# МИНОБРНАУКИ РОССИИ

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра ЭТПТ**

**КУРСОВАЯ РАБОТА**

**по дисциплине «Информационные технологии»**

# Тема: База данных вагонов метро

Студент гр.3406 Чалых П.А.

|  |  |  |
| --- | --- | --- |
| Преподаватель |  | Чмиленко Ф.В |

Санкт-Петербург

2024

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 3 |
| 1. | Описание модулей программы | 4 |
| 1.1. | Файл «Menu.cpp» и «Fileworker.h» | 4 |
| 1.2. | Файл «edit.h» | 4 |
| 2. | Описание функций программы | 5 |
| 2.1. | Функция «int main()» | 5 |
| 2.2.  2.3.  2.4  2.5 | Функция «CreateFile()»  Функция «DeleteFile()»  Функция «OpenFile()»  Функция «Fileread(char\* fin)» | 5  5  5  5 |
|  | Заключение  Список использованных источников | 6  7 |
|  | Приложение А. Код программы | 8 |
|  | Приложение Б. Интерфейс программы | 28 |

**ВВЕДЕНИЕ**

Цель работы – получение навыков модульной разработки на примере создания программы для работы с базой данных.

Задание:

Создать программу, которая работает с базой данных. Код программы должен поддерживать модульный принцип разработки (состоять не менее чем из трех модулей). При создании программы необходимо использовать функции, массивы, структуры, указатели, выполнять форматное преобразование данных, чтение и запись в файлы.

СУБД (Система Управления Базой Данных) должна уметь выполнять следующие действия:

а) чтение данных из файла, имя которого задано пользователем;

б) просмотр данных на консоли в табличном виде;

в) редактирование записи, номер которой задан пользователем;

г) добавление записи в конец списка;

д) удаление записи, номер которой задан пользователем;

е) сортировку записей по возрастанию и убыванию;

ж) поиск записей по названию;

з) запись данных в файл, имя которого задано пользователем;

и) численную обработку данных по выбору учащегося (подсчет среднего,

поиск максимального и т. п.).

Необходимо предусмотреть обработку ошибочных ситуаций:

а) на диске нет файла, имя которого задано для чтения;

б) для новой записи нет места в массиве;

в) введен неправильный номер для удаляемой или редактируемой записи.

# ОПИСАНИЕ МОДУЛЕЙ ПРОГРАММЫ

**1.1.** Файл «Menu.cpp» и «Fileworker.h»

В файле «Menu.cpp» содержится функция main(), наполненная инструкциями switch(), реализующими выбор необходимых для пользователя опций, происходит задание первоначальных параметров консоли в виде установки цвета выводимой информации и подключения русского алфавита.

Файл “Menu.cpp” отвечает за вызов функций создания и удаления файла. Файл “Fileworker.h” отвечает за создание, удаление и чтение файла с вызовом функции из “edit.h” для дальнейшей работы с данными.

**1.2.** Файл «edit.h»

В данном модуле происходит работа с полученными из файла данными.

Содержит в себе:

а) необходимую структуру

б) выделение динамической памяти

в) функции сортировки

г) функции добавления и редактирования строк

д) функции поиска по номеру

е) функцию сохранения данных в файл

# ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ

**2.1. Функция «int main()»**

Данная функция позволяет пользователю создать, удалить или открыть файл для последующей работы с базой данных.

**2.2. Функция «CreateFile()»**

Эта функция используется для создания файла. Если файл уже существует, выводится сообщение об ошибке. Если же файла с таким именем ещё нет, файл создается и пользователь получает сообщение об успешном создании файла. Файл появляется в папке Debug.

**2.3. Функция «DeleteFile()»**

Эта функция используется для удаления файла. Если файл удален успешно, выводится соответствующее сообщение. В противном случае, выводится сообщение об ошибке.

**2.4. Функция «OpenFile()»**

Эта функция используется для открытия файла. Функция получает имя файла от пользователя, добавляет к нему расширение “.txt” и вызывает функцию FileRead() для чтения содержимого этого файла.

**2.5. Функция «Fileread(char\* fin)»**

Самая большая и основная функция программы. В ней ведётся работа с полученными из файла данными. Она считывает данные с файла и заносит их в массив структур.

Позволяет добавлять, редактировать или удалять “строки”, сортировать номера по возрастанию и убыванию, искать “строки” по введённому типу вагона.

**ЗАКЛЮЧЕНИЕ**

Выполнив курсовую работу удалось создать программу для работы с базой данных. Темой курсовой стала база данных вагонов метро.

Были освоены навыки работы с основными функциями языков “C” и “С++”.

Получившаяся программа работает по следующему алгоритму:

При запуске программа считывает данные из файла filename.txt.

Данные из файла преобразуются в массив структур cp, где каждая структура cp хранит информацию об одном вагоне (модель, дата постройки, статус, номер).

Далее, пользователю представлен набор действий, которые он может выполнять с базой данных.

Разработанная программа включает в себя функционал, позволяющий:

динамически выделять память для хранения базы данных, проверять корректность вводимых данных, выводить базу данных в табличном виде на экран, записывать данные и выполнять другие операции с ними, включая сортировку и поиск.

**список использованных источников**

1. Основы программирования на языке С++ сост / Чмиленко Ф.В.,

Бондарь А.С., Хоршев А.А

Ч74 Основы программирования на языке С++: учеб.-метод. Пособие СПб.:

Изд-во СПбГЭТУ “ЛЭТИ”, 2021, 40с2.

2. https://ru.stackoverflow.com

3. https://vec.etu.ru

**ПРИЛОЖЕНИЕ А**

**КОД ПРОГРАММЫ**

**Main.cpp**

#include <iostream>

#include <windows.h>

#include <edit.h>

#include <Fileworker.h>

using namespace std;

int main()

{

int mn;

system("chcp 1251");

system("color F0");

system("cls");//чистим экран

while(true){

while (true){

system("cls");//чистим экран

cout << "Выберите действие:" << endl; //выбор действия

cout << "1 - Открыть файл" << endl;

cout << "2 - Создать файл" << endl;

cout << "3 - Удалить файл" << endl;

cout << "0 - Выход из программы" << endl;

cin >> mn;

switch(mn){ // выбор действия

case 0: cout << "Ну тогда катись отсюда" << endl; Sleep(1500); break;

case 1: OpenFile(); break;

case 2: CreateFile(); break;

case 3: DeleteFile(); break;

default: cout << "Ошибка" << endl; Sleep(1500); break;

}

}

}

return 0;

}

**Fileworker.h**

#include <iostream>

#include <windows.h>

#include <fstream>

#include <cstdio>

#include <sstream>

using namespace std;

char filename[50];

char text[] = ".txt";

char\* ourchar;

bool WriteFlag;

bool NeedWrite;

void CreateFile();

void DeleteFile();

void OpenFile();

void CreateFile(){

FILE\* tst;

cout << "Введите имя файла: " << endl;

cin >> filename;

strcat(filename, ".txt"); // Добавляем расширение файла

//tst = fopen(filename, "r");

tst = fopen(filename, "r");

if (tst != NULL) {

cout << "Файл с таким названием уже существует!" << endl;

Sleep(1000);

fclose(tst);

return;

}

fclose(tst);

ofstream File(filename);

File.close();

cout << "Файл создан! Проверьте папку Debug." << endl;

Sleep(1000);

}

void DeleteFile(){

char name[256];

cout << "Введите имя файла: " << endl;

cin >> name;

strcat(name, ".txt"); // Добавляем расширение .txt

if (remove(name) != 0) { // Удаление файла

cout << "Ошибка удаления файла" << endl;

} else {

cout << "Файл успешно удален" << endl;

}

Sleep(1000);

}

void OpenFile(){

cout << "" << endl;

cout << "Введите название файла без формата(.txt)" << endl;

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

cin >> filename; // вводим название файла

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

strcat(filename, text); // Добавляем текст к названию файла

FileRead(filename);

Sleep(1000);

}

**edit.h**

#include <iostream>

#include <stdio.h>

#include <sstream>

#include <iomanip>

#include <cstring>

#include <string.h>

#include <fstream>

#include <cstdlib>

#include <cstdio>

struct WAG{

char vagon[10];

char dat[10];

char status[50];

int nomer;

};

using namespace std;

void FileRead(char\* fin){

bool needSave;

bool nmenu;

FILE \*file;

int lineCount = 0;

file = fopen(fin, "r");

if (file == NULL) {

cout << "Ошибка чтения" << endl; Sleep(1000);

}

char buffer[256];

while (fgets(buffer, sizeof(buffer), file) != NULL) {

lineCount++;

}

WAG\* cp = (WAG\*)malloc(lineCount\*sizeof(struct WAG));

if (cp == NULL) {

cout << "Ошибка выделения памяти" << endl; Sleep(1000);

fclose(file);

}

rewind(file); //Возвращаем указатель в начало файла

// Читаем данные в массив структур

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

fscanf(file, "%\*[\n]%[^,], %[^,], %[^,], %d", cp[i].vagon, cp[i].dat, cp[i].status, &cp[i].nomer);

}

fclose(file); // Закрываем файл

while(true){

while(true){

char mec;

system("cls"); // чистим экран

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

cout << "|№ | Модель вагона | Дата постройки | Статус | Номер |" << endl;

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

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

cout << "|" << setw(2) << left << i+1 << " | "

<< setw(20) << left << cp[i].vagon << " | "

<< setw(20) << left << cp[i].dat << " | "

<< setw(15) << left << cp[i].status << " | "

<< setw(15) << left << cp[i].nomer << " | "

<< endl;

}

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

cout << "1 - Редактировать строку" << endl;

cout << "2 - Добавить строку" << endl;

cout << "3 - Удалить строку" << endl;

cout << "4 - Записать в файл" << endl;

cout << "5 - Поиск" << endl;

cout << "6 - Сортировка" << endl;

cout << "7 - Выход в меню" << endl;

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

cin >> mec;

if(mec == '1'){

int editIndex;

cout << "Введите номер строки для редактирования: ";

cin >> editIndex;

cin.ignore();

if (editIndex > 0 && editIndex <= lineCount) {

char rp;

//редактируем

editIndex--; // Индексы массива начинаются с 0

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

cout << "Где будем редактировать?" << endl;

cout << "1 - Тип вагона" << endl;

cout << "2 - Дата постройки" << endl;

cout << "3 - Статус" << endl;

cout << "4 - Номер" << endl;

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

cin >> rp;

if(rp == '1'){

cout << "Тип вагона: "; cin.ignore(); cin.getline(cp[editIndex].vagon, 50);

} else if(rp == '2'){

cout << "Дата: ";

\_SYSTEMTIME s;

GetLocalTime(&s);

int y;

cin >> y;

if(s.wYear >= y){

snprintf(cp[editIndex].dat, sizeof(cp[editIndex].dat), "%d", y);

}

else{

cout << "Ошибка!" << endl;

Sleep(1500);

}

} else if (rp == '3'){

cout << "" << endl;

cout << "Статус:" << endl;

cout << "1. Эксплуатируется" << endl;

cout << "2. Списан" << endl;

cout << "3. Консервация" << endl;

cout << "4. КВР" << endl;

char stat;

cin >> stat;

if (stat == '1') {

strcpy(cp[lineCount - 1].status, "Эксплуатируется");

} else if(stat == '2'){

strcpy(cp[lineCount - 1].status, "Списан");

} else if(stat == '3'){

strcpy(cp[lineCount - 1].status, "Консервация");

} else if (stat == '4'){

strcpy(cp[lineCount - 1].status, "КВР");

}

cout << "" << endl;

} else if (rp == '4'){

int ex;

cout << "Номер: "; cin >> ex;

bool errorflag = false;

for (int o = 0; o < lineCount; o++) {

if (o != editIndex && ex == cp[o].nomer) {

errorflag = true;

break; // Выход из цикла, если дубликат найден

}

}

if (errorflag) {

cout << "Ошибка!" << endl;

Sleep(1500);

}

else{

cp[editIndex].nomer = ex;

}

}

else{

cout << "Ошибка!" << endl; Sleep(1000);

}

} else {

cout << "Некорректный номер строки." << endl;

cin.clear();

cin.ignore();

cout << "Нажмите Enter для продолжения" << endl;

cin.get();

}

}else if(mec == '2'){

WAG newData;

// +1 к массиву

lineCount++;

cp = (WAG\*)realloc(cp, lineCount \* sizeof(WAG));

// и добавляем эту структуру в конец массива

cout << "Тип вагона: ";

cin.ignore(numeric\_limits<streamsize>::max(), '\n'); // чистим буфер

cin.getline(cp[lineCount - 1].vagon, 50);

cout << "Год постройки: ";

\_SYSTEMTIME p;

GetLocalTime(&p);

int yr;

cin >> yr;

if(p.wYear >= yr){

snprintf(cp[lineCount - 1].dat, sizeof(cp[lineCount - 1].dat), "%d", yr);

}

else{

cout << "Ошибка!" << endl;

lineCount--;

Sleep(1500);

break;

}

cout << "" << endl;

cout << "Статус:" << endl;

cout << "1. Эксплуатируется" << endl;

cout << "2. Списан" << endl;

cout << "3. Консервация" << endl;

cout << "4. КВР" << endl;

char stat;

cin >> stat;

if (stat == '1') {

strcpy(cp[lineCount - 1].status, "Эксплуатируется");

} else if(stat == '2'){

strcpy(cp[lineCount - 1].status, "Списан");

} else if(stat == '3'){

strcpy(cp[lineCount - 1].status, "Консервация");

} else if (stat == '4'){

strcpy(cp[lineCount - 1].status, "КВР");

}

cout << "" << endl;

cout << "Номер: ";

int in;

cin >> in;

bool ef = false;

for (int p = 0; p < lineCount; p++) {

if (p != lineCount && in == cp[p].nomer) {

ef = true;

break; // Выход из цикла, если дубликат найден

}

}

if (ef) {

cout << "Ошибка!" << endl;

lineCount--;

Sleep(1500);

}

else{

cp[lineCount - 1].nomer = in;

}

}else if(mec == '3'){

int deleteIndex;

cout << "Введите номер строки для удаления: ";

cin >> deleteIndex;

if (deleteIndex > 0 && deleteIndex <= lineCount) {

// удаление строки

deleteIndex--; // !индексы массива начинаются с 0!

for (int i = deleteIndex; i < lineCount - 1; i++) {

cp[i] = cp[i + 1];

}

lineCount--;

cout << "Строка удалена." << endl;

Sleep(750);

} else {

cout << "Некорректный номер строки." << endl;

Sleep(750);

}

}else if(mec == '4'){

char full[10];

char filenameInput[50];

char fullFilename[50];

cout << "Введите название файла для сохранения данных: ";

cin >> full;

strcat(full, ".txt"); // добавляем .txt

ofstream outfile(full);

if (outfile.is\_open()) {

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

outfile << cp[i].vagon << ","

<< cp[i].dat << ","

<< cp[i].status << ","

<< cp[i].nomer << endl;

}

outfile.close();

cout << "Данные записаны в файл '" << full << "'." << endl;

nmenu = true;

free(cp);

break;

Sleep(750);

cin.clear();

cin.ignore();

cout << "Нажмите Enter для продолжения" << endl;

cin.get();

} else {

cout << "Ошибка открытия файла." << endl;

Sleep(750);

}

needSave = true;

}else if(mec == '5'){

char po[50];

bool cflag;

cout << "Введите параметр поиска" << endl;

cin >> po;

for(int u = 0; u < lineCount; u++){

if(strcmp(po,cp[u].vagon) == 0){

cflag = true;

}

}

if(cflag == true){

system("cls"); // чистим экран

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

cout << "|№ | Модель вагона | Дата постройки | Статус | Номер |" << endl;

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

for(int u = 0; u < lineCount; u++){

if(strcmp(po,cp[u].vagon) == 0){

cout << "|" << setw(2) << left << u+1 << " | "

<< setw(20) << left << cp[u].vagon << " | "

<< setw(20) << left << cp[u].dat << " | "

<< setw(15) << left << cp[u].status << " | "

<< setw(15) << left << cp[u].nomer << " |"

<< endl;

}

}

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

getchar();

}

}else if(mec == '6'){

int so;

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

cout << "По возрастанию или убыванию? " << endl;

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

cout << "1 - По возрастанию" << endl;

cout << "2 - По убыванию" << endl;

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

cin >> so;

if(so == 1){

//сортировка

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

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

if (cp[i].nomer < cp[j].nomer)

{

WAG t = cp[j];

cp[j] = cp[i];

cp[i] = t;

}

}

}

}else if(so == 2){

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

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

if (cp[i].nomer > cp[j].nomer)

{

WAG t = cp[j];

cp[j] = cp[i];

cp[i] = t;

}

}

}

}else{

cout <<"no"<<endl;

Sleep(1000);

}

}else if(mec == '7'){

if(needSave == false){

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

cout << "Данные не сохранены. Может, стоит их сохранить?" << endl;

cout << "1 - Да, надо сохранить" << endl;

cout << "2 - Нет, давай в меню" << endl;

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

char nmc;

cin >> nmc;

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

if (nmc == '1') {

}else{

free(cp); //исправление

cout << "Катапультируемся в меню" << endl;

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

nmenu = true;

Sleep(1000);

break;

}

}

}else{

cout << "Ошибка" << endl; Sleep(1500);

}

}

}

}

# ПРИЛОЖЕНИЕ Б

# ИНТЕРФЕЙС ПРОГРАММЫ

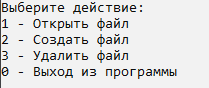


Рисунок 1 – Главное меню программы

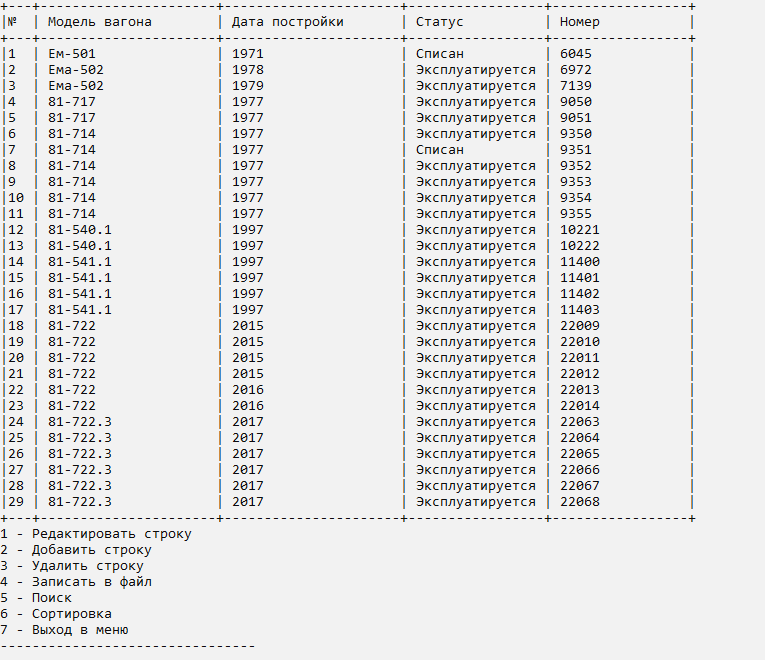


Рисунок 2 – Отображаемая база данных и меню для работы с ней

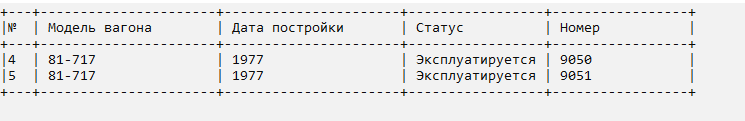


Рисунок 3 – Вывод данных по искомой модели вагона

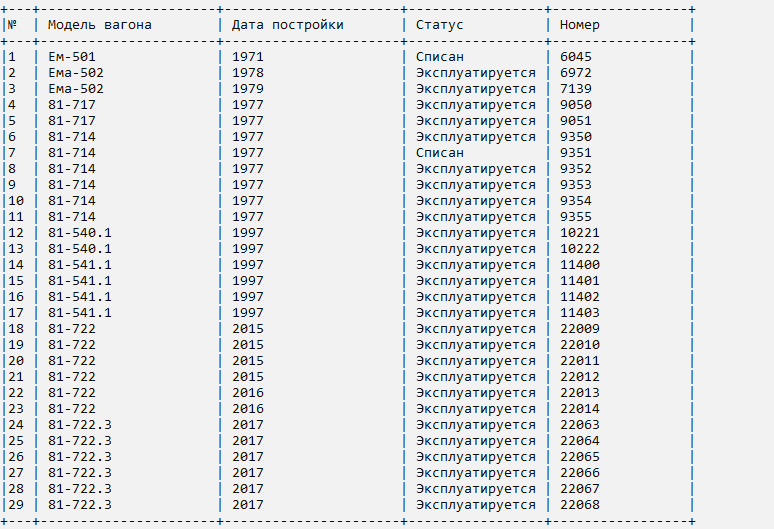


Рисунок 4 – Отображаемая база данных после сортировки по возрастанию номера вагона