**Project\_1: Address Book**

**Main.c**

/\*

Name : Nestin Gregorios Sunny

Date : 03.07.2025

Description :

Project 1 - Implement Address-Book using C programing

Functions Demonstrated:

1. Create Contact into the Address Book

2. Search an already existing contact

3. Edit an already existing contact

4. Delete an already existing contact

5. List all contacts in alphabetical order

6. Save and Exit to .csv file

\*/

#include "contact.h"

int main()

{

    int choice;

    AddressBook addressBook;

    addressBook.contactCount = 0;

    initialize(&addressBook); // Initialize the address book

    do

    {

        printf("\nAddress Book Menu:\n");

        printf("1. Create contact\n");

        printf("2. Search contact\n");

        printf("3. Edit contact\n");

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

        printf("5. List all contacts\n");

        printf("6. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice)

        {

            case 1:

            createContact(&addressBook);

            break;

            case 2:

            searchContact(&addressBook);

            break;

            case 3:

            editContact(&addressBook);

            break;

            case 4:

            deleteContact(&addressBook);

            break;

            case 5:

            listContacts(&addressBook);

            break;

            case 6:

            printf("Saving and Exiting...\n");

            saveAndExit(&addressBook);

            break;

            default:

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

        }

    } while (choice != 6);

    //cleanup(); // Cleanup any resources before exiting

    return 0;

}

**Contact.h**

#ifndef CONTACT\_H

#define CONTACT\_H

#include<stdio.h>

#define MAX\_CONTACTS 100

typedef struct Contact

{

    char name[50];

    char phone[20];

    char email[50];

} Contact;

typedef struct

{

    Contact contacts[MAX\_CONTACTS];

    int contactCount;

} AddressBook;

void createContact(AddressBook \*addressBook);

int validate\_phone(AddressBook \*addressBook, char \*number, int flag);

int validate\_email(AddressBook \*addressBook, char \*mail, int flag);

void searchContact(AddressBook \*addressBook);

void editContact(AddressBook \*addressBook);

int contact\_search(AddressBook \*addressBook, int s\_choice, int \*found\_i);

void deleteContact(AddressBook \*addressBook);

void listContacts(AddressBook \*addressBook);

void swapContacts(Contact \*a, Contact \*b);

void initialize(AddressBook \*addressBook);

void saveContactsToFile(AddressBook \*addressBook);

void saveAndExit(AddressBook \*addressBook);

#endif

**Contact.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "contact.h"

#include "file.h"

#include "populate.h"

// function to swap in order to list our contacts in alphabetical order

void swapContacts(Contact \*a, Contact \*b)

{

    Contact temp = \*a;

    \*a = \*b;

    \*b = temp;

}

void listContacts(AddressBook \*addressBook)

{

    /\* Define the logic for print the contacts \*/

    int i, j;

    //checking whether data is inserted or not

    if (addressBook -> contactCount == 0)

    {

        printf("Address Book is Empty!!!\n");

        return ;

    }

    //Bubble sorting to print in alphabetical order

    for (i = 0; i < addressBook -> contactCount - 1; i++)

    {

        for (j = 0; j < addressBook -> contactCount - i - 1; j++)

        {

            //condition to whether contacts are in alphabetical order

            if (strcasecmp(addressBook -> contacts[j].name, addressBook -> contacts[j + 1].name) > 0)

            {

                swapContacts(&addressBook -> contacts[j], &addressBook -> contacts[j + 1]);

            }

        }

    }

    //Listing every contacts in address book in alphabetical order

    printf("Contacts in Address Book\n");

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    for (i = 0; i < addressBook -> contactCount; i++)

    {

        printf("Contact %d :\n", i + 1);

        printf("Name: %s\n", addressBook -> contacts[i].name);

        printf("Phone: %s\n", addressBook -> contacts[i].phone);

        printf("Email: %s\n\n", addressBook -> contacts[i].email);

    }

}

//initializing dummyContacts from populate.c

void initialize(AddressBook \*addressBook)

{

    addressBook->contactCount = 0;

    //populateAddressBook(addressBook);

    // Load contacts from file during initialization (After files)

    loadContactsFromFile(addressBook);

}

void saveAndExit(AddressBook \*addressBook)

{

    saveContactsToFile(addressBook); // Save contacts to file

    exit(EXIT\_SUCCESS); // Exit the program

}

void createContact(AddressBook \*addressBook)

{

    /\* Define the logic to create a Contacts \*/

    //condition to whether the addressbook is FULL, if FULL can't add anymore contact

    if (addressBook -> contactCount >= MAX\_CONTACTS)

    {

        printf("Address Book is FULL, Cannot add more contacts");

        return;

    }

    // Read the name from the user

    printf("Enter contact name : ");

    scanf(" %[^\n]", addressBook -> contacts[addressBook -> contactCount].name);

    // Read the contact

    char number[15];

    printf("Enter phone number : ");

    scanf(" %[^\n]", number);

    // - Check whether the count is 10 or not

    // - Check all 10 characters are digits or not.

    // - Check the given number is already exist or not

    while(!validate\_phone(addressBook, number, 1))

    {

        printf("Enter phone number : ");

        scanf(" %[^\n]", number);

    }

    //if validation is done add phone number to contacts[]

    strcpy(addressBook -> contacts[addressBook -> contactCount].phone, number);

    char mail[20];

    printf("Enter email id : ");

    scanf(" %[^\n]", mail);

    // Read the email ID

    // - Check whether the character array contains lowercase, digits and special characters or not

    // - Check whether char - @ and .com is present or not

    while(!validate\_email(addressBook, mail, 1))

    {

        printf("Enter email id : ");

        scanf(" %[^\n]", mail);

    }

    //if validation is done add email id to contacts[]

    strcpy(addressBook -> contacts[addressBook -> contactCount].email, mail);

    // Increment the contact count.

    addressBook -> contactCount++;

    printf("Congrats, New Contact Created!!!");

}

int validate\_phone(AddressBook \*addressBook, char \*number, int flag)

{

    int i = 0;

    //checking whether user entered 10 characters or not

    if(strlen(number) != 10)

    {

        printf("Invalid phone number!! Enter Again...\n");

        return 0;

    }

    //checking whether every characters are numerical

    for(i = 0; i < 10; i++)

    {

        if (number[i] < '0' || number[i] > '9')

            {

                printf("Invalid phone number!! Enter Again...\n");

                return 0;

            }

    }

    //checking whether same number is entered earlier or not

    if(flag == 1)

    {

        for (i = 0; i < addressBook -> contactCount; i++)

        {

            if (strcmp(addressBook -> contacts[i].phone, number) == 0)

            {

                printf("Phone number already exist in address book!!!\n");

                return 0;

            }

        }

    }

    else

    {

        return 1;

    }

    return 1;

}

int validate\_email(AddressBook \*addressBook, char \*mail, int flag)

{

    int i;

    int at\_i = -1;

    int dot\_i = -1;

    //checking position of '@' and '.'

    for (i = 0; i < strlen(mail); i++)

    {

        if (mail[i] == '@')

        {

            at\_i = i;

        }

        else if (mail[i] == '.')

        {

            dot\_i = i;

        }

    }

    // check if email ends with ".com"

    if (strlen(mail) < 4 || strcmp(mail + strlen(mail) - 4, ".com") != 0)

    {

        printf("Invalid Email ID!!! Email should end with .com\n");

        return 0;

    }

    //checking for invalid case

    if (at\_i == -1 || dot\_i == -1 || dot\_i < at\_i || dot\_i > strlen(mail) - 4)

    {

        printf("Invalid Email ID!!! Enter Again..\n");

        return 0;

    }

    //checking for duplicate

    //this validation is only needed while creating a contact

    if(flag == 1)

    {

        for (i = 0; i < addressBook -> contactCount; i++)

        {

            if (strcmp(addressBook -> contacts[i].email, mail) == 0)

            {

                printf("Email already exist in address book!!!\n");

                return 0;

            }

        }

    }

    else

    {

        return 1;

    }

    return 1;

}

void searchContact(AddressBook \*addressBook)

{

    /\* Define the logic for search \*/

    int choose, found = 0;

    char search[50];

    //Menu Card

    printf("What are you searching : \n");

    printf("1.Name\n");

    printf("2.Phone Number\n");

    printf("3.Email\n");

    scanf("%d", &choose);

    //to search a contact by that specified case

    switch (choose)

    {

    case 1:

        printf("Enter contact name : ");

        scanf(" %[^\n]", search);

        int matched = 0;

        for (int i = 0; i < addressBook->contactCount; i++)

        {

            char \*space = strchr(addressBook -> contacts[i].name, ' ');

            if (space != NULL)

            {

                //compare 1st name

                if (strncasecmp(addressBook-> contacts[i].name, search, strlen(search)) == 0)

                {

                    printf("\nContact found:\n");

                    printf("Name: %s\n", addressBook->contacts[i].name);

                    printf("Phone: %s\n", addressBook->contacts[i].phone);

                    printf("Email: %s\n", addressBook->contacts[i].email);

                    matched++;

                }

            }

            else

            {

                //Compare entire name

                if (strcasecmp(addressBook-> contacts[i].name, search) == 0)

                {

                    printf("\nContact found:\n");

                    printf("Name: %s\n", addressBook->contacts[i].name);

                    printf("Phone: %s\n", addressBook->contacts[i].phone);

                    printf("Email: %s\n", addressBook->contacts[i].email);

                    matched++;

                }

            }

        }

        if (matched == 0)

        {

            printf("Contact not Found!!!\n");

        }

        else

        {

            printf("%d contact(s) found with name '%s'\n", matched, search);

        }

        break;

    case 2:

        printf("Enter phone number : ");

        scanf(" %[^\n]", search);

        while(!validate\_phone(addressBook, search, 2))

        {

            printf("Enter phone number : ");

            scanf(" %[^\n]", search);

        }

        for (int i = 0; i < addressBook->contactCount; i++)

        {

            if (strcmp(addressBook->contacts[i].phone, search) == 0)

            {

                printf("Contact found:\n");

                printf("Name: %s\n", addressBook->contacts[i].name);

                printf("Phone: %s\n", addressBook->contacts[i].phone);

                printf("Email: %s\n", addressBook->contacts[i].email);

                found = 1;

            }

        }

        if (!found)

        {

            printf("Contact not Found!!!\n");

        }

        break;

    case 3:

        printf("Enter email ID : ");

        scanf(" %[^\n]", search);

        while(!validate\_email(addressBook, search, 2))

        {

            printf("Enter email ID : ");

            scanf(" %[^\n]", search);

        }

        for (int i = 0; i < addressBook->contactCount; i++)

        {

            if (strcmp(addressBook->contacts[i].email, search) == 0)

            {

                printf("Contact found:\n");

                printf("Name: %s\n", addressBook->contacts[i].name);

                printf("Phone: %s\n", addressBook->contacts[i].phone);

                printf("Email: %s\n", addressBook->contacts[i].email);

                found = 1;

            }

        }

        if (!found)

        {

            printf("Contact not Found!!!\n");

        }

        break;

    default:

        printf("Wrong Choice!!");

        break;

    }

}

void editContact(AddressBook \*addressBook)

{

    /\* Define the logic for Editcontact \*/

    if (addressBook -> contactCount == 0)

    {

        printf("Address Book is empty!!!\n");

        return;

    }

    int s\_choice, e\_choice;

    int i, index;

    int found\_i[MAX\_CONTACTS];

    int f\_count = 0;

    char edit[50];

    printf("Search By : \n");

    printf("1.Name\n");

    printf("2.Phone Number\n");

    printf("3.Email\n");

    scanf("%d", &s\_choice);

    //calling a seperate search function to avoid confusion with orginal one

    f\_count = contact\_search(addressBook, s\_choice, found\_i);

    //condition for unavailable case

    if(f\_count == 0)

    {

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

        return;

    }

    //List contacts if multiple contacts are availble

    if(f\_count > 1)

    {

        printf("Multiple contacts Found. Please select one : \n");

        for(i = 0; i < f\_count; i++)

        {

            printf("%d. Name: %s, Phone: %s, Email: %s\n", i + 1, addressBook -> contacts[found\_i[i]].name, addressBook -> contacts[found\_i[i]].phone, addressBook -> contacts[found\_i[i]].email);

        }

        int serial;

        printf("Enter serial number : ");

        scanf("%d", &serial);

        if(serial < 1 || serial > f\_count)

        {

            printf("Invalid Serial Number!!!\n");

            return;

        }

        index = found\_i[serial - 1];

    }

    //if only one name is there, go to else condition

    else

    {

        index = found\_i[0];

    }

    //Confirming with user which field they have to modify

    printf("What do you want to edit?\n");

    printf("1.Name\n");

    printf("2.Phone Number\n");

    printf("3.Email\n");

    scanf("%d", &e\_choice);

    switch (e\_choice)

    {

        case 1:

            //editing already existing name to new one

            printf("Enter new name : ");

            scanf(" %[^\n]", edit);

            strcpy(addressBook->contacts[index].name, edit);

            printf("Name updated successfully.\n");

            break;

        case 2:

            //editing already existing phone number to new one

            printf("Enter new phone number : ");

            scanf(" %[^\n]", edit);

            while(!validate\_phone(addressBook, edit, 3))

            {

                printf("Enter new phone number : ");

                scanf(" %[^\n]", edit);

            }

            strcpy(addressBook->contacts[index].phone, edit);

            printf("Phone Number updated successfully.\n");

            break;

        case 3:

            //editing already existing email to new one

            printf("Enter new email ID : ");

            scanf(" %[^\n]", edit);

            while(!validate\_email(addressBook, edit, 3))

            {

                printf("Enter new email ID : ");

                scanf(" %[^\n]", edit);

            }

            strcpy(addressBook->contacts[index].email, edit);

            printf("Email ID updated successfully.\n");

            break;

        default:

            printf("Invalid Choice !!!\n");

            break;

    }

}

//defining a seperate search function to avoid call void searchContact()

int contact\_search(AddressBook \*addressBook, int s\_choice, int \*found\_i)

{

    int i, found = 0;

    char search[50];

    switch (s\_choice)

    {

        case 1:

            printf("Enter contact name : ");

            break;

        case 2:

            printf("Enter phone number : ");

            break;

        case 3:

            printf("Enter email : ");

            break;

    }

    scanf(" %[^\n]", search);

    for (i = 0; i < addressBook -> contactCount; i++)

    {

        switch (s\_choice)

        {

            case 1:

                if (strcmp(addressBook->contacts[i].name, search) == 0)

                {

                    printf("Contact found:\n");

                    printf("Name: %s\n", addressBook->contacts[i].name);

                    printf("Phone: %s\n", addressBook->contacts[i].phone);

                    printf("Email: %s\n", addressBook->contacts[i].email);

                    found\_i[found++] = i;

                }

                break;

            case 2:

                if (strcmp(addressBook->contacts[i].phone, search) == 0)

                {

                    printf("Contact found:\n");

                    printf("Name: %s\n", addressBook->contacts[i].name);

                    printf("Phone: %s\n", addressBook->contacts[i].phone);

                    printf("Email: %s\n", addressBook->contacts[i].email);

                    found\_i[found++] = i;

                }

                break;

        case 3:

                if (strcmp(addressBook->contacts[i].email, search) == 0)

                {

                    printf("Contact found:\n");

                    printf("Name: %s\n", addressBook->contacts[i].name);

                    printf("Phone: %s\n", addressBook->contacts[i].phone);

                    printf("Email: %s\n", addressBook->contacts[i].email);

                    found\_i[found++] = i;

                }

                break;

        }

    }

    return found;

}

void deleteContact(AddressBook \*addressBook)

{

    /\* Define the logic for deletecontact \*/

    //condition for Empty Address Book

    if (addressBook -> contactCount == 0)

    {

        printf("Address Book is empty!!!\n");

        return;

    }

    int s\_choice;

    int i, index;

    int found\_i[MAX\_CONTACTS];

    int f\_count = 0;

    //Menu to search

    printf("Search By : \n");

    printf("1.Name\n");

    printf("2.Phone Number\n");

    printf("3.Email\n");

    scanf("%d", &s\_choice);

    //calling search function to check whether given input is existing or not

    f\_count = contact\_search(addressBook, s\_choice, found\_i);

    if(f\_count == 0)

    {

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

        return;

    }

    if(f\_count > 1)

    {

        printf("Multiple contacts found. Please select one:\n");

        for(i = 0; i <f\_count; i++)

        {

            printf("%d. Name: %s, Phone: %s, Email: %s\n", i + 1, addressBook -> contacts[found\_i[i]].name, addressBook -> contacts[found\_i[i]].phone, addressBook -> contacts[found\_i[i]].email);

        }

        int serial;

        printf("Enter serial number : ");

        scanf("%d", &serial);

        if(serial < 1 || serial > f\_count)

        {

            printf("Invalid serial number!!!\n");

            return;

        }

        char ch;

        //reconfirming with user whether they want to delete that selected contact

        printf("Are you sure you want to delete this contact (y/n) ?");

        scanf(" %c", &ch);

        index = found\_i[serial - 1];

        if (ch == 'Y' || ch == 'y')

        {

            for (i = index; i < addressBook -> contactCount - 1; i++)

            {

                //overwriting to previous contacts

                strcpy(addressBook -> contacts[i].name, addressBook -> contacts[i + 1].name);

                strcpy(addressBook -> contacts[i].phone, addressBook -> contacts[i + 1].phone);

                strcpy(addressBook -> contacts[i].email, addressBook -> contacts[i + 1].email);

            }

            //deleting unwanted contactCount to save memory

            addressBook -> contactCount--;

            printf("Contact Deleted Successfully.\n");

        }

        else

        {

            printf("Deletion Cancelled!!!\n");

        }

    }

    //condition for only unique contact searched

    else

    {

        index = found\_i[0];

        char ch;

        printf("Are you sure you want to delete this contact (y/n) ?");

        scanf(" %c", &ch);

        if (ch == 'Y' || ch == 'y')

        {

            for (i = index; i < addressBook -> contactCount - 1; i++)

            {

                //overwriting to previous contacts

                strcpy(addressBook -> contacts[i].name, addressBook -> contacts[i + 1].name);

                strcpy(addressBook -> contacts[i].phone, addressBook -> contacts[i + 1].phone);

                strcpy(addressBook -> contacts[i].email, addressBook -> contacts[i + 1].email);

            }

            //deleting unwanted contactCount to save memory

            addressBook -> contactCount--;

            printf("Contact Deleted Successfully.\n");

        }

        //if any charcater other than 'Y' or 'y' is pressed it will cancel the process

        else

        {

            printf("Deletion Cancelled!!!\n");

        }

    }

}

**Populate.h**

void populateAddressBook(AddressBook\* addressBook);

**Populate.c**

#include "contact.h"

// Dummy contact data

static Contact dummyContacts[] = {

    {"John Doe", "1234567890", "john@example.com"},

    {"Alice Smith", "0987654321", "alice@example.com"},

    {"Bob Johnson", "1112223333", "bob@company.com"},

    {"Carol White", "4445556666", "carol@company.com"},

    {"David Brown", "7778889999", "david@example.com"},

    {"Eve Davis", "6665554444", "eve@example.com"},

    {"Frank Miller", "3334445555", "frank@example.com"},

    {"Grace Wilson", "2223334444", "grace@example.com"},

    {"Hannah Clark", "5556667777", "hannah@example.com"},

    {"Ian Lewis", "8889990000", "ian@example.com"}

};

void populateAddressBook(AddressBook\* addressBook)

{

    //printf("%zu\n", sizeof(dummyContacts));

    int numDummyContacts = sizeof(dummyContacts) / sizeof(dummyContacts[0]);

    for (int i = 0; i < numDummyContacts; i++) {

        addressBook->contacts[addressBook->contactCount++] = dummyContacts[i];

    }

}

**File.h**

#ifndef FILE\_H

#define FILE\_H

#include "contact.h"

void saveContactsToFile(AddressBook \*addressBook);

void loadContactsFromFile(AddressBook \*addressBook);

#endif

**File.c**

#include <stdio.h>

#include "file.h"

void saveContactsToFile(AddressBook \*addressBook)

{

  FILE \*fp;

  int i;

  //open file in write mode

  fp = fopen("addressbook.csv", "w");

  fprintf(fp, "#%d\n", addressBook -> contactCount);

  //store contacts into file

  for (i = 0; i < addressBook -> contactCount; i++)

  {

    fprintf(fp,"%s,%s,%s\n", addressBook -> contacts[i].name,addressBook -> contacts[i].phone, addressBook -> contacts[i].email);

  }

  //closing file

  fclose(fp);

}

void loadContactsFromFile(AddressBook \*addressBook)

{

  FILE \*fp;

  int i;

  //open file in read mode

  fp = fopen("addressbook.csv", "r");

  fscanf(fp, "#%d\n", &addressBook -> contactCount);

  //read contacts from file

  for (i = 0; i < addressBook -> contactCount; i++)

  {

    fscanf(fp, " %[^,], %[^,], %[^\n]\n", addressBook -> contacts[i].name, addressBook -> contacts[i].phone, addressBook -> contacts[i].email);

  }

  //closing file

  fclose(fp);

}