Министерство образования Республики Беларусь

Учреждение образование

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Дисциплина: Основы конструирования программ

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему

**Разработка программы расписания движения автобусов**

Выполнил:

студент гр. 910101 Марабет К.

Проверил:

Мытник Н.П

Минск 2020

**Задание:**

В справочной автовокзала хранится расписание движения автобусов. Для каждого рейсов указаны: номер рейса; тип автобуса; пункт назначения; время отправления; время прибытия на конечный пункт. Вывести информацию о рейсах, которыми можно воспользоваться для прибытия в пункт назначения раньше заданного времени.

**Требования к программной части проекта:**

1. Разработать функциональную схему задачи, а также схемы двух модулей по указанию руководителя.
2. Создать программу (консольный режим), выполнив функциональное разбиение, реализующую следующие функции работы с бинарным/текстовым файлом. Предусмотреть создание меню 1-го уровня со следующими пунктами:
3. Вход под администратором
4. Вход под пользователем
5. Выход

Учетные записи администратора и обычного пользователя должна содержать логин и пароль. Сведения об учетных записях должны храниться в отдельном файле, желательно предусмотреть механизм шифрования/дешифрования логинов/паролей.

1. Меню 2-го уровня:

* Для администратора:

1. Создание/открытие файла с данными
2. Добавление записи
3. Редактирование записи
4. Удаление записи
5. Просмотр всех данных в табличной форме
6. Различные процедуры поиска и фильтрации данных
7. Управление пользователями
8. Выход в меню 1-го уровня

* Для обычного пользователя:

1. Открытие файла с данными
2. Просмотр всех данных в табличной форме
3. Различные процедуры поиска и фильтрации данных
4. Выход в меню 1-го уровня
5. Данные должны быть организованы в структуре (struct), не менее 7-ми полей, обязательно предусмотреть использование вложенных структур.
6. Предусмотреть обработку исключительных ситуаций
7. Построение программного кода должно соответствовать правилам, определенным в документе «Code Conventions» для соответствующего языка (технологии).
8. Предусмотреть использование различных встроенных функций для работы со строками
9. **Каждая записка, как и программная часть - должна быть уникальной, т.к. подразумевает работу над индивидуальным заданием. Обратите, пожалуйста, на это внимание во избежание ошибок.**
10. Для зачета необходимо иметь рабочую программу на диске.

ПРИЛОЖЕНИЕ А

Header.h

#pragma once

void adminAuthorization();

void userAuthorization();

void adminMenu();

void userMenu();

void userManagement();

int saveStruct(struct Trips\*, int);

int saveStruct(struct Admins\*, int);

int saveStruct(struct Users\*, int);

int loadStruct(struct Trips\*);

int loadStruct(struct Admins\*);

int loadStruct(struct Users\*);

Trips\* addStruct(Trips\*, const int);

Admins\* addStruct(Admins\*, const int);

Users\* addStruct(Users\*, const int);

template <typename T>

void createFile(const char\*, T\*);

Trips\* changeData(Trips\*);

Users\* changeData(Users\*);

Trips\* deleteData(Trips\*);

Admins\* deleteData(Admins\*);

Users\* deleteData(Users\*);

Trips\* signUp(Trips\*, int&);

Admins\* signUp(Admins\*, int&);

Users\* signUp(Users\*, int&);

bool signIn(Admins\*);

bool signIn(Users\*);

void findTripNumber();

void findBusType();

void findEarlierTrip();

void printSomeTrip(int);

ПРИЛОЖЕНИЕ Б

Function.cpp

#include <stdio.h>

#include <iostream>

#include <locale.h>

#include <windows.h>

#include "Header.h"

struct Trips

{

int numberBusTrip;

struct typeBus

{

char colour[20];

char size[20];

} type;

char destination[20];

char departureTime[20];

char arrivalTime[20];

};

struct Admins

{

char login[20];

char password[20];

};

struct Users

{

char login[20];

char password[20];

};

const char\* tripFile = "trip.txt";

const char\* adminFile = "admins.txt";

const char\* userFile = "users.txt";

struct Trips\* trip = 0;

int tripAmount = 0;

struct Admins\* admin = 0;

int adminAmount = 0;

struct Users\* user = 0;

int userAmount = 0;

void adminAuthorization()

{

int choice;

while (true)

{

std::cout <<

"1. Создать файл администраторов" << std::endl <<

"2. Регистрация" << std::endl <<

"3. Вход" << std::endl <<

"4. Удалить администратора" << std::endl <<

"5. Просмотреть файл администраторов" << std::endl <<

"6. Выход" << std::endl;

std::cin >> choice;

system("CLS");

switch (choice)

{

case 1:

createFile(adminFile, admin);

break;

case 2:

admin = signUp(admin, adminAmount);

break;

case 3:

if (signIn(admin))

adminMenu();

else

std::cout << "Неправильный логин или пароль" << std::endl;

break;

case 4:

admin = deleteData(admin);

break;

case 5:

loadStruct(admin);

break;

default:

return;

}

}

}

void userAuthorization()

{

int choice;

while (true)

{

std::cout <<

"1. Регистрация" << std::endl <<

"2. Войти" << std::endl <<

"3. Выход" << std::endl;

std::cin >> choice;

system("CLS");

switch (choice)

{

case 1:

user = signUp(user, userAmount);

break;

case 2:

if (signIn(user))

userMenu();

else

std::cout << "Неправильный логин или пароль" << std::endl;

break;

default:

return;

}

}

}

void userManagement()

{

int choice;

while (true)

{

std::cout <<

"1. Создать файл пользователей" << std::endl <<

"2. Регистрация пользователя" << std::endl <<

"3. Изменить данные пользователя" << std::endl <<

"4. Просмотреть файл пользователей" << std::endl <<

"5. Удалить пользователя" << std::endl <<

"6. Выход" << std::endl;

std::cin >> choice;

system("CLS");

switch (choice)

{

case 1:

createFile(userFile, user);

break;

case 2:

user = signUp(user, userAmount);

break;

case 3:

user = changeData(user);

break;

case 4:

loadStruct(user);

break;

case 5:

user = deleteData(user);

break;

default:

return;

}

}

}

void userMenu()

{

while (true)

{

std::cout <<

"1. Просмотреть все автобусные рейсы" << std::endl <<

"2. Искать рейсы по типу автобуса" << std::endl <<

"3. Вывести информацию о рейсах, которыми можно воспользоваться" << std::endl <<

" для прибытия в пункт назначения раньше заданного времени" << std::endl <<

"4. Выход" << std::endl;

int choice;

std::cin >> choice;

system("CLS");

switch (choice)

{

case 1:

loadStruct(trip);

break;

case 2:

findBusType();

break;

case 3:

findEarlierTrip();

break;

default:

return;

}

}

}

void adminMenu()

{

int choice;

while (true)

{

std::cout <<

"1. Создать файл с автобусными рейсами" << std::endl <<

"2. Добавить новый рейс" << std::endl <<

"3. Изменить рейс" << std::endl <<

"4. Удалить рейс" << std::endl <<

"5. Просмотреть все автобусные рейсы в файле " << std::endl <<

"6. Управление пользователями" << std::endl <<

"7. Искать рейсы по номеру" << std::endl <<

"8. Выход" << std::endl;

std::cin >> choice;

system("CLS");

switch (choice)

{

case 1:

createFile(tripFile, trip);

break;

case 2:

trip = signUp(trip, tripAmount);

break;

case 3:

trip = changeData(trip);

break;

case 4:

trip = deleteData(trip);

break;

case 5:

loadStruct(trip);

break;

case 6:

userManagement();

break;

case 7:

findTripNumber();

break;

default:

return;

}

}

}

template <typename T>

void createFile(const char\* filename, T\* someStruct)

{

if (FILE\* T = fopen(filename, "w+"))

std::cout << "Файл успешно создан" << std::endl;

else

std::cout << "Ошибка создания файла" << std::endl;

}

int saveStruct(struct Trips\* trip, int n)

{

FILE\* fileTemp;

char\* c;

// число записываемых байтов

int size = n \* sizeof(struct Trips);

if ((fileTemp = fopen(tripFile, "wb")) == NULL)

{

perror("Error occured while opening file");

return 1;

}

// записываем количество структур

c = (char\*)&n;

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

{

putc(\*c++, fileTemp);

}

// посимвольно записываем в файл все структуры

c = (char\*)trip;

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

{

putc(\*c, fileTemp);

c++;

}

fclose(fileTemp);

return 0;

}

int saveStruct(struct Admins\* admin, int n)

{

FILE\* fileTemp;

char\* c;

// число записываемых байтов

int size = n \* sizeof(struct Admins);

if ((fileTemp = fopen(adminFile, "wb")) == NULL)

{

perror("Error occured while opening file");

return 1;

}

// записываем количество структур

c = (char\*)&n;

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

{

putc(\*c++, fileTemp);

}

// посимвольно записываем в файл все структуры

c = (char\*)admin;

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

{

putc(\*c, fileTemp);

c++;

}

fclose(fileTemp);

return 0;

}

int saveStruct(struct Users\* user, int n)

{

FILE\* fileTemp;

char\* c;

// число записываемых байтов

int size = n \* sizeof(struct Users);

if ((fileTemp = fopen(userFile, "wb")) == NULL)

{

perror("Error occured while opening file");

return 1;

}

// записываем количество структур

c = (char\*)&n;

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

{

putc(\*c++, fileTemp);

}

// посимвольно записываем в файл все структуры

c = (char\*)user;

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

{

putc(\*c, fileTemp);

c++;

}

fclose(fileTemp);

return 0;

}

int loadStruct(struct Trips\* trip)

{

FILE\* fileTemp;

char\* c;

int m = sizeof(int);

int n, i;

// выделяем память для количества данных

int\* pti = (int\*)malloc(m);

if ((fileTemp = fopen(tripFile, "r")) == NULL)

{

perror("Error occured while opening file");

return 1;

}

// считываем количество структур

c = (char\*)pti;

while (m > 0)

{

i = getc(fileTemp);

if (i == EOF) break;

\*c = i;

c++;

m--;

}

//получаем число элементов

n = \*pti;

// выделяем память для считанного массива структур

struct Trips\* ptr = (struct Trips\*)malloc(n \* sizeof(struct Trips));

c = (char\*)ptr;

// после записи считываем посимвольно из файла

while ((i = getc(fileTemp)) != EOF)

{

\*c = i;

c++;

}

// перебор загруженных элементов и вывод на консоль

std::cout << "Количество автобусных рейсов: " << n << std::endl;

for (int k = 0; k < n; k++)

{

std::cout << std::endl <<

k + 1 << " Номер рейса: " << (ptr + k)->numberBusTrip << std::endl <<

k + 1 << " Тип автобуса (цвет): " << (ptr + k)->type.colour << std::endl <<

k + 1 << " Тип автобуса (размер): " << (ptr + k)->type.size << std::endl <<

k + 1 << " Путь назначения: " << (ptr + k)->destination << std::endl <<

k + 1 << " Время отправления: " << (ptr + k)->departureTime << std::endl <<

k + 1 << " Время прибытия: " << (ptr + k)->arrivalTime << std::endl;

}

std::cout << std::endl;

free(pti);

free(ptr);

fclose(fileTemp);

return 0;

}

int loadStruct(struct Admins\* admin)

{

FILE\* fileTemp;

char\* c;

int m = sizeof(int);

int n, i;

// выделяем память для количества данных

int\* pti = (int\*)malloc(m);

if ((fileTemp = fopen(adminFile, "r")) == NULL)

{

perror("Error occured while opening file");

return 1;

}

// считываем количество структур

c = (char\*)pti;

while (m > 0)

{

i = getc(fileTemp);

if (i == EOF) break;

\*c = i;

c++;

m--;

}

//получаем число элементов

n = \*pti;

// выделяем память для считанного массива структур

struct Admins\* ptr = (struct Admins\*)malloc(n \* sizeof(struct Admins));

c = (char\*)ptr;

// после записи считываем посимвольно из файла

while ((i = getc(fileTemp)) != EOF)

{

\*c = i;

c++;

}

// перебор загруженных элементов и вывод на консоль

std::cout << "Количество администраторов: " << n << std::endl;

for (int k = 0; k < n; k++)

{

std::cout << std::endl <<

k + 1 << " Логин: " << (ptr + k)->login << std::endl <<

k + 1 << " Пароль: " << (ptr + k)->password << std::endl;

}

std::cout << std::endl;

free(pti);

free(ptr);

fclose(fileTemp);

return 0;

}

int loadStruct(struct Users\* user)

{

FILE\* fileTemp;

char\* c;

int m = sizeof(int);

int n, i;

// выделяем память для количества данных

int\* pti = (int\*)malloc(m);

if ((fileTemp = fopen(userFile, "r")) == NULL)

{

perror("Error occured while opening file");

return 1;

}

// считываем количество структур

c = (char\*)pti;

while (m > 0)

{

i = getc(fileTemp);

if (i == EOF) break;

\*c = i;

c++;

m--;

}

//получаем число элементов

n = \*pti;

// выделяем память для считанного массива структур

struct Users\* ptr = (struct Users\*)malloc(n \* sizeof(struct Users));

c = (char\*)ptr;

// после записи считываем посимвольно из файла

while ((i = getc(fileTemp)) != EOF)

{

\*c = i;

c++;

}

// перебор загруженных элементов и вывод на консоль

std::cout << "Количество пользователей: " << n << std::endl;

for (int k = 0; k < n; k++)

{

std::cout << std::endl <<

k + 1 << " Логин: " << (ptr + k)->login << std::endl <<

k + 1 << " Пароль: " << (ptr + k)->password << std::endl;

}

std::cout << std::endl;

free(pti);

free(ptr);

fclose(fileTemp);

return 0;

}

Trips\* addStruct(Trips\* trip, const int amount)

{

if (amount == 0)

{

trip = new Trips[amount + 1];

}

else

{

Trips\* temp = new Trips[amount + 1];

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

{

temp[i] = trip[i];

}

delete[] trip;

trip = temp;

}

return trip;

}

Admins\* addStruct(Admins\* admin, const int index)

{

if (index == 0)

{

admin = new Admins[index + 1];

}

else

{

Admins\* temp = new Admins[index + 1];

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

{

temp[i] = admin[i];

}

delete[] admin;

admin = temp;

}

return admin;

}

Users\* addStruct(Users\* user, const int index)

{

if (index == 0)

{

user = new Users[index + 1];

}

else

{

Users\* temp = new Users[index + 1];

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

{

temp[i] = user[i];

}

delete[] user;

user = temp;

}

return user;

}

Admins\* signUp(Admins\* admin, int& adminAmount)

{

admin = addStruct(admin, adminAmount);

std::cout << "Новый логин: " << std::endl;

std::cin >> admin[adminAmount].login;

std::cout << "Новый пароль: " << std::endl;

std::cin >> admin[adminAmount].password;

adminAmount++;

saveStruct(admin, adminAmount);

return admin;

}

Users\* signUp(Users\* user, int& userAmount)

{

user = addStruct(user, userAmount);

std::cout << "Новый логин: " << std::endl;

std::cin >> user[userAmount].login;

std::cout << "Новый пароль: " << std::endl;

std::cin >> user[userAmount].password;

userAmount++;

saveStruct(user, userAmount);

return user;

}

Trips\* signUp(Trips\* trip, int& tripAmount)

{

trip = addStruct(trip, tripAmount);

std::cout << "Введите номер рейса" << std::endl;

std::cin >> trip[tripAmount].numberBusTrip;

std::cout << "Введите тип автобуса (цвет)" << std::endl;

std::cin >> trip[tripAmount].type.colour;

std::cout << "Введите тип автобуса (размер)" << std::endl;

std::cin >> trip[tripAmount].type.size;

std::cout << "Введите пункт назначения" << std::endl;

std::cin >> trip[tripAmount].destination;

std::cout << "Введите время отправления" << std::endl;

std::cin >> trip[tripAmount].departureTime;

std::cout << "Введите время прибытия" << std::endl;

std::cin >> trip[tripAmount].arrivalTime;

tripAmount++;

saveStruct(trip, tripAmount);

return trip;

}

bool signIn(Admins\* admin)

{

char login[20];

char password[20];

std::cout << "Логин: " << std::endl;

std::cin >> login;

std::cout << "Пароль: " << std::endl;

std::cin >> password;

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

{

if (!strcmp(login, admin[i].login) && !strcmp(password, admin[i].password))

{

std::cout << "Вы успешно вошли под администратора" << std::endl;

return true;

}

}

return false;

}

bool signIn(Users\* user)

{

char login[20];

char password[20];

std::cout << "Логин: " << std::endl;

std::cin >> login;

std::cout << "Пароль: " << std::endl;

std::cin >> password;

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

{

if (!strcmp(login, user[i].login) && !strcmp(password, user[i].password))

{

std::cout << "Вы успешно вошли под пользователя" << std::endl;

return true;

}

}

return false;

}

Trips\* changeData(Trips\* trip)

{

int i;

std::cout << "Выберите номер рейса. Всего рейсов: " << tripAmount << std::endl;

std::cin >> i;

if (i < 1 || i > tripAmount)

{

std::cout << "Таких рейсов не существует" << std::endl;

return trip;

}

i--;

int whatChange;

std::cout <<

"Выберите, что хотите изменить в рейсе: " << std::endl <<

"1. Номер рейса" << std::endl <<

"2. Тип автобуса" << std::endl <<

"3. Пункт назначения" << std::endl <<

"4. Время отправления" << std::endl <<

"5. Время прибытия" << std::endl <<

"6. Выйти в меню администратора" << std::endl;

std::cin >> whatChange;

system("CLS");

switch (whatChange)

{

case 1:

std::cout << "Введите номер рейса" << std::endl;

std::cin >> trip[i].numberBusTrip;

break;

case 2:

std::cout << "Введите тип автобуса (цвет)" << std::endl;

std::cin >> trip[i].type.colour;

std::cout << "Введите тип автобуса (размер)" << std::endl;

std::cin >> trip[i].type.size;

break;

case 3:

std::cout << "Введите пункт назначения" << std::endl;

std::cin >> trip[i].destination;

break;

case 4:

std::cout << "Введите время отправления" << std::endl;

std::cin >> trip[i].departureTime;

break;

case 5:

std::cout << "Введите время прибытия" << std::endl;

std::cin >> trip[i].arrivalTime;

break;

default:

break;

}

saveStruct(trip, tripAmount);

return trip;

}

Users\* changeData(Users\* user)

{

int i;

std::cout << "Выберите номер пользователя. Всего пользователей: " << userAmount << std::endl;

std::cin >> i;

if (i < 1 || i > userAmount)

{

std::cout << "Таких пользователей не существует" << std::endl;

return user;

}

i--;

int whatChange;

std::cout <<

"Выберите, что хотите изменить у пользователя: " << std::endl <<

"1. Логин" << std::endl <<

"2. Пароль" << std::endl <<

"3. Выйти" << std::endl;

std::cin >> whatChange;

system("CLS");

switch (whatChange)

{

case 1:

std::cout << "Введите новый логин: " << std::endl;

std::cin >> user[i].login;

break;

case 2:

std::cout << "Введите новый пароль: " << std::endl;

std::cin >> user[i].password;

break;

default:

break;

}

saveStruct(user, userAmount);

return user;

}

Trips\* deleteData(struct Trips\* trip)

{

int number;

std::cout << "Выберите номер рейса для удаления. Всего рейсов: " << tripAmount << std::endl;

std::cin >> number;

if (number < 1 || number > tripAmount)

{

std::cout << "Таких рейсов не существует" << std::endl;

return trip;

}

for (int i = number - 1; i < tripAmount - 1; i++)

{

trip[i] = trip[i + 1];

}

tripAmount--;

saveStruct(trip, tripAmount);

return trip;

}

Users\* deleteData(struct Users\* user)

{

int number;

std::cout << "Выберите номер пользователя для удаления. Всего пользователей: " << userAmount << std::endl;

std::cin >> number;

if (number < 1 || number > userAmount)

{

std::cout << "Таких пользователей не существует" << std::endl;

return user;

}

for (int i = number - 1; i < userAmount - 1; i++)

{

user[i] = user[i + 1];

}

userAmount--;

saveStruct(user, userAmount);

return user;

}

Admins\* deleteData(struct Admins\* admin)

{

int number;

std::cout << "Выберите номер администратора для удаления. Всего администраторов: " << adminAmount << std::endl;

std::cin >> number;

if (number < 1 || number > adminAmount)

{

std::cout << "Таких администраторов не существует" << std::endl;

return admin;

}

for (int i = number - 1; i < adminAmount - 1; i++)

{

admin[i] = admin[i + 1];

}

adminAmount--;

saveStruct(admin, adminAmount);

return admin;

}

void findTripNumber()

{

int number;

std::cout << "Введите номер рейса: " << std::endl;

std::cin >> number;

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

{

if (number == trip[i].numberBusTrip)

{

printSomeTrip(i);

}

}

}

void findBusType()

{

char colour[20], size[20];

std::cout << "Введите тип автобуса (цвет): " << std::endl;

std::cin >> colour;

std::cout << "Введите тип автобуса (размер): " << std::endl;

std::cin >> size;

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

{

if (!strcmp(colour, trip[i].type.colour) && !strcmp(size, trip[i].type.size))

{

printSomeTrip(i);

}

}

}

void findEarlierTrip()

{

char city[20], time[20];

std::cout << "Введите пункт назначения: " << std::endl;

std::cin >> city;

std::cout << "Введите время отправления (в формате 00:00): " << std::endl;

std::cin >> time;

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

{

if (!strcmp(city, trip[i].destination) && strcmp(time, trip[i].departureTime) > 0)

{

printSomeTrip(i);

}

}

}

void printSomeTrip(int i)

{

if (i < 0 || i > tripAmount) return;

std::cout <<

i + 1 << " Номер рейса: " << trip[i].numberBusTrip << std::endl <<

i + 1 << " Тип автобуса (цвет): " << trip[i].type.colour << std::endl <<

i + 1 << " Тип автобуса (размер): " << trip[i].type.size << std::endl <<

i + 1 << " Путь назначения: " << trip[i].destination << std::endl <<

i + 1 << " Время отправления: " << trip[i].departureTime << std::endl <<

i + 1 << " Время прибытия: " << trip[i].arrivalTime << std::endl;

}

ПРИЛОЖЕНИЕ В

Main.cpp

#include <iostream>

#include <locale.h>

#include <windows.h>

#include "Header.h"

int main()

{

setlocale(LC\_ALL, "Russian");

int choice;

while (true)

{

system("CLS");

std::cout <<

"1. Вход под администратором" << std::endl <<

"2. Вход под пользователем" << std::endl <<

"3. Выход" << std::endl;

std::cin >> choice;

system("CLS");

switch (choice)

{

case 1:

adminAuthorization();

break;

case 2:

userAuthorization();

break;

default:

return 0;

}

}

}