Lab 1 - Dictionary

Generated by Doxygen 1.8.8

Fri Sep 8 2017 14:00:59

Contents

1 Specification

This is the English-Italian Dictionary program. The user can input an English word and the program will provide translation of the word. The current file the program uses to translate is and English to Italian Dictionary If the word that user enters is not in the program the the user can add the word into the dictionary by following the instructions given by the program.

Freatures:

1) The instructures are clear and concise to understand.

2) The ability to add words into the dictionary/program for later use.

3) Operator overlaoding to improve effiecncy of program.

2 Analysis

When the program begins to run the first thing it will do is tell the user what the Dictionary is and then ask the user to input a word he/she wants translated. If the user enters a word in the dictionary the program will give the traslation and then ask if the user has another word that needs to be translated. If the user enters a word that is not in the dictionary the program will ask the user if he/she want to add it in. Typing "y" (the program asks this) will then prompt the program to ask the user the translated word so that it will then be added into the dictionary and be used again when needed. Typing "q" will quit the program.

3 Design

There are 7 parts of this lab which can be seen examined more closely later in the document. Each part has 1 specific function or purpose so as not to create confusion as to what file does what or where something is located. There will be a function to add the words back into the dictionary called addwords and there will be a selection control structure added into the main fucntion to allow the user to choose and add a new word into the dictionary. The pages that contain these functions provide more detail as to the algorithms and logic of how this is done.

The fundamental and most important design of this project is that it can and will work with another dictionary. The name of the dictionary has to be added into the program but other than that the code will work well with another file. This greatly improves portabilty and the simplicity of the code.

4 Test

This test shows that the program functions

```
English-Italian Dictionary
Program by Samuel Jothimuthu
Enter a word or 'q' to quit ==> love
amare

Enter a word or 'q' to quit ==> program
programma

Enter a word or 'q' to quit ==> car
auto

Enter a word or 'q' to quit ==> dictionary
dictionary --not in the dictionary.
 Would you like to add it? (y/n)
y
What is the Italian translation for dictionary?
dizionario
word added
Enter a word or 'q' to quit ==> dictionary
dizionario

Enter a word or 'q' to quit ==> q
```

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:

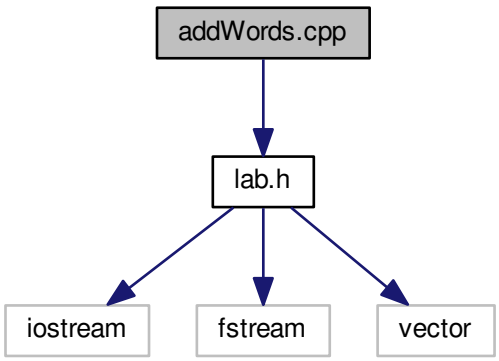
- lab.h

8 File Documentation

8.1 addWords.cpp File Reference

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

Include dependency graph for addWords.cpp:



Functions

- bool addwords (string filename, string inputstring)

8.1.1 Function Documentation

8.1.1.1 bool addwords (string *filename,* string *inputstring*)

This is the function that adds unknown words the user inputs back into the dictionary

Parameters

in	*filename*	the name of the file
in	*inputstring*	the new string created by the user and added to the file
out	*inputstring*	is now added into the dictionary

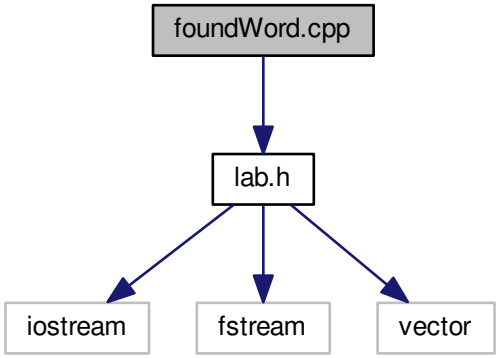
Returns

a boolean to check if the process was successful

```
11 {
12     //string test = "test"; //to test the fucntion worked
13     ofstream inpfile(filename, ios::app); //opens the file to write; to append the information to it
14     inpfile << inputstring << endl;
15     inpfile.close(); //closes the filestream
16     cout << "word added" <<endl;
17     return true;
18
19 }
```

8.2 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)

8.2.1 Function Documentation

8.2.1.1 bool foundword (const vector< Entry > & *dict*, const string & *word*, string & *translation*)

A boolean function that looks for the word in the the dictionary file

Parameters

in	*dict*	a reference to the vector Entries
in	*word*	a reference to the word in the vector
in	*translation*	a reference to the translation in the vector
out	*translation*	the translated word can then be output

Returns

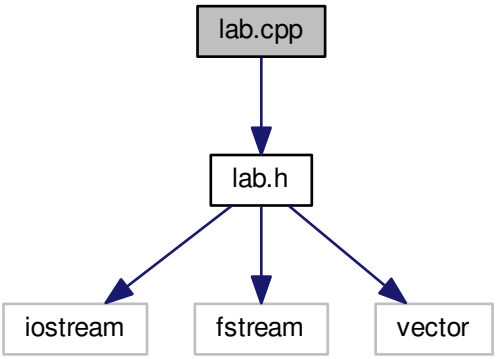
a boolean value if the function worked successfully

```
11 {
12     bool found = false;
13     int i, len = dict.size(); //length of the size of the vector
14 
15     for(i = 0; !found && i < len; i++)
16     {
17         if (dict[i].word == word) {
18             translation = dict[i].translation; //this sets the translation in the file to the translation
       variable allowing to ouput
19             found = true;
20         }
21     }
22     return found;
23 }
```

8.3 lab.cpp File Reference

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

Include dependency graph for lab.cpp:



Functions

- std::ostream & operator<< (std::ostream &o, const Entry &e)
- std::istream & operator>> (std::istream &i, Entry &e)

8.3.1 Function Documentation

8.3.1.1 std::ostream& operator<< (std::ostream & *o*, const Entry & *e*)

This file contains the code for the operator overload. the insertion ">>" and extraction "<<" operator is what is overloaded. You can see it the load dictionary function

```
7 {
8     o << e.word << "\t";
9     o << e.translation;
10     return o;
11 }
```

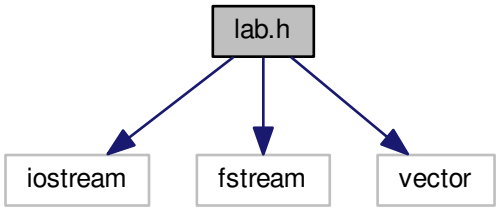
8.3.1.2 std::istream& operator>> (std::istream & *i*, Entry & *e*)

```
14 {
15     i >> e.word >> e.translation;
16     return i;
17
18 }
```

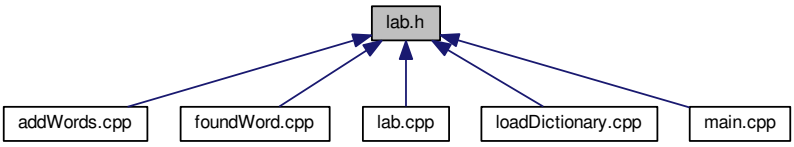
8.4 lab.h File Reference

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

Include dependency graph for lab.h:



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



Classes

- struct Entry

Functions

- std::ostream & operator<< (std::ostream &o, const Entry &e)
- std::istream & operator>> (std::istream &i, Entry &e)
- bool loaddictionary (string filename, vector< Entry > &dict)
- bool foundword (const vector< Entry > &dict, const string &word, string &translation)
- bool addwords (string filename, string inputstring)

8.4.1 Function Documentation

8.4.1.1 bool addwords (string *filename,* string *inputstring*)

This is the function that adds unknown words the user inputs back into the dictionary

Parameters

in	*filename*	the name of the file
in	*inputstring*	the new string created by the user and added to the file

out	*inputstring*	is now added into the dictionary

Returns

a boolean to check if the process was successful

```
11 {
12     //string test = "test"; //to test the fucntion worked
13     ofstream inpfile(filename, ios::app); //opens the file to write; to append the information to it
14     inpfile << inputstring << endl;
15     inpfile.close(); //closes the filestream
16     cout << "word added" <<endl;
17     return true;
18 
19 }
```

8.4.1.2 bool foundword (const vector< Entry > & *dict*, const string & *word*, string & *translation*)

A boolean function that looks for the word in the the dictionary file

Parameters

in	*dict*	a reference to the vector Entries
in	*word*	a reference to the word in the vector
in	*translation*	a reference to the translation in the vector
out	*translation*	the translated word can then be output

Returns

a boolean value if the function worked successfully

```
11 {
12     bool found = false;
13     int i, len = dict.size(); //length of the size of the vector
14 
15     for(i = 0; !found && i < len; i++)
16     {
17         if (dict[i].word == word) {
18             translation = dict[i].translation; //this sets the translation in the file to the translation
    variable allowing to ouput
19             found = true;
20         }
21     }
22     return found;
23 }
```

8.4.1.3 bool loaddictionary (string *filename*, vector< Entry > & *dict*)

Parameters

in	*the*	filename of the dictionary
in	*dict*	a reference to a vector of Entries

Returns

a boolean that indicates if the file was opened successfully

```
8 {
9 
10 
11     ifstream inpfile(filename.c_str()); //opening file
12     if (!inpfile) return false;
13 
14 
15     Entry e; // structure e declared
16     while (inpfile >> e) { //operator overloading done here
17         dict.push_back(e);
18         inpfile.ignore(80, '\n'); // skip the rest of the line
19 
20     }
```

```
21
22
23     return true;
24 }
```

8.4.1.4 std::ostream& operator<< (std::ostream & *o,* const Entry & *e*)

This file contains the code for the operator overload. the insertion ">>" and extraction "<<" operator is what is overloaded. You can see it the load dictionary function

```
7 {
8     o << e.word << "\t";
9     o << e.translation;
10     return o;
11 }
```

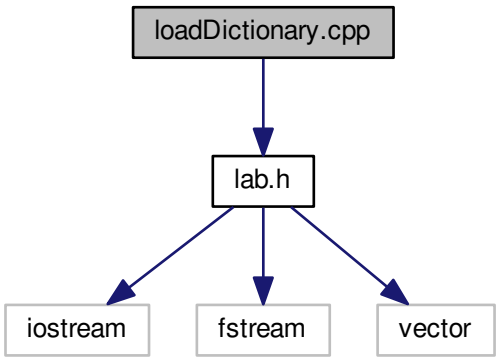
8.4.1.5 std::istream& operator>> (std::istream & *i,* Entry & *e*)

```
14 {
15     i >> e.word >> e.translation;
16     return i;
17
18 }
```

8.5 loadDictionary.cpp File Reference

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

Include dependency graph for loadDictionary.cpp:



Functions

- bool loaddictionary (string filename, vector< Entry > &dict)

8.5.1 Function Documentation

8.5.1.1 bool loaddictionary (string *filename,* vector< Entry > & *dict*)

Parameters

in	*the*	filename of the dictionary
in	*dict*	a reference to a vector of Entries

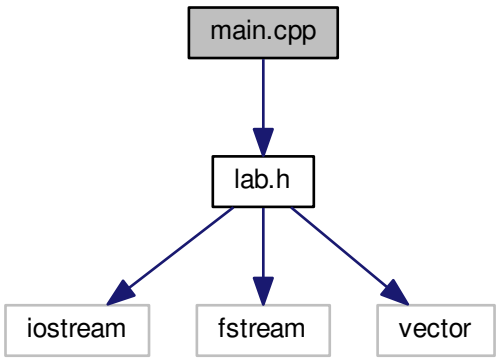
Returns

a boolean that indicates if the file was opened successfully

```
8 {
9
10
11     ifstream inpfile(filename.c_str()); //opening file
12     if (!inpfile) return false;
13
14
15     Entry e; // structure e declared
16     while (inpfile >> e) { //operator overloading done here
17         dict.push_back(e);
18         inpfile.ignore(80, '\n'); // skip the rest of the line
19
20     }
21
22
23     return true;
24 }
```

8.6 main.cpp File Reference

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



Functions

- int main ()

8.6.1 Function Documentation

8.6.1.1 int main ( )

This is main function of the project. Contains the three fucntions (loaddictionary, foundwords, addwords) See comments for better understanding

```
11 {
12     vector<Entry> dict;
13     string word; // user inputs word
14     string translation; // outputs translation
15     bool ok, quit;
16 
17     ok = loaddictionary("dict.dat", dict);
18     if (!ok) {
19         cout << " **** Cannot load Dictionary ***** \n";
20         return 1; //ERROR
21     }
22 
23 
24     string line;
25 
26  ifstream inpfile("dict.dat"); //opening file again so that once updated
27     if (!inpfile) return false; //the new information can be called without restarting the program
28 
29     getline(inpfile, line);
30     cout << line << endl;
31     cout << "Program by Samuel Jothimuthu" <<endl;
32 
33     quit = false;
34     while (!quit) { //iteration control structure
35         loaddictionary("dict.dat", dict);
36 
37         string choice; //simple choice of yes or no
38         string newtran; //new translation for word user enters
39         string inputstring; // the full string that will be appened to the file "dict.dat"
40         cout << "Enter a word or 'q' to quit ==> ";
41         cin >> word;
42         cin.ignore(80, '\n'); //this allows the console argument to execute but skipping the line
43         if (word == "q")
44             quit = true;
45         else if (foundword( dict, word, translation)) //function must return true
46             cout << translation << "\n\n";
47         else //The selection control structure
48            { cout << word << " --not in the dictionary. \n Would you like to add it? (y/n)\n";
49             cin >> choice;
50             if (choice == "y") { //any other input does not work.
51                 cout << "What is the Italian translation for " << word << "?" << endl;
52                 cin >> newtran;
53                 inputstring = word + "\t" + newtran; // this builds the full string that the program can
    then call upon.
54                 addwords("dict.dat", inputstring); //runs the addwords function, adding it into the
    file.
55                 }
56         }
57         }
58         return 0;
59     }
```

8.7 specification.dox File Reference

Index