**PHONE BOOK MANAGEMENT**

***Submitted by***

# N Bhavyasri (RA2311027020014)

# P Devadharshini (RA2311027020049)

# Keerthana Ranganathan (RA2311027020061)

#### Under the guidance of

**DR N SUGANTHI**

**(Assistant Professors, Department of Computer Science and Engineering)**

***in partial fulfillment for the award of the degree***

***of***

## BACHELOR OF TECHNOLOGY

***in***

## COMPUTER SCIENCE AND ENGINEERING

***of***

## COLLEGE OF ENGINEERING AND TECHNOLOGY



**RAMAPURAM , CHENNAI-600089**

**OCTOBER 202****4**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

(Deemed to be University Under Section 3 of UGC Act, 1956)

**BONAFIDE CERTIFICATE**

Certified that the Mini Project titled “Phone Book Management” is the bonafide certificate of Bhavyasri, Devadharshini and Keerthana of II Year CSE/Specialization submitted for the course 21CSC201J Data Structures and Algorithms for the Academic Year 2023 – 24 Odd Semester.

##### SIGNATURE

##### Dr N Suganthi

Assistant Professor

Computer Science & Engineering

SRM Institute of Science and Technology Ramapuram, Chennai.

**SCOPE AND OBJECTIVE**

The scope of this phone book management system using a linked list in C covers basic functionalities required to manage and organize contacts efficiently. The system allows users to add, search, display, delete, and sort contacts using a simple command-line interface. It demonstrates fundamental concepts of data structures, algorithms, and memory management in C.

Key points of the project's scope and objective include:

* Basic Contact Management Operations: Users can perform essential operations like adding new contacts, displaying the list of contacts, searching for a contact by name, deleting a contact, and sorting the contacts alphabetically.
* Dynamic Data Handling with Linked Lists: The project uses a linked list to store and manage contacts, allowing for dynamic allocation and resizing as contacts are added or removed.
* Sorting Capability: The program provides functionality to sort contacts alphabetically by name, enhancing the organization of the phone book.
* Command-line Interface (CLI): The project is implemented as a console application, suitable for learning purposes and basic contact management tasks.
* Demonstration of C Programming Concepts: The project illustrates the use of structures, pointers, dynamic memory allocation, string handling, and user interaction in C.

**Problem Statement**

In today's digital age, keeping an organized and easily accessible list of contacts is essential for both personal and professional communication. However, many people still struggle to manage their contacts efficiently due to the lack of simple tools that allow for dynamic and flexible contact management.

Traditional paper-based address books are outdated and cumbersome to maintain, while some digital solutions may be too complex for basic needs or require extensive configuration and setup. Additionally, existing solutions may not offer an easy way to handle frequent operations such as adding, searching, deleting, or sorting contacts in a user-friendly manner**.**

**Problem Description**

Managing a growing list of contacts is a common challenge for many individuals. Whether for personal use or small business purposes, keeping track of phone numbers, names, and other details can become overwhelming without an efficient system. Existing digital contact management solutions may be too complex, requiring a steep learning curve or additional resources, while simpler manual methods such as paper-based address books are inefficient, prone to errors, and difficult to maintain.

This project addresses the problem of creating a simple, user-friendly phone book management system that enables basic operations such as adding, searching, deleting, displaying, and sorting contacts. The solution aims to manage contacts dynamically, allowing for easy updates and flexibility to handle an expanding list of entries without pre-defining its size.

The phone book management system will be implemented as a command-line application written in C, utilizing a linked list to manage the contacts dynamically. The system will provide the following features:

* Add Contact: Users can add new contacts with names and phone numbers.
* Display Contacts: The current list of contacts can be displayed to the user.
* Search Contact: Users can search for a specific contact by name.
* Delete Contact: Contacts can be removed from the list based on the name.
* Sort Contacts: The contacts can be sorted alphabetically by name to improve organization**.**

**IMPLEMENTATION**

**CODE**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>// Structure to represent a contact

typedef struct Contact {

char name[50];

char phone[15];

struct Contact\* next;

} Contact;// Function to create a new contact

Contact\* createContact(const char\* name, const char\* phone) {

Contact\* newContact = (Contact\*)malloc(sizeof(Contact));

strcpy(newContact->name, name);

strcpy(newContact->phone, phone);

newContact->next = NULL;

return newContact;

}

// Function to add a contact to the phone book

Contact\* addContact(Contact\* head, const char\* name, const char\* phone) {

Contact\* newContact = createContact(name, phone);

if (!head) {

return newContact; }

Contact\* temp = head;

while (temp->next) {

temp = temp->next;

}

temp->next = newContact;

return head;

}// Function to display all contacts

void displayContacts(Contact\* head) {

if (!head) {

printf("Phone book is empty.\n");

return;

}

Contact\* temp = head;

while (temp) {

printf("Name: %s, Phone: %s\n", temp->name, temp->phone);

temp = temp->next;

}

}// Function to search for a contact by name

void searchContact(Contact\* head, const char\* name) {

Contact\* temp = head;

while (temp) {

if (strcmp(temp->name, name) == 0) {

printf("Contact found: Name: %s, Phone: %s\n", temp->name, temp->phone);

return;

}

temp = temp->next;

}

printf("Contact not found.\n");

}// Function to delete a contact by name

Contact\* deleteContact(Contact\* head, const char\* name) {

if (!head) {

printf("Phone book is empty.\n");

return NULL;

}

Contact\* temp = head;

Contact\* prev = NULL;

// If the contact to be deleted is the head

if (strcmp(head->name, name) == 0) {

head = head->next;

free(temp);

printf("Contact deleted.\n");

return head;

} // Search for the contact to be deleted

while (temp && strcmp(temp->name, name) != 0) {

prev = temp;

temp = temp->next;

} // If contact not found

if (!temp) {

printf("Contact not found.\n");

return head;

} // Unlink the contact from the linked list

prev->next = temp->next;

free(temp);

printf("Contact deleted.\n");

return head;}// Function to sort the contacts alphabetically by name

Contact\* sortContacts(Contact\* head) {

if (!head || !head->next) {

return head;

}

Contact\* current = head;

Contact\* index = NULL;

char tempName[50];

char tempPhone[15];// Bubble sort algorithm for linked list

while (current) {

index = current->next;

while (index) {

if (strcmp(current->name, index->name) > 0) {

// Swap the names

strcpy(tempName, current->name);

strcpy(current->name, index->name);

strcpy(index->name, tempName); // Swap the phone numbers

strcpy(tempPhone, current->phone);

strcpy(current->phone, index->phone);

strcpy(index->phone, tempPhone); }

index = index->next;

}

current = current->next;

}

printf("Contacts sorted alphabetically.\n");

return head;

}// Function to free the memory used by the linked list

void freeContacts(Contact\* head) {

Contact\* temp;

while (head) {

temp = head;

head = head->next;

free(temp);

}

}

int main() {

Contact\* phoneBook = NULL;

int choice;

char name[50], phone[15];

while (1) {

printf("\nPhone Book Management System\n");

printf("1. Add Contact\n");

printf("2. Display Contacts\n");

printf("3. Search Contact\n");

printf("4. Delete Contact\n");

printf("5. Sort Contacts\n");

printf("6. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

getchar(); // Consume the newline character left by scanf

switch (choice) {

case 1:

printf("Enter name: ");

fgets(name, sizeof(name), stdin);

name[strcspn(name, "\n")] = '\0'; // Remove newline character

printf("Enter phone number: ");

fgets(phone, sizeof(phone), stdin);

phone[strcspn(phone, "\n")] = '\0'; // Remove newline character

phoneBook = addContact(phoneBook, name, phone);

printf("Contact added successfully.\n");

break;

case 2:

displayContacts(phoneBook);

break;

case 3:

printf("Enter name to search: ");

fgets(name, sizeof(name), stdin);

name[strcspn(name, "\n")] = '\0'; // Remove newline character

searchContact(phoneBook, name);

break;

case 4:

printf("Enter name to delete: ");

fgets(name, sizeof(name), stdin);

name[strcspn(name, "\n")] = '\0'; // Remove newline character

phoneBook = deleteContact(phoneBook, name);

break;

case 5:

phoneBook = sortContacts(phoneBook);

break;

case 6:

freeContacts(phoneBook);

printf("Exiting program.\n");

exit(0);

default:

printf("Invalid choice. Please try again.\n");

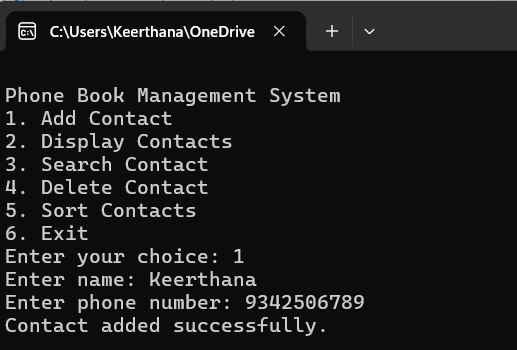
}

}

return 0;

}

**OUTPUT AND SCREENSHOTS**

****

