Telephone Directory

Akshad Patel (19BEC0836)
Anmol Krishna Narain (19BEC0838)

***Abstract*— A telephone directory (also called as telephone book or phone book) is a type of document which is used to store the contact details of people who are known to the user. A contact can be stored in the directory by entering the first name, the last name and the telephone number of the person whose contact is to be stored. To implement this telephone directory, linked list data structure has been used.**

***Keywords*— Linked list**

I. Introduction

In order to implement a telephone directory, a linked list data structure is used. A linked list is a data structure which is a linear collection of data elements of various types. Based on the inputs given by the user, functions like adding a contact, searching a contact, editing a contact, deleting a contact, displaying all contacts in the directory can be implemented.

II. Linked List

As mentioned in the above heading, a linked list is a data structure which is a linear collection of data elements of various types like character type, integer type, floating point type, string, etc. It is a collection of nodes which together represent a sequence. Each node contains: data and a reference (or link) to the next node in the sequence. The last node in the sequence is linked to a terminator which is used to signify the end of thelist.

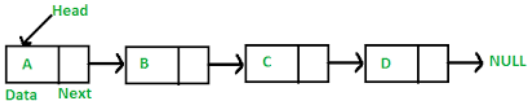

Fig. 1 General representation of a linked list

III. Creation of A Linked List

In order to create a linked list, first a node has to be created. Each node consists of character strings which take in the first name, last name and telephone number of the contact. A linking variable is also created to establish a link between two nodes.

Two pointers "start" and "temp" are created for the linked list. "start" is used to point to the top most node of the linked list and "temp" is used to create new nodes upon insertion of new contacts. Once the user enters the contact details of one person, if the user wants to add another, "temp" pointer creates another node to store the contact details of the next person. If no new contact is to be stored, the sequence is terminated by a null pointer.

.IV. Functions Involved with the Directory

A. Addition of a New Contact into the Directory

During the insertion of a contact, a new node is created in which the first name, last name and telephone number is stored in the respective strings initialized. After this process, the user can continue inserting new contacts or need not insert new contacts. After inserting one contact, if the pointer "start" is empty, it is initialized with the first node of the list. If a new contact is to be inserted, another node will be created using "temp" pointer to insert details of the new contact.

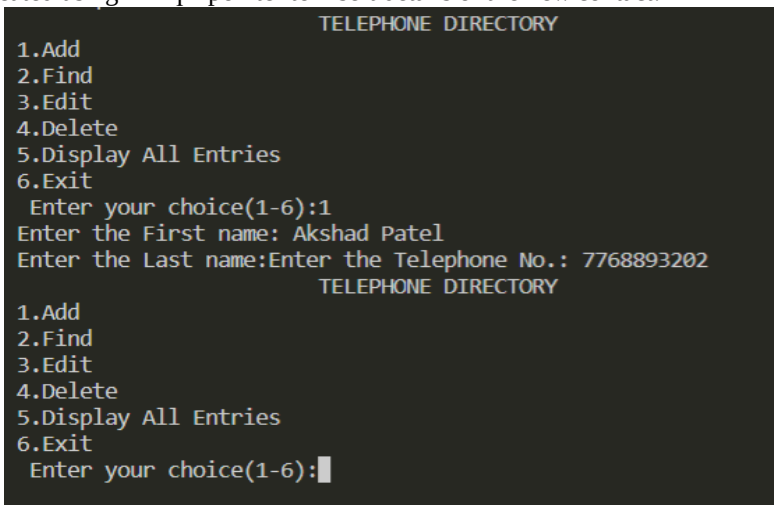

Fig.2 Output screen where new contacts are entered

B. Finding if a Contact is Present in the Directory

Once all the contacts are inserted by the user, the user can use this function to find out whether any contact is present in the directory. In this, the first name of the contact to be found is obtained from the user. A pointer variable "ptr" is created and is initialized to the top of the list. Until the first name of the contact which is required is not obtained, the variable "ptr" traverses through the full list. If there is a match, it is displayed else no record exists.

Fig.3 Output screen where a contact is searched and found in the list

C. Editing Any Existing Contact in the Directory

If there are situations in which the user entered any detail incorrectly, this function can be used to correct the details. A pointer variable "ptr" is declared and is initialized to the top of the list. The first name of the contact whose contact details are to be changed is obtained from the user. Once the first name is obtained, the list is traversed using the pointer variable "ptr" until a match is obtained. If a match is obtained, the user can enter the new details of the contact and it will be stored in the list. If no match is obtained, then "No Matching Records Found" is printed.

```
void edit()
{
    struct node *ptr;
    char str[20];
    if(start==NULL)
    {
        printf("\n\t\t\t\tTelephone Directory is Empty....\n");
        return;
    }
    printf("Enter the First Name to Edit : ");
    scanf("%s",str);
    ptr=start;
    while(strcmp(ptr->fname,str)!=0)
    {
        ptr=ptr->next;
        if(ptr==NULL)
        break;
    }
    if(ptr!=NULL)
    {
        printf("Enter the new First Name : ");
        scanf("%s",ptr->fname);
        printf("Enter the new Last Name :");
        scanf("%s",ptr->lname);
        printf("Enter the new Phone Number :");
        scanf("%s",ptr->telno);
    }
    else
    {
        printf("No Matching Records Found .......\n");
    }
}
```

Fig. 4 Code for editing an existing contact using C

```
                    TELEPHONE DIRECTORY
1.Add
2.Find
3.Edit
4.Delete
5.Display All Entries
6.Exit
 Enter your choice(1-6):3
Enter the First Name to Edit : Akshad
Enter the new First Name : Anmol
Enter the new Last Name :Narain
Enter the new Phone Number :9187678034
                    TELEPHONE DIRECTORY
1.Add
2.Find
3.Edit
4.Delete
5.Display All Entries
6.Exit
 Enter your choice(1-6):
```

Fig. 5 Output screen where new details for contact are entered

D. Deleting a Contact from the Directory

If the user wants to delete any existing contact from the directory, this function can be used. Two pointer variables "ptr" and "prev" are declared and both are initialized to the top of the list. The first name of the contact to be deleted is obtained from the user. Until the contact is obtained, the list is traversed and "prev" is initialized with "ptr" and "ptr" is moved o the next node. If the name is obtained and it is at the beginning, then "temp" which is another pointer variable is initialized with second node and the variable pointing to the top node is deleted. If the contact is in the middle then "temp" is initialized with node after the node which is pointed by "ptr" and "ptr" variable is deleted. If no match is obtained, "No Matching Records Found" is printed on the screen.

```
                    TELEPHONE DIRECTORY
1.Add
2.Find
3.Edit
4.Delete
5.Display All Entries
6.Exit
 Enter your choice(1-6):4
Enter the First Name to be deleted : Akshad

Deleting Record.....Confirm [y/n]:

-------------------------------------------------------
First Name : Akshad
Last Name : Patel
Phone Number : 7768893202
-------------------------------------------------------

 Record Deleted....
                    TELEPHONE DIRECTORY
1.Add
2.Find
3.Edit
4.Delete
5.Display All Entries
6.Exit
 Enter your choice(1-6):
```

Fig.6. Output screen where a contact is deleted and current list after deletion is shown

E. Displaying all Contacts in the Directory

If the user wants to see all the contacts stored in the directory, this function can be used. A pointer variable "ptr" is initialized to the top of the list. The list is traversed until the last contact is reached and the contacts are displayed on the screen.

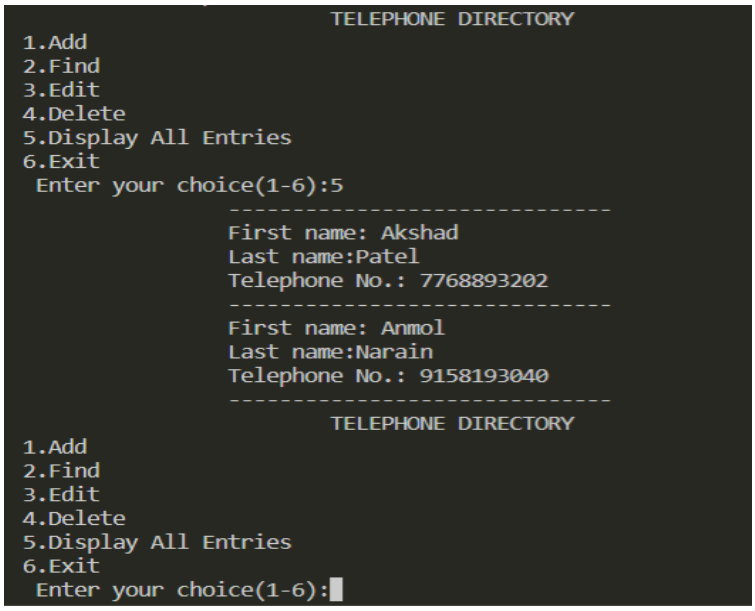

Fig. 7 Output screen displaying all contacts present in the directory

V. Conclusion

Linked List data structure is one of the ways to implement a telephone directory. It allows you to perform different operations on the directory which are mentioned in this document.

A lot of additional features like adding location of the contact, email address of the contact can also be done and operations like finding contacts in a particular area can be achieved and the program's efficiency can be improved.

References

[1] https://codingboost.com/understanding-and-implementing-linked-list-in-c

[2] https://wwwgeeksforgeeks.org/implement-a-phone-directory/