//File.h

#pragma once

#include "shablon.h"

#include "fstream"

template <class A>

class text\_files

{

private:

ofstream file1;

ifstream file2;

};

//Exeption.h

#pragma once

#include "shablon.h"

class Exeption

{

private:

string err;

int m;

public:

Exeption(const Exeption& temp);

Exeption(int num, const string str);

~Exeption();

void show();

};

string chek\_str();

int chek\_num();

//Exeption.cpp

#include "Exeptoin.h"

Exeption::Exeption(const Exeption& temp)

{

m = temp.m;

err = temp.err;

}

Exeption::Exeption(int num,const string str)

{

m = num;

err = str;

}

Exeption::~Exeption()

{

}

void Exeption::show() //вывод ошибки на экран

{

cout << err << endl;

}

string chek\_str()

{

string t\_str;

bool flag;

do

{

try

{

cin >> t\_str;

flag = 1;

for (int i = 0; t\_str[i]; i++)

{

if (!(t\_str[i] >=65 && t\_str[i]<=90)&&

(!(t\_str[i] >= 97 && t\_str[i] <= 122))&&

(!(t\_str[i] >= 192 && t\_str[i] <= 255))&&

((!(t\_str[i] >= 65 && t\_str[i] <= 90) &&

(!(t\_str[i] >= 97 && t\_str[i] <= 122))&& (!(t\_str[i+1] >= 192 && t\_str[i+1] <= 255))))

)

{

throw Exeption(1, "Используйте буквы латинского или русского алфавита при вводе");

}

}

}

catch (Exeption& obj)

{

flag = 0;

obj.show();

}

} while (!flag);

return t\_str;

}

int chek\_num()

{

bool flag;

int t\_num = 0;

do

{

fflush(stdin);

try

{

flag = 1;

if(!(cin >> t\_num)||t\_num <= 0 || t\_num> 999999999)

{

rewind(stdin);

cin.clear();

throw Exeption(2, "Введите число больше 0 и меньше 999999999");

}

}

catch (Exeption& obj)

{

flag = 0;

obj.show();

}

} while (!flag);

return t\_num;

}

//shablon.h

#pragma once

#include <iostream>

#include <string>

#include <conio.h>

#include <iomanip>

#include <Windows.h>

#include"salon.h"

#include"microavtobus.h"

#include"electrobus.h"

#include"BY.h"

#include "File.h"

using namespace std;

template <class A>

struct cohered //шаблонная структура для создания элементов

{ // очереди для каждой категории

A object;

ochered\* p\_right;

ochered\* p\_left;

};

template <class A>

class shablon //шаблонный класс для создания контейнеров для каждой категории

{

private:

int size;

ochered<A>\* p\_first = nullptr;

ochered<A>\* p\_last = nullptr;

public:

shablon();

~shablon();

void getObj(A t);

ochered<A>\* getFirst();

void add(A p);

void del();

void nastr();

void out();

bool poisk(A peremennaia);

void writeF();

void readF();

void del\_all();

};

//shablon.cpp

#include "shablon.h"

template<class A>

shablon<A>::shablon()

{

p\_first = nullptr;

p\_last = nullptr;

size = 0;

}

template<class A>

shablon<A>::~shablon()

{}

template<class A>

ochered<A>\* shablon<A>:: getFirst()

{

return p\_first;

}

template<class A>

void shablon<A>:: getObj(A t)

{

cout << endl << t;

}

template<class A>

void shablon<A>:: add(A p) //добавление элемента в очередь

{

ochered<A>\* car = new ochered<A>;

car->object = p;

++size;

if (p\_first == nullptr || p\_last == nullptr)

{

car->p\_right = nullptr;

car->p\_left = nullptr;

p\_first = car;

p\_last = car;

}

else

{

car->p\_right = p\_first;

car->p\_left = nullptr;

p\_first->p\_left = car;

p\_first = car;

}

}

template<class A>

bool shablon<A>::poisk(A peremennaia)//метод поиска элемента по параметрам в контейнере

{

ochered<A>\* obj = p\_first;

peremennaia.shapka();

while (obj != NULL) //пока не конец очереди

{

if (obj->object == peremennaia)

{

cout << obj->object << endl;

}

obj = obj->p\_right;

}

system("pause");

return false;

}

template<class A>

void shablon<A>::out() //вывод элементов очереди на экран в виде таблицы

{

ochered<A>\* t;

t = this->p\_first;

t->object.shapka();

while (t != nullptr)

{

getObj(t->object);

t = t->p\_right;

}

cout << endl;

}

template<class A>

void shablon<A>:: del() //метод удаления любого элемента очереди из контейнера

{

ochered<A>\* t;

t = this->p\_first;

while (t != nullptr)

{

system("cls");

t->object.shapka();

getObj(t->object);

cout << endl<<"Нажмите d - чтобы листать вправо или a - чтобы влево\n0-чтобы выйти\nc-чтобы удалить" << endl;

switch (\_getch())

{

case'd':

if (t ->p\_right == nullptr)

cout << "\nЭлемент отсутсвует" << endl;

else

t = t->p\_right;

break;

case'a':

if (t ->p\_left == nullptr)

cout << "\nЭлемент отсутсвует" << endl;

else

t = t->p\_left;

break;

case'0':

t = nullptr;

break;

case 'c':

if (t == p\_first)

p\_first = t->p\_right;

if (t == p\_last)

p\_last = t->p\_left;

if (t->p\_left != nullptr)

t->p\_left->p\_right = t->p\_right;

if (t->p\_right != nullptr)

t->p\_right->p\_left = t->p\_left;

delete t;

t = p\_first;

break;

default:cout << "ERROR(возможно у вас русская раскладка)";

}

}

}

template<class A>

void shablon<A>::del\_all() //удаляет все элементы очереди

{

ochered<A>\* t;

t = this->p\_first;

while (t != nullptr)

{

if (t == p\_first)

p\_first = t->p\_right;

if (t->p\_right != nullptr)

t->p\_right->p\_left = t->p\_left;

delete t;

t = p\_first;

}

}

template<class A>

void shablon<A>::nastr() //метод изменения полей объектов

{

ochered<A>\* t;

t = this->p\_first;

while (t != nullptr)

{

system("cls");

t->object.shapka();

getObj(t->object);

cout << endl<<"Нажмите d - чтобы листать вправо или a - чтобы влево\n0-чтобы выйти\ns-чтобы открыть меню настройки" << endl;

switch (\_getch())

{

case'd':

if (t->p\_right == nullptr)

cout << "\nЭлемент отсутсвует" << endl;

else

t = t->p\_right;

break;

case'a':

if (t->p\_left == nullptr)

cout << "\nЭлемент отсутсвует" << endl;

else

t = t->p\_left;

break;

case'0':

t = nullptr;

break;

case 's':

t->object.setting();

break;

default:cout << "ERROR(возможно у вас русская раскладка)";

}

}

}

template <class A>

void shablon<A>::writeF() //запись в файл

{

ofstream f1(typeid(A).name());

if (!f1.is\_open())

{

cout << "файл не открыт\n";

}

else

{

system("cls");

ochered<A>\* temp;

temp = getFirst();

while (temp)

{

f1 << temp->object << endl;

temp = temp->p\_right;

}

}

f1.close();

}

template <class A>

void shablon<A>::readF() //чтение из файла

{

ifstream f2(typeid(A).name());

A obj = A();

while (f2 >> obj)

{

this->add(obj);

}

f2.close();

}

//Header.h

#pragma once

#include <iostream>

#include <string>

using namespace std;

class transrport

{

protected:

string name;

int price;

int maxspeed;

public:

transrport()

{

name = "";

price = 0;

maxspeed = 0;

}

string getName()

{

return name;

}

int getPrice()

{

return price;

}

int getMS()

{

return maxspeed;

}

};

//legkovie.h

#pragma once

#include "Header.h"

class legkovie : public transrport

{

protected:

string model;

string kyzov;

public:

legkovie()

{

model = "";

kyzov = "";

}

string getModel()

{

return model;

}

string getKyzov()

{

return kyzov;

}

};

//avtobusi.h

#pragma once

#include "Header.h"

class avtobusi : public transrport

{

protected:

int KolMest;

string toplivo;

public:

avtobusi()

{

KolMest = 0;

toplivo = "";

}

int getKM()

{

return KolMest;

}

string getToplivo()

{

return toplivo;

}

};

//salon.h

#pragma once

#include "legkovie.h"

#include <Windows.h>

#include <conio.h>

#include <iomanip>

#include <fstream>

class salon : public legkovie

{

protected:

string garant;

string test;

public:

salon();

string getGarant();

string getTest();

friend istream& operator >> (istream& in, salon& obj);

friend ostream& operator << (ostream& out, salon& obj);

friend ofstream& operator << (ofstream& out, salon& obj);

friend ifstream& operator >> (ifstream& in, salon& obj);

void setting();

void shapka();

bool operator==(salon& obj);

};

//salon.cpp

#include "salon.h"

#include "Exeptoin.h"

salon::salon()

{

name = "";

garant = "";

test = "";

}

string salon:: getGarant()

{

return garant;

}

string salon::getTest()

{

return test;

}

void salon::shapka()

{

cout << setiosflags(ios::left)

<< setw(14) << "Название"

<< setw(14) << "Модель"

<< setw(13) << "Кузов"

<< setw(15) << "Макс скорость"

<< setw(15) << "Гарантия"

<< setw(15) << "Тест-драйв"

<< setw(12) << "Цена"

<< endl;

}

ifstream& operator >> (ifstream& in, salon& obj)

{

in >> obj.name >> obj.price >> obj.maxspeed >> obj.model >> obj.kyzov >> obj.garant >> obj.test;

return in;

}

ofstream& operator << (ofstream& out, salon& obj)

{

out << obj.name << ' ' << obj.price << ' ' << obj.maxspeed << ' ' << obj.model << ' ' << obj.kyzov << ' ' << obj.garant << ' ' << obj.test << endl;

return out;

}

istream& operator >> (istream& in, salon& obj)

{

cout << "введите новое имя - ";

obj.name = chek\_str();

cout << "введите новую цену - ";

obj.price = chek\_num();

cout << "введите новую макс скорость - ";

obj.maxspeed= chek\_num();

cout << "введите новое название модели - ";

cin>>obj.model;

cout << "введите новый тип кузова - ";

obj.kyzov = chek\_str();

cout << "есть ли гарантия? - ";

obj.garant= chek\_str();

cout << "есть ли тест-драйв? - ";

obj.test= chek\_str();

return in;

}

ostream& operator << (ostream& out, salon& obj)

{

return out <<

endl << setw(14) << obj.getName()

<< setw(14) << obj.getModel()

<< setw(13) << obj.getKyzov()

<< setw(15) << obj.getMS()

<< setw(15) << obj.getGarant()

<< setw(15) << obj.getTest()

<< setw(12) << obj.getPrice();

}

bool salon:: operator==(salon& obj)

{

if (

(obj.name == this->name || obj.name == "-") &&

(obj.model == this->model || obj.model == "-") &&

(obj.kyzov == this->kyzov || obj.kyzov == "-") &&

(obj.maxspeed == this->maxspeed || obj.maxspeed == 1) &&

(obj.price == this->price || obj.price == 1) &&

(obj.garant == this->garant || obj.garant == "-") &&

(obj.test == this->test || obj.test == "-")

)

return true;

else

return false;

}

void salon::setting()

{

cout << "1)Название\n2)Цена\n3)Макс скорость\n4)Модель\n5)Кузов\n6)Гарантия\n7)Тест-драйв" << endl;

switch (\_getch())

{

case '1':

cout << getName() << endl;

cout << "введите новое имя - ";

name = chek\_str(); break;

case '2':

cout << getPrice() << endl;

cout << "введите новую цену - ";

price= chek\_num(); break;

case '3':

cout << getMS() << endl;

cout << "введите новую макс скорость - ";

maxspeed= chek\_num(); break;

case '4':

cout << getModel() << endl;

cout << "введите новое название модели - ";

cin >> model; break;

case '5':

cout << getKyzov() << endl;

cout << "введите новый тип кузова - ";

kyzov= chek\_str(); break;

case '6':

cout << getGarant() << endl;

cout << "есть ли гарантия? - ";

garant= chek\_str(); break;

case '7':

cout << getTest() << endl;

cout << "есть ли тест-драйв? - ";

test = chek\_str(); break;

default:cout << "ERROR" << endl; break;

}

}

//BY.h

#pragma once

#include "legkovie.h"

#include <Windows.h>

#include <conio.h>

#include <iomanip>

class BY : public legkovie

{

protected:

string gorod;

int probeg;

public:

BY();

string getGorod();

int getProbeg();

friend istream& operator >> (istream& in, BY& obj);

friend ostream& operator << (ostream& out, BY& obj);

friend ofstream& operator << (ofstream& out, BY& obj);

friend ifstream& operator >> (ifstream& in, BY& obj);

void setting();

void shapka();

bool operator==(BY& obj);

};

//BY.cpp

#include "BY.h"

#include"Exeptoin.h"

BY::BY()

{

name = "";

gorod = "";

probeg = 0;

}

string BY:: getGorod()

{

return gorod;

}

int BY:: getProbeg()

{

return probeg;

}

void BY::shapka()

{

cout << setiosflags(ios::left)

<< setw(14) << "Название"

<< setw(14) << "Модель"

<< setw(13) << "Кузов"

<< setw(15) << "Макс скорость"

<< setw(15) << "Город продажи"

<< setw(15) << "Пробег"

<< setw(12) << "Цена"

<< endl;

}

ifstream& operator >> (ifstream& in, BY& obj)

{

in >> obj.name >> obj.price >> obj.maxspeed >> obj.model >> obj.kyzov >> obj.gorod >> obj.probeg;

return in;

}

ofstream& operator << (ofstream& out, BY& obj)

{

out << obj.name << ' ' << obj.price << ' ' << obj.maxspeed << ' ' << obj.model << ' ' << obj.kyzov << ' ' << obj.gorod << ' ' << obj.probeg << endl;

return out;

}

istream& operator >> (istream& in, BY& obj)

{

cout << "введите новое имя - ";

obj.name= chek\_str();

cout << "введите новую цену - ";

obj.price= chek\_num();

cout << "введите новую макс скорость - ";

obj.maxspeed= chek\_num();

cout << "введите новое название модели - ";

obj.model= chek\_str();

cout << "введите новый тип кузова - ";

obj.kyzov= chek\_str();

cout << "введите город продажи - ";

obj.gorod= chek\_str();

cout << "введите пробег - ";

obj.probeg= chek\_num();

return in;

}

ostream& operator << (ostream& out, BY& obj)

{

return out <<

endl << setw(14) << obj.getName()

<< setw(14) << obj.getModel()

<< setw(13) << obj.getKyzov()

<< setw(15) << obj.getMS()

<< setw(15) << obj.getGorod()

<< setw(15) << obj.getProbeg()

<< setw(12) << obj.getPrice();

}

bool BY:: operator==(BY& obj)

{

if (

(obj.name == this->name || obj.name == "-") &&

(obj.model == this->model || obj.model == "-") &&

(obj.kyzov == this->kyzov || obj.kyzov == "-") &&

(obj.maxspeed == this->maxspeed || obj.maxspeed == 1) &&

(obj.price == this->price || obj.price == 1) &&

(obj.gorod == this->gorod || obj.gorod == "-") &&

(obj.probeg == this->probeg || obj.probeg == 1)

)

return true;

else

return false;

}

void BY::setting()

{

cout << "1)Название\n2)Цена\n3)Макс скорость\n4)Модель\n5)Кузов\n6)Город продажи\n7)Пробег" << endl;

switch (\_getch())

{

case '1':

cout << getName() << endl;

cout << "введите новое имя - ";

name= chek\_str(); break;

case '2':

cout << getPrice() << endl;

cout << "введите новую цену - ";

price= chek\_num(); break;

case '3':

cout << getMS() << endl;

cout << "введите новую макс скорость - ";

maxspeed= chek\_num(); break;

case '4':

cout << getModel() << endl;

cout << "введите новое название модели - ";

cin>>model; break;

case '5':

cout << getKyzov() << endl;

cout << "введите новый тип кузова - ";

kyzov= chek\_str(); break;

case '6':

cout << getGorod() << endl;

cout << "введите город продажи - ";

gorod= chek\_str(); break;

case '7':

cout << getProbeg() << endl;

cout << "введите пробег - ";

probeg= chek\_num(); break;

default:cout << "ERROR" << endl; break;

}

}

//microavtobus.h

#pragma once

#include "avtobusi.h"

#include <Windows.h>

#include <conio.h>

#include <iomanip>

class microavtobus :public avtobusi

{

protected:

int rashod;

public:

microavtobus();

int getRashod();

void shapka();

friend istream& operator >> (istream& in, microavtobus& obj);

friend ostream& operator << (ostream& out, microavtobus& obj);

friend ofstream& operator << (ofstream& out, microavtobus& obj);

friend ifstream& operator >> (ifstream& in, microavtobus& obj);

void setting();

bool operator==(microavtobus& obj);

};

//microavtobus.cpp

#include "microavtobus.h"

#include "Exeptoin.h"

microavtobus::microavtobus()

{

name = "no\_name";

rashod = 0;

}

int microavtobus::getRashod()

{

return rashod;

}

void microavtobus::shapka()

{

cout << setiosflags(ios::left)

<< setw(10) << "Название"

<< setw(15) << "Макс скорость"

<< setw(17) << "Кол-во мест"

<< setw(10) << "Топливо"

<< setw(15) << "Расход"

<< setw(12) << "Цена"

<< endl;

}

ifstream& operator >> (ifstream& in, microavtobus& obj)

{

in >> obj.name >> obj.price >> obj.maxspeed >> obj.KolMest >> obj.toplivo >> obj.rashod >> obj.price;

return in;

}

ofstream& operator << (ofstream& out, microavtobus& obj)

{

out << obj.name << ' ' << obj.price << ' ' << obj.maxspeed << ' ' << obj.KolMest << ' ' << obj.toplivo << ' ' << obj.rashod << ' ' << obj.price << endl;

return out;

}

istream& operator >> (istream& in, microavtobus& obj)

{

cout << "введите новое имя - ";

obj.name= chek\_str();

cout << "введите новую макс скорость - ";

obj.maxspeed= chek\_num();

cout << "введите новое кол-во мест - ";

obj.KolMest= chek\_num();

cout << "введите вид топливо - ";

obj.toplivo= chek\_str();

cout << "введите новый расход - ";

obj.rashod= chek\_num();

cout << "введите новую цену - ";

obj.price = chek\_num();

return in;

}

ostream& operator << (ostream& out, microavtobus& obj)

{

return out <<

endl << setw(10) << obj.getName()

<< setw(15) << obj.getMS()

<< setw(17) << obj.getKM()

<< setw(10) << obj.getToplivo()

<< setw(15) << obj.getRashod()

<< setw(12) << obj.getPrice();

}

bool microavtobus::operator==(microavtobus& obj)

{

if (

(obj.name == this->name || obj.name == "-") &&

(obj.KolMest == this->KolMest || obj.KolMest == 1) &&

(obj.rashod == this->rashod || obj.rashod == 1) &&

(obj.maxspeed == this->maxspeed || obj.maxspeed == 1) &&

(obj.price == this->price || obj.price == 1) &&

(obj.toplivo == this->toplivo || obj.toplivo == "-")

)

return true;

else

return false;

}

void microavtobus::setting()

{

cout << "1)Название\n2)Цена\n3)Макс скорость\n4)Кол-во мест\n5)Вид топлива\n6)Расход" << endl;

switch (\_getch())

{

case '1':

cout << getName() << endl;

cout << "введите новое имя - ";

name= chek\_str(); break;

case '2':

cout << getPrice() << endl;

cout << "введите новую цену - ";

price= chek\_num(); break;

case '3':

cout << getMS() << endl;

cout << "введите новую макс скорость - ";

maxspeed= chek\_num(); break;

case '4':

cout << getKM() << endl;

cout << "введите новое кол-во мест - ";

KolMest= chek\_num(); break;

case '5':

cout << getToplivo() << endl;

cout << "введите вид топлива - ";

toplivo= chek\_num(); break;

case '6':

cout << getRashod() << endl;

cout << "введите новый расход - ";

rashod= chek\_num(); break;

default:cout << "ERROR" << endl; break;

}

}

//electrobus.h

#pragma once

#include "avtobusi.h"

#include <Windows.h>

#include <conio.h>

#include <iomanip>

class electrobus :public avtobusi

{

protected:

int PotrebEnerg;

public:

electrobus();

int getPE();

void shapka();

friend istream& operator >> (istream& in, electrobus& obj);

friend ostream& operator << (ostream& out, electrobus& obj);

friend ofstream& operator << (ofstream& out, electrobus& obj);

friend ifstream& operator >> (ifstream& in, electrobus& obj);

void setting();

bool operator==(electrobus& obj);

};

//electrobus.cpp

#include "electrobus.h"

#include "Exeptoin.h"

electrobus::electrobus()

{

name = "no\_name";

PotrebEnerg = 0;

}

int electrobus::getPE()

{

return PotrebEnerg;

}

void electrobus:: shapka()

{

cout << setiosflags(ios::left)

<< setw(10) << "Название"

<< setw(15) << "Макс скорость"

<< setw(17) << "Кол-во мест"

<< setw(10) << "Доп.топливо"

<< setw(15) << "Потреб.Энерг."

<< setw(12) << "Цена"

<< endl;

}

ifstream& operator >> (ifstream& in, electrobus& obj)

{

in >> obj.name >> obj.price >> obj.maxspeed >> obj.KolMest >> obj.toplivo >> obj.PotrebEnerg >> obj.price;

return in;

}

ofstream& operator << (ofstream& out, electrobus& obj)

{

out << obj.name << ' ' << obj.price << ' ' << obj.maxspeed << ' ' << obj.KolMest << ' ' << obj.toplivo << ' ' << obj.PotrebEnerg << ' ' << obj.price << endl;

return out;

}

istream& operator >> (istream& in, electrobus& obj)

{

cout << "введите новое имя - ";

obj.name= chek\_str();

cout << "введите новую макс скорость - ";

obj.maxspeed= chek\_num();

cout << "введите новое кол-во мест - ";

obj.KolMest= chek\_num();

cout << "введите вид топлива - ";

obj.toplivo= chek\_str();

cout << "введите новое потреб.энерг. - ";

obj.PotrebEnerg= chek\_num();

cout << "введите новую цену - ";

obj.price= chek\_num();

return in;

}

ostream& operator << (ostream& out, electrobus& obj)

{

return out <<

endl << setw(10) << obj.getName()

<< setw(15) << obj.getMS()

<< setw(17) << obj.getKM()

<< setw(10) << obj.getToplivo()

<< setw(15) << obj.getPE()

<< setw(12) << obj.getPrice();

}

bool electrobus::operator==(electrobus& obj)

{

if (

(obj.name == this->name || obj.name == "-") &&

(obj.KolMest == this->KolMest || obj.KolMest == 1) &&

(obj.PotrebEnerg == this->PotrebEnerg || obj.PotrebEnerg == 1) &&

(obj.maxspeed == this->maxspeed || obj.maxspeed == 1) &&

(obj.price == this->price || obj.price == 1) &&

(obj.toplivo == this->toplivo || obj.toplivo == "-")

)

return true;

else

return false;

}

void electrobus:: setting()

{

cout << "1)Название\n2)Цена\n3)Макс скорость\n4)Кол-во мест\n5)Вид топлива\n6)Расход" << endl;

switch (\_getch()){

case '1':

cout << getName() << endl;

cout << "введите новое имя - ";

name= chek\_str(); break;

case '2':

cout << getPrice() << endl;

cout << "введите новую цену - ";

price= chek\_num(); break;

case '3':

cout << getMS() << endl;

cout << "введите новую макс скорость - ";

maxspeed= chek\_num(); break;

case '4':

cout << getKM() << endl;

cout << "введите новое кол-во мест - ";

KolMest= chek\_num(); break;

case '5':

cout << getToplivo() << endl;

cout << "введите вид топлива - ";

toplivo= chek\_str(); break;

case '6':

cout << getPE() << endl;

cout << "введите новое потреб.энерг. - ";

PotrebEnerg= chek\_num(); break;

default:cout << "ERROR" << endl; break;

}

}

//Source.cpp

#include "shablon.h"

#include "shablon.cpp"

#include"Header.h"

#include"legkovie.h"

#include"avtobusi.h"

#include"salon.h"

#include"microavtobus.h"

#include"electrobus.h"

#include"BY.h"

using namespace std;

template<class A>

void menu(shablon <A>& obj, text\_files<A>&t\_file,int adm) //шаблонный метод меню

{

obj.readF();

int u = 1;

while (u == 1)

{

system("cls");

char i = 'y';

A temp\_obj;

if(adm==1)

cout << "1)Просмотреть\n2)Поиск\n3)В главное меню\n4)Добавить\n5)Изменить\n6)Удалить\n7)Очистить файл\n8)Отменить посл. действие" << endl;

else

cout << "1)Просмотреть\n2)Поиск\n3)В главное меню" << endl;

switch (\_getch())

{

case '1':

system("cls");

obj.out();

system("pause");

break;

case '2':

system("cls");

cout << "Для пропуска строковых полей введите '-'\nчисловых - '1'" << endl;

cin >> temp\_obj;

obj.poisk(temp\_obj);

break;

case '3':

system("cls");

obj.writeF();

obj.del\_all();

u = 0;

break;

case '4':

if (adm == 1)

{

system("cls");

while (i == 'y')

{

cin >> temp\_obj;

obj.add(temp\_obj);

cout << "добавить еще?(y/n)" << endl;

i = \_getch();

}

}

break;

case '5':

if (adm == 1)

{

system("cls");

obj.nastr();

}

break;

case '6':

if (adm == 1)

{

system("cls");

obj.del();

}

break;

case '7':

if (adm == 1)

{

obj.del\_all();

obj.writeF();

cout << "Файл очищен" << endl;

system("pause");

}

break;

case '8':

if (adm == 1)

{

obj.del\_all();

obj.readF();

cout << "Действие отмененно" << endl;

system("pause");

}

break;

}

}

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

shablon<salon> p\_s;

shablon<microavtobus>p\_a;

shablon<electrobus> p\_t;

shablon<BY> p\_y;

text\_files<salon>f\_s;

text\_files<microavtobus>f\_a;

text\_files<electrobus>f\_t;

text\_files<BY> f\_y;

ifstream admin\_;

ofstream admin\_1;

string password;

string admin="";

int adm = 0; //уровень доступа(0-пользователь/1-администратор)

while (1)

{

if(adm==0)

cout << "1)Автомобили из салона\n2)Автомобили б/у\n3)Микроавтобусы\n4)Электробусы\n5)Авторизоваться с правами администратора\n0)выход" << endl;

else

cout << "1)Автомобили из салона\n2)Автомобили б/у\n3)Микроавтобусы\n4)Электробусы\n6)Изменить пароль\n7)Вернуться в режим пользователя\n0)выход" << endl;

rewind(stdin);

switch (\_getch())

{

case'1':

menu(p\_s,f\_s,adm);

break;

case'2':

menu(p\_y,f\_y,adm);

break;

case'3':

menu(p\_a,f\_a,adm);

break;

case'4':

menu(p\_t,f\_t,adm);

break;

case'5': //авторизация от имени администратора

if (adm == 0)

{

admin\_.open("password.txt", ifstream::app);

admin\_ >> password;

cout << "Введите пароль - ";

cin >> admin;

if (admin == password) //проверка пароля

{

cout << "Доступ разрешен, добро пожаловать" << endl;

adm = 1;

}

else

{

cout << "Отказано!" << endl;

}

system("pause");

admin\_.close();

}

break;

case'0':

exit(0);

break;

case '6':

admin\_1.open("password.txt");

cout << "Введите новый пароль - ";

cin >> admin;

admin\_1 << admin; //изменение пароля в файле

cout << "Пароль изменен "<<endl;

system("pause");

admin\_1.close();

break;

case '7':

adm = 0;

break;

}

system("cls");

}

return 0;

}