Содержание

|  |  |
| --- | --- |
|  |  |
| Введение | 3 |
| 1 Описание предметной области | 4 |
| 1.1 Постановка задачи | 4 |
| 1.2 Объектно-ориентированный анализ предметной области | 5 |
| 2 Проектирование программного средства | 6 |
| 2.1 Проектирование структур данных | 6 |
| 2.2 Проектирование взаимодействия с пользователем | 8 |
| 2.3 Проектирование интерфейса | 9 |
| 3 Реализация программного средства | 11 |
| 3.1 Реализация классов и структур данных | 11 |
| 3.2 Реализация требований | 13 |
| 3.3 Реализация концепций ООП | 13 |
| 3.4 Описание структуры проекта | 13 |
| 3.5 Реализация интерфейса взаимодействия с пользователем | 14 |
| 4 Тестирование программного модуля | 16 |
| Заключение | 23 |
| Список использованных источников | 24 |
| Приложение А. Код программы | 25 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Введение

Целью курсовой работы является создание программного обеспечения для ведения учета заработной платы сотрудников предприятия.

Задачи курсовой работы: разработать программное обеспечение, протестировать программное обеспечение, описать работу программного обеспечения.

Ожидаемый результат курсового проекта: должно быть полное рабочее программное обеспечение. Оно должно выводить всю информацию о заработной плате сотрудников. Добавлять, удалять, сортировать записи. Должно быть подробное и понятное меню для пользователя. Также должна содержаться кнопка выхода из программного обеспечения. Программа должна работать корректно, исключая ошибки ввода от пользователя.

В пояснительной записке описан процесс создания программного продукта, описаны принципы его работы и результаты тестирования.

1 Описание предметной области

* 1. Постановка задачи

Целью курсовой работы является создание программного обеспечения на языке С++ для ведения учета заработной платы сотрудников предприятия.

Данные программы должны храниться в файлах. Разрабатываемое меню должно быть понятным и защищенным от случайных ошибок, вылетов и ввода некорректных данных. Ввод исходных данных должен осуществляться с клавиатуры.

Разрабатываемая программа должна обеспечить возможность выполнения перечисленных ниже функций:

* вывод записей о заработной плате в консоль;
* добавление записей о заработной плате;
* удаление записей;
* сортировка записей по дате создания, имени и фамилии сотрудника;
* чтение из файла;
* запись в файл.

Требования к организации ввода входных данных:

* Входные данные должны соответствовать меню программного обеспечения (одна из цифр пункта меню)

Требования к организации вывода входных данных:

Информация о записях в виде сгруппированной текстовой информации в консоли.

Тип пользовательского интерфейса-консольное приложение.

Функциональные требования к разрабатываемому ПО представлены на рисунке 1.1.

При выполнении программы необходимо предусмотреть обработку следующих ошибок:

* несоответствие номера пункта меню
* неправильный формат исходных данных

Функциональные требования представлены на рисунке 1.1 в виде диаграммы прецедентов.

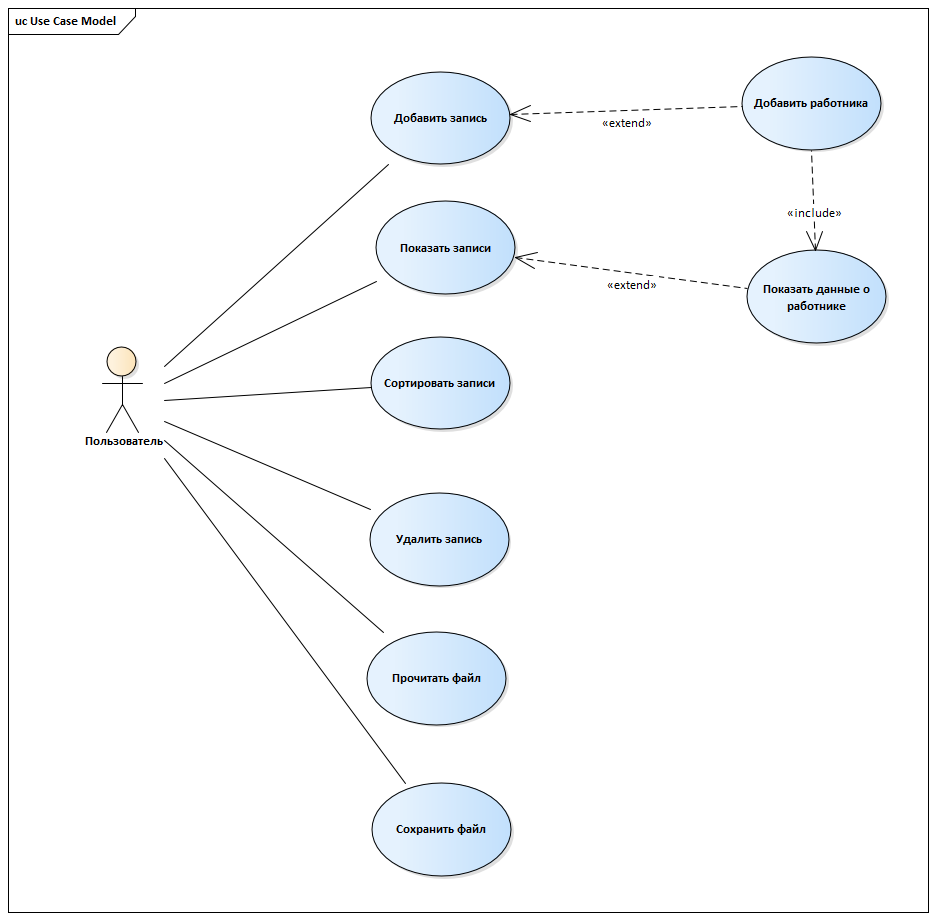


Рисунок 1.1 – Диаграмма вариантов использования

1.2 Объектно-ориентированный анализ предметной области

Запись характеризуется датой, сотрудником.

Дата характеризуется годом, месяцем, числом создания.

Сотрудник предприятия характеризуется именем, фамилией, отчеством, адресом, контактами, должностью, отработанными часами, заработной платой.

2.Проектирование программного средства

* 1. Проектирование структур данных

В соответствии с пунктом 1.2 для реализации предметной области выделены классы:

* запись;
* сотрудник;
* дата;

Для реализации хранения данных предусмотрен класс Storage.

Меню организованно с использованием класса Menu.

В разрабатываемом ПО имеется один вектор: accounts.

Диаграмма классов представлена на рисунке 2.1.

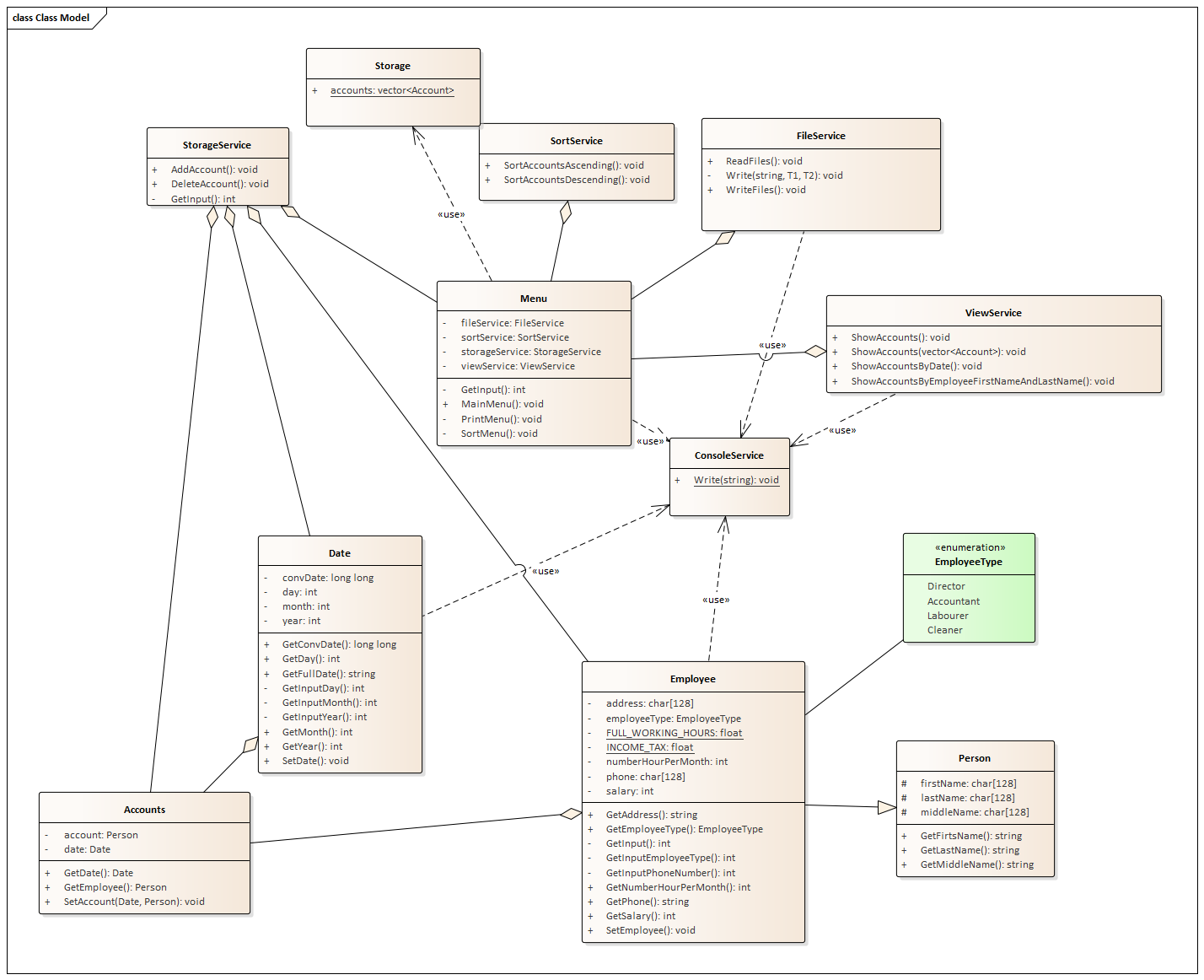


Рисунок 2.1 – Диаграмма классов

Структура классов хранящих данные предметной области представлена в таблицах 2.1-2.4.

Таблица 2.1 – Описание полей класса Товар

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Ограничение |
| Наименование | char | Длина от 1 до 128 символов |
| Категория | char | Длина от 1 до 128 символов |
| Цена | int | Положительное |

Таблица 2.2 – Описание полей класса Сотрудник

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Ограничение |
| Имя | char | Длина от 1 до 128 символов |
| Фамилия | char | Длина от 1 до 128 символов |
| Отчество | int | Положительное |
| Должность | char | Длина от 1 до 128 символов |
| Заработная плата | int | Положительное |
| Контакты | char | Длина от 1 до 128 символов |

Таблица 2.3 – Описание полей класса Клиент

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Ограничение |
| Имя | char | Длина от 1 до 128 символов |
| Фамилия | char | Длина от 1 до 128 символов |
| Отчество | int | Положительное |
| Контакты | char | Длина от 1 до 128 символов |
| Адрес проживания | char | Длина от 1 до 128 символов |

Таблица 2.4 – Описание полей класса Заказ

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Ограничение |
| Товар | char | Длина от 1 до 128 символов |
| Цена товара | char | Длина от 1 до 128 символов |
| Контакты покупателя | int | Положительное |
| Количество | int | Положительное |
| Доставка | int | 1 или 2 |
| Итоговая цена | int | Положительное |
| Дата заказа | char | Длина от 1 до 128 символов |
| Адрес доставки | char | Длина от 1 до 128 символов |

* 1. Проектирование взаимодействия с пользователем

Реализация прецедент Добавление сотрудника представлена в виде диаграммы взаимодействия, рисунок 2.2.

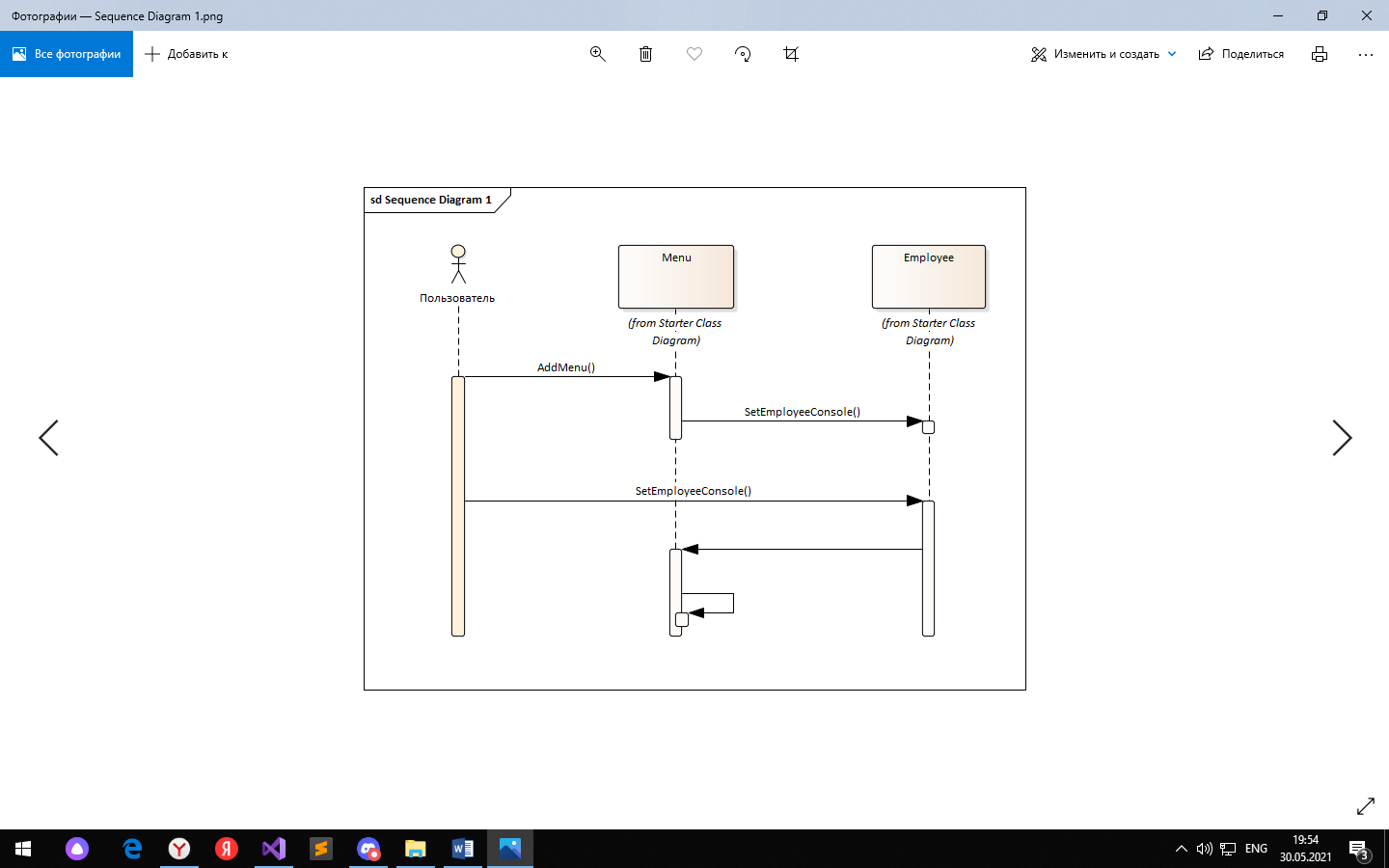


Рисунок 2.2 – прецедент Добавление сотрудника

Реализация прецедент Добавление сотрудника представлена в виде диаграммы взаимодействия, рисунок 2.3.

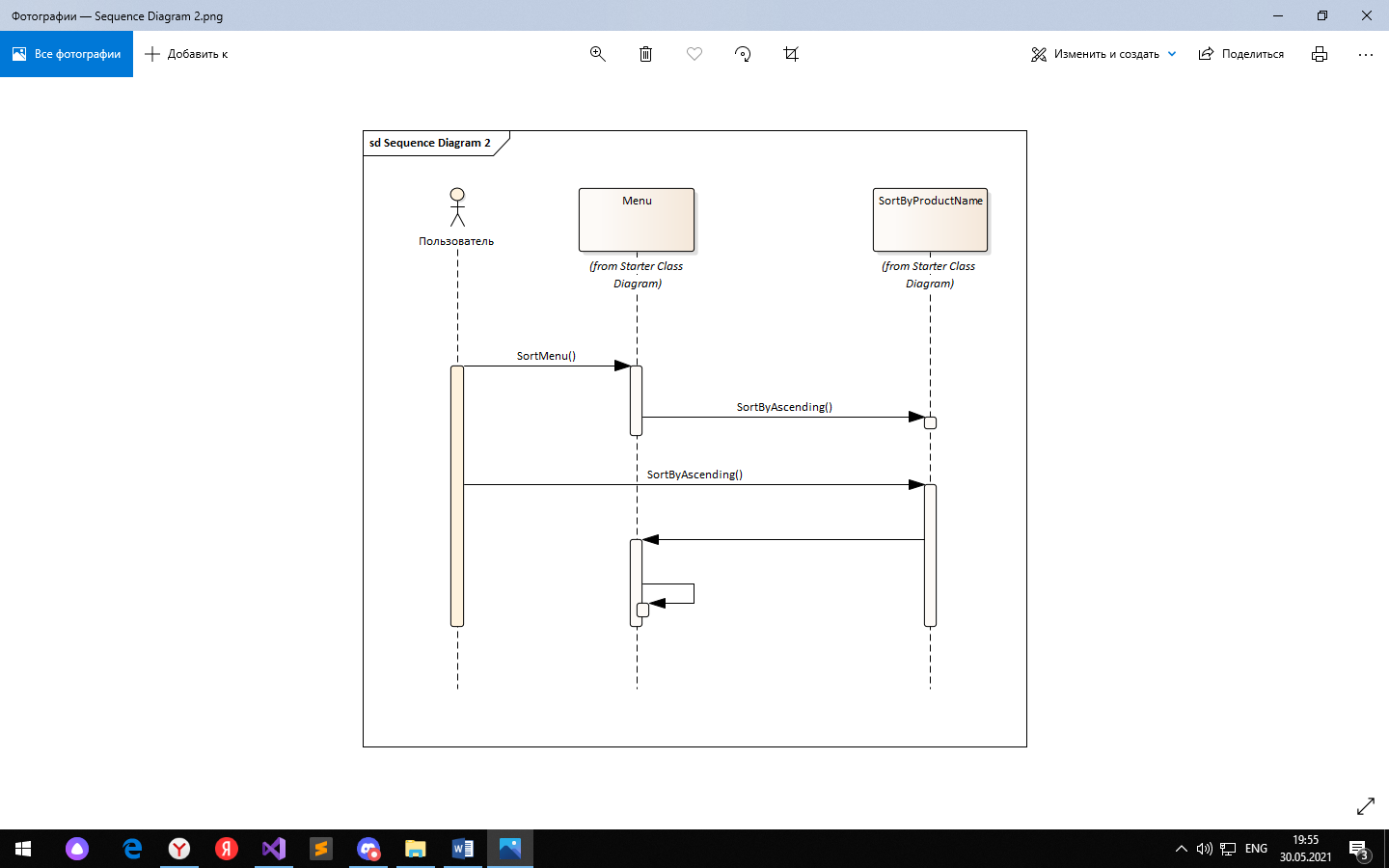


Рисунок 2.3 – прецедент Сортировка по возрастанию

Реализация прецедент Добавление сотрудника представлена в виде диаграммы взаимодействия, рисунок 2.4.

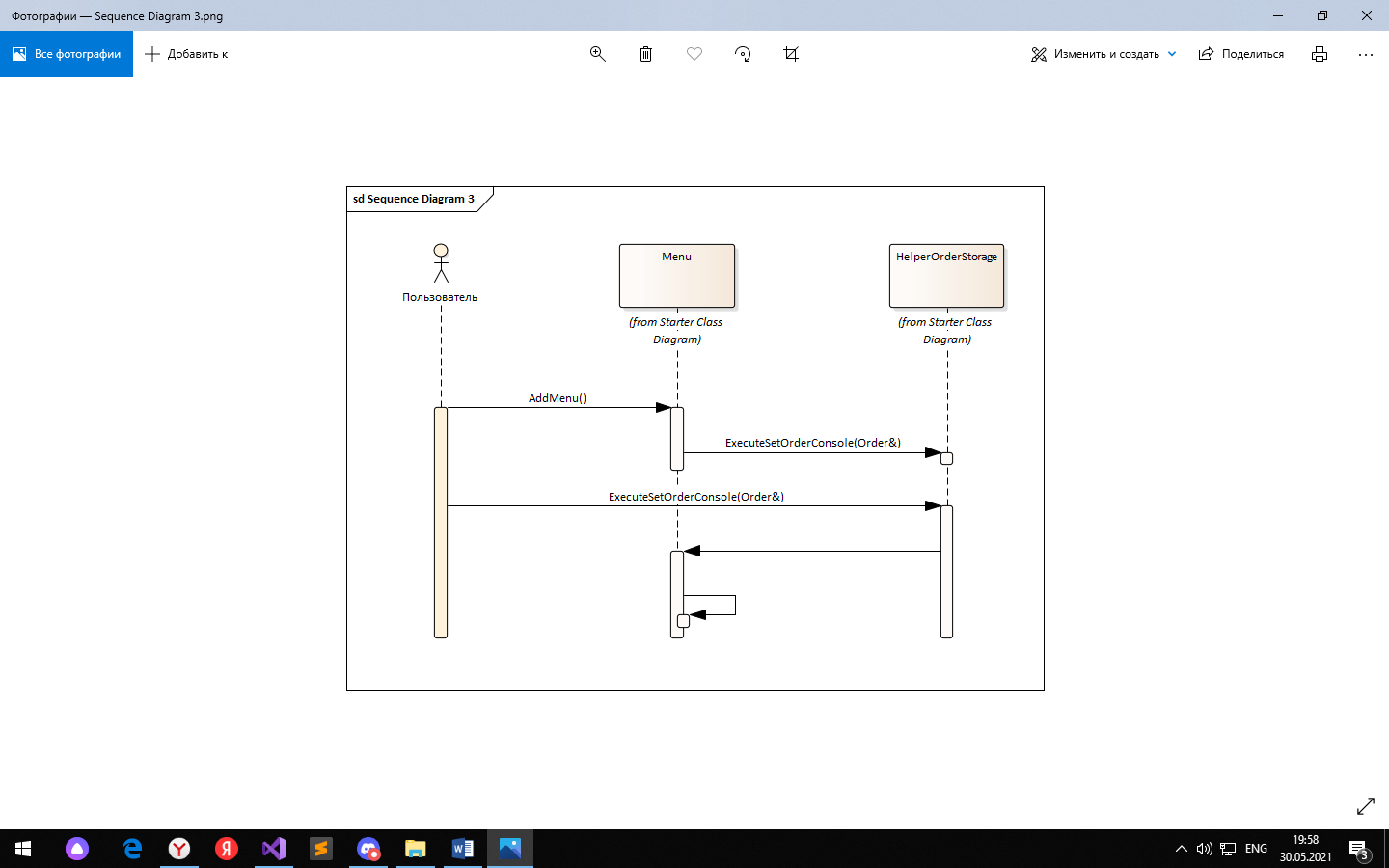


Рисунок 2.4 – прецедент Добавление заказа

* 1. Проектирование интерфейса

Схема меню для работы с пользователем продемонстрирована на рисунке 2.3

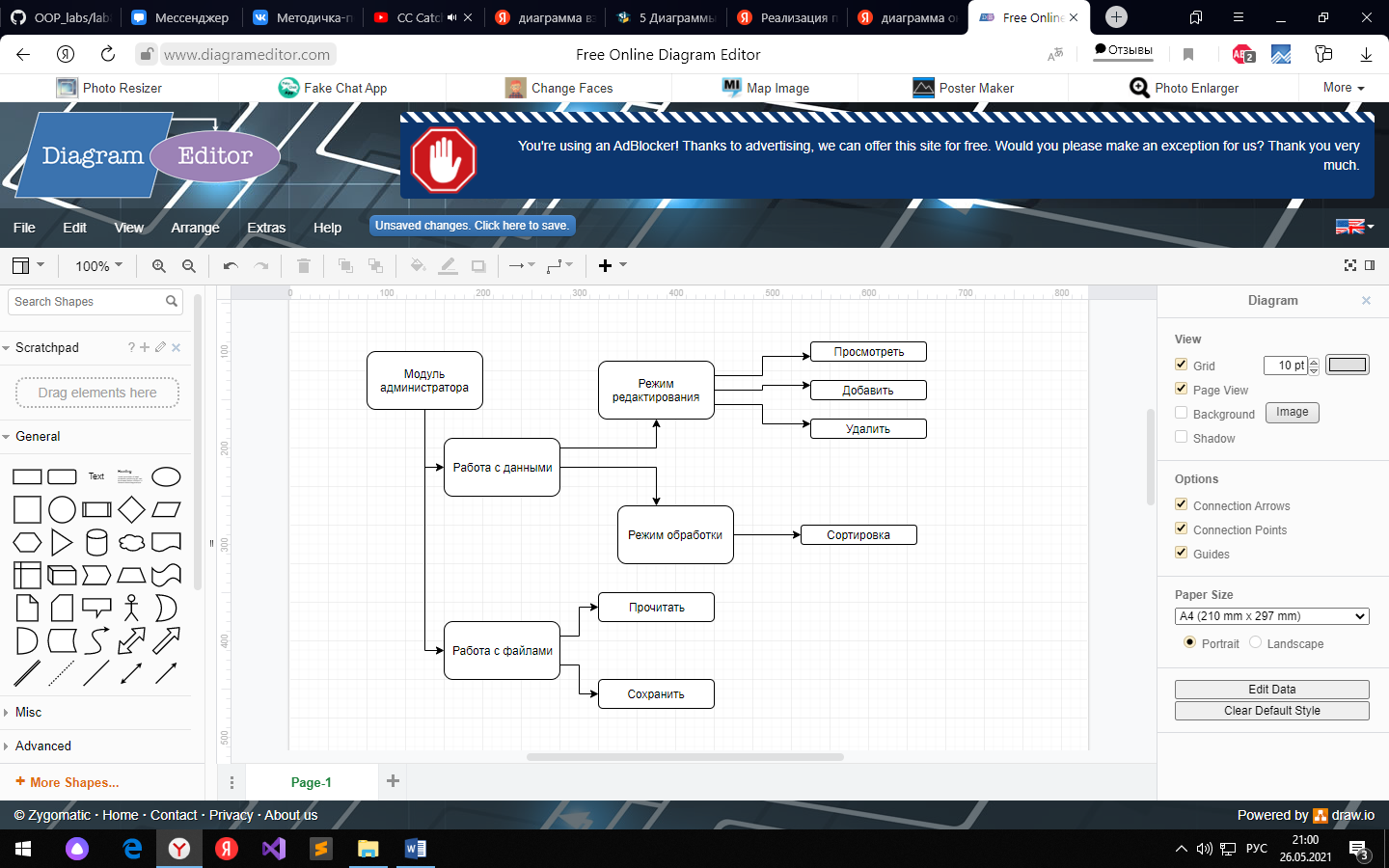


Рисунок 2.3 –Схема меню

1. Реализация программного средства
   1. Реализация классов и структур данных

Реализация класса Buyers

public:

Buyers();

void SetBuyersConsole();

char\* GetFirstName();

char\* GetSecondName();

char\* GetThirdName();

char\* GetContacts();

char\* GetAdress();

Переменные firstName, secondName, thirdName,adress,contacts класс Buyers наследует у класса Human.

Реализация класса Human

protected:

char firstName[128];

char secondName[128];

char thirdName[128];

char contacts[128];

char adress[128];

Реализация класса Product

private:

ProductType productType;

int price;

char nameProduct[128];

int quantity;

public:

Product();

void SetProductConsole();

ProductType GetProductType();

char\* GetNameProduct();

int GetPrice();

int GetInput();

int GetInputProductType();

Реализация класса Employee

private:

static int id;

int employeeId;

char post[128];

int wages;

public:

Employee();

void SetEmployeeConsole();

int GetWages();

int GetInput();

int GetEmployeeId();

char\* GetFirstName();

char\* GetSecondName();

char\* GetThirdName();

char\* GetContacts();

char\* GetPost();

Переменные firstName, secondName, thirdName,adress,contacts класс Buyers наследует у класса Human.

Реализация класса Order

private:

Employee employee;

Product product;

Buyers buyers;

DeliveryType deliveryType;

int quantity;

int fullPrice;

DateOrder date;

AdressDelivery adress;

public:

Order()

void SetOrderConsole(Product product, Buyers buyers, int quantity)

void SetQuantity(int quantity)

void CalculatePrice()

DeliveryType GetDeliveryType()

string GetNumberOfFlat()

string GetNumberOfHome()

string GetStreetName()

string GetCityName()

int GetYear()

int GetMonthOfDate()

int GetDayOfDate()

int GetFullPrice()

int GetQuantity()

int GetProductPrice()

string GetNameProduct()

string GetContacts()

int GetOrderQuestion();

Подробный код программы продемонстрирован в приложении А

3.2 Реализация требований

Реализация требований согласно пункту 1.1 представлена в таблице 3.1.

Таблица 3.1 – Реализация требований

|  |  |  |
| --- | --- | --- |
| Функциональное требование | Реализация | Примечание |
| Должна производиться сортировка заказов по цене в порядке возрастания | Метод SortByAscending() | Параметр – копия вектора объектов класса Order |
| Должна производиться сортировка заказов по цене в порядке убывания | Метод SortByDescending() | Параметр – копия вектора объектов класса Order |
| Должно производится удаление товаров | Метод DeleteProduct() | Параметр – вектор объектов класса Product |
| Должно производится удаление заказов | Метод DeleteOrder() | Параметр – вектор объектов класса Order |
| Должно производится удаление сотрудников | Метод DeleteEmployee() | Параметр – вектор объектов класса Employee |
| Должно производится удаление клиентов | Метод DeleteBuyers() | Параметр – вектор объектов класса Buyers |

3.3 Реализация концепций ООП

Для реализации инкапсуляции все поля имеют спецификатор доступа private. В public занесены все методы, которые необходимы для использования программы. Геттеры и сеттеры необходимы для чтения и записи переменных в класс и из класса.Перегрузка операторов в программировании — один из способов реализации полиморфизма. Наследование позволяет строить иерархии, в которых классы-потомки получают свойства классов-предков и могут дополнять их или изменять.

3.4 Описание структуры проекта

Описание структуры проекта представлено на рисунке 3.4-1

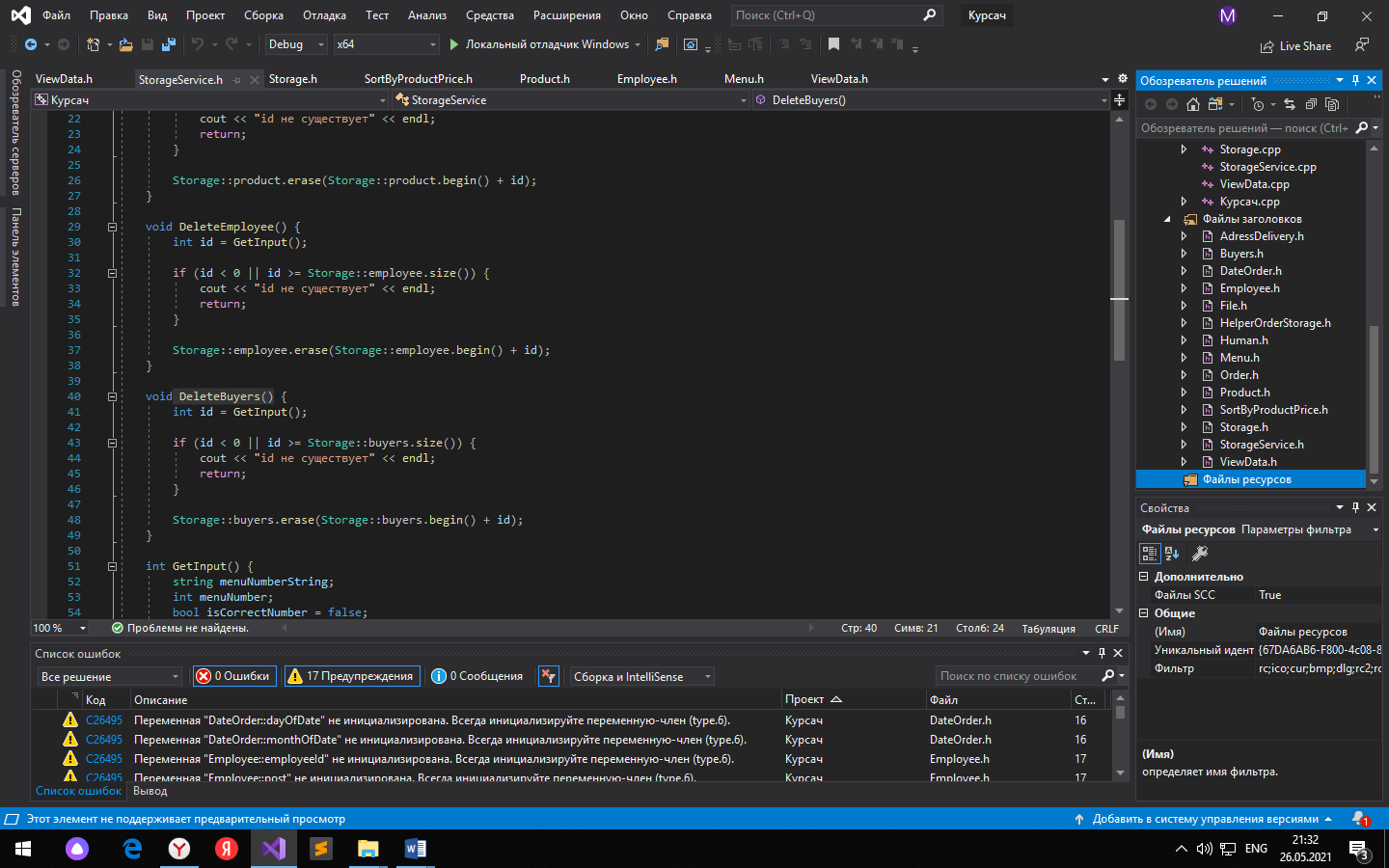
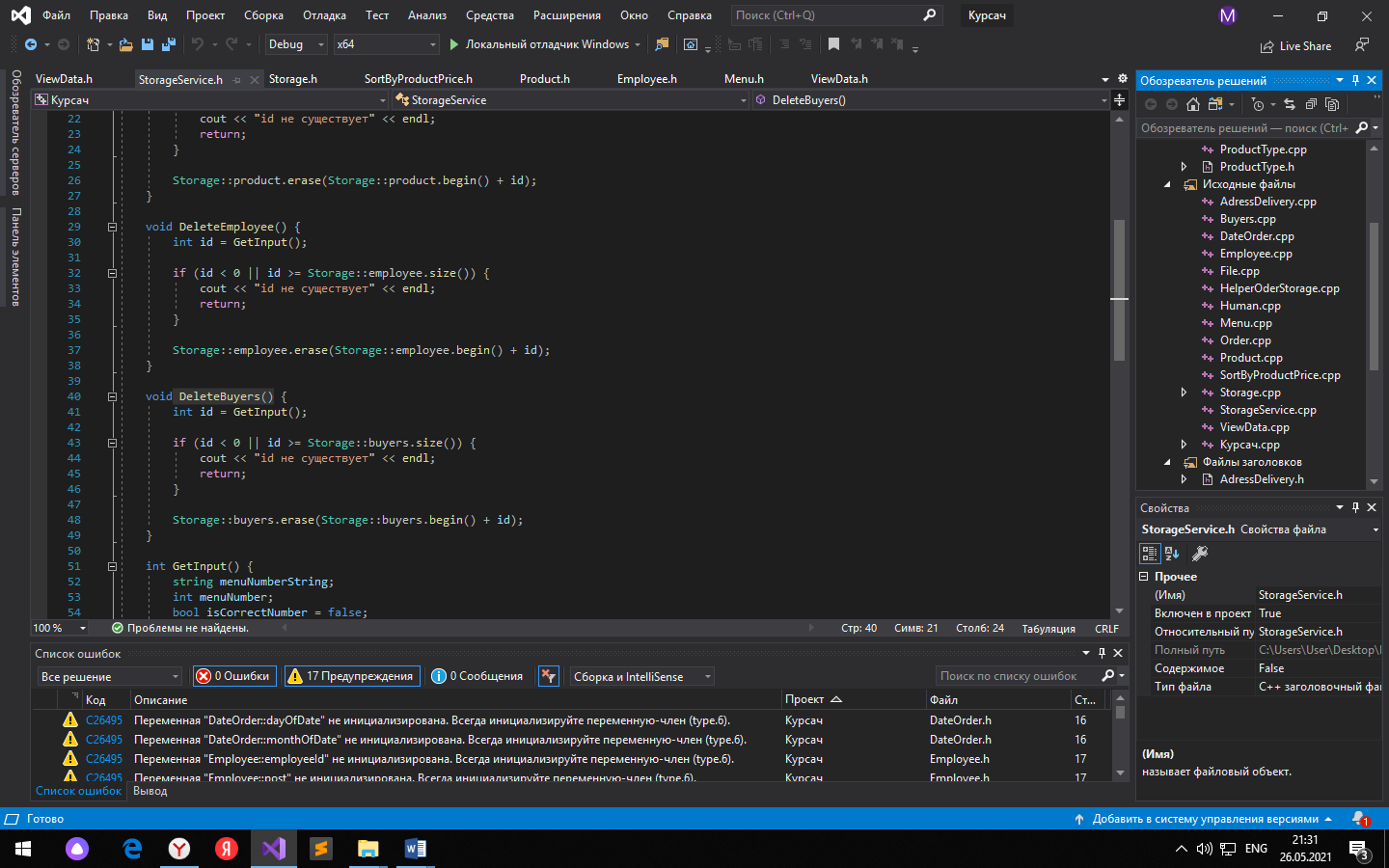
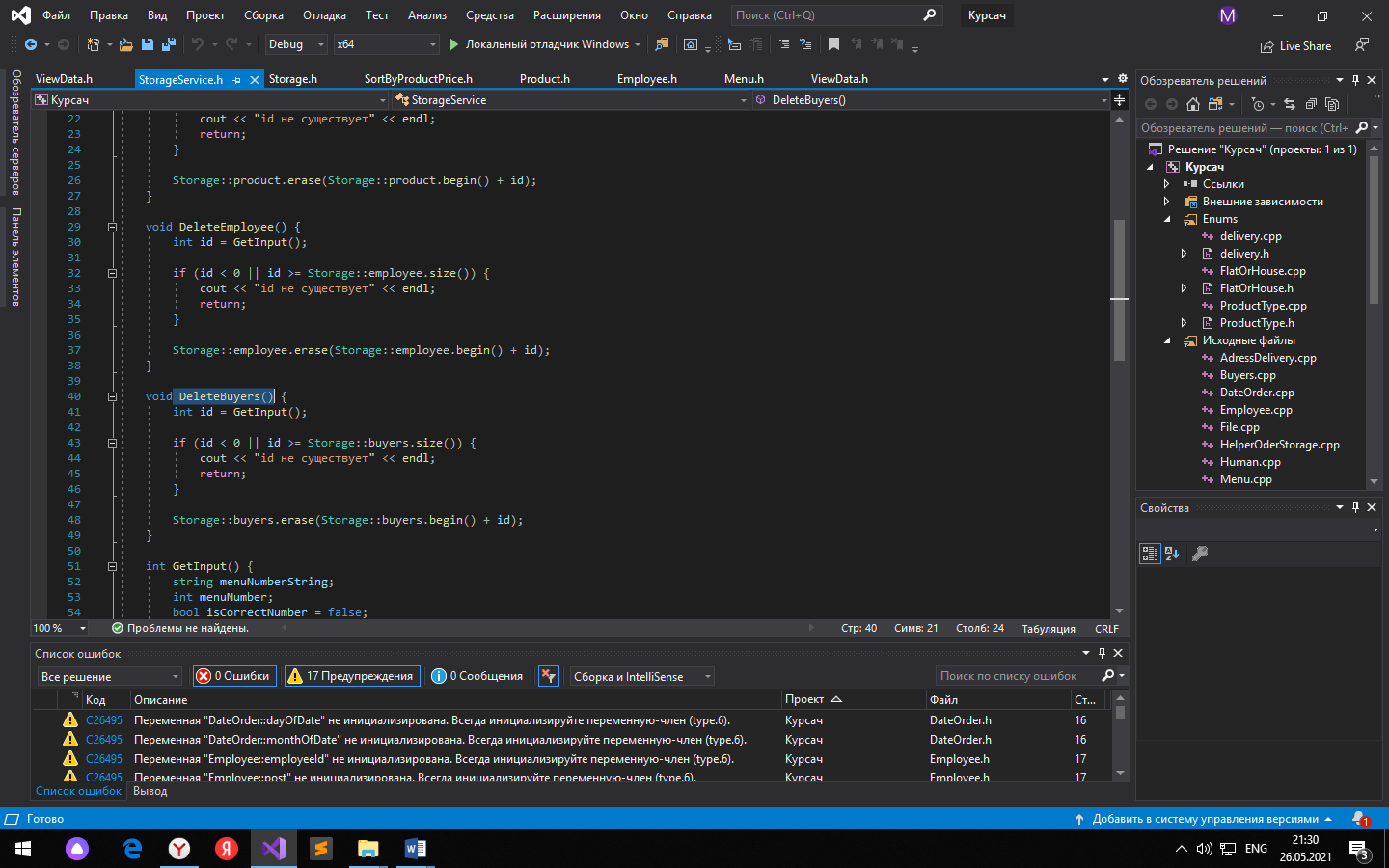


Рисунок 3.4-1 – Структура проекта

3.5 Реализация интерфейса взаимодействия с пользователем

При запуске программы пользователь увидит главное меню программы. (рисунок 3.1), в котором семь пунктов:

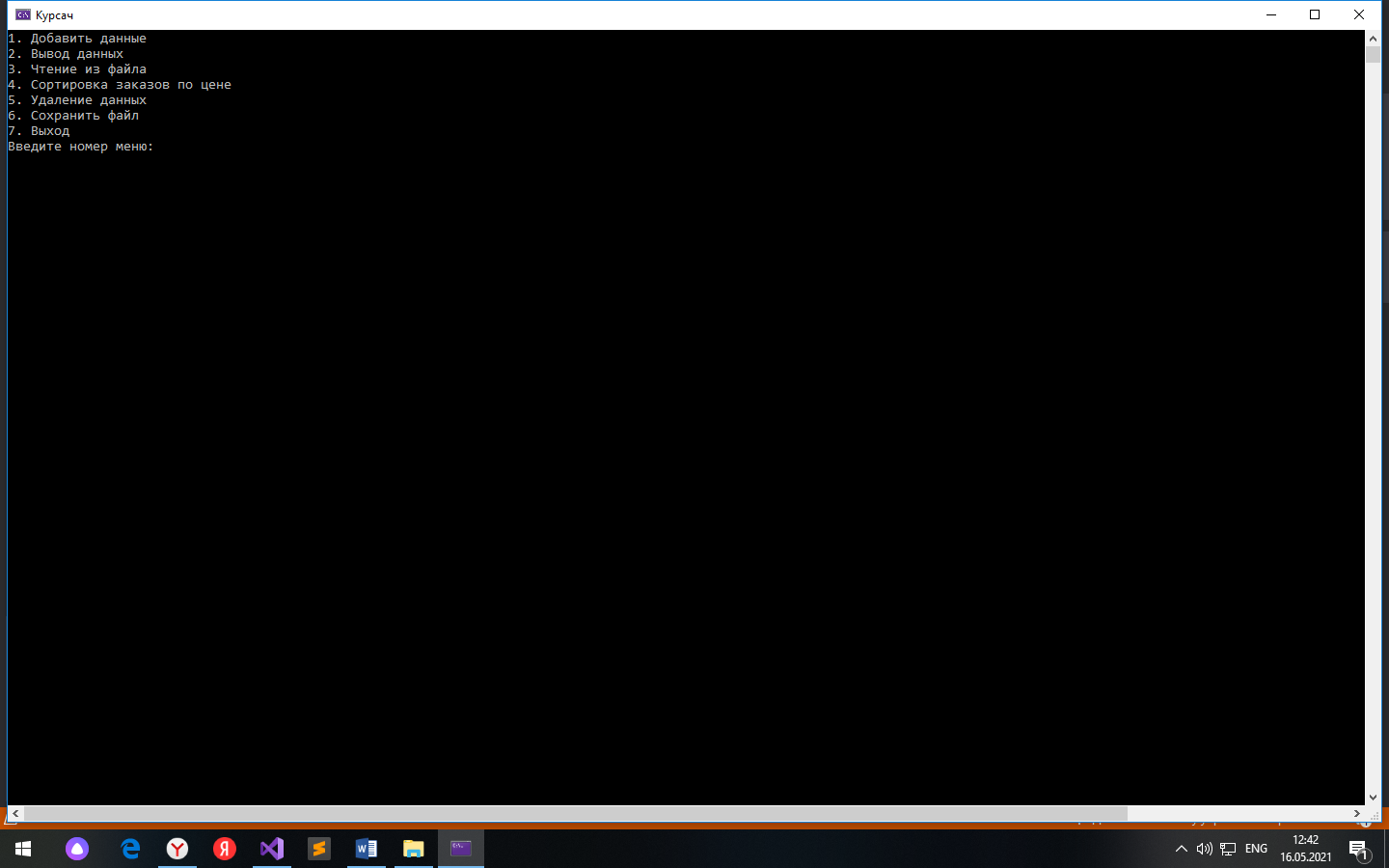


Рисунок 3.1 –Главное меню

При выборе пункта “Добавить данные” пользователь попадает в подменю. (рисунок 3.2),в котором может выбрать что ему необходимо добавить.

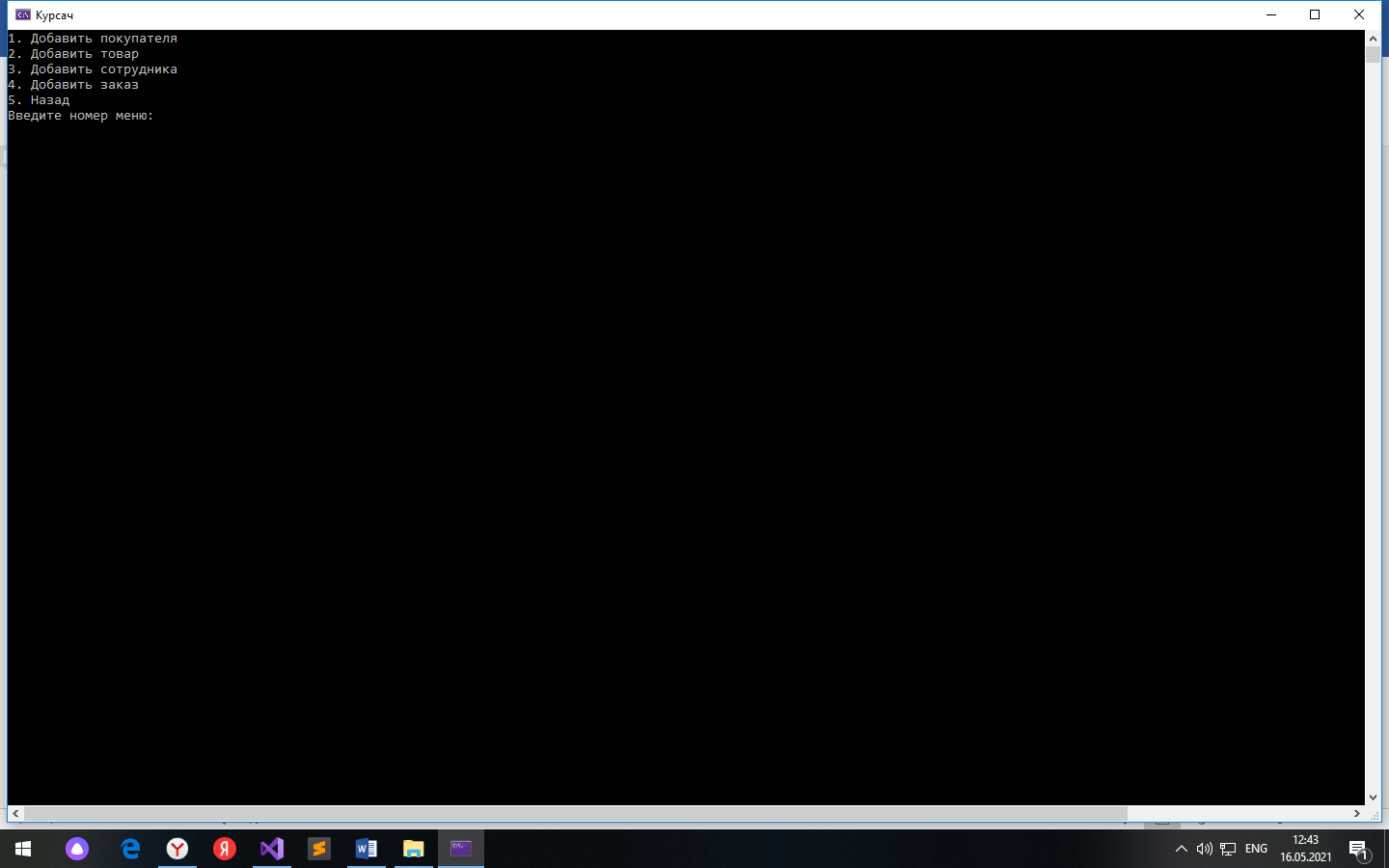


Рисунок 3.2 –Подменю пункта “Добавить данные”

При выборе пункта “Вывод данных” пользователь попадает в меню, в котором может выбрать что ему необходимо вывести.(рисунок 3.3)

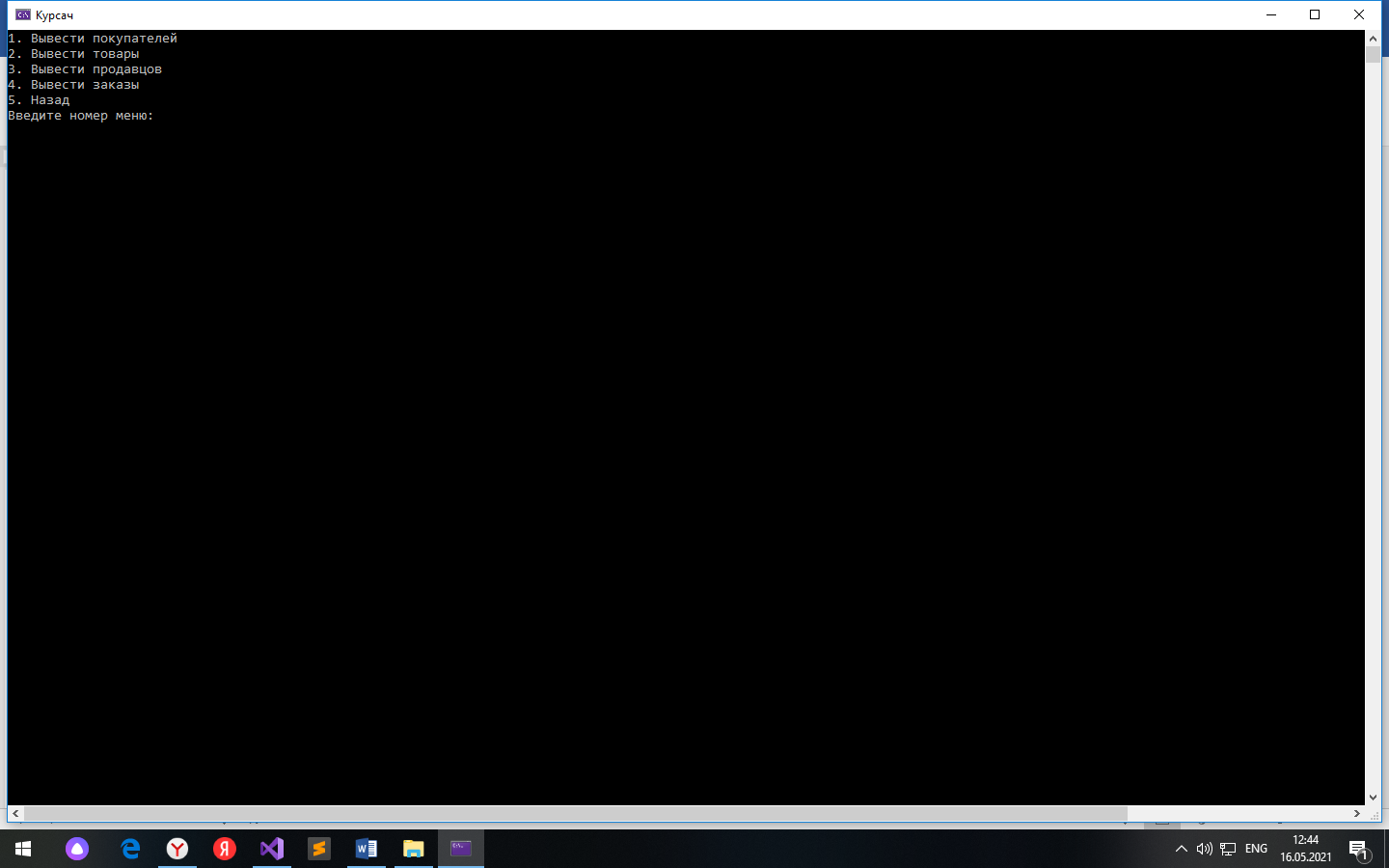


Рисунок 3.3 – Подменю пункта “Вывод данных”

При выборе пункта “Чтение из файла” программа считывает данные из файлов с названием “ Employee.dat”,“ Order.dat”, “Buyers.dat”, “Product.dat”.

При выборе пункта “Сортировка заказов по цене” пользователь попадает в подменю с выбором сортировки. (рисунок 3.4)

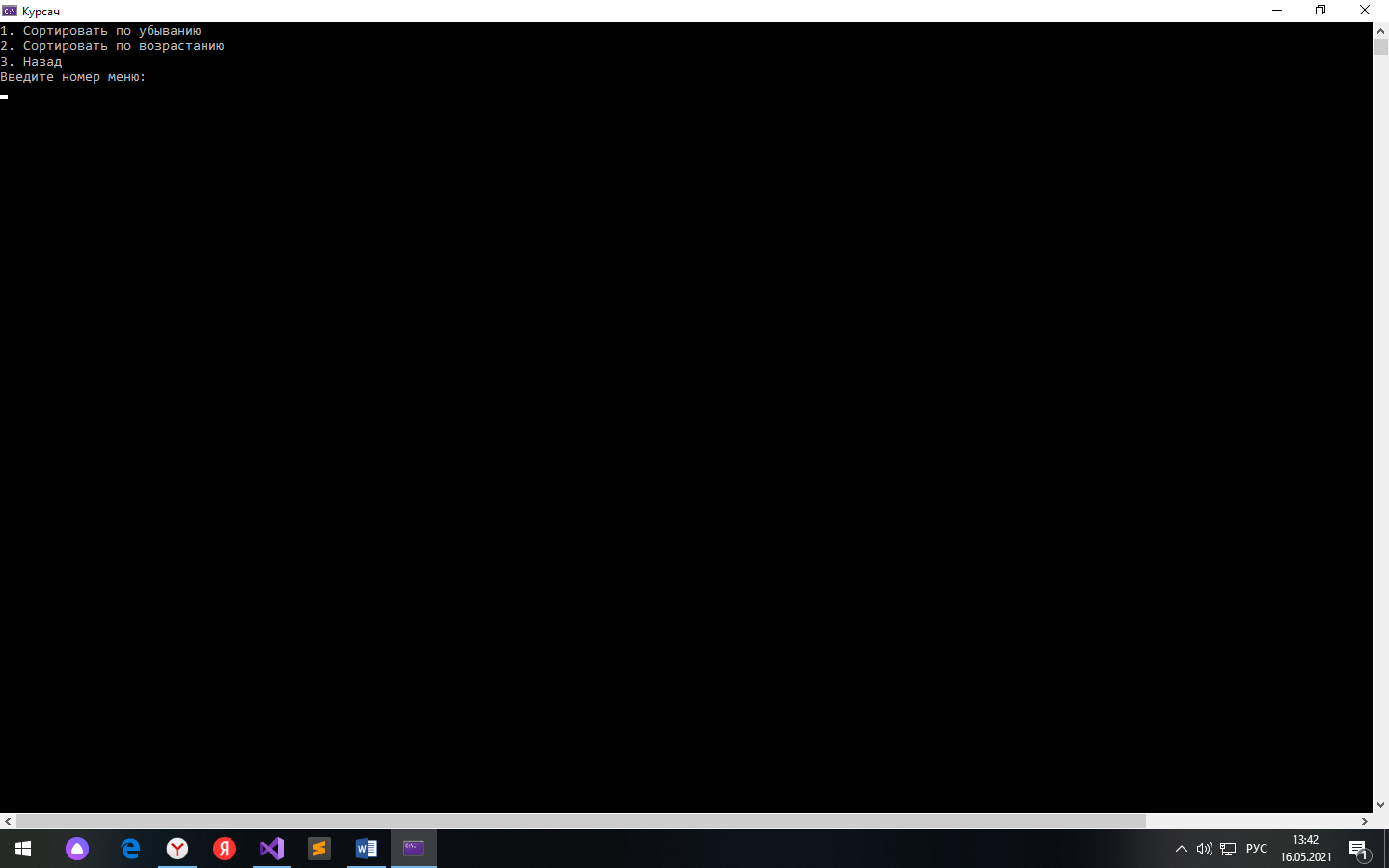


Рисунок 3.4 – Подменю пункта “Сортировка заказов по цене”

При выборе пункта “Удаление данных” пользователь попадает в подменю с выбором, что он хочет удалить. (Рисунок 3.5)

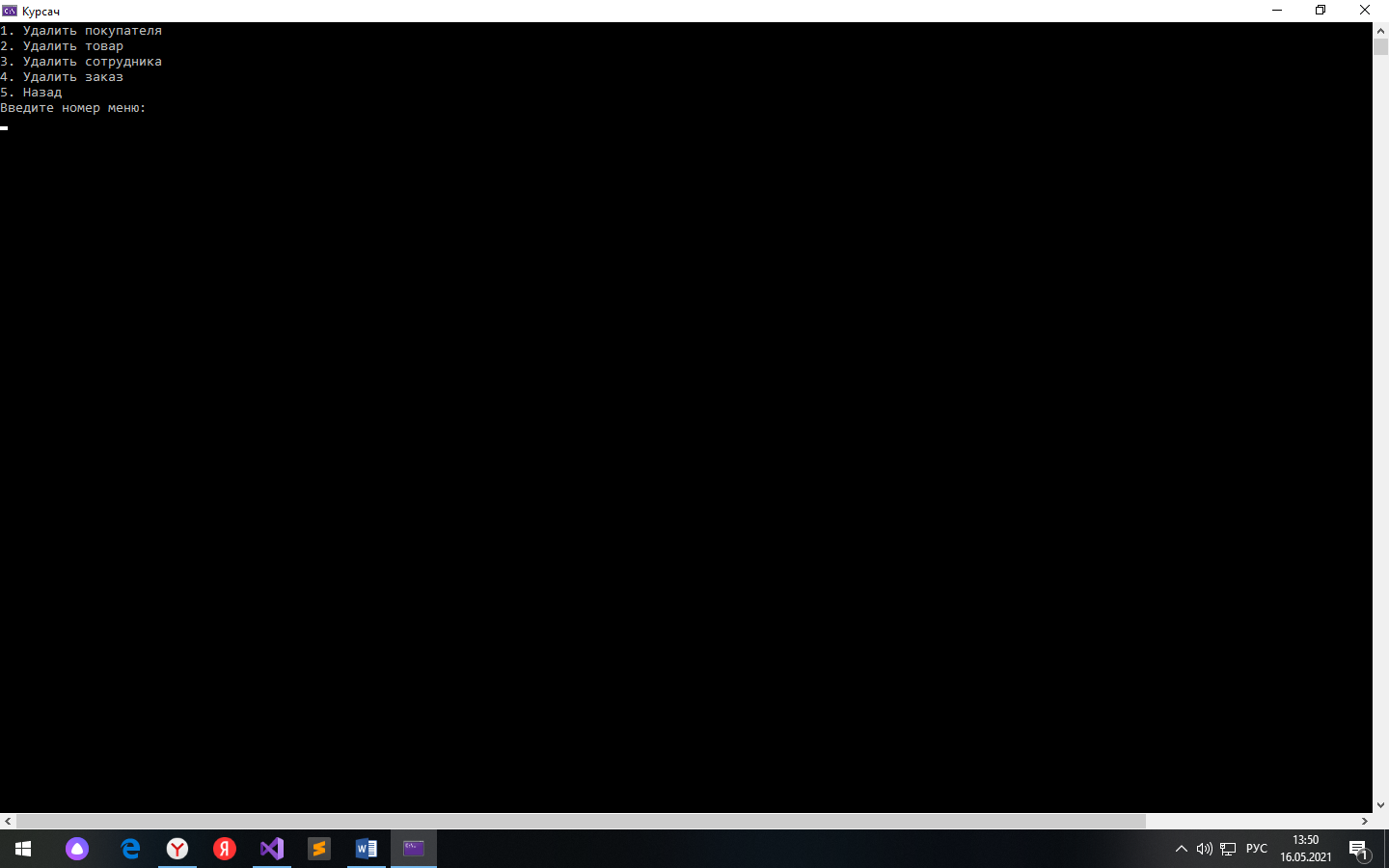


Рисунок 3.5 – Подменю пункта “Удаление данных”

При выборе пункта “Сохранить файл” программа создаёт новые файлы с названием “ Employee.dat”,“ Order.dat”, “Buyers.dat”, “Product.dat”(Рисунок 3.7)

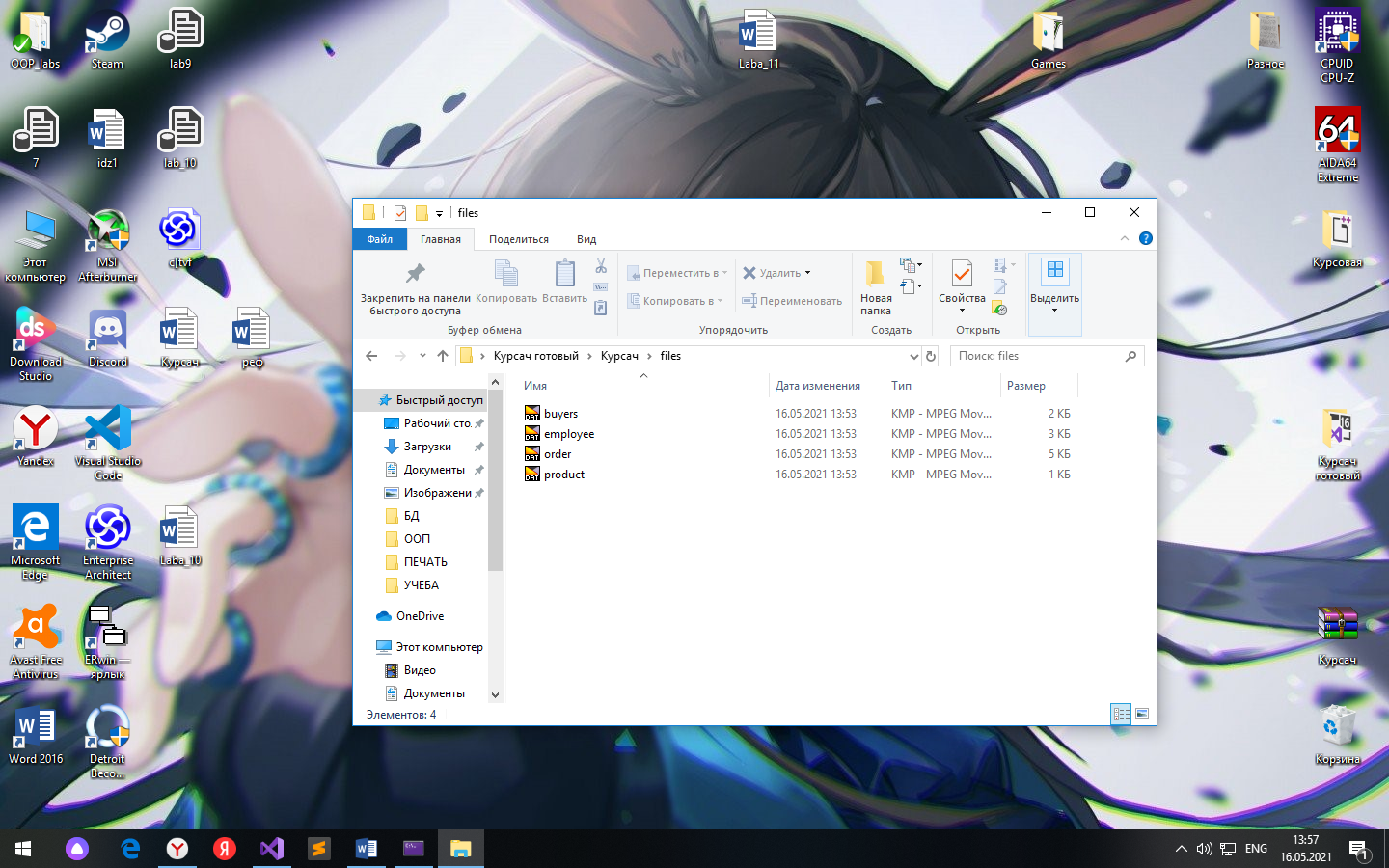


Рисунок 3.6 –Файлы из папки проекта

При выборе пункта “Выход” программа закрывается.

4 Тестирование программного модуля

При запуске программы пользователь увидит главное меню программы. (рисунок 4.1), в котором семь пунктов:

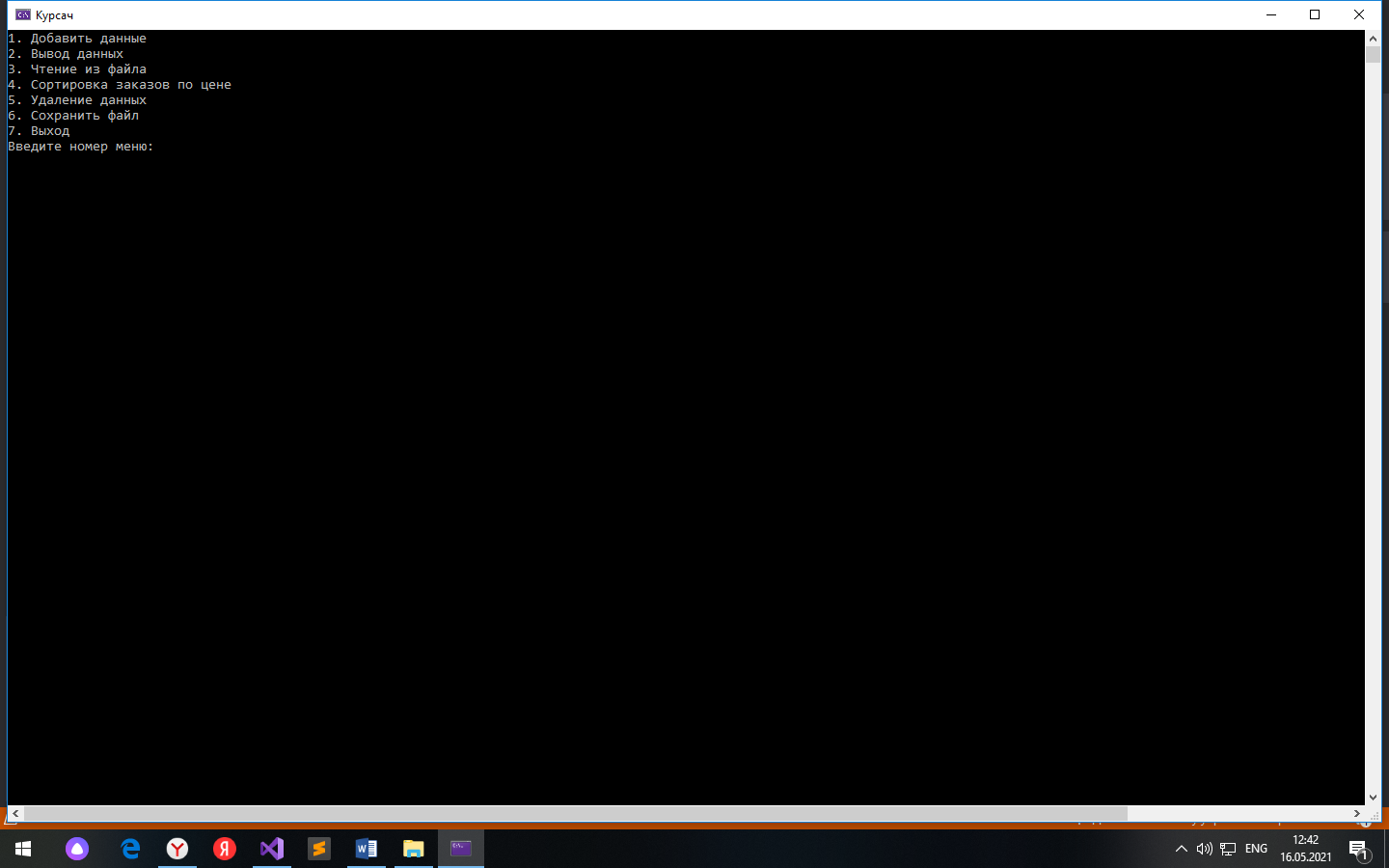


Рисунок 4.1 – Главное меню ПО

При выборе пункта “Добавить данные” пользователь попадает в подменю. (рисунок 4.2),в котором может выбрать что ему необходимо добавить.

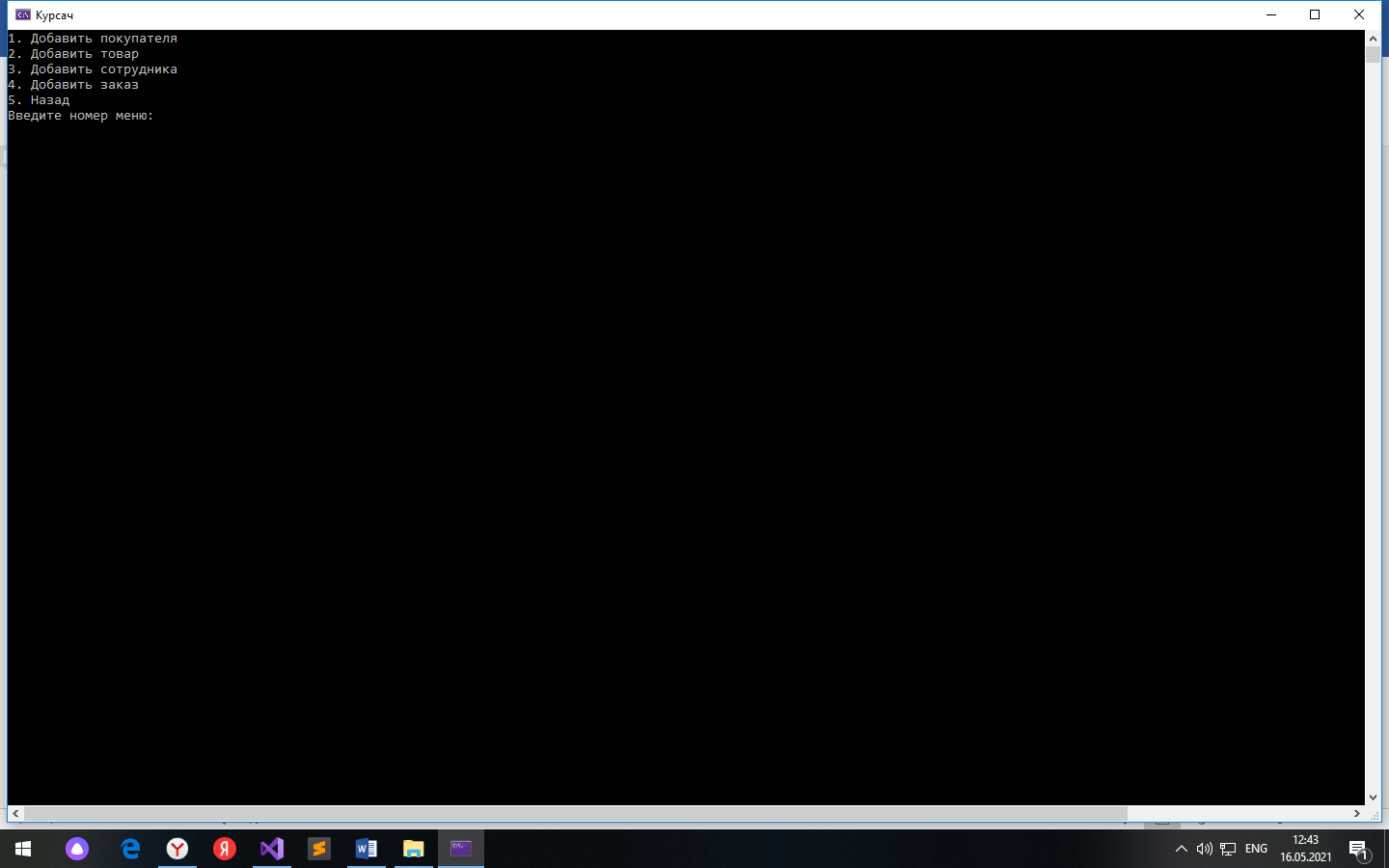


Рисунок 4.2 – Подменю пункта “Добавить данные”

При выборе первого пункта пользователь попадает в окно внесения данных покупателя. (рисунок 4.3)

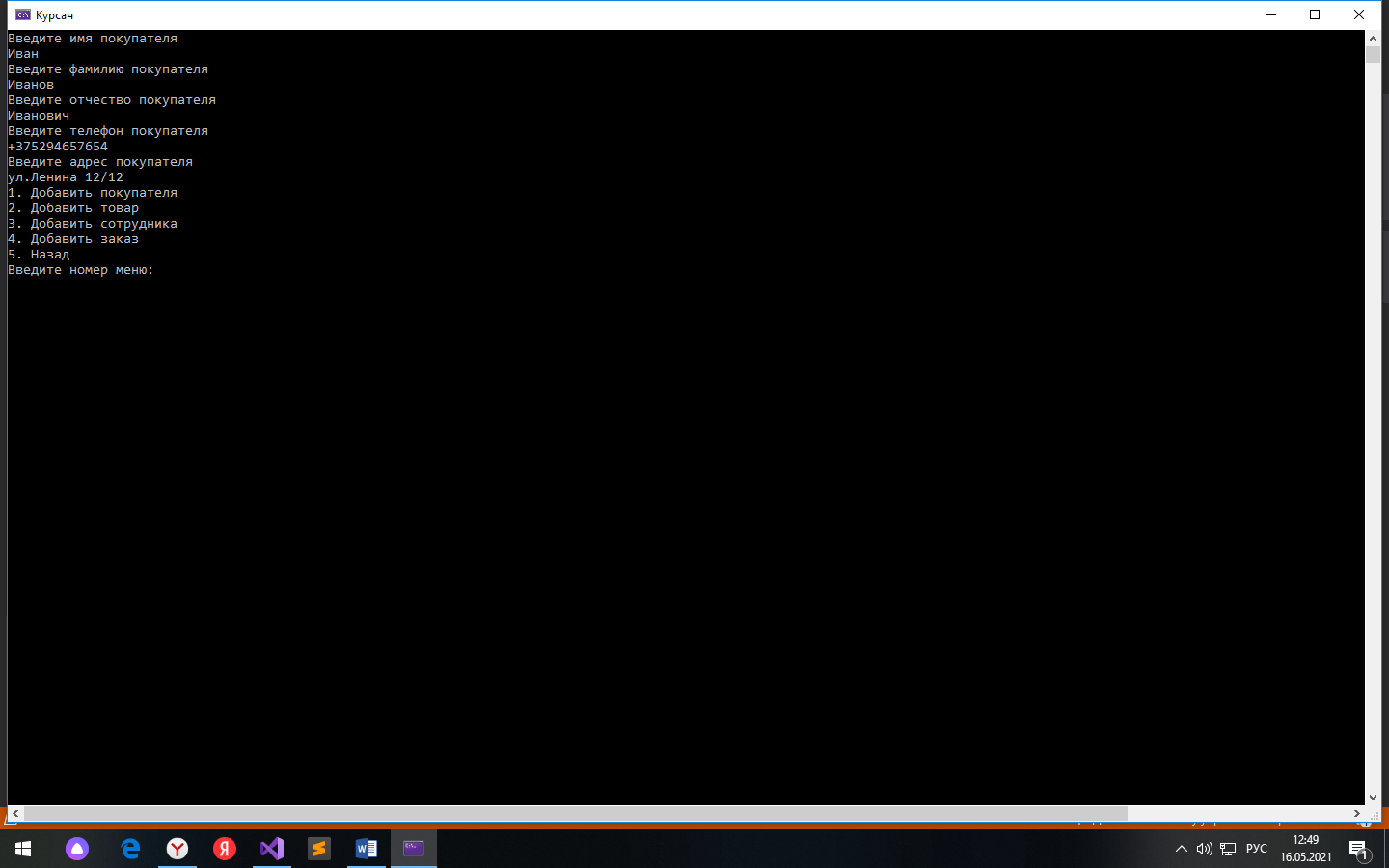


Рисунок 4.3 – Демонстрация внесения данных покупателя

При выборе второго пункта пользователь попадает в окно внесения данных о товаре. (рисунок 4.4)

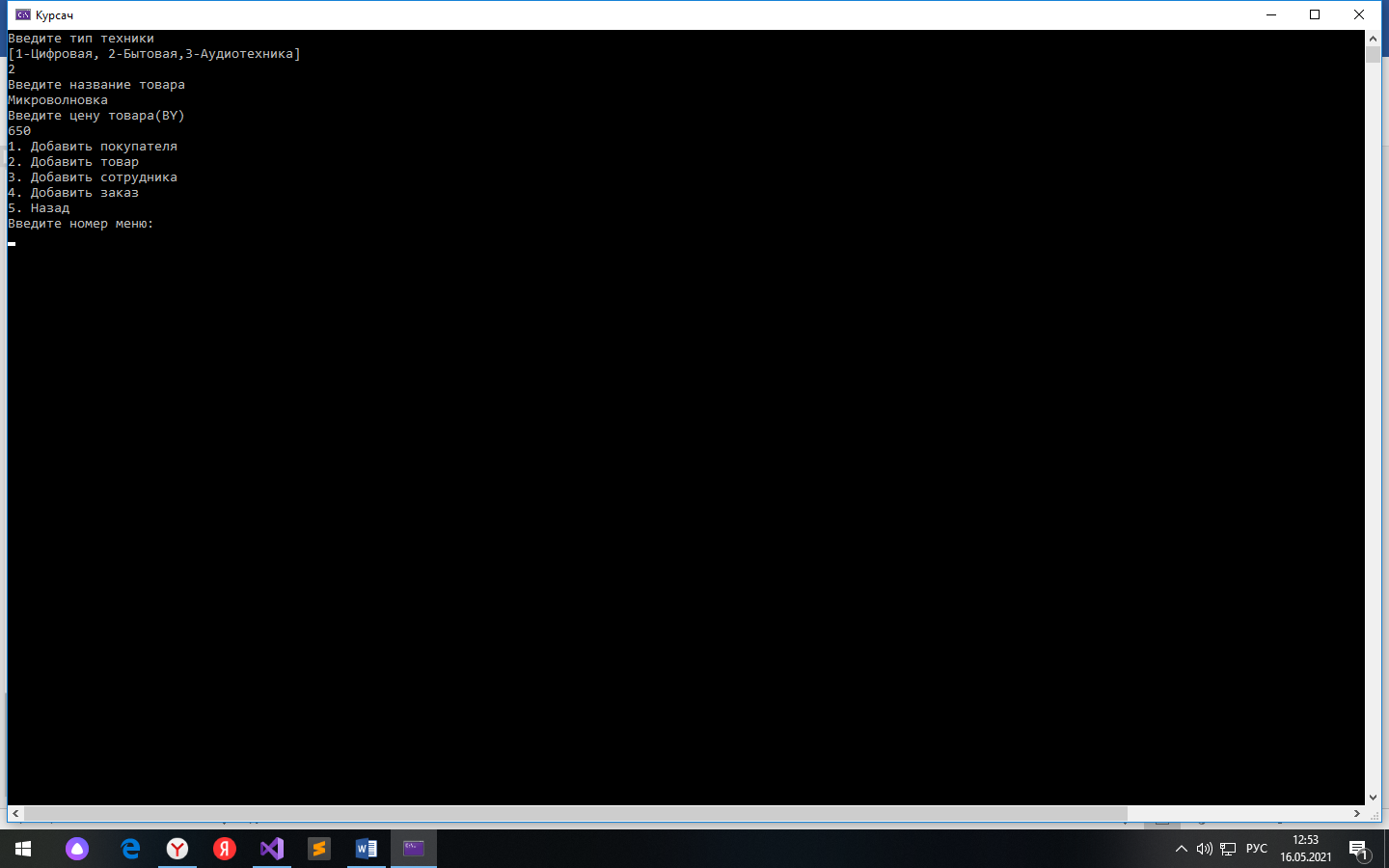


Рисунок 4.4 – Демонстрация внесения данных о товаре

При выборе третьего пункта пользователь попадает в окно внесения данных сотрудника. (рисунок 4.5)

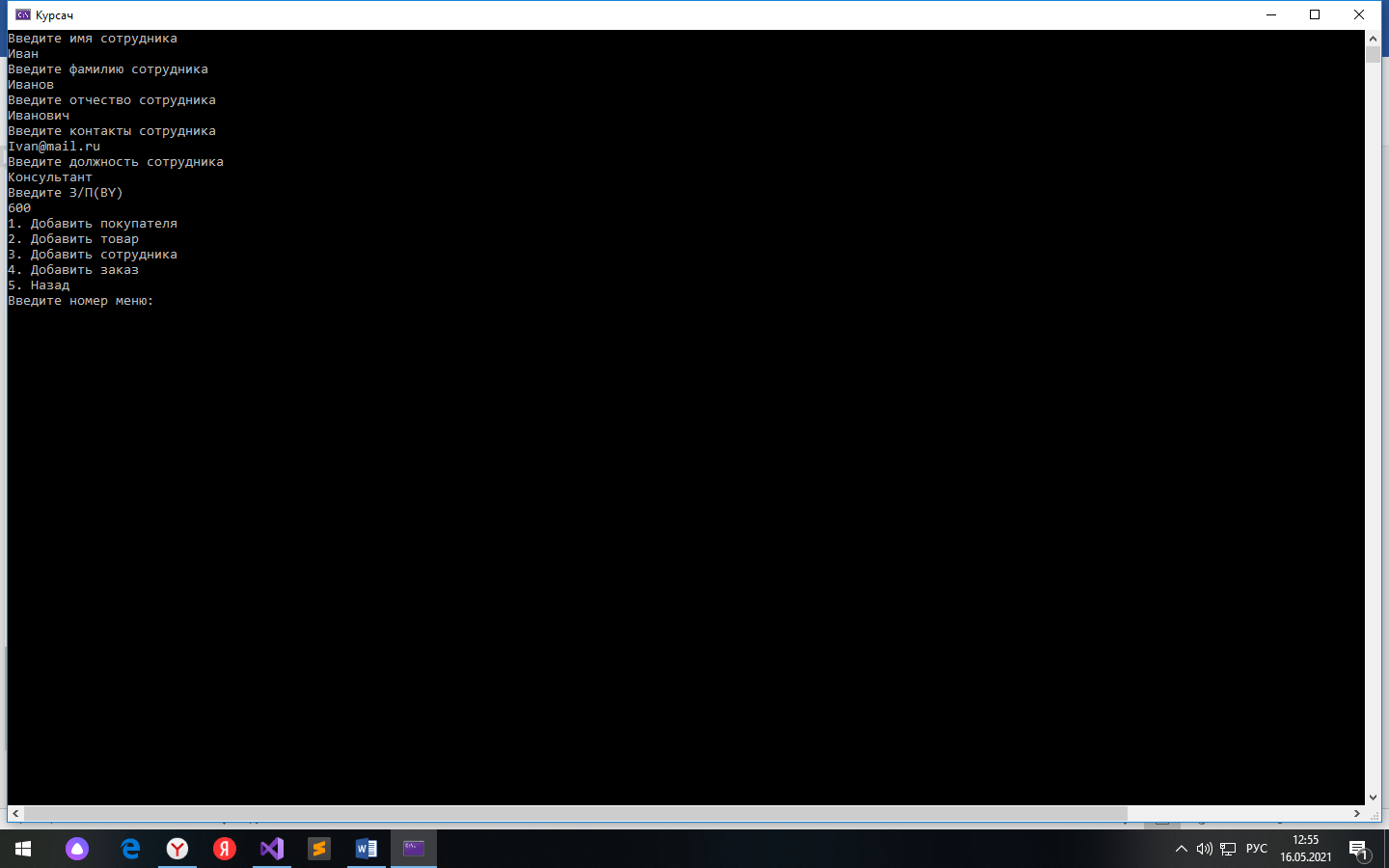


Рисунок 4.5 – Демонстрация внесения данных сотрудника

При выборе четвертого пункта пользователь попадает в окно внесения данных о заказе. (рисунок 4.6,рисунок 4.7)

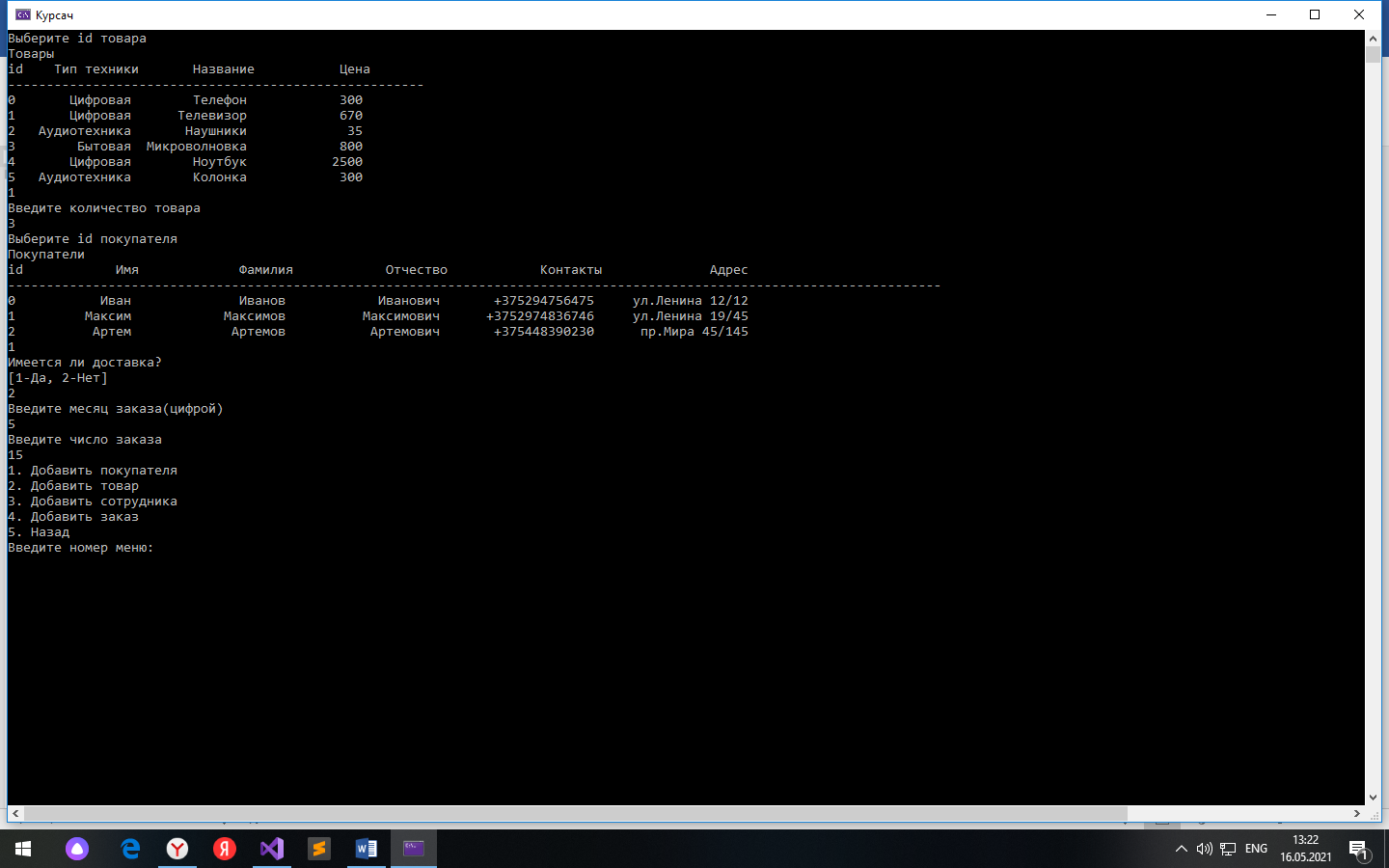


Рисунок 4.6 – Демонстрация оформления заказа без доставки

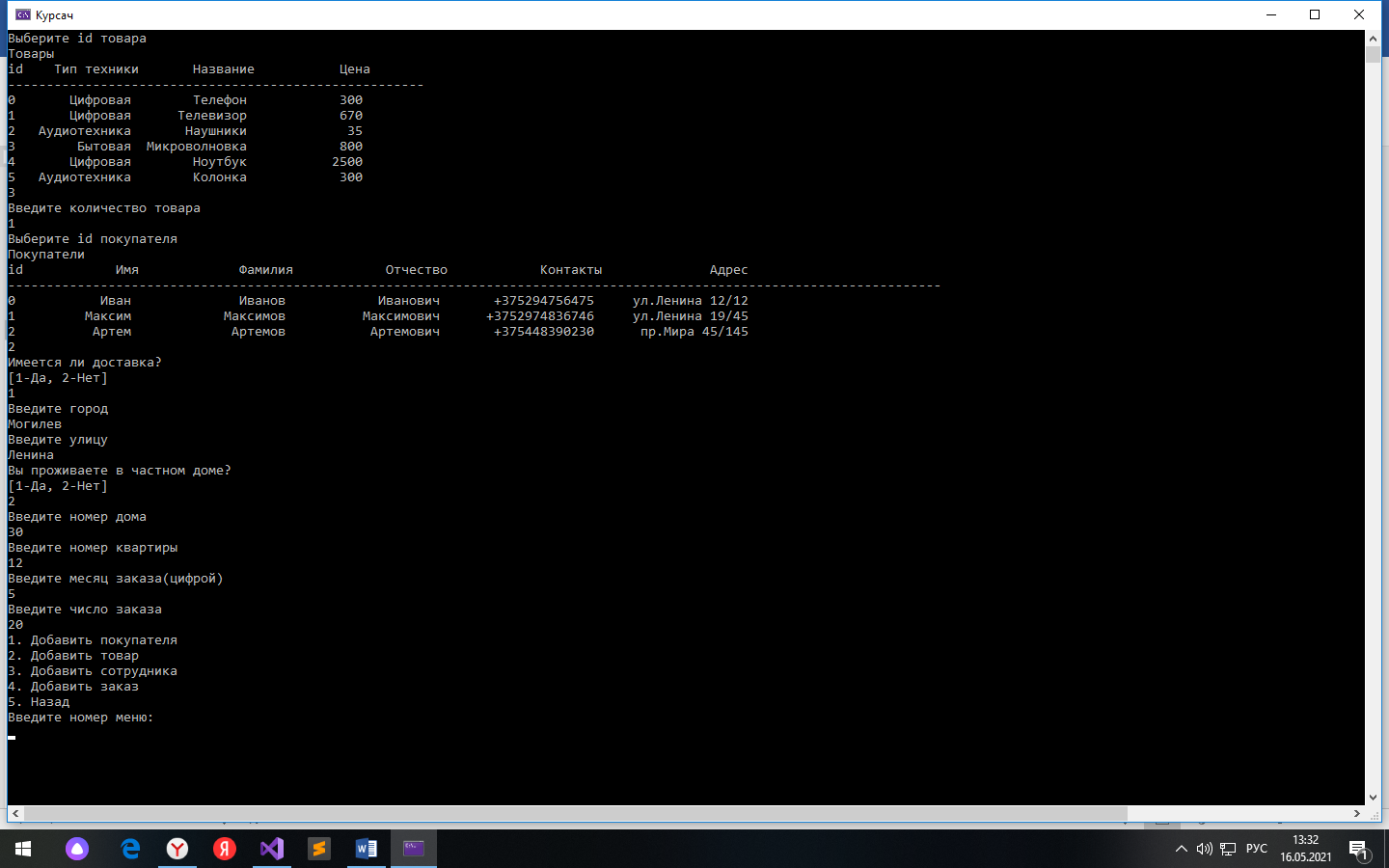


Рисунок 4.7 – Демонстрация оформления заказа с доставкой

При выборе пункта “Вывод данных” пользователь попадает в меню, в котором может выбрать что ему необходимо вывести.(рисунок 4.8)

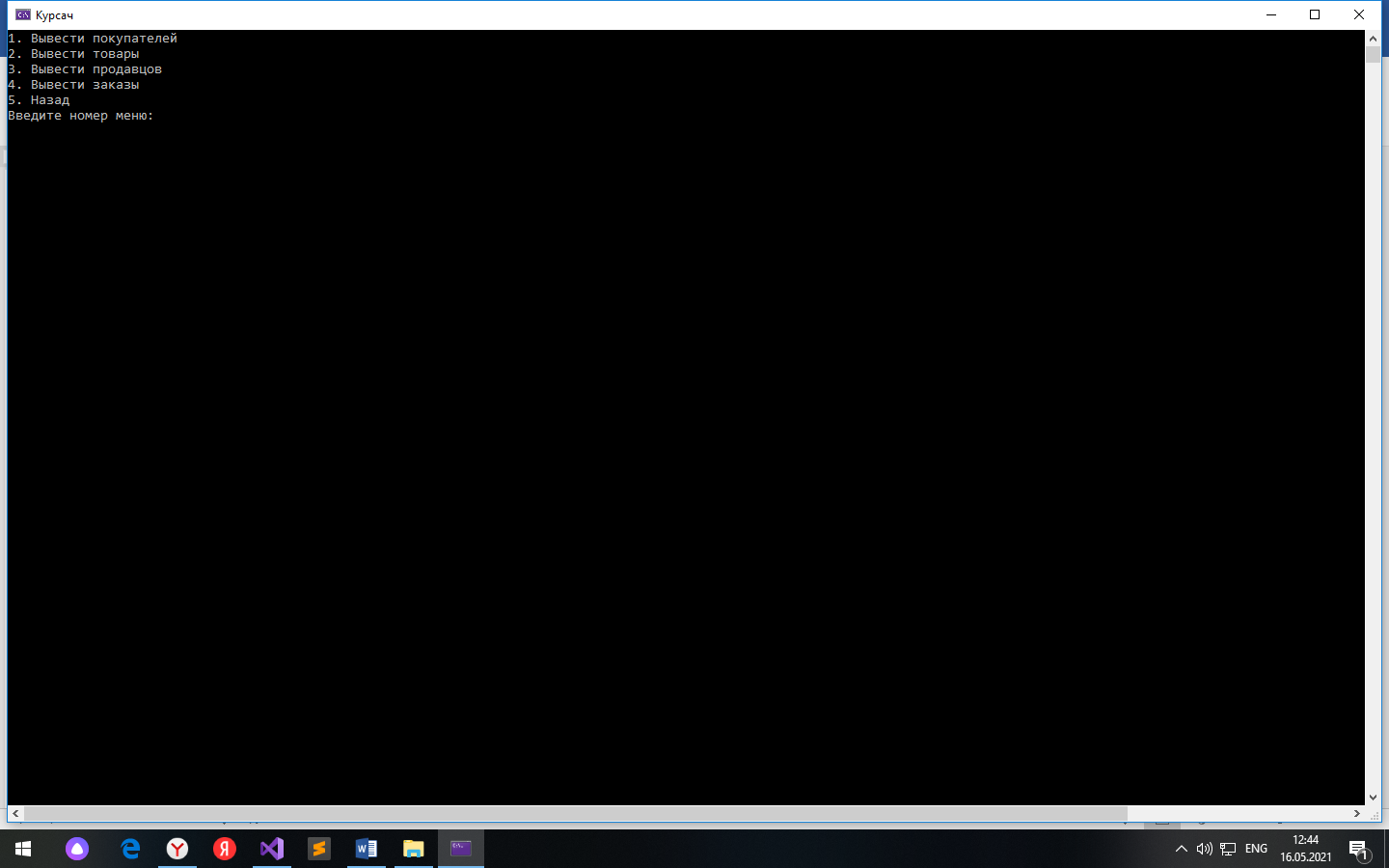


Рисунок 4.8 – Меню вывода данных

При выборе первого пункта пользователь получает данные о покупателях (рисунок 4.9)

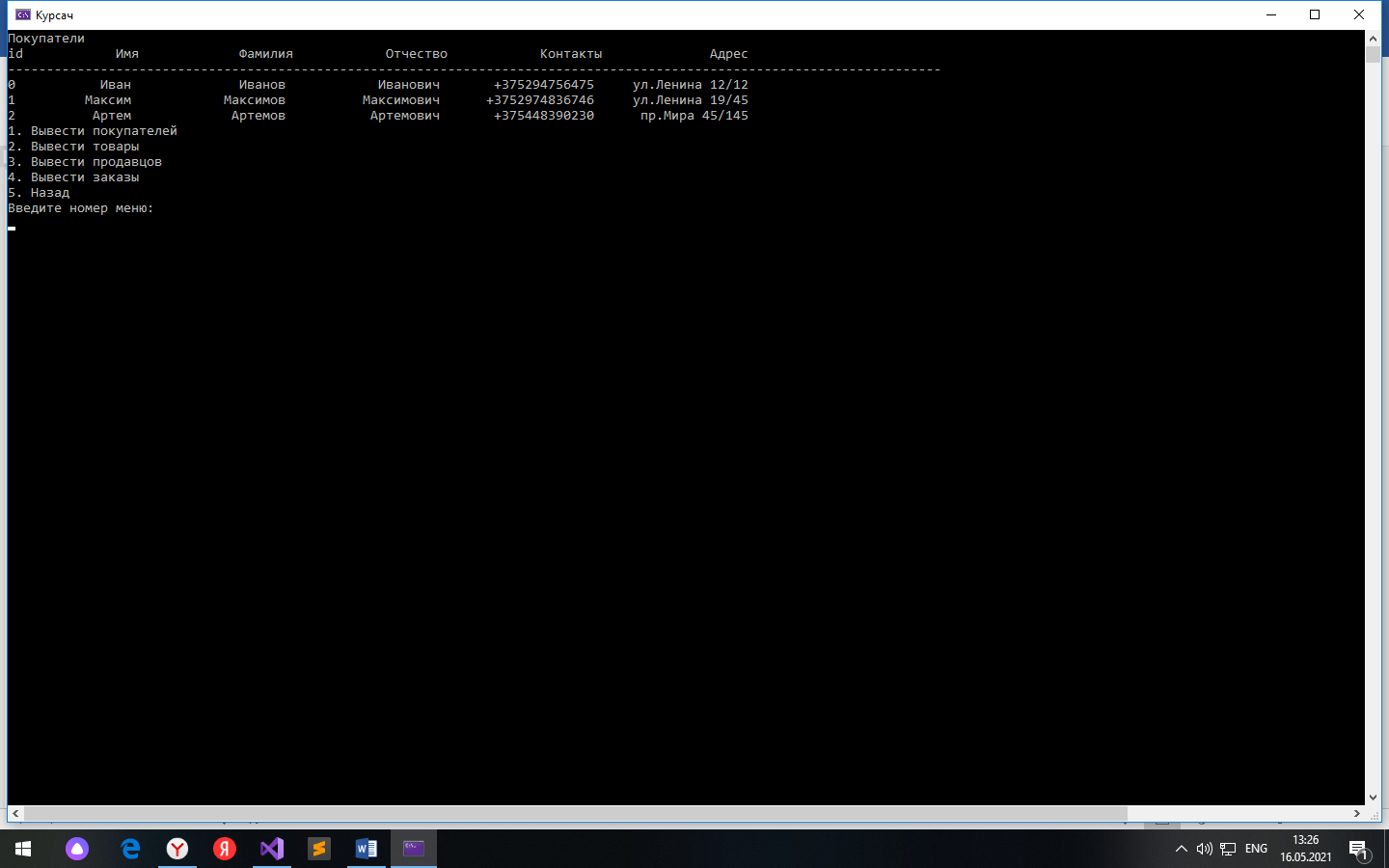


Рисунок 4.9 – Демонстрация вывода данных о покупателях

При выборе второго пункта пользователь получает данные о товарах (рисунок 4.10)

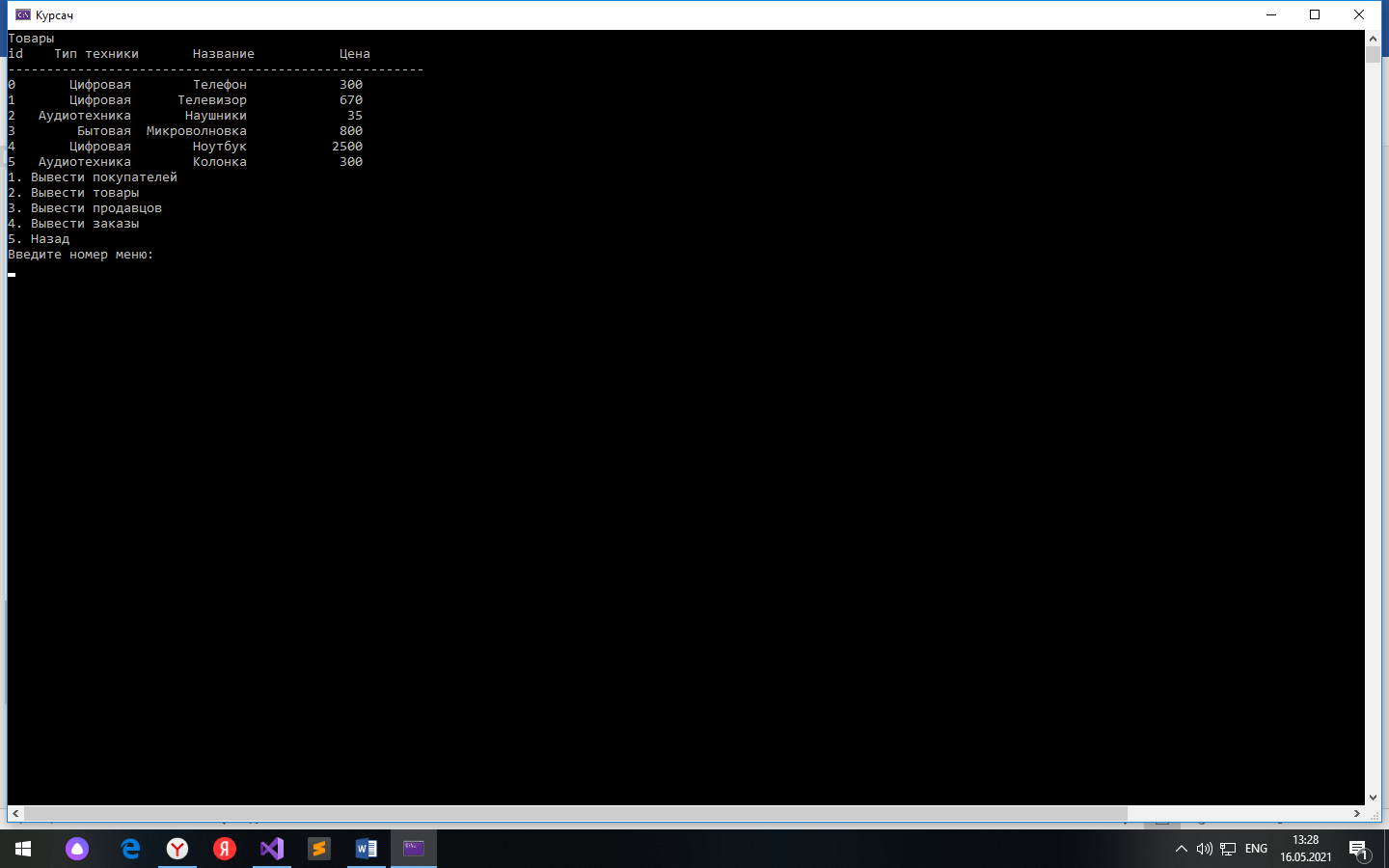


Рисунок 4.10 – Демонстрация вывода данных о товарах

При выборе третьего пункта пользователь получает данные сотрудника (рисунок 4.11)

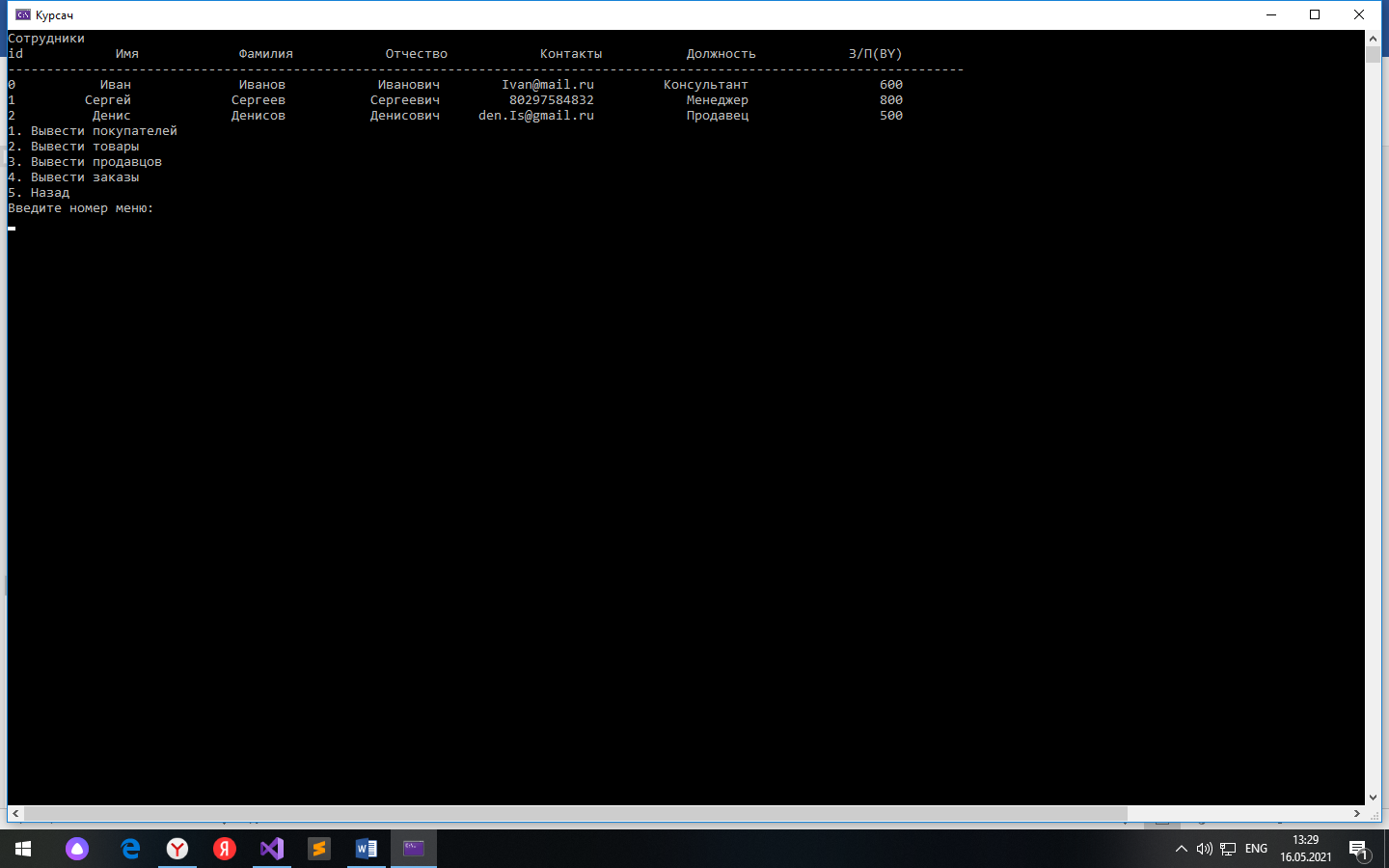


Рисунок 4.11 – Демонстрация вывода данных сотрудника

При выборе четвертого пункта пользователь получает данные о заказах (рисунок 4.12)

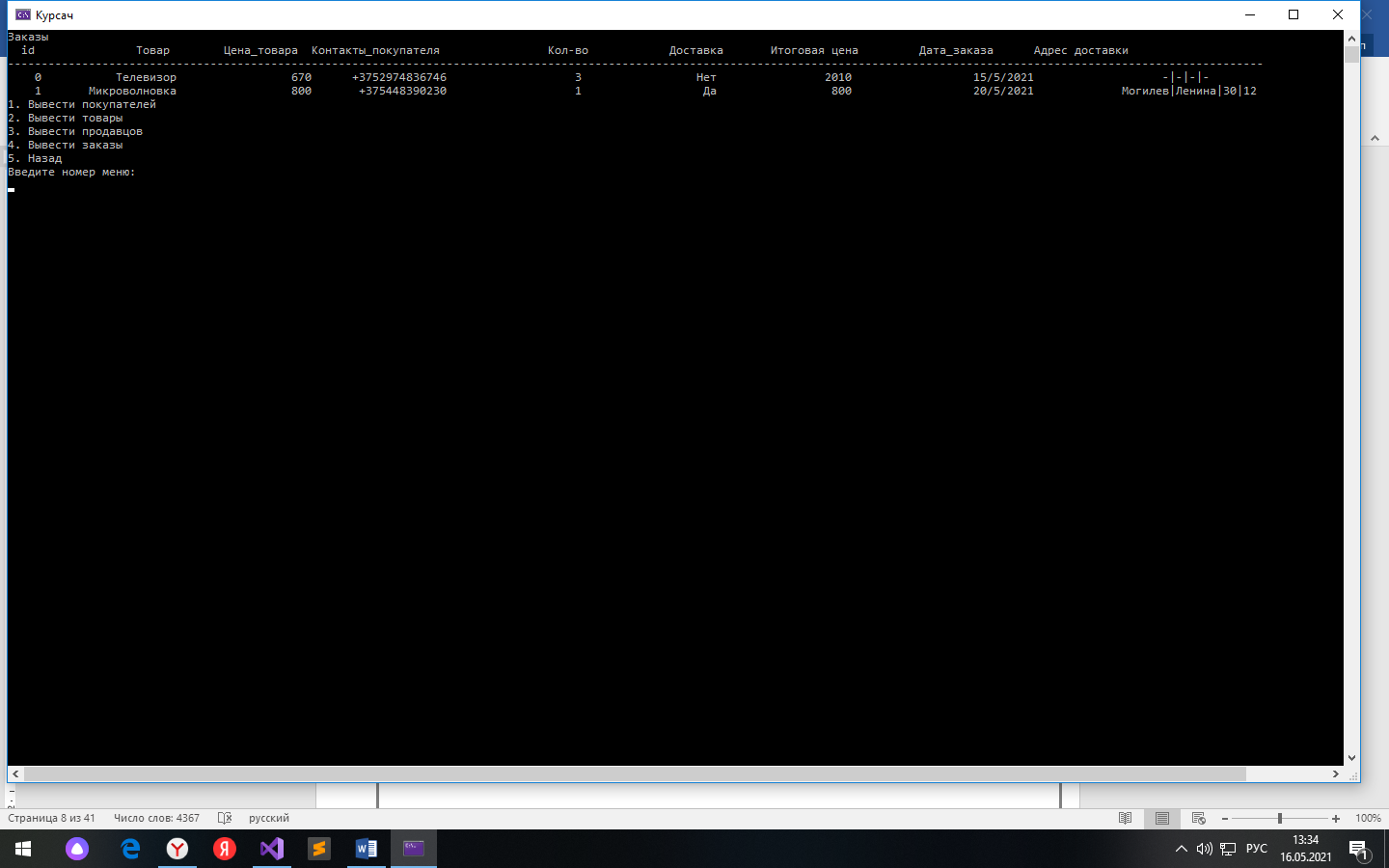


Рисунок 4.12 – Демонстрация вывода данных о заказах

При выборе пункта “Чтение из файла” программа считывает данные из файлов с названием “ Employee.dat”,“ Order.dat”, “Buyers.dat”, “Product.dat”.

При выборе пункта “Сортировка заказов по цене” пользователь попадает в подменю с выбором сортировки. (рисунок 4.13)

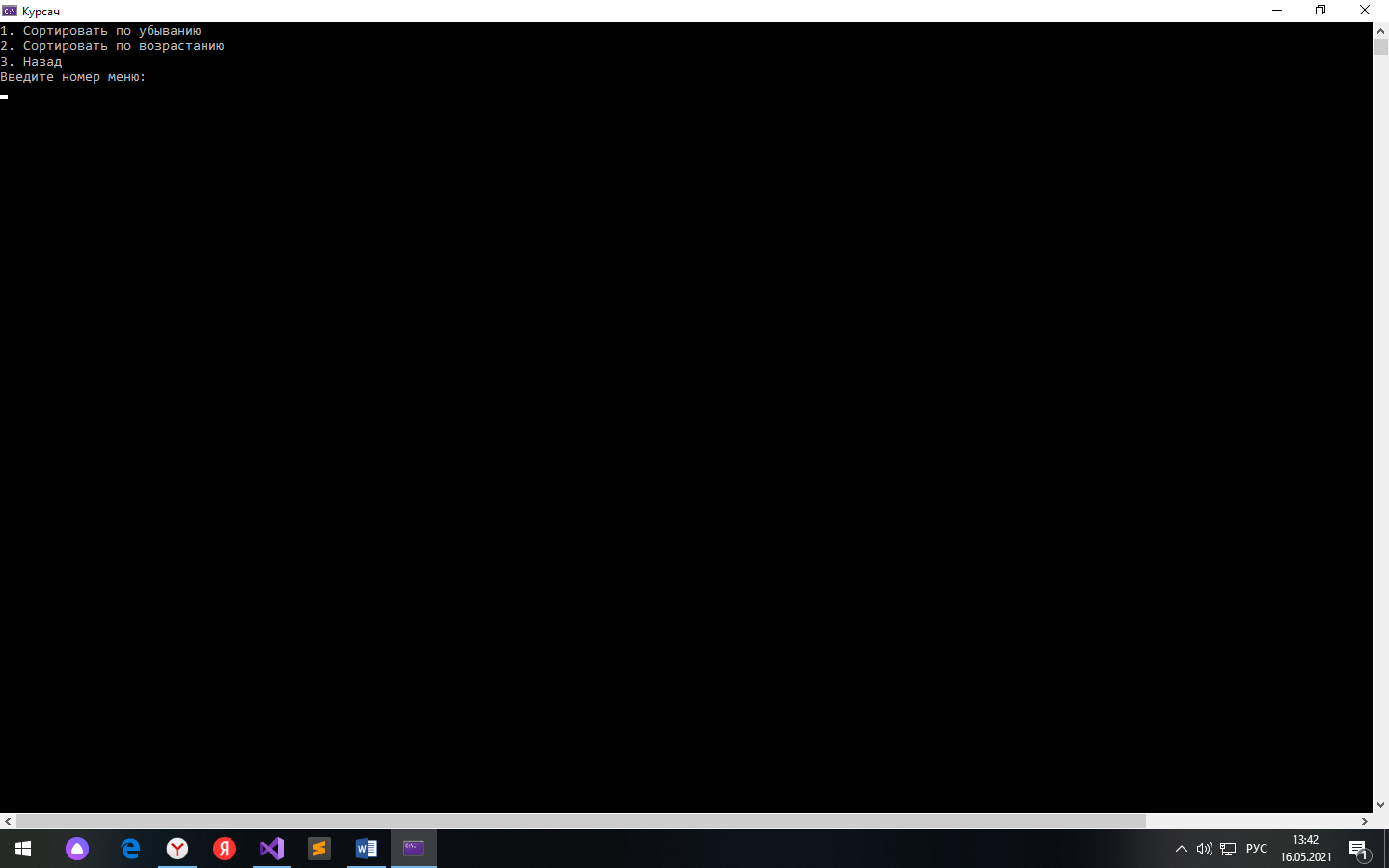


Рисунок 4.13 – Подменю пункта “Сортировка заказов по цене”

При выборе первого пункта, заказы сортируются по убыванию. (рисунок 4.14)

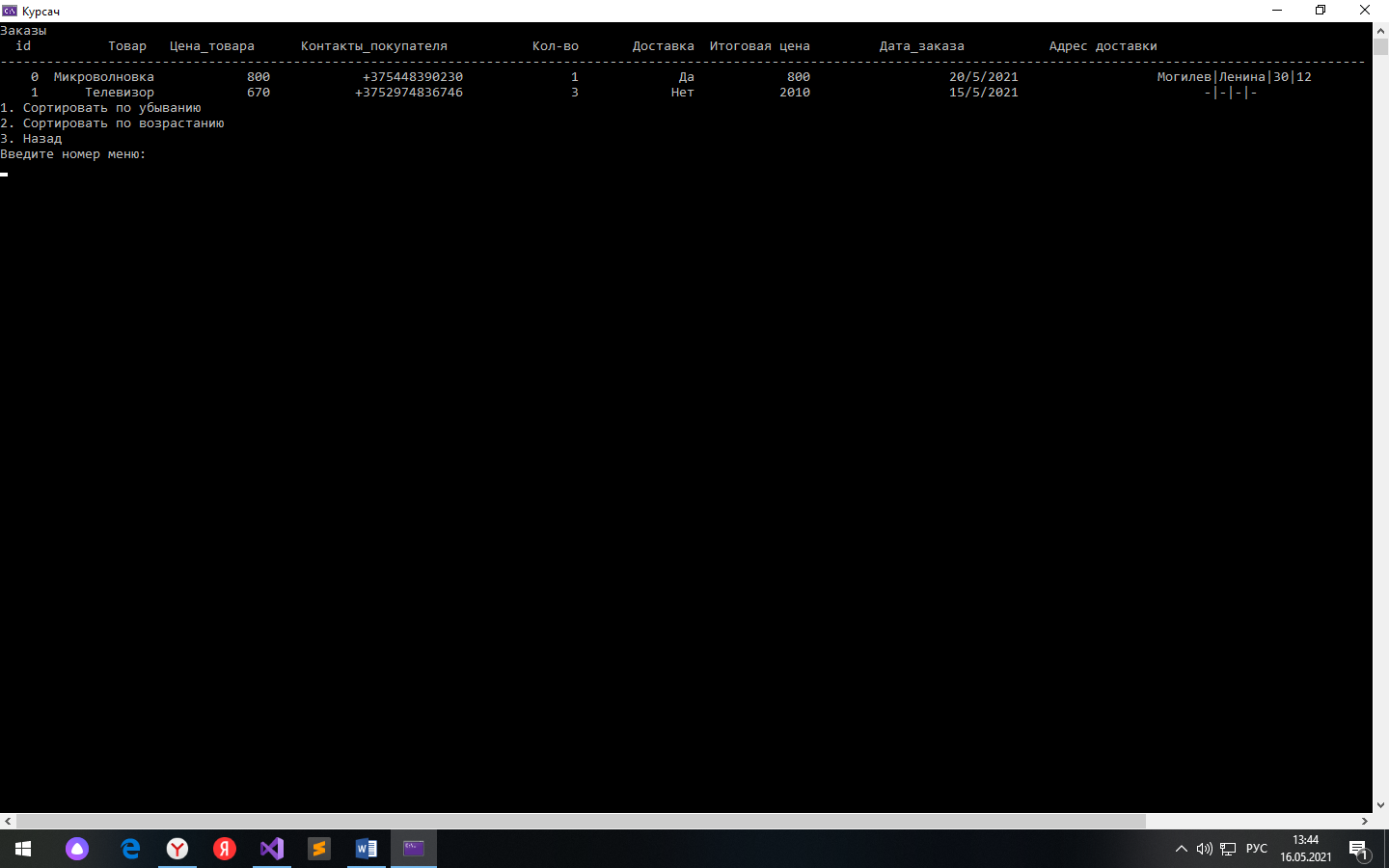


Рисунок 4.14 – Демонстрация сортировки по убыванию

При выборе второго пункта, заказы сортируются по возрастанию. (рисунок 4.15)

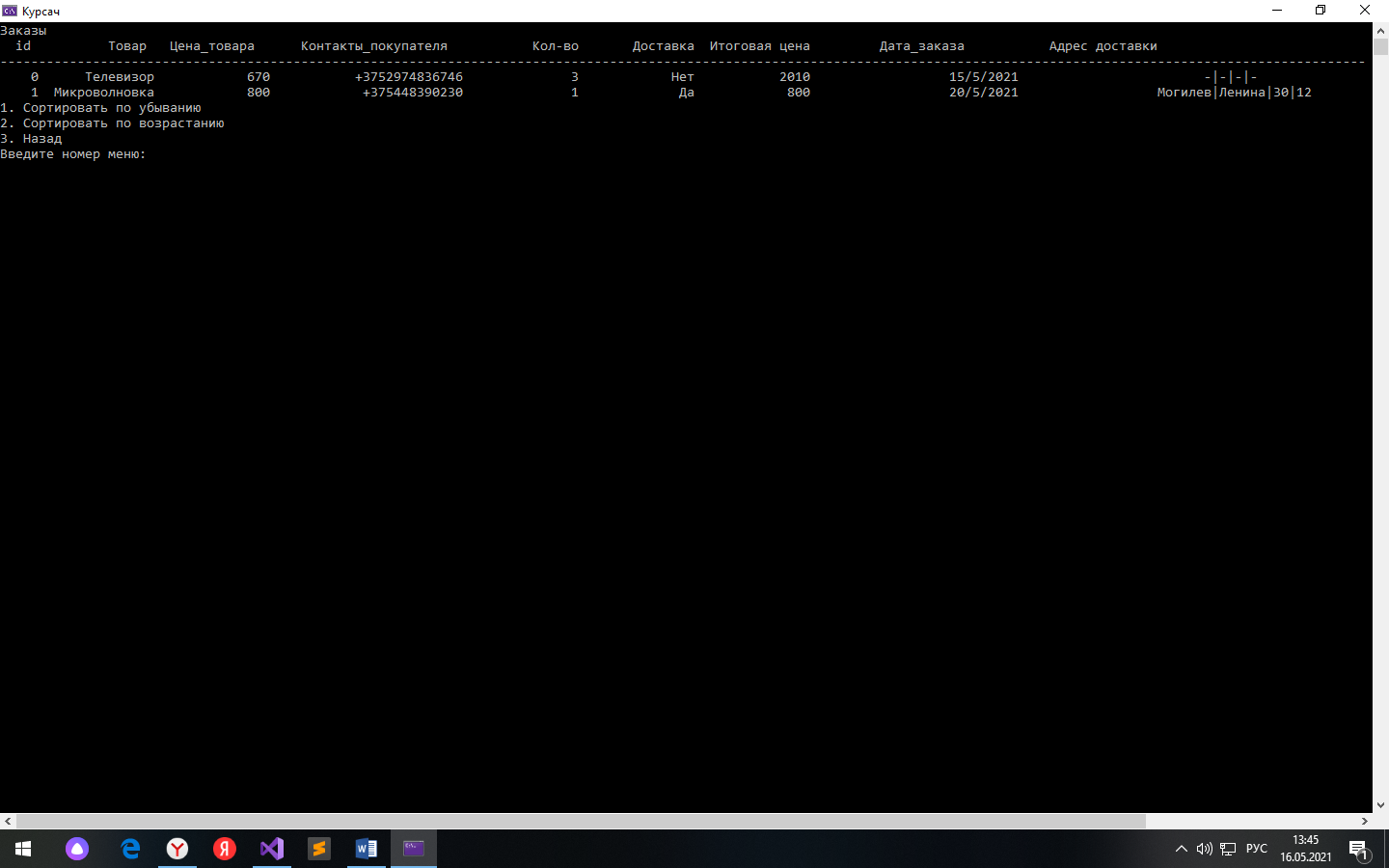


Рисунок 4.15 – Демонстрация сортировки по убыванию

При выборе пункта “Удаление данных” пользователь попадает в подменю с выбором, что он хочет удалить. (Рисунок 4.16)

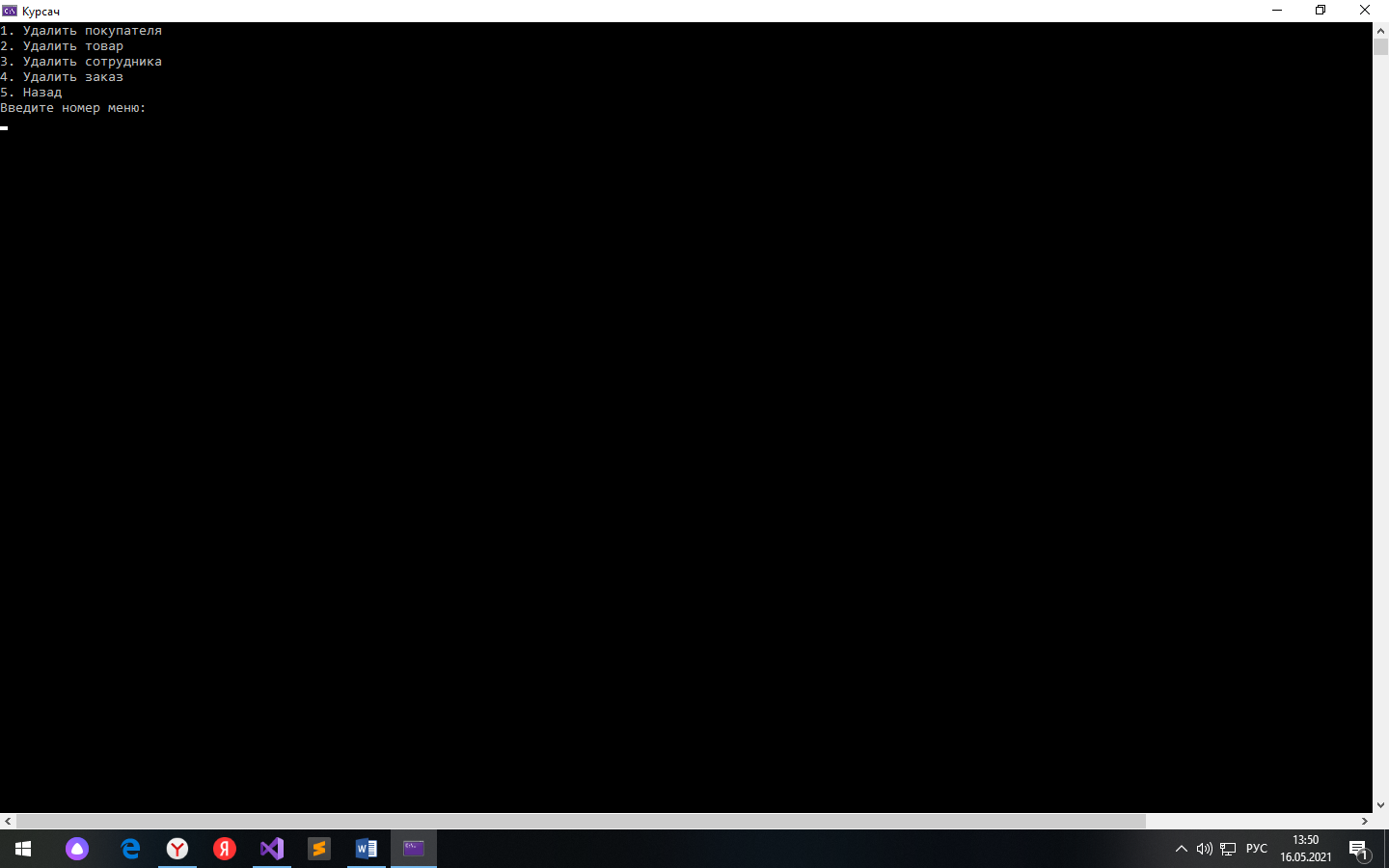


Рисунок 4.16 – Подменю пункта “Удаление данных”

При выборе первого пункта пользователь может удалить покупателя по id. (рисунок 4.17)

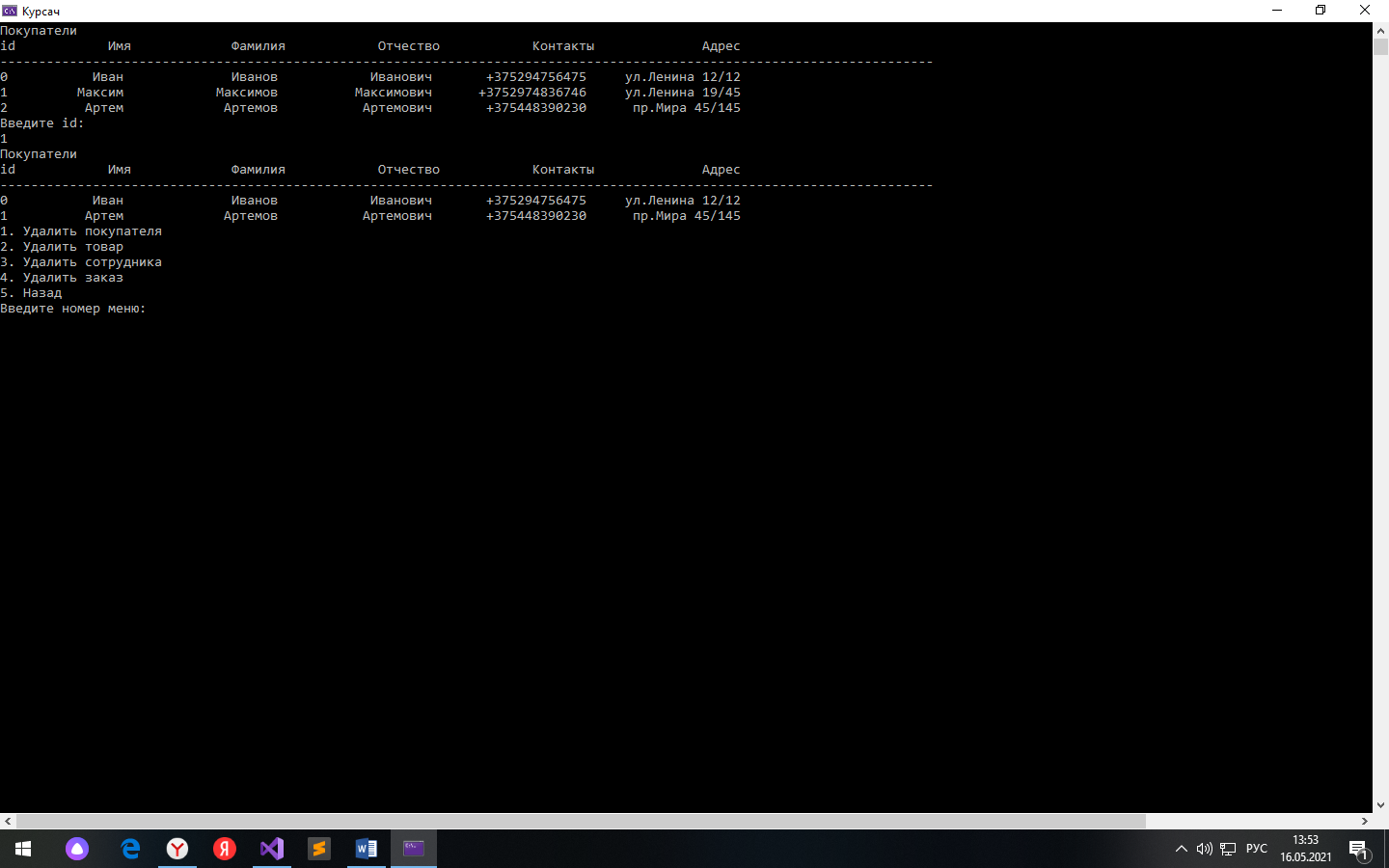


Рисунок 4.17 – Демонстрация удаления покупателя

При выборе второго пункта пользователь может удалить товар по id. (рисунок 4.18)

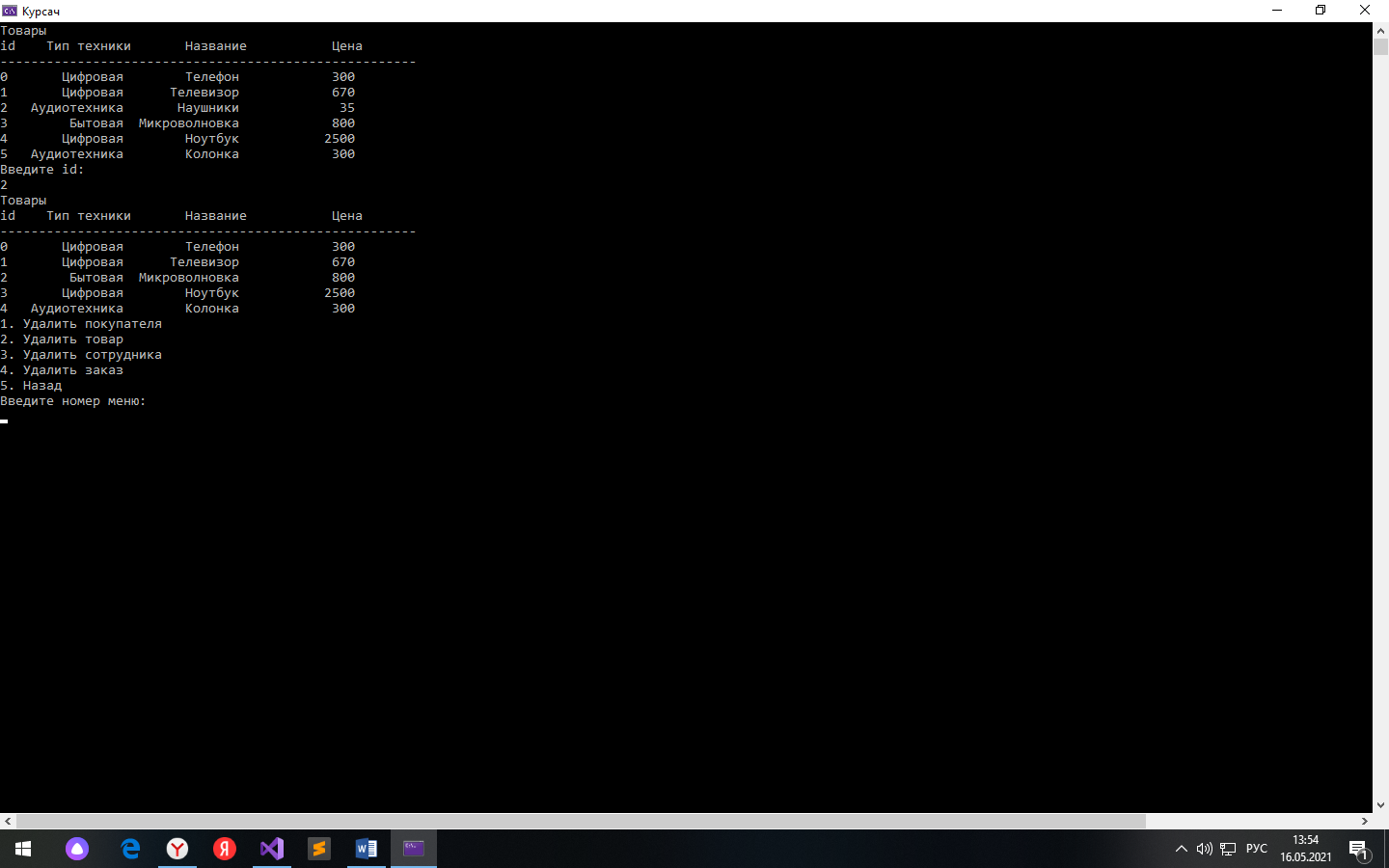


Рисунок 4.18 – Демонстрация удаления товара

При выборе третьего пункта пользователь может удалить сотрудника по id. (рисунок 4.19)

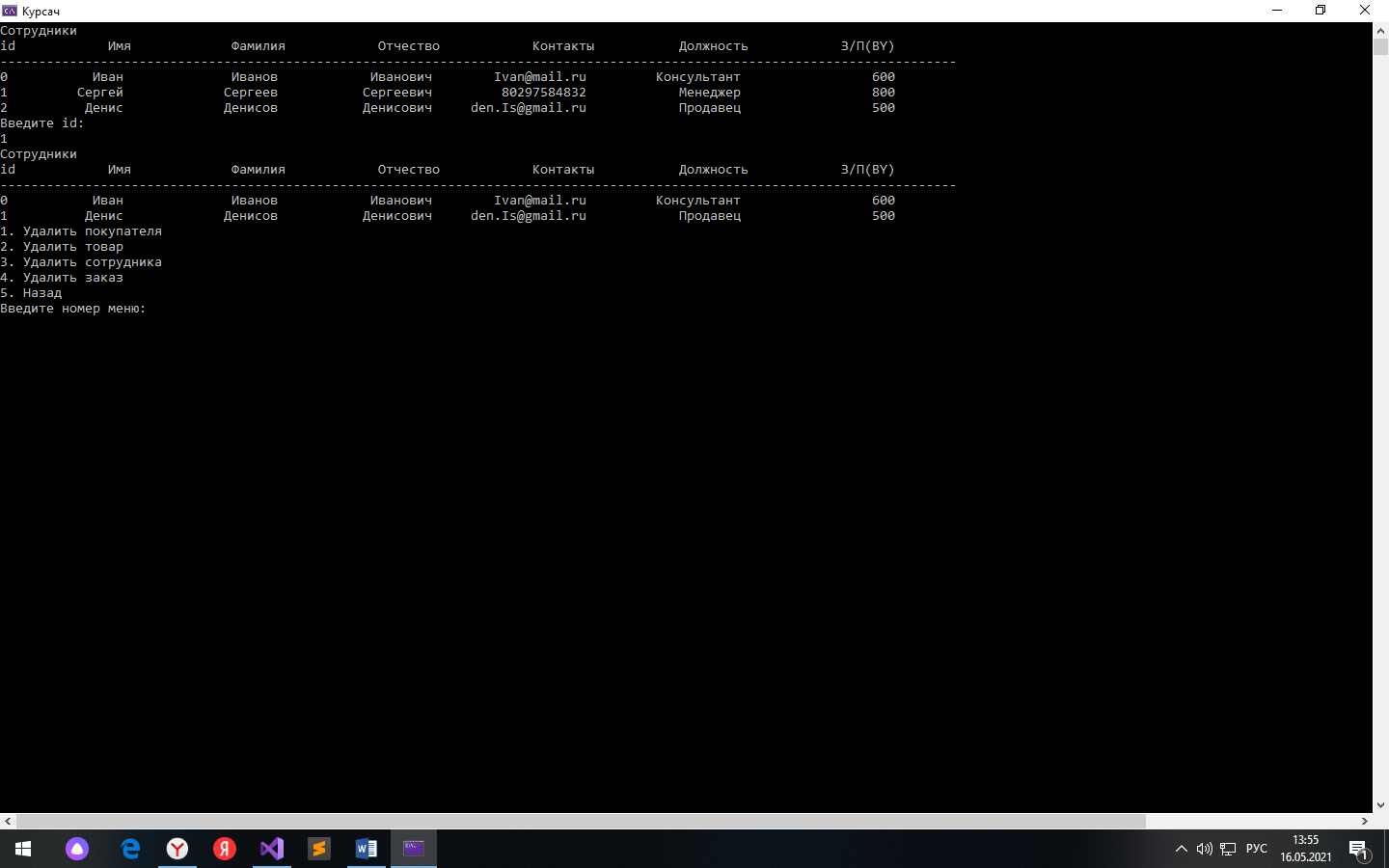


Рисунок 4.19 – Демонстрация удаления сотрудника

При выборе четвертого пункта пользователь может удалить заказ по id. (рисунок 4.20)

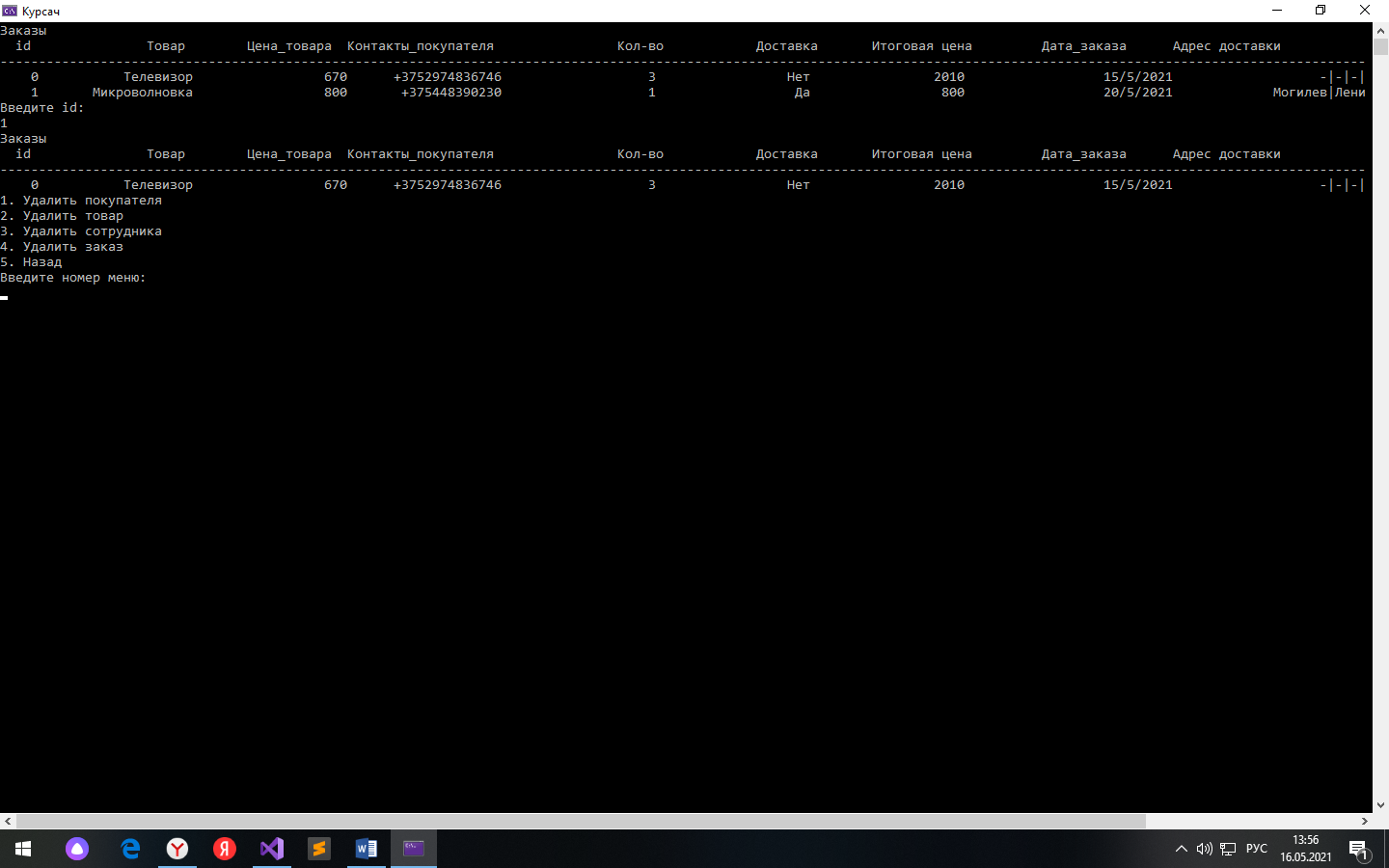


Рисунок 4.20 – Демонстрация удаления заказа

При выборе пункта “Сохранить файл” программа создаёт новые файлы с названием “ Employee.dat”,“ Order.dat”, “Buyers.dat”, “Product.dat”(Рисунок 4.21), где данные записываются в эти файлы, которые можно прочесть или использовать при новом запуске программы. Если эти файлы уже созданы, то программа перезаписывает данные в файлы.

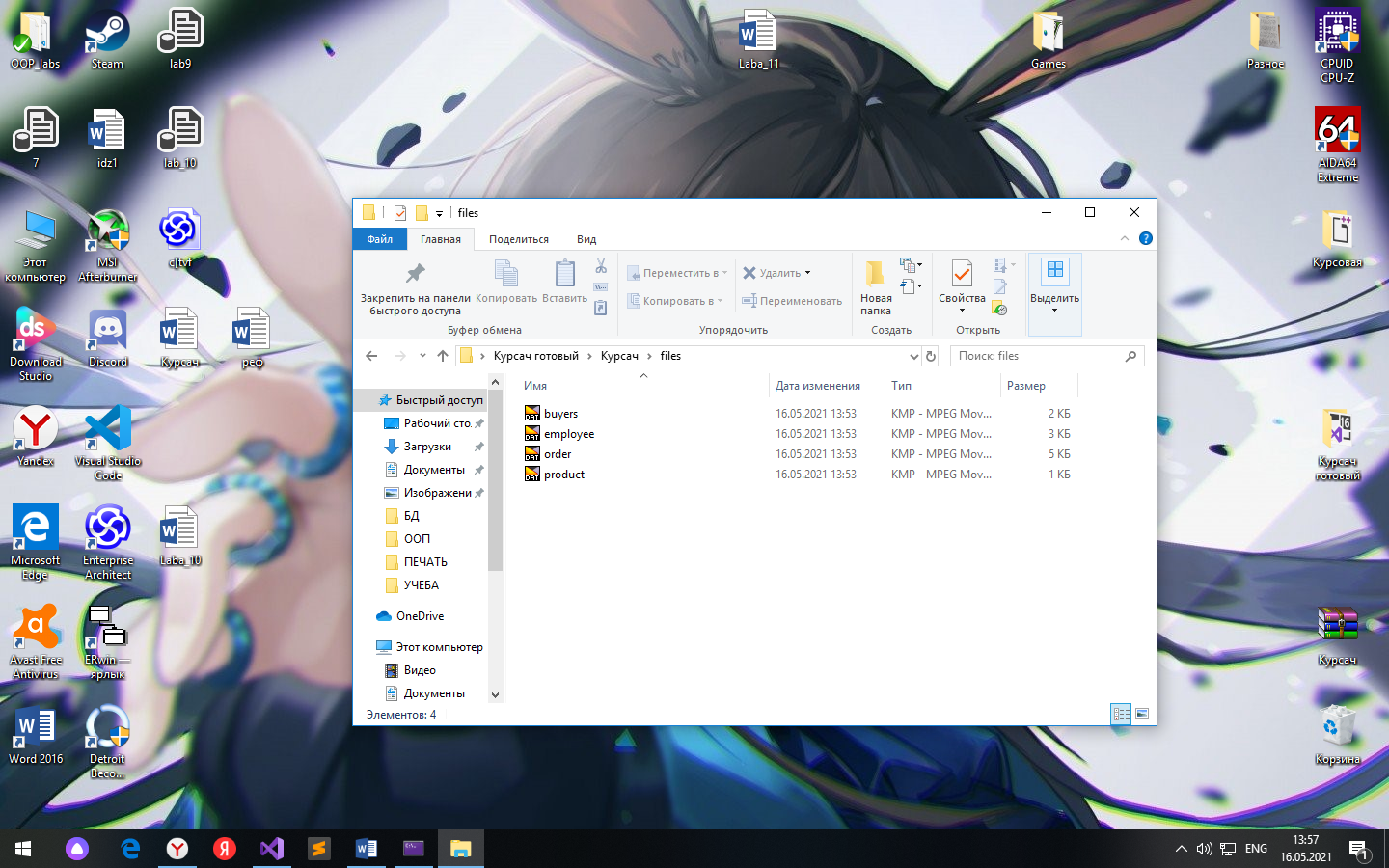


Рисунок 4.21 – Файлы из папки проекта

При выборе пункта “Выход” программа закрывается.

# Заключение

В ходе проделанной работы была разработана программа для создания базы данных магазина электроники. Корректность работы программы подтверждено тестами.

В процессе выполнения работы были закреплены знания программирования на языке С++.Также получены практические навыки. Углублены знания от постановки задачи до ее реализации.

Данная программа предназначена для работы с данными магазина электроники.

Достоинства данного программного обеспечения-это простой и понятный интерфейс для пользователя, множество проверок на корректность ввода, что исключает ошибки.

Список использованных источников

1.<https://metanit.com/>

2.<https://www.cyberforum.ru/>

3.Васильев, А.Н. Программирование на C++ в примерах и задачах / А.Н. Васильев. - М.: ЭКСМО, 2017. - 416 c.

4. Методические указания к курсовому проектированию для студентов специальности 1-40 05 01 «Информационные системы и технологии (по направлениям)» дневной и заочной форм обучения

Приложение А

Код программы

delivery.h

enum class DeliveryType

{

Да=1,

Нет=2

};

FlatOrHouse.h

enum class FlatOrHouse

{

Да=1,

Нет=2

};

ProductType.h

enum class ProductType

{

Цифровая = 1,

Бытовая = 2,

Аудиотехника=3

};

AdressDelivery.h

#pragma once

#include <iostream>

#include <string>

#include "ProductType.h"

#include "Buyers.h"

#include "delivery.h"

#include <iomanip>

#include "FlatOrHouse.h"

class AdressDelivery

{

private:

char cityName[128] = { '-' };

char streetName[128] = { '-' };

char numberOfHome[128] = { '-' };

char numberOfFlat[128] = { '-' };

char adress[128];

FlatOrHouse flatOrHouse;

public:

AdressDelivery() {

}

void SetAdressDelivery() {

cout << "Введите город" << endl;

cin.ignore();

cin.getline(cityName, 128);

cout << "Введите улицу" << endl;

cin.getline(streetName, 128);

flatOrHouse = (FlatOrHouse)GetOrderQuestion();

}

char\* GetCityName() {

return cityName;

}

char\* GetStreetName() {

return streetName;

}

char\* GetNumberOfHome() {

return numberOfHome;

}

char\* GetNumberOfFlat() {

return numberOfFlat;

}

int GetOrderQuestion()

{

bool isNumber = false;

string questionString;

int question;

while (!isNumber)

{

cout << "Вы проживаете в частном доме?\n[1-Да, 2-Нет]\n";

cin >> questionString;

try

{

question = stoi(questionString);

if (question >= 1 && question <= 2)

{

isNumber = true;

}

else

{

cout << "Error" << endl;

isNumber = false;

continue;

}

if (question == 1)

{

cout << "Введите номер дома" << endl;

cin >> numberOfHome;

if (!IsNumber(numberOfHome))

{

cout << "Error" << endl;

isNumber = false;

continue;

}

}

else

{

cout << "Введите номер дома" << endl;

cin >> numberOfHome;

if (!IsNumber(numberOfHome))

{

cout << "Error" << endl;

isNumber = false;

continue;

}

cout << "Введите номер квартиры" << endl;

cin >> numberOfFlat;

if (!IsNumber(numberOfFlat))

{

cout << "Error" << endl;

isNumber = false;

continue;

}

}

}

catch (...)

{

cout << "Error" << endl;

isNumber = false;

}

}

return question;

}

bool IsNumber(string text)

{

try

{

int num = stoi(text);

return true;

}

catch(...)

{

return false;

}

}

};

Buyers.h

#pragma once

#include <iostream>

#include <string>

#include "Human.h"

#include <iomanip>

using namespace std;

class Buyers : public Human

{

public:

Buyers() {

}

void SetBuyersConsole()

{

cout << "Введите имя покупателя\n";

cin.ignore();

cin.getline(firstName, 128);

cout << "Введите фамилию покупателя\n";

cin.getline(secondName, 128);

cout << "Введите отчество покупателя \n";

cin.getline(thirdName, 128);

cout << "Введите телефон покупателя\n";

cin.getline(contacts, 128);

cout << "Введите адрес покупателя\n";

cin.getline(adress, 128);

}

char\* GetFirstName()

{

return firstName;

}

char\* GetSecondName()

{

return secondName;

}

char\* GetThirdName()

{

return thirdName;

}

char\* GetContacts()

{

return contacts;

}

char\* GetAdress()

{

return adress;

}

};

DateOrder.h

#pragma once

#include <iostream>

#include <string>

#include "ProductType.h"

#include "Buyers.h"

#include "delivery.h"

#include <iomanip>

class DateOrder

{

private:

int dayOfDate;

int monthOfDate;

int yearOfDate = 2021;

public:

DateOrder() {}

void SetDateOrder() {

monthOfDate = GetInput1();

dayOfDate = GetInput();

}

int GetDayOfDate() {

return dayOfDate;

}

int GetInput() {

string dayString;

int day;

bool isCorrectNumber = false;

while (!isCorrectNumber)

{

cout << "Введите число заказа \n";

cin >> dayString;

try {

day = stoi(dayString);

if (monthOfDate == 2) {

if (day <= 0 || day>28)

throw 0;

}

else if (day <= 0||day>31) {

throw 0;

}

}

catch (...) {

cout << "ERROR\n";

isCorrectNumber = false;

continue;

}

isCorrectNumber = true;

}

return day;

}

int GetMonthOfDate() {

return monthOfDate;

}

int GetInput1() {

string monthString;

int month;

bool isCorrectNumber = false;

while (!isCorrectNumber)

{

cout << "Введите месяц заказа(цифрой) \n";

cin >> monthString;

try {

month = stoi(monthString);

if (month <= 0 || month>12) {

throw 0;

}

}

catch (...) {

cout << "ERROR\n";

isCorrectNumber = false;

continue;

}

isCorrectNumber = true;

}

return month;

}

int GetYear() {

return yearOfDate;

}

};

Employee.h

#pragma once

#include <iostream>

#include <string>

#include "Human.h"

#include "Product.h"

using namespace std;

class Employee : public Human

{

private:

static int id;

int employeeId;

char post[128];

int wages;

public:

Employee() {}

void SetEmployeeConsole() {

cout << "Введите имя сотрудника\n";

cin.ignore();

cin.getline(firstName, 128);

cout << "Введите фамилию сотрудника\n";

cin.getline(secondName, 128);

cout << "Введите отчество сотрудника\n";

cin.getline(thirdName, 128);

cout << "Введите контакты сотрудника\n";

cin.getline(contacts, 128);

cout << "Введите должность сотрудника\n";

cin.getline(post, 128);

wages = GetInput();

}

int GetWages() {

return wages;

}

int GetInput() {

string wagesString;

int wages;

bool isCorrectNumber = false;

while (!isCorrectNumber)

{

cout << "Введите З/П(BY)\n";

cin >> wagesString;

try {

wages = stoi(wagesString);

if (wages < 0) {

throw 0;

}

}

catch (...) {

cout << "ERROR\n";

isCorrectNumber = false;

continue;

}

isCorrectNumber = true;

}

return wages;

}

int GetEmployeeId() {

return employeeId;

}

char\* GetFirstName() {

return firstName;

}

char\* GetSecondName() {

return secondName;

}

char\* GetThirdName() {

return thirdName;

}

char\* GetContacts() {

return contacts;

}

char\* GetPost() {

return post;

}

};

File.h

#pragma once

#include <iostream>

#include <string>

#include "Storage.h"

#include <fstream>

#include "Employee.h"

#include "Buyers.h"

#include "Product.h"

#include "Order.h"

using namespace std;

static class File

{

private:

template<typename T1, typename T2>

static void Write(string file, T1 object, T2 collection) {

ofstream out(file, ofstream::out);

if (out.is\_open())

{

for (const T1& object : collection)

{

out.write((char\*)&object, sizeof(T1));

}

}

out.close();

}

public:

static void WriteFiles() {

Employee employee;

Product product;

Buyers buyers;

Order order;

Write("files\\employee.dat", employee, Storage::employee);

Write("files\\product.dat", product, Storage::product);

Write("files\\buyers.dat", buyers, Storage::buyers);

Write("files\\order.dat", order, Storage::order);

}

static void ReadFiles() {

Storage::product.clear();

Storage::employee.clear();

Storage::order.clear();

Storage::buyers.clear();

ifstream in("files\\employee.dat", ios::app);

if (in.is\_open())

{

Employee employee;

while (in.read((char\*)&employee, sizeof(Employee))) {

Storage::employee.push\_back(employee);

}

}

in.close();

in.open("files\\product.dat", ios::app);

if (in.is\_open())

{

Product product;

while (in.read((char\*)&product, sizeof(Product))) {

Storage::product.push\_back(product);

}

}

in.close();

in.open("files\\buyers.dat", ios::app);

if (in.is\_open())

{

Buyers buyers;

while (in.read((char\*)&buyers, sizeof(Buyers))) {

Storage::buyers.push\_back(buyers);

}

}

in.close();

in.open("files\\order.dat", ios::app);

if (in.is\_open())

{

Order order;

while (in.read((char\*)&order, sizeof(Order))) {

Storage::order.push\_back(order);

}

}

in.close();

}

};

HelperOrderStorage.h

#pragma once

#include <iostream>

#include <string>

#include "Product.h"

#include "Employee.h"

#include "Buyers.h"

#include "Storage.h"

#include "DateOrder.h"

#include "ViewData.h"

using namespace std;

class HelperOrderStorage

{

private:

ViewData view;

public:

void ExecuteSetOrderConsole(Order& reception) {

int numberProduct;

int numberBuyers;

cout << "Выберите id товара\n";

view.ShowProductsFromApplicationStorage();

cin >> numberProduct;

cout << "Введите количество товара" << endl;

quantity = GetQuantity();

cout << "Выберите id покупателя\n";

view.ShowBuyersFromApplicationStorage();

cin >> numberBuyers;

reception.SetOrderConsole(Storage::product.at(numberProduct), Storage::buyers.at(numberBuyers), quantity);

}

int quantity;

int GetQuantity() {

string str;

int num;

bool isCorrectNumber = false;

while (!isCorrectNumber)

{

cin >> str;

try

{

num = stoi(str);

}

catch (...)

{

cout << "Error" << endl;

isCorrectNumber = false;

continue;

}

isCorrectNumber = true;

}

return num;

}

};

Human.h

#pragma once

#include <string.h>

#include <iostream>

using namespace std;

class Human

{

protected:

char firstName[128];

char secondName[128];

char thirdName[128];

char contacts[128];

char adress[128];

};

Menu.h

#pragma once

#include <iostream>

#include <string>

#include "Product.h"

#include "Storage.h"

#include "HelperOrderStorage.h"

#include "File.h"

#include "SortByProductPrice.h"

#include "ViewData.h"

#include "StorageService.h"

using namespace std;

class Menu

{

private:

ViewData viewdata;

StorageService storageService;

SortByProductName sort;

public:

void MainMenu() {

while (true)

{

cout << "1. Добавить данные " << endl;

cout << "2. Вывод данных" << endl;

cout << "3. Чтение из файла" << endl;

cout << "4. Сортировка заказов по цене" << endl;

cout << "5. Удаление данных" << endl;

cout << "6. Сохранить файл" << endl;

cout << "7. Выход" << endl;

int menuNumber = GetInput();

system("cls");

switch (menuNumber)

{

case 1: {

AddMenu();

break;

}

case 2: {

PrintMenu();

break;

}

case 3: {

cout << "Файл считан" << endl;

File::ReadFiles();

break;

}

case 4: {

SortMenu();

break;

}

case 5: {

DeleteMenu();

break;

}

case 6: {

cout << "Файл сохранен" << endl;

File::WriteFiles();

break;

}

case 7: {

return;

}

default:

cout << "ERROR\n";

break;

}

}

}

private:

void AddMenu() {

while (true) {

cout << "1. Добавить покупателя \n" << "2. Добавить товар\n" << "3. Добавить сотрудника\n" << "4. Добавить заказ\n" << "5. Назад\n";

int menuNumber = GetInput();

system("cls");

switch (menuNumber)

{

case 1: {

Buyers buyers = Buyers();

buyers.SetBuyersConsole();

Storage::buyers.push\_back(buyers);

break;

}

case 2: {

Product product = Product();

product.SetProductConsole();

Storage::product.push\_back(product);

break;

}

case 3: {

Employee employee = Employee();

employee.SetEmployeeConsole();

Storage::employee.push\_back(employee);

break;

}

case 4: {

Order reception = Order();

HelperOrderStorage helperReceptionStorage = HelperOrderStorage();

helperReceptionStorage.ExecuteSetOrderConsole(reception);

Storage::order.push\_back(reception);

break;

}

case 5: {

return;

}

default:

cout << "ERROR.Введите номер из меню\n\n";

break;

}

}

}

void PrintMenu() {

bool isOpenMenu = true;

while (isOpenMenu) {

cout << "1. Вывести покупателей" << endl;

cout << "2. Вывести товары" << endl;

cout << "3. Вывести продавцов" << endl;

cout << "4. Вывести заказы" << endl;

cout << "5. Назад" << endl;

int menuNumber = GetInput();

system("cls");

switch (menuNumber)

{

case 1: {

viewdata.ShowBuyersFromApplicationStorage();

break;

}

case 2: {

viewdata.ShowProductsFromApplicationStorage();

break;

}

case 3: {

viewdata.ShowEmployeesFromApplicationStorage();

break;

}

case 4: {

viewdata.ShowOrdersFromApplicationStorage();

break;

}

case 5: {

return;

}

default:

cout << "ERROR\n";

break;

}

}

}

void SortMenu() {

bool isOpenMenu = true;

while (isOpenMenu) {

cout << "1. Сортировать по убыванию" << endl;

cout << "2. Сортировать по возрастанию" << endl;

cout << "3. Назад" << endl;

int menuNumber = GetInput();

system("cls");

switch (menuNumber)

{

case 1: {

sort.SortByDescending();

sort.ShowSort();

break;

}

case 2: {

sort.SortByAscending();

sort.ShowSort();

break;

}

case 3: {

return;

}

default:

cout << "ERROR\n";

break;

}

}

}

void DeleteMenu() {

bool isOpenMenu = true;

while (isOpenMenu) {

cout << "1. Удалить покупателя" << endl;

cout << "2. Удалить товар" << endl;

cout << "3. Удалить сотрудника" << endl;

cout << "4. Удалить заказ" << endl;

cout << "5. Назад" << endl;

int menuNumber = GetInput();

system("cls");

switch (menuNumber)

{

case 1: {

viewdata.ShowBuyersFromApplicationStorage();

storageService.DeleteBuyers();

viewdata.ShowBuyersFromApplicationStorage();

break;

}

case 2: {

viewdata.ShowProductsFromApplicationStorage();

storageService.DeleteProduct();

viewdata.ShowProductsFromApplicationStorage();

break;

}

case 3: {

viewdata.ShowEmployeesFromApplicationStorage();

storageService.DeleteEmployee();

viewdata.ShowEmployeesFromApplicationStorage();

break;

}

case 4: {

viewdata.ShowOrdersFromApplicationStorage();

storageService.DeleteOrder();

viewdata.ShowOrdersFromApplicationStorage();

break;

}

case 5: {

return;

}

default:

cout << "ERROR\n";

break;

}

}

}

int GetInput() {

string menuNumberString;

int menuNumber;

bool isCorrectNumber = false;

while (!isCorrectNumber)

{

cout << "Введите номер меню:\n";

cin >> menuNumberString;

try {

menuNumber = stoi(menuNumberString);

}

catch (...) {

cout << "ERROR.Введите номер из меню\n";

isCorrectNumber = false;

continue;

}

isCorrectNumber = true;

}

return menuNumber;

}

};

Order.h

#pragma once

#include <iostream>

#include <string>

#include "Product.h"

#include "Employee.h"

#include "Buyers.h"

#include "delivery.h"

#include "DateOrder.h"

#include "AdressDelivery.h"

using namespace std;

class Order

{

private:

Employee employee;

Product product;

Buyers buyers;

DeliveryType deliveryType;

int quantity;

int fullPrice;

DateOrder date;

AdressDelivery adress;

public:

Order()

{

product = Product();

buyers = Buyers();

//deliveryType = (DeliveryType)0;

date = DateOrder();

}

void SetOrderConsole(Product product, Buyers buyers, int quantity)

{

this->product = product;

this->buyers = buyers;

this->quantity = quantity;

this->date = date;

CalculatePrice();

GetOrderQuestion();

date.SetDateOrder();

}

void SetQuantity(int quantity)

{

this->quantity = quantity;

}

void CalculatePrice()

{

fullPrice = product.GetPrice() \* quantity;

}

DeliveryType GetDeliveryType()

{

return deliveryType;

}

string GetNumberOfFlat()

{

return adress.GetNumberOfFlat();

}

string GetNumberOfHome()

{

return adress.GetNumberOfHome();

}

string GetStreetName()

{

return adress.GetStreetName();

}

string GetCityName()

{

return adress.GetCityName();

}

int GetYear()

{

return date.GetYear();

}

int GetMonthOfDate()

{

return date.GetMonthOfDate();

}

int GetDayOfDate()

{

return date.GetDayOfDate();

}

int GetFullPrice()

{

return fullPrice;

}

int GetQuantity()

{

return quantity;

}

int GetProductPrice()

{

return product.GetPrice();

}

string GetNameProduct()

{

return product.GetNameProduct();

}

string GetContacts()

{

return buyers.GetContacts();

}

int GetOrderQuestion()

{

bool isNumber = false;

string questionString;

int question;

while (!isNumber)

{

cout << "Имеется ли доставка?\n[1-Да, 2-Нет]\n";

cin >> questionString;

try

{

question = stoi(questionString);

if (question >= 1 && question <= 2)

{

deliveryType = (DeliveryType)question;

isNumber = true;

}

else

{

cout << "Error" << endl;

continue;

}

if (question == 1)

{

adress.SetAdressDelivery();

}

else

{

adress = AdressDelivery();

}

}

catch (...)

{

cout << "Error" << endl;

}

}

return question;

}

};

Product.h

#pragma once

#include <iostream>

#include <string>

#include "ProductType.h"

#include "Buyers.h"

#include "delivery.h"

#include <iomanip>

using namespace std;

class Product

{

private:

ProductType productType;

int price;

char nameProduct[128];

int quantity;

public:

Product()

{

productType = (ProductType)0;

}

void SetProductConsole()

{

productType = (ProductType)GetInputProductType();

cout << "Введите название товара\n";

cin >> nameProduct;

price = GetInput();

}

ProductType GetProductType()

{

return productType;

}

char\* GetNameProduct()

{

return nameProduct;

}

int GetPrice()

{

return price;

}

int GetInput()

{

string priceString;

int price;

bool isCorrectNumber = false;

while (!isCorrectNumber)

{

cout << "Введите цену товара(BY)\n";

cin >> priceString;

try

{

price = stoi(priceString);

if (price < 0)

{

throw 0;

}

}

catch (...)

{

cout << "ERROR\n";

isCorrectNumber = false;

continue;

}

isCorrectNumber = true;

}

return price;

}

int GetInputProductType()

{

bool isNumber = false;

string numberString;

int number;

while (!isNumber)

{

cout << "Введите тип техники\n[1-Цифровая, 2-Бытовая,3-Аудиотехника]\n";

cin >> numberString;

try

{

number = stoi(numberString);

if (number >= 1 && number <= 3)

{

isNumber = true;

}

else

{

cout << "Error" << endl;

}

}

catch (...)

{

cout << "Error" << endl;

}

}

return number;

}

};

SortByProductName.h

#pragma once

#include <vector>

#include <iostream>

#include <string>

#include "Product.h"

#include "Order.h"

#include "Employee.h"

#include "Buyers.h"

#include "Storage.h"

#include "ViewData.h"

class SortByProductName

{

private:

vector<Order> orderCopy;

public:

void SortByDescending() {

orderCopy = Storage::order;

for (unsigned int i = 0; i < orderCopy.size() - 1; i++) {

for (int j = 0; j < orderCopy.size() - i - 1; j++) {

if (orderCopy[j].GetProductPrice() < orderCopy[j + 1].GetProductPrice()) {

Order temp = orderCopy[j];

orderCopy[j] = orderCopy[j + 1];

orderCopy[j + 1] = temp;

}

}

}

}

void SortByAscending() {

orderCopy = Storage::order;

for (unsigned int i = 0; i < orderCopy.size() - 1; i++) {

for (int j = 0; j < orderCopy.size() - i - 1; j++) {

if (orderCopy[j].GetProductPrice() > orderCopy[j + 1].GetProductPrice()) {

Order temp = orderCopy[j];

orderCopy[j] = orderCopy[j + 1];

orderCopy[j + 1] = temp;

}

}

}

}

void ShowSort(){

ViewData viewdate;

viewdate.ShowOrdersFromApplicationStorage(orderCopy);

}

};

Storage.h

#pragma once

#include <vector>

#include <iostream>

#include <string>

#include "Product.h"

#include "Order.h"

#include "Employee.h"

#include "Buyers.h"

using namespace std;

class Storage

{

public:

static vector<Product> product;

static vector<Employee> employee;

static vector<Buyers> buyers;

static vector<Order> order;

};

StorageService.h

#pragma once

#include "Storage.h"

class StorageService

{

public:

void DeleteOrder() {

int id = GetInput();

if (id < 0 || id >= Storage::order.size()) {

cout << "id не существует" << endl;

return;

}

Storage::order.erase(Storage::order.begin()+ id);

}

void DeleteProduct() {

int id = GetInput();

if (id < 0 || id >= Storage::product.size()) {

cout << "id не существует" << endl;

return;

}

Storage::product.erase(Storage::product.begin() + id);

}

void DeleteEmployee() {

int id = GetInput();

if (id < 0 || id >= Storage::employee.size()) {

cout << "id не существует" << endl;

return;

}

Storage::employee.erase(Storage::employee.begin() + id);

}

void DeleteBuyers() {

int id = GetInput();

if (id < 0 || id >= Storage::buyers.size()) {

cout << "id не существует" << endl;

return;

}

Storage::buyers.erase(Storage::buyers.begin() + id);

}

int GetInput() {

string menuNumberString;

int menuNumber;

bool isCorrectNumber = false;

while (!isCorrectNumber)

{

cout << "Введите id:\n";

cin >> menuNumberString;

try {

menuNumber = stoi(menuNumberString);

}

catch (...) {

cout << "ERROR\n";

isCorrectNumber = false;

continue;

}

isCorrectNumber = true;

}

return menuNumber;

}

};

ViewData.h

#pragma once

#include <iostream>

#include <string>

#include "Storage.h"

class ViewData

{

public:

void ShowEmployeesFromApplicationStorage() {

cout << "Сотрудники\n";

cout << "id" << setw(15) << "Имя" << setw(20) << "Фамилия" << setw(20) << "Отчество" << setw(20) << "Контакты" << setw(20) << "Должность" << setw(20) << "З/П(BY)\n";

cout << "----------------------------------------------------------------------------------------------------------------------------\n";

for (unsigned int i = 0; i < Storage::employee.size(); i++)

{

cout << i << setw(15) << Storage::employee[i].GetFirstName() << setw(20) << Storage::employee[i].GetSecondName() << setw(20)

<< Storage::employee[i].GetThirdName() << setw(20) << Storage::employee[i].GetContacts() <<

setw(20) << Storage::employee[i].GetPost() << setw(20) << Storage::employee[i].GetWages() << endl;

}

}

void ShowProductsFromApplicationStorage() {

cout << "Товары\n";

cout << "id" << setw(15) << "Тип техники" << setw(15) << "Название" << setw(15) << "Цена" << "\n";

cout << "------------------------------------------------------\n";

for (unsigned int i = 0; i < Storage::product.size(); i++)

{

string type;

switch (Storage::product[i].GetProductType())

{

case ProductType::Цифровая: {

type = "Цифровая";

break;

}

case ProductType::Бытовая: {

type = "Бытовая";

break;

}

case ProductType::Аудиотехника: {

type = "Аудиотехника";

break;

}

default: {

break;

}

}

cout << i << setw(15) << type << setw(15) << Storage::product[i].GetNameProduct() << setw(15) << Storage::product[i].GetPrice() << endl;

}

}

void ShowBuyersFromApplicationStorage() {

cout << "Покупатели\n";

cout << "id" << setw(15) << "Имя" << setw(20) << "Фамилия" << setw(20) << "Отчество" << setw(20) << "Контакты" << setw(20) << "Адрес\n";

cout << "-------------------------------------------------------------------------------------------------------------------------\n";

for (unsigned int i = 0; i < Storage::buyers.size(); i++)

{

cout << i << setw(15) << Storage::buyers[i].GetFirstName() << setw(20) << Storage::buyers[i].GetSecondName() << setw(20) <<

Storage::buyers[i].GetThirdName() << setw(20) << Storage::buyers[i].GetContacts() << setw(20) <<

Storage::buyers[i].GetAdress() << endl;

}

}

void ShowOrdersFromApplicationStorage() {

cout << "Заказы\n";

cout << setw(5) << "id " << setw(20) << "Товар " << setw(20) << "Цена\_товара " << setw(20) << "Контакты\_покупателя "

<< setw(20) << "Кол-во" << setw(20) << "Доставка" << setw(20) << "Итоговая цена" << setw(20) << "Дата\_заказа" << setw(20)

<< "Адрес доставки"<< "\n";

cout << "------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------\n";

for (unsigned int i = 0; i < Storage::order.size(); i++)

{

string type = (int)Storage::order[i].GetDeliveryType() == 1 ? "Да" : "Нет";

cout << setw(5) << i << setw(20) << Storage::order[i].GetNameProduct() << setw(20) << Storage::order[i].GetProductPrice() << setw(20) <<

Storage::order[i].GetContacts() << setw(20) << Storage::order[i].GetQuantity() << setw(20) << type<< setw(20) <<

Storage::order[i].GetFullPrice() << setw(20) << Storage::order[i].GetDayOfDate() << "/" << Storage::order[i].GetMonthOfDate() <<

"/" << Storage::order[i].GetYear() << setw(20) << Storage::order[i].GetCityName() << "|" << Storage::order[i].GetStreetName() <<

"|" << Storage::order[i].GetNumberOfHome() << "|" << Storage::order[i].GetNumberOfFlat() << endl;

}

}

void ShowOrdersFromApplicationStorage(vector<Order> order) {

cout << "Заказы\n";

cout << setw(5) << "id " << setw(15) << "Товар " << setw(15) << "Цена\_товара " << setw(25) << "Контакты\_покупателя " << setw(15) << "Кол-во" << setw(15) << "Доставка" << setw(15) << "Итоговая цена" << setw(20) << "Дата\_заказа" << setw(25) << "Адрес доставки" << "\n";

cout << "------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------\n";

for (unsigned int i = 0; i < order.size(); i++)

{

string type = (int)order[i].GetDeliveryType() == 1 ? "Да" : "Нет";

cout << setw(5) << i << setw(15) << order[i].GetNameProduct() << setw(15) << order[i].GetProductPrice() << setw(25) <<

order[i].GetContacts() << setw(15) << order[i].GetQuantity() << setw(15) << type << setw(15) <<

order[i].GetFullPrice() << setw(20) << order[i].GetDayOfDate() << "/" << order[i].GetMonthOfDate() <<

"/" << order[i].GetYear() << setw(25) << order[i].GetCityName() << "|" << order[i].GetStreetName() <<

"|" << order[i].GetNumberOfHome() << "|" << order[i].GetNumberOfFlat() << endl;

}

}

};