Internship Program (Batch 2)

Task.#.2

Name: Mian Muhammad Awais

Section: C++ (Programming)

Creating a Contact Management System:

Objective:

- Implement a simple system to manage contacts.

Description:

- Develop a C++ program that allows users to add, view, and delete contacts. Each contact should have a name and a phone number.

Key Steps:

- Defining a Contact class with appropriate attributes
- Using vectors to store contact objects
- Implementing functions for adding, viewing, and deleting contacts
- Providing a menu-driven interface for user interaction

_____

✞ Code with explaination:

1. Includes:

```
#include <iostream>
#include <vector>
#include <string>
#include <limits>
using namespace std;
```

- **iostream: For input and output operations.**
- **vector: For handling dynamic arrays.**
- **string: For handling strings.**
- **limits: For defining properties of fundamental types (used for input validation).**

2. Contact Class:

```
class Contact {
private:
    string name;
    string phoneNumber;

public:
    Contact(const string& n, const string& p) : name(n), phoneNumber(p) {}

    string getName() const { return name; }
    string getPhoneNumber() const { return phoneNumber; }
};
```

- This class represents a contact with two private data members: name and phoneNumber.
- The constructor Contact(const string& n, const string& p) initializes these members.
- getName() and getPhoneNumber() are public member functions that return the name and phone number, respectively.

3. **ContactManager Class:**

```
class ContactManager {
private:
    vector<Contact> contacts;

public:
    void addContact(const string& name, const string& phoneNumber) {
        contacts.emplace_back(name, phoneNumber);
        cout << "Contact added successfully.\n";
    }

    void viewContacts() const {
        if (contacts.empty()) {
            cout << "No contacts found.\n";
            return;
        }

        cout << "Contacts:\n";
        for (size_t i = 0; i < contacts.size(); ++i) {
            cout << i + 1 << ". " << contacts[i].getName() << " - " << contacts[i].getPhoneNumber() << "\n";
        }
    }

    void deleteContact(size_t index) {
        if (index >= 1 && index <= contacts.size()) {
            contacts.erase(contacts.begin() + index - 1);
            cout << "Contact deleted successfully.\n";
        } else {
            cout << "Invalid contact number.\n";
        }
    }
};
```

- This class manages a list of contacts using a vector of Contact objects.
- Methods:
    - addContact(const string& name, const string& phoneNumber): Adds a new contact to the contacts vector.
    - viewContacts() const: Displays the list of contacts. If there are no contacts, it informs the user.
    - deleteContact(size_t index): Deletes a contact by its position in the list (1-based index). It checks if the index is valid before deletion.

4. **Menu Display Function:**

```
49  void displayMenu() {
50      cout << "\nContact Management System\n";
51      cout << "1. Add Contact\n";
52      cout << "2. View Contacts\n";
53      cout << "3. Delete Contact\n";
54      cout << "4. Exit\n";
55      cout << "Enter your choice: ";
56  }
57
```

- This function displays the main menu for the contact management system.

5. **Main Function:**

```
58  int main() {
59      ContactManager manager;
60      int choice;
61      string name, phoneNumber;
62      size_t index;
63
64      while (true) {
65          displayMenu();
66          cin >> choice;
67          cin.ignore(numeric_limits<streamsize>::max(), '\n');
68
69          switch (choice) {
70              case 1:
71                  cout << "Enter name: ";
72                  getline(cin, name);
73                  cout << "Enter phone number: ";
74                  getline(std::cin, phoneNumber);
75                  manager.addContact(name, phoneNumber);
76                  break;
77              case 2:
78                  manager.viewContacts();
79                  break;
80              case 3:
81                  cout << "Enter the number of the contact to delete: ";
82                  cin >> index;
83                  manager.deleteContact(index);
84                  break;
85              case 4:
86                  cout << "Exiting program. Goodbye!\n";
87                  return 0;
88              default:
89                  cout << "Invalid choice. Please try again.\n";
90          }
91      }
92
93      return 0;
94  }
```

Workflow:

- The program enters an infinite loop displaying the menu and processing user input.
- Based on the user's choice, it either adds a contact, views all contacts, deletes a contact, or exits the program.
- `cin.ignore(numeric_limits<streamsize>::max(), '\n');` is used to discard any leftover characters in the input buffer after reading the user's choice.
- The `switch` statement handles the user's menu selection:
    - Case `1`: Prompts for a name and phone number, then adds the contact.
    - Case `2`: Displays the list of contacts.
    - Case `3`: Prompts for the contact number to delete.
    - Case `4`: Exits the program.
    - Default: Handles invalid menu choices.

Output:

```
Contact Management System
1. Add Contact
2. View Contacts
3. Delete Contact
4. Exit
Enter your choice: 1
Enter name: Awais
Enter phone number: 03045808111
Contact added successfully.

Contact Management System
1. Add Contact
2. View Contacts
3. Delete Contact
4. Exit
Enter your choice: 1
Enter name: Huzaifa
Enter phone number: 0374826284
Contact added successfully.

Contact Management System
1. Add Contact
2. View Contacts
3. Delete Contact
4. Exit
Enter your choice: 2
Contacts:
1. Awais - 03045808111
2. Huzaifa - 0374826284
```

```
Contact Management System
1. Add Contact
2. View Contacts
3. Delete Contact
4. Exit
Enter your choice: 3
Enter the number of the contact to delete: 2
Contact deleted successfully.

Contact Management System
1. Add Contact
2. View Contacts
3. Delete Contact
4. Exit
Enter your choice: 4
Exiting program. Goodbye!
```
