

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Новосибирский государственный университет экономики и управления «НИНХ»**

**(ФГБОУ ВО «НГУЭУ», НГУЭУ)**

Кафедра информационных технологий

**ТЕКСТОВОЕ ОПИСАНИЕ ЗАДАНИЯ**

По дисциплине «Прикладные информационные технологии»

Программное приложение «Заказы похоронный дом»

Ф.И.О студента: Строгов Кирилл Дмитриевич

Направление: 02.03.02 Фундаментальная информатика и информационные технологии

Направленность (профиль): Инженерия программного обеспечения

Номер группы: ФИ-101

Номер зачетной книжки: 212098

Проверил: Ковригин Алексей Викторович

Новосибирск 2023

**Программный код, полученный в результате разработки**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

// Тема: #8 Заказы похоронный дом

// Задание:

// Создать программу по управлению объектами данных по указанной теме с использованием ООП и реализовывать изученные паттерны.

// Необходимо создать минимум 3 объекта сущности разных видов и указать взаимодействие между объектами. Реализовать интерфейсы ввод,

// вывода и редактирования сущностей. Реализовать ручной ввод сущностей, загрузку из файла и сохранения в файл, удаление выбранных сущностей.

// Данные между запусками программы хранить в структурированном текстовом файле.

// Услуга

class Service {

protected:

string name;

int cost;

public:

Service() { name = "\t"; cost = 0; }

void create(const string n, const int c) { name = n; cost = c; }

string getName() { return name; }

int getCost() { return cost; }

};

// Фабрика

class fabrikaA {

public:

Service\* create(string\* mas) {

Service\* res = new Service;

res->create(mas[0], stoi(mas[1]));

return res;

}

};

// Заказ

class Order {

protected:

fabrikaA agency;

int id = 10;

int countItems = 0;

Service\* items[100] = { 0 };

string Clientname, Number;

public:

int getID() { return id; }

int getCount() { return countItems; }

Service\*\* getItems() { return items; }

string getName() { return Clientname; }

string getNumber() { return Number; }

void create(const string n, const string num, const int count, string(\*ms)[2], const int id\_in) {

if (id\_in != 0) id = id\_in;

Clientname = n;

Number = num;

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

items[i] = agency.create(ms[i]);

countItems++;

}

}

void printOrder() {

cout << id << endl;

cout << '\t' << Clientname << endl;

cout << '\t' << Number << endl;

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

cout << "\t\t" << items[i]->getName() << " : " << items[i]->getCost() << endl;

}

}

};

class fabrikaB {

public:

Order\* create(const string n, const string tb, const int count, string(\*ms)[2], const int id) {

Order\* res = new Order;

res->create(n, tb, count, ms, id);

return res;

}

};

// Сотрудник

class employee {

private:

fabrikaB agency;

Order\* orders[100] = {};

int countOrders = 0;

public:

int getCount() { return countOrders; }

Order\*\* getOrders() { return orders; }

void printOrders() {

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

cout << '[' << i + 1 << "] ";

orders[i]->printOrder();

}

}

void newOrder(const string n, const string tb, const int count, string(\*ms)[2], const int id = 0) {

orders[countOrders] = agency.create(n, tb, count, ms, id);

countOrders++;

}

void delOrder(const int ind) {

if ((ind >= 0) && (ind < countOrders)) {

for (int i = ind; i < countOrders - 1; i++) { orders[i] = orders[i + 1]; }

countOrders--;

}

}

};

// Меню

class Goodctrl {

private:

int choice1 = 0;

int kg = 0;

employee\* Employee = new employee;

void mainMenu(void) {

system("cls");

cout << "Главное меню\n";

cout << "!!!Все названия должны быть на английском языке!!!\n\n";

cout << "[1] Показать заказы\n";

cout << "[2] Новый заказ\n";

cout << "[3] Удалить из списка\n";

cout << "[4] Сохранить в файл\n";

cout << "[5] Взять из файла\n";

cout << "[0] Выйти\n";

cout << "Ваш выбор: ";

cin >> choice1;

}

// Вывод всех заказов в консоль

void print() {

system("cls");

Employee->printOrders();

cout << "Введите 0 для продолжения:";

cin >> kg;

}

// Создание нового заказа

void newOrderMenu(void) {

string c, name, num;

string mas[100][2];

system("cls");

cout << "Введите имя клиента - ";

getline(cin >> ws, name);

cout << "Введите номер клиента - ";

getline(cin >> ws, num);

cout << "Введите колличество пунктов заказа - ";

getline(cin >> ws, c);

for (int i = 0; i < stoi(c); i++) {

cout << "Название услуги - ";

getline(cin >> ws, mas[i][0]);

cout << "Цена - ";

getline(cin >> ws, mas[i][1]);

}

Employee->newOrder(name, num, stoi(c), mas);

cout << "Выполнено\n";

cout << "Введите 0 для продолжения: ";

cin >> kg;

}

// Удаление заказа

void removeOrderMenu(void) {

string c;

system("cls");

cout << "Меню удаления заказа\n\n";

cout << "[0] Назад\n";

Employee->printOrders();

cout << "Пожалуйста, выберете заказ: ";

getline(cin >> ws, c);

Employee->delOrder(stoi(c) - 1);

system("cls");

cout << "Выполнено успешно" << endl;

cout << "Введите 0 для продолжения";

cin >> kg;

}

// Сохранение данных в файл "base.txt"

void saveToFile() {

ofstream fout;

string path = "base.txt";

fout.open(path);

if (!fout.is\_open()) {

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

}

else {

Order\*\* ord = Employee->getOrders();

fout << Employee->getCount() << '\n'; // количество клиентов

for (int i = 0; i < Employee->getCount(); i++) {

fout << ord[i]->getID() << '\n'; // номер заказа

fout << ord[i]->getName() << '\n'; // имя клиента

fout << ord[i]->getNumber() << '\n'; // номер клиента

fout << ord[i]->getCount() << '\n'; // количество пунктов в заказе

Service\*\* prod = ord[i]->getItems();

for (int j = 0; j < ord[i]->getCount(); j++) {

fout << prod[j]->getName() << endl; // название услуги

fout << prod[j]->getCost() << endl; // цена услуги

}

}

system("cls");

cout << "Выполнено успешно" << endl;

cout << "Введите 0 для продолжения";

cin >> kg;

}

fout.close();

}

// Импорт из файла "base.txt"

void takeFromFile() {

ifstream fin;

string str, id, nm, num, cnt, path = "base.txt";

string mas[100][2];

fin.open(path);

if (!fin.is\_open()) {

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

}

else {

getline(fin, str, '\n');

int ordersCount = stoi(str);

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

getline(fin, id, '\n');

getline(fin, nm, '\n');

getline(fin, num, '\n');

getline(fin, cnt, '\n');

int itemsCount = stoi(cnt);

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

getline(fin, mas[j][0], '\n');

getline(fin, mas[j][1], '\n');

}

Employee->newOrder(nm, num, stoi(cnt), mas, stoi(id));

}

system("cls");

cout << "Complete!" << endl;

cout << "Enter 0 to continue:";

cin >> kg;

}

fin.close();

}

public:

void menu() {

do {

mainMenu();

switch (choice1) {

case 1:

print();

break;

case 2:

newOrderMenu();

break;

case 3:

removeOrderMenu();

break;

case 4:

saveToFile();

break;

case 5:

takeFromFile();

break;

case 0:

break;

}

} while (choice1 != 0);

}

};

int main()

{

setlocale(LC\_ALL, "Rus");

Goodctrl ctrl;

ctrl.menu();

return 0;

}