Student: BĂNESARU ANDREEA-BIANCA

DATA STRUCTURES AND ALGORITHMS PROJECT

* **Description of the problem in natural language**

The following C program is a simple phone contacts tracking system

implemented through a doubly linked list structure. The program uses a modular design, offering various functions for creating, accessing, and manipulating contacts within a phone book. Each contact is represented by a structure called ”Contact”, which contains fields for the contact's name, phone number, and pointers to the previous and next contacts in the list.

The main function of the program is a menu-driven interface that allows users to choose from these functionalities interactively. The program continues to execute until the user chooses to exit. Additionally, the code includes error handling by notifying the user when a requested contact operation cannot be performed due to the contact not being found. Memory management is appropriately handled through the use of the “free” function when deleting contacts, minimizing the risk of memory leaks. Overall, the program provides a basic yet functional system for managing phone contacts through a doubly linked list.

The functions of the program include:

* **Inserting a New Contact**: The user can add a new contact to the phone book by providing a name and phone number. The program dynamically allocates memory for the new contact and inserts it at the beginning of the linked list.
* **Updating a Contact**: Users can update the details of an existing contact by specifying the contact's name. The program traverses the linked list to find the contact and updates its name and phone number.
* **Deleting a Contact**: Contacts can be deleted from the phone book by specifying the contact's name. The program searches for the contact in the linked list, adjusts pointers, and frees the allocated memory for the deleted contact.
* **Searching Contacts by Name**: Users can search for contacts by providing a name. The program traverses the linked list, looking for a match, and displays the contact details if found.
* **Sort alphabetically:** Performs the sorting of the phone book alphabetically based on the names of the contacts.
* **Displaying the Phone Book**: The program provides an option to display the entire phone book, presenting the names and phone numbers of all stored contacts.
* **The listing of the entire program (with useful comments)**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//Structure for a phone contact

struct Contact {

char name[30];

char number[15];

struct Contact \*prev;

struct Contact \*next;

};

//Structure for the phone book

struct PhoneBook {

struct Contact \*head;

};

//Ensuring that the correct number and types of arguments are provided

void displayPhoneBook(struct PhoneBook \*phoneBook);

void insertContact(struct PhoneBook \*phoneBook);

void updateContact(struct PhoneBook \*phoneBook, char name[]);

void deleteContact(struct PhoneBook \*phoneBook, char name[]);

void searchByName(struct PhoneBook \*phoneBook, char name[]);

//Function to insert a new contact

void insertContact(struct PhoneBook \*phoneBook) {

//Allocate memory for a new contact

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

//Ask user to enter contact details

printf("Enter the name: ");

scanf("%s", newContact->name);

printf("Enter the number: ");

scanf("%s", newContact->number);

newContact->prev = NULL;

newContact->next = phoneBook->head;

if (phoneBook->head != NULL) {

phoneBook->head->prev = newContact;

}

phoneBook->head = newContact;

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

}

//Function to update contact details

void updateContact(struct PhoneBook \*phoneBook, char name[]) {

//Start with the first contact in the list

struct Contact \*current = phoneBook->head;

//Traverse the list to find the contact to update

while (current != NULL) {

//Check if the current contact's name matches the specified name

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

//Ask user to enter new contact details

printf("Enter the new name: ");

scanf("%s", current->name);

printf("Enter the new number: ");

scanf("%s", current->number);

//Display success message and return from the function

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

return;

}

//Move to the next contact in the list

current = current->next;

}

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

}

//Function to delete a contact

void deleteContact(struct PhoneBook \*phoneBook, char name[]) {

struct Contact \*current = phoneBook->head;

//Traverse the list to find the contact to delete

while (current != NULL) {

//Check if the current contact's name matches the specified name

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

if (current->prev != NULL) {

current->prev->next = current->next;

} else {

phoneBook->head = current->next;

}

if (current->next != NULL) {

current->next->prev = current->prev;

}

free(current);

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

return;

}

current = current->next;

}

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

}

//Function to search contacts by name

void searchByName(struct PhoneBook \*phoneBook, char name[]) {

struct Contact \*current = phoneBook->head;

int found = 0;

while (current != NULL) {

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

printf("Contact found!\n");

printf("Name: %s, Number: %s\n", current->name, current->number);

found = 1;

}

current = current->next;

}

if (!found) {

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

}

}

// Function to sort alphabetically

void sortPhoneBook(struct PhoneBook \*phoneBook) {

struct Contact \*current, \*index;

char tempName[30], tempNumber[15];

if (phoneBook->head == NULL) {

// Empty phone book, nothing to sort

return;

}

for (current = phoneBook->head; current->next != NULL; current = current->next) {

for (index = current->next; index != NULL; index = index->next) {

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

// Swap names

strcpy(tempName, current->name);

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

strcpy(index->name, tempName);

// Swap numbers

strcpy(tempNumber, current->number);

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

strcpy(index->number, tempNumber);

}

}

}

}

int main() {

struct PhoneBook phoneBook;

phoneBook.head = NULL;

int choice;

char name[30];

do {

printf("\nPhone Contacts Tracking System Menu:\n");

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

printf("|1. Display Phone Book |\n");

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

printf("|2. Insert New Contact |\n");

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

printf("|3. Update Contact |\n");

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

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

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

printf("|5. Search by Name |\n");

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

printf("|6. Sort alphabetically |\n");

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

printf("|0. Exit |\n");

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

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

displayPhoneBook(&phoneBook);

break;

case 2:

insertContact(&phoneBook);

break;

case 3:

printf("Enter the name of the contact to update: ");

scanf("%s", name);

updateContact(&phoneBook, name);

break;

case 4:

printf("Enter the name of the contact to delete: ");

scanf("%s", name);

deleteContact(&phoneBook, name);

break;

case 5:

printf("Enter the name to search: ");

scanf("%s", name);

searchByName(&phoneBook, name);

break;

case 6:

sortPhoneBook(&phoneBook);

printf("Phone book sorted alphabetically.\n");

break;

case 0:

printf("Exiting program. Thank you!\n");

break;

default:

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

}

} while (choice != 0);

return 0;

}

// Function to display the phone book

void displayPhoneBook(struct PhoneBook \*phoneBook) {

struct Contact \*current = phoneBook->head;

if (current == NULL) {

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

return;

}

printf("Phone Book:\n");

while (current != NULL) {

printf("Name: %s, Number: %s\n", current->name, current->number);

current = current->next;

}

}

CODE IN DEV C++



* Instances of running program (screenshots)

O imagine care conține text, captură de ecran, Font, număr

Descriere generată automat

O imagine care conține text, captură de ecran, Font, număr

Descriere generată automat

O imagine care conține text, captură de ecran, Font, număr

Descriere generată automat

O imagine care conține text, captură de ecran, Font

Descriere generată automat

O imagine care conține text, captură de ecran, Font, meniu

Descriere generată automat

O imagine care conține text, captură de ecran, Font, număr

Descriere generată automat

O imagine care conține text, captură de ecran, Font, meniu

Descriere generată automat

* **Work flow for every function**

**Inserting a New Contact function**

O imagine care conține text, scris de mână, tablă albă de scris, document

Descriere generată automat

**Updating a Contact function**

**O imagine care conține text, scris de mână, tablă albă de scris

Descriere generată automat**

**Deleting a Contact function**

**O imagine care conține text, scris de mână, tablă albă de scris, proiectare

Descriere generată automat**

**Searching Contacts by Name function**

***O imagine care conține text, scris de mână, tablă albă de scris

Descriere generată automat***

**Displaying the Phone Book function**

***O imagine care conține text, tablă albă de scris, scris de mână, desen

Descriere generată automat***

Sort alphabetically function

O imagine care conține text, scris de mână, tablă albă de scris, document

Descriere generată automat

**Int main function**

**O imagine care conține text, tablă albă de scris, scris de mână, proiectare

Descriere generată automat**

**O imagine care conține text, tablă albă de scris, scris de mână

Descriere generată automat**

**O imagine care conține text, scris de mână, tablă albă de scris

Descriere generată automat**

**Bibliography:**

* [**https://www.programiz.com/dsa/doubly-linked-list**](https://www.programiz.com/dsa/doubly-linked-list)
* [**https://github.com/PRITI24/Phonebook-management-using-doubly-linked-list/blob/master/main.cpp**](https://github.com/PRITI24/Phonebook-management-using-doubly-linked-list/blob/master/main.cpp)
* [**https://medium.com/@noransaber685/efficient-data-manipulation-with-doubly-linked-lists-insertion-deletion-and-positional-6d4c6b79cdeb**](https://medium.com/@noransaber685/efficient-data-manipulation-with-doubly-linked-lists-insertion-deletion-and-positional-6d4c6b79cdeb)
* [**https://www.codecademy.com/article/doubly-linked-list-conceptual**](https://www.codecademy.com/article/doubly-linked-list-conceptual)
* [**https://www.geeksforgeeks.org/introduction-and-insertion-in-a-doubly-linked-list/**](https://www.geeksforgeeks.org/introduction-and-insertion-in-a-doubly-linked-list/)
* [**https://www.sanfoundry.com/c-program-sort-names-alphabetical-order/**](https://www.sanfoundry.com/c-program-sort-names-alphabetical-order/)