Department of Computer Science
Islamia University Bahawalpur.

(Project)

Project Title: ***Phone Directory Application.***

Subject: ***Data Structures & Algorithms.***

Submitted to: ***Mr. Muhammad Usman Ghani.***

Submitted by:.Ghulam Ali

Roll No:.S23BDOCS1M01104

Section: BS-CS(2M).

First, let's discuss my project, which is implemented in C++. I'll outline the steps involved to explain it more effectively. This project utilizes a doubly linked list for efficient data management and improved application flow in my Phone Directory Application. Additionally, I integrated a small database to save contacts, enabling access and reference outside the application.

Contact.h:

Initially, I created a header file named Contact.h to define multiple classes. This approach improved the program's modularity and manageability, facilitating the development of the Phone Directory Application. Additionally, I implemented a small database to save contacts, allowing for external access and reference outside the application.

Getter and Setter Functions:

Provide access to private member variables and allow modification of the contact's details.

```h
#ifndef CONTACT_H
#define CONTACT_H
#include <string>
using namespace std;
class Contact {
private:
    string firstName;
    string lastName;
    string mobileNumber;
    string homeNumber;
    string workNumber;
    string emailAddress;
    string birthday;
public:
    Contact(const string& firstName, const string& lastName,
            const string& mobileNumber, const string& homeNumber,
            const string& workNumber, const string& emailAddress,
            const string& birthday);
            //Getter
    string getFirstName() const;
    string getLastName() const;
    string getMobileNumber() const;
    string getHomeNumber() const;
    string getWorkNumber() const;
    string getEmailAddress() const;
    string getBirthday() const;
 // Setter
    void setMobileNumber(const string& mobile);
    void setHomeNumber(const string& home);
    void setWorkNumber(const string& work);
    void setEmailAddress(const string& email);
    void setBirthday(const string& bday);
    string toString() const;
    static Contact fromString(const string& contactStr);
};

#endif // CONTACT_H
```

Contact Function.cpp:

```
Contact::Contact(const string& firstName, const string& lastName,
                 const string& mobileNumber, const string& homeNumber,
                 const string& workNumber, const string& emailAddress,
                 const string& birthday)
    : firstName(firstName), lastName(lastName), mobileNumber(mobileNumber),
      homeNumber(homeNumber), workNumber(workNumber), emailAddress(emailAddress),
      birthday(birthday) {}

string Contact::getFirstName() const { return firstName; }
string Contact::getLastName() const { return lastName; }
string Contact::getMobileNumber() const { return mobileNumber; }
string Contact::getHomeNumber() const { return homeNumber; }
string Contact::getWorkNumber() const { return workNumber; }
string Contact::getEmailAddress() const { return emailAddress; }
string Contact::getBirthday() const { return birthday; }

void Contact::setMobileNumber(const string& mobile) { mobileNumber = mobile; }
void Contact::setHomeNumber(const string& home) { homeNumber = home; }
void Contact::setWorkNumber(const string& work) { workNumber = work; }
void Contact::setEmailAddress(const string& email) { emailAddress = email; }
void Contact::setBirthday(const string& bday) { birthday = bday; }

string Contact::toString() const {
    return firstName + "," + lastName + "," + mobileNumber + "," + homeNumber + "," +
           workNumber + "," + emailAddress + "," + birthday;
}
Contact Contact::fromString(const string& contactStr) {
    stringstream ss(contactStr);
    string firstName, lastName, mobileNumber, homeNumber, workNumber, emailAddress, birthday;
    getline(ss, firstName, ',');
    getline(ss, lastName, ',');
    getline(ss, mobileNumber, ',');
    getline(ss, homeNumber, ',');
    getline(ss, workNumber, ',');
    getline(ss, emailAddress, ',');
    getline(ss, birthday, ',');
    return Contact(firstName, lastName, mobileNumber, homeNumber, workNumber, emailAddress, birthday);
}
```

Constructor:

- *In this above code firstName(firstName): Initializes the firstName member variable with the firstName parameter.*
- *lastName(lastName): Initializes the lastName member variable with the lastName parameter.*
- *mobileNumber(mobileNumber): Initializes the mobileNumber member variable with the mobileNumber parameter.*
- *homeNumber(homeNumber): Initializes the homeNumber member variable with the homeNumber parameter.*

- *workNumber(workNumber): Initializes the workNumber member variable with the workNumber parameter.*
- *emailAddress(emailAddress): Initializes the emailAddress member variable with the emailAddress parameter.*
- *birthday(birthday): Initializes the birthday member variable with the birthday parameter.*
- *next(NULL): Initializes the next pointer to NULL.*

Getter Function:

*These are the definitions of the getter functions declared in the header file. Each function returns the value of the corresponding private member variable. They are marked const because they do not modify the object**.***

Setter Functions:

These are the definitions of the setter functions declared in the header file. Each function sets the value of the corresponding private member variable to the value provided as a parameter.

Phone Directory Header.h:

```cpp
#ifndef PHONEDIRECTORY_H
#define PHONEDIRECTORY_H

#include "Contact.h"
#include <unordered_map>
#include <vector>
#include <string>

using namespace std;

class PhoneDirectory {
private:
    unordered_map<string, vector<Contact*>> lastNameMap;
    vector<Contact*> contacts;
    const string filename = "contacts.txt";

public:
    PhoneDirectory();
    ~PhoneDirectory();

    void addContact(const string& firstName, const string& lastName,
                    const string& mobileNumber, const string& homeNumber,
                    const string& workNumber, const string& emailAddress,
                    const string& birthday);
    void displayContacts() const;
    void editContact(const string& firstName, const string& lastName, const string& newMobile,
                     const string& newHome, const string& newWork, const string& newEmail,
                     const string& newBirthday);
    void deleteContact(const string& firstName, const string& lastName);
    void searchContact(const string& query) const;
};

#endif // PHONEDIRECTORY_H
```

- *PhoneDirectory(): The constructor. Initializes a new PhoneDirectory object.*
- *void addContact(...): Adds a new contact to the directory with the provided details (first name, last name, mobile number, home number, work number, email address, birthday).*
- *void displayContacts(const std::string& sortBy = "") const: Displays all contacts. The optional sortBy parameter can specify a sorting criterion (e.g., by last name, first name, etc.). It's marked const because it does not modify the PhoneDirectory object.*
- *void editContact(...): Edits an existing contact identified by first name and last name with new details (new mobile number, new home number, new work number, new email, new birthday).*
- *void deleteContact(const std::string& firstName, const std::string& lastName): Deletes a contact identified by first name and last name.*
- *void searchContact(const std::string& query, const std::string& sortBy = "") const: Searches for contacts that match the query string and*

optionally sorts the results by the specified criterion. It's marked const because it does not modify the PhoneDirectory object.

- *~PhoneDirectory(): The destructor. Cleans up any resources used by the PhoneDirectory object.*

Phone Directory Function.cpp:

```cpp
#include "PhoneDirectory.h"
#include <iostream>
#include <algorithm>

using namespace std;

PhoneDirectory::PhoneDirectory() {

}

PhoneDirectory::~PhoneDirectory() {
    for (Contact* contact : contacts) {
        delete contact;
    }
}

void PhoneDirectory::addContact(const string& firstName, const string& lastName,
                                const string& mobileNumber, const string& homeNumber,
                                const string& workNumber, const string& emailAddress,
                                const string& birthday) {
    Contact* newContact = new Contact(firstName, lastName, mobileNumber, homeNumber, workNumber, emailAddress, birthday);
    contacts.push_back(newContact);
    lastNameMap[lastName].push_back(newContact);

}

void PhoneDirectory::displayContacts() const {
    if (contacts.empty()) {
        cout << "There are no contacts in the directory." << endl;
        return;
    }
```

```
    for (const Contact* contact : contacts) {
        cout << "Name: " << contact->getFirstName() << " " << contact->getLastName() << endl;
        cout << "Mobile Number: " << contact->getMobileNumber() << endl;
        cout << "Home Number: " << contact->getHomeNumber() << endl;
        cout << "Work Number: " << contact->getWorkNumber() << endl;
        cout << "Email Address: " << contact->getEmailAddress() << endl;
        cout << "Birthday: " << contact->getBirthday() << endl;
        cout << endl;
    }
}

void PhoneDirectory::editContact(const string& firstName, const string& lastName, const string& newMobile,
                                 const string& newHome, const string& newWork, const string& newEmail,
                                 const string& newBirthday) {
    for (Contact* contact : contacts) {
        if (contact->getFirstName() == firstName && contact->getLastName() == lastName) {
            contact->setMobileNumber(newMobile);
            contact->setHomeNumber(newHome);
            contact->setWorkNumber(newWork);
            contact->setEmailAddress(newEmail);
            contact->setBirthday(newBirthday);
            cout << "Contact updated successfully!" << endl;
            return;
        }
    }
    cout << "Contact not found!" << endl;
}
void PhoneDirectory::deleteContact(const string& firstName, const string& lastName) {
    auto it = remove_if(contacts.begin(), contacts.end(), [&](Contact* contact) {
        return contact->getFirstName() == firstName && contact->getLastName() == lastName;
    });

    if (it != contacts.end()) {
        for (auto iter = it; iter != contacts.end(); ++iter) {
            delete *iter;
        }
        contacts.erase(it, contacts.end());
        lastNameMap.erase(lastName);
        cout << "Contact deleted successfully!" << endl;
    } else {
        cout << "Contact not found!" << endl;
    }
}

void PhoneDirectory::searchContact(const string& query) const {
    vector<Contact*> results;
    auto it = lastNameMap.find(query);
    if (it != lastNameMap.end()) {
        results.insert(results.end(), it->second.begin(), it->second.end());
    }

    for (const Contact* contact : contacts) {
        if (contact->getFirstName() == query || contact->getMobileNumber() == query ||
            contact->getHomeNumber() == query || contact->getWorkNumber() == query) {
            results.push_back(const_cast<Contact*>(contact));
        }
    }
```

```
    if (!results.empty()) {
        cout << "Search results:" << endl;
        for (const Contact* contact : results) {
            cout << "Name: " << contact->getFirstName() << " " << contact->getLastName() << endl;
            cout << "Mobile Number: " << contact->getMobileNumber() << endl;
            cout << "Home Number: " << contact->getHomeNumber() << endl;
            cout << "Work Number: " << contact->getWorkNumber() << endl;
            cout << "Email Address: " << contact->getEmailAddress() << endl;
            cout << "Birthday: " << contact->getBirthday() << endl;
            cout << endl;
        }
    } else {
        cout << "Contact not found!" << endl;
    }
}
```

Constructor:

- *Initializes the `head` pointer to NULL, indicating an empty directory.*

Destructor:

- *Deallocates memory by deleting all contacts in the linked list.*

Add Contact Method:

- *Adds a new contact, updates linked list pointers, adds to the last name map, and saves contacts to a file.*

Get All Contacts Method:

- *Retrieves all contacts and returns them as a vector of pointers to `Contact` objects.*

Display Contacts Method:

- *Displays all contacts, optionally sorted by name or mobile number.*

Edit Contact Method:

- *Edits the information of an existing contact.*

Delete Contact Method:

- *Deletes a contact based on first name and last name.*

Search Contact Method:

- *Searches for a contact based on the query string and optionally sorts the results.*

Main.cpp:

- ***displayMenu****():*
  Displays the main menu options for the phone directory application.
- ***switch (choice) :***
  The user's choice is processed using a switch statement, where each case corresponds to a menu option.

```
#include <iostream>
#include "PhoneDirectory.h"
using namespace std;
void displayMenu() {
    cout << "Phone Directory Application" << endl;
    cout << "1. Add New Contact" << endl;
    cout << "2. Edit Existing Contact" << endl;
    cout << "3. Delete Contact" << endl;
    cout << "4. Display All Contacts" << endl;
    cout << "5. Search Contact" << endl;
    cout << "6. Exit" << endl;
    cout << "Enter your choice: ";
}
int main() {
    PhoneDirectory directory;
    int choice;
    string firstName, lastName, mobile, home, work, email, birthday, query;

    while (true) {
        displayMenu();
        cin >> choice;
        cin.ignore(); // Ignore newline character left in the input buffer
```

```
    while (true) {
        displayMenu();
        cin >> choice;
        cin.ignore(); // Ignore newline character left in the input buffer

        switch (choice) {
        case 1:
            cout << "Enter first name: ";
            getline(cin, firstName);
            cout << "Enter last name: ";
            getline(cin, lastName);
            cout << "Enter mobile number: ";
            getline(cin, mobile);
            cout << "Enter home number: ";
            getline(cin, home);
            cout << "Enter work number: ";
            getline(cin, work);
            cout << "Enter email address: ";
            getline(cin, email);
            cout << "Enter birthday: ";
            getline(cin, birthday);
            directory.addContact(firstName, lastName, mobile, home, work, email, birthday);
            cout << "Contact added successfully!" << endl;
            break;

        case 2:
            cout << "Enter first name of the contact to edit: ";
            getline(cin, firstName);
            cout << "Enter last name of the contact to edit: ";
            getline(cin, lastName);
            cout << "Enter new mobile number: ";
            getline(cin, mobile);
            cout << "Enter new home number: ";
            getline(cin, home);
            cout << "Enter new work number: ";
            getline(cin, work);
            cout << "Enter new email address: ";
            getline(cin, email);
            cout << "Enter new birthday: ";
            getline(cin, birthday);
            directory.editContact(firstName, lastName, mobile, home, work, email, birthday);
            break;

        case 3:
            cout << "Enter first name of the contact to delete: ";
            getline(cin, firstName);
            cout << "Enter last name of the contact to delete: ";
            getline(cin, lastName);
            directory.deleteContact(firstName, lastName);
            break;

        case 4:
            directory.displayContacts();
            break;

        case 5:
            cout << "Enter name or phone number to search: ";
            getline(cin, query);
            directory.searchContact(query);
            break;

        case 6:
            cout << "Exiting..." << endl;
            return 0;

        default:
            cout << "Invalid choice. Please try again." << endl;
            break;
        }
    }

    return 0;
}
```

Output:

```
Phone Directory Application
1. Add New Contact
2. Edit Existing Contact
3. Delete Contact
4. Display All Contacts
5. Search Contact
6. Exit
Enter your choice: 1
Enter first name: Ghulam
Enter last name: Ali
Enter mobile number: 1
Enter home number: 11
Enter work number: 111
Enter email address: @
Enter birthday: 123
Contact added successfully!
```

```
Enter your choice: 2
Enter first name of the contact to edit: Ghulam
Enter last name of the contact to edit: Ali
Enter new mobile number: 1
Enter new home number: 11
Enter new work number: 22
Enter new email address: @1
Enter new birthday: 321
Contact not found!
```

```
Enter your choice: 4
Name: Ghulam Ali
Mobile Number: 1
Home Number: 11
Work Number: 111
Email Address: @
Birthday: 123
```