Project: Phone Book

Muhammad Athar

1200

3rd semester (3M)

BSCS

Submitted to: Mr. Usman Ghani

The Islamia University of Bahawalpur

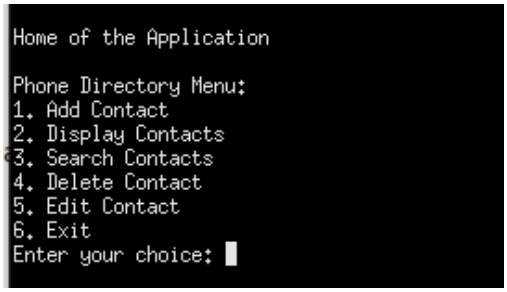
Description:

The Phone Directory application is designed to provide a user-friendly interface for managing contacts. It utilizes an unordered_map data structure to efficiently store and retrieve contact information based on unique keys. This project report provides a comprehensive overview of the application, including its functionality, data structure, and implementation details.

Main Functionalities of Program are:

- Add Contact
- Edit Contact
- Delete Contact
- Search Contact
    i. Through Name
    ii. Through Number
- Display Contacts

I have use Structure to store multiple data from input. I have name struct as **phoneBook** & then created class **PhoneDirectory**.



Functional Working (USER-view):

1 – Add Contact:

This Function create a new contact by taking Name [FULL], Phone Numbers [Home,Work,Office] ,Email ,& Address. Then store them in Linked List [Circular] which is used as data structures. For this user have to select **1** from directory.

Email is optional but remaining are mandatory.

- Example and Output:
-
```
Enter your choice: 1
Enter First Name: ali
Enter Last Name: hamdan
Enter Mobile Number: 034580859045
Enter Home Number: 23455555556
Enter Work Number: 45456543
Enter Email: alihamdan7878@gmail.com
Enter Home Address: hsp
Contact added in directory successfully!
```

2 – Edit Contact:

This Function edits already created contact if user made a mistake during contact saving. For this option user have to select **5** from the directory. Then user have to input **Fullname** of the contact that is required to edit.

Then have to add new information I.e **New** Name, Phone Numbers, Email, Address. After it the already present contact will be updated.

Before edit:

```
Enter your choice: 1
Enter First Name: athar
Enter Last Name: ali
Enter Mobile Number: 03338786554
Enter Home Number: 45678
Enter Work Number: 45678
Enter Email: atharali998@gmail.com
Enter Home Address: hsp
Contact added in directory successfully!
```

After:

```
Enter your choice: 5
Enter full name of contact to edit: athar ali
Enter new First Name: muhammad
Enter new Last Name: athar
Enter new Mobile Number: 3456784
Enter new Home Number: 456789
Enter new Work Number: 65432
Enter new Email: atharm.ali337@gmail.com
Enter new Address: lahore
Contact Edited & Updated successfully!
```

Here you can check in both before & after old name was Ali Abbas which is later changed to Ali Akbar , also Email.

3 – Delete Contact:

As name show, This Function will delete the contact, if user saved a wrong number or due to any other problem if user wants to delete the contact, then he can chose **4** from directory and can delete any contact.

For this user have to enter Fullname of contact and simply press enter will delete the contact.

```
Enter your choice: 4
Enter Full Name of contact to delete: zaheer altaf
Contact Deleted Permanently!
```

Note: After deleting there is no option to recover except to add again.

In case if user enter wrong spelling , it will through error ”Try Spelling Again, No Contact Found by This Name.” so user have to re-enter the full name.

4 – Search Contact:

This Function is built for user to search the contact, it have two different features:

- **Searching by Name:**

  In this, user will enter either **Fullname** or **FirstName** or **LastName.**. From here two conditions arise:

  1. If user have enter FullName, then algo will print contacts with that Fullname.
  2. If user either enter firstname or lastname, algo will print all the contacts from directory that have that name in it, even if the user’s input is other’s contact is middle name.

CASE 1 :

```
Enter your choice: 3
Enter Full Name or Contact Number: 03108419045
Search Results:
Name: ali zain
Phone Number:
- 03108419045
- 345678
- 7654
Email: xzaino8250@gmail.com
Address: bwp
```

CASE 2:

```
Enter your choice: 3
Enter Full Name or Contact Number: ali
Search Results:
Name: ali zain
Phone Number:
- 03108419045
- 345678
- 7654
Email: xzaino8250@gmail.com
Address: bwp
Name: athar ali
Phone Number:
- 03338786554
- 45678
- 45678
Email: atharali998@gmail.com
Address: hsp
Name: ali hamdan
Phone Number:
- 034580859045
- 23455555556
- 45456543
Email: alihamdan7878@gmail.com
Address: hsp
```

- **Searching by Number:**

  In this user will enter a numeric form of query I.e number, when user will press enter algo will print all the contact having that numeric query in them.

```
Enter your choice: 3
Enter Full Name or Contact Number: 03108419045
Search Results:
Name: ali zain
Phone Number:
- 03108419045
- 345678
- 7654
Email: xzaino8250@gmail.com
Address: bwp
```

5 – Display Contacts:

This Function is built for user to Display all contacts, present in the directory. For this user have to select **2** from directory. It will print all the contacts with in the directory.

```
Enter your choice: 2
Name: ali zain
Phone Number:
 - 03108419045
 - 345678
 - 7654
Email: xzaino8250@gmail.com
Address: bwp
Name: zaheer altaf
Phone Number:
 - 03042673006
 - 4567
 - 654
Email: zaheerjheedo8798@gmail.com
Address: bwp
Name: athar ali
Phone Number:
 - 03338786554
 - 45678
 - 45678
Email: atharali998@gmail.com
Address: hsp
```

*Note: I have used garbage values as **CONTACTS**.*

Functionalities (Back-end):

1 – Add Contacts:

- Takes a reference to a `phoneBook` struct containing new contact info.
- Creates a full name by combining first and last name with a space.
- Adds the full name as a key and the entire `phoneBook` struct as the value to the phone directory (unordered_map).
- Prints a success message upon successful addition.

2 – Display Contacts:

- Checks if the directory is empty. If empty, prompts the user to add contacts.
- If not empty, iterates through each contact in the directory.
- For each contact, prints name, phone numbers (excluding empty ones), email, and address.
- Separates each contact's details with a line.

3 – Search Contacts:

- Takes a reference to a string containing the search query.
- Initializes a flag to track if any matching contacts are found (initially false).
- Iterates through each contact in the directory.
- For each contact, checks if the search query matches the name, any phone number, email, or address.
- If a match is found, prints the contact's details (name, phone numbers, email, address) and sets the flag to true.
- If no matches are found after iterating through all contacts, prints a message indicating no contacts were found.

4 – Edit Contacts:

- Takes full name of the contact to edit and a `phoneBook` struct with updated information.
- Checks if the contact exists in the directory.
- If found:
    - Creates a new full name using the updated information.
    - Removes the old contact using the original full name.
    - Adds a new entry with the updated full name and information.
- Displays success message if edited or "Contact not found" message if not found.

5 – Delete Contacts:

- Takes the full name of the contact to delete.
- Attempts to remove the contact from the directory using the full name as the key.
- If successful, displays "Contact Deleted Permanently!" message.
- If unsuccessful (contact not found), displays "Either contact is not present or Spelling mistake!" message.

Data Structures and Functions Used in the Phone Directory Project

Data Structure	Function(s) used	Purpose in the Project
Linked List (Implemented using Node struct)	- `addContacts` -`displayContacts` - `searchContacts` - `deleteContacts` - `editContact`	- Stores and manages contacts. - Enables efficient insertion, deletion, and traversal of contacts.
Array (within `phoneBook` struct)	- None (directly accessed)	- Holds phone numbers (up to 3) for a single contact.

Explanation:

- The code utilizes a lined list to store contacts. Each Node in the list holds a phoneBook struct containing contact information.
- The linked list functions manage the overall phone directory:
  I. `addContacts` adds a new contact to the end of the list.
  II. `displayContacts` iterates through the list and displays each contact's details.
  III. `searchContacts` traverses the list, searching for contacts based on the provided query (name or phone number).
  IV. `deleteContacts` finds and removes a specific contact from the list based on their full name.
  V. `editContact` locates a contact by name and updates their information with the provided new details.
- An array is used within the `phoneBook` struct to store up to three phone numbers for a single contact. However, the code doesn't directly utilize array functions; it accesses phone numbers within the array based on their indices (0, 1, or 2).

Note:

- The code likely uses internal implementations of the insert, find, and erase functions provided by the unordered map data structure.
- Iterators are not explicitly shown in the provided code snippet, but they are likely used behind the scenes to iterate through the key-value pairs in the unordered map.

By Athar

Header’s

- <iostream>: This header provides input/output functionalities for the program, such as displaying messages on the screen and taking user input.
- <iomanip>: This header offers tools for manipulating input/output formatting, potentially used for aligning output in the phone directory display.

- <string>: This header provides functions for working with strings, which is essential for various aspects of the program. It's likely used for manipulating text data like names, phone numbers, email addresses, and the search query.

Conclusion:

The code implements a phone directory application. It utilizes an unordered map to efficiently store contacts with full name (combined first and last name) as the key and a phoneBook struct containing details as the value. The program allows adding, displaying, searching, deleting, and editing contacts.

By Athar