Министерство образования Республики Беларусь

Учреждение образования

«Гомельский государственный университет   
имени Франциска Скорины»

Факультет физики и

информационных технологий

Кафедра общей физики

Разработка программы

«Продажа авиабилетов»

Курсовая работа

по учебной дисциплине

"Технология разработки программного обеспечения"

Исполнитель:

студент группы МС-12 Дахненко М.С.

Руководитель:

ст. преподаватель Грищенко В.В.

**Реферат**

**Цель курсовой работы:**

**Задача курсовой работы:**

**Курсовая работа:**

**Ключевые слова:**

**Предмет исследования:**

**Методы исследования:** разработка программы.

**Вывод:**

Содержание

**1. Введение**

* 1. **Язык Си**

Си— универсальный язык программирования, который завоевал популярность у программистов, благодаря сочетанию возможностей языков программирования высокого и низкого уровней. Большинство программистов предпочитают использовать язык Си для серьезных разработок потому, что их привлекают такие особенности языка, как свобода выражения мыслей, мобильность и чрезвычайная доступность. Язык Си даёт возможность программисту осуществлять непосредственный доступ к ячейкам памяти и регистрам компьютера, требуя при этом знания особенностей функционирования ЭВМ. В этом Си схож с языком низкого уровня — ассемблером, хотя на самом деле он представляет собой гораздо более мощное средство решения трудных задач и создания сложных программных систем. Си оказал существенное влияние на развитие индустрии программного обеспечения, а его синтаксис стал основой для таких языков программирования как C++, C#, Java, PHP и др.

Ключевой характеристикой языка программирования Си является его минималистичность. Авторы задумывали его, как простой инструмент для написания программ, которые будут легко компилироваться с использованием однопроходного компилятора. При этом чтобы в результате для каждого элемента программы было задействовано как можно меньше машинных команд, и чтобы примененные базовые составляющие языка никак не касались библиотеки времени выполнения.

Основные особенности Си:

1. простая языковая база, из которой в стандартную библиотеку
2. вынесены многие существенные возможности, вроде математических функций или функций работы с файлами;
3. ориентация на процедурное программирование;
4. система типов, предохраняющая от бессмысленных операций;
5. использование препроцессора для абстрагирования однотипных операций;
6. доступ к памяти через использование указателей;
7. небольшое число ключевых слов;
8. передача параметров в функцию по значению, а не по ссылке (передача по ссылке эмулируется с помощью указателей);
9. наличие указателей на функции и статических переменных;
10. области видимости имён;
11. структуры и объединения — определяемые пользователем собирательные типы данных, которыми можно манипулировать как одним целым.
12. В то же время в Си отсутствуют:
13. вложенные функции;
14. прямое возвращение нескольких значений из функций;
15. сопрограммы;
16. средства автоматического управления памятью;
17. встроенные средства объектно-ориентированного программирования;
18. средства функционального программирования.

**1.2 Цель курсовой работы и среда реализации кода**

Целью курсовой работы является получение студентами теоретических знаний и практических навыков по конструированию и реализации программ на языке высокого уровня в рамках процедурной парадигмы.

Курсовая работа была написана на языке Си в свободной интегрированной среде Dev-C++.

**2.Описание работы программы**

**2.1 Функции и структуры**

Разработка программы продажи авиабилетов состоит из авторизации под ролью администратора или пользователя, основной части кода и индивидуального задания. Код включает в себя ряд функций и структур, которые будут описаны ниже

**2.2 Функция void arrayNulling(char \*array, int size)**

Предназначена для зануления массив для сравнения

**2.3 Функция void reEntry(int \*stopmain)**

Обеспечивает повторный вход в систему

**2.4 Функция void userVerification(char \*login, char \*password)**

Обеспечивает проверку пользователя с базой пользователей

**2.5 Функция void logIn(int \*stopmain)**

Обеспечивает основной вход в программу с запросом логина и пароля

**2.6 Функция void defaultAdminCreate()**

Предназначена для того, чтобы при отсутствии базы с пользователями создавался пользователь администратор(admin)

**2.7 Функция int load()**

Предназначена для загрузки базы пользователей и рейсов

**2.8 Функция void save()**

Предназначена для сохранения базы пользователей и рейсов

**2.9 Функция void outputFlights()**

Предназначена для вывода всех рейсов

**2.10 Функция void editFlight()**

Предназначена для изменения информации о рейсе

**2.11 Функция void deleteFlight()**

Предназначена для удаления рейса

**2.12 Функция void doAddUser()**

Предназначена для добавления нового пользователя

**2.13 Функция void outputUsers()**

Предназначена для вывода всех пользователей

**2.14 Функция void addFlight()**

Предназначена для добавления нового рейса

**2.15 Функция void editUsers()**

Предназначена для изменения существующего пользователя

**2.16 Функция void deleteUser()**

Предназначена для удаления существующего пользователя

**2.17 Функция void buyEconomyTicket(int \*n)**

Предназначена для покупки билета эконом класса

**2.18 Функция void buyBusinessTicket(int \*n)**

Предназначена для покупки билета бизнес класса

**2.19 Функция void buyTicket(int \*n)**

Выводит меню оставшихся билетов

**2.20 Функция void buyTicketMenu()**

Выводит начальное меню рейсов для покупки билета

**2.21 Функция void userList()**

Выводит меню для взаимодействия с базой пользователей(delete user, edit user, add user, display users, exit)

**2.22 Функция void flightList()**

Выводит меню для взаимодействия с базой рейсов(delete flight, edit flight, add flight, display flights, buy tickets, exit)

**2.23 Функция void usercapabilities()**

Выводит меню возможностей обычного пользователя(display flights, buy tickets, log out) с использованием цикла while.

**2.24 Функция void admincapabilities()**

Вывода меню возможностей админа(users, flight tickets, log out) c помощью цикла while.

**2.25 Функция int main()**

Является главной функцией, она запрашивает логин, пароль, вызывает функции проверки логина, загрузки данных о билетах.

**3. Работа программы**

**3.1 Администратор**

Программа начинается с авторизации под ролью администратора:



Рисунок 1

В случае неправильного логина или пароля выводит на экран ошибку:



Рисунок 2

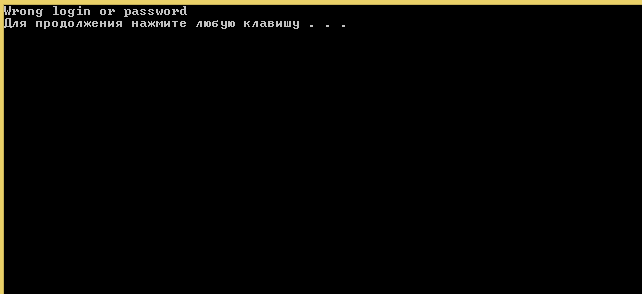


Рисунок 3

Далее выводится начальное меню администратора:



Рисунок 4

При выборе users, появляется возможность удалить, отредактировать, добавить или вывести на экран пользователя (-ей), а также вернуться обратно:



Рисунок 5

При выборе Flight tickets выводится меню с возможностью удаления, редактирования или добавления рейса, вывода всех рейсов или покупки билетов, а также возврата обратно:



Рисунок 6

Чтобы использовать эти функции, для начала нужно добавить основную информацию о рейсе:

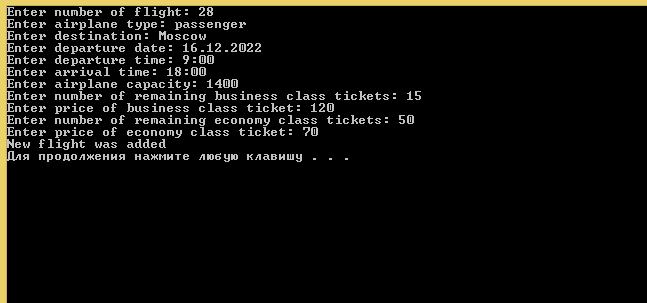


Рисунок7

После введения элементов рейса можно создать нового пользователя и перейти к нему.

3.2 Пользователь

При правильном вводе логина и пароля пользователя выводится меню:



Рисунок 8

При выборе покупки билетов можно выбрать какие и в каком количестве билеты нужны(бизнес или эконом класс):

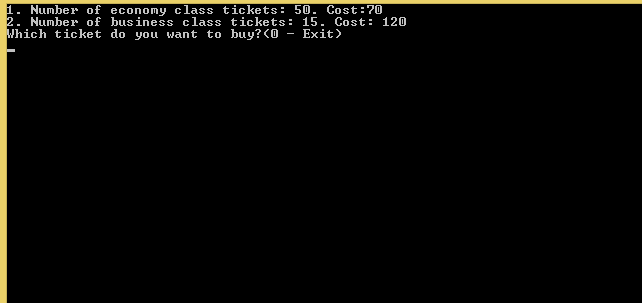


Рисунок 9

Заключение

В итоге выполнения курсовой работы был написан код для продажи авиабилетов, который реализует все необходимые функции.

Были получены теоретические знания и практические навыки по конструированию и реализации программ на языке высокого уровня в рамках процедурной парадигмы.

Список использованных источников

<https://prog-cpp.ru/c/>

<https://ru.wikipedia.org/wiki/Си_(язык_программирования)>

<https://dfe.petrsu.ru/koi/posob/c/c.htm>

<https://metanit.com/cpp/c/4.1.php>

# Приложение А

**Текст основной программы**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct ticket // Задаем структуру билета рейса

{

int numofFlight; // Номер рейса

char airplaneType[30]; // Тип самолета

char destination[30]; // Пункт назначения

int departureDay; // Дата вылета

int departureMonth; // Дата вылета

int departureYear; // Дата вылета

int departureHour; // Время вылета

int departureMinutes; // Время вылета

int arrivalHour; // Время прилета

int arrivalMinutes; // Время прилета

int airplaneCapacity; // Вместимость самолета

int businessClassTickets; // Количество оставшихся билетов бизнесс класса

int priceBusinessClassTicket; // Стоимость билета бизнес класса

int economyClassTickets; // Количество оставшихся билетов эконом класса

int priceEconomyClassTicket; // Стоимость билета эконом класса

} tickets[100];

struct user // Задаем структуру пользователей

{

char login[15];

char password[15];

int userRole;

} users[50];

int sizeOfUsers = 0, sizeOfFlights = 0, userNum = 0, userRole = 0; // Инициализируем глобальные переменные

void arrayNulling(char \*array, int size) // Зануляем массив для сравнения

{

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

{

\*(array + i) = '\000';

}

}

void reEntry(int \*stopmain) // Повторный вход в систему

{

int n, flag = 0;

while (flag == 0)

{

system("cls");

printf("1 - Log in again\n0 - Close\n");

scanf("%d", &n);

switch (n)

{

case 1:

userNum = 0;

flag = 1;

break;

case 0:

userNum = -1;

flag = 1;

\*stopmain = 1;

break;

default:

system("cls");

printf("Wrong number, please try again\n");

system("pause");

break;

}

}

}

void userVerification(char \*login, char \*password) // Проверка пользователя с базой пользователей

{

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

{

if (strcmp(users[i].login, login) == 0 && strcmp(users[i].password, password) == 0) // Функция из string.h для сравнения

{

userNum = i + 1; // Определение номера пользователя

userRole = users[i].userRole; // Определение роли пользователя

printf("The login was completed succesfully\n");

}

}

}

void logIn(int \*stopmain) // Основной вход в программу с запросом логина и пароля

{

int n, flag;

char login[15], password[15];

while (userNum == 0)

{

system("cls");

flag = 0;

userRole = 0;

arrayNulling(login, 15);

arrayNulling(password, 15);

printf("Login: ");

scanf("%s", &login);

printf("Password: ");

scanf("%s", &password);

userVerification(login, password);

if (userNum == 0)

{

system("cls");

printf("Wrong login or password\n");

system("pause");

reEntry(stopmain);

}

}

if (userNum != -1)

system("pause");

}

void defaultAdminCreate() // При отсутствии базы с пользователеми создается пользователь админ

{

sizeOfUsers++;

strcpy(users[0].login, "admin");

strcpy(users[0].password, "admin");

users[0].userRole = 1;

printf("Default admin was created. Login: admin Password: admin\n");

system("pause");

}

int load() // Загрузка базы пользователей и рейсов

{

FILE \*fTickets, \*fUsers;

fTickets = fopen("data.dat", "rb"); /\*Открытие потоков\*/

fUsers = fopen("users.dat", "r");

if (fUsers) // Условие для проверки существования файла с пользователями

{

// Чтение из файла

fread(&sizeOfUsers, sizeof(int), 1, fUsers);

if (sizeOfUsers > 50)

{

printf("The users file is damaged\n");

system("pause");

return 1;

}

}

else

defaultAdminCreate();

if (fTickets)

{

fread(&sizeOfFlights, sizeof(int), 1, fTickets);

if (sizeOfFlights > 100)

{

printf("The data is damaged\n");

system("pause");

return 1;

}

}

if (sizeOfUsers > 0)

fread(users, sizeof(struct user), sizeOfUsers, fUsers);

if (sizeOfFlights > 0)

fread(tickets, sizeof(struct ticket), sizeOfFlights, fTickets);

fclose(fTickets); // Закрытие потоков

fclose(fUsers);

return 0;

}

void save() // Сохранение базы пользователей и рейсов

{

FILE \*fTickets, \*fUsers;

fTickets = fopen("data.dat", "wb"); // Открытие потоков

fUsers = fopen("users.dat", "w");

// Запись в файл

fwrite(&sizeOfUsers, sizeof(int), 1, fUsers);

fwrite(&sizeOfFlights, sizeof(int), 1, fTickets);

if (sizeOfUsers > 0)

fwrite(users, sizeof(struct user), sizeOfUsers, fUsers);

if (sizeOfFlights > 0)

fwrite(tickets, sizeof(struct ticket), sizeOfFlights, fTickets);

fclose(fTickets); // Закрытие потоков

fclose(fUsers);

}

void outputFlights() // Вывод всех рейсов

{

system("cls");

if (sizeOfFlights > 0)

{

printf("A list of flights:\n");

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

printf("%2d.|Number of flight: %2d | Airplane type: %s | Destination: %s | Departure date: %2d.%2d.%2d | Departure time: %2d:%2d | Arrival time: %2d:%2d | Airplane capacity: %3d | Number of remaining business class tickets: %3d | Price of business class ticket: %2d | Number of remaining economy class tickets: %3d | Price of economy class ticket: %2d\n\n", i + 1, (tickets + i)->numofFlight, (tickets + i)->airplaneType, (tickets + i)->destination, (tickets + i)->departureDay, (tickets + i)->departureMonth, (tickets + i)->departureYear, (tickets + i)->departureHour, (tickets + i)->departureMinutes, (tickets + i)->arrivalHour, (tickets + i)->arrivalMinutes, (tickets + i)->airplaneCapacity, (tickets + i)->businessClassTickets, (tickets + i)->priceBusinessClassTicket, (tickets + i)->economyClassTickets, (tickets + i)->priceEconomyClassTicket);

}

else

printf("There are no tickets in the data\n");

}

void editFlight() // Изменение информации о рейсе

{

int n, flag = 0, n1, flag1;

while (flag == 0)

{

outputFlights();

if (sizeOfFlights > 0)

{

printf("Enter number of ticket (0-Exit): ");

scanf("%d", &n1);

flag1 = 0;

if ((n1 < 0) || (n1 > sizeOfFlights))

printf("Wrong number, please try again\n");

else if (n1 == 0)

flag = 1;

else

while (flag1 == 0)

{

system("cls");

printf("1 - Edit number of flight\n2 - Edit airplane type\n3 - Edit destination\n4 - Edit departure date\n5 - Edit departure time\n6 - Edit arrival time\n7 - Edit airplane capacity\n8 - Edit number of remaining business class tickets\n9 - Edit Price of business class ticket\n10 - Edit number of remaining economy class tickets\n11 - Edit price of economy class ticket\n0 - Exit\n");

scanf("%d", &n);

switch (n)

{

case 1:

printf("\nEnter new number of flight: ");

scanf("%d", &tickets[n1 - 1].numofFlight);

printf("Number of flight has been changed\n");

break;

case 2:

printf("\nEnter new airplane type: ");

scanf("%s", &tickets[n1 - 1].airplaneType);

printf("Airplane type has been changed\n");

break;

case 3:

printf("\nEnter new destination: ");

scanf("%s", &tickets[n1 - 1].destination);

printf("Destination has been changed\n");

break;

case 4:

printf("\nEnter new departure date(Example: 11.05.2023): ");

scanf("%d.%d.%d", &tickets[n1 - 1].departureDay, &tickets[n1 - 1].departureMonth, &tickets[n1 - 1].departureYear);

printf("Departure date has been changed\n");

break;

case 5:

printf("\nEnter new departure time: ");

scanf("%d:%d", &tickets[n1 - 1].departureHour, &tickets[n1 - 1].departureMinutes);

printf("Departure time has been changed\n");

break;

case 6:

printf("\nEnter new arrival time: ");

scanf("%d:%d", &tickets[n1 - 1].arrivalHour, &tickets[n1 - 1].arrivalMinutes);

printf("Arrival time has been changed\n");

break;

case 7:

printf("\nEnter new airplane capacity: ");

scanf("%d", &tickets[n1 - 1].airplaneCapacity);

printf("Airplane capacity has been changed\n");

break;

case 8:

printf("\nEnter new number of remaining business class tickets: ");

scanf("%d", &tickets[n1 - 1].businessClassTickets);

printf("Number of remaining business class tickets has been changed\n");

break;

case 9:

printf("\nEnter new price of business class tickets: ");

scanf("%d", &tickets[n1 - 1].priceBusinessClassTicket);

printf("Price of business class tickets has been changed\n");

break;

case 10:

printf("\nEnter new number of remaining economy class tickets: ");

scanf("%d", &tickets[n1 - 1].economyClassTickets);

printf("Number of remaining economy class tickets has been changed\n");

break;

case 11:

printf("\nEnter new price of economy class ticket: ");

scanf("%d", &tickets[n1 - 1].priceEconomyClassTicket);

printf("Price of economy class ticket has been changed\n");

break;

case 0:

flag1 = 1;

break;

default:

printf("Wrong number, please try again\n");

break;

}

}

}

else

{

flag = 1;

system("pause");

}

}

}

void deleteFlight() // Удаление рейса

{

int flag = 0, n, i;

while (flag == 0)

{

outputFlights();

if (sizeOfFlights > 0)

{

printf("Enter number of flight (0-Exit): ");

scanf("%d", &n);

if ((n < 0) || (n > sizeOfFlights))

printf("Wrong number, please try again\n");

else if (n == 0)

flag = 1;

else

{

// Смещение всех рейсов в массиве

for (i = n - 1; i < sizeOfFlights - 1; i++)

{

tickets[i].numofFlight = tickets[i + 1].numofFlight;

strcpy(tickets[i].airplaneType, tickets[i + 1].airplaneType);

strcpy(tickets[i].destination, tickets[i + 1].destination);

tickets[i].departureDay = tickets[i + 1].departureDay;

tickets[i].departureMonth = tickets[i + 1].departureMonth;

tickets[i].departureYear = tickets[i + 1].departureYear;

tickets[i].departureHour = tickets[i + 1].departureHour;

tickets[i].arrivalHour = tickets[i + 1].arrivalHour;

tickets[i].arrivalMinutes = tickets[i + 1].arrivalMinutes;

tickets[i].airplaneCapacity = tickets[i + 1].airplaneCapacity;

tickets[i].businessClassTickets = tickets[i + 1].businessClassTickets;

tickets[i].priceBusinessClassTicket = tickets[i + 1].priceBusinessClassTicket;

tickets[i].economyClassTickets = tickets[i + 1].economyClassTickets;

tickets[i].priceEconomyClassTicket = tickets[i + 1].priceEconomyClassTicket;

}

sizeOfFlights--;

printf("Delete flight successfully\n");

}

}

else

{

flag = 1;

system("pause");

}

}

}

void doAddUser() // Добавление нового пользователя

{

int i, flag = 0;

while (flag == 0)

{

system("cls");

flag = 1;

printf("Enter Login: ");

scanf("%s", &((users + sizeOfUsers)->login));

for (i = 0; i < sizeOfUsers; i++)

{

if (strcmp((users + sizeOfUsers)->login, (users + i)->login) == 0)

{

printf("This login already exists\n");

flag = 0;

system("pause");

break;

}

}

}

if (flag == 1)

{

printf("Enter Password: ");

scanf("%s", &((users + sizeOfUsers)->password));

while (flag == 1)

{

printf("Enter Role(1-admin or 2-user): ");

scanf("%d", &((users + sizeOfUsers)->userRole));

if ((users + sizeOfUsers)->userRole == 1 || (users + sizeOfUsers)->userRole == 2)

{

printf("New user was added\n");

sizeOfUsers++;

flag = 0;

system("pause");

}

else

{

printf("Wrong number of role, please try again\n");

system("pause");

}

}

}

}

void outputUsers() // Вывод всех пользователей

{

system("cls");

printf("A list of users:\n");

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

printf("%d. Login: %s Password: %s Role: %d\n", i + 1, (users + i)->login, (users + i)->password, (users + i)->userRole);

}

void addFlight() // Добавление нового рейса

{

system("cls");

printf("Enter number of flight: ");

scanf("%d", &(tickets + sizeOfFlights)->numofFlight);

printf("Enter airplane type: ");

scanf("%s", &(tickets + sizeOfFlights)->airplaneType);

printf("Enter destination: ");

scanf("%s", &(tickets + sizeOfFlights)->destination);

printf("Enter departure date: ");

scanf("%d.%d.%d", &(tickets + sizeOfFlights)->departureDay, &(tickets + sizeOfFlights)->departureMonth, &(tickets + sizeOfFlights)->departureYear);

printf("Enter departure time: ");

scanf("%d:%d", &(tickets + sizeOfFlights)->departureHour, &(tickets + sizeOfFlights)->departureMinutes);

printf("Enter arrival time: ");

scanf("%d:%d", &(tickets + sizeOfFlights)->arrivalHour, &(tickets + sizeOfFlights)->arrivalMinutes);

printf("Enter airplane capacity: ");

scanf("%d", &(tickets + sizeOfFlights)->airplaneCapacity);

printf("Enter number of remaining business class tickets: ");

scanf("%d", &(tickets + sizeOfFlights)->businessClassTickets);

printf("Enter price of business class ticket: ");

scanf("%d", &(tickets + sizeOfFlights)->priceBusinessClassTicket);

printf("Enter number of remaining economy class tickets: ");

scanf("%d", &(tickets + sizeOfFlights)->economyClassTickets);

printf("Enter price of economy class ticket: ");

scanf("%d", &(tickets + sizeOfFlights)->priceEconomyClassTicket);

printf("New flight was added\n");

sizeOfFlights++;

system("pause");

}

void editUsers() // Изменение существующего пользователя

{

int flag = 0, n, flag1, n1, flag2;

while (flag == 0)

{

outputUsers();

flag1 = 0;

flag2 = 0;

printf("Enter number of user(0-Exit): ");

scanf("%d", &n);

if ((n < 0) || (n > sizeOfUsers))

printf("Wrong number, please try again\n");

else if (n == 0)

flag = 1;

else

while (flag1 == 0)

{

printf("1 - Edit login\n2 - Edit password\n3 - Edit role\n0 - Exit\n");

scanf("%d", &n1);

switch (n1)

{

case 1:

system("cls");

printf("Enter new login: ");

scanf("%s", &(users + n - 1)->login);

printf("Login has been changed\n");

system("pause");

break;

case 2:

system("cls");

printf("Enter new password: ");

scanf("%s", &(users + n - 1)->password);

printf("Password has been changed\n");

system("pause");

break;

case 3:

while (flag2 == 0)

{

system("cls");

printf("Enter new role(1-admin or 2-user): ");

scanf("%d", &(users + n - 1)->userRole);

if ((users + n - 1)->userRole == 1 || (users + n - 1)->userRole == 2)

flag2 = 1;

else

printf("Wrong number, please try again\n");

}

printf("Role has been changed\n");

system("pause");

break;

case 0:

flag1 = 1;

break;

default:

system("cls");

printf("Wrong number, please try again\n");

system("pause");

break;

}

}

}

}

void deleteUser() // Удаление существующего пользователя

{

int flag = 0, n;

while (flag == 0)

{

system("cls");

outputUsers();

printf("Enter number of user(0-Exit): ");

scanf("%d", &n);

if ((n < 0) || (n > sizeOfUsers))

printf("Wrong number, please try again\n");

else if (n == 0)

flag = 1;

else if (n == userNum)

{

printf("You can't delete an active user\n");

system("pause");

}

else

{

for (int i = n - 1; i < sizeOfUsers - 1; i++)

{

// Смещение всех пользователей в массиве

strcpy((users + i)->login, (users + i + 1)->login);

strcpy((users + i)->password, (users + i + 1)->password);

(users + i)->userRole = (users + i + 1)->userRole;

}

sizeOfUsers--;

if (userNum > n)

userNum--;

printf("User was deleted\n");

system("pause");

}

}

}

void buyEconomyTicket(int \*n) // Покупка билета эконом класса

{

int flag = 0, n1;

while (flag == 0)

{

system("cls");

printf("How many economy class tickets do you want to buy?(0 - Exit)\n");

scanf("%d", &n1);

if (n1 == 0)

flag = 1;

else if (n1 < 0 || n1 > tickets[\*n - 1].economyClassTickets)

printf("Wrong number, please try again\n");

else

{

tickets[\*n - 1].economyClassTickets = tickets[\*n - 1].economyClassTickets - n1;

printf("You bought %d tickets\n", n1);

flag=1;

system("pause");

}

}

}

void buyBusinessTicket(int \*n) // Покупка билета бизнес класса

{

int flag = 0, n1;

while (flag == 0)

{

system("cls");

printf("How many economy class tickets do you want to buy?(0 - Exit)\n");

scanf("%d", &n1);

if (n1 == 0)

flag = 1;

else if (n1 < 0 || n1 > tickets[\*n - 1].businessClassTickets)

printf("Wrong number, please try again\n");

else

{

tickets[\*n - 1].businessClassTickets = tickets[\*n - 1].businessClassTickets - n1;

printf("You bought %d tickets\n", n1);

flag=1;

system("pause");

}

}

}

void buyTicket(int \*n) // меню с выводом оставшихся билетов

{

int flag = 0, n1;

while (flag == 0)

{

system("cls");

if (tickets[\*n - 1].economyClassTickets > 0 && tickets[\*n - 1].businessClassTickets > 0)

{

printf("1. Number of economy class tickets: %d. Cost:%d\n2. Number of business class tickets: %d. Cost: %d\n", tickets[\*n - 1].economyClassTickets, tickets[\*n - 1].priceEconomyClassTicket, tickets[\*n - 1].businessClassTickets, tickets[\*n - 1].priceBusinessClassTicket);

printf("Which ticket do you want to buy?(0 - Exit)\n");

scanf("%d", &n1);

if (n1 == 1)

buyEconomyTicket(n);

else if (n1 == 2)

buyBusinessTicket(n);

else if (n1 == 0)

flag = 1;

}

else if (tickets[\*n - 1].economyClassTickets > 0 && tickets[\*n - 1].businessClassTickets == 0)

{

printf("Only economy class tickets left: %d. Cost:%d\n", tickets[\*n - 1].economyClassTickets, tickets[\*n - 1].priceEconomyClassTicket);

printf("1 - Buy\n0 - Exit\n");

scanf("%d", &n1);

if (n1 == 1)

buyEconomyTicket(n);

else if (n1 == 0)

flag = 1;

}

else if (tickets[\*n - 1].economyClassTickets == 0 && tickets[\*n - 1].businessClassTickets > 0)

{

printf("Only business class tickets left: %d. Cost:%d\n", tickets[\*n - 1].businessClassTickets, tickets[\*n - 1].priceBusinessClassTicket);

printf("1 - Buy\n0 - Exit\n");

scanf("%d", &n1);

if (n1 == 1)

buyBusinessTicket(n);

else if (n1 == 0)

flag = 1;

}

}

}

void buyTicketMenu() // Начальное меню с выводом рейсов для покупки билета

{

int flag = 0, n;

while (flag == 0)

{

outputFlights();

if (sizeOfFlights > 0)

{

printf("Enter number of ticket (0-Exit): ");

scanf("%d", &n);

if ((n < 0) || (n > sizeOfFlights))

printf("Wrong number, please try again\n");

else if (n == 0)

flag = 1;

else

buyTicket(&n);

}

else

{

flag = 1;

system("pause");

}

}

}

void userList() // Меню для взаимодействия с базой пользователей

{

int flag = 0, n;

while (flag == 0)

{

system("cls");

printf("1 - Delete user\n2 - Edit user\n3 - Add user\n4 - Display users\n0 - Exit\n");

scanf("%d", &n);

switch (n)

{

case 1:

deleteUser();

break;

case 2:

editUsers();

break;

case 3:

doAddUser();

break;

case 4:

outputUsers();

system("pause");

break;

case 0:

flag = 1;

break;

default:

printf("Wrong number, please try again\n");

system("pause");

break;

}

}

}

void flightList() // Меню для взаимодействия с базой рейсов

{

int flag = 0, n;

while (flag == 0)

{

system("cls");

printf("1 - Delete flight\n2 - Edit flight\n3 - Add flight\n4 - Display flights\n5 - Buy Tickets\n0 - Exit\n");

scanf("%d", &n);

switch (n)

{

case 1:

deleteFlight();

break;

case 2:

editFlight();

break;

case 3:

addFlight();

break;

case 4:

outputFlights();

system("pause");

break;

case 5:

buyTicketMenu();

break;

case 0:

flag = 1;

break;

default:

printf("Wrong number, please try again\n");

system("pause");

break;

}

}

}

void usercapabilities() // Меню возможностей обычного пользователя

{

int flag = 0, n;

while (flag == 0)

{

system("cls");

printf("1 - Display flights\n2 - Buy tickets\n0 - Log out\n");

scanf("%d", &n);

switch (n)

{

case 1:

outputFlights();

system("pause");

break;

case 2:

buyTicketMenu();

break;

case 0:

flag = 1;

userNum = 0;

break;

default:

printf("Wrong number, please try again\n");

break;

}

}

}

void admincapabilities() // Меню возможностей админа

{

int flag = 0, n;

while (flag == 0)

{

system("cls");

printf("1 - Users\n2 - Flight tickets\n0 - Log out\n");

scanf("%d", &n);

switch (n)

{

case 1:

userList();

break;

case 2:

flightList();

break;

case 0:

flag = 1;

userNum = 0;

break;

default:

printf("Wrong number, please try again\n");

system("pause");

break;

}

}

}

int main()

{

int stopmain = 0;

if (load() == 0)

{

while (stopmain == 0)

{

system("cls");

logIn(&stopmain);

if (userRole == 2)

usercapabilities();

else if (userRole == 1)

admincapabilities();

if (userNum == 0)

reEntry(&stopmain);

}

save();

}

return 0;

}