****

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

**«РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА И ГОСУДАРСТВЕННОЙ СЛУЖБЫ ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

КОЛЛЕДЖ МНОГОУРОВНЕВОГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ОТЧЕТ

По учебной практике УП.01.01 «Pазработка программных модулей»

Специальность №09.02.07 «Информационные системы и программирование»

Выполнила студент:

Руднева А.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

Гусятинер Л. Б.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(оценка)

Москва, 2023

**Содержание**

[**Раздел 1. Проектирование консольного приложения** 2](#_Toc153356734)

[1.1 Общая информация 2](#_Toc153356735)

[1.2 Изучение входной и выходной документации 3](#_Toc153356736)

[1.3 Разработка требований к проекту. Построение диаграммы использования. 6](#_Toc153356737)

[1.3 Разработка сценария проекта 7](#_Toc153356738)

[1.4 Построение диаграммы классов 8](#_Toc153356739)

[**Раздел 2. Разработка программного модуля** 10](#_Toc153356740)

[2.1 Разработка консольного приложения 10](#_Toc153356741)

[2.2 Тестирование и отладка 16](#_Toc153356742)

[Приложение 1. Руководство программиста 19](#_Toc153356743)

[Приложение 2. Руководство пользователя 20](#_Toc153356744)

[Приложение 3. Листинги 21](#_Toc153356745)

## **Раздел 1. Проектирование консольного приложения**

## Общая информация

Проект представляет собой консольное приложение на языке Python.

**Описание бизнес-процесса**

Описание процесса **оформления заказа**

1. Клиент приходит в кондитерскую
2. Клиент выбирает столик
3. Официант подходит к столику
4. Официант дает клиенту меню
5. Официант уточняет информацию об аллергиях клиента
6. Официант дает рекомендации о том, что не стоит заказывать и говорит стоп лист
7. Клиент выбирает блюда
8. Если клиент попросил время подумать, официант уходит, приходит через 5 минут и спрашивает о готовности, если клиент не готов - цикл идет сначала, если готов -следующий пункт
9. Клиент сообщает желаемые блюда
10. Официант записывает позиции в блокнот
11. Официант говорит гостю среднее время ожидания
12. Официант записывает заказ в кассу
13. Официант относит чек на кухню

## 1.2 Изучение входной и выходной документации

**Входные данные:**

1. Формат документа, содержащего клиентов

**Client**

Client\_id(int)

stol(int)

name(text)

age(int)

* 1. Содержание файла client.txt

Листинг 1. Содержание файла client.txt

01;1;Анна Руднева;18  
02;3;Татьяна Королёва;42  
03;4;Алексей Руднев;16

1. Формат файла, содержащего заказы

**Zakaz**

Zakaz\_id(int)

Client\_id(int)

Waiter\_id(int)

bludo1\_id(int)

bludo2\_id (int)

bludo3\_id (int)

bludo4\_id (int)

bludo5\_id (int)

* 1. Содержание файла zakaz.txt

Листинг 2. Содержание файла zakaz.txt

01;03;01;01  
02;02;02;02;03;04  
03;01;02;02;05

1. Формат файла, содержащего официантов

**Waiter**

Waiter\_id(int)

Fio(text)

Telefonе(str)

Email(str)

* 1. Содержание файла waiter.txt

Листинг 3. Содержание файла waiter.txt

01;Руднева Анастасия Алексеевна;79037830916;Ru-nastia@mail.ru  
02;Горбин Михаил Васильевич;79037835614;mihagor@gmail.com  
03;Васильева Екатерина Сергеевна;79060933671;Vasia-kater23@mail.ru

1. Формат файла, содержащего блюда

**Bludo**

Dludo\_id(int)

Name (text)

price(int)

ves(int)

* 1. Содержание файла bludo.txt

Листинг 4. Содержание файла bludo.txt

01;Клубничный чизкейк;300;200  
02;Шоколадный пирог;700;300  
03;Анна Павлова;600;100  
04;Пирожоное картошка;100;150  
05;Эклер шоколадный;300;200

**Выходные данные:**

1. Формат файла суммы заказа

**Order\_price**

Order\_id(int)

Price(int)

* 1. Содержание файла order\_price

Листинг 5. Содержание файла order\_pricе

01;300  
02;1400  
03;500

1. Формат файла заказов официанта

**Zakaz\_of\_waiter**

Waiter\_id(int)

Name(text)

Orders(text)

* 1. Содержание файла zakaz\_of\_waiter

Листинг 6. Содержание файла zakaz\_of\_waiter

01;Руднева Анна Алекссевна;01  
02;Горбин Михаил Васильевич;03 02  
03;Васильева Екатерина Сергеевна;

1. Формат файла количества заказов блюда

**Bludo\_number**

Bludo\_id(int)

Number(int)

* 1. Содержание файла bludo\_number

Листинг 7. Содержание файла bludo\_number

01;1  
02;2  
03;1  
04;1  
05;1

## 1.3 Разработка требований к проекту. Построение диаграммы использования.

Программа должна содержать функции оформления заказа, учета клиентов, учета официантов, учета блюд и заказов. Также в программе можно будет сделать выборку всех заказов, обслуженых определенным официантом, посчитать сумму для оплаты заказа, посчитать сколько раз определенное блюдо было заказано. Варианты использования приложения разными пользователями представлены на диаграмме использования (Рисунок 1).

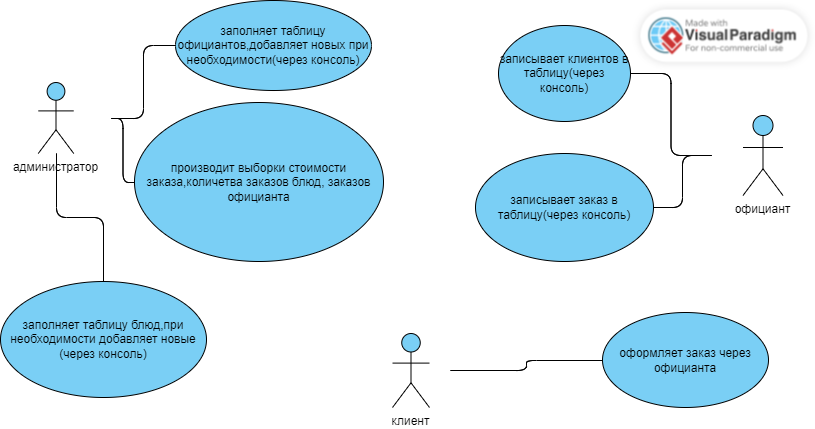


Рисунок 1. Диаграмма использования

## 1.3 Разработка сценария проекта

При открытии приложения пользователь видит консольное меню,в которм указано какие цифры нужно ввести,чтобы произвести определенное действие.В программе предусмотрены функции вывода спиков клиентов,блюд,заказов и официантов,которые выводят всю информацию из соответствующих файлов(список) на экран;функции добавления новых клиентов,официантов,заказов и блюд,с помощью которые вносится новая строка в соотвествующий файл(список);функции выборок:вывода кол-ва заказов блюда,списка заказов официанта,суммы для оплаты заказа ,функции вывода делятся на два типа:вывод на экран по id и вывод общей сводки в соответствующий файл; функция выхода из приложения.

На рисунке 2 изображен сценарий использования программы пользователем.

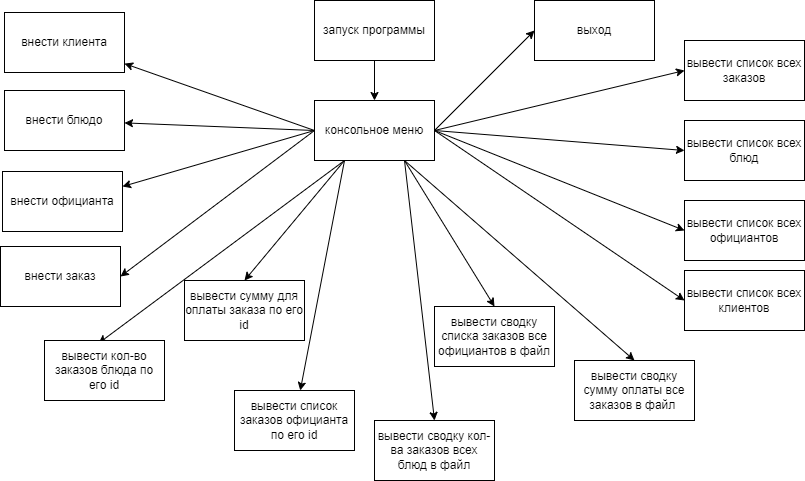


Рисунок 2. Сценарий проекта

## 1.4 Построение диаграммы классов

В данном разделе находится диаграмма классов приложения.(Рисунок 3)

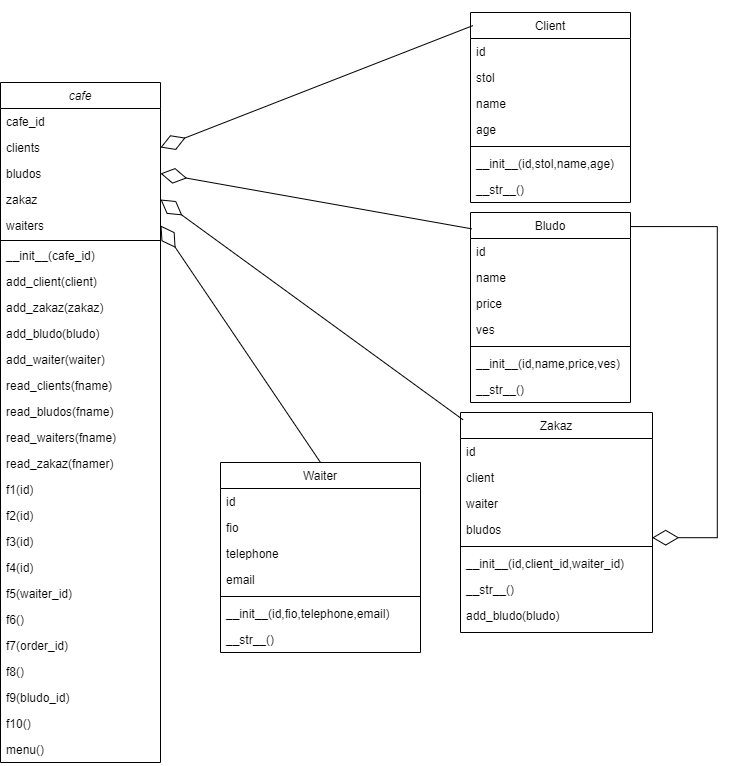


Рисунок 3. Диаграмма классов

Система классов проекта включает в себя следующие классы:

1.Класс Cafe (Кафе) - основной класс, который представляет собой само кафе. В этом классе реализуются основные методы и свойства кафе, такие как добавление блюда в меню, оформление заказа,вывод сводок и т.д.

2. Класс Bludo (Блюдо) - класс, представляющий отдельное блюдо в меню кафе. В этом классе содержится информация о названии блюда, ,цене и весе. Также в классе Bludo реализован методы, отвечающий за печать блюда.

3. Класс Zakaz (Заказ) - класс, представляющий оформленный заказ. В этом классе содержится информация о выбранных блюдах,id клиентаи id официанта,оформляющего заказ. В классе Zakaz содержаться методы для добавления и печати блюда.

4. Класс Waiter (Официант) - класс, представляющий официанта, который обслуживает клиентов в кафе. В этом классе содержится информация о фио официанта, его номере телефона и электронной почте и метод печати.

5. Класс Client (Клиент) - класс, представляющий клиента кафе. В этом классе содержится информация о имени клиента, его столе и возрасте. Класс также содержит метод для печати.

## **Раздел 2. Разработка программного модуля**

## 2.1 Разработка консольного приложения

Описание спецификаций к модулям

Всего разработано 5 модулей:

* Основной(main)
* Модуль класса Client(client)
* Модуль класса Zakaz(zakaz)
* Модуль класса Waiter(waiter)
* Модуль класса Bludo(bludo)

Взаимодействие между модулями можно наглядно увидеть по диаграммам классов из раздела 1.

**Описание модулей:**

**Основной модуль** – в этом модуле содержиться основной класс проекта Сafе ,в котором реализовано большинство функций для работы с данными ,и код для запуска приложения.

**Модули классов** –содержат описание классов (их атрибуты и методы).

Листинг 8. Код запуска приложения

if \_\_name\_\_ == '\_\_main\_\_':  
 cafe = Cafe(1)  
 cafe.read\_bludos("bludo.txt")  
 cafe.read\_clients("clients.txt")  
 cafe.read\_zakaz("zakaz.txt")  
 cafe.read\_waiters('waiter.txt')  
 cafe.menu()

Рассмотрим основную функцию модуля – функцию menu():

Листинг 9. Функция menu()

def menu(self):  
 print(''' Меню:   
 Введите 1,если хотите добавить нового клиента  
 Введите 2,если хотите добавить нового официанта  
 Введите 3,если хотите добавить новое блюдо  
 Введите 4,если хотите добавить новый заказ  
 Введите 5,если хотите посмотреть номера всех заказов,которые обслужил официант  
 Введите 6,если хотите вывести сводку заказов всех официантов в файл  
 Введите 7,если хотите узнать цену заказа  
 Введите 8,если хотите вывести сводку цен всех заказов в файл  
 Введите 9,если хотите узнать сколько раз было заказано блюдо  
 Введите 10,если хотите вывести кол-во заказов всех блюд в файл  
 Введите 11,если хотите помотреть список всех клиентов  
 Введите 12,если хотите посмтореть список всех официантов  
 Введите 13,если хотите посмотреть список всех блюд  
 Введите 14,если хотите посмотреть списое всех заказов  
 Введите 15,если хотите выйти из приложения''')  
 s = int(input())  
 if s == 1:  
 print('Введите id клиента')  
 id = int(input())  
 self.f1(id)  
  
 elif s == 2:  
 print('Введите id официанта')  
 id = int(input())  
 self.f2(id)  
  
 elif s == 3:  
 print('Введите id блюда')  
 id = int(input())  
 self.f3(id)  
 elif s == 4:  
 print('Введите id заказа')  
 id = int(input())  
 self.f4(id)  
  
 elif s == 5:  
 print('Введите id официанта')  
 waiter1\_id = int(input())  
 self.f5(waiter1\_id)  
  
 elif s == 6:  
 self.f6()  
  
 elif s == 7:  
 print('Введите id заказа')  
 order\_id = int(input())  
 self.f7(order\_id)  
 elif s == 8:  
 self.f8()  
  
 elif s == 9:  
 print('Введите номер блюда')  
 bludo\_id = int(input())  
 self.f9(bludo\_id)  
  
 elif s == 10:  
 self.f10()  
  
 elif s == 11:  
 for client in cafe.clients:  
 print(client)  
 self.menu()  
  
 elif s == 12:  
 for waiter in cafe.waiters:  
 print(waiter)  
 self.menu()  
  
 elif s == 13:  
 for bludo in cafe.bludos:  
 print(bludo)  
 self.menu()

elif s ==14:  
 for order in cafe.zakaz:  
 print(order)  
 self.menu()  
  
 elif s == 15:  
 sys.exit()  
  
 else:  
 print('Ошибка!Введите число от 1 до 15!')  
 self.menu()

Через эту функцию происходит взаимодействие с пользователем и обращение к множеству функций, выполняющих запрос пользователя. Рассмотрим одну, из таких функций - функцию f10.

Листинг 10. Функция f10()

f f10(self):#функция заполнения файла кол-ва заказов блюд  
 with open('bludo\_number', 'w', encoding='utf-8') as b\_n:  
 b\_n.truncate(0)  
 j = 0  
 for bludo in cafe.bludos:  
 with open('zakaz.txt', 'r', encoding='utf-8') as order\_f:  
 for line in order\_f:  
 id, client\_id, waiter\_id, orders = line.split(";")  
 a = len(orders)  
 f = orders.split(',')  
 for i in range(0, (a + 1) // 3):  
 if bludo.id == int(f[i]):  
 j += 1  
 with open('bludo\_number', 'a', encoding='utf-8') as b\_n:  
 b\_n.write(str(bludo.id) + ';' + str(j) + '\n')  
 j = 0  
 print('Файл успешно заполнен!')  
 self.menu()

Для лучшего понимания логики работы функции предлаается посмотреть её блок-схему (Рисунки 4-6).

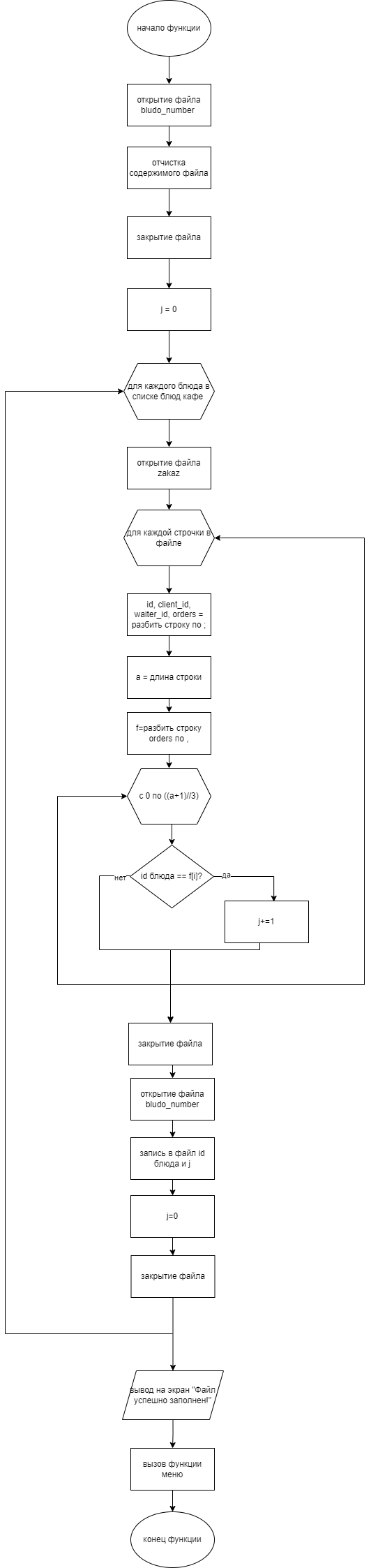
****

Рисунок 4. Блок-схема функции f10(). Часть 1.

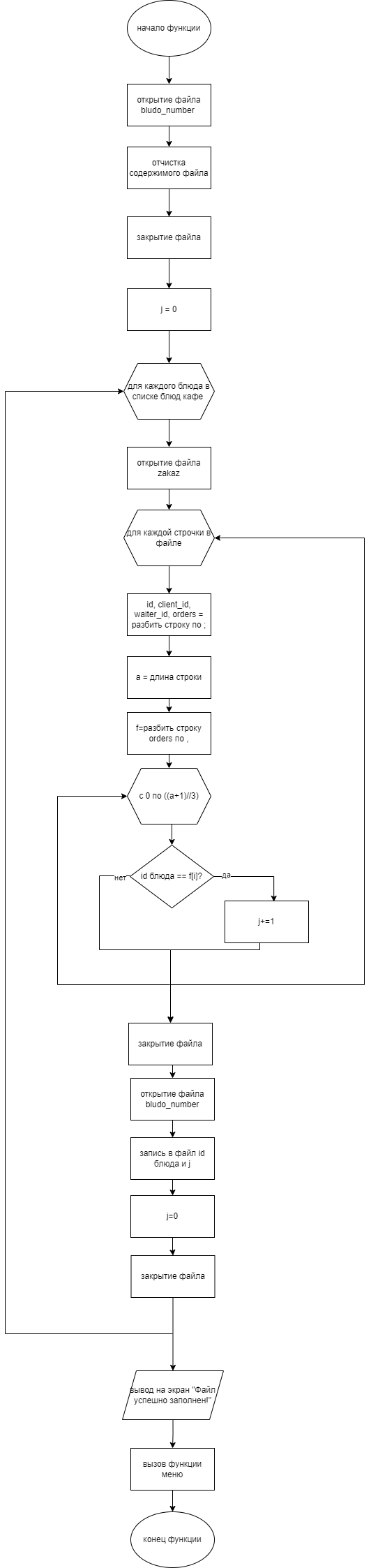
****

Рисунок 5. Блок-схема функции f10(). Часть 2.

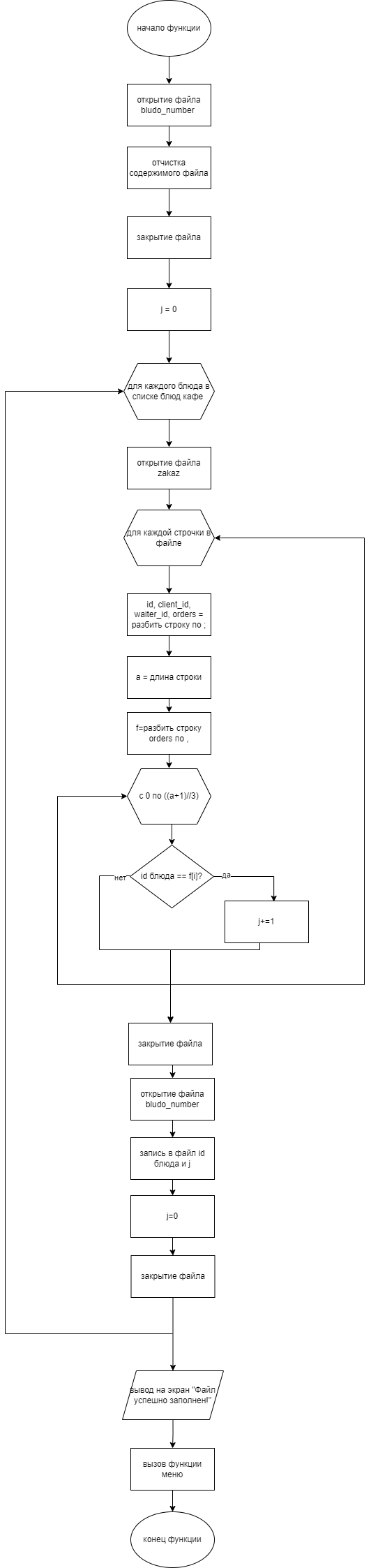
****

Рисунок 6. Блок-схема функции f10(). Часть 3.

## 2.2 Тестирование и отладка

В ходе написания проекта при попытках запуска программы было обнаружено множество ошибок, мешающих запуску программы и влияющих на её логику. Ошибки были своевременно исправлены, в результате при итоговом запуске программы ошибок не было:

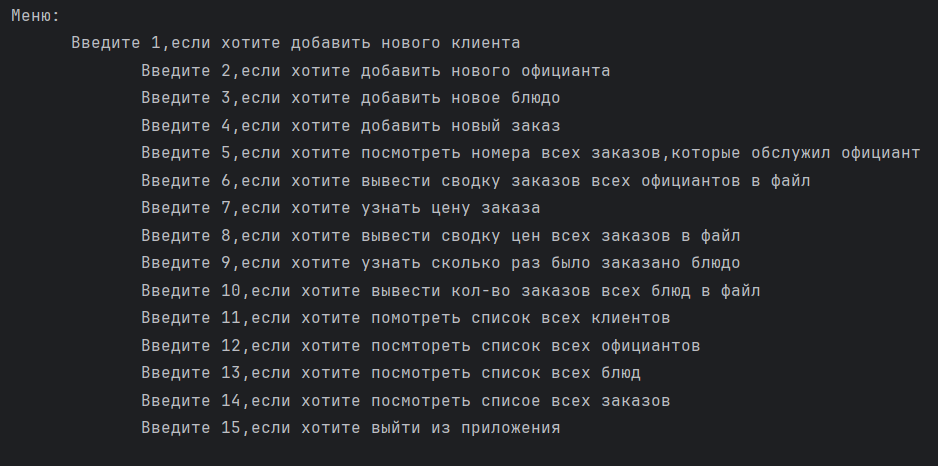


Рисунок 7. Успешный запуск

## **Дневник**

|  |  |  |
| --- | --- | --- |
| Дата | Содержание работ | Отметка о выполнении |
| 01.09.2023 | Вводный инструктаж  Формирование алгоритма решения вычислительной задачи  Формирование алгоритма решения задачи по обработке данных |  |
| 04.09.2023 | Изучение средств визуализации алгоритмов |  |
| 09.09.2023 | . Оформление математического алгоритма при помощи программных средств |  |
| 11.09.2023 | Разработка простой спецификации учебной задачи: Введение. Общее описание |  |
| 16.09.2023 | Разработка простой спецификации учебной задачи: Функции системы |  |
| 22.09.2023 | Разработка простой спецификации учебной задачи: требования к данным, требования к внешним интерфейсам, требования к внешним интерфейсам |  |
| 29.09.2023 | Оформление спецификации программного продукта. Оформление введения и раздела "Общее описание"  Оформление раздела "Функции системы" |  |
| 04.10.2023 | . Оформление раздела "Требования к внешним интерфейсам"  Оформление раздела "Атрибуты качества" |  |
| 11.10.2023 | Разработка классов главного модуля  Разработка методов классов и функций главного модуля |  |
| 18.10.2023 | Разработка классов подсистемы вычислений  Разработка классов подсистемы связи с внешними данными |  |
| 25.10.2023 | Разработка классов подсистемы отчётов  Разработка тестовых наборов |  |
| 01.11.2023 | Провести тестирование главного модуля  Провести тестирование модулей |  |
| 08.11.2023 | Провести тестирование программного продукта  По результатам тестирования провести рефакторинг программного продукта |  |
| 22.11.2023 | Разработать руководство программиста: Обращение к программе |  |
| 29.11.2023 | Разработать руководство программиста: Характеристика программы  Разработать руководство программиста: Обращение к программе |  |
| 06.12.2023 | Разработать руководство программиста: Входные и выходные данные  Разработать руководство программиста: Сообщения |  |
| 13.12.2023 | Разработать руководство пользователя: Введение. Назначение и условия применения.  Разработать руководство пользователя: Подготовка к работе. |  |
| 20.12.2023 | Разработать руководство пользователя: Описание операций.  Разработать руководство пользователя: Аварийные ситуации |  |
| 27.12.2023 | Разработать руководство пользователя: Рекомендации по освоению. |  |

## Приложение 1. Руководство программиста

## Приложение 2. Руководство пользователя

## Приложение 3. Листинги

Листинг 9. Главный модуль приложения

from client import Client  
from bludo import Bludo  
from zakaz import Zakaz  
from waiter import Waiter  
import sys  
  
class Cafe:  
 def \_\_init\_\_(self,cafe\_id:int):  
 self.cafe\_id = cafe\_id  
 self.clients =[]  
 self.bludos =[]  
 self.zakaz =[]  
 self.waiters =[]  
  
 def add\_client(self,client:Client):  
 self.clients.append(client)  
 def add\_zakaz(self,zakaz:Zakaz):  
 self.zakaz.append(zakaz)  
 def add\_bludo(self,bludo:Bludo):  
 self.bludos.append(bludo)  
 def add\_waiter(self,waiter:Waiter):  
 self.waiters.append(waiter)  
  
  
 def read\_clients(self,fname):  
 with open (fname,'r',encoding ='utf-8') as clients\_f:  
 for line in clients\_f:  
 id, stol, name, age = line.split(";")  
 id = int(id)  
 stol = int(stol)  
 age = int(age)  
 client = Client(id, stol,name,age)  
 self.clients.append(client)  
  
 def read\_bludos(self,fname):  
 with open (fname,'r',encoding ='utf-8') as bludo\_f:  
 for line in bludo\_f:  
 id, name,price, ves = line.split(";")  
 id = int(id)  
 price = int(price)  
 ves = int(ves)  
 bludo = Bludo(id,name,price,ves)  
 self.bludos.append(bludo)  
  
 def read\_waiters(self,fname):  
 with open (fname,'r',encoding ='utf-8') as waiters\_f:  
 for line in waiters\_f:  
 id, fio, telephone, email = line.split(";")  
 id = int(id)  
 waiter = Waiter(id, fio, telephone, email)  
 self.waiters.append(waiter)  
  
 def read\_zakaz(self,fname):  
 with open (fname,'r',encoding ='utf-8') as waiters\_f:  
 for line in waiters\_f:  
 id, client\_id, waiter\_id,orders = line.split(";")  
 id = int(id)  
 client\_id = int(client\_id)  
 waiter\_id = int(waiter\_id)  
 zakaz = Zakaz(id, client\_id, waiter\_id)  
 s = len(orders)  
 f = orders.split(',')  
 for i in range(0,(s+1)//3):  
 zakaz.add\_bludo(f[i])  
 self.zakaz.append(zakaz)  
  
 def f1(self,id):#функция записи клиента  
 print('Введите номер стола')  
 stol = int(input())  
 print('Введите имя и фамилию клиента')  
 name = input()  
 print('Введите возраст клиента')  
 age = int(input())  
 client = Client(id, stol, name, age)  
 cafe.add\_client(client)  
 client\_file = open('clients.txt', 'a', encoding='utf-8')  
 client\_file.write('\n' + str(id) + ';' + str(stol) + ';' + name + ';' + str(age))  
 client\_file.close()  
 print('Клиент записан успешно!\n')  
 self.menu()  
 def f2(self,id):#функция записи официанта  
 print('Введите фио официанта')  
 fio = input()  
 print('Введите телефон')  
 phone = input()  
 print('Введите электронную почту')  
 email = (input())  
 waiter = Waiter(id, fio, phone, email)  
 cafe.add\_waiter(waiter)  
 waiter\_file = open('waiter.txt', 'a', encoding='utf-8')  
 waiter\_file.write('\n' + str(id) + ';' + fio + ';' + phone + ';' + email)  
 waiter\_file.close()  
 print('Официант записан успешно!\n')  
 self.menu()  
 def f3(self,id):#функция записи блюда  
 print('Введите название блюда')  
 name = input()  
 print('Введите цену')  
 price = int(input())  
 if price <= 0:  
 print('Ошибка!Цена должна быть больше 0!')  
 self.menu()  
 print('Введите вес')  
 ves = int(input())  
 bludo = Bludo(id, name, price, ves)  
 cafe.add\_bludo(bludo)  
 bludo\_file = open('bludo.txt', 'a', encoding='utf-8')  
 bludo\_file.write('\n' + str(id) + ';' + name + ';' + str(price) + ';' + str(ves))  
 bludo\_file.close()  
 print('Блюдо записано успешно!\n')  
 self.menu()  
 def f4(self,id):#функция записи заказа  
 print('Введите id клиента')  
 client\_id = int(input())  
 print('Введите id официанта')  
 waiter\_id = int(input())  
 zakaz = Zakaz(id, client\_id, waiter\_id)  
 print('Введите колличетво блюд')  
 col = int(input())  
 if col > 5:  
 print('Ошибка!Блюд должно быть не больше 5!')  
 self.menu()  
 for i in range(1, col + 1):  
 print(f'Введите id {i} блюда')  
 bludo = input()  
 zakaz.add\_bludo(bludo)  
 cafe.add\_zakaz(zakaz)  
 zakaz\_file = open('zakaz.txt', 'a', encoding='utf-8')  
 stroka = ','.join(zakaz.bludos)  
 zakaz\_file.write('\n' + str(id) + ';' + str(client\_id) + ';' + str(waiter\_id) + ';' + stroka)  
 zakaz\_file.close()  
 print('Заказ записан успешно!\n')  
 self.menu()  
 def f5(self,waiter1\_id):#функция вывода заказов официанта  
 with open('waiter.txt', 'r', encoding='utf-8') as waiters\_f:  
 for line in waiters\_f:  
 id, fio, telephone, email = line.split(";")  
 id = int(id)  
 if waiter1\_id == id:  
 break  
 waiter1\_id = id  
 a = []  
 with open('zakaz.txt', 'r', encoding='utf-8') as zakaz\_f:  
 for line in zakaz\_f:  
 id, client\_id, waiter\_id, orders = line.split(";")  
 id = int(id)  
 client\_id = int(client\_id)  
 waiter\_id = int(waiter\_id)  
 if waiter\_id == waiter1\_id:  
 a.append(str(id))  
 stroka = ','.join(a)  
 if stroka == '':  
 stroka = 'нет заказов'  
 print('id:' + str(waiter\_id) + ' фио:' + fio + 'заказы:' + stroka)  
 self.menu()  
 def f6(self):#функция заполнения файла заказов официантов  
 with open('zakaz\_of\_waiter', 'w', encoding='utf-8') as z\_w:  
 z\_w.truncate(0)  
  
 for waiter in cafe.waiters:  
 with open('waiter.txt', 'r', encoding='utf-8') as waiters\_f:  
 for line in waiters\_f:  
 id, fio, telephone, email = line.split(";")  
 id = int(id)  
 if waiter.id == id:  
 break  
 a = []  
 with open('zakaz.txt', 'r', encoding='utf-8') as zakaz\_f:  
 for line in zakaz\_f:  
 id, client\_id, waiter\_id, orders = line.split(";")  
 id = int(id)  
 waiter\_id = int(waiter\_id)  
 if waiter\_id == waiter.id:  
 a.append(str(id))  
 stroka = ','.join(a)  
 if stroka == '':  
 stroka = 'нет заказов'  
 with open('zakaz\_of\_waiter', 'a', encoding='utf-8') as z\_w:  
 z\_w.write(str(waiter.id) + ';' + fio + ';' + stroka + '\n')  
 print('Файл успешно заполнен!')  
 self.menu()  
 def f7(self,order\_id):#функция вывода цены заказа  
 with open('zakaz.txt', 'r', encoding='utf-8') as order\_f:  
 for line in order\_f:  
 id, client\_id, waiter\_id, orders = line.split(";")  
 id = int(id)  
 client\_id = int(client\_id)  
 waiter\_id = int(waiter\_id)  
 if order\_id == id:  
 if '\n' in orders:  
 orders = orders[:-1]  
 a = len(orders)  
 f = orders.split(',')  
 bllist = []  
 for i in range(0, (a + 2) // 3):  
 bllist.append(f[i])  
  
 with open('bludo.txt', 'r', encoding='utf-8') as bludo\_f:  
 sum = 0  
 for line in bludo\_f:  
 id, name, price, ves = line.split(";")  
  
 price = int(price)  
 ves = int(ves)  
 if id in bllist:  
 sum += price  
  
 print('id заказа:' + str(order\_id) + ' цена:' + str(sum))  
 self.menu()  
 def f8(self):#функция заполнения файла цен заказов  
 with open('order\_price', 'w', encoding='utf-8') as o\_p:  
 o\_p.truncate(0)  
 for zakaz in cafe.zakaz:  
 with open('zakaz.txt', 'r', encoding='utf-8') as order\_f:  
 for line in order\_f:  
 id, client\_id, waiter\_id, orders = line.split(";")  
 id = int(id)  
 if zakaz.id == id:  
 if '\n' in orders:  
 orders = orders[:-1]  
 a = len(orders)  
 f = orders.split(',')  
 bllist = []  
 for i in range(0, (a + 2) // 3):  
 bllist.append(f[i])  
  
 with open('bludo.txt', 'r', encoding='utf-8') as bludo\_f:  
 sum = 0  
 for line in bludo\_f:  
 id, name, price, ves = line.split(";")  
  
 price = int(price)  
 ves = int(ves)  
 if id in bllist:  
 sum += price  
 with open('order\_price', 'a', encoding='utf-8') as o\_p:  
 o\_p.write(str(zakaz.id) + ';' + str(sum) + '\n')  
 print('Файл успешно заполнен!')  
 self.menu()  
 def f9(self,bludo\_id):#функция вывода кол-ва заказов блюда  
 j = 0  
 with open('zakaz.txt', 'r', encoding='utf-8') as order\_f:  
 for line in order\_f:  
 id, client\_id, waiter\_id, orders = line.split(";")  
 a = len(orders)  
 f = orders.split(',')  
 for i in range(0, (a + 1) // 3):  
 if bludo\_id == int(f[i]):  
 j += 1  
 print('id блюда:' + str(bludo\_id) + ' кол-во заказов:' + str(j))  
 self.menu()  
 def f10(self):#функция заполнения файла кол-ва заказов блюд  
 with open('bludo\_number', 'w', encoding='utf-8') as b\_n:  
 b\_n.truncate(0)  
 j = 0  
 for bludo in cafe.bludos:  
 with open('zakaz.txt', 'r', encoding='utf-8') as order\_f:  
 for line in order\_f:  
 id, client\_id, waiter\_id, orders = line.split(";")  
 a = len(orders)  
 f = orders.split(',')  
 for i in range(0, (a + 1) // 3):  
 if bludo.id == int(f[i]):  
 j += 1  
 with open('bludo\_number', 'a', encoding='utf-8') as b\_n:  
 b\_n.write(str(bludo.id) + ';' + str(j) + '\n')  
 j = 0  
 print('Файл успешно заполнен!')  
 self.menu()  
 def menu(self):  
 print(''' Меню:   
 Введите 1,если хотите добавить нового клиента  
 Введите 2,если хотите добавить нового официанта  
 Введите 3,если хотите добавить новое блюдо  
 Введите 4,если хотите добавить новый заказ  
 Введите 5,если хотите посмотреть номера всех заказов,которые обслужил официант  
 Введите 6,если хотите вывести сводку заказов всех официантов в файл  
 Введите 7,если хотите узнать цену заказа  
 Введите 8,если хотите вывести сводку цен всех заказов в файл  
 Введите 9,если хотите узнать сколько раз было заказано блюдо  
 Введите 10,если хотите вывести кол-во заказов всех блюд в файл  
 Введите 11,если хотите помотреть список всех клиентов  
 Введите 12,если хотите посмтореть список всех официантов  
 Введите 13,если хотите посмотреть список всех блюд  
 Введите 14,если хотите посмотреть список всех заказов  
 Введите 15,если хотите выйти из приложения''')  
 s = int(input())  
 if s == 1:  
 print('Введите id клиента')  
 id = int(input())  
 self.f1(id)  
  
 elif s == 2:  
 print('Введите id официанта')  
 id = int(input())  
 self.f2(id)  
  
 elif s == 3:  
 print('Введите id блюда')  
 id = int(input())  
 self.f3(id)  
 elif s == 4:  
 print('Введите id заказа')  
 id = int(input())  
 self.f4(id)  
  
 elif s == 5:  
 print('Введите id официанта')  
 waiter1\_id = int(input())  
 self.f5(waiter1\_id)  
  
 elif s == 6:  
 self.f6()  
  
 elif s == 7:  
 print('Введите id заказа')  
 order\_id = int(input())  
 self.f7(order\_id)  
  
 elif s == 8:  
 self.f8()  
  
 elif s == 9:  
 print('Введите номер блюда')  
 bludo\_id = int(input())  
 self.f9(bludo\_id)  
  
 elif s == 10:  
 self.f10()  
  
 elif s == 11:  
 for client in cafe.clients:  
 print(client)  
 self.menu()  
  
 elif s == 12:  
 for waiter in cafe.waiters:  
 print(waiter)  
 self.menu()  
  
 elif s == 13:  
 for bludo in cafe.bludos:  
 print(bludo)  
 self.menu()  
  
 elif s ==14:  
 for order in cafe.zakaz:  
 print(order)  
 self.menu()  
  
 elif s == 15:  
 sys.exit()  
  
 else:  
 print('Ошибка!Введите число от 1 до 15!')  
 self.menu()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 cafe = Cafe(1)  
 cafe.read\_bludos("bludo.txt")  
 cafe.read\_clients("clients.txt")  
 cafe.read\_zakaz("zakaz.txt")  
 cafe.read\_waiters('waiter.txt')  
 cafe.menu()

Листинг 10. Модуль bludo

class Bludo:  
 def \_\_init\_\_(self,id:int,name:str,price:int,ves:int):  
 self.id = id  
 self.name = name  
 self.price = price  
 self.ves = ves  
  
 def \_\_str\_\_(self):  
 return f"{self.id} Название:{self.name} Цена:{self.price} Вес:{self.ves}"

Листинг 11. Модуль client

class Client:  
 def \_\_init\_\_(self,id:int,stol:int,name:str,age:int):  
 self.id = id  
 self.stol = stol  
 self.name = name  
 self.age = age  
  
 def \_\_str\_\_(self):  
 return f"{self.id} стол:{self.stol} имя:{self.name} возраст:{self.age}"

Листинг 12. Модуль waiter

class Waiter:  
 def \_\_init\_\_(self,id:int,fio:str,telephone:str,email:str):  
 self.id = id  
 self.fio = fio  
 self.telephone = telephone  
 self.email = email  
  
 def \_\_str\_\_(self):  
 return f"{self.id} ФИО:{self.fio} Номер телефона:{self.telephone} Почта:{self.email}"

Листинг 13. Модуль zakaz

class Zakaz:  
 def \_\_init\_\_(self,id:int,client\_id:int,waiter\_id:int):  
 self.bludos =[]  
 self.id = id  
 self.client = client\_id  
 self.waiter = waiter\_id  
 def add\_bludo(self,bludo):  
 self.bludos.append(bludo)  
  
 def \_\_str\_\_(self):  
 stroka = ','.join(self.bludos)  
 if '\n' in stroka:  
 stroka = stroka[:-1]  
 return f"{self.id} клиент:{self.client} официант:{self.waiter} блюда:{stroka}"