Joshua Brummet

105795915

Data Structures

May 11th, 2016

Record Database Final Report

Description:

This program, developed in the programming high-level language C++, is a simple object orientated database for a person’s contact information. The information includes, all of the following.

* Unique ID number
* First name
* Middle name
* Last name
* Company name
* Home Phone
* Office Phone
* Email
* Mobile
* Street Address
* City
* State
* Zip Code
* Country
* List of Affiliates

A contact’s information is displayed in this order to the user of the program. The database uses the base data structure called an AVL tree, to do an assortment of different manipulations and searching methods of the contacts to the user’s liking. After the user has finished searching/modifying/sorting the program, he/she can finally output a report to the filename of their liking, usually a text document.

Overall Software Architecture:

The entire program is constructed around the user-defined class called “RecordType”, which is has been implemented to sufficiently support each major function in the program. RecordType creates a record for one contact, allowing multiple algorithms to operate on a single contact, which is implemented inside the data structure AVL tree. The AVL tree data structure is used to store each object of the RecordType inside an AVL node, which goes inside the AVL Tree, creating the database of each contact and their personal information. Using the AVL tree and RecordType, the main driver user-defined class RecordList, creates menu and performs major functions allowing the programs user to do as they wish.

* **RecordType Major Functions**

1. getField(unsigned int index);

This function is one of the most important functions inside RecordType. The reason it is so important that allows the user, to easily choose the Field of the contact they wish to sort/search/modify with a simple index number (0-13).

1. std::istream &operator>>(std::istream &is, RecordType& contact);

This major function is the overloaded insertion stream operator for the RecordType class. It’s main role is to allow a user to retrieve a data file from their computer, and insert it to the program, so they can transform their file, or search it.

1. Void printf();

PrintF() is a important function inside the program, because it allows the user to display a report of their contacts throughout the program!

1. RecordType CreateRecord();

The CreateRecord() function also plays a important roll because it allows the user to create a contact to add into the main data structure.

1. Data retrieval functions for the Affiliates.

There are a series of different functions inside the RecordType to allow the user to add/modify/search/delete the affiliates of a contact with the main driver class RecordList. These play an important role to allowing the driver function to perform these tasks inside this scope.

* **AVL Tree Major Functions**

1. bool insert(RecordType newData);

The AVL tree insert function allows the other functions to insert data inside the tree, which holds all the information of the program. Functions inside the main driver class RecordList utilize this function the most.

1. std::vector<RecordType> &addtoVector(AVLNode \*node, std::vector<RecordType> &vec);

The, add to Vector function, could possibly be the most important function inside the AVL tree. It allows the main driver class to create a temporary data structure called “Vector” and simply transfer all the data from the AVL tree into this Vector. This allows multiple performances on the data from sorting to searching.

1. bool remove(RecordType data)

This function simply allows the user to a certain record inside the main data structure for good. If they wish.

* **AVL Node Major Functions**

1. Set and Get data functions

These two data functions are important because they allow the AVL tree to add a list of nodes of the data type RecordType.

* **RecordList Major Functions**

|  |  |
| --- | --- |
| **RecordList.cpp** | **RecordList2.cpp** |
| 1. Void Mainmenu()   This is the main driver function, which opens up the main menu of the program. Allows user to choose multiple options to their database and exiting the program. | * Void Modify\_Search()   This function creates a temporary vector of RecordType and adds the data from the AVL tree with the addtoVector() function. Then uses a series of algorithms and loops to allow the user to modify a contacts field (first name -> affiliates). Modify\_Search() also takes use of the remove function to remove the original contact and inserts the new contact they have created. This function utilizes the RecordTypes “Major Functions”. |
| * Void MenuDisplay()     Allows user to display the records in their database. | * Void Contains\_Search()   Contains\_Search() creates a temporary vector of RecordType and adds the data from the AVL tree with the addtoVector() function. Then uses a series of functions to search the database by using the STL function find() which allows the user to type in only a certain string of letters to obtain a contact which contains those letters. For example, if a user wanted adailey@yahoo in the affiliates. The algorithm will return all contacts in the database, which contain those letters in that order. This function utilizes the RecordTypes “Major Functions”. |
| * Void MenuInsert()   Allows user to create a record from the CreateRecord() function inside the RecordType class, and then using the AVL Tree insert() function to insert the new contact inside the main data structure. | * Void General\_Search()   Is a basic searching function which creates a temporary vector of RecordType adds the AVL tree to the vector and does a series of algorithms and loops to search the database. This function utilizes the RecordTypes “Major Functions”. |
| * Void Menusearch()   Displays the sub-search menu, allowing the user to choose from multiple different functions to either, search, search (contains), modify, and delete. | * Void Delete\_Search()   This function creates a temporary vector of RecordType and adds the data from the AVL tree with the addtoVector() function. After a series of algorithms utilizing the RecordTypes “Major Functions” it  locates the contact a person will want to delete, and uses the AVLtree delete function to delete this contact for good. |
| * Void MenuSort()   Displays the sub-sort menu, allowing the user to sort the data they have and display it, sort within the sorted data, and or sort and save the data to a new data file. | * Void sort\_and\_Display()   Creates a temporary vector of RecordType, then uses a derived class from the main driver class RecordList called, sortContact. This allows the user to choose a field number to sort the data and display it to the user. |
| * std::istream &operator>>(std::istream &is, RecordList &rl)   This is the overloaded insertion stream operator for the main driver class. Utilizes the RecordType insertion operator to insert into a object created from the RecordType class then insert that object into the AVL tree using the insert function, until the end of the file. | * Void Sort\_and\_Save()   Creates a temporary vector of RecordType, then uses a derived class from the main driver class RecordList called, sortContact. Then allows the user to save their newly sorted data to a new data file or a existing data file. |
| * void MenuSave()   The MenuSave() function allows the user to save their database to a new file name, or same file name if the wish, after their series of modifications and searching. |  |

Select Search Option:

* GeneralSearch()
* Contains\_Search()
* Modify\_Search()
* Delete\_Search

View Records

CreateRecord()

Main Menu:

MenuLoad();

MenuInsert();

MenuDisplay();

Menusearch();

MenuSave();

Exit.

Display disclaimer:

Tells user if data has been added, from file.