LL-OPERATORuser documentation

Mr.Erden Basar Instructor

Driss Bouadoud Coder

Issue:

In order to do operations on a file that contains members with the following data:

Number, Name, Surname

we need to do it either manually or automatically, but we need to do it in automatic way, so we need to store the pieces of information either in an array or “linked list” and do operations like, sortation( by name or number), insertion, search, deletion, so this program contains all the mentioned functions, so what’s the problem with it?

Problem

Sorting, search, deleting and inserting are operations on file that contains data may require a whole group of people to deal with it without considering the mistakes and the long time it takes to do all the operation on the list, because of that, an easier solution comes which is, LL-Operator program which I made to solve the mentioned problems, by using certain operations like selection-sort for both(Number, Full name), Insertion in case if you suddenly want to add a new information with no need to create the whole list again, and deletion which is simply deleting a certain information and search to find a member among hundreds of members, let’s see how it works.

Before we start the explain how some algorithms work.

Selection-sort:

Selection-sort works by choosing one element from the list and compare it with all other members in the list using a loop, and depends on whether ascending or descending order, it switches the location of both selected members(the small, greater member).

Insertion operation:

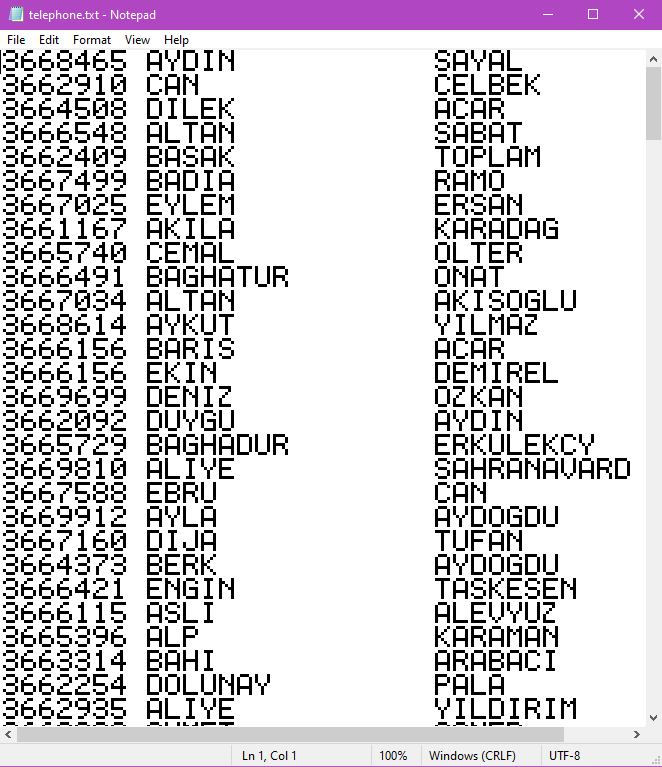
Basically, it is adding a new piece of information to an existing list or file.

Deletion operation:

it is the operation that allows you to delete any information you want without any change in other data.

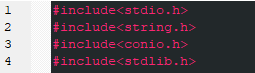
Content

External environment

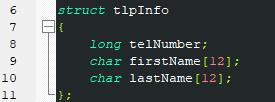


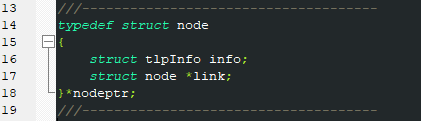
In order to use the LL-Operator first we need a file like the one on the left that contains those pieces of information Telephone number, Name, surname, whatever the number of members it really doesn’t matter.

Internal environment

Headers:

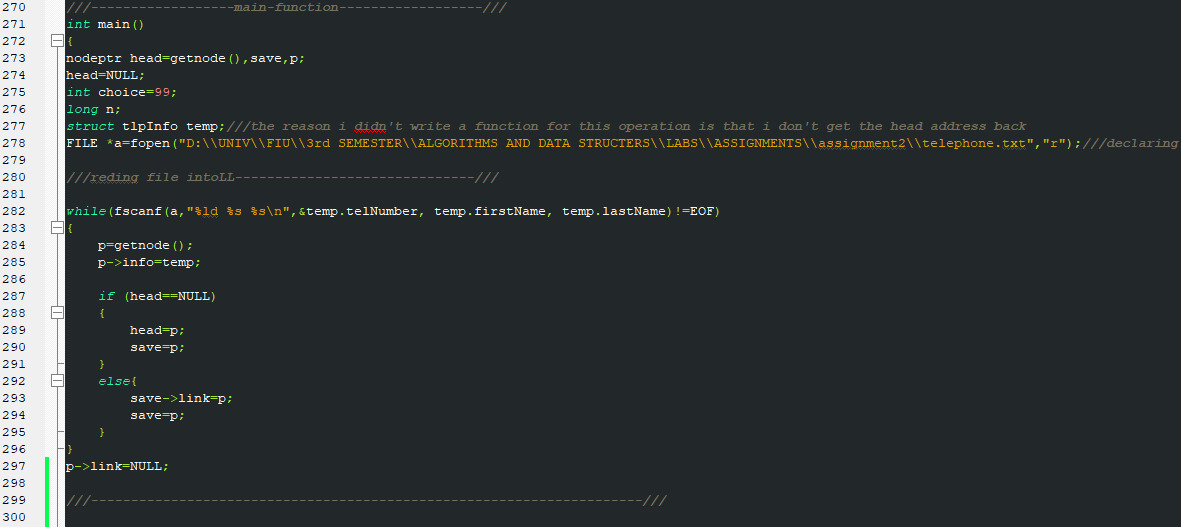
In order to make the task much easier I used the headers on the left so I can use more functions.

Structures:

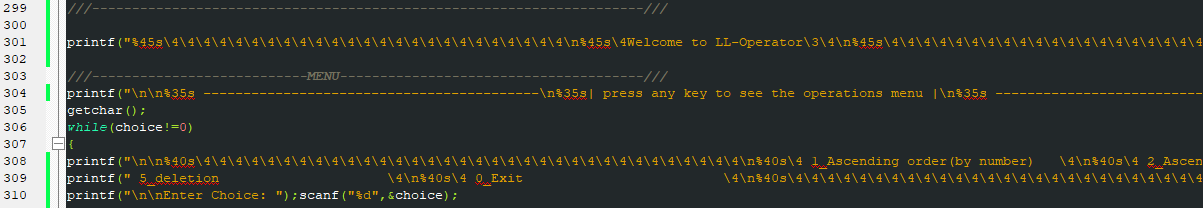
Telephone information structure defines the nature of each node that will contain the data in the file.

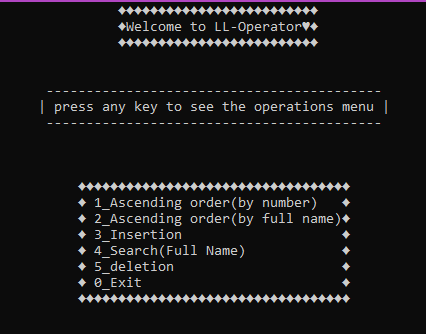
Node’s structure that contains info with “struct tlpinfo” nature and the link that links each node the one next to it, and not to forget the \*nodeptr which I’m going to use for nodes declaration.

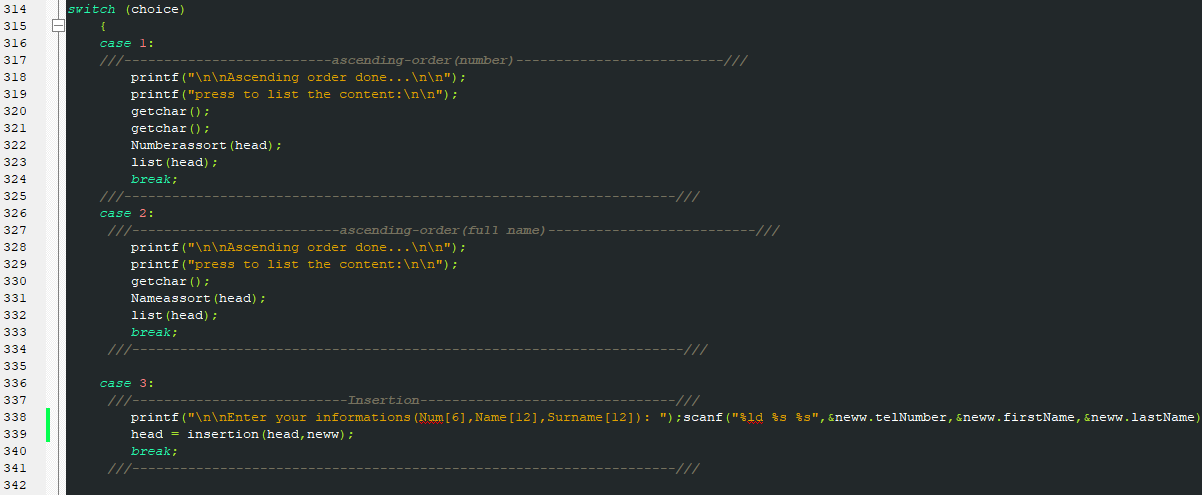
Functions:

The Main function as usual plays the role of the program’s spine, it has a loop that will create as many nodes as the file’s members,

so, starting with the variables head, save, p(previous), as you can find in any linked list base program we are going to use them to do operations on the linked list, then reading data from telephone.txt and insert it one by one into the nodes that are created by the while loop until the end of the file.

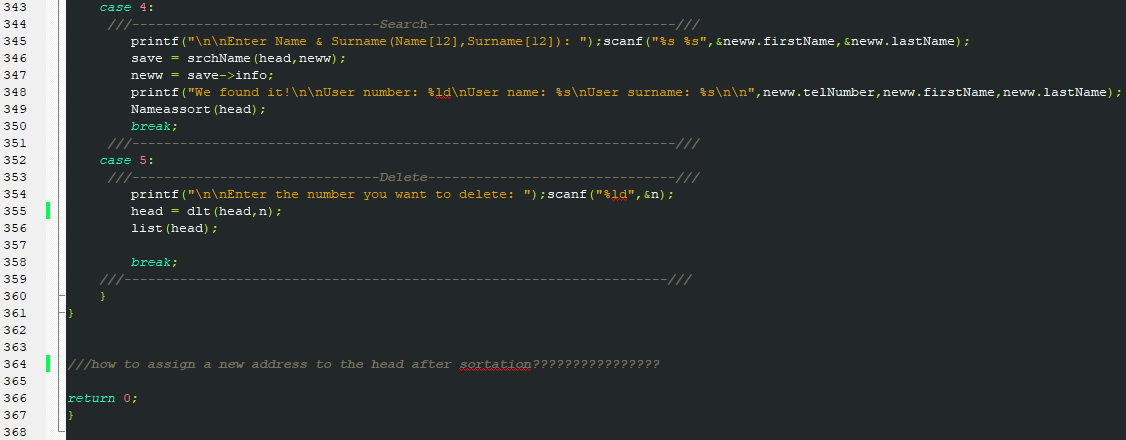
it contains the menu with a design that I made,



a switch case that will determine which function it’s going to launch, I started the main function by declaring some variables that We’re going to use later and function callers and file reader that will take information from the file we have.

Case1 is calling the Numberassort function to sort the list in ascending way by number which we will talk deeply about it later.

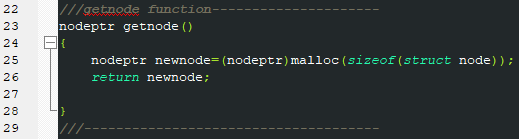
Case2, calling the Nameassort function to sort the list in ascending way by full name.

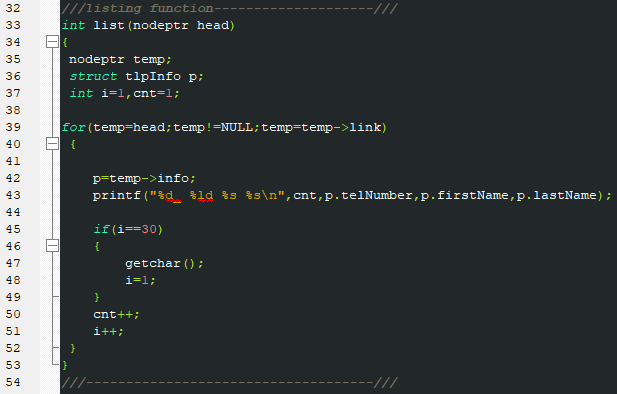
Case3, if the user choses to insert new member to the linked list using insertion function

Case4, in order to search for a certain user.

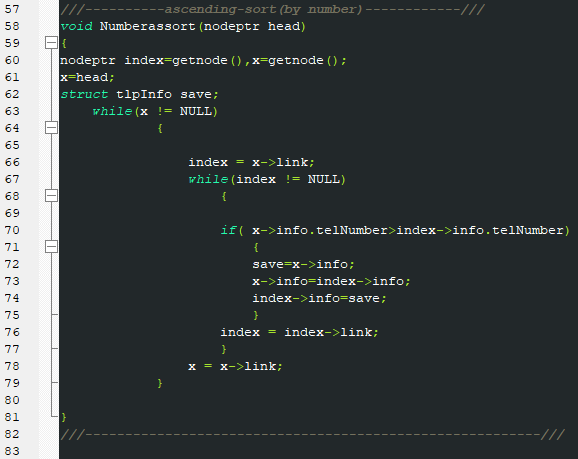
Case5, to delete a member using its number.

getnode():

 getnode() is a function which is known by allocating new random space of memory for any variable that is declared with “\*nodeptr”, or we can say it allocates memory for any pointer or variable, and we use in the program in order to allocates memories for all the linked list nodes so we can put data on them.

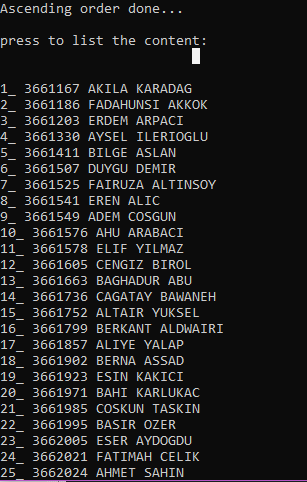
list():

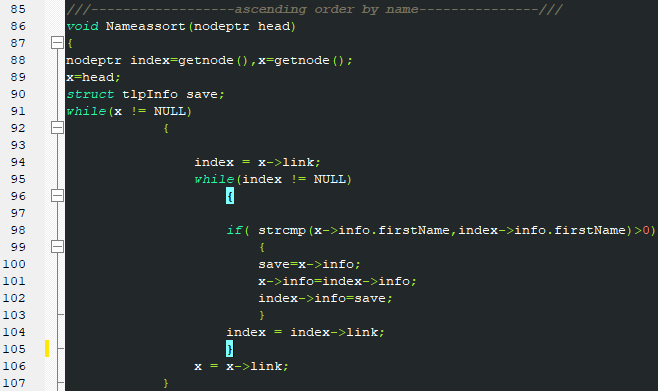
list function which has only one parameter which is the head of the linked list is used to print out the content of the linked list node by node, I use it in all the cases we’ve seen before, after calling each operational function so it will print the data after the modification, using a for loop we print out node after node by passing the variable temp to temp->link, the output will be 30 by 30 member for each press on any keyboard’s key, just as follows

Numberassort():

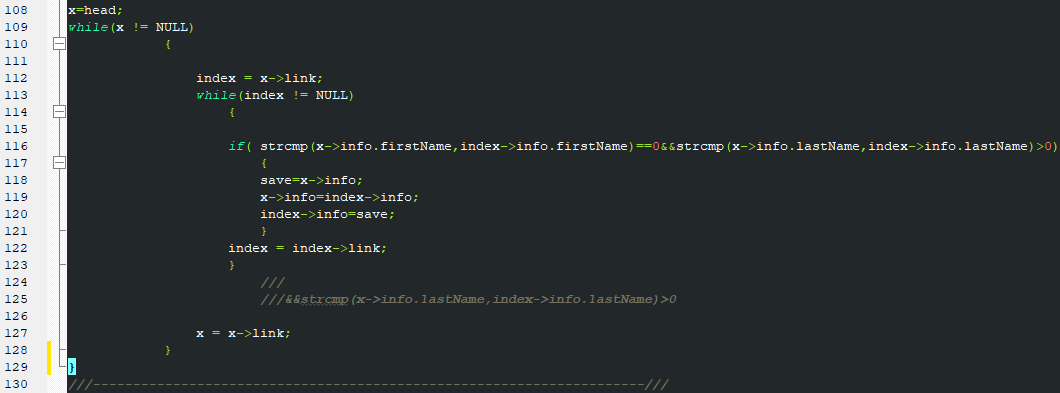
The first option in the menu is sort by number, so this function is responsible of sortation duty, the function has 1 parameter “head”, it uses selection sort algorithm to sort out all the data in the linked list in ascending way, the logic behind it is nested loop the first is checking all nodes once, the node “index” is checking the linked list starting from the node after x then compare the numbers which is one is smaller and which is greater then swipe them, which give us a sorted linked list.

The output after the sort operation is just like:

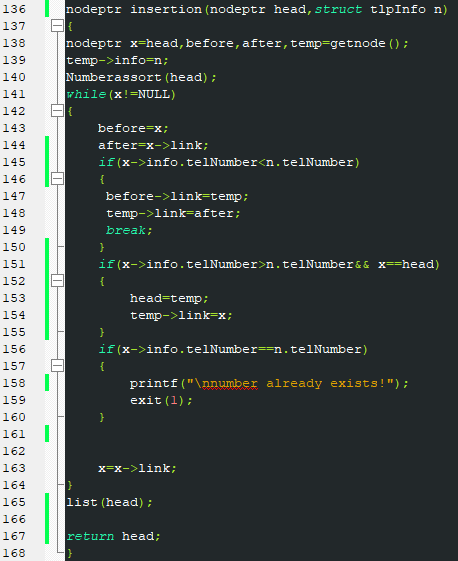
before after

Nameassort():

After the main() function this function is one of the bigger functions, it is responsible of sorting the linked list based on full name following a selection sort algorithm, has 2 while loops the first one sorts the linked list by name only, then the next while loop sorts the names by there surnames



 The output is like

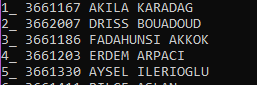
Insertion():

Insertion() with the type nodeptr responsible of adding a new member to the linked list, it has head and n parameters n is the new insertion’s information, the operation is done with respect to ascending order, it finds the node which its number is less than n.telNumber then it insert it after that node, the operation is operated with respect to 3 if conditions, if the location is before the head, and if the location after any other node except the head, and if the number is already exists.

To use the function follow the steps:

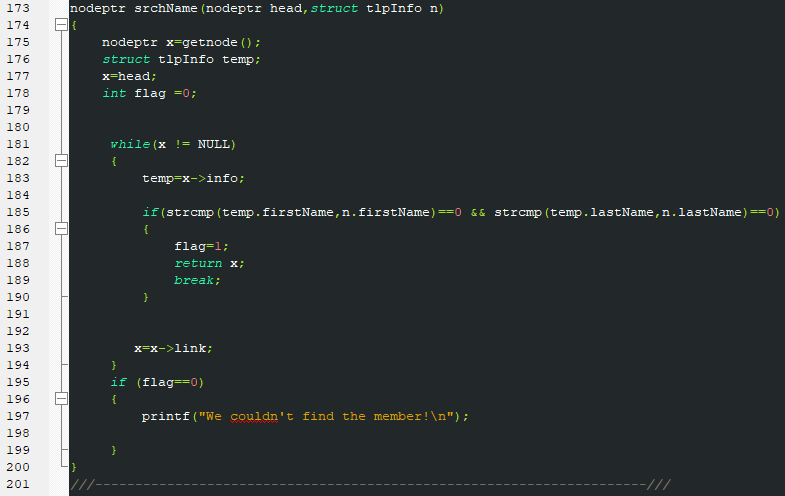
The program will ask you to enter your information





Then the member is in the list

In case if the given number does match another number in the list an error is raised

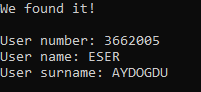
Search():

The search function has 2 parameters, the head and the full name, because the search is done by full name, when the member is detected it will display all of his information otherwise it will raise and error of “we couldn’t find the member”, it is used in the following way:

The program asks for full name

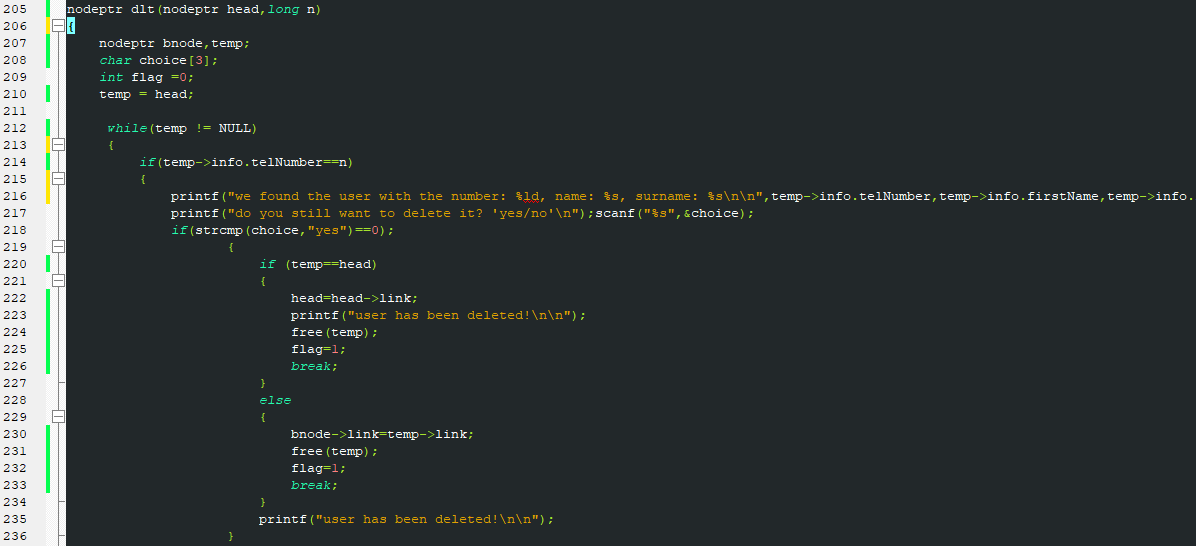


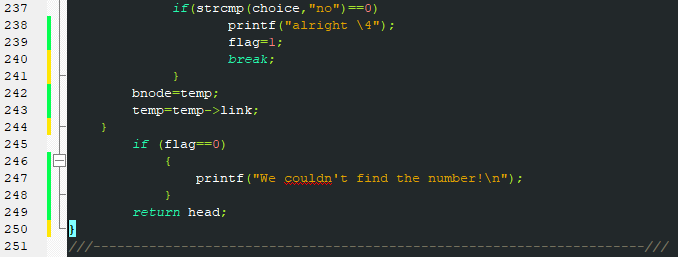


If it finds it

if it does not

Delete():

This function is responsible of fifth option in the menu, it is deleting by entering the number of the member you want to delete and it checks whether it is in the list or not, the script of it is kind of long because it has a lot if conditions, if the node was a head or if it is anywhere else.



The flag variable is initially zero, and as soon as the member is found it turns to 1, which will determine whether the user is found or not, it will list content in both cases.

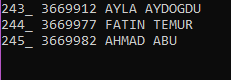
The steps for using as follows:



We choose the member 246 to be deleted





By typing yes the user is not anymore in the

List

Time spend

Design & Analyze: around 1h

Code: around 5h

Debugging: around 4h

Documentation: 3h

Total time spend: 13h

FINAL WORDS

I want to thank you for reaching this part and being interested in reading documentation of my program, the pictures might not be clear but you can check the number of the line within the main code.