

Санкт-Петербургский Национальный  
Исследовательский Университет Информационных  
технологий, механики и оптики

## **Домашнее задание**

### **Реализация программной модели инфокоммуникационной системы**

Выполнил: Абдулов  
Илья Александрович  
Группа № К3121  
Проверила: Казанова  
Полина Петровна

Санкт-Петербург  
2022

# СОДЕРЖАНИЕ

Стр.

<b>1</b>	<b>Введение .....</b>	<b>3</b>
1.1	Задание.....	3
1.2	Цель работы .....	3
1.3	Задачи .....	3
<b>2</b>	<b>Ход работы .....</b>	<b>4</b>
2.1	Анализ предметной области и требований .....	4
2.2	Реализация программы.....	4
<b>3</b>	<b>Вывод .....</b>	<b>18</b>

# **1 Введение**

## **1.1 Задание**

Создать программное обеспечение системы обработки данных: «Программа для контроля собственных денежных средств».

Необходимо реализовать следующие функции, позволяющие:

1. Добавлять продукт в коллекцию (тип коллекции на ваш выбор).
2. Просматривать все записанное в программу.
3. Просматривать покупки по дате и категории.
4. Распределять их по стоимости от минимальной к максимальной или наоборот.
5. Удалять требуемые записи и выходить из программы.

Дополнительные указания: интерфейс программы реализовать в отдельной функции на ваше усмотрение.

Рекомендуется реализовать возможность сохранения данных в файл.

## **1.2 Цель работы**

Создать программное обеспечение системы обработки данных в виде интерактивного консольного приложения в Python3: «Программа для контроля собственных денежных средств».

## **1.3 Задачи**

1. Описать этапы анализа предметной области и требований
2. Описать решение задания с указанием применения элементов программирования
3. Представить графические материалы, блок-схемы алгоритмов
4. Графически продемонстрировать работу информационной системы

## **2   Ход работы**

### **2.1   Анализ предметной области и требований**

Система предназначена для учета собственных денежных средств и для повышения финансовой грамотности пользователя. Программа напоминает мобильные приложения банков, где используют системы контроля денежных средств, которые предоставляются клиентам банка.

Требуется, чтобы приложение:

1. имело интуитивно-понятный интерфейс,
2. имело функции добавления/удаления продукта,
3. умело отображать все продукты или в соответствии с датой/категорией,
4. умело сортировать продукты по стоимости,
5. умело обрабатывать некорректно введенные значения, любые действия пользователя,
6. умело сохранять данные о продуктах в коллекции.

Взаимодействие с пользователем должно происходить в режиме реального времени, то есть, если пользователь делает запрос к системе, то получает ответ по завершении обработки процесса в программе.

Все продукты, которые пользователь захочет сохранить в коллекции по завершении работы программы, будут храниться в текстовом файле, из которого программа извлекает данные после запуска.

Проектирование интеллектуальной системы будет реализовано на высокоуровневом языке программирования Python3, который имеет достаточно эффективности и функционала, чтобы реализовать программу, соответствующую требованиям. Взаимодействие с пользователем будет происходить в интерактивной командной строке.

### **2.2   Реализация программы**

В начале программы реализовано приветствие пользователя (рис. 2.1, 2.4).

```

127 print('Программа для контроля собственных денежных средств запущена\n')
128 helper() # Выводит команды, которые обрабатывает программа
129 extract() # Извлекает сохраненные продукты из файла в коллекцию

```

Рисунок 2.1 — Код приветствия пользователя

Функция `helper()` выводит команды, которые распознаёт программа (рис. 2.2).

```

106 def helper():
107     print('!add — добавляет продукт в коллекцию')
108     print('!all — показывает все продукты')
109     print('!bycd — просмотреть продукты по дате и категории')
110     print('!sort — распределяет продукты по стоимости')
111     print('!del — удаляет запись')
112     print('!save — сохраняет коллекцию в файл')
113     print('!exit — выходит из программы')
114     print()

```

Рисунок 2.2 — Код функции `helper()`

Функция `extract()` извлекает из текстового файла все имеющиеся записи с прошлых сессий работы программы и добавляет в коллекцию (рис. 2.3).

```

90 def extract():
91     global collection
92     with open('data.txt') as w: # data.txt — текстовый файл с продуктами
93         for line in w.readlines():
94             li = line.split('_')
95             collection += [[int(li[0]), li[1], li[2], li[3]]]

```

Рисунок 2.3 — Код функции `extract()`

- o `ilyaa@Ilyas-MacBook-Air-2 uDZ % /usr/local/bin/python3 /Users`  
Программа для контроля собственных денежных средств запущена

```
!add — добавляет продукт в коллекцию
!all — показывает все продукты
!bycd — просмотреть продукты по дате и категории
!sort — распределяет продукты по стоимости
!del — удаляет запись
!save — сохраняет коллекцию в файл
!exit — выходит из программы
```

Введите команду: █

---

Рисунок 2.4 — Вывод приветствия в консоли

Далее запускается бесконечный цикл, в котором пользователь взаимодействует с системой, посредством набора команд (рис. 2.5). В бесконечном цикле, программа требует ввести команду для выполнения, чтобы запустить соответствующую этой команде функцию. Пример корректной команды: рис. 2.6. Примеры набора некорректных команд: рис. 2.7.

```
130 err_c = 0
131 while True:
132     s = input('Введите команду: ')
133     if s: s = s.split()[0]
134     else: s = '!'
135     if s[0] != '!': s = '!' + s
136     if s in command_dict:
137         command_dict[s]()
138         err_c = 0
139     else:
140         err_c += 1
141         if err_c > 2:
142             print('Превышено количество попыток ввода, программа закрывается')
143             break
144         print('Команда введена неверно, попробуйте ещё раз!')
145         continue
```

Рисунок 2.5 — Бесконечный цикл, в котором вводятся команды

!exit — выходит из программы

Введите команду: all

Продукт	Категория	Цена	Дата покупки
огурцы соленые	овощи	249	12.10.2020
блинчики с мясом	блины	180	11.09.2018
хлеб	хлеб	100	23.11.2022
кефир	молочные продукты	65	15.11.2022
бананы	фрукты	64	13.06.2020
батон французский	хлеб	34	08.02.2019

Рисунок 2.6 — Пример ввода команды all для вывода всех записей

батон французский

хлеб

Введите команду: привет мир

Команда введена неверно, попробуйте ещё раз!

Введите команду: laksjdf

Команда введена неверно, попробуйте ещё раз!

Введите команду: команда 1

Превышено количество попыток ввода, программа закрывается

silva@TLVas-MacBook-Air-2: ~\$

Рисунок 2.7 — Пример обработки неверных команд

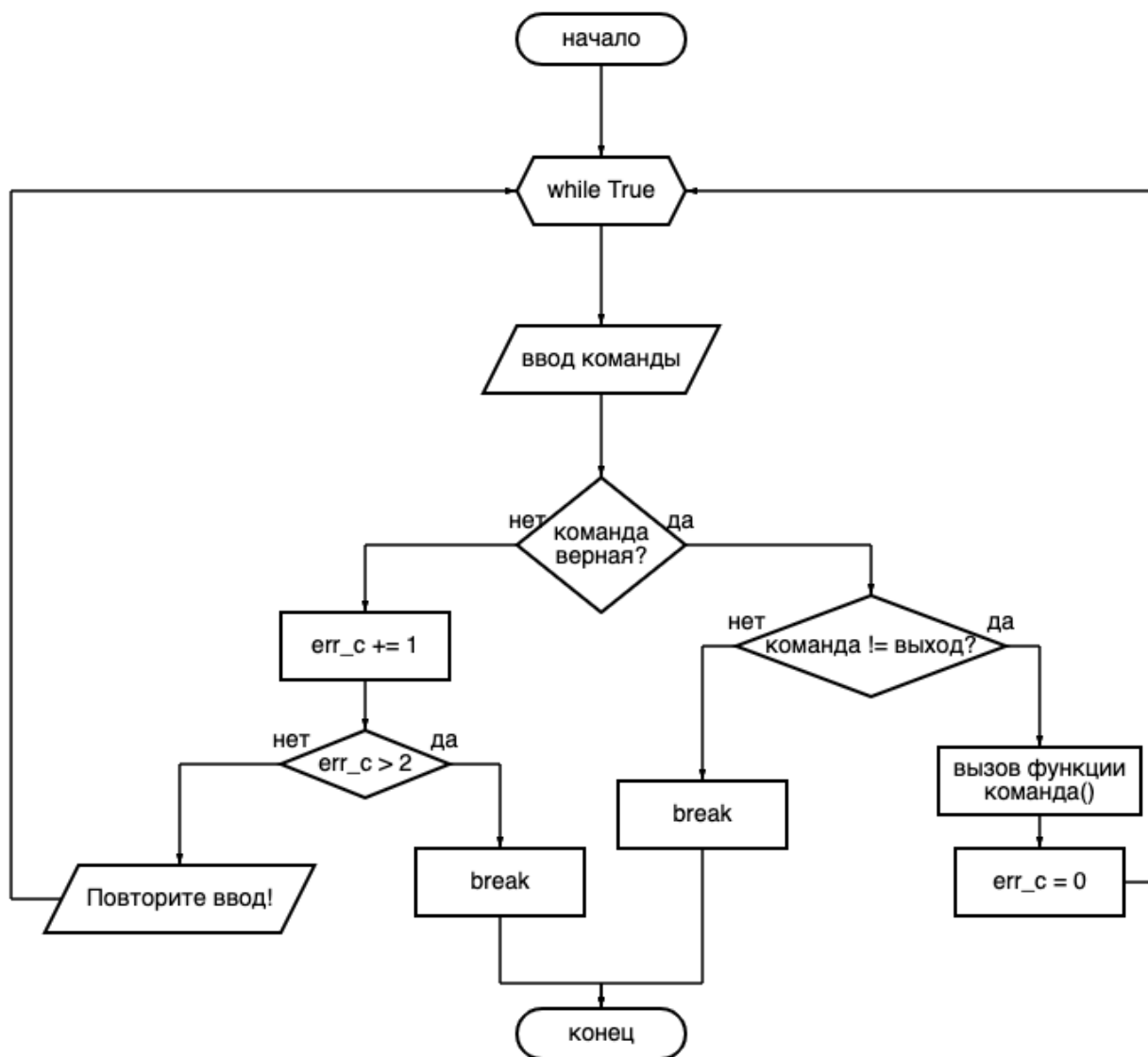


Рисунок 2.8 — Блок-схема кода с бесконечным циклом

Все взаимодействие, что может выполнять программа реализовано в отдельных функциях, которые вызываются при вводе соответствующей им команды. Рассмотрим функции по порядку.

Функция `add()` для добавления продукта в коллекцию (рис. 2.9) при вызове требует ввести название продукта, категорию, стоимость, дату покупки (рис. 2.10). Функция `add()` добавляет продукт в двумерный список `collection`, который хранит все данные о покупках.



```

11 def add():
12     global collection
13     product = input('Введите название продукта: ')
14     category = input('Введите название категории: ')
15     cost = input('Введите стоимость продукта (в рублях): ')
16     date = input('Введите корректную дату покупки в формате ДД.ММ.ГГГГ: ')
17     if not product or not category or not cost or not date or \
18         not cost.isnumeric() or not validate(date):
19         print('Ввод не распознан!')
20         return
21     print('Продукт добавлен в коллекцию!')
22     collection += [[int(cost), product, category, date]]

```

Рисунок 2.9 — Код функции add()

```

Введите команду: add
Введите название продукта: белый хлеб
Введите название категории: хлеб
Введите стоимость продукта (в рублях): 35
Введите корректную дату покупки в формате ДД.ММ.ГГГГ: 07.12.2022
Продукт добавлен в коллекцию!

```

Рисунок 2.10 — Пример работы команды add

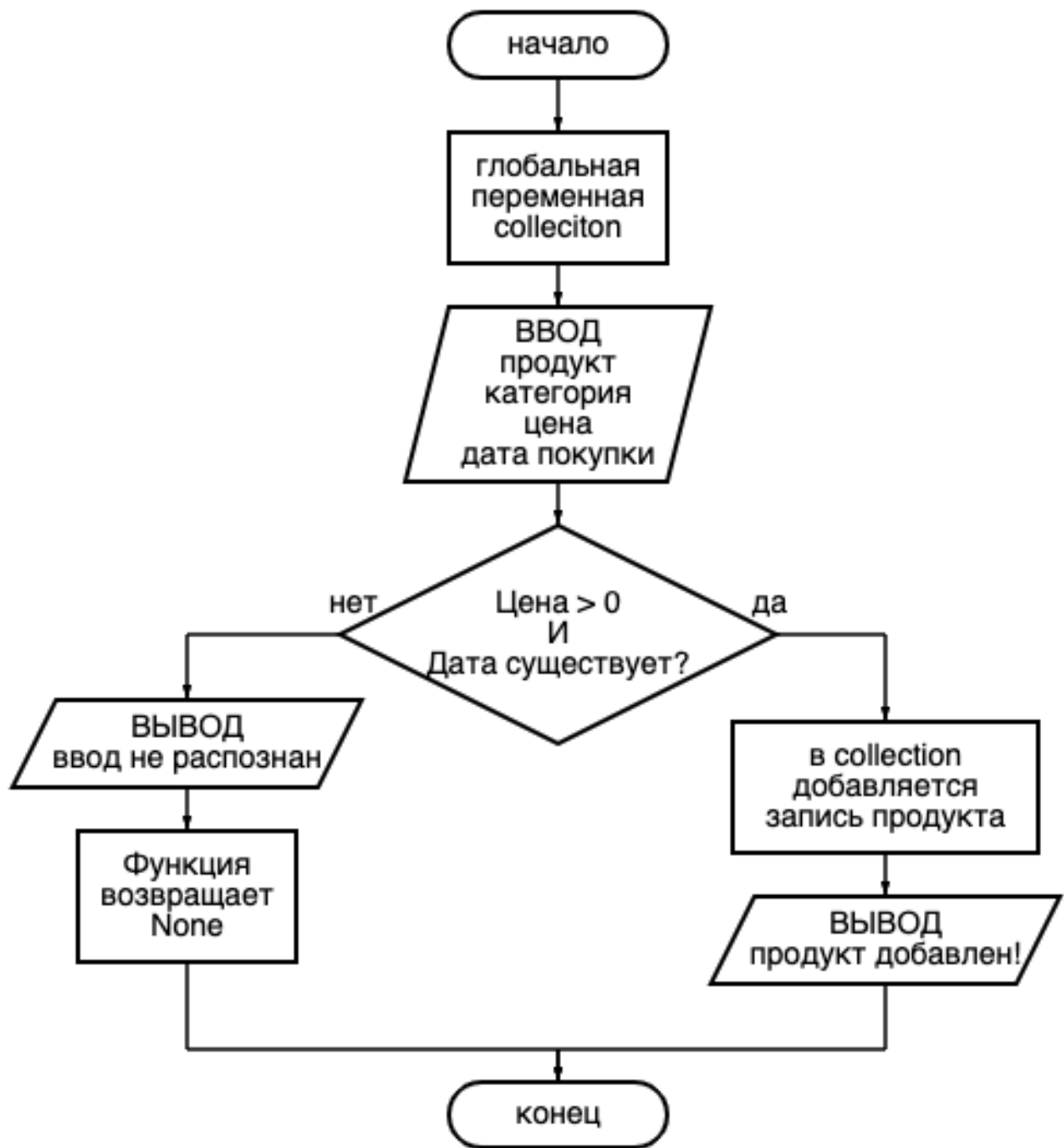


Рисунок 2.11 — Блок-схема функции add()

Функция All() для просмотра всех добавленных в коллекцию продуктов (рис. 2.12). Данная функция выводит все продукты, которые хранятся в двумерном списке collection. Для красивого вывода в виде таблицы (рис. 2.13) используются вспомогательные функции table\_title() и table\_item() (рис.2.14, 2.15).

```

34 def All():
35     table_title()
36     for item in collection:
37         table_item(item)
38     print()

```

Рисунок 2.12 — Код функции All()

Введите команду: all

Продукт	Категория	Цена	Дата покупки
огурцы соленые	овощи	249	12.10.2020
блинчики с мясом	блины	180	11.09.2018
хлеб	хлеб	100	23.11.2022
кефир	молочные продукты	65	15.11.2022
бананы	фрукты	64	13.06.2020
батон французский	хлеб	34	08.02.2019
огурцы соленые	овощи	100	07.12.2022
белый хлеб	хлеб	35	07.12.2022

Введите команду: █

Рисунок 2.13 — Пример работы команды all

```

29 def table_title():
30     print(f'{"Продукт":32s} {"Категория":32s} {"Цена":<10s} {"Дата покупки":15s}')
31     print('_'*89)
32

```

Рисунок 2.14 — Код функции table\_title()

```

25 def table_item(item):
26     print(f'{item[1]:32s} {item[2]:32s} {item[0]:<10d} {item[3]:15s}')
27

```

Рисунок 2.15 — Код функции table\_item()

Функция `byscd()` для просмотра продуктов по дате или категории (рис. 2.16). Данная функция требует от пользователя ввода названия категории или даты и выводит все записи в коллекции, которые имеют ту же дату или категорию (рис. 2.17, 2.18). Если в коллекции отсутствуют соответствующие продукты, то выводится таблица без продуктов (рис. 2.19).

```

41 def bycd():
42     flag = input('Просмотреть продукты по дате (0) или категории (1)? ')
43     if flag == '1':
44         cat = input('Введите категорию: ')
45         table_title()
46         for item in collection:
47             if item[2] == cat:
48                 table_item(item)
49     elif flag == '0':
50         date = input('Введите дату в формате ДД.ММ.ГГГГ: ')
51         if not validate(date):
52             print('Дата не распознана!')
53             return
54         table_title()
55         for item in collection:
56             if item[3] == date:
57                 table_item(item)
58     else:
59         print('Ввод не распознан!')
60     print()

```

Рисунок 2.16 — Код функции bycd()

Введите команду: bycd

Просмотреть продукты по дате (0) или категории (1)? 0

Введите дату в формате ДД.ММ.ГГГГ: 07.12.2022

Продукт	Категория	Цена	Дата покупки
огурцы соленые	овощи	100	07.12.2022
белый хлеб	хлеб	35	07.12.2022

Рисунок 2.17 — Пример работы команды bycd по дате

Введите команду: bycd

Просмотреть продукты по дате (0) или категории (1)? 1

Введите категорию: хлеб

Продукт	Категория	Цена	Дата покупки
хлеб	хлеб	100	23.11.2022
батон французский	хлеб	34	08.02.2019
белый хлеб	хлеб	35	07.12.2022

Рисунок 2.18 — Пример работы команды bycd по категории

Введите команду: bycd

Просмотреть продукты по дате (0) или категории (1)? 0

Введите дату в формате ДД.ММ.ГГГГ: 19.12.2022

Продукт

Категория

Цена

Дата покупки

Введите команду: █

Рисунок 2.19 — Пример работы команды bycd

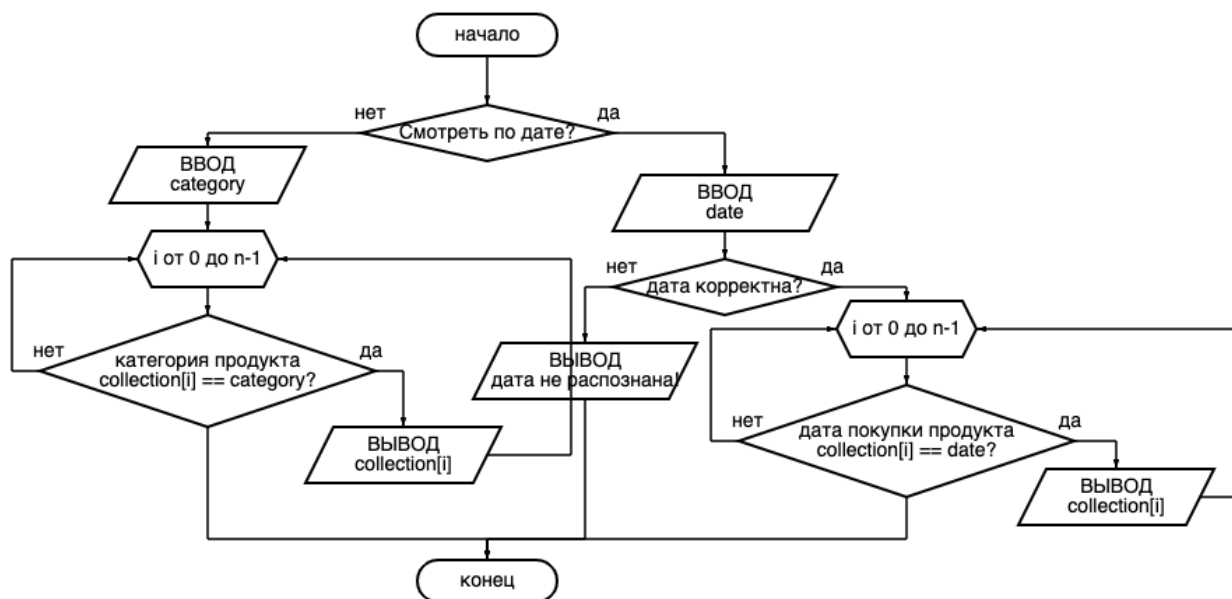


Рисунок 2.20 — Блок-схема функции bycd()

Функция Sort() для сортировки продуктов в коллекции по убыванию / возрастанию (рис. 2.21). Пользователь выбирает, как хочет сортировать продукты, программа выводит все продукты (рис. 2.22). Для вывода вызывается ранее описанная функция All().

```
63 def Sort():
64     global collection
65     flag = input('Сортировать стоимость продуктов по убыванию (0) или по возрастанию (1)? ')
66     if flag == '1':
67         collection.sort()
68     elif flag == '0':
69         collection.sort(reverse=True)
70     else:
71         print('Ввод не распознан!')
72         return
73     All()
```

Рисунок 2.21 — Код функции Sort()

Введите команду: sort

Сортировать стоимость продуктов по убыванию (0) или по возрастанию (1)? 1

Продукт	Категория	Цена	Дата покупки
батон французский	хлеб	34	08.02.2019
белый хлеб	хлеб	35	07.12.2022
бананы	фрукты	64	13.06.2020
кефир	молочные продукты	65	15.11.2022
огурцы соленые	овощи	100	07.12.2022
хлеб	хлеб	100	23.11.2022
блинчики с мясом	блины	180	11.09.2018
огурцы соленые	овощи	249	12.10.2020

Введите команду: █

Рисунок 2.22 — Пример работы команды sort

Функция Del() для удаления продукта из коллекции (рис. 2.23). Пользователь вводит номер продукта, который желает удалить из коллекции (рис. 2.24). Если пользователь ввел существующий номер продукта, то при выводе коллекции его там не окажется (рис. 2.25).

```
76 def Del():
77     for i in range(len(collection)):
78         item = collection[i]
79         print(f'{str(i+1)}. [{item[1]}, {item[2]}, {item[0]}, {item[3]}]')
80     flag = input('Введите номер записи, которую желаете удалить: ')
81     if flag.isnumeric() and 1 <= int(flag) <= len(collection):
82         flag = int(flag)-1
83         item = collection.pop(flag)
84         print(f'Запись [{item[1]}, {item[2]}, {item[0]}, {item[3]}] удалена из коллекции')
85         print()
86     else:
87         print('Ввод не распознан!')
88
```

Рисунок 2.23 — Код функции Del()

Введите команду: del

1. [батон французский, хлеб, 34, 08.02.2019]
2. [белый хлеб, хлеб, 35, 07.12.2022]
3. [бананы, фрукты, 64, 13.06.2020]
4. [кефир, молочные продукты, 65, 15.11.2022]
5. [огурцы соленые, овощи, 100, 07.12.2022]
6. [хлеб, хлеб, 100, 23.11.2022]
7. [блинчики с мясом, блины, 180, 11.09.2018]
8. [огурцы соленые, овощи, 249, 12.10.2020]

Введите номер записи, которую желаете удалить: 4

Запись [кефир, молочные продукты, 65, 15.11.2022] удалена из коллекции

Рисунок 2.24 — Пример работы команды del

Введите номер записи, которую желаете удалить: 4

Запись [кефир, молочные продукты, 65, 15.11.2022] удалена из коллекции

Введите команду: all

Продукт	Категория	Цена	Дата покупки
батон французский	хлеб	34	08.02.2019
белый хлеб	хлеб	35	07.12.2022
бананы	фрукты	64	13.06.2020
огурцы соленые	овощи	100	07.12.2022
хлеб	хлеб	100	23.11.2022
блинчики с мясом	блины	180	11.09.2018
огурцы соленые	овощи	249	12.10.2020

Рисунок 2.25 — Проверка удаления командой all



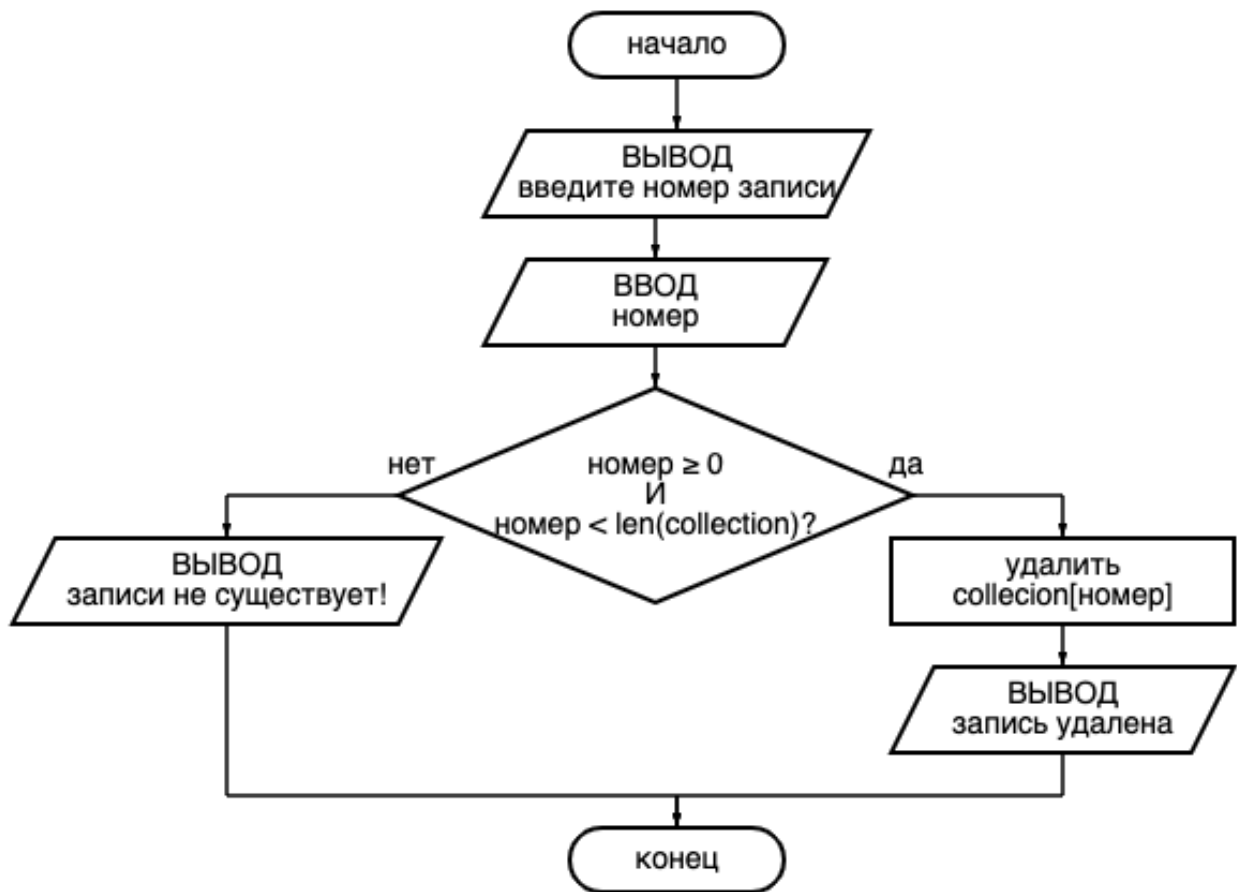


Рисунок 2.26 — Блок-схема функции Del()

Функция save() сохраняет все продукты в коллекции (рис. 2.27). Файл data.txt, который лежит в директории программы, перезаписывается данными из списка collection. Продукт «белый хлеб», который был добавлен ранее, успешно записался в файл (рис. 2.29)

```

98  def save():
99      with open('data.txt', 'w') as w:
100         for item in collection:
101             w.write(f'{item[0]}_{item[1]}_{item[2]}_{item[3]}\n')
102     print('Коллекция сохранена в файл!')
103     print()

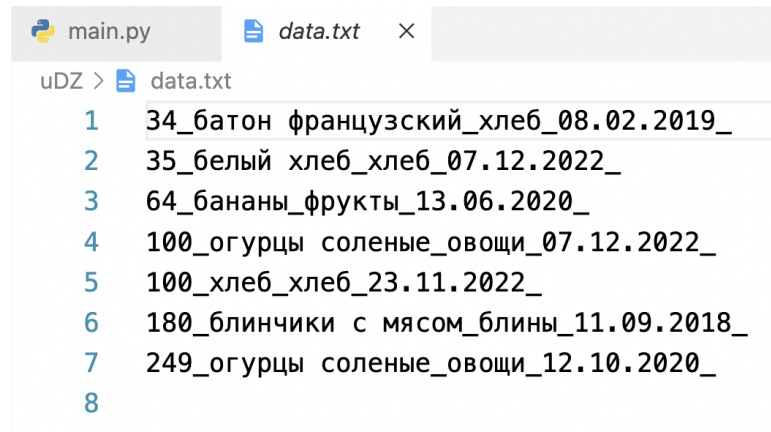
```

Рисунок 2.27 — Код функции save()



```
Введите команду: save
Коллекция сохранена в файл!
```

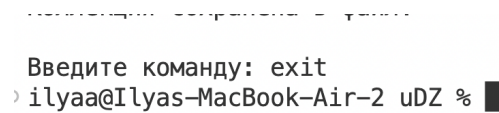
Рисунок 2.28 — Вывод команды save

A screenshot of a code editor with two tabs: 'main.py' and 'data.txt'. The 'data.txt' tab is active, showing a list of items with their counts and dates. The items are: 34\_батон французский\_хлеб\_08.02.2019\_, 35\_белый хлеб\_хлеб\_07.12.2022\_, 64\_бананы\_фрукты\_13.06.2020\_, 100\_огурцы соленые\_овощи\_07.12.2022\_, 100\_хлеб\_хлеб\_23.11.2022\_, 180\_блинчики с мясом\_блины\_11.09.2018\_, and 249\_огурцы соленые\_овощи\_12.10.2020\_. The list is numbered 1 through 8.

```
uDZ > data.txt
1 34_батон французский_хлеб_08.02.2019_
2 35_белый хлеб_хлеб_07.12.2022_
3 64_бананы_фрукты_13.06.2020_
4 100_огурцы соленые_овощи_07.12.2022_
5 100_хлеб_хлеб_23.11.2022_
6 180_блинчики с мясом_блины_11.09.2018_
7 249_огурцы соленые_овощи_12.10.2020_
8
```

Рисунок 2.29 — Содержимое файла data.txt после выполнения функции

Когда пользователь закончит работу с программой он может ввести команду `exit` и программа завершит работу (рис. 2.30)

A screenshot of a terminal window. It shows a prompt 'ilyaa@Ilyas-MacBook-Air-2 uDZ %' and the command 'exit' being entered. Above the command, there is a faint, partially visible text 'Введите команду: exit'.

```
Введите команду: exit
ilyaa@Ilyas-MacBook-Air-2 uDZ %
```

Рисунок 2.30 — Команда exit

### **3 Вывод**

Цель работы была достигнута. Было создано программное обеспечение системы обработки данных в виде интерактивного консольного приложения в Python3: «Программа для контроля собственных денежных средств».

Были описаны этапы анализа предметной области и требований. Было описано решение задания с указанием элементов программирования. Были представлены графические материалы, блок-схемы алгоритмов. Была продемонстрирована работа программы.