**Simple Addresh Book**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_NAME\_LENGTH 50

#define MAX\_PHONE\_LENGTH 15

#define MAX\_EMAIL\_LENGTH 50

// Structure to store a contact's information

typedef struct Contact {

char name[MAX\_NAME\_LENGTH];

char phone[MAX\_PHONE\_LENGTH];

char email[MAX\_EMAIL\_LENGTH];

struct Contact \*next; // Pointer to next contact (linked list)

} Contact;

// Function prototypes

Contact\* createContact();

void addContact(Contact \*\*head);

void displayContacts(Contact \*head);

void freeContacts(Contact \*head);

int main() {

Contact \*head = NULL; // Head of the linked list

int choice;

while (1) {

// Menu

printf("\nSimple Address Book (Linked List)\n");

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

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

printf("3. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

getchar(); // Consume newline left by scanf

switch (choice) {

case 1:

// Add a contact

addContact(&head);

break;

case 2:

// Display contacts

if (head == NULL) {

printf("No contacts available!\n");

} else {

displayContacts(head);

}

break;

case 3:

// Exit program

freeContacts(head); // Free all allocated memory before exiting

printf("Exiting the Address Book. Goodbye!\n");

exit(0);

default:

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

}

}

return 0;

}

// Function to create and initialize a new contact

Contact\* createContact() {

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

if (newContact == NULL) {

printf("Memory allocation failed!\n");

exit(1);

}

printf("Enter name: ");

fgets(newContact->name, MAX\_NAME\_LENGTH, stdin);

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

printf("Enter phone number: ");

fgets(newContact->phone, MAX\_PHONE\_LENGTH, stdin);

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

printf("Enter email address: ");

fgets(newContact->email, MAX\_EMAIL\_LENGTH, stdin);

newContact->email[strcspn(newContact->email, "\n")] = '\0'; // Remove newline

newContact->next = NULL; // Initialize next to NULL

return newContact;

}

// Function to add a contact to the linked list

void addContact(Contact \*\*head) {

Contact \*newContact = createContact();

if (\*head == NULL) {

\*head = newContact; // If the list is empty, the new contact is the first one

} else {

Contact \*temp = \*head;

while (temp->next != NULL) {

temp = temp->next; // Traverse to the end of the list

}

temp->next = newContact; // Add new contact at the end

}

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

}

// Function to display all contacts in the linked list

void displayContacts(Contact \*head) {

Contact \*temp = head;

printf("\n--- Contact List ---\n");

while (temp != NULL) {

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

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

printf("Email: %s\n", temp->email);

printf("---------------------\n");

temp = temp->next;

}

}

// Function to free all allocated memory for contacts

void freeContacts(Contact \*head) {

Contact \*temp;

while (head != NULL) {

temp = head;

head = head->next;

free(temp); // Free the memory allocated for each contact

}

}

Click for [Output](https://www.programiz.com/online-compiler/0CcH0ddRlpxkN)