Homework 2 - Trie

Part of the glossary (vocabulary, clavis) of your thesis is stored in a text file. However, the thesis is not yet finished and you want to be able to add and remove words so that the glossary is kept in alphabetical order. Sorting big amounts of data is quite slow. You decide to use a different approach and use a specific data structure to help with the task. The data stucture is called a Trie.

Commands

Command	Description
R <t>	Reads data from the file <t>.
A <w>	Adds the word <w> to the datastructure, if it did not exist there already. If the word was added, returns true, otherwise returns false.
D <w>	Removes the word <w> from the datastructure. If the word existed and it was removed, returns true, otherwise returns false.
F <w>	Finds the word <w> from the datastructure. If the word was found returns true, otherwise returns false.
N	Solves and returns the amount of words in the datastructure.
E	Empties the data structure.
P	Prints the contents of the datastructure. Words are separated from each other with the SEPARATOR constant.
Q	Quits the program.

Word Data File

A word file consists of words which are separated from each other with one space-charater. All words consist of lowercase letters a-z, i.e. no special characters nor scandinavian letters are used. An example of a word file:

```
cat␣pub␣dog␣aardwolf␣zebra␣publish␣alpha␣a␣b␣aardvark
```

Files Provided by the Course

The course provides two files that are needed in the implementation of the assignment: main.cc and datastructure.hh. Both are available in the course version control.

main.cc

The main program creates an instance of a Datastructure class and calls its public interface based on the given commands. The main program prints the given command and its parameters, if any, before executing the command. The main program reads the data file described in Section Word Data File and uses the add function to add words into the data structure.

Changing the main function is prohibited.

datastructure.hh

`datastructure.hh` defines the Datastructure class and the related interface. The student's task is to implement the member functions. Everything that needs to be added is added to the private part.

Functionality of program

The program stores words read from the given file into the trie datastructure. The possible operations are: find, add and delete.

It is also possible to empty the datastructure and the program must work with an empty datastructure as well. It is not allowed to limit the length of the words or the size of the vocabulary. All characters are latin alphabets from a to z as described before.

All datastructures used in homework have to be implemented by the student. Any use of STL is forbidden. `vector` initialized and used with a static size and `array` make an exception. The `string` is not a part of STL, so its use is allowed.

Public Test Case

A sample run of the program is shown below. The program is executed with a public test data also given to the students. The character $ stands for the command prompt and the program prog is run in the current working directory. Commands in the file test_public.in are directed to the program with the < character. The program then reads the file words_pub, which is located in the current working directory.

The public test data is recommended to be downloaded as a file into the testing environment to avoid any problems due to using copy&paste or writing directly. Files related to the public test data are the following: test_public.in, test_public.out, public_words. The files are available under the course version control. A Makefile is also provided.

```
$ ./prog < test_public.in
> R public_words
> N
Amount of words in the datastructure: 10
> F zebroid
Word zebroid not found.
> P
a aardvark aardwolf alpha b cat dog pub publish zebra
> F zebra
Word zebra found.
> A lion
Word lion added.
> N
Amount of words in the datastructure: 11
> A lion
Word lion not added.
> D zebroid
Word zebroid not found.
> D a
Word a removed.
> E
> N
Amount of words in the datastructure: 0
> P
> Q
```