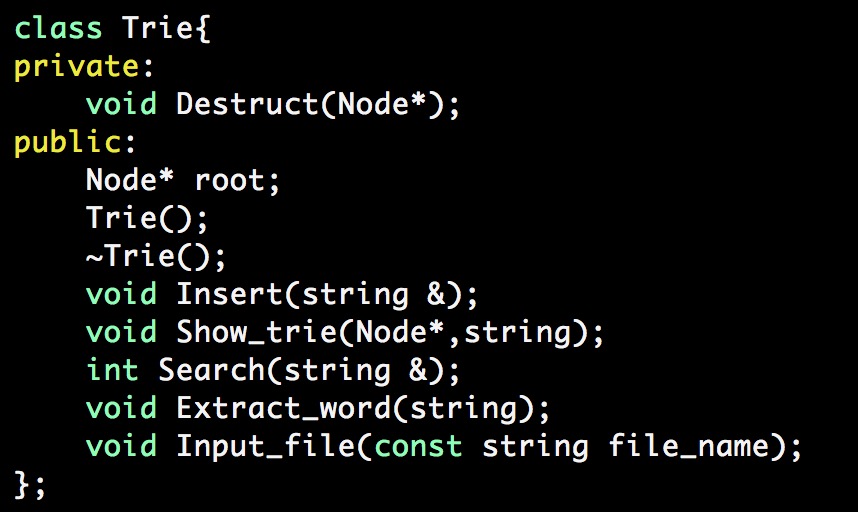
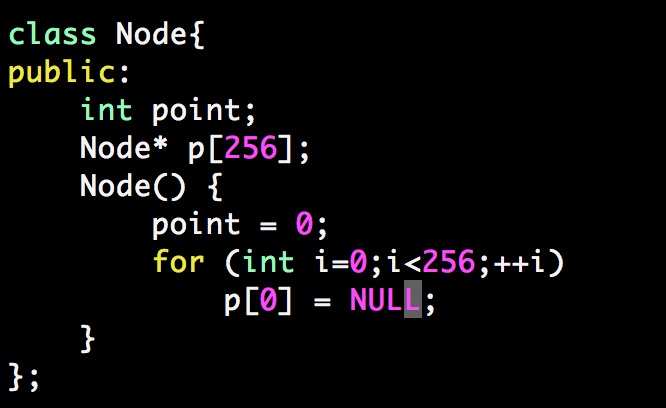
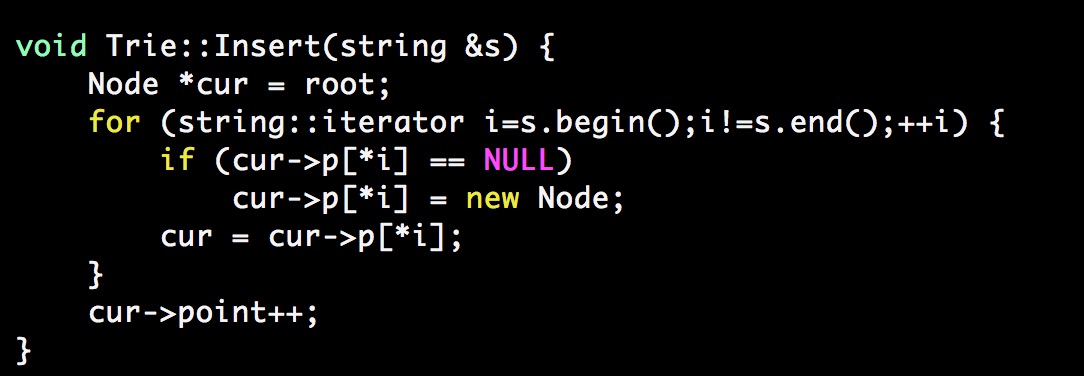
**CS163 Final Project Proposal**

**Data structure:** Trie

Trie is an ordered tree data structure that is used to store a dynamic set or associative array where the keys are usually strings. Unlike a binary search tree, no node in the tree stores the key associated with that node; instead, its position in the tree defines the key with which it is associated. All the descendants of a node have a common prefix of the string associated with that node, and the root is associated with the empty string. Values are not necessarily associated with every node. Rather, values tend only to be associated with leaves, and with some inner nodes that correspond to keys of interest.



Firstly, we will process the documents and store all words in our trie (one trie/one file).



For every input query, we will process the query and search its keywords in the trie for each file and implement algorithm correspond with our query.

**Solution for each query:**

1. ***AND***

For each text file we have one corresponding trie, we just need to search each operand in this trie. One file is fine if it contains all input operand of AND operator.

1. ***OR***

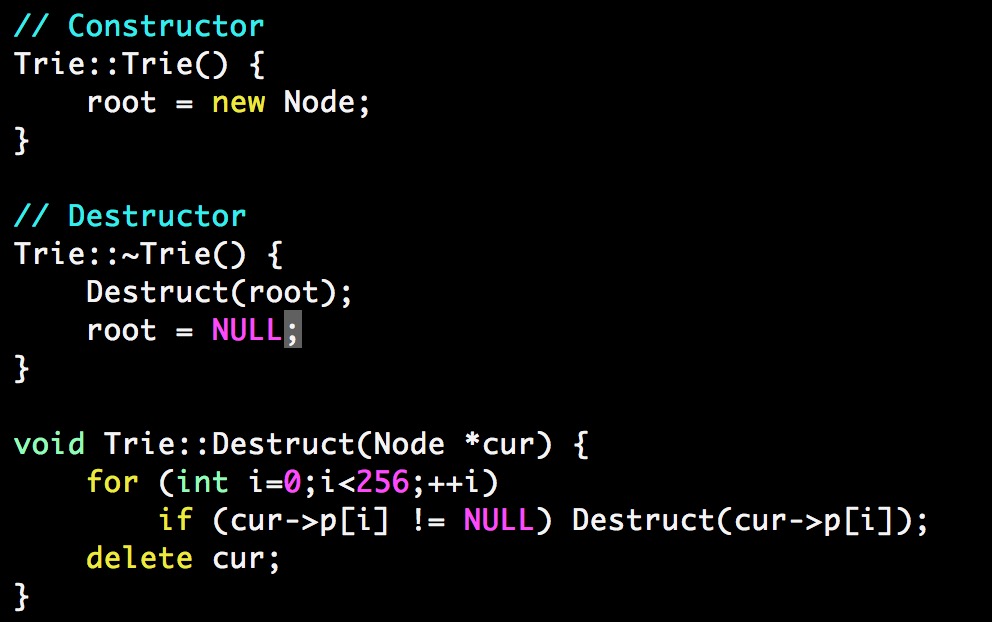
For each text file we have one corresponding trie, we just need to search operand one by one in this trie. One file is fine if it contains at least one operand of OR operator.

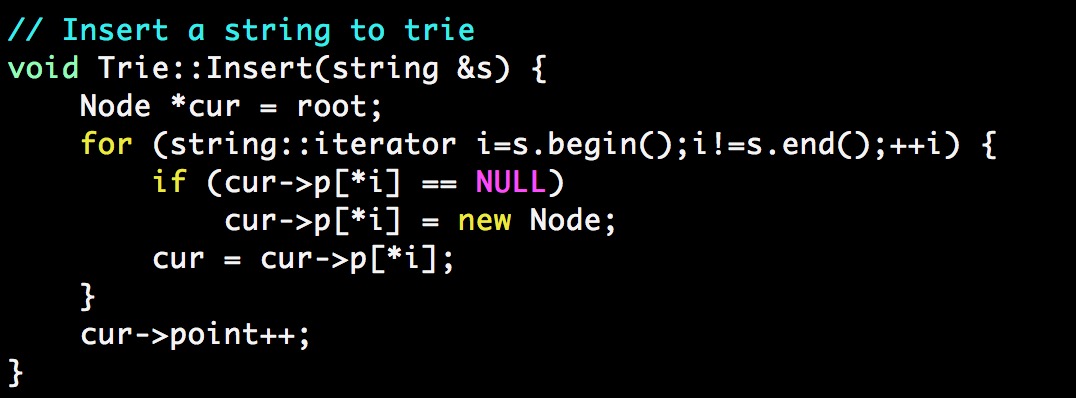
1. ***Manchester –united***

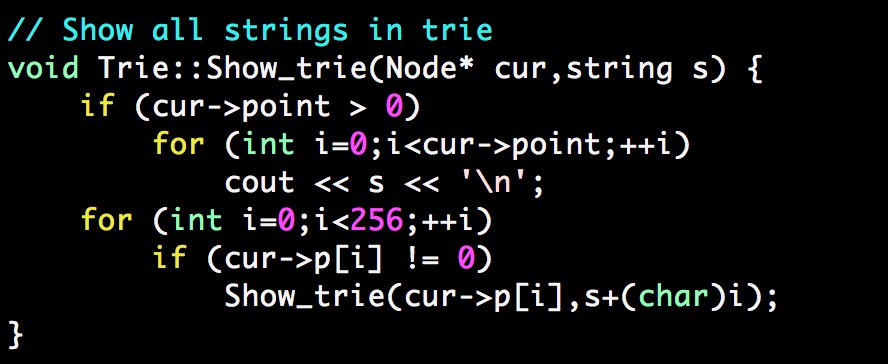
For example: A -B

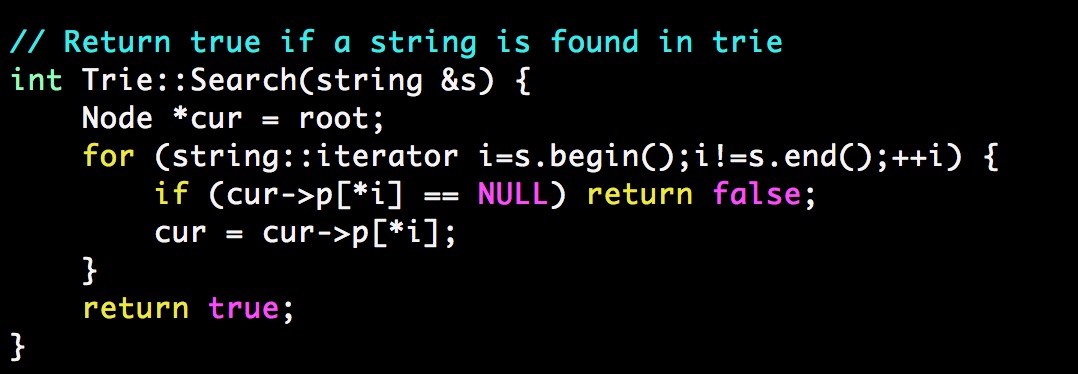
For each text file we have one corresponding trie, we just need to search for **A** in this trie. If a file has **A** we’ll continue to search for **B** in this trie. One file is fine if it contains **A** and doesn’t contain **B**.

Some function of data structure

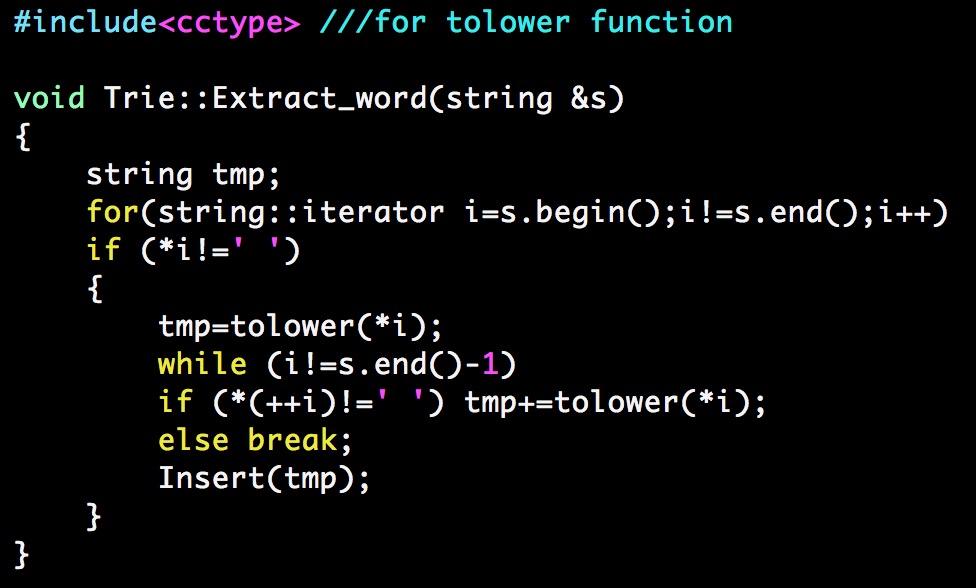


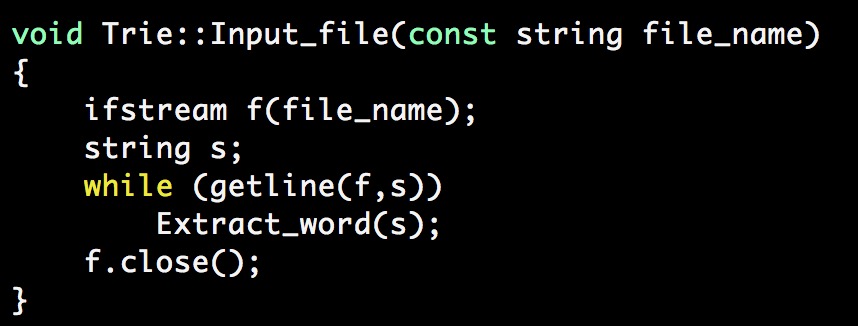






Source code for parsing documents, inserting data

****

****