

**МИНОБРНАУКИ РОССИИ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Новосибирский государственный университет экономики и управления «НИНХ» (ФГБОУ ВО «НГУЭУ», НГУЭУ)**

Кафедра информационных технологий

Отечет

Дисциплина: Прикладные информационные технологии

Ф.И.О. студента: Аглиулин Тимур Равилевич

Направление: 09.03.02 Информационные системы и технологии

Номер группы: ИС101

Номер зачетной книжки: 220031

Номер варианта расчетно-графической работы: 1

Проверил: Ковригин А.В.

Новосибирск 2023

СОДЕРЖАНИЕ

[1.Задача 3](#_Toc124374458)

[2.Решение 4](#_Toc124374459)

[3. Вид приложения в консоли 11](#_Toc124374460)

# 1.Задача

Создание программного приложения «Заказы в кафе»

Необходимо 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.

# 2.Решение

С самого начала я объявляю структуры данных для дальнейшей работы с ними:

struct human

{

string name;

string surname;

};

struct characteristics

{

string name;

string order;

int quantity;

double price;

};

Далее я создаю сущности для работы с ними (сущность заказов, клиентов, официантов),т.к. функционал методов у них однообразный, а различаются они лишь структурами данных, я опишу их работу на примере сущности клиентов:

private:

human cli[20] = {

{ "Тимур","Кузнецов" },

{ "Игорь","Сакулян" },

{ "Артем","Богданов" },

};

Для хранения изначальных данных я использовал модификатор доступа privat, так как эти данные будут использоваться лишь в методах этого класса.

Для методов класса я использовал модификатор доступа public,т.к. методы класса еще понадобятся для создания меню с помощью паттерна controller.

Метод вывода сущностей:

void Print() {

cout << " Клиенты кафе ""KFC"" " << endl;

cout << endl;

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

cout << setw(6) << " Имя клиента:" << setw(10) << (cli + i)->name<< setw(10);

cout << setw(6) << " Фамилия клиента:" << setw(10) << (cli + i)->surname << endl;

}

}

Метод добавления сущности:

void add() {

if (n == 20) {

cout << "Список клиентов полон!!!" << endl;

return;

}

cout << "Имя клиента:" << endl;

cin >> cli[n].name;

cout << " Фамилия клиента:" << endl;

cin >> cli[n].surname;

n++;

}

Метод удаления сущности:

void del() {

if (n == 0) {

cout << "Клиентов нет!!!" << endl;

return;

}

cout << "Имя клиента:" << endl;

string cliname;

cin >> cliname;

int icli = -1;

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

if (cliname == cli[i].name) {

icli = i;

break;

}

}

if (icli == -1) {

cout << "Такого клиента нет" << endl;

return;

}

cli[icli] = cli[n - 1];

n--;

cout << "Успешное удаление клиента!" << endl;

}

Метод редактирования сущности:

void change() {

if (n == 0) {

cout << "Клиентов нет!!!" << endl;

return;

}

cout << "Введите имя клиента" << endl;

string cliname;

cin >> cliname;

int icli = -1;

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

if (cliname == cli[i].name) {

cout << "Имя клиента:" << endl;

cin >> cli[i].name;

cout << "Фамилия клиента:" << endl;

cin >> cli[i].surname;

icli = i;

break;

}

}

if (icli == -1) {

cout << "Такого клиента нет" << endl;

return;

}

cout << "Успешное редактирование клиента!" << endl;

}

Метод записи данных сущности в файл:

void recording() {

fout.open("клиенты.txt");

fout << n << endl;

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

fout << cli[i].name << endl;

fout << cli[i].surname << endl;

}

fout.close();

}

Метод считывания данных из файла в сущность:

void readData() {

fin.open("Вставлялка.txt");

fin >> n;

fin.ignore();

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

getline(fin, cli[i].name);

getline(fin, cli[i].surname);

fin.ignore();

}

fin.close();

}

Далее я бы хотел описать класс с паттерном (и названием) controller:

class controller : public Data , public waiter,public client{

private:

bool work;

Делаю класс дочерним для предыдущих сущностей для создания общего меню.

В модификации доступа private создается логическая переменная work для работы паттерна.

public:

controller() {

work = 1;

}

bool getwork() {

return work;

}

void showMenu() {

cout << "Меню:" << endl

<< "(1) Официанты:" << endl

<< "(2) Клиенты:" << endl

<< "(3) Заказ:" << endl

<< "(0) Выход из программы:" << endl;

while (getMenuItem() == -1);

}

void showMenu1() {

cout << "Меню:" << endl

<< "(1) Вывод всех официантов:" << endl

<< "(2) Добавление нового официанта:" << endl

<< "(3) Удаление официанта с указанным именем:" << endl

<< "(4) Запись списка официантов в файл:" << endl

<< "(5) Считывание списка официантов из файла:" << endl

<< "(6) Редактирование официанта с указанным именем"<< endl

<< "(7) Выход в главное меню"<< endl

<< "(0) Выход из программы:" << endl;

while (getMenuItem1() == -1);

}

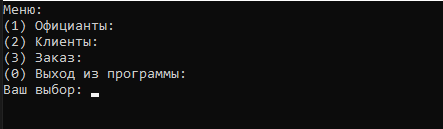
Для создания меню, мы сделали класс controller для предыдущих сущностей для создания общего меню.

В модификации доступа private создается логическая переменная work для работы паттерна.

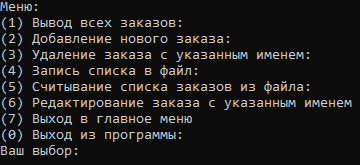
Для создания меню, мы прописали все возможные команды, которые сможет выполнять программа.

# 3. Вид приложения в консоли

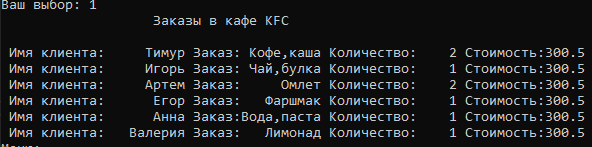
* Главное меню



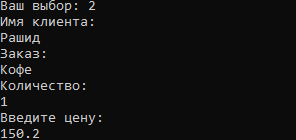
* На примере пункта 3 –Меню заказов.



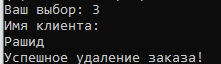
* Пункт 1 меню



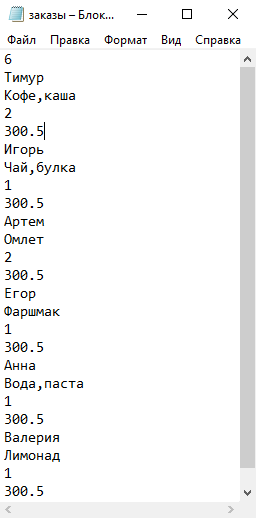
* Пункт 2 меню



* Пункт 3 меню



* Пункт 4 меню



* Пункт 6 меню

