Lebanese University
Faculty of Sciences 1

Imperative Programming II

Duration: 1 month

Project Title: Dictionary

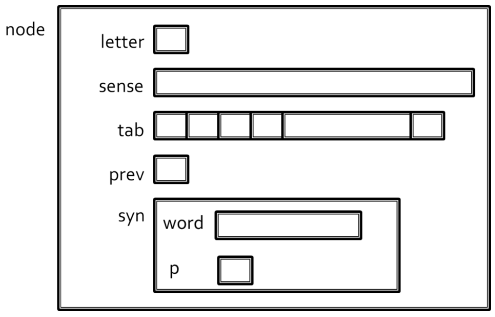
Subject

This mini-project aims to achieve an English dictionary that largely inspired the form of an n-ary tree (n=26) where nodes represent the different letters in words. An example is provided at end of document.

Used Types

1. type dictionary (dico):
   - array of 26 pointers to nodes (as there is 26 letters in the alphabet).
2. type node containing :
   - field letter.
   - field prev (pointer to the previous node; nodes directly pointed by the pointers of dico are equal to NULL).
   - field tab: array of 26 pointers to nodes.
   - field sense: string (array of 200 characters); meaning of the word ending on that node, otherwise empty string "\ 0".
   - field syn: variable of type synonym
3. type synonym :
   - word (20 characters)
   - p pointer to the node that contains the dictionary sense of that synonym, i.e. the node corresponding to the last letter of the word).

Below, you can find a graphical representation of the various fields constituting a node:



Remark:

Other type of data can be defined upon need.

Required Functions

1. createDico: creates and returns an empty dictionary whose table pointer is initialized to NULL everywhere.
2. exists: function that checks the existence of a given word in a given dictionary. Note though not enough to find all nodes corresponding to the successive letters of the word to say that it exists, but we must, in addition, ensure that the last node contains a sense (nonzero string), if not its existence returns to the fact that it is a prefix of another word in the dictionary. If the word exists in the dictionary (makes sense), this function returns a pointer to the node containing the sense (node corresponding to the last letter of the word), otherwise the function returns NULL.
3. addWord: adds a given word to a given dictionary (creates and links the different nodes and fills the corresponding sense field of this word in the last node). Returns 1 on success and 0 on failure (failure: word already exists ie, it has a meaning in the dictionary). Obviously, this function uses the function exists.
4. changeSense: changes the field sense for a word that already exists in the dictionary.
5. search : function that searches for and returns the meaning of a word in the dictionary. Obviously, this function uses the function exists.
6. getWord: function that, from a pointer to a node in the dictionary, return the word (string) ending with this node. (Use the prev pointer in each node to iterate back).
7. setSyn : function that takes as parameters two existing words in the dictionary and shows (updates the value in the syn field) that each of them is the synonym of the other.
8. deleteWord: function that deletes a word from a dictionary. In this case, it must release all nodes previously used only for that word. (Before releasing each node, verify that it is not part of other words in the dictionary that they should continue to exist. Must also remove the link of synonymy.
9. print: function that displays all the dictionary words and their meanings, and potential synonyms. This display must be in alphabetical order (natural order of the dictionary).
10. liberate: function that releases the entire dictionary of heap memory.
11. save: that writes the entire dictionary into an XML file, in order to load it again later.
12. load: rebuilding a dictionary from an XML file.

Hint

To find the index of element of a letter in the array of pointers, just do index = letter - 'a'. For example, the index of pointer corresponding to letter 'c': index = 'c'-' a '= 2, because the indices start with 0.

Detailed Example

Below, is a graphical representation of a dictionary containing three words:

- ball: A solid or hollow sphere or ovoid.
- bay: Inlet of the sea where the land curves inward. (syn : cove)
- cove: A small sheltered bay.(syn : bay)

We thus give the content of the XML file representing the dictionary:

<dico> <a> <l> <l> <sens> A solid or hollow sphere or ovoid. </sens> </l> </l> <y> <sens> Inlet of the sea where the land curves inward. </sens> <syn> cove </syn> </y> <c> <o> <v> <e> <sens> A small sheltered bay. </sens> <syn> bay </syn> </e> </v> </o> </c> </dico>