CONTACT MANAGEMENT SYSTEM

•

PRESENTED BY:

1. CH. LOKESH
2. M.LOHITH
3. S. YASHWANTH
4. M.V.N. SAHITHI

JUNE 08, 2023

SRM University, Neeru Konda

Andhra Pradesh, India

- AIM: The aim of the provided code is to implement a contact management system using C programming language. The code allows users to add, modify, and delete contacts, as well as display all contacts or only the favourite ones. It also includes functionality for creating and storing contacts in a file. Additionally, the code provides the option to add private contacts that can be accessed with a password. The objective is to efficiently manage and organize contact information, providing users with a user-friendly interface to interact with their contacts.

- ABSTRACT: The aim of this project is to develop a contact management system using the C programming language. The system provides functionalities to add, delete, modify, and display contacts. It also allows the user to mark contacts as favourites and create private contacts protected by a password. The program utilizes linked lists to store and manage the contacts efficiently. Each contact is represented by a structure containing attributes such as name, phone number, and email address. The program ensures that duplicate contacts are not added by performing a duplication check based on the contact's name. The system provides a user-friendly interface with input validation to ensure the correctness of the entered data. It includes options to display all contacts, display favourites contacts, and display private contacts after entering the correct password. Furthermore, the program allows the user to export the contacts to a file, both for general contacts and private contacts. This feature enables the user to store and retrieve the contacts for future reference or sharing with others.

  Overall, this project provides a comprehensive and efficient solution for managing contacts. It showcases the implementation of fundamental programming concepts such as dynamic memory allocation, linked lists, file handling, and input validation. The contact management system offers convenience and organization to users in managing their contact information effectively.

- TECHNOLOGY USED: This project is done by using the following technologies:
    - 1. Dev C++

    - 2. online c compiler
    - 3. Visual studio code

- SAMPLE SCENARIO: Imagine a busy professional who needs to manage a vast network of contacts across different industries. With this contact management system, they can effortlessly add, update, and delete contacts, ensuring their address book is always up to date. They can categorize contacts, mark favourites, and even set passwords for sensitive entries. The system allows them to quickly search for specific contacts and retrieve detailed information whenever needed. Whether it's staying connected with clients, colleagues, or friends, this contact management system streamlines the process and enhances efficiency in managing their extensive network.

- ASSUMPTIONS TAKEN: It assumes that users will have basic knowledge of how to operate the device and navigate through the system's user interface. The system assumes that users will input accurate and up-to-date contact information and will take responsibility for maintaining the data. The system assumes that appropriate security measures are in place to protect the confidentiality and integrity of the stored contact information. The project estimates a total space requirement of approximately 100 MB to accommodate the application, its database, and any associated files. This estimation accounts for future scalability and allows for efficient storage and retrieval of contact information while ensuring optimal system performance.

- DATA HANDLING: The project incorporates efficient handling of input data in the contact management system. When a user interacts with the system, they provide input data in the form of contact information, such as names, phone numbers, and email addresses. The system ensures proper validation and sanitization of the input data to prevent any potential security vulnerabilities or data inconsistencies. Additionally, user input for marking favourites or setting passwords is carefully processed and stored securely. The input data handling mechanism guarantees a seamless and error-free experience for users while maintaining the integrity and confidentiality of their contact information.

- SAMPLE INPUT AND OUTPUT:

The input taken by the program depends on the specific menu option selected. Here is a breakdown of the input required for each menu option:

If the user enters choice 1:
Add a contact.
Enter 1 to add in favorites (else enter zero): [user input]
Enter the name: [user input]
Enter the phone number: [user input]
Enter the Email Id: [user input]

If the user enters choice 2:
Display all contacts.
No additional input is required.

```
-------------------------Welcome to Contact Store-------------------------

 1 to add contact
 2 to display contacts
 3 to delete contact
 4 to modify contact
 5 to display favourite contacts only
 6 to display number of contacts created till now
 7 to save all the contacts in a file
 8 to search contact and display if exists
 9 to add private contact
10 to display private contacts
11 to save private contacts in a file
12 to exit
enter the choice: 1

Enter 1 to add in favourites(else enter zero): 1

***It is a favourite contact now***
Enter the name: srmap
Enter the phone number: 2023
Enter the EmailId:data_structures@srmap.edu.in

--Contact added successfully--

 1 to add contact
 2 to display contacts
 3 to delete contact
 4 to modify contact
 5 to display favourite contacts only
 6 to display number of contacts created till now
 7 to save all the contacts in a file
 8 to search contact and display if exists
 9 to add private contact
10 to display private contacts
11 to save private contacts in a file
12 to exit
enter the choice: 2

************Contacts created are ************

Name: srmap
Phone: 2023
Email ID: data_structures@srmap.edu.in
--------------------
```

If the user enters choice 3:
Delete a contact.
Enter the name: [user input]

```
 1 to add contact
 2 to display contacts
 3 to delete contact
 4 to modify contact
 5 to display favourite contacts only
 6 to display number of contacts created till now
 7 to save all the contacts in a file
 8 to search contact and display if exists
 9 to add private contact
10 to display private contacts
11 to save private contacts in a file
12 to exit
enter the choice: 3
Enter the name: srmap
## Contact deleted successfully ##
```

If the user enters choice 4:
Modify a contact.
Enter a name to modify: [user input]
Enter 1 to modify the name:
Enter 2 to modify the number:
Enter 3 to modify email:
Enter 4 to modify all data:
Enter your choice: [user input]
Enter new name: [user input]
Enter the phone number: [user input]
Enter the Email Id: [user input]

```
 1 to add contact
 2 to display contacts
 3 to delete contact
 4 to modify contact
 5 to display favourite contacts only
 6 to display number of contacts created till now
 7 to save all the contacts in a file
 8 to search contact and display if exists
 9 to add private contact
10 to display private contacts
11 to save private contacts in a file
12 to exit
enter the choice: 4

Enter name to modify: srmap
Enter 1 to modify name:
Enter 2 to modify number:
Enter 3 to modify email_Id:
Enter 4 to modify all data:
1

Enter the new name: Srmap
##Contact modified successfully##
```

If the user enters choice 5:
Display all favorite contacts.
No additional input is required.

```
 1 to add contact
 2 to display contacts
 3 to delete contact
 4 to modify contact
 5 to display favourite contacts only
 6 to display number of contacts created till now
 7 to save all the contacts in a file
 8 to search contact and display if exists
 9 to add private contact
10 to display private contacts
11 to save private contacts in a file
12 to exit
enter the choice: 5

******** Favourite Contacts created are********

Name: Srmap
Phone: 2023
Email ID: data_structures@srmap.edu.in
--------------------
```

If the user enters choice 6:
Display how many contacts were created.
No additional input is required.

```
 1 to add contact
 2 to display contacts
 3 to delete contact
 4 to modify contact
 5 to display favourite contacts only
 6 to display number of contacts created till now
 7 to save all the contacts in a file
 8 to search contact and display if exists
 9 to add private contact
10 to display private contacts
11 to save private contacts in a file
12 to exit
enter the choice: 6

Number of contacts created:1
Number of favourite contacts created:1
```

If the user enters choice 7:
To save all contacts in a file.
No additional input is required.

```
 1 to add contact
 2 to display contacts
 3 to delete contact
 4 to modify contact
 5 to display favourite contacts only
 6 to display number of contacts created till now
 7 to save all the contacts in a file
 8 to search contact and display if exists
 9 to add private contact
10 to display private contacts
11 to save private contacts in a file
12 to exit
enter the choice: 7
###### Contacts are sucessfully stored in file 'contact.txt' ######

 Kindly open the folder to access the contacts
```

```
**********Contacts created are **********

Name: Srmap
Phone: 2023
Email: data_structures@srmap.edu.in
--------------------
```

If the user enters choice 8:
Searching for a contact by name.

Enter a name to search: [User input]

```
 1 to add contact
 2 to display contacts
 3 to delete contact
 4 to modify contact
 5 to display favourite contacts only
 6 to display number of contacts created till now
 7 to save all the contacts in a file
 8 to search contact and display if exists
 9 to add private contact
10 to display private contacts
11 to save private contacts in a file
12 to exit
enter the choice: 8
Enter name to search
Srmap
###### Contact found Sucessfully ######
It is a favourite contact

Name: Srmap
Phone: 2023
Email ID: data_structures@srmap.edu.in
--------------------
```

If the User enters choice 9:
Add a private contact
Enter password: [user input]
Enter the name: [user input]
Enter the phone number: [user input]
Enter the Email Id: [user input]

```
 1 to add contact
 2 to display contacts
 3 to delete contact
 4 to modify contact
 5 to display favourite contacts only
 6 to display number of contacts created till now
 7 to save all the contacts in a file
 8 to search contact and display if exists
 9 to add private contact
10 to display private contacts
11 to save private contacts in a file
12 to exit
enter the choice: 9
Enter password:
srmap123
Enter the name: srm
Enter the phone number: 2023
Enter the EmailId:data_structures@srmap.edu.in

--Private Contact added successfully--
```

If the user enters choice 10:
Display all private contacts:
Enter password: [user input]

```
 1 to add contact
 2 to display contacts
 3 to delete contact
 4 to modify contact
 5 to display favourite contacts only
 6 to display number of contacts created till now
 7 to save all the contacts in a file
 8 to search contact and display if exists
 9 to add private contact
10 to display private contacts
11 to save private contacts in a file
12 to exit
enter the choice: 10
Enter password:
srmap123

************Private Contacts created are ************

Name: srm
Phone: 2023
Email ID: data_structures@srmap.edu.in
--------------------
```

If the user enters choice 11:
Saves all private contacts in a separate file.
Enter password: [User Input]:

```
 1 to add contact
 2 to display contacts
 3 to delete contact
 4 to modify contact
 5 to display favourite contacts only
 6 to display number of contacts created till now
 7 to save all the contacts in a file
 8 to search contact and display if exists
 9 to add private contact
10 to display private contacts
11 to save private contacts in a file
12 to exit
enter the choice: 11
Enter password:
srmap123
###### Contacts are sucessfully stored in file 'priv_contact.txt' ######

###### kindly open the folder to access contacts
```

If the User enters choice 12:
Exits from the program.
No additional input is required.

```
-------------------------Welcome to Contact Store-------------------------

 1 to add contact
 2 to display contacts
 3 to delete contact
 4 to modify contact
 5 to display favourite contacts only
 6 to display number of contacts created till now
 7 to save all the contacts in a file
 8 to search contact and display if exists
 9 to add private contact
10 to display private contacts
11 to save private contacts in a file
12 to exit
enter the choice: 12

*******Thank You*******
------------------------------
Process exited after 3.759 seconds with return value 0
Press any key to continue . . .
```

- ## ALGORITHM:

Step 1:
Start by including the required header files: **stdio.h**, **stdlib.h**, and **string.h**.

Step2:
Define two structures: **Contact** and **Private**:
Contact structure stores information about a contact, including favorite status, name, phone number, email, and pointers to the previous and next contacts.
Private structure stores information about a private contact, including name, phone number, email, and pointers to the previous and next private contacts.

Step 3:
Declare global variables:
head and **tail** of type **Contact** to store the head and tail pointers of the contact list.
p_head and **p_tail** of type **Private** to store the head and tail pointers of the private contact list.

Step 4:
PASS as a constant string to store the password.

Step 5:
Declare function prototypes:
duplication_check() to check for duplicate contacts.
add_contact() to add a contact.
add_private_contact() to add a private contact.
display_contact() and **priv_display_contact()** to display contact details.
display_contacts() and **priv_display_contacts()** to display all contacts.
display_fav() to display favorite contacts.
delete_contact() to delete a contact.
modify_contact() to modify a contact.
number_of_contacts() to count the number of contacts.
create_file() and **priv_create_file()** to create a file and store contacts in it.

Step 6:
Implement the **duplication_check()** function:
Iterate through the contact list and compare each contact's name with the given name.
If a contact with the same name exists, return 1 (indicating duplication), else return 0.

Step 7:

Implement the **add_contact()** function:
Allocate memory for a new **Contact** node.
Prompt the user to enter contact details: favorite status, name, phone number, and email.
Check for duplication using **duplication_check()**.
If the contact already exists, display a message and return.
If the contact is the first contact, update **head** and **tail**.
Otherwise, update the **next** and **prev** pointers of the new node and the existing tail node.
Display a success message.

Step 8:
Implement the **add_private_contact()** function:
Prompt the user to enter the password.
If the password is correct, proceed; otherwise, display an error message and return.
Allocate memory for a new **Private** node.
Prompt the user to enter private contact details: name, phone number, and email.
If the private contact is the first contact, update **p_head** and **p_tail**.
Otherwise, update the **next** and **prev** pointers of the new node and the existing **p_tail** node.
Display a success message.

Step 9:
Implement the **display_contact()** and **priv_display_contact()** functions:
Display the name, phone number, and email of the given contact.

Step 10:
Implement the **display_contacts()** function:
Check if any contacts are stored.
If no contacts exist, display a message.
Otherwise, iterate through the contact list and call **display_contact()** for each contact.

Step 11:
Implement the **priv_display_contacts()** function:
Prompt the user to enter the password.
If the password is correct, proceed; otherwise, display an error message and return.
Check if any private contacts are stored.
Otherwise, iterate through the private contact list and call **priv_display_contact()** for each private contact.
Implement the **display_fav()** function:

Check if any contacts are stored.
If no contacts exist, display a message.
Otherwise, iterate through the contact list and check if the contact's favorite status is true.
If the contact is a favorite, call **display_contact()** to display its details.

Step 12:
Implement the **delete_contact()** function:
Prompt the user to enter the name of the contact to be deleted.
Iterate through the contact list to find the contact.
If the contact is found, update the **next** and **prev** pointers of adjacent nodes.
Free the memory allocated to the deleted node.
Display a success message.
If the contact is not found, display an error message.

Step 13:
Implement the **modify_contact()** function:
Prompt the user to enter the name of the contact to be modified.
Iterate through the contact list to find the contact.
If the contact is found, prompt the user to enter new contact details.
Update the contact's details.
Display a success message.
If the contact is not found, display an error message.

Step 14:
Implement the **number_of_contacts()** function:
Initialize a counter variable to 0.
Iterate through the contact list and increment the counter for each contact.
Return the counter.

Step 15:
Implement the **create_file()** function:
Prompt the user to enter the filename.
Open the file in write mode.
Iterate through the contact list and write each contact's details to the file.
Close the file.
Display a success message.

Step 16:
Implement the **priv_create_file()** function:
Prompt the user to enter the filename.

Prompt the user to enter the password.
If the password is correct, proceed; otherwise, display an error message and return.
Open the file in write mode.
Iterate through the private contact list and write each private contact's details to the file.
Close the file.
Display a success message.

Step 17:
In the **main()** function:
Initialize the **head**, **tail**, **p_head**, and **p_tail** pointers to **NULL**.
Display a menu of options to the user and prompt for their choice.
Based on the user's choice, call the corresponding functions.
Continue the menu loop until the user chooses to exit.
Free the memory allocated to the contact and private contact lists.
That completes the algorithm for the given code.

- ## CONCLUSION:

In conclusion, the contact management system presented above provides a user-friendly interface for managing contacts. It allows users to add, display, modify, and delete contacts, as well as handle private contacts with password protection. The system also includes features like displaying favorite contacts, checking the total number of contacts, and exporting contact details to files.
With this contact management system, users can easily organize their contact information, ensuring quick access to important details such as names, phone numbers, and email addresses. The ability to mark favorite contacts and manage private contacts adds flexibility and security to the system.
Overall, the system provides an efficient solution for contact management, streamlining the process of storing, retrieving, and updating contact information. Users can rely on this system to keep their contacts organized and accessible, enhancing productivity and simplifying communication.

ROLL NUMBERS:

CH LOKESH - AP22110011114
M LOHITH - AP22110011121
S YASHWANTH - AP22110011149
M.V.N SAHITHI - AP22110011133

- THE END -