Project 10

The purpose of this program is to read a text file and store words and the line numbers on which they occur, to be printed to another text file and screen. This project in a way brings you a tiny step closer to the search engine technology. You must use a Binary search tree as well as any other data structure needed. Proceed as follows:

1. Design a class called WordList, which is given below partially. You will need to add to it as per need of the program. In place of class you can also use WordList struct. Still all the same member functions and data members (as WordList class) would be needed.

Class WordList{

private:

/\*\*

\*Stores the word from text file

\*/

string Word;

/\*\*

\*Stores the line numbers (as integers) on which the Word is found

\*/

SinglyLinkedList LineNumList;

//Add any other data members as per your need.

public:

/\*\*

\*Reads data from the file or keyboard to fill the WordList object

\*/

friend istream & operator >>(istream & in, WordList& WL);

/\*\*

\*Writes data to the file or to screen to print the WordList object

\*/

friend ostream & operator <<(ostream & out, const WordList& WL);

/\*\*

\*Virtual destructor

\*/

virtual ~ WordList ( );

//Add any other member functions, as you need. For example, you

//will need to overload the following operators, so that WordList

//can be placed in a binary search tree: operators >, <, and ==.

};

The suggested UML of WordList class may look like below.



Insert WordList objects into the Binary search tree such that when printing the tree, using an In-order iterator, the words are printed alphabetically and right after the word, the line numbers on which they occur in the text file are also printed. The user should be allowed to search the binary search tree for the words in there and print the words and its line numbers if it exists.

Feel free to add any new member functions or write stand-alone functions to the data structure classes you use. Only restriction is that you cannot remove any data members or existing member functions from the data structure classes you use and you cannot alter the access restrictions already placed. (For example, you cannot make the private members public). Use the string Word as the comparison key to build binary search tree. I give you freedom to organize the files as you wish. Only restriction is that the driver program file should have no classes defined in it.

Planning is very important for this project, therefore ask yourself the following questions and get answers to them.

1. Which classes would I need and how would I organize them in different files.
2. How would I test the member functions of my WordList class as I write them?
3. What stand-alone member functions would I need to write and how would I test them?
4. What strategy would I use to filter out the non-words from the file (such as comma, period, and other symbols) so that they are not read as words?
5. What strategy would I use so that once a word is read and first line number in which it occurs is recorded; for later occureneces, only its line number is entered in the linked list?
6. How would I modularize my class member functions and stand-alone functions the best?
7. What program bugs am I most prone to, and how would I avoid them?

Allocate time for each step so that you get done before the due date.

Member and other functions needed

For Binary Search Tree Class (you need to add this function if not there already).

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | template<typename Type> | | | | | const [SearchTreeNode](C:CoursesCS2ExperimentsLab7CS2Lab7htmlclassSearchTreeNode.html)< Type > \* [BinarySearchTree](C:CoursesCS2ExperimentsLab7CS2Lab7htmlclassBinarySearchTree.html)< Type >::get | ( | Type | obj | ) |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Returns a constant pointer to a BinarySearchTreeNode which contains the object obj,**  **else returns NULL.**  Parameters:   |  |  | | --- | --- | | obj | is the object searched in the binary search tree. |   Returns:  NULL if obj is not in Tree, else returns a constant pointer to the [Node](C:CoursesCS2ExperimentsLab7CS2Lab7htmlclassNode.html) which  contains obj. For WordList ClassFriends and Related Function Documentation  |  |  | | --- | --- | | |  | | --- | | ostream& operator<<  ( ostream & out, const [WordList](C:CoursesCS2ExperimentsLab7CS2Lab7htmlclassWordList.html) & WL) [friend] | |  |  |  | | --- | --- | |  | **Writes data to the file or to screen to print the WordList object.** |  |  |  | | --- | --- | | |  | | --- | | istream& operator>> ( istream & in, [WordList](C:CoursesCS2ExperimentsLab7CS2Lab7htmlclassWordList.html) & WL) [friend] | |  |  |  | | --- | --- | |  | **Reads data from the file or keyboard to fill the WordList object.** |  Member Function Documentation  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | const [SinglyLinkedList](C:CoursesCS2ExperimentsLab7CS2Lab7htmlclassSinglyLinkedList.html)<int> WordList::getList | ( |  | ) | [inline] | |  |  |  | | --- | --- | |  | **returns the singly linked list member.** |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | string WordList::getWord | ( |  | ) | [inline] | |  |  |  | | --- | --- | |  | **Returns the Word member.** |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | bool WordList::operator< | ( | [WordList](C:CoursesCS2ExperimentsLab7CS2Lab7htmlclassWordList.html) | W | ) | [inline] | |  |  |  | | --- | --- | |  | **Returns true if W is greater than the Word member, else returns false.** |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | bool WordList::operator== | ( | [WordList](C:CoursesCS2ExperimentsLab7CS2Lab7htmlclassWordList.html) | WL | ) | [inline] | |  |  |  | | --- | --- | |  | **Returns true if W is equal to the Word member, else returns false.** |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | bool WordList::operator> | ( | [WordList](C:CoursesCS2ExperimentsLab7CS2Lab7htmlclassWordList.html) | W | ) | [inline] | |  |  |  | | --- | --- | |  | **Returns true if W is less than the Word member, else returns false.** |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | void WordList::setList | ( | [SinglyLinkedList](C:CoursesCS2ExperimentsLab7CS2Lab7htmlclassSinglyLinkedList.html)< int > | L | ) | [inline] | |  |  |  | | --- | --- | |  | **Sets the Linked List member equal to the argument L.** | |  |  |  |  |  | | --- | --- | | |  | | --- | | void WordList::setWord ( string WL  )  [inline] | |  |  |  | | --- | --- | |  | **Sets the Word member equal to the function argument.** | |

Other functions (Could be stand alone or member of a utility class)

The exact prototype of the functions given below depends upon how you write them. The list below only describes possible names for them and their functionality.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | void buildTree | ( |  | ) |  | |

|  |  |
| --- | --- |
|  | **Builds the WordList tree as it reads the input file.** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | string convertToLower | ( | char \* | token | ) |  | |

|  |  |
| --- | --- |
|  | **converts the input character array to lower case and returns corresponding c++ string.** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | void printWordListTree | ( | ostream & | out | ) |  | |

|  |  |
| --- | --- |
|  | **prints the** [**WordList**](C:CoursesCS2ExperimentsLab7CS2Lab7htmlclassWordList.html) **tree.** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | void searchWordListTree | ( |  | ) |  | |

|  |  |
| --- | --- |
|  | **Searches the word list tree for a given word.** |

See me in person to see how a finished program works.

Files Given to you

The following file is given to you to be used in this project.

Iterator.h

LinkedListInterface.h

The above files have following classes whose UML is shown below.



FIG. 1. Node class used as node of SinglyLinkedList class



Fig. 2 Iterator class



FIG. 3 LinkedListInterface and SinglyLinkedList class that implements Linked List Interface.



FIG. 4 SearchTreeNode class



FIG. 5: BinaryTree Class and BinarySearchTree classes