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

**Домашнее задание**

**Реализация программной модели инфокоммуникационной системы**

Выполнил: Лазуренко

Александр Витальевич

Группа № K3121

Проверила: Казанова

Полина Петровна

Санкт-Петербург

2023

**Цель работы:**

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

**Задачи:**

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.

**Ход работы:**

**Описание этапа анализа предметной области и требований:**

Предстояло реализовать приложение, способное выводить информацию о купленных товаров. Поэтому было решено создать класс «продукт», обладающий следующими свойствами:

* индекс,
* название,
* цена,
* категория,
* дата покупки.

Это минимальный набор параметров, необходимый для выполнения ТЗ. Прототип взаимоотношений объектов в программе изображен в Приложении А на [рисунке 1](#рис1)*.*

В ходе анализа ТЗ было принято решение реализовать следующие основные функции:

* вывод таблицы с информацией об о всех продуктах,
* добавление нового продукта в коллекцию (пользователь может ввести все параметры продукта),
* удаление продукта из коллекции по индексу или названию,
* вывод всех продуктов, относящихся к одной категории (категорию выбирает пользователь)
* вывод всех продуктов, купленных в один день (дату указывает пользователь)
* вывод всех продуктов в порядке возрастания/убывания цены (формат вывода указывает пользователь),
* сохранение коллекции в файл.

Был продуман прототип интерфейса и коды ввода для команд. Было принято решение реализовать интерфейс в терминале, чтобы улучшить навыки обработки исключений и формата ввода, так как утилиты для реализации оконного интерфейса (например PyQt и QtDesigner) упрощают процесс обработки исключений.

**Описание решения задания с указанием применения элементов программирования.**

В ходе разработки приложения распределил функции на 5 категорий, создав под них соответствующие .py файлы:

* main – функции обработки ввода основных команд и запуск приложения
* myfunctions – реализация основных функций приложения, вызываемых кодовыми командами из main
* storage – расположение массива со всеми объектами продуктов, сохраняемые индивидуальные настройки, функции управления массивом, сохранения и загрузки данных
* product – описание класса продукта
* customoutput – функция форматированного вывода текста и ввода с ограничением количества символов

**Скрипт main**

В скрипте реализованы функции ожидания ввода и обработки введенных команд. Список доступных для ввода команд реализован в скрипте myfunctions в формате словаря, где ключам- индексам сопоставлены значения – строки.

Блок-схема скрипта main представлена в Приложении Б на [рисунке 2](#рис2).

Программа не учитывает регистр введенных команд, что уменьшает количество исключений, а также по возможности обрабатываются варианты ввода идентичных по отображению в терминале команд на разных раскладках клавиатуры, например “X” на русской и английской раскладке.

Общение с пользователем реализовано подсказками на русском языке. Основные функции вызываются командами на английском языке.

Пример запуска программы представлен в Приложении Г на [рисунке 10](#рис10).

**Скрипт (Класс) Product**

В скрипте определен класс продукта (товара), который принимает и при инициализации обрабатывает следующие параметры:

* id – int, определяется автоматически на основе длины массива всех продуктов,
* name – string, вводится пользователем, имеет ограничение на длину: 25 символов,
* cost – float, вводится пользователем, имеет ограничение в 10 символов,
* category – string, вводится пользователем, при отсутствии введенных данных автоматически выставляется категория «none», имеет ограничение в 10 символов,
* дата покупки – string, вводится пользователем, введенные данные обрабатываются и переводятся в формат даты «ДД.ММ.ГГГГ» с помощью библиотек datetime и maya. Если введенные данные отсутствуют, автоматически выставляется сегодняшняя дата.

Код представлен в Приложении В на [рисунке 5](#рис5).

**Скрипт storage**

В скрипте определен массив с объектами продуктов и пользовательские настройки вывода (настройки сортировки)

Реализована функция удаления продукта по индексу и имени с обработкой исключений и переназначением индексов товаров.

Также реализованы функции сохранения данных в формате json и из загрузки, соответственно.

В функции сохранения создается словарь, где в качестве ключей указаны названия параметров, а в качестве значений – соответствующие переменные (строки со значениями пользовательских настроек и список со всеми продуктами). Очередной продукт также сохраняется в формате словаря: ключу-названию параметра класса сопоставляется его прямое значение.

При загрузке файла происходит обратный процесс распаковки словаря и записи его значений в переменные.

Код функций сохранения и загрузки данных представлен в приложении В на [рисунке 6](#рис6).

**Скрипт myfunctions**

В ходе разработки список основных команд и функций был расширен для улучшения взаимодействия пользователя с приложением.

Основные команды и соответствующие функции:

* Menu – последовательно выводит информацию о всех доступных командах и их назначении
* View – Для вывода используется библиотека tabulate. В цикле формируется массив, подходящий по формату под требования библиотеки и выводится соответствующая таблица. Заголовки сохранены в константе. Пример вывода таблицы товаров представлен в Приложении Г на [рисунке 12](#рис12)
* Autoview – Опция, которая позволяет автоматически выводить таблицу после ее изменения (добавления и удаления товаров). Содержит варианты с выводом сортированной таблицы. Эта пользовательская настройка хранится в формате кортежа (формат вывода, флаг – включен ли автовывод) и сохраняется в json с остальными настройками. Код функции представлен в Приложении В на [рисунке 7](#рис7).
* Add – добавляет новый продукт в список. Последовательно выводятся подсказки с полями для ввода и в цикле идет обработка введенных данных. Если данные не соответствуют необходимому формату, процесс их ввода повторяется. Блок-схема работы этой функции представлена в Приложении Б на [рисунке 3](#рис3). В результате новый объект продукта добавляется в хранилище и выводится сообщение о его успешном создании. Пример работы этой функции представлен в Приложении Г на рисунке 11. Код функции представлен в Приложении В на [рисунке 8](#рис8).
* Delete – пользователю выводятся все товары и предлагается ввести индекс или название товара, который он хочет удалить. По формату введенных данных определяется тип переменной и происходит обращение к скрипту storage. Если переданные параметры есть в таблице, то этот элемент удаляется. Иначе выводится сообщение о том, что указанный товар не найден в таблице. Также, как и в функции добавления товара есть вариант с финальным выводом обновленной таблицы. Код функции представлен в Приложении В на [рисунке 9](#рис9).
* SortC – сортировка таблицы по цене. Пользователь вводит опцию – выводить по возрастанию цены (min) или по убыванию (max). Введенные команды обрабатываются на ошибки, формат вывода сохраняется как пользовательская настройка. Новая таблица формируется сортировкой хранилища по цене. Есть возможность автоматически выводить таблицу с этой настройкой после ее изменения.
* SortK – сортировка по категории. Программа выводит подсказку со всеми известными категориями, и пользователь вводит нужную ему. Если введенной категории нет среди товаров, программа предлагает попробовать еще раз. Есть возможность досрочно завершить функцию, введя «Х». Таблица для вывода формируется выборкой из хранилища товаров с нужной категорией. Есть возможность автоматически выводить таблицу с этой настройкой после ее изменения.
* SortD – сортировка по дате. Программа предлагает ввести дату, проверка ее соответствия формату происходит в отдельной функции, которая также используется в функции Add. Блок-схема данной функции представлена в Приложении Б на [рисунке 4](#рис4). Таблица для вывода формируется выборкой из хранилища товаров с нужной датой. Если в этот день не было совершено покупок, выводится соответствующее сообщение. Есть возможность автоматически выводить таблицу с этой настройкой после ее изменения.
* Exit – завершение работы приложения с сохранением данных
* Save – сохранение данных в json (описана в storage)

Скрипт customoutput

Содержит 2 функции:

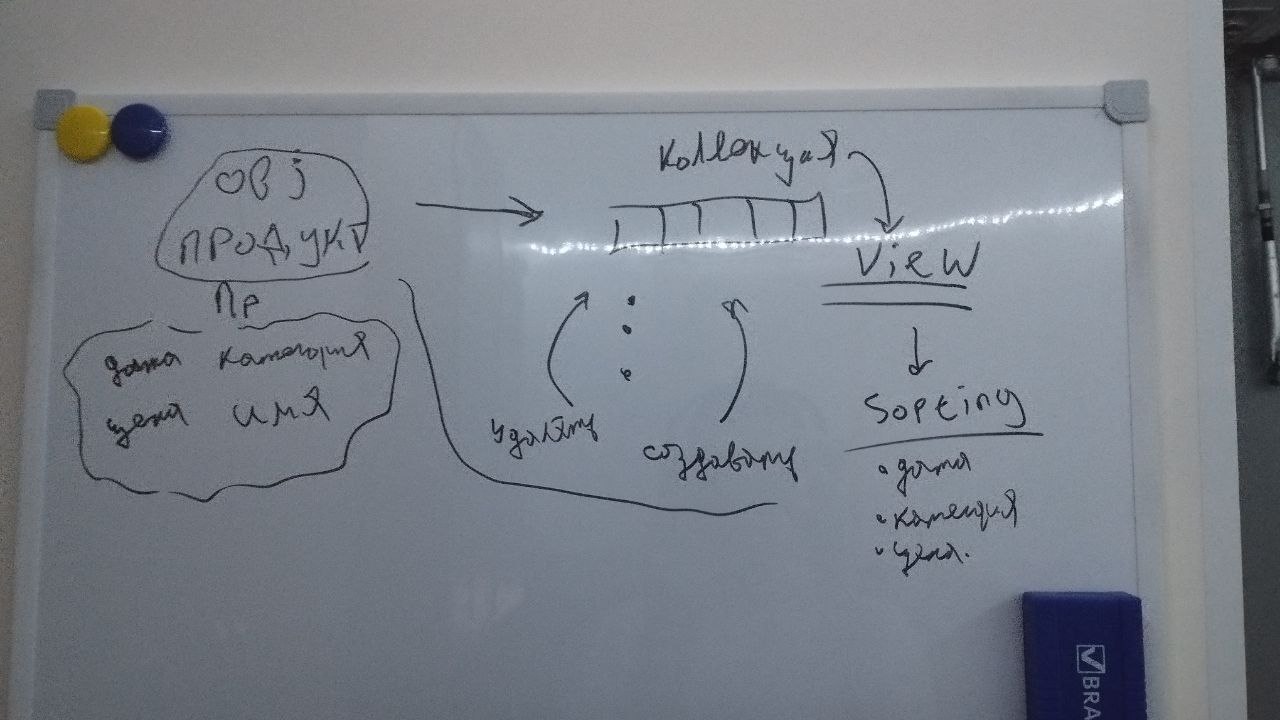
1. Print\_with\_lines – форматирует введенную строку так, чтобы при выводе она была ограничена линиями из символов. Имеет параметры:
   1. text – форматируемая строка,
   2. linelen – длина линий-ограничителей, по умолчанию выставляет длину, соответствующую длине текста,
   3. symbol – символ, из которого будет строиться строка, по умолчанию равен “-“,
   4. needspace – нужно ли делать отступ после вывода, по умолчанию включен.
2. Liminput - переопределение функции input с добавлением нового параметра – максимального числа символов. При вводе обрезает строку до указанной длины. По умолчанию ограничение – 10 символов.

**Вывод:**

В ходе выполнения задания создал Программу для контроля собственных денежных средств в формате консольного приложения, улучшил свои навыки обработки исключений, общения с пользователем через программу и форматирования вводимых данных.

Ссылка на GitHub репозиторий: https://github.com/GeraDot4815/CostyanHW

**Приложение А:**



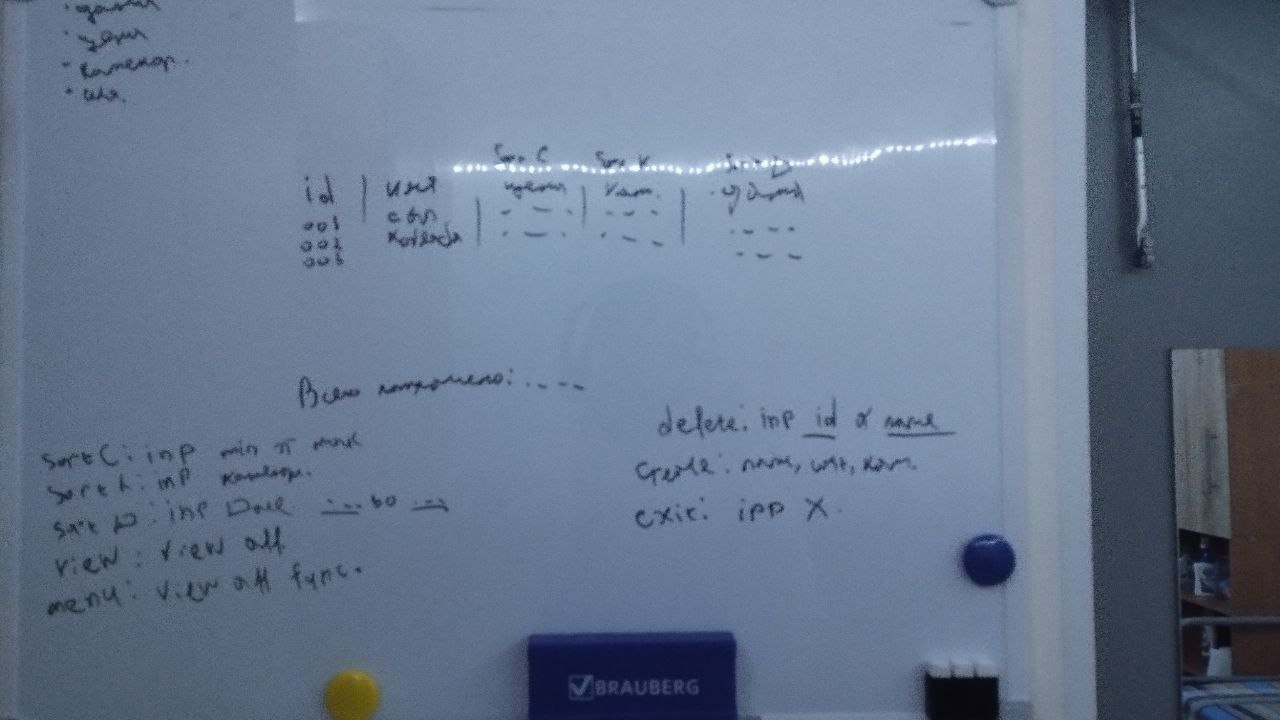


Рисунок 1 – прототип архитектуры проекта

**Приложение Б**

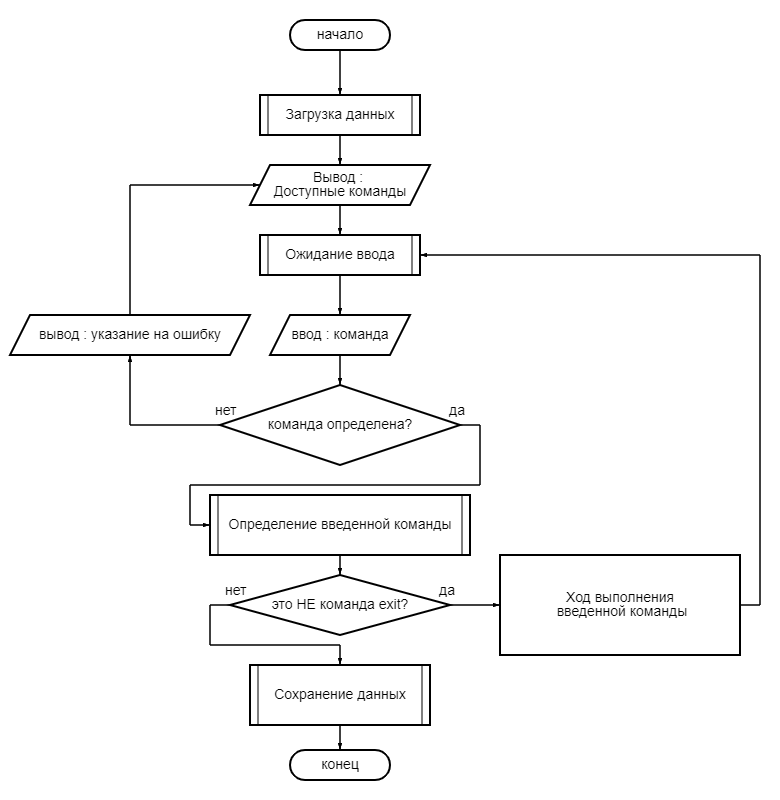


Рисунок 2 – блок-схема обработки ввода в приложении

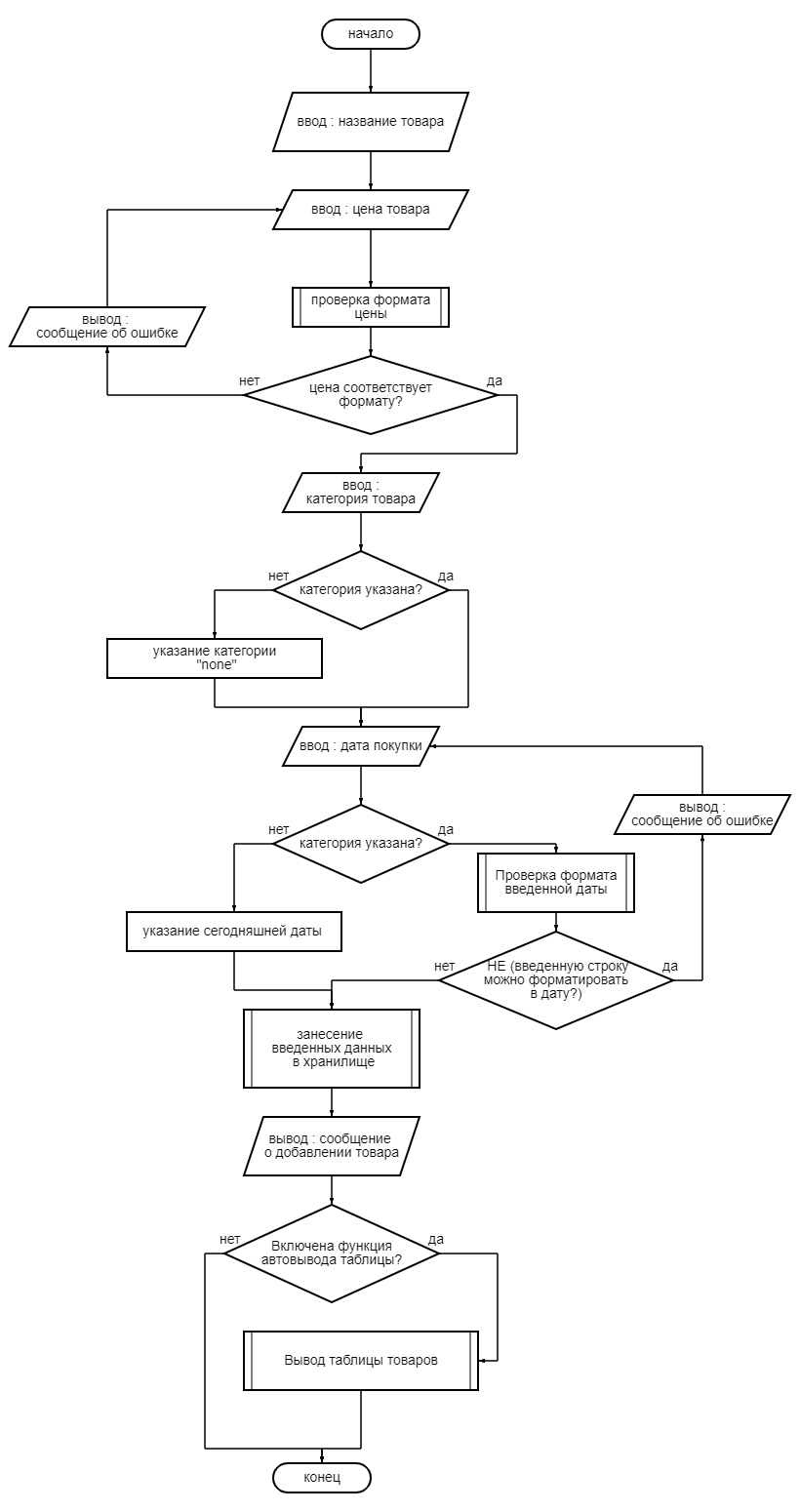


Рисунок 3 – Блок-схема функции добавления нового товара

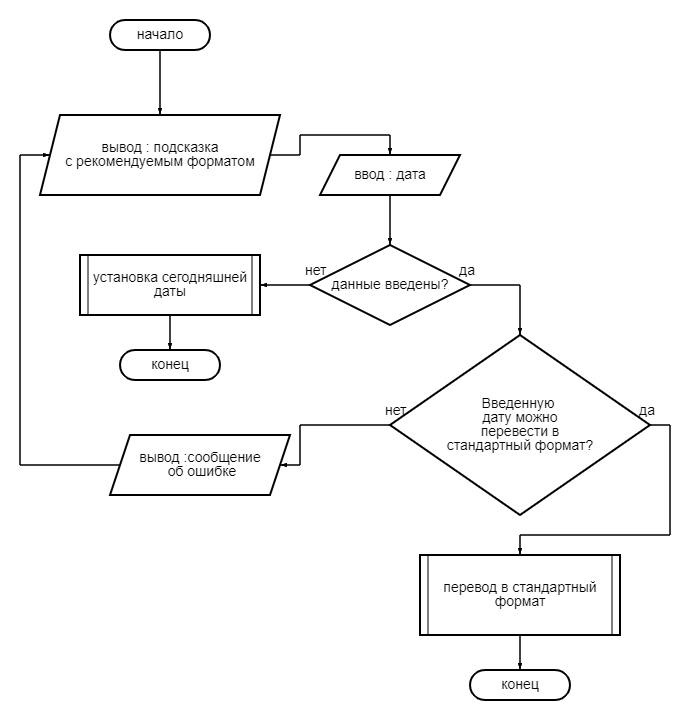


Рисунок 4 – Блок-схема функции проверки вводимой даты

**Приложение В**

import datetime  
import maya  
  
class Product:  
 def \_\_init\_\_(self, id, name, cost, category=None, date=None):  
 self.id = id  
 self.name = name  
 self.cost = cost  
 self.category = category if category!=None else "none"  
 self.date = maya.parse(date).datetime().date().strftime("%d.%m.%Y") if date!=None\  
 else datetime.date.today().strftime("%d.%m.%Y")

Рисунок 5 – код класса Product

def save\_data():  
 data={}  
  
 data['products']=[]  
 for pr in productlist:  
 (data['products'].append  
 ({  
 'id': pr.id,  
 'name': pr.name,  
 'cost': pr.cost,  
 'category': pr.category,  
 'date': pr.date  
 }))  
  
 data['costfilter']=filtercost  
 data['datefilter']=filterdate  
 data['catfilter']=filtercategory  
 data['autotab']=autotab  
  
 with open('data.txt', 'w') as outfile:  
 json.dump(data, outfile)  
  
 print("\*Данные упешно соранены\*")  
def load\_data():  
 global productlist  
 global filtercost  
 global filterdate  
 global filtercategory  
 global autotab  
  
 with open('data.txt') as json\_file:  
 data = json.load(json\_file)  
 productlist=[]  
 for spr in data['products']:  
 npr=Product(spr['id'], spr['name'], spr['cost'], spr['category'], spr['date'])  
 productlist.append(npr)  
 filtercost = data['costfilter']  
 filterdate = data['datefilter']  
 filtercategory = data['catfilter']  
 autotab = data['autotab']  
  
 print("\*Данные упешно загружены\*")

Рисунок 6 – функции сохранения и загрузки данных

def auto\_view\_settings():  
 valids="ackdf"  
 tip=co.print\_with\_lines("A - выводить всю таблицу"+'\n'+  
 "C - выводить с сортировкой по цене"+'\n'+  
 "K - выводить с сортировкой по категории"+'\n'+  
 "D - выводить с сортировкой по дате"+'\n'+  
 '\n'+"F - отключить автовывод", linelen=len("K - выводить с сортировкой по категории"))  
 wrongtip="!Опция введена неверно!"+'\n'+"Для ввода доступны команды: A, C, K, D"  
 inptip="Выберите одну из опций --> "  
 print(tip)  
 inp=co.liminput(inptip, 1).lower()  
 while not inp in valids:  
 print(wrongtip)  
 inp=co.liminput(inptip, 1).lower()  
 canview=False if inp=="f" else True  
  
 storage.autotab=(canview, inp)  
 auto\_view\_tab(storage.autotab)  
def auto\_view\_tab(settings):  
 if settings[0]==False: return  
 elif settings[1]=="a":  
 view\_all\_table()  
 elif settings[1]=="c":  
 view\_sort\_cost(storage.filtercost)  
 elif settings[1]=="k":  
 view\_sort\_category(storage.filtercategory)  
 elif settings[1]=="d":  
 view\_sort\_date(storage.filterdate)

Рисунок 7 – функции авто вывода таблицы

def add\_element():  
 def check\_cost\_format(cost: str):  
 digits = "0123456789"  
 dots = ".,"  
  
 dcost = ""  
 for i in cost:  
 if i in digits or i in dots: dcost = dcost + i  
 if dcost=="": return (False, -1)  
 dcost = dcost.replace(",", ".")  
 try:  
 dcost=float(dcost)  
 except ValueError:  
 return (False, -1)  
 return (True, dcost)  
  
 nametip="Ведите название товара --> "  
  
 costtip="Введите цену товара --> "  
 wrongcosttip=("!Цена введена неправильно!"+'\n'+  
 "Введите цену в одном из указанных форматов"+'\n'+  
 "000"+'\n'+  
 "00.00"+'\n'+  
 "00,00"+'\n')  
  
 categorytip="Введите категорию товара (Enter - пропустить) --> "  
 datetip = "Введите дату покупки товара (Enter - сегодня) --> "  
  
 name = co.liminput(nametip, 25)  
 cost = co.liminput(costtip)  
 check\_result=check\_cost\_format(cost)  
 while not check\_result[0]:  
 print(wrongcosttip)  
 cost = co.liminput(costtip)  
 check\_result = check\_cost\_format(cost)  
  
  
 cost=check\_result[1]  
  
 category = co.liminput(categorytip) or None  
 date = enter\_date\_with\_check(datetip)  
  
 idx=len(storage.productlist)  
 pr = Product(idx, name, cost, category, date)  
 storage.add\_product(pr)  
  
 print(co.print\_with\_lines("Товар успешно добавлен!"))  
 if (storage.autotab[0]): auto\_view\_tab(storage.autotab)

Рисунок 8 - функция добавления нового товара в колекцию

def delete\_product(id=-1, name="-"):  
 errortip = co.print\_with\_lines("Простите, товара с таким именем или индексом не найдено", symbol="?")  
 if id!=-1:  
 if id<0 or len(productlist)<id:  
 print(errortip)  
 else:  
 productlist.pop(id)  
 reset\_idxs()  
 return  
 elif name!="-":  
 for i in range(len(productlist)):  
 if productlist[i].name==name:  
 productlist.pop(i)  
 reset\_idxs()  
 return  
 print(errortip)  
 else:  
 print(errortip)  
def reset\_idxs():  
 for i in range(len(productlist)):  
 productlist[i].id=i

Рисунок 9 – Функция удаления товара из коллекции

**Приложение Г**

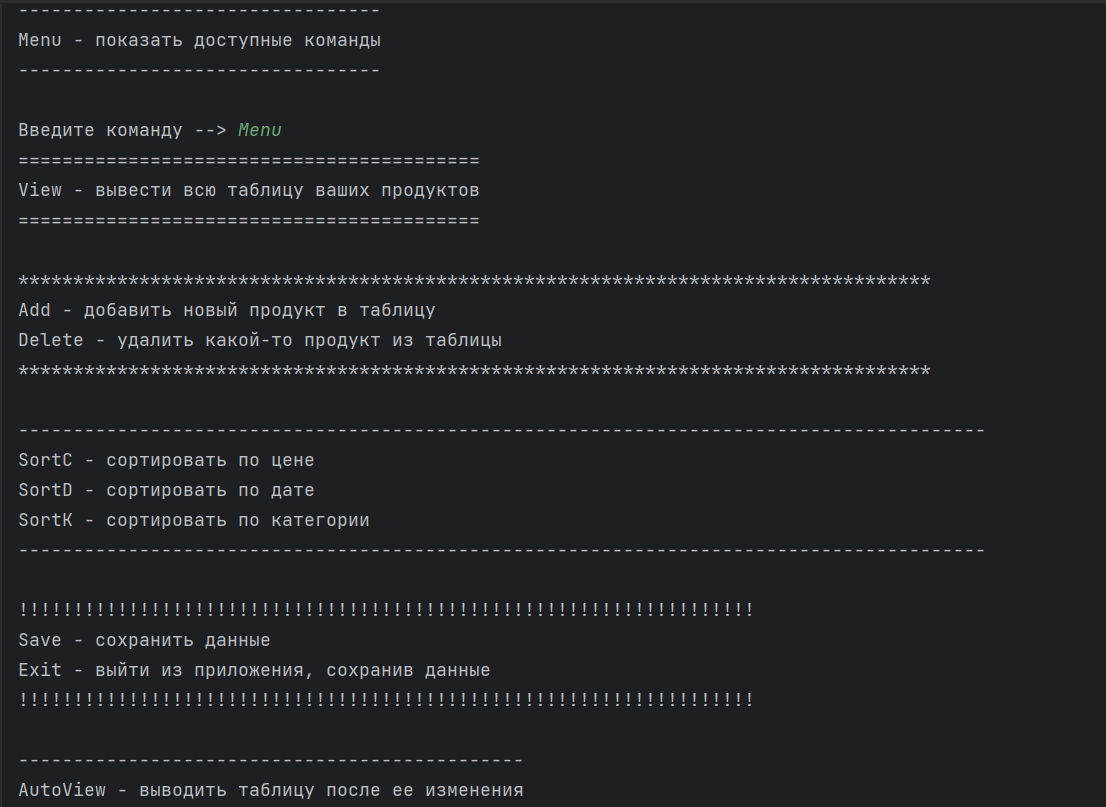


Рисунок 10 – Запуск приложения с выводом доступных функций.

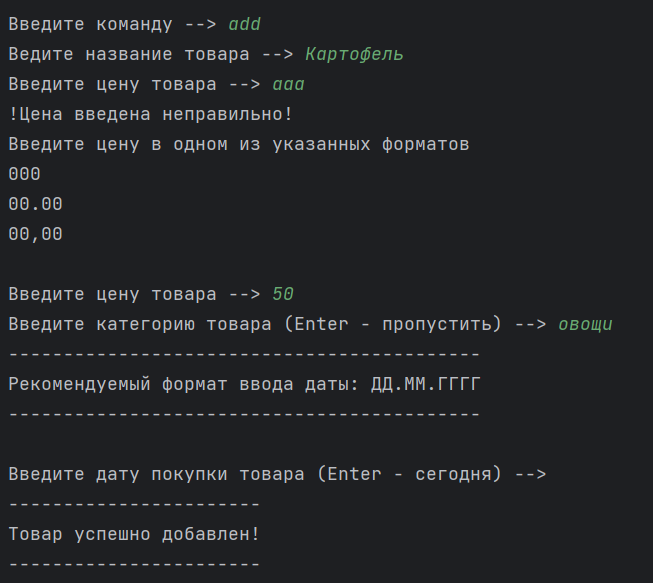


Рисунок 11 – пример добавления нового товара

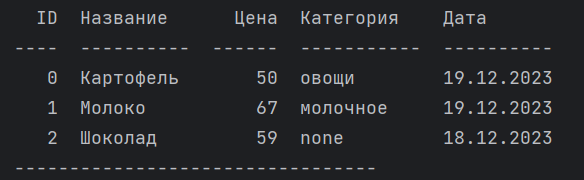


Рисунок 12 – пример вывода таблицы