Project Title: Simple Address Book

Submitted by: 1. Divyam Bansal (UID: 24BCD10096); 2. Manik Pathak (UID:24BCD10091)

Introduction

We have completed a project in C programming that demonstrates fundamental concepts in file handling, data structures, and input validation. This project, titled "Simple Address Book," has provided me with hands-on experience in developing an application for managing and storing contact information, including functions for adding, updating, deleting, and searching contacts. Throughout the project, we have implemented error handling and ensured fundamental input validation.

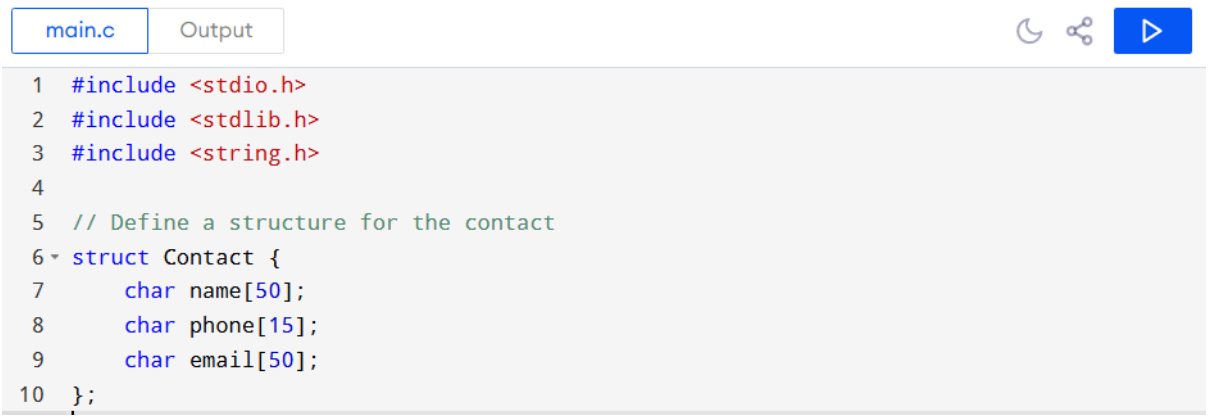
Project Description

The Simple Address Book application is designed to manage contacts effectively. Each contact entry includes essential fields such as name, phone number, and email address. We have structured the program to allow users to add new contacts, update existing entries, delete contacts, and search for specific contacts by name or other parameters. Contact details are stored using file handling, which preserves the information even after the program is closed.

Code Implementation

1. Structure Definition

To organize the contact information, we have used a struct in C. This structure includes fields for the contact's name, phone number, and email. We have chosen these fields as they represent the most basic yet important details for any address book.



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Define a structure for the contact
struct Contact {
    char name[50];
    char phone[15];
    char email[50];
};
```

The struct Contact defines three fields:

- name: Stores the name of the contact (up to 49 characters).
- phone: Stores the phone number, using a string to allow flexible formatting.
- email: Stores the email address.

2. Adding a New Contact

We have implemented the function addContact to add a new contact to the address book. The function prompts the user for details, validates the inputs, and then appends the contact to a file.

```c
void addContact() {
    struct Contact c;
    FILE *fp = fopen("addressbook.dat", "ab");

    printf("Enter name: ");
    fgets(c.name, sizeof(c.name), stdin);
    printf("Enter phone number: ");
    fgets(c.phone, sizeof(c.phone), stdin);
    printf("Enter email: ");
    fgets(c.email, sizeof(c.email), stdin);

    // Validate input (basic example, can be expanded)
    if (strlen(c.phone) < 10) {
        printf("Invalid phone number!\n");
        fclose(fp);
        return;
    }

    fwrite(&c, sizeof(struct Contact), 1, fp);
    printf("Contact added successfully.\n");
    fclose(fp);
}
```

the addContact function:

1. Opens the addressbook.dat file in append-binary mode.
2. Prompts the user for contact details.
3. Validates the phone number to ensure it meets a basic length requirement.
4. Writes the contact information to the file and closes the file.

3. Storing Contact Details with File Processing

To ensure data persistence, we have utilized file processing to store contact details in a file named addressbook.dat. This file allows the program to save contact information permanently, so it remains available even after the program closes. Every contact operation—such as adding, updating, and deleting—is designed to read from and write to this file, keeping the data synchronized.

Example of opening the file for adding a contact:

```c
FILE *fp = fopen("addressbook.dat", "ab");
```

In this code, the file is opened in append-binary mode (ab), which allows new contact records to be added at the end of the file. Using file processing in this way has been essential for managing the contact data effectively and providing a persistent storage solution for the address book application.

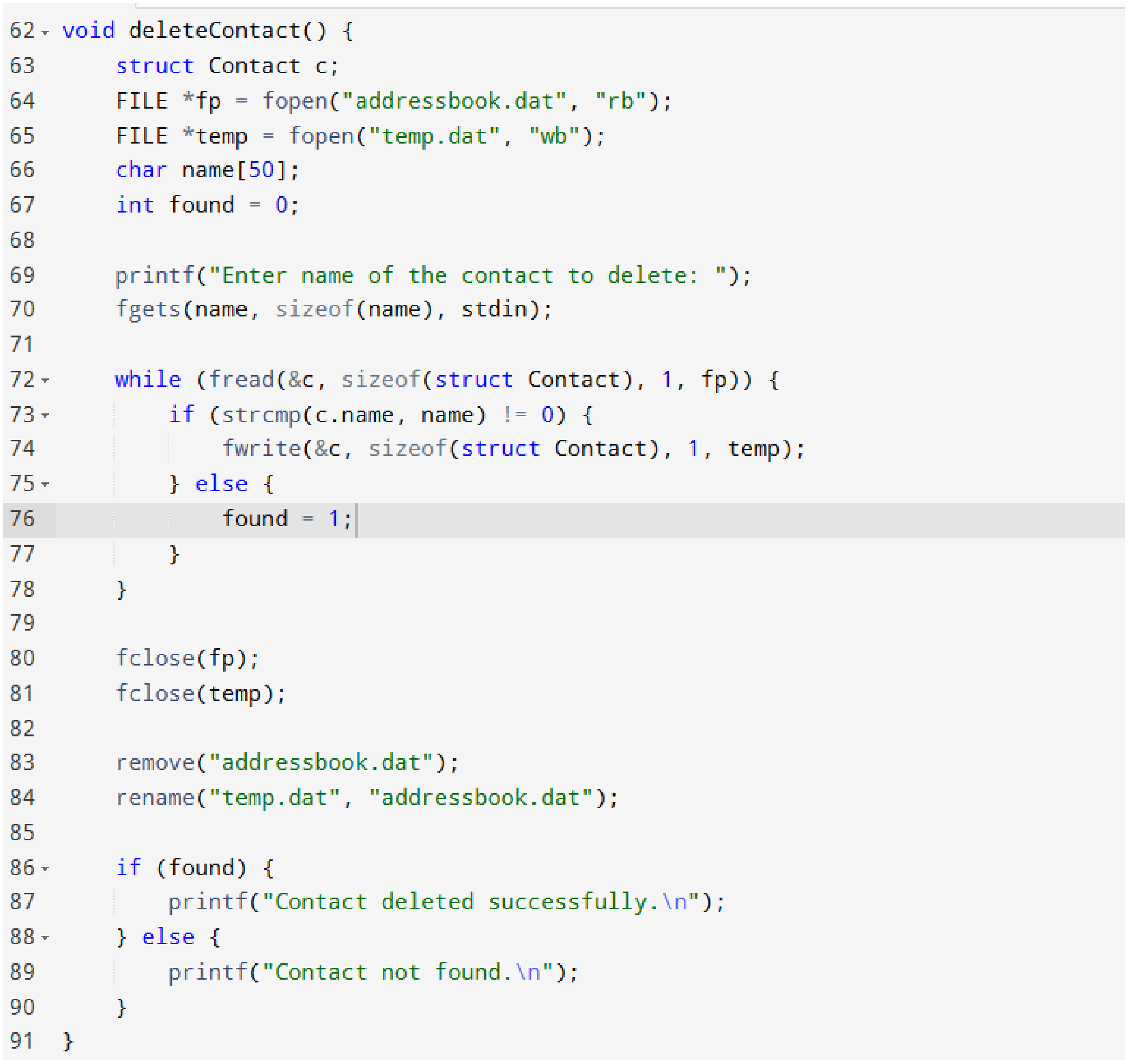
4. Updating a Contact

To update an existing contact, we have used the updateContact function. It reads the file line-by-line, searching for a specific contact name. Once found, it allows the user to enter updated details.

```c
void updateContact() {
    struct Contact c;
    FILE *fp = fopen("addressbook.dat", "rb+");
    char name[50];
    int found = 0;

    printf("Enter name of the contact to update: ");
    fgets(name, sizeof(name), stdin);

    while (fread(&c, sizeof(struct Contact), 1, fp)) {
        if (strcmp(c.name, name) == 0) {
            printf("Enter new phone number: ");
            fgets(c.phone, sizeof(c.phone), stdin);
            printf("Enter new email: ");
            fgets(c.email, sizeof(c.email), stdin);

            fseek(fp, -sizeof(struct Contact), SEEK_CUR);
            fwrite(&c, sizeof(struct Contact), 1, fp);
            found = 1;
            printf("Contact updated successfully.\n");
            break;
        }
    }

    if (!found) {
        printf("Contact not found.\n");
    }
    fclose(fp);
}
```

5. Deleting a Contact

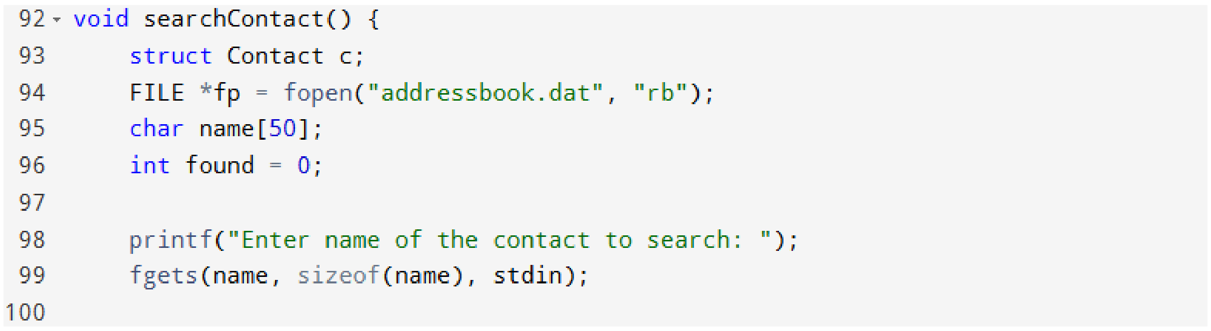
The deleteContact function allows the user to delete a contact by copying all entries except the one to be deleted into a temporary file, then replacing the original file with the temporary one.



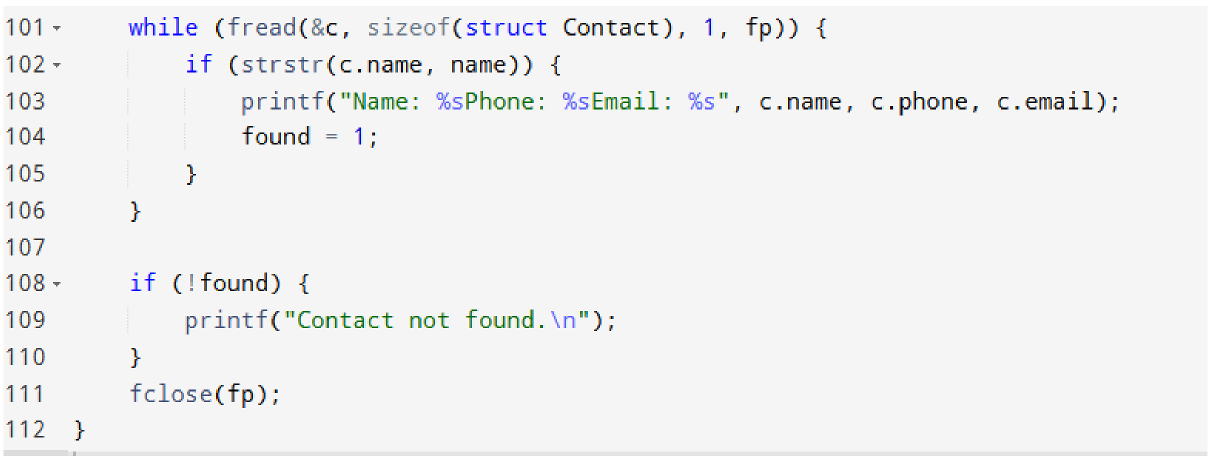
```c
void deleteContact() {
    struct Contact c;
    FILE *fp = fopen("addressbook.dat", "rb");
    FILE *temp = fopen("temp.dat", "wb");
    char name[50];
    int found = 0;

    printf("Enter name of the contact to delete: ");
    fgets(name, sizeof(name), stdin);

    while (fread(&c, sizeof(struct Contact), 1, fp)) {
        if (strcmp(c.name, name) != 0) {
            fwrite(&c, sizeof(struct Contact), 1, temp);
        } else {
            found = 1;
        }
    }

    fclose(fp);
    fclose(temp);

    remove("addressbook.dat");
    rename("temp.dat", "addressbook.dat");

    if (found) {
        printf("Contact deleted successfully.\n");
    } else {
        printf("Contact not found.\n");
    }
}
```

6. Searching for a Contact

The searchContact function enables users to search for contacts by name. It reads each record from the file and displays details if there's a match.

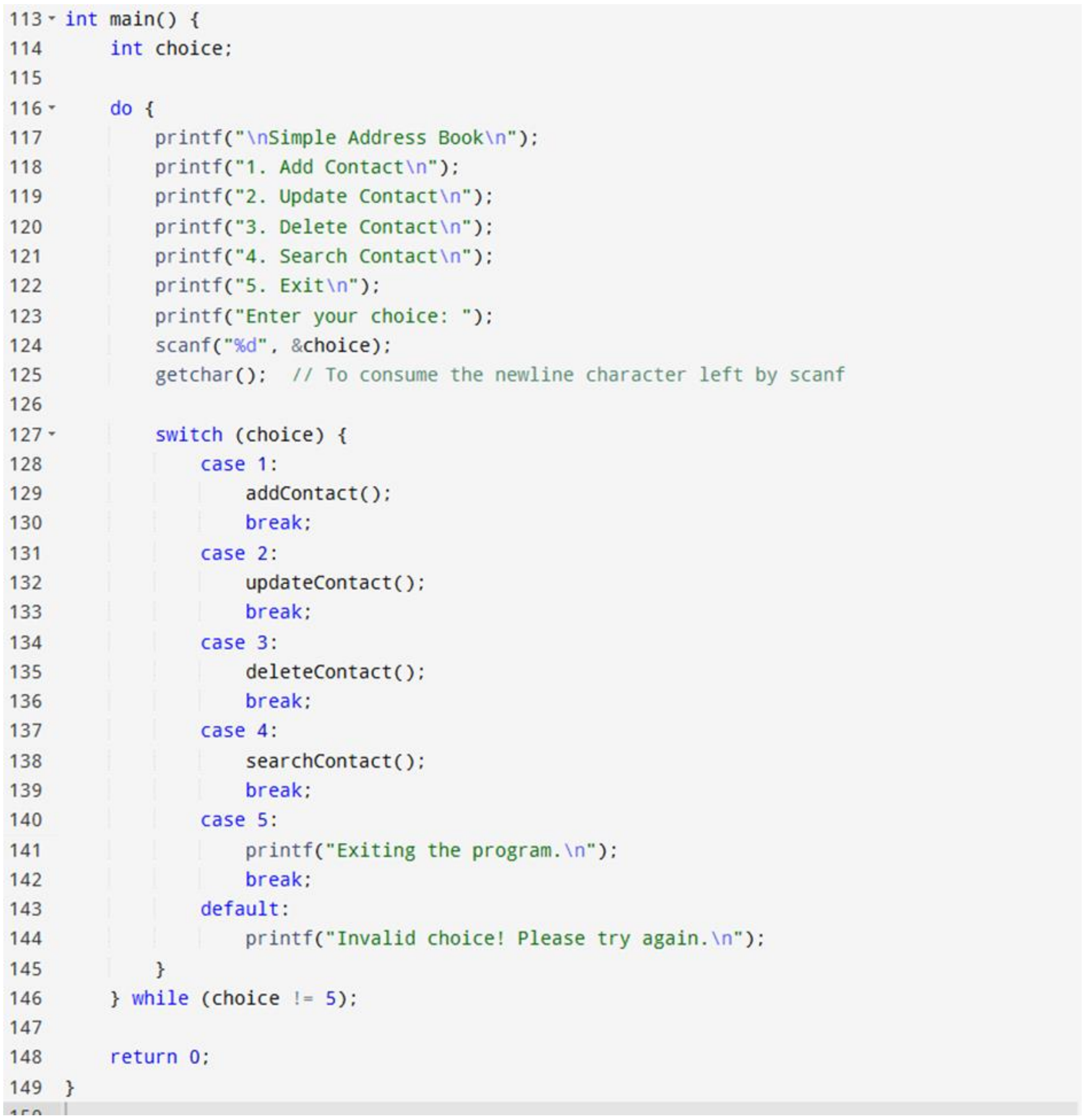


```c
void searchContact() {
    struct Contact c;
    FILE *fp = fopen("addressbook.dat", "rb");
    char name[50];
    int found = 0;

    printf("Enter name of the contact to search: ");
    fgets(name, sizeof(name), stdin);

```


```c
    while (fread(&c, sizeof(struct Contact), 1, fp)) {
        if (strstr(c.name, name)) {
            printf("Name: %sPhone: %sEmail: %s", c.name, c.phone, c.email);
            found = 1;
        }
    }

    if (!found) {
        printf("Contact not found.\n");
    }
    fclose(fp);
}
```

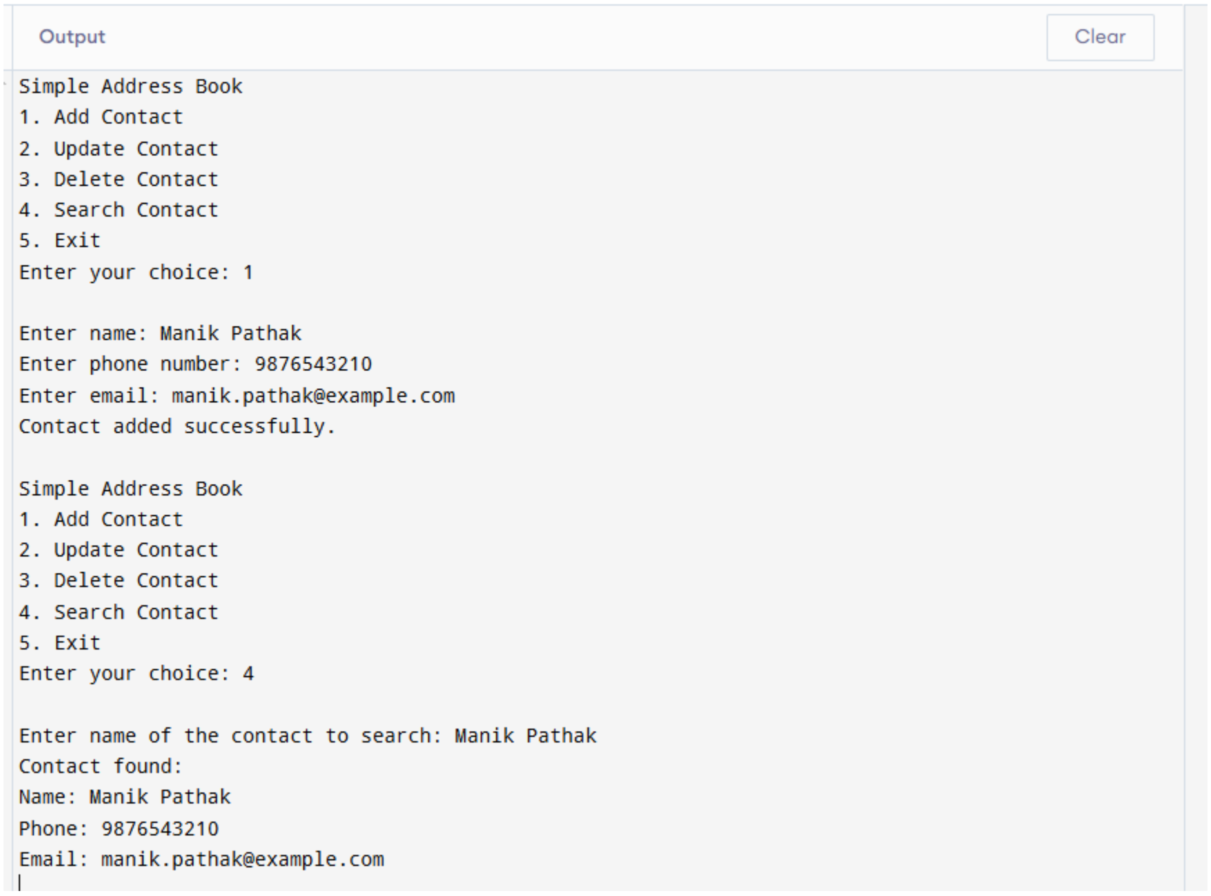
7. Main Function

The main() function controls the flow of the program, allowing users to select options for adding, updating, deleting, or searching for contacts. It provides a simple menu interface and loops until the user selects the exit option.


```c
int main() {
    int choice;

    do {
        printf("\nSimple Address Book\n");
        printf("1. Add Contact\n");
        printf("2. Update Contact\n");
        printf("3. Delete Contact\n");
        printf("4. Search Contact\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar();  // To consume the newline character left by scanf

        switch (choice) {
            case 1:
                addContact();
                break;
            case 2:
                updateContact();
                break;
            case 3:
                deleteContact();
                break;
            case 4:
                searchContact();
                break;
            case 5:
                printf("Exiting the program.\n");
                break;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    } while (choice != 5);

    return 0;
}
```

8. Example of Using the Program

Here is an example of how a user might interact with the Simple Address Book program:



```
Simple Address Book
1. Add Contact
2. Update Contact
3. Delete Contact
4. Search Contact
5. Exit
Enter your choice: 1

Enter name: Manik Pathak
Enter phone number: 9876543210
Enter email: manik.pathak@example.com
Contact added successfully.

Simple Address Book
1. Add Contact
2. Update Contact
3. Delete Contact
4. Search Contact
5. Exit
Enter your choice: 4

Enter name of the contact to search: Manik Pathak
Contact found:
Name: Manik Pathak
Phone: 9876543210
Email: manik.pathak@example.com
```

Conclusion

The Simple Address Book project has allowed me to implement file handling, data storage, and input validation in C. By using structures to define contact details, file processing for persistent storage, and functions to manage contacts, we have developed a fully functional address book application. The project has enhanced my understanding of C programming and provided me with valuable experience in file management and data manipulation.