

SCHOOL OF COMPUTING

BACHELOR OF COMPUTER SCIENCE ENGINEERING

LOVELY PROFESSIONAL UNIVERSITY, PHAGWARA, PUNJAB, INDIA.

**CSE101:COMPUTERPROGRAMMING**

**Topic := phonebook management system**

|  |  |
| --- | --- |
| **SUBMITTED BY** | **SUBMITTED TO** |
| NIPEKSH CHAUDHARY | Amandeep Kaur |
| Reg. No.: - 12207550 | Department of CSE |

Roll No.: - 58

Section: - K22EN

# project description

This project aims to develop a phonebook management system in C, which allows users to store and manage their contacts efficiently. The system will be implemented using a struct to represent each contact, with fields for name and phone number.

The user interface will be menu-driven, allowing users to perform various actions on their phonebook. These actions will include:

Add a new contact: This option will prompt the user to enter a new name and phone number, and then add the contact to the phonebook.

Delete a contact: This option will prompt the user to enter the name of the contact they wish to delete, and then delete that contact from the phonebook.

Search for a contact: This option will prompt the user to enter the name of the contact they wish to search for, and then search the phonebook for a match. If a match is found, the contact's name and phone number will be displayed.

Print all contacts: This option will print a list of all contacts currently stored in the phonebook, along with their names and phone numbers.

Exit the program: This option will allow the user to exit the program.

To implement these features, the program will use several functions, including:

A function to add a new contact to the phonebook.

A function to delete a contact from the phonebook.

A function to search for a contact in the phonebook.

A function to print all contacts currently stored in the phonebook.

A main function that presents the user with a menu of options and calls the appropriate function based on the user's selection.

# module explanation

The program consists of several functions that work together to implement a simple phonebook management system.

The main function initializes an array of phonebook structs, tracks the number of contacts currently stored in the phonebook, and provides a menu-driven interface for the user to perform various actions on the phonebook. The menu allows the user to add new contacts, delete existing contacts, search for a specific contact, print all contacts, or exit the program.

The addContact function prompts the user to enter a new name and number for a contact, and then stores that information in the next available slot in the phonebook array.

The deleteContact function prompts the user to enter the name of a contact they want to delete, and then searches the phonebook array for a matching name. If a match is found, the function shifts all subsequent entries up by one to fill the gap left by the deleted contact.

The searchContact function prompts the user to enter the name of a contact they want to search for, and then searches the phonebook array for a matching name. If a match is found, the function prints the contact's name and number.

The printContacts function simply iterates over the phonebook array and prints the name and number of each contact in turn.

Overall, this program provides a basic example of how to implement a simple phonebook management system in C using a struct and several supporting functions.

# programming code

#include <stdio.h>

#include <string.h>

struct phonebook { char name[50]; char number[20];

};

void addContact(struct phonebook \*pb, int \*count) { printf("Enter name: "); scanf("%s", pb[\*count].name); fflush(stdin); printf("Enter number: "); scanf("%s", pb[\*count].number);

(\*count)++; printf("Contact added successfully.\n");

}

void deleteContact(struct phonebook \*pb, int \*count) {

int i, j;

char name[50]; printf("Enter name: "); scanf("%s", name); for (i = 0; i < \*count; i++) { if (strcmp(pb[i].name, name) == 0) { for (j = i; j < \*count - 1; j++) { strcpy(pb[j].name, pb[j+1].name); strcpy(pb[j].number, pb[j+1].number);

}

(\*count)--; printf("Contact deleted successfully.\n");

return;

}

}

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

}

void searchContact(struct phonebook \*pb, int count) {

int i;

char name[50]; printf("Enter name: "); scanf("%s", name); for (i = 0; i < count; i++) { if (strcmp(pb[i].name, name) == 0) { printf("Name: %s\n", pb[i].name); printf("Number: %s\n", pb[i].number);

return;

}

}

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

}

void printContacts(struct phonebook \*pb, int count) {

int i;

printf("Contacts:\n"); for (i = 0; i < count; i++) { printf("%s\t%s\n", pb[i].name, pb[i].number);

}

}

int main() { struct phonebook pb[100]; int count = 0; int choice; while (1) {

printf("\nPhone Book Management System\n"); printf("1. Add contact\n"); printf("2. Delete contact\n"); printf("3. Search contact\n"); printf("4. Print contacts\n"); printf("5. Exit\n"); printf("Enter your choice: "); scanf("%d", &choice); switch (choice) { case 1:

addContact(pb, &count);

break; case 2:

deleteContact(pb, &count);

break; case 3:

searchContact(pb, count);

break; case 4: printContacts(pb, count); break; case 5:

return 0; default:

printf("Invalid choice.\n");

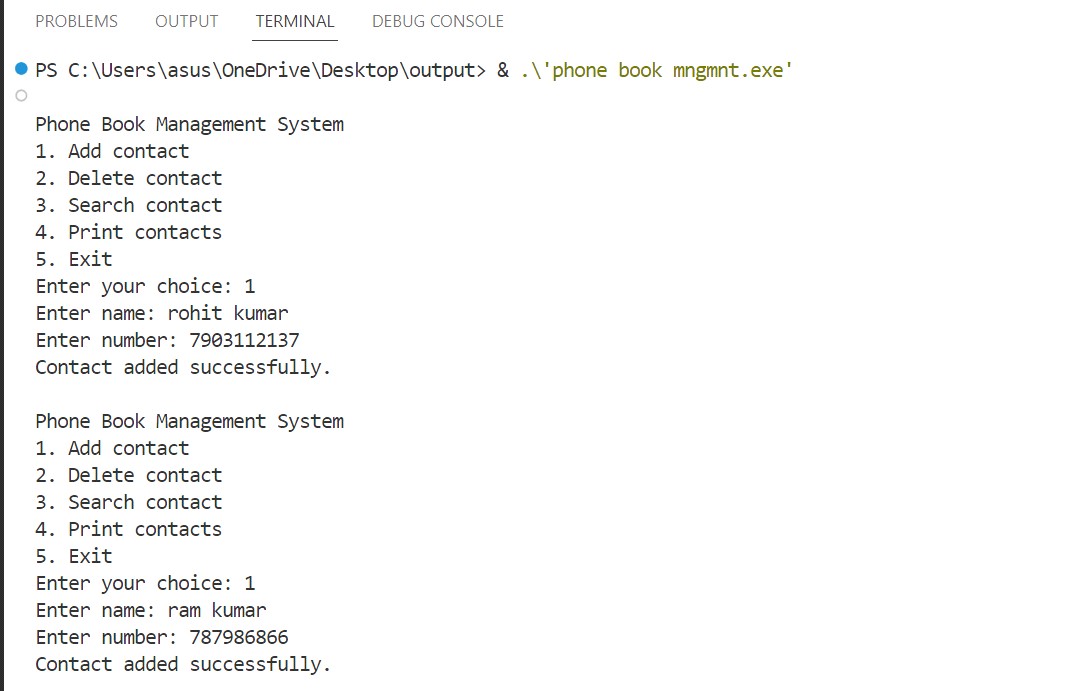
}

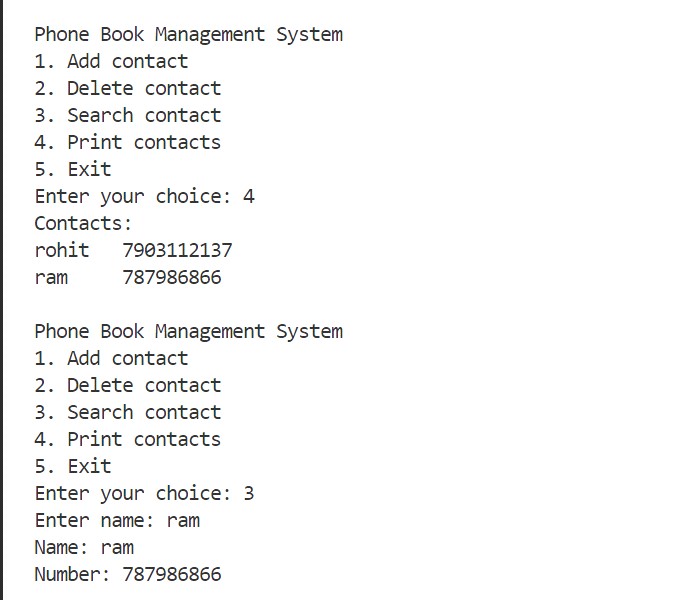
}

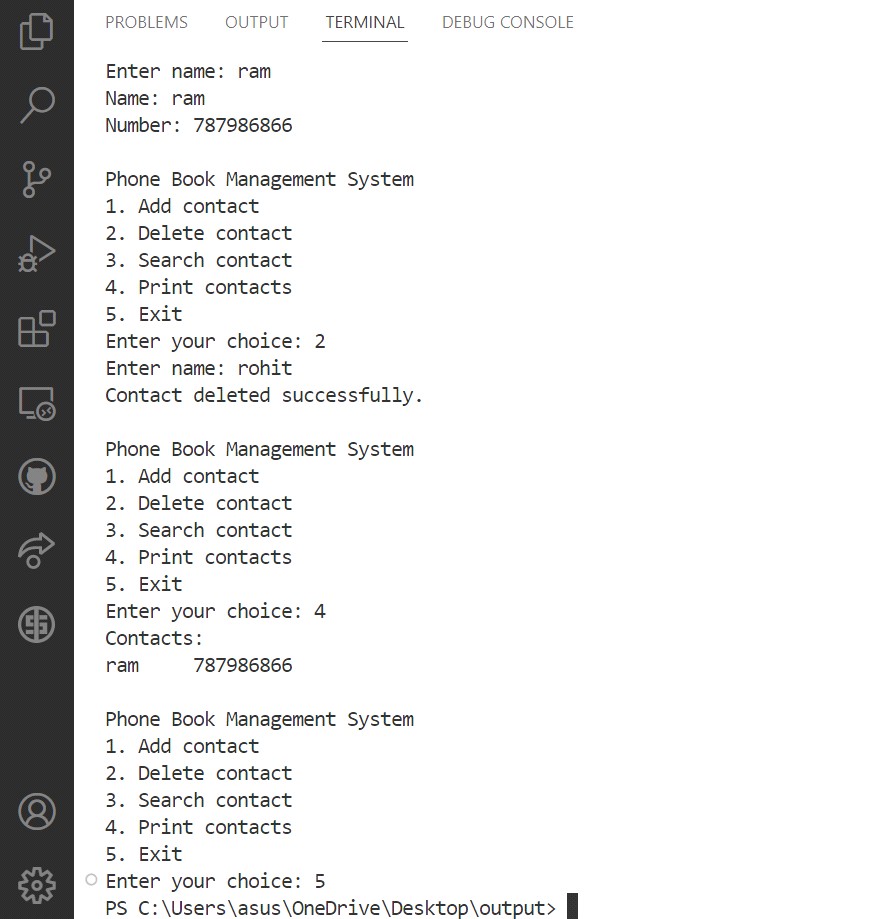
return 0;

}

# output snapshot







Level 0 DFD

