**Start.h**

#pragma once

#include <iostream>

#include <windows.h>

#include <conio.h>

#include "Entities.h"

#include "Functions.h"

using namespace std;

void Start()

{

Human\* h1 = new Human{

new char[] {"Chevonne Lew"},

new char[] {"+44 4232 959154"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h2 = new Human{

new char[] {"Madelina Lanford"},

new char[] {"+44 7361 497981"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h3 = new Human{

new char[] {"Norton Ermintrude"},

new char[] {"+44 1911 854437"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h4 = new Human{

new char[] {"Felecia Gerry"},

new char[] {"+44 8911 820111"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h5 = new Human{

new char[] {"Kevin Fletcher"},

new char[] {"+44 8457 586732"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h6 = new Human{

new char[] {"David Morrison"},

new char[] {"+44 9911 800183"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h7 = new Human{

new char[] {"Jane Platt"},

new char[] {"+44 7178 152771"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h8 = new Human{

new char[] {"Andrew Young"},

new char[] {"+44 5457 039187"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h9 = new Human{

new char[] {"Raymond Babb"},

new char[] {"+44 4654 347825"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h10 = new Human{

new char[] {"Brendan Caden"},

new char[] {"+44 7368 147242"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h11 = new Human{

new char[] {"Glenys Horton"},

new char[] {"+44 0246 287237"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h12 = new Human{

new char[] {"Evafe Moore"},

new char[] {"+44 2236 684425"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h13 = new Human{

new char[] {"Suzanne Lock"},

new char[] {"+44 4585 746332"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h14 = new Human{

new char[] {"Mark Llewellyn"},

new char[] {"+44 3333 435432"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h15 = new Human{

new char[] {"Timose Rothwell"},

new char[] {"+44 2545 325774"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h16 = new Human{

new char[] {"Timothy Moss"},

new char[] {"+44 2562 234578"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h17 = new Human{

new char[] {"Evans David"},

new char[] {"+44 3787 324492"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h18 = new Human{

new char[] {"Nicholas Holmes"},

new char[] {"+44 4523 345443"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h19 = new Human{

new char[] {"Mary Potts"},

new char[] {"+44 0246 256244"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h20 = new Human{

new char[] {"Billy Skinner"},

new char[] {"+44 2236 342353"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h21 = new Human{

new char[] {"Charles Turner"},

new char[] {"+44 4382 265174"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h22 = new Human{

new char[] {"Tracey Bradshaw"},

new char[] {"+44 2568 527554"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h23 = new Human{

new char[] {"Lena Cameron"},

new char[] {"+44 5729 257357"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h24 = new Human{

new char[] {"Billy Moss"},

new char[] {"+44 6252 457292"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h25 = new Human{

new char[] {"Suresh Jones"},

new char[] {"+44 3452 346753"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

Human\* h26 = new Human{

new char[] {"Mariam Cape"},

new char[] {"+44 3653 463634"},

\_\_DATE\_\_,

\_\_TIME\_\_

};

auto humans = new Human \* [] {h1, h2, h3, h4, h5, h6,

h7, h8, h9, h10, h11, h12,

h13 ,h14, h15 ,h16, h17, h18,

h19, h20, h21, h22 ,h23, h24,

h25, h26};

Contact\* contact = new Contact{ humans,26 };

char selection = ' ';

ShowMenu();

while (true)

{

selection = 0;

cout << "\n Enter your choice : ";

cin >> selection;

system("cls");

if (selection == '1')

{

ShowAll(\*contact);

GoBackToMenu();

}

else if (selection == '2')

{

cout << "\n =========== SORTED A TO Z (A-Z) =========== "<< endl;

SortAToZ(\*contact);

ShowAll(\*contact);

GoBackToMenu();

}

else if (selection == '3')

{

cout << "\n =========== SORTED Z TO A (Z-A) =========== "<< endl;

SortAToZ(\*contact, true);

ShowAll(\*contact);

GoBackToMenu();

}

else if (selection == '4')

{

auto newhuman = GetNewHuman(\*contact);

if (newhuman != 0)

{

AddNewHuman(\*contact, \*newhuman);

SortAToZ(\*contact);

Sleep(2000);

system("cls");

ShowMenu();

}

else

{

Sleep(2000);

system("cls");

ShowMenu();

}

}

else if (selection == '5')

{

if (contact->count != 0)

{

cout << "\n How do you want to delete the contact ? " << endl;

cout << " By Name (n) | By Phone Number (p) - ";

char choice = ' ';

cin >> choice;

system("cls");

cout << "\n =========== DELETING A CONTACT ============" << endl;

if (choice == 'n' || choice == 'N')

{

ShowAll(\*contact);

cin.ignore();

cin.clear();

char\* name = new char[100]{};

cout << "\n Enter the name of the contact : ";

cin.getline(name, 100);

DeleteByName(\*contact, name );

}

else if (choice == 'p' || choice == 'P')

{

ShowAll(\*contact);

cin.ignore();

cin.clear();

char\* phonenumber = new char[100]{};

cout << "\n Enter the phone number of the contact : ";

cin.getline(phonenumber, 100);

DeleteByPhoneNumber(\*contact, phonenumber);

}

else

{

system("cls");

ColorTextRed();

cout << "\n Incorrect Input!" << endl;

Sleep(1500);

system("cls");

ShowMenu();

}

}

else

{

cout << "\n There is no one in the contact." << endl;

}

}

else if (selection == '6')

{

SearchEngine(\*contact);

}

else if (selection > 54 || selection < 49 )

{

system("cls");

ColorTextRed();

cout << "\n Incorrect Input!" << endl;

Sleep(1500);

system("cls");

ShowMenu();

}

}

}

**Source.cpp**

#include<iostream>

#include "Start.h"

using namespace std;

void Run()

{

Start();

}

int main()

{

Run();

return 0;

}

**Funtions.h**

#pragma once

#include <conio.h>

#include <iostream>

#include "Entities.h"

using namespace std;

void ColorTextRed()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 12);

}

void ColorTextGreen()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 10);

}

void ColorTextLightBlue()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 11);

}

void ColorTextPink()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 13);

}

void ColorTextYellow()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 14);

}

void ColorTextBlue()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 9);

}

void ShowMenu()

{

ColorTextYellow();

cout << "\n ======== MAIN MENU ========" << endl;

cout << "\n Show All Contacts 1" << endl;

cout << " Show Contacts A-Z 2 " << endl;

cout << " Show Contacts Z-A 3 " << endl;

cout << " Add New Contact 4 " << endl;

cout << " Delete Contact 5 " << endl;

cout << " Search Contact 6 " << endl;

cout << "\n ===========================" << endl;

}

void GoBackToMenu()

{

cout << "\n";

system("pause");

system("cls");

ShowMenu();

}

void ShowHuman(const Human& human) {

cout << " Name : " << human.name << endl;

cout << " Mobile Phone Number : " << human.phonenumber << endl;

cout << " Date of creation : " << human.createdDate << " " << human.createdTime << endl;

}

void ShowAll(const Contact& contact) {

ColorTextYellow();

if (contact.count != 0)

{

cout << "\n ================ CONTACTS =================" << endl;

cout << "\n";

for (size\_t i = 0; i < contact.count; i++)

{

cout << " -------------------------------------------" << endl;

ShowHuman(\*contact.humans[i]);

}

cout << " -------------------------------------------" << endl;

cout << "\n ===========================================" << endl;

}

else

{

ColorTextRed();

cout << "\n There is no one in the contact." << endl;

}

}

void GoToRowColumn(int row\_position, int col\_position)

{

HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);

int x\_position = col\_position, y\_position = row\_position;

COORD screen;

HANDLE hOutput = GetStdHandle(STD\_OUTPUT\_HANDLE);

screen.X = x\_position;

screen.Y = y\_position;

SetConsoleCursorPosition(hOutput, screen);

}

char\* LowerCase(char\* str)

{

int l = strlen(str);

char\* newstr = new char[l + 1]{};

for (int x = 0; x < l; x++)

{

if (str[x] >= 65 && str[x] <= 90)

{

newstr[x] = str[x] + 32;

}

else

{

newstr[x] = str[x];

}

}

newstr[l + 1] = '\0';

return newstr;

}

char\* UpperCase(char\* str)

{

int l = strlen(str);

char\* newstr = new char[l + 1]{};

for (int x = 0; x < l; x++)

{

if (str[x] >= 97 && str[x] <= 122)

{

newstr[x] = str[x] - 32;

}

else

{

newstr[x] = str[x];

}

}

newstr[l + 1] = '\0';

return newstr;

}

bool IsSuffix(char\* name, char\* word)

{

int l\_name = strlen(name);

int l\_word = strlen(word);

int counter = 0;

name = LowerCase(name);

word = LowerCase(word);

counter = 0;

for (int x = 0; x < l\_name; x++)

{

counter = 0;

for (int y = 0; y < l\_word; y++)

{

if (name[x + y] == word[y])

{

counter++;

}

}

if (counter == l\_word)

{

return true;

}

}

return false;

}

bool IsPrefixLetter(char\* name, char\* word)

{

int l = strlen(word);

char\* name\_lowercase = LowerCase(name);

char\* word\_lowercase = LowerCase(word);

for (int x = 0; x < l; x++)

{

if (name\_lowercase[x] != word\_lowercase[x])

{

return false;

}

}

return true;

}

bool ExistInArray(int\* arr, int size, int number)

{

for (int x = 0; x < size; x++)

{

if (arr[x] == number)

{

return true;

}

}

return false;

}

char\* AddPlusIfNeeded(char\* phone\_number);

void ShowAllForGetch(Contact& contact, char\* word)

{

cout << endl;

int l = 0;

int counter = 0;

int index = 0;

int\* arr = new int[index] {};

// First displaying contacts which begins with searched word

for (int x = 0; x < contact.count; x++)

{

l = strlen(contact.humans[x]->name);

for (int y = 0; y < l; y++)

{

if (IsPrefixLetter(contact.humans[x]->name, word) || IsPrefixLetter(contact.humans[x]->phonenumber, word))

{

ColorTextYellow();

counter++;

cout << "\n -------------------------------------------" << endl;

ShowHuman(\*contact.humans[x]);

arr[index] = x;

index++;

cout << " -------------------------------------------" << endl;

break;

}

}

}

// Then showing the rest of them

for (int x = 0; x < contact.count; x++)

{

l = strlen(contact.humans[x]->name);

for (int y = 0; y < l; y++)

{

if (IsSuffix(contact.humans[x]->name, word) || IsSuffix(contact.humans[x]->phonenumber, word))

{

// Not to show the same person again

if (!ExistInArray(arr, index, x))

{

ColorTextYellow();

counter++;

cout << "\n -------------------------------------------" << endl;

ShowHuman(\*contact.humans[x]);

cout << " -------------------------------------------" << endl;

break;

}

}

}

}

if (counter == 0)

{

ColorTextRed();

cout << "\n No results match your search :(" << endl;

}

}

void SearchEngine(Contact& contact)

{

ColorTextYellow();

cout << "\n - - Press ENTER to finish the search - -" << endl;

cout << "\n Search someone : ";

char\* word = new char[100]{};

int i = 0;

cin.ignore();

cin.clear();

while (ShowCursor(false) >= 0);

ShowCursor(false);

while ((word[i] = \_getch()) != '\r')

{

GoToRowColumn(3, i + 18);

if (int(word[i]) == 8) // Backspace

{

if (i > 0)

{

word[i] = NULL;

word[i - 1] = NULL;

cout << "\b \b";

i--;

if (i == 0)

{

ColorTextYellow();

system("cls");

cout << "\n - - Press ENTER to finish the search - -" << endl;

cout << "\n Search someone : ";

}

else

{

ShowAllForGetch(contact, word);

GoToRowColumn(3, i + 18);

}

}

GoToRowColumn(3, i + 18);

}

else

{

ColorTextYellow();

system("cls");

cout << "\n - - Press ENTER to finish the search - -" << endl;

cout << "\n Search someone : ";

int no = word[i];

i++;

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

{

cout << word[x];

}

ShowAllForGetch(contact, word);

GoToRowColumn(3, i + 18);

}

}

Sleep(300);

system("cls");

ShowMenu();

}

inline bool isDigit(char ch)

{

if ((int)ch > 57 || (int)ch < 48)

{

return false;

}

return true;

}

bool isFormatted(char\* phone\_number)

{

if (phone\_number[0] != '+')

return false;

for (int x = 1; x < 3; x++)

{

if (!isDigit(phone\_number[x]))

return false;

}

if (phone\_number[3] != ' ')

return false;

for (int x = 4; x < 8; x++)

{

if (!isDigit(phone\_number[x]))

return false;

}

if (phone\_number[8] != ' ')

return false;

for (int x = 9; x < 15; x++)

{

if (!isDigit(phone\_number[x]))

return false;

}

return true;

}

bool is12Digited(char\* phone\_number)

{

int l = strlen(phone\_number);

int counter = 0;

for (int x = 0; x < l; x++)

{

if (isDigit(phone\_number[x]))

counter++;

}

if (counter == 12)

return true;

return false;

}

char\* AddPlusIfNeeded(char\* phone\_number)

{

if (phone\_number[0] != '+')

{

int l = strlen(phone\_number);

char\* newNumber = new char[l + 2];

newNumber[0] = '+';

for (int x = 1; x < l + 2; x++)

{

newNumber[x] = phone\_number[x - 1];

}

newNumber[l + 2] = '\0';

phone\_number = newNumber;

newNumber = nullptr;

return phone\_number;

}

return phone\_number;

}

int numberOfSpaces(const char\* phone\_number)

{

int counter = 0;

int l = strlen(phone\_number);

for (int x = 0; x < l; x++)

{

if (phone\_number[x] == ' ')

{

counter++;

}

}

return counter;

}

char\* RemoveSpacesFromNumber(char\* phone\_number)

{

int numberOfSpacesV = numberOfSpaces(phone\_number);

int l = strlen(phone\_number);

char\* newPhoneNumber = new char [l - numberOfSpacesV + 1]{};

int index = 0;

for (int x = 0; x < l+1; x++)

{

if (phone\_number[x] != 32)

{

newPhoneNumber[index] = phone\_number[x];

index++;

}

}

newPhoneNumber[l - numberOfSpacesV + 1] = '\0';

phone\_number = newPhoneNumber;

newPhoneNumber = nullptr;

return phone\_number;

}

char\* FormatNumber(char\* phone\_number)

{

phone\_number = AddPlusIfNeeded(phone\_number);

phone\_number = RemoveSpacesFromNumber(phone\_number);

char\* newPhoneNumber = new char [16] {};

for (int x = 0; x < 4; x++)

{

newPhoneNumber[x] = phone\_number[x];

}

newPhoneNumber[3] = ' ';

for (int y = 4; y < 8; y++)

{

newPhoneNumber[y] = phone\_number[y - 1];

}

newPhoneNumber[8] = ' ';

for (int z = 9; z < 15; z++)

{

newPhoneNumber[z] = phone\_number[z - 2];

}

newPhoneNumber[15] = '\0';

phone\_number = newPhoneNumber;

newPhoneNumber = nullptr;

return phone\_number;

}

bool UniqueName(Contact& contact, const char\* name)

{

for (int x = 0; x < contact.count; x++)

{

if (strcmp(contact.humans[x]->name, name) == 0)

{

return false;

}

}

return true;

}

bool UniquePhoneNumber(Contact& contact,const char\* phone\_number)

{

for (int x = 0; x < contact.count; x++)

{

if (strcmp(contact.humans[x]->phonenumber, phone\_number) == 0)

{

return false;

}

}

return true;

}

Human\* GetNewHuman(Contact& contact)

{

cout << "\n ================= CREATING A NEW CONTACT ==================" << endl;

cin.ignore();

cin.clear();

char\* name = new char[100]{};

cout << "\n Enter name of contanct : ";

cin.getline(name, 100);

char\* phonenumber = new char[100]{};

cout << " Enter phone number of contanct (12 digit) : ";

cin.getline(phonenumber, 100);

cout << "\n ===========================================================" << endl;

phonenumber = FormatNumber(phonenumber);

if (!is12Digited(phonenumber))

{

ColorTextRed();

cout << "\n The phone number should be 12-digits." << endl;

cout << " New contact was not created!" << endl;

return 0;

}

else if (!UniqueName(contact, name))

{

ColorTextRed();

cout << "\n There is a contact with this name." << endl;

cout << " New contact was not created!" << endl;

return 0;

}

else if (!UniquePhoneNumber(contact, phonenumber))

{

ColorTextRed();

cout << "\n There is a contact with this phone number." << endl;

cout << " New contact was not created!" << endl;

return 0;

}

else

{

Human\* human = new Human{ name,phonenumber,\_\_DATE\_\_,\_\_TIME\_\_ };

return human;

}

}

void AddNewHuman(Contact& contact, Human& human)

{

if (!isFormatted(human.phonenumber))

{

human.phonenumber = FormatNumber(human.phonenumber);

}

int count = contact.count;

auto newhumans = new Human \* [count + 1]{};

for (size\_t i = 0; i < count; i++)

{

newhumans[i] = contact.humans[i];

}

newhumans[count] = new Human{ human };

contact.humans = newhumans;

newhumans = NULL;

contact.count++;

ColorTextGreen();

cout << "\n New contact was added successfully!";

}

void SortAToZ(Contact& contact, bool reverse = false) {

int data = 1;

if (reverse)

{

data = -1;

}

for (size\_t i = 0; i < contact.count - 1; i++)

{

for (size\_t k = 0; k < contact.count - i - 1; k++)

{

if (strcmp(contact.humans[k]->name, contact.humans[k + 1]->name) == data) {

auto temp = contact.humans[k];

contact.humans[k] = contact.humans[k + 1];

contact.humans[k + 1] = temp;

}

}

}

}

int GetHumanIndexName(Contact& contact, char\* name)

{

int count = contact.count;

name = LowerCase(name);

int l = strlen(contact.humans[0]->name);

char\* newName = new char[l+1] {};

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

{

newName = LowerCase(contact.humans[x]->name);

if (strcmp(newName, name) == 0)

{

return x;

}

}

return -1;

}

void DeleteByName(Contact& contact, char\* name)

{

int count = contact.count;

auto newhumans = new Human \* [count - 1]{};

int indexOFHuman = GetHumanIndexName(contact, name);

if (indexOFHuman >= 0)

{

ColorTextGreen();

cout << "\n \'" << contact.humans[indexOFHuman]->name << "\' was deleted successfully!" << endl;

int index = 0;

for (int x = 0; x < indexOFHuman; x++)

{

newhumans[index] = contact.humans[x];

index++;

}

for (int y = indexOFHuman+1; y < count; y++)

{

newhumans[index] = contact.humans[y];

index++;

}

contact.humans = newhumans;

newhumans = NULL;

contact.count--;

Sleep(2000);

system("cls");

ShowAll(contact);

GoBackToMenu();

}

else

{

ColorTextRed();

cout << "\n There is no one with this name in the contact." << endl;

Sleep(2000);

system("cls");

ShowMenu();

}

}

int GetHumanIndexNumber(Contact& contact, const char\* phonenumber)

{

int count = contact.count;

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

{

if (strcmp(contact.humans[x]->phonenumber, phonenumber) == 0)

{

return x;

}

}

return -1;

}

void DeleteByPhoneNumber(Contact& contact, char\* phonenumber)

{

if (!isFormatted(phonenumber))

{

phonenumber = FormatNumber(phonenumber);

}

int count = contact.count;

auto newhumans = new Human \* [count - 1]{};

int indexOFHuman = GetHumanIndexNumber(contact, phonenumber);

if (indexOFHuman >= 0)

{

ColorTextGreen();

cout << "\n \'" << contact.humans[indexOFHuman]->name << "\' was deleted successfully!" << endl;

int index = 0;

for (int x = 0; x < indexOFHuman; x++)

{

newhumans[index] = contact.humans[x];

index++;

}

for (int y = indexOFHuman + 1; y < count; y++)

{

newhumans[index] = contact.humans[y];

index++;

}

contact.humans = newhumans;

newhumans = NULL;

contact.count--;

Sleep(2000);

system("cls");

ShowAll(contact);

GoBackToMenu();

}

else

{

ColorTextRed();

cout << "\n There is no one with this phone number in the contact." << endl;

Sleep(2000);

system("cls");

ShowMenu();

}

}

**Entities.h**

#pragma once

struct Human {

char name;

char phonenumber;

const char createdDate;

const char createdTime;

};

struct Contact {

Human humans;

int count = 0;

};