

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

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

**(ФГБОУ ВО «НГУЭУ», НГУЭУ)**

Кафедра

**ТЕКСТОВОЕ ОПИСАНИЕ ЗАДАНИЯ**

По дисциплине «Прикладные информационные технологии»

Программное приложение «Заказы похоронный дом»

Ф.И.О студента: Спиридонов Илья Максимович

Номер группы: ФИ-101

Номер зачетной книжки: 212525

Проверил:

Новосибирск

**Программный код, полученный в результате разработки**

Цель работы создание программы способной вести учет заказов фирмы разработки ПО , добавлять новые заказы, изменять уже сушествуюшие заказы. Дополнительными задачами являлись загрузку из файла и сохранения в файл, удаление выбранных сущностей, все эти поставленные задачи были реализованны в программе. Для решения поставленной задачи я воспользовался объектно-ориентированным программированим для структурирования программы. В программе сушествуют 4 сушности(класса) Product, OrderProgramm , Manipulator , Menu. Все сушестности взаимедействуют между собой , дополняя друг друга функционалом, в совокупности они и дают рабочуюю программу.

**Класс Product**

class Product {

protected:

string productname;

int cost;

public:

Product() { productname = "\t"; cost = 0; }

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

string getName() {

return productname;

}

int getCost() {

return cost;

}

};

**Класс OrderProgramm**

class OrderProgramm {

protected:

Productfabrika store;

int countItems = 0;

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

string Clientname, Timework;

public:

int getCount() {

return countItems;

}

Product\*\* getItems() {

return items;

}

string getName() {

return Clientname;

}

string getTime() {

return Timework;

}

void order\_create(const string m, const string tm, const int count, string(\*ms)[2], const int id\_in) {

Clientname = m;

Timework = tm;

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

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

countItems++;

}

}

void printOrder() {

cout << "\t" << "Имя клиента: " << Clientname << endl;

cout << "\t" << "Предпологаемое вами время разроботки: " << Timework << endl;

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

cout << "\t" << "Title заказа: " << items[i]->getTitle() << endl

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

}

}

};

**Класс Manipulator**

class Manipulator {

private:

Orderfabrika store;

OrderProgramm\* orders[100] = {};

int countOrders = 0;

public:

int getCount() { return countOrders; }

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

void printOrders() {

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

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

orders[i]->printOrder();

}

}

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

orders[countOrders] = store.create(m, tm, 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--;

}

}

};

**Класс Manu**

class Menu {

private:

int choice1 = 0;

int exit = 0;

Manipulator\* manipulator = new Manipulator;

void navigationMenu(void) {

system("cls");

cout << "Меню навигации\n";

cout << "~~~~~~~~~~~~~~~~~~~~~~\n\n";

cout << "1 - Заказы\n";

cout << "2 - Новая разработка\n";

cout << "3 - Удалить из списка\n";

cout << "4 - Сохранить данных\n";

cout << "5 - Загружка данных из файла\n";

cout << "0 - Выйти\n\n";

cout << "~~~~~~~~~~~~~~~~~~~~~~\n\n";

cout << "Выбор ";

cin >> choice1;

}

// Вывод всех разработок в консоль

void print() {

system("cls");

manipulator->printOrders();

cout << "Введите 0 для продолжения:";

cin >> exit;

}

// Создание нового заказа

void newOrderMenu(void) {

string n, name, deadlines;

string ms[100][2];

system("cls");

cout << "Имя клиента - ";

getline(cin >> ws, name);

cout << "Предполагаемые сроки выполнения работы - ";

getline(cin >> ws, deadlines);

cout << "Вид проекта - ";

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

cout << "Цена - ";

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

manipulator->newOrder(name, deadlines, stoi("1"), ms);

cout << "Выполнено\n";

cout << "Введите 0 для продолжения: ";

cin >> exit;

}

// Удаление заказа

void removeOrderMenu(void) {

string n;

system("cls");

cout << "Меню удаления заказа\n\n";

cout << "0 - Назад\n";

manipulator->printOrders();

cout << "Пожалуйста, выберете заказ: ";

getline(cin >> ws, n);

manipulator->delOrder(stoi(n) - 1);

system("cls");

cout << "Выполнено успешно" << endl;

cout << "Введите 0 для продолжения";

cin >> exit;

}

// Сохранение данных в файл "orders.txt"

void saveToFile() {

ofstream fout;

string path = "orders.txt";

fout.open(path);

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

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

}

else {

OrderProgramm\*\* ord = manipulator->getOrders();

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

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

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

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

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

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

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

fout << prod[j]->getTitle() << endl; // описание разработки

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

}

}

system("cls");

cout << "Выполнено успешно" << endl;

cout << "Введите 0 для продолжения";

cin >> exit;

}

fout.close();

}

// Импорт из файла "orders.txt"

void takeFromFile() {

ifstream fin;

string str, nm, tm, cnt, path = "orders.txt";

string ms[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, tm, '\n');

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

int itemsCount = stoi(cnt);

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

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

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

}

manipulator->newOrder(nm, tm, stoi(cnt), ms);

}

system("cls");

cout << "Complete!" << endl;

cout << "Enter 0 to continue:";

cin >> exit;

}

fin.close();

}

public:

void menu() {

do {

navigationMenu();

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);

}

};