OOP Project

Contact Manager

Submitted by

Roopesh O R
Roshna Palatty Santhosh
Sadhnan Shameem Thappi
Sandeep S

Abstract

Here we have implemented a simple contact manager in which a user can store and manage contacts. The program saves the contacts in a file in the same directory (where the binary is situated). It saves the contacts in the CSV format. This makes it easy to port data to other applications as well.
The application can do following operations:

1. **List contacts**: Lists name and phone number of all contacts along with an index (starting from 1)
2. **Find contacts**: Find all contacts whose name *contains* the search query. It will do a case-insensitive search.
3. **Create new contact**: User will be asked to enter name and phone number and it will be saved in the file
4. **Edit existing contact**: The user will be asked the index of contact (starting from 1) to edit. Then, a new name and phone number will be asked for for that contact. The values will not be changed if user types nothing and hits enter.
5. **Delete a contact**: The user will be asked to the index of contact. Only one contact can be deleted at a time.

After each manipulations (create/edit/delete) the contact will be immediately saved to the file so that no data will be lost due to any sudden interruptions.

Source Code

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <sstream>
#include <algorithm>

#define FILENAME "contacts.txt"
using namespace std;

struct Contact {
  string name;
  string phone_number;
};


/////////// Utility functions ///////////

/// Converts a string to lowercase
void toLower(string& str) {
  transform(str.begin(), str.end(), str.begin(), ::tolower);
```

```cpp
}

/// Asks user to input a index. The function validates the index
/// and returns it if its valid. if its not valid it returns -1
int getIndexFromUser(int maxIndex) {
  int index;
  cout << "Enter contact index to edit: ";
  cin >> index;
  cin.ignore();

  if (index > maxIndex || index < 1) {
    cout << "invalid index" << endl;
    return -1;
  }

  return index - 1;
}

/////////// Contact Manager functions ///////////

/// reads the contact file and loads them to program's memory.
/// ie, it reads the file converts it to a Contactvector and returns it.
vector<Contact> loadContacts() {
  vector<Contact> contacts;
  ifstream file(FILENAME);
  if (file.is_open()) {
    string line;
    while (getline(file, line)) {
      stringstream ss(line);
      string name, phone_number;
      getline(ss, name, ',');
      getline(ss, phone_number);
      contacts.push_back({name, phone_number});
    }
    file.close();
  }
  return contacts;
}

// saves contacts in program memory to the file
void saveContacts(vector<Contact>& contacts) {
  ofstream file(FILENAME);
  if (file.is_open()) {
    for (int i = 0; i < contacts.size(); i++) {
      Contact contact = contacts[i];
      file << contact.name << "," << contact.phone_number << endl;
    }
    file.close();
  }
}

/// lists contacts
void listContacts(vector<Contact>& contacts) {
  if (contacts.empty()) {
    cout << "No contacts found." << endl;
    return;
  }

  cout << "Contacts:" << endl;
  for (int i = 0; i < contacts.size(); ++i) {
    cout << i + 1 << ". "
         << contacts[i].name << " - "
         << contacts[i].phone_number << endl;
  }
}

// creates a new contact by asking name & phone number from user
void createContact(vector<Contact>& contacts) {
  string name, phone_number;
  cout << "Name: ";
  getline(cin, name);
  cout << "Phone number: ";
  getline(cin, phone_number);
  contacts.push_back({name, phone_number});

  saveContacts(contacts);
  cout << "Contact added successfully" << endl;
}

/// edits a contact based on the position of the
/// contact (number that is displayed on the list)
void editContact(vector<Contact>& contacts) {
  int index = getIndexFromUser(contacts.size());
  if (index < 0) return;

  string old_name = contacts[index].name;
  string old_phone = contacts[index].phone_number;
  string new_name, new_phone_number;

  cout << "Enter new name (" << old_name << "): ";
  getline(cin, new_name);
  if (!new_name.empty()) {
    contacts[index].name = new_name;
  }

  cout << "Enter new phone number (" << old_phone << "): ";
  getline(cin, new_phone_number);
  if (!new_phone_number.empty()) {
    contacts[index].phone_number = new_phone_number;
  }

  saveContacts(contacts);
  cout << "Contact edited successfully" << endl;
}

/// Deletes contact at a particular location
void deleteContact (vector<Contact>& contacts) {
  int index = getIndexFromUser(contacts.size());
  if (index < 0) return;
  contacts.erase(contacts.begin() + index);
  saveContacts(contacts);
  cout << "Contact deleted successfully" << endl;
}

/// returns index of all contacts whose name contains the
/// name we're searching (searchQuery). The function
/// goes through all contacts and checks for any match
vector<int> matchingContactIndexes(
  vector<Contact>& contacts,
  string searchQuery
) {
  // to match all cases, convert both name and query to lowercase
  toLower(searchQuery);
  vector<int> foundIndexes;
  for (int i = 0; i < contacts.size(); i++) {
    string name = contacts[i].name;
    toLower(name);
    if (name.find(searchQuery) != string::npos) {
      foundIndexes.push_back(i);
    }
  }
  return foundIndexes;
}

/// takes a search query from user and lists all matching names
/// from the contacts
void searchContact(vector<Contact>& contacts) {
  string name;
  cout << "Enter name to search: ";
  getline(cin, name);

  vector<int> matchingIndexes = matchingContactIndexes(contacts, name);
  if (matchingIndexes.empty()) {
    cout << "No contacts found" << endl;
    return;
  }

  cout << endl;
  for (int i = 0; i < matchingIndexes.size(); i++) {
    int index = matchingIndexes[i];
    cout << contacts[index].name
      << " - " << contacts[index].phone_number << endl;
  }
}

int main() {
  vector<Contact> contacts = loadContacts();

  int choice;
  cout << endl << "------Contact Manager------";
  do {
    cout << endl;
    cout << "(1) List   (2) Find   (3) Create" << endl;
    cout << "(4) Edit   (5) Delete (6) Exit" << endl;
    cout << "Enter your choice: ";
    cin >> choice;
    cin.ignore();

    switch (choice) {
      case 1:
        listContacts(contacts);
        break;
      case 2:
        searchContact(contacts);
        break;
      case 3:
        createContact(contacts);
```

```cpp
          break;
        case 4:
          editContact(contacts);
          break;
        case 5:
          deleteContact(contacts);
          break;
        case 6:
          saveContacts(contacts);
          cout << "Exiting..." << endl;
          break;
        default:
          cout << "Invalid choice." << endl;
      }
    } while (choice != 6);

    return 0;
  }
```

Output

```
------Contact Manager------
(1) List  (2) Find   (3) Create
(4) Edit  (5) Delete (6) Exit
Enter your choice: 3
Name: Amil
Phone number: +91193720382
Contact added successfully

(1) List  (2) Find   (3) Create
(4) Edit  (5) Delete (6) Exit
Enter your choice: 3
Name: Sandra C
Phone number: 9927362332
Contact added successfully

(1) List  (2) Find   (3) Create
(4) Edit  (5) Delete (6) Exit
Enter your choice: 3
Name: mukesh
Phone number: 188333371
Contact added successfully

(1) List  (2) Find   (3) Create
(4) Edit  (5) Delete (6) Exit
Enter your choice: 3
Name: Mukeshan
Phone number: 9823472342
Contact added successfully

(1) List  (2) Find   (3) Create
(4) Edit  (5) Delete (6) Exit
Enter your choice: 1
Contacts:
```

```
1. Amil - +91193720382
2. Sandra C - 9927362332
3. mukesh - 188333371
4. Mukeshan  - 9823472342

(1) List  (2) Find   (3) Create
(4) Edit  (5) Delete (6) Exit
Enter your choice: 2
Enter name to search: muke

mukesh - 188333371
Mukeshan  - 9823472342

(1) List  (2) Find   (3) Create
(4) Edit  (5) Delete (6) Exit
Enter your choice: 4
Enter contact index to edit: 3
Enter new name (mukesh): Mukesh D
Enter new phone number (188333371):
Contact edited successfully

(1) List  (2) Find   (3) Create
(4) Edit  (5) Delete (6) Exit
Enter your choice: 1
Contacts:
1. Amil - +91193720382
2. Sandra C - 9927362332
3. Mukesh D - 188333371
4. Mukeshan  - 9823472342

(1) List  (2) Find   (3) Create
(4) Edit  (5) Delete (6) Exit
Enter your choice: 5
Enter contact index to edit: 1
Contact deleted successfully

(1) List  (2) Find   (3) Create
(4) Edit  (5) Delete (6) Exit
Enter your choice: 1
Contacts:
1. Sandra C - 9927362332
2. Mukesh D - 188333371
3. Mukeshan  - 9823472342

(1) List  (2) Find   (3) Create
(4) Edit  (5) Delete (6) Exit
Enter your choice: 5
Enter contact index to edit: 4
invalid index

(1) List  (2) Find   (3) Create
(4) Edit  (5) Delete (6) Exit
Enter your choice: 6
Exiting...
```