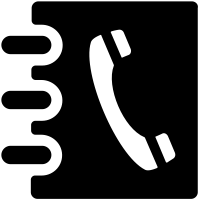
**Project: Phone Directory Application**



**Created by Maryam zahra**

**BS CS**

**1174**

**The Islamiya University of Bahawalpur**

**Description:**

The Phone Directory application offers a user-friendly interface for managing contacts. It employs an `unordered\_map` data structure to efficiently store and retrieve contact information using unique keys. This project report presents a thorough overview of the application, covering its functionality, data structure, and implementation details. **The main functionalities of the program include**:

* Adding a contact
* Editing a contact
* Deleting a contact
* Searching for a contact
* By name
* By number

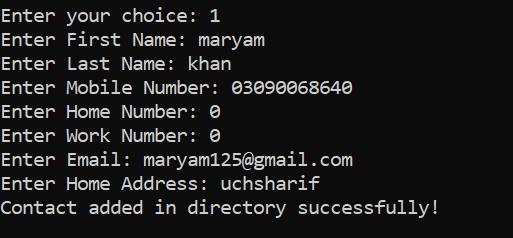
I have used a structure to store multiple pieces of data from input. The structure is named `phoneBook`, and I have created a class called `PhoneDirectory`.

.

****

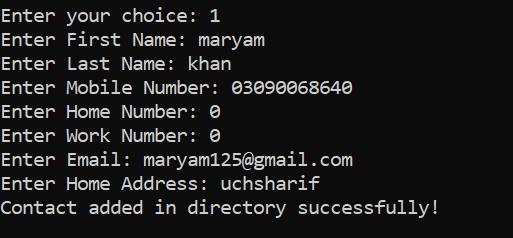
**Functional Working (USER-view):**

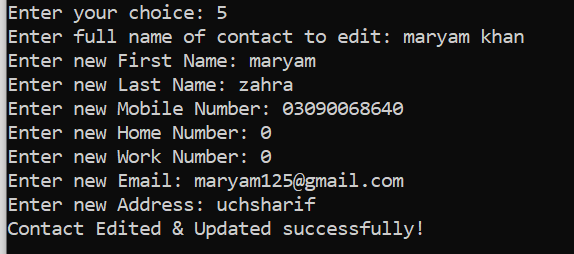
**1 – Add Contact:**This function creates a new contact by taking the following details: Full Name, Phone Numbers (Home, Work, Office), Email, and Address. The contact information is then stored in a circular linked list, which is used as the data structure. The user must select option 1 from the directory to create a new contact. While the email is optional, all other fields are mandatory.

* Example and Output:
* 

**2 – Edit Contact:**This function allows the user to edit an existing contact if a mistake was made during the initial saving. To access this option, the user must select option 5 from the directory. The user will then be prompted to enter the full name of the contact that needs to be edited. After providing the new information (i.e., New Name, Phone Numbers, Email, Address), the existing contact will be updated with the new details.

**Before edit:**



**After:**

Here you can check in both before & after old name was Maryam khan which is

later changed to Maryam zahra , also Email.

**3 – Delete Contact:**As the name suggests, this function will delete a contact. If the user has saved a wrong number or wants to delete a contact for any other reason, they can select option 4 from the directory to delete the desired contact.

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

*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:**

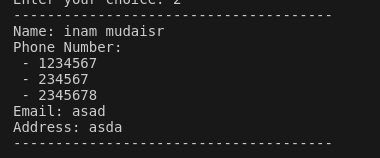
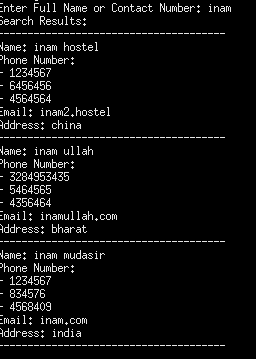
This Function is built for user to search the contact, it have two different features:

* **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 :**



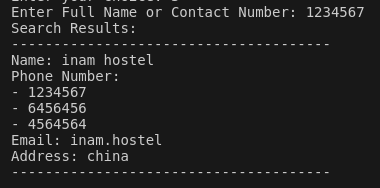
**CASE 2:**

* **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.



**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.



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

**Functionalities (Back-end):**

**1 – Add Contacts:**

* Accept a reference to a **phoneBook** struct that contains new contact information.
* Create a full name by concatenating the first and last name, separated by a space.
* Add an entry to the phone directory (an **unordered\_map**), using the full name as the key and the entire **phoneBook** struct as the value.
* Print a success message to confirm the contact has been successfully added.
  1. **– Display Contacts:**
  + ***Check if the directory is empty. If it is empty, prompt the user to add contacts.***
* ***If the directory is not empty, iterate through each contact in the directory.***
* ***For each contact, print the name, phone numbers (excluding any empty ones), email, and address.***
* ***Separate the details of each contact with a line.***
  1. **-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:

1. addContacts adds a new contact to the end of the list.
2. displayContacts iterates through the list and displays each contact's details.
3. searchContacts traverses the list, searching for contacts based on the provided query (name or phone number).
4. deleteContacts finds and removes a specific contact from the list based on their full name.
5. 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.

## 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.
* <chrono>: This header deals with time-related functionalities. In the provided code, it's likely used for creating the simulated loading bar using the sleep\_for function from the <thread> library.
* <thread>: This header allows working with multiple threads of execution. In this case, it's probably used for the loading bar animation, where a separate thread might be responsible for displaying the progress bar characters.
* <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.