

**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: Пантыгин Евгений Анатольевич

Направление: 02.03.02 Фундаментальная информатика и информационные технологии

Направленность (профиль): Инженерия программного обеспечения

Номер группы: ФИ-101

Номер зачетной книжки: 212095

Проверил: Ковригин Алексей Викторович

Новосибирск 2023

**Программный код, полученный в результате разработки**

Тема: 5. Заказы фирмы разработки ПО

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.

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

// Service

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; }

};

class fabrikaA

{

public:

    Service\* create(string\* mas) {

        Service\* res = new Service;

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

        return res;

    }

};

class Order {

protected:

    fabrikaA company;

    int count\_it = 0;

    Service\* items[100] = {0};

    string name, Num;

public:

    int getCount () {return count\_it;}

    string getNum () {return Num;}

    string getName () {return name;}

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

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

        name = nm;

        Num = num;

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

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

            count\_it++;

        }

    }

    void printOrder() {

        cout << '\t' << name << endl;

        cout << '\t' << Num << endl;

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

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

        }

    }

};

class fabrikaB {

public:

    Order\* create(const string nm, const string num, const int count, string(\*ms)[2]) {

        Order\* res = new Order;

        res->create(nm, num, count, ms);

        return res;

    }

};

class Programmer {

private:

    fabrikaB company;

    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 nm, const string num, const int count, string(\*ms)[2]) {

        orders[countOrders] = company.create(nm, num, count, ms);

        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--;

        }

    }

};

// Menu

class Menu {

private:

    int choice1 = 0;

    int ext = 0;

    Programmer\* programmer = new Programmer;

    void mainNavigation(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 << "Выбор: ";

        cin >> choice1;

    }

    // Вывод всех проектов в консоль

    void print() {

        system("cls");

        programmer->printOrders();

        cout << "Введите 0 для продолжения:";

        cin >> ext;

    }

    // 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 << "Срок сдачи - ";

        getline(cin >> ws, num);

        cout << "Область разработки - ";

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

        cout << "Стоимость - ";

        getline(cin >> ws, mas[0][1]);

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

        cout << "Выполнено\n";

        cout << "Введите 0 для продолжения: ";

        cin >> ext;

    }

    // Удаление заказа

    void removeOrderMenu(void) {

        string c;

        system("cls");

        cout << "Меню удаления заказа\n\n";

        cout << "[0] Назад\n";

        programmer->printOrders();

        cout << "Пожалуйста выберете заказ: ";

        getline(cin >> ws, c);

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

        system("cls");

        cout << "Выполнено успешно" << endl;

        cout << "Введите 0 для продолжения";

        cin >> ext;

    }

    // 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 = programmer->getOrders();

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

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

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

                fout << ord[i]->getNum() << '\n'; // срок сдачи

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

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

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

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

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

                }

            }

            system("cls");

            cout << "Выполнено успешно" << endl;

            cout << "Введите 0 для продолжения";

            cin >> ext;

        }

        fout.close();

    }

    // Импорт из файла "base.txt"

    void takeFromFile() {

        ifstream fin;

        string str, 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, 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');

                }

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

            }

            system("cls");

            cout << "Complete!" << endl;

            cout << "Enter 0 to continue:";

            cin >> ext;

        }

        fin.close();

    }

public:

    void menu() {

        do {

            mainNavigation();

            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, "Ru");

    Menu app;

    app.menu();

    return 0;

}