictionary ***\n";
27          return 1;
28      }//end of if
29      quit = false;
30      while (!quit) {
31          cout << "Enter a word or 'q' to quit ==> ";
32          cin >> word;            //Read one word
33          cin.ignore(80, '\n');   //skip the rest of the line
34          //if loop that ends the program if user enters 'q'
35          if (word== "q")
36              quit = true;
37          /*enters user input into foundWord which prints out the
38           word's translation if it is in the dictionary*/
39          else if (foundWord(dict, word, translation))
40              cout << translation << "\n\n";
41          /*allows the user to enter a word and it's translation
42           in the dictionary if the word is not in the dictionary */
43          else{
44                  saveDictionary("dict.dat",dict);
45
46              } //end of else
47      }//end of while
48      return 0;
49
50  }// end main
```

8.6 saveDictionary.cpp File Reference

`#include "lab.h"`
Include dependency graph for saveDictionary.cpp:



Functions

- bool saveDictionary (string fileName, vector< ENTRY > &dict)

8.6.1 Function Documentation

8.6.1.1 bool saveDictionary (string *fileName,* vector< ENTRY > & *dict*)

```
20 {
21     string input; //to get user's input
22     ENTRY e; //to access the vector ENTRY
23     fstream saveFile; //object to access text file 'dict.dat'
24     saveFile.open(fileName.c_str(), ios::app); //to open text file
25     //to tell the user the word is not in the dictionary
26     cout << e.word << " -- not in the dictionary.\n\n";
27     cout << "Would you like to add translation? (y/n) ";
28     cin >> input; //taking in user's input
29     /*if loop that if the user wants to add their word and translation
30       into the dictionary, they must type 'y', if not, anything else
31       to continue the code.
32     */
33             if(input == "y")
34                 {
35                     ENTRY e;
36                     cout << "Enter the word you want to put in the dictionary " << endl;
37                     cin >> e.word; //take in user input for word
38                     cout << "Enter the translation: " << endl;
39                     cin >> e.translation; //take in user input for translation
40                     dict.push_back(e); // enters user input to last element of vector
41                     //saves user's input into the text file
42                     saveFile << e.word << " " << e.translation << endl;
43
44                 }
45 }
```

8.7 specification.dox File Reference