**main.cpp**

#include <iostream>

#include <windows.h>

#include <string>

#include "ContactsBook.h"

#include "Search.h"

using namespace std;

void printMenu()

{

system("color 6d");

system("cls");

cout << "\t\t------------------------------------------------------------------------" << endl;

cout << "\t\t| Press 1 to Add New Contact" << endl;

cout << "\t\t| Press 2 to edit Contact" << endl;

cout << "\t\t| Press 3 to delete Contact" << endl;

cout << "\t\t| Press 4 to merge Duplicates" << endl;

cout << "\t\t| Press 5 to store To in File" << endl;

cout << "\t\t| Press 6 to load From in File" << endl;

cout << "\t\t| Press 7 to print Contacts Sorted by First Name or Last Name" << endl;

cout << "\t\t| Press 8 to print all Contacts" << endl;

cout << "\t\t| Press 9 to search contacts by First Name, Last Name or Number" << endl;

cout << "\t\t| Press 10 to display Count of Contacts" << endl;

cout << "\t\t| Press 0 to exit" << endl;

cout << "\t\t------------------------------------------------------------------------" << endl;

}

int main()

{

system("color 0A");

int size = 0;

cout << "\n\n\t\t\t---------------------------------------------------- ";

cout << "\n\t\t\t| |";

cout << "\n\t\t\t| Contact Book C++ |";

cout << "\n\t\t\t| |";

cout << "\n\t\t\t---------------------------------------------------- \n\n";

Sleep(5000);

cout << "\n\n\n\n\t\t\t\t--------------------------------- ";

cout << "\n\t\t\t\t| |";

cout << "\n\t\t\t\t| Talha Adullah Bangyal |";

cout << "\n\t\t\t\t| Abdullah Hissan |";

cout << "\n\t\t\t\t| |";

cout << "\n\t\t\t\t--------------------------------- \n\n";

Sleep(5000);

system("CLS");

cout << "Enter the size of the ContactsBook\_\_\_\_\_";

cin >> size;

ContactsBook book(size);

cout << endl;

while (true) {

printMenu();

int choice;

cout << "Enter the value: ";

cin >> choice;

cout << endl;

switch (choice) {

case 1: {

// Add New Contact

string firstName, lastName, phone, email;

string street, country, house;

cout << "Enter first name: ";

cin.ignore();

getline(cin, firstName);

cout << "Enter last name: ";

getline(cin, lastName);

cout << "Enter phone: ";

getline(cin, phone);

cout << "Enter email: ";

getline(cin, email);

cout << "Enter house: ";

getline(cin, house);

cout << "Enter street: ";

getline(cin, street);

cout << "Enter country: ";

getline(cin, country);

Address address(street, country, house);

Contact contact(firstName, lastName, phone, email, address);

book.addContact(contact);

cout << endl;

break;

}

case 2: {

// Edit Contact

string first, last;

cout << "Enter First name: ";

cin >> first;

cout << "Enter Last name: ";

cin >> last;

book.UpdateContact(first, last);

break;

}

case 3: {

// delete Contact

string first, last;

cout << "Enter First name: ";

cin >> first;

cout << "Enter Last name: ";

cin >> last;

book.DeleteContact(first, last);

break;

}

case 4: {

// Merge Duplicates contacts

book.mergeDuplicates();

break;

}

case 5: {

// Store To File

string filename;

cout << "Enter file name: ";

cin.ignore();

getline(cin, filename);

book.storeToFile(filename);

cout << endl;

cout << "Data has been stored in file" << endl;

break;

}

case 6: {

// Load From File

string filename;

cout << "Enter file name: ";

cin.ignore();

getline(cin, filename);

book.loadFromFile(filename);

cout << endl;

break;

}

case 7: {

string choice;

cout << "Press f for first name or l for last name: ";

cin.ignore();

getline(cin, choice);

if (choice == "f" || choice == "F") {

book.printContactsSortedByFirstName();

}

else if (choice == "l" || choice == "L") {

book.printContactsSortedByLastName();

}

else {

cout << "Wrong input." << endl;

}

break;

}

case 8: {

// Print Contacts

cout << "Contact are " << endl;

book.printContacts();

cout << endl;

break;

}

case 9: {

// Search contacts (all three choices)

int choice;

cout << "Press 1 to Search contact by first name" << endl;

cout << "Press 2 to Search contact by last name" << endl;

cout << "Press 3 to Search contact phone number" << endl;

cin >> choice;

string searchTerm;

switch (choice) {

case 1: {

cout << "Enter first name to search: ";

cin.ignore();

getline(cin, searchTerm);

Search search(book);

search.searchByFirstName(searchTerm);

break;

}

case 2: {

cout << "Enter last name to search: ";

cin.ignore();

getline(cin, searchTerm);

Search search(book);

search.searchByLastName(searchTerm);

break;

}

case 3: {

cout << "Enter phone number to search: ";

cin.ignore();

getline(cin, searchTerm);

Search search(book);

search.searchByPhoneNumber(searchTerm);

break;

}

default: {

cout << "Wrong input" << endl;

}

break;

}

}

case 10: {

cout << "Count of Contacts are: " << book.getCount() << endl;

break;

}

case 0: {

return 0;

}

default: {

cout << "Are you mad!!! You entered wrong input." << endl;

}

}

}

return 0;

}

**Address.h**

#pragma once

#ifndef ADDRESS\_H

#define ADDRESS\_H

#include <iostream>

using namespace std;

class Address {

public:

string street;

string country;

string house;

Address();

Address( string street, string country, string house);

~Address();

void setStreet(string street);

void setCountry(string country);

void setHouse(string house);

string getStreet() const;

string getCountry() const;

string getHouse() const;

};

#endif

**Address.cpp**

#include "Address.h"

Address::Address() : street(""), country(""), house("") {}

Address::Address( string street, string country, string house) : street(street), country(country), house(house) {}

Address::~Address() {}

void Address::setStreet( string street) {

this->street = street;

}

void Address::setCountry( string country) {

this->country = country;

}

void Address::setHouse( string house) {

this->house = house;

}

string Address::getStreet() const {

return street;

}

string Address::getCountry() const {

return country;

}

string Address::getHouse() const {

return house;

}

**Search.h**

#pragma once

#ifndef SEARCH\_H

#define SEARCH\_H

#include <iostream>

#include "ContactsBook.h"

using namespace std;

class Search {

public:

Search(ContactsBook& book);

~Search();

void searchByFirstName(const string& firstName);

void searchByLastName(const string& lastName);

void searchByPhoneNumber(const string& phoneNumber);

private:

ContactsBook& book;

};

#endif

**Search.cpp**

#include "Search.h"

//parameterised constructer

Search::Search(ContactsBook& book) : book(book) {}

//default distructer

Search::~Search() {}

//search by first name

void Search::searchByFirstName(const string& firstName) {

for (int i = 0; i < book.getCount(); i++) {

if (book.getContact(i).getFirstName() == firstName) {

cout << "Contact found: " << book.getContact(i).toString() << endl;

}

}

}

//search by last name

void Search::searchByLastName(const string& lastName) {

for (int i = 0; i < book.getCount(); i++) {

if (book.getContact(i).getLastName() == lastName) {

cout << "Contact found: " << book.getContact(i).toString() << endl;

}

}

}

//search by number

void Search::searchByPhoneNumber(const string& phoneNumber) {

for (int i = 0; i < book.getCount(); i++) {

if (book.getContact(i).getPhone() == phoneNumber) {

cout << "Contact found: " << book.getContact(i).toString() << endl;

}

}

}

**Contact.h**

#pragma once

#ifndef CONTACT\_H

#define CONTACT\_H

#include <iostream>

#include "Address.h"

#include <sstream>

using namespace std;

class Contact {

private:

string firstName;

string lastName;

string phone;

string email;

Address address;

public:

Contact();

Contact( string firstName, string lastName, string phone, string email, Address address);

~Contact();

void setFirstName(string firstName);

void setLastName(string lastName);

void setPhone(string phone);

void setEmail(string email);

void setAddress(Address address);

string getFirstName() const;

string getLastName() const;

string getPhone() const;

string getEmail() const;

Address getAddress() const;

string toString() const;

};

#endif

**Contact.cpp**

#include "Contact.h"

Contact::Contact() : firstName(""), lastName(""), phone(""), email(""), address(Address()) {}

Contact::Contact( string firstName, string lastName, string phone, string email, Address address)

: firstName(firstName), lastName(lastName), phone(phone), email(email), address(address) {}

Contact::~Contact() {}

void Contact::setFirstName( string firstName) {

this->firstName = firstName;

}

void Contact::setLastName( string lastName) {

this->lastName = lastName;

}

void Contact::setPhone( string phone) {

this->phone = phone;

}

void Contact::setEmail( string email) {

this->email = email;

}

void Contact::setAddress(Address address) {

this->address = address;

}

string Contact::getFirstName() const {

return firstName;

}

string Contact::getLastName() const {

return lastName;

}

string Contact::getPhone() const {

return phone;

}

string Contact::getEmail() const {

return email;

}

Address Contact::getAddress() const {

return address;

}

string Contact::toString() const {

stringstream ss;

ss << "First name: " << firstName << "\n";

ss << "Last name: " << lastName << "\n";

ss << "Phone: " << phone << "\n";

return ss.str();

}

**ContactBook.h**

#pragma once

#ifndef CONTACTSBOOK\_H

#define CONTACTSBOOK\_H

#include <iostream>

#include "Contact.h"

#include "Address.h"

using namespace std;

class ContactsBook {

public:

Contact\* contacts;

int size;

int count;

ContactsBook(int size);

~ContactsBook();

void addContact(Contact contact);

void printContacts() const;

void printContactsSortedByFirstName() const;

void printContactsSortedByLastName() const;

void mergeDuplicates();

void storeToFile(const string& filename) const;

void loadFromFile(const string& filename);

int getCount() const;

Contact& getContact(int index);

const Contact& getContact(int index) const;

void UpdateContact(string str\_first, string str\_last);

void DeleteContact(string str\_first, string str\_last);

};

#endif

**ContactBook.cpp**

#include "ContactsBook.h"

#include <fstream>

ContactsBook::ContactsBook(int size) : size(size), count(0) {

contacts = new Contact[size];

}

/\*==========================================end=======================================================\*/

ContactsBook::~ContactsBook() {

delete[] contacts;

}

/\*==========================================end=======================================================\*/

void ContactsBook::addContact(Contact contact) {

if (count < size) {

contacts[count] = contact;

count++;

}

else {

cerr << "ContactsBook is full." << endl;

}

}

/\*==========================================end=======================================================\*/

void ContactsBook::printContacts() const {

for (int i = 0; i < count; i++) {

cout << "First Name: " << contacts[i].getFirstName() << endl;

cout << "Last Name: " << contacts[i].getLastName() << endl;

cout << "Phone: " << contacts[i].getPhone() << endl;

cout << "Email: " << contacts[i].getEmail() << endl;

cout << "Address: " << contacts[i].getAddress().getStreet() << ", "

<< contacts[i].getAddress().getCountry() << ", "

<< contacts[i].getAddress().getHouse() << endl;

cout << endl;

}

}

/\*==========================================end=======================================================\*/

void ContactsBook::printContactsSortedByFirstName() const {

Contact tempContacts[100];

for (int i = 0; i < count; i++) {

tempContacts[i] = contacts[i];

}

bool swapped;

do {

swapped = false;

for (int i = 1; i < count; i++) {

if (tempContacts[i - 1].getFirstName() > tempContacts[i].getFirstName()) {

Contact temp = tempContacts[i - 1];

tempContacts[i - 1] = tempContacts[i];

tempContacts[i] = temp;

swapped = true;

}

}

} while (swapped);

for (int i = 0; i < count; i++) {

cout << "First Name: " << tempContacts[i].getFirstName() << endl;

cout << "Last Name: " << tempContacts[i].getLastName() << endl;

cout << "Phone: " << tempContacts[i].getPhone() << endl;

cout << "Email: " << tempContacts[i].getEmail() << endl;

cout << "Address: " << tempContacts[i].getAddress().getStreet() << ", "

<< tempContacts[i].getAddress().getCountry() << ", "

<< tempContacts[i].getAddress().getHouse() << endl;

cout << endl;

}

}

/\*==========================================end=======================================================\*/

void ContactsBook::printContactsSortedByLastName() const {

Contact tempContacts[100];

for (int i = 0; i < count; i++) {

tempContacts[i] = contacts[i];

}

bool swapped;

do {

swapped = false;

for (int i = 1; i < count; i++) {

if (tempContacts[i - 1].getLastName() > tempContacts[i].getLastName()) {

Contact temp = tempContacts[i - 1];

tempContacts[i - 1] = tempContacts[i];

tempContacts[i] = temp;

swapped = true;

}

}

} while (swapped);

for (int i = 0; i < count; i++) {

cout << "First Name: " << tempContacts[i].getFirstName() << endl;

cout << "Last Name: " << tempContacts[i].getLastName() << endl;

cout << "Phone: " << tempContacts[i].getPhone() << endl;

cout << "Email: " << tempContacts[i].getEmail() << endl;

cout << "Address: " << tempContacts[i].getAddress().getStreet() << ", "

<< tempContacts[i].getAddress().getCountry() << ", "

<< tempContacts[i].getAddress().getHouse() << endl;

cout << endl;

}

}

/\*==========================================end=======================================================\*/

void ContactsBook::mergeDuplicates() {

for (int i = 0; i < count; i++) {

for (int j = i + 1; j < count; j++) {

if (contacts[i].getFirstName() == contacts[j].getFirstName() &&

contacts[i].getLastName() == contacts[j].getLastName()) {

for (int k = j; k < count - 1; k++) {

contacts[k] = contacts[k + 1];

}

count--;

j--;

}

}

for (int j = 0; j < i; j++) {

if (contacts[i].getFirstName() == contacts[j].getFirstName() &&

contacts[i].getLastName() == contacts[j].getLastName()) {

for (int k = i; k > j; k--) {

contacts[k] = contacts[k - 1];

}

count--;

break;

}

}

}

}

/\*==========================================end=======================================================\*/

void ContactsBook::storeToFile(const string& filename) const {

ofstream out(filename);

if (out.is\_open()) {

for (int i = 0; i < count; i++) {

out << contacts[i].getFirstName() << ","

<< contacts[i].getLastName() << ","

<< contacts[i].getPhone() << ","

<< contacts[i].getEmail() << ","

<< contacts[i].getAddress().getStreet() << ","

<< contacts[i].getAddress().getCountry() << ","

<< contacts[i].getAddress().getHouse() << endl;

}

out.close();

}

else {

cout << "Unable to open file: " << filename << endl;

}

}

/\*==========================================end=======================================================\*/

void ContactsBook::loadFromFile(const string& filename) {

ifstream in(filename);

if (in.is\_open()) {

count = 0;

char comma;

string firstName, lastName, phone, email, street, country, house;

while (in >> firstName >> comma >> lastName >> comma >> phone >> comma >> email >> comma >> street >> comma >> country >> comma >> house) {

Address address(street, country, house);

Contact contact(firstName, lastName, phone, email, address);

addContact(contact);

}

in.close();

}

else {

cout << "Unable to open file: " << filename << endl;

}

}

/\*==========================================end=======================================================\*/

int ContactsBook::getCount() const {

return count;

}

/\*==========================================end=======================================================\*/

Contact& ContactsBook::getContact(int index) {

return contacts[index];

}

/\*==========================================end=======================================================\*/

const Contact& ContactsBook::getContact(int index) const {

return contacts[index];

}

/\*==============================================end===================================================\*/

void ContactsBook::UpdateContact(string str\_first, string str\_last)

{

int re = 0;

string mail, num, houseno, city, country, street;

for (int i = 0; i < count; i++)

{

if (str\_first == contacts[i].getFirstName() && str\_last == contacts[i].getLastName())

{

cout << "enter first name:";

cin >> str\_first;

cout << "enter last name:";

cin >> str\_last;

cout << "enter mail:";

cin >> mail;

cout << "enter mobile number:";

cin >> num;

cout << "enter house:";

cin >> houseno;

cout << "enter city:";

cin >> city;

cout << "enter country:";

cin >> country;

cout << "enter street:";

cin >> street;

contacts[i].setFirstName(str\_first);

contacts[i].setLastName(str\_last);

contacts[i].setEmail(mail);

contacts[i].setPhone(num);

/\*contacts[i].address.setStreet(street);

contacts[i].address.setCountry(country);

contacts[i].address.setHouse(houseno);\*/

re++;

}

}

if (re == 0)

{

cout << "your data is not found in the ContactBook" << endl;

}

}

void ContactsBook::DeleteContact(std::string str\_first, std::string str\_last)

{

for (int i = 0; i < count; i++)

{

if (contacts[i].getFirstName() == str\_first && contacts[i].getLastName() == str\_last)

{

contacts[i].setFirstName("-1");

contacts[i].setFirstName("-1");

}

}

for (int i = 0; i < count; i++)

{

if (contacts[i].getFirstName() == "-1" && i + 1 < count)

{

swap(contacts[i], contacts[i + 1]);

}

}

int count = 0;

for (int i = 0; i < count; i++)

{

if (contacts[i].getFirstName() != "-1")

{

count++;

}

}

count = count;

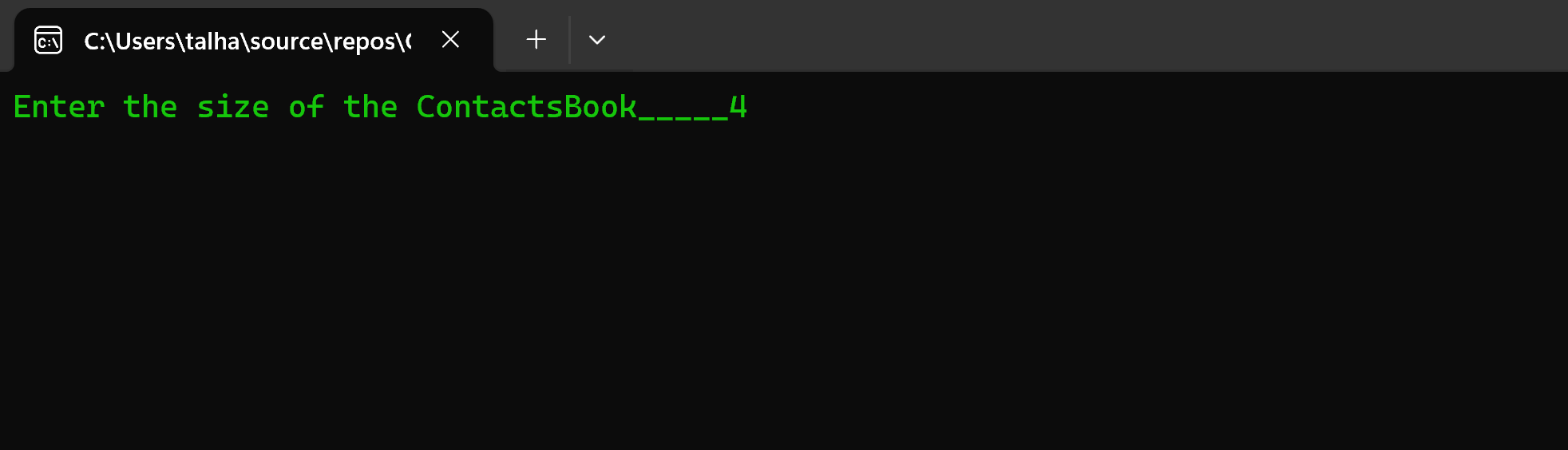
}

**Output**

**A screenshot of a computer

Description automatically generated**

**.**

****

**.**

**A screenshot of a computer

Description automatically generated**

**.**