

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

Лабораторная работа 1

Реализация программной модели инфокоммуникационной системы

Выполнил: Касьяненко
Вера Михайловна
Группа № K3121
Проверила: Казанова
Полина Петровна

Санкт-Петербург
2022

Цель работы:

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

Задачи:

- Провести анализ предметной области и требований.
- Придумать алгоритмы для решения конкретных задач.
- Написать программу для данных алгоритмов на языке Python.
- Получить требуемый результат, убедившись в работоспособности программы.

Задание:

Создать программное обеспечение системы обработки данных: «Программа для контроля собственных денежных средств» со следующими функциями:

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.

Ход работы:

Перед началом создания алгоритма необходимо провести анализ предметной области и требований. Наиболее похожим программным обеспечением являются приложения банков, которые позволяют контролировать расходы. Однако в отличие от приложений банков, в программном обеспечении, которое предстоит создать, пользователь сам вводит данные о покупках. Подобные

приложения чаще всего используется обычными пользователями, которые не обладают навыками программиста, поэтому необходимо создать программу, которая будет понятна и проста в использовании. Также программа должна выполнять поставленные перед ней задачи, например создание и просмотр отчетов о покупках, совершенных пользователем.

Для того, чтобы программа функционировала, необходимо создать файлы с базами данных, которые будут использоваться в дальнейшем:

- 1) Файл с именем пользователя.
- 2) Файл с базой продуктов, их стоимостью и категорией.
- 3) Файл записями о покупках.
- 4) Файл с записью о текущем балансе.

Вся программа состоит из функций, которые вызываются при необходимости. Рассмотрим их:

- 1) Функция приветствия пользователя. Программа считывает имя из файла в переменную (рисунок 1) и выдает его при начале работы программы (рисунок 2). Если имени в файле не обнаружено, то программа печатает приветствие на экран без имени (рисунок 3).

```
def begin():  
    with open('name.txt', 'r', encoding='UTF-8') as name:  
        name = name.read()  
    if name != '':  
        print(f'\nC возвращением, {name}!')  
        main()  
    else:  
        print(f'\nC возвращением!')  
        main()
```

Рисунок 1 – Функция, показывающая приветствие

C возвращением, Вера!

Рисунок 2 – Приветствие с именем

С возвращением!

Рисунок 3 – Приветствие без имени

Более того, в программе предусмотрено изменение имени. На рисунке 4 наглядно представлен алгоритм изменения имени, а также функция на рисунке 5 и ее демонстрация на рисунке 6.

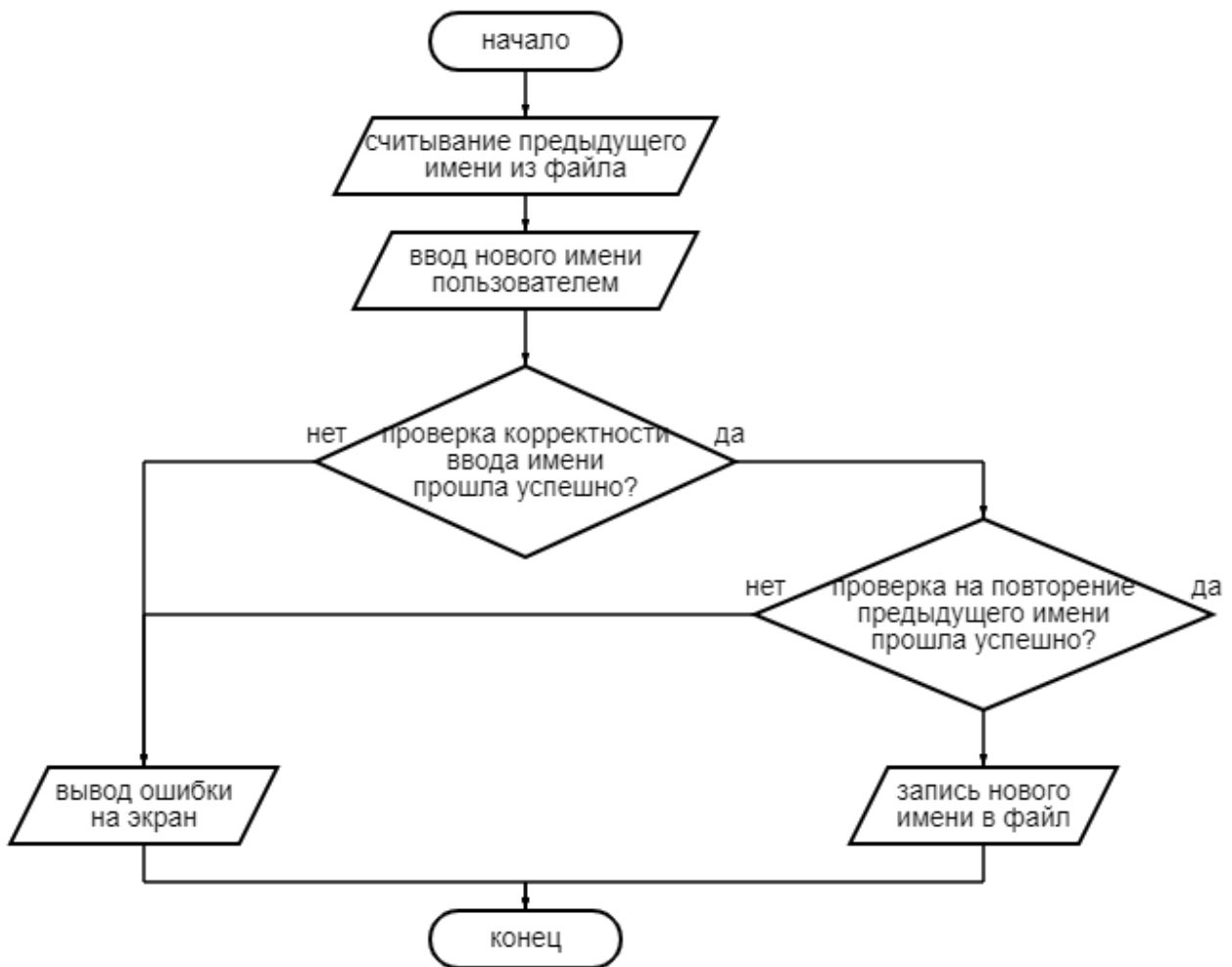


Рисунок 4 – Алгоритм изменения имени

```
def ch_name():
    print('-----\n')
    with open('name.txt', 'r', encoding='UTF-8') as name:
        name = name.read()
    new_name = input('Введите новое имя, которое будет появляться при входе: ')
    if new_name != '.' and new_name != ',' and new_name != ' ':
        if new_name != name:
            with open('name.txt', 'w', encoding='UTF-8') as name:
                name.write(new_name)
            print('Имя было успешно обновлено!')
        else:
            print('Ошибка! Это имя уже используется.')
    else:
        print('Ошибка! Неправильный ввод.')
main()
```

Рисунок 5 – Функция изменения имени

```
Введите новое имя, которое будет появляться при входе: Маша
Имя было успешно обновлено!
```

Рисунок 6 – Демонстрация изменения имени

При вызове функции изменения имени введенное имя записывается в переменную, после чего проверяется на корректность ввода и повторение предыдущего имени при помощи условных операторов, если проверки пройдены, то новое имя записывается в файл, иначе выводится сообщение об ошибке (рисунок 7).

```
Введите новое имя, которое будет появляться при входе: Маша
Ошибка! Это имя уже используется.
```

Рисунок 7 – Ошибка, если имя совпадает с предыдущим

- 2) Функция просмотра (рисунок 8) и изменения баланса (рисунок 9). Для лучшего контроля расходов была написана функция, которая считывает сумму, добавленную пользователем в приложение, и работает с ней. Данные о балансе так же, как и имя, хранятся в файле. При работе с

приложением пользователь может просмотреть или изменить свой баланс.

```
def balance():
    print('-----\n')
    with open('bank.txt', 'r', encoding='UTF-8') as bank:
        bank = bank.read()
    print('Текущий баланс:', bank, 'рублей.')
    print()
    main()
```

Рисунок 8 – Функция просмотра баланса


```
def delete_balance():
    print('-----\n')
    with open('bank.txt', 'r', encoding='UTF-8') as bank:
        bank = bank.read()
    print('Текущий баланс:', bank, 'рублей.')

    otvet = input('Введите сумму в рублях, которая будет записана как новая: ')
    if otvet != '':
        for i in otvet:
            if i in alf:
                print('Ошибка при вводе.')
                break
        else:
            sm = int(otvet)
            if sm != int(bank):
                with open('bank.txt', 'w', encoding='UTF-8') as bank:
                    bank.write(str(sm))
                print('\nБаланс успешно изменен!')
                print('Текущий баланс:', sm, 'рублей.')
            else:
                print('Ошибка! Сумма должна отличаться от предыдущей.')
    else:
        print('Ошибка при вводе.')
    print()
    main()
```

Рисунок 9 – Функция изменения баланса

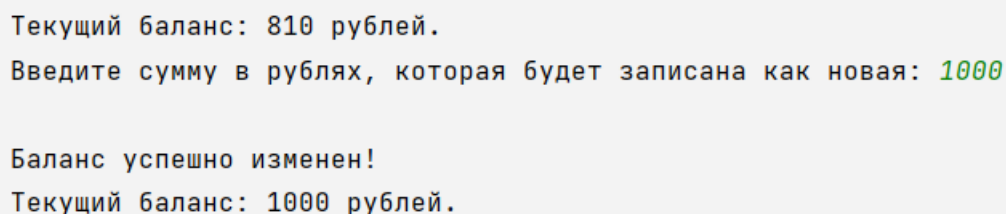
При просмотре баланса производится считывание из файла в переменную и печать этой фразы на экран с этой переменной (рисунок 10). При изменении баланса так же происходит считывание текущей суммы из файла. Далее пользователь вводит новую сумму, которая проверяется на правильность ввода (например сумма не может быть отрицательной или пустой) при

помощи поэлементного сравнения с недопустимыми символами, а также проверяется на повторение предыдущей суммы с помощью условного оператора, после чего сумма записывается в файл, используется как новая, а также выводится на экран (рисунок 11). В противном случае выводится сообщение об ошибке.



```
Текущий баланс: 810 рублей.
```

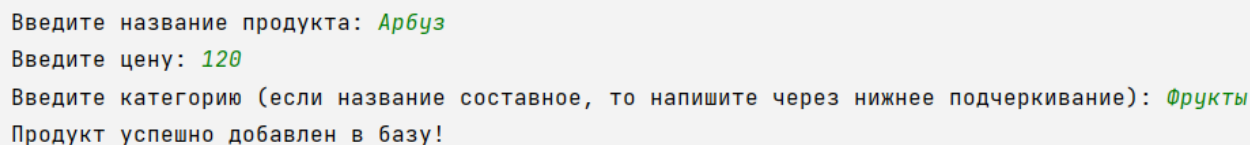
Рисунок 10 – Вывод баланса



```
Текущий баланс: 810 рублей.  
Введите сумму в рублях, которая будет записана как новая: 1000  
  
Баланс успешно изменен!  
Текущий баланс: 1000 рублей.
```

Рисунок 11 – Изменение баланса

3) Функция просмотра и изменения базы продуктов. Для последующей работы пользователь добавляет продукты в базу, указывая их стоимость и категорию (рисунок 12). При вводе данных происходит проверка на присутствие данного продукта в базе (рисунок 13). Во избежание ошибок с регистром, введенные данные обрабатываются методом `capitalize()`.



```
Введите название продукта: Арбуз  
Введите цену: 120  
Введите категорию (если название составное, то напишите через нижнее подчеркивание): Фрукты  
Продукт успешно добавлен в базу!
```

Рисунок 12 – Функция добавления продукта в базу

```

baza = []
with open('products.txt', 'r', encoding='UTF-8') as prod_baza:
    for line in prod_baza:
        baza.append(line.split('\n'))
new_product = input('Введите название продукта: ')
new_product = new_product.capitalize()
if new_product != '':
    for j in range(len(baza)):
        baza_new = baza[j][0].split(' ')
        if new_product.casefold() == baza_new[0].casefold():
            print('Продукт продукт уже находится в базе.')
            break

```

Рисунок 13 – Проверка введенных данных

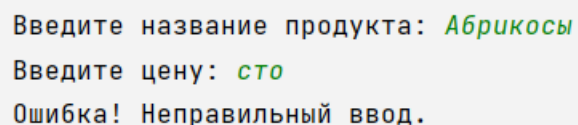
Если проверка прошла успешно, то пользователь вводит цену и категорию продукта, которые проверяются на корректность данных при помощи цикла for и поэлементного сравнения символов (рисунок 14), например в стоимости не может быть букв (рисунок 15). Если же все проверки пройдены, то продукт добавляется в базу путем записи его в файл, иначе выдается сообщение об ошибке.

```

otvet = input('Введите цену: ')
if otvet != '':
    for k in otvet:
        if k in alf:
            print('Ошибка! Неправильный ввод.')
            break
    else:
        price = int(otvet)
        category = input(
            'Введите категорию (если название составное, то напишите через нижнее подчеркивание): ')
        category = category.capitalize()
        if category != '':
            prod_for_write = '\n' + new_product + ' ' + str(price) + ' ' + category
            with open('products.txt', 'a', encoding='UTF-8') as new_prod_baza:
                new_prod_baza.write(prod_for_write)
            print('Продукт успешно добавлен в базу!')
        else:
            print('Ошибка! Нет данных.')
    else:
        print('Ошибка! Нет данных.')

```

Рисунок 14 – Проверка корректности цены и категории, а также вывод



Введите название продукта:

Введите цену:

Ошибка! Неправильный ввод.

Рисунок 15 – Неправильный ввод при добавлении

Пользователь также может удалить продукты из базы (рисунок 16). При удалении продукта из базы происходит предварительное считывание данных в список и вывод его элементов для более удобного использования. После чего пользователь вводит имя продукта, который он хочет удалить. Имя обрабатывается методом `capitalize()`, после чего проверяется на правильность ввода и наличие в базе. Если проверки пройдены, то совпадающее значение удаляется из списка при помощи метода `pop()` и остальной список записывается в файл, после чего выводится сообщение об успехе (рисунок 17). В противном случае пользователю выводится ошибка.

```

with open('products.txt', 'r', encoding='UTF-8') as prod_baza:
    for line in prod_baza:
        baza.append(line.split('\n'))

for j in range(len(baza)):
    baza_prod = baza[j][0].split(' ')
    cur_list += ', ' + baza_prod[0]
cur_list = cur_list[2:]
print('Текущая база: ' + cur_list)

otvet = input('Какой продукт желаете удалить? ')
otvet = otvet.capitalize()
if otvet != '':
    for i in range(len(baza)):
        baza_prod = baza[i][0].split(' ')
        if otvet == baza_prod[0]:
            for k in range(len(baza)):
                if otvet in (baza[k-1][0]):
                    baza.pop(k-1)
            with open('products.txt', 'w', encoding='UTF-8') as baz:
                for i in range(len(baza)):
                    baz.write(baza[i][0])
                    if i != len(baza) - 1:
                        baz.write('\n')
            print('Продукт успешно удален из базы!')
            break
        else:
            print('Ошибка. Такой записи нет.')
    else:
        print('Ошибка. Неправильный ввод.')

```

Рисунок 16 – Функция удаления продукта из базы

```

Текущая база: Яблоки, Апельсины, Молоко, Сок, Клубника, Капуста, Вода, Курица, Свинина, Говядина, Лимонад, Пирожные, Ананас, Мороженое, Кабачки
Какой продукт желаете удалить? Кабачки
Продукт успешно удален из базы!

```

Рисунок 17 – Демонстрация удаления продукта

Функция просмотра выдает пользователю все продукты, записанные в базу (рисунок 18). Функция считывает данные из файла, после чего используется цикл for для построчного вывода продуктов в удобном для пользователя виде (рисунок 19).

Продукты, которые уже есть в базе:

Яблоки, 50 рублей, категория - Фрукты
Апельсины, 120 рублей, категория - Фрукты
Молоко, 80 рублей, категория - Напитки
Сок, 60 рублей, категория - Напитки
Клубника, 150 рублей, категория - Фрукты
Капуста, 50 рублей, категория - Овощи
Вода, 30 рублей, категория - Напитки
Курица, 120 рублей, категория - Мясо
Свинина, 140 рублей, категория - Мясо
Говядина, 150 рублей, категория - Мясо
Лимонад, 50 рублей, категория - Напитки
Пирожные, 200 рублей, категория - Сладости
Ананас, 200 рублей, категория - Фрукты
Мороженое, 45 рублей, категория - Сладости
Кабачки, 50 рублей, категория - Овощи
Арбуз, 120 рублей, категория - Фрукты

Рисунок 18 – Просмотр базы продуктов

```
with open('products.txt', 'r', encoding='UTF-8') as prod_baza:
    for line in prod_baza:
        baza.append(line.split('\n'))
for k in range(len(baza)):
    prod_baza = baza[k][0].split(' ')
    prod_baza2.append(prod_baza)
print('Продукты, которые уже есть в базе:\n')
for i in range(len(baza)):
    line = ''
    for j in range(len(prod_baza2[i]) - 2):
        line += prod_baza2[i][j] + ', '
    line += prod_baza2[i][-2] + ' рублей, категория - ' + prod_baza2[i][-1]
    print(line)
```

Рисунок 19 – Функция просмотра базы продуктов

- 4) Функция добавления продуктов в список, а также их удаления и просмотра. При вызове функции добавления в начале программа считывает данные из файла об уже записанных продуктах, далее пользователь вводит продукты, которые он хочет добавить в список, а

также их количество. Если продукта нет в базе, то пользователю будет предложено его добавить туда, если пользователь согласится и все проверки будут пройдены, то продукт будет записан в базу и список (рисунок 20). Алгоритм добавления продуктов в список представлен на рисунке 21.

Введите название продукта(ов) через запятую, который(ые) Вы хотели бы добавить в список (если название составное, то напишите через нижнее подчеркивание): *Молоко, Индейка*

Введите количество продукта "Молоко": *2*

Продукт "Молоко" успешно добавлен в список!
Текущий список: Молоко, Молоко

Введите количество продукта "Индейка": *1*

Продукта нет в базе.
Желаете добавить продукт в базу (введите Да или Нет)? *да*

Введите цену: *200*

Введите категорию (если название составное, то напишите через нижнее подчеркивание): *мясо*

Продукт успешно добавлен в базу и список!
Текущий список: Молоко, Молоко, Индейка

Рисунок 20 – Демонстрация добавления продуктов в список

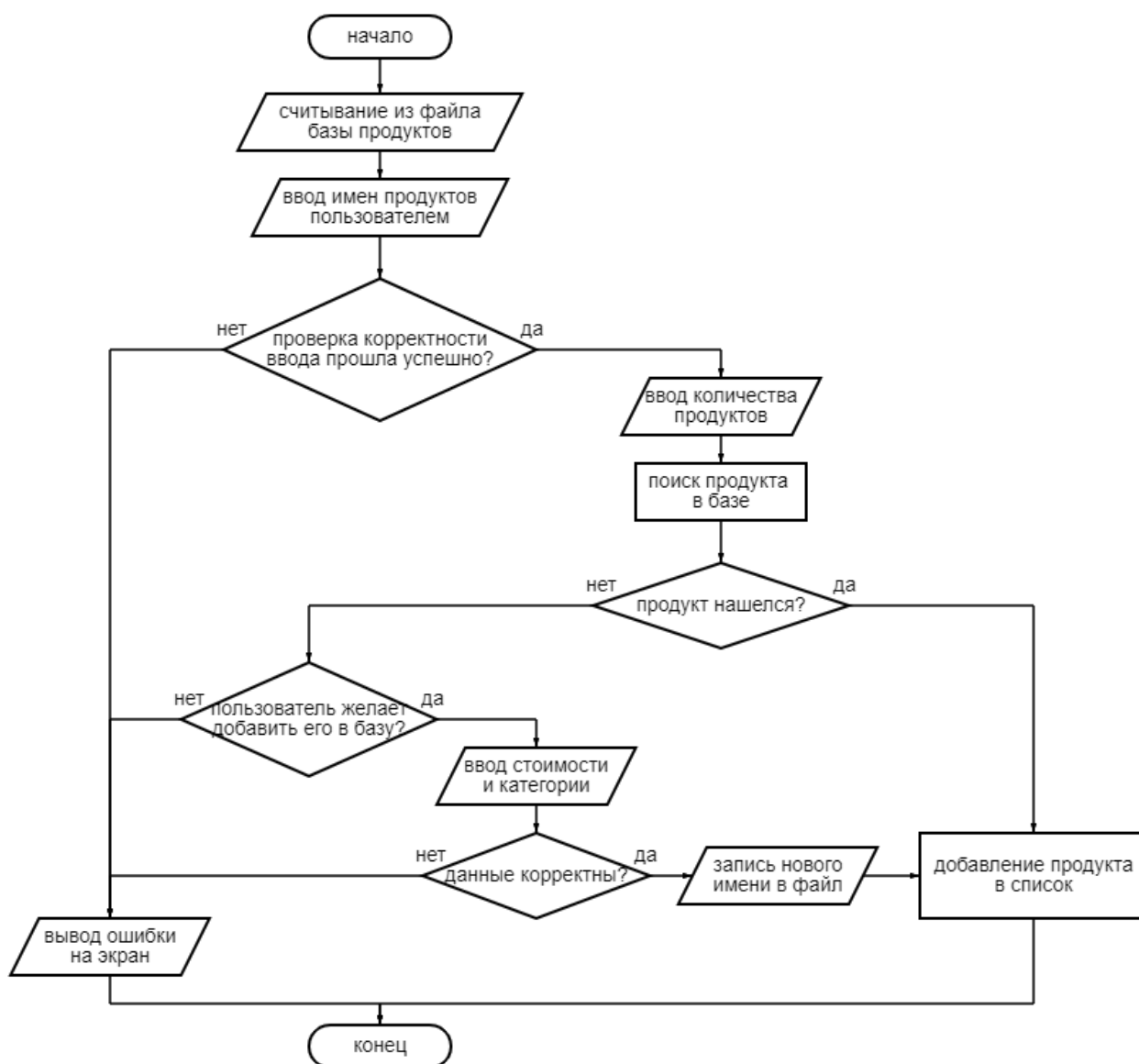


Рисунок 21 – Алгоритм добавления продуктов в список

Если пользователь хочет удалить что-то из списка, он может воспользоваться функцией удаления. Программа выведет все продукты, которые находятся в списке. Далее пользователь вводит продукт, который он хочет убрать, а также его количество, если он найден, иначе пользователю вернет ошибку. Если же продукт этого типа был один в список, то программа сразу удалить его из списка, не запрашивая количество у пользователя (рисунок 22).

```

if cart != []:
    print('Текущий список: ' + ', '.join(cart))

    otvet = input('Какой продукт желаете удалить? ')
    otvet = otvet.capitalize()
    if otvet not in cart:
        print('Ошибка. Такой записи нет.')
    else:
        cnt = 0
        for j in range(len(cart)):
            if otvet == cart[j - 1]:
                cnt += 1
        if cnt == 1:
            for i in range(len(cart)):
                if otvet == cart[i-1]:
                    cart.pop(i-1)
                    print('Продукт успешно удален из списка!')
                    print('Текущий список: ' + ', '.join(cart))

```

Рисунок 22 – Функция удаления одного продукта

Если же продуктов этого типа было несколько, то введенное пользователем число пройдет проверку на недопустимые символы при помощи посимвольного сравнения, а также проверку на нахождение в диапазоне (введенное число не может быть больше, чем количество продуктов в списке, а также меньше либо равно нулю). Далее программа записывает в отдельный список все продукты, которые надо удалить и после методом `remove()` убирает их из списка продуктов (рисунок 23), далее пользователю показывается конечный список (рисунок 24).

```

otvet2 = input(f'Количество продуктов у Вас в списке: {cnt}. Сколько желаете удалить? ')
if otvet2 != '':
    for k in otvet2:
        if k in alf:
            break
    else:
        ans = int(otvet2)
    if cnt < ans or ans < 1:
        print('Ошибка! Неправильный ввод.')
    else:
        for n in range(len(cart)):
            if otvet == cart[n]:
                un_cart.append(cart[n])
                ans -= 1
            if ans == 0:
                break
        for h in un_cart:
            cart.remove(h)
        print('Продукт успешно удален из списка!')
        print('Текущий список: ' + ', '.join(cart))
else:
    print('Ошибка! Неправильный ввод.')

```

Рисунок 23 – Функция удаления из списка нескольких продуктов

```

Текущий список: Молоко, Молоко, Индейка
Какой продукт желаете удалить? молоко
Количество продуктов у Вас в списке: 2. Сколько желаете удалить? 2
Продукт успешно удален из списка!
Текущий список: Индейка

```

Рисунок 24 – Демонстрация удаления продуктов из списка

Пользователь также может просмотреть текущий список продуктов. Важно отметить, что функция будет выдавать ошибку, если список пуст. При выводе используется метод `join()`, который объединяет элементы в строку (рисунок 25), для более привычного восприятия (рисунок 26).

```

def view_cart():
    print('-----\n')
    if cart != []:
        print('Текущий список: ' + ', '.join(cart))
    else:
        print('Текущий список пуст.')
    print()
main()

```

Рисунок 25 – Функция вывода списка

Текущий список: Индейка, Молоко, Молоко

Рисунок 26 – Демонстрация вывода списка продуктов

- 5) Функция создания, просмотра и удаления записей о покупке. После того, как пользователь добавил продукты в список он может создать запись о покупке. При вызове функции создания записи программа считывает данные о балансе из файла. Далее функция выведет сумму, которая будет необходима для «оплаты» и проверит достаточно ли у пользователя средств, если нет, то вернет ошибку (рисунок 27).

```
with open('bank.txt', 'r', encoding='UTF-8') as bank:
    bank = bank.read()
baza = []
sm = 0
bought = ''

if cart != []:
    with open('products.txt', 'r', encoding='UTF-8') as prod_baza:
        for line in prod_baza:
            baza.append(line.split('\n'))
    for i in range(len(cart)):
        for j in range(len(baza)):
            baza_prod = baza[j][0].split(' ')
            if cart[i] == baza_prod[0]:
                sm += int(baza_prod[1])
                bought += ', ' + str(baza_prod[0])
    print(f'Общая стоимость покупки в рублях: {sm}')
    if sm > int(bank):
        print(f'Ошибка! Недостаточно средств. Данные о покупке не записаны.')
```

Рисунок 27 – Проверка средств для «оплаты»

Если средств хватает, то пользователю предоставляется выбор: ввести данные о дате вручную или позволить программе автоматически записать сегодняшнюю дату. В случае, если пользователь решил ввести дату вручную, то ему будет необходимо ввести год, месяц и день, которые записываются в отдельные переменные. Если все проверки пройдены, то при помощи библиотеки datetime будет записана эта дата. Важно сказать,

что дата не будет записана, если она еще не наступила. Если же пользователь отказался вводить дату вручную, то с помощью `datetime` будет записана сегодняшняя дата. Если дата записана, то произойдет запись о покупке в файл, вычитание суммы из текущего баланса, а также очищение списка продуктов с помощью `del` (рисунок 28). После чего пользователю сообщится об успехе (рисунок 29).

```
today = datetime.date.today()
if today != '':
    bought = '\n' + bought[2:] + ', ' + str(sm) + ', ' + str(today)
    del cart[:]
    new_bank = int(bank) - sm
    with open('bought.txt', 'a', encoding='UTF-8') as new_bought:
        new_bought.write(bought)
    with open('bank.txt', 'w', encoding='UTF-8') as bank:
        bank.write(str(new_bank))
    print('Данные о покупке успешно записаны!')
```

Рисунок 28 – Создание записи о покупке

```
Общая стоимость покупки в рублях: 360
Желаете ввести дату вручную (введите Да или Нет (запишет сегодняшнее число))? нет
Данные о покупке успешно записаны!
```

Рисунок 29 – Демонстрация создания записи о покупке

Пользователю также доступно удаление записей. При вызове этой функции программа выведет на экран считанные из файла пронумерованные записи, далее пользователю будет предложено ввести номер одной из записей (рисунок 30). Если введенный номер находится в диапазоне, то программа удалит эту запись из списка, а затем перезапишет файл. Более того, сумма, которая была потрачена на покупку, вернется и так же запишется в файл (рисунок 31).

1) Молоко, Яблоки, 130, 2022-11-11
2) Молоко, Яблоки, 130, 2022-11-12
3) Яблоки, Яблоки, 100, 2022-11-12
4) Молоко, Сок, Сок, Ананас, 400, 2022-11-13
5) Молоко, 80, 2022-11-13
6) Ананас, Ананас, 400, 2022-11-13
7) Ананас, Ананас, 400, 2022-11-13
8) Молоко, Молоко, Клубника, 310, 2022-11-12
9) Груша, Груша, 400, 2022-11-13
10) Молоко, 80, 2022-11-14
11) Мороженое, Мороженое, Мороженое, Мороженое, Мороженое, 225, 2022-11-15
12) Мороженое, Мороженое, Мороженое, Молоко, Молоко, Яблоки, Яблоки, Яблоки, Яблоки, Яблоки, 545, 2022-11-15
13) Молоко, Яблоки, Яблоки, Яблоки, Яблоки, 280, 2022-11-15
14) Индейка, Молоко, Молоко, 360, 2022-11-16

Какую запись желаете удалить (введите число)? 14

Баланс пополнен!

Запись успешно удалена.

Рисунок 30 – Демонстрация удаления записи

```
if otvet > len(baza) or otvet < 1:
    print('Ошибка. Такой записи нет.')
else:
    baza_clone.append(baza[otvet-1][0].split(', '))
    sm = int(baza_clone[0][-2])
    baza.pop(otvet-1)

    with open('bought.txt', 'w', encoding='UTF-8') as baz:
        for i in range(len(baza)):
            baz.write(baza[i][0])
            if i != len(baza)-1:
                baz.write('\n')

    with open('bank.txt', 'r', encoding='UTF-8') as bank:
        bank = bank.read()
    sm += int(bank)
    with open('bank.txt', 'w', encoding='UTF-8') as bank:
        bank.write(str(sm))
    print('Баланс пополнен!')

    print('Запись успешно удалена.')
```

Рисунок 31 – Функция удаления записи

Также пользователю доступен просмотр записей. Программа считывает записи из файла в список (рисунок 31), после чего выводит их в привычном пользователю виде (рисунок 32).

```
def view_bought():
    print('-----\n')
    bought = []
    baza_bought2 = []
    with open('bought.txt', 'r', encoding='UTF-8') as bought_baza:
        for line in bought_baza:
            bought.append(line.split('\n'))
    for k in range(len(bought)):
        baza_bought = bought[k][0].split(', ')
        baza_bought2.append(baza_bought)
    for i in range(len(bought)):
        line = ''
        for j in range(len(baza_bought2[i]) - 2):
            line += baza_bought2[i][j] + ', '
        line += baza_bought2[i][-2] + ' рублей, ' + baza_bought2[i][-1]
        print(line)
    print()
    main()
```

Рисунок 31 – Функция просмотра записей

```
Молоко, Яблоки, 130 рублей, 2022-11-11
Молоко, Яблоки, 130 рублей, 2022-11-12
Яблоки, Яблоки, 100 рублей, 2022-11-12
Молоко, Сок, Сок, Ананас, 400 рублей, 2022-11-13
Молоко, 80 рублей, 2022-11-13
Ананас, Ананас, 400 рублей, 2022-11-13
Ананас, Ананас, 400 рублей, 2022-11-13
Молоко, Молоко, Клубника, 310 рублей, 2022-11-12
Груша, Груша, 400 рублей, 2022-11-13
Молоко, 80 рублей, 2022-11-14
Мороженое, Мороженое, Мороженое, Мороженое, Мороженое, 225 рублей, 2022-11-15
Мороженое, Мороженое, Мороженое, Молоко, Молоко, Яблоки, Яблоки, Яблоки, Яблоки, Яблоки, 545 рублей, 2022-11-15
Молоко, Яблоки, Яблоки, Яблоки, Яблоки, 280 рублей, 2022-11-15
```

Рисунок 32 – Вывод записей о покупках

Пользователь может посмотреть записи по конкретной дате, категории или стоимости. Программа считывает записи в список, далее предлагает пользователю показать их по какому-то параметру. Если пользователь выбрал показ по дате, то ему необходимо ввести год, месяц и число. Далее если данные прошли проверку, то покажутся все записи за это число, если записей нет, то программа выведет соответствующее сообщение (рисунок 33).

```

print(f'Покупки по дате {today}: ')
cnt = 0
for i in range(len(bought)):
    if str(today) in (bought_baza2[i][-1]):
        line = ''
        for j in range(len(bought_baza2[i]) - 2):
            line += bought_baza2[i][j] + ', '
        line += bought_baza2[i][-2] + ' рублей, ' + bought_baza2[i][-1]
        print(line)
        cnt += 1
        for_sort.append(bought_baza2[i])
if cnt == 0:
    print('Покупок за эту дату нет.')

```

Рисунок 33 – Показ покупок по дате

Если пользователь выбрал показ по категории, то программа выведет все доступные категории при помощи считывания данных из файла в список, а далее применив специальный метод с ключевым словом «set», чтобы избавиться от дубликатов. После чего пользователю будет доступен ввод категории и построчный вывод соответствующих записей, которые содержат эту категорию. В случае если записей нет, то пользователю выведется сообщение (рисунок 34).

```

category = set(category)
print('Доступные категории:', ', '.join(category))
otvet = input('По какой категории Вы хотите просмотреть покупки (введите одну из доступных категорий)? ')
if otvet in category:
    print('\nПокупки, содержащие категорию:')

    for i in range(len(bought)):
        cnt = 0
        for j in range(len(bought_baza2[i]) - 2):
            for k in range(len(all_baza_prod)):

                if (bought_baza2[i][j]) == all_baza_prod[k][0] and cnt == 0:
                    if otvet == all_baza_prod[k][2]:
                        cnt += 1
                        line = ''
                        for n in range(len(bought_baza2[i]) - 2):
                            line += bought_baza2[i][n] + ', '
                        line += bought_baza2[i][-2] + ' рублей, ' + bought_baza2[i][-1]
                        print(line)
                        for_sort.append(bought_baza2[i])
        if line == '':
            print('Покупок в этой категории нет.')

```

Рисунок 33 – Показ записей по категории

При показе по критерию стоимости пользователю предлагается ввести сумму, по которой будет произведен показ. После проверок программа выведет все записи с этой стоимостью или сообщение, что записей не нашлось (рисунок 34).

```
elif otvet == 'Стоимость' or otvet == 'стоимость' or int(otvet) == 3:
    for k in range(len(bought)):
        bought_baza = bought[k][0].split(', ')
        bought_baza2.append(bought_baza)
    otvet = input('Введите стоимость в рублях, по которой Вы хотите просмотреть покупки: ')
    if otvet != '':
        for k in otvet:
            if k in alf:
                print('Ошибка! Неправильный ввод.')
                break
            else:
                cost = int(otvet)
                print(f'\nПокупки со стоимостью {cost} рублей: ')
                cnt = 0
                for i in range(len(bought)):
                    if str(cost) in (bought_baza2[i][-2]):
                        line = ''
                        for j in range(len(bought_baza2[i]) - 2):
                            line += bought_baza2[i][j] + ', '
                        line += bought_baza2[i][-2] + ' рублей, ' + bought_baza2[i][-1]
                        print(line)
                        cnt += 1

                if cnt == 0:
                    print('Покупок с такой стоимостью нет.')
    else:
        print('Ошибка! Неправильный ввод.')
```

Рисунок 34 – Показ записей по стоимости

Для показа записей по дате и категории доступна сортировка по стоимости. Можно отсортировать по возрастанию или убыванию при помощи метода `sort()`, который берет для сравнения стоимость в записях (рисунок 35). Далее производится построчный вывод записей (рисунок 36). Полный алгоритм представлен на рисунке 37.

```

        'Желаете отсортировать по возрастанию или убыванию (введите цифры 1 (по возрастанию) или 2 (по убыванию))?'
    )
if otvet == 'По возрастанию' or otvet == 'по возрастанию' or int(otvet) == 1:
    for_sort.sort(key=lambda i: int(i[-2]))
    print('\nОтсортированный список:')
    for i in range(len(for_sort)):
        line = ''
        for n in range(len(for_sort[i]) - 2):
            line += for_sort[i][n] + ', '
        line += for_sort[i][-2] + ' рублей, ' + for_sort[i][-1]
        print(line)
elif otvet == 'По убыванию' or otvet == 'по убыванию' or int(otvet) == 2:
    for_sort.sort(key=lambda i: int(i[-2]), reverse=True)
    print('\nОтсортированный список:')
    for i in range(len(for_sort)):
        line = ''
        for n in range(len(for_sort[i]) - 2):
            line += for_sort[i][n] + ', '
        line += for_sort[i][-2] + ' рублей, ' + for_sort[i][-1]
        print(line)
else:
    print('Ошибка! Неправильный ввод.')

```

Рисунок 35 – Сортировка по стоимости

```

По какому параметру Вы хотите просмотреть покупки (введите цифру 1 (Дата), 2 (Категория) или 3 (Стоимость))? 1

Какой год? 2022
Какой месяц (введите число)? 11
Какой день? 12

Покупки по дате 2022-11-12:
Молоко, Яблоки, 130 рублей, 2022-11-12
Яблоки, Яблоки, 100 рублей, 2022-11-12
Молоко, Молоко, Клубника, 310 рублей, 2022-11-12

Желаете отсортировать по стоимости (введите Да или Нет)? да
Желаете отсортировать по возрастанию или убыванию (введите цифры 1 (по возрастанию) или 2 (по убыванию))? 2

Отсортированный список:
Молоко, Молоко, Клубника, 310 рублей, 2022-11-12
Молоко, Яблоки, 130 рублей, 2022-11-12
Яблоки, Яблоки, 100 рублей, 2022-11-12

```

Рисунок 36 – Демонстрация показа записей по дате с использованием сортировки

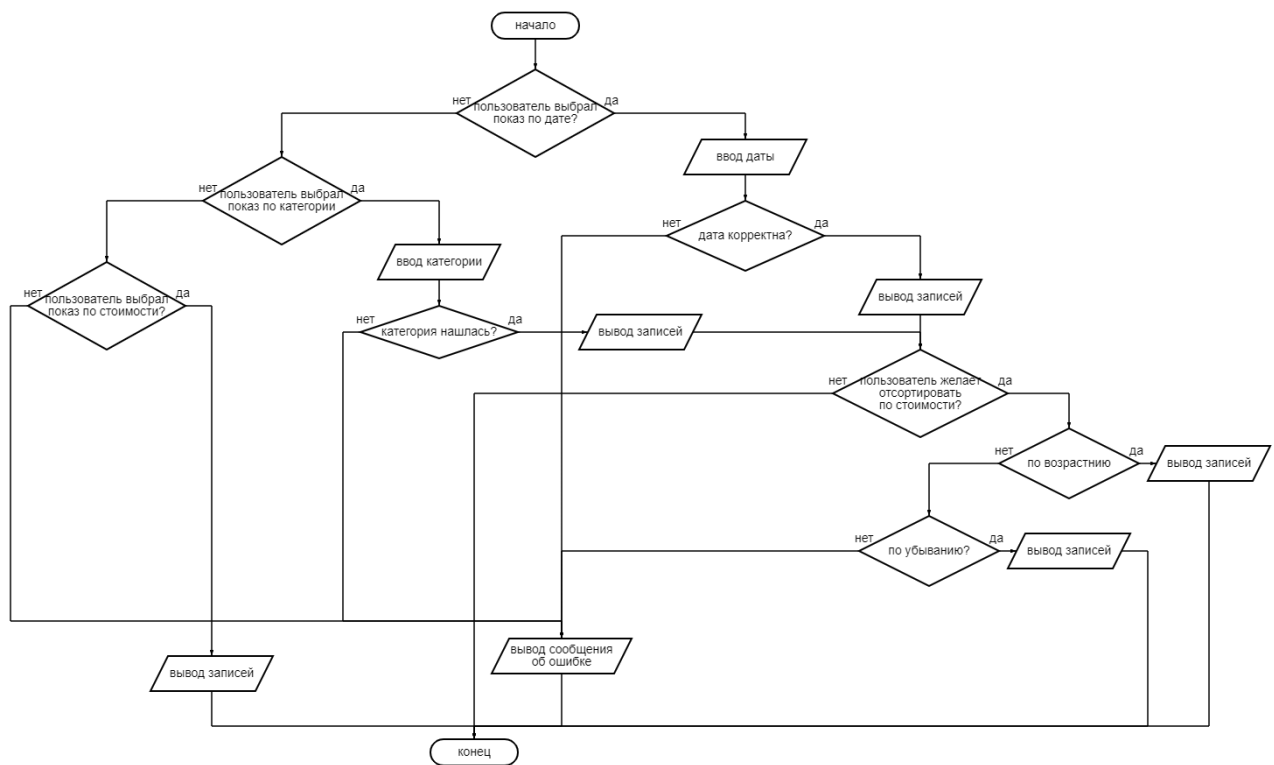


Рисунок 37 – Алгоритм показа записей по категориям

- 6) Функции интерфейса. В качестве интерфейса используется меню, где команды пронумерованы. Пользователь, вводя ту или иную цифру, перемещается по разным меню и выбирает действия (рисунок 38). Здесь так же предусмотрена проверка на корректность ввода, а также используется условный оператор (рисунок 39).

Быстрый доступ:

- 1) Добавить продукт(ы) в список покупок
- 2) Сделать новую запись о покупке
- 3) Просмотреть записи о покупках
- 4) Просмотреть текущий баланс
- 5) Другие функции
- 6) Выход из программы

Введите номер действия: 5

Другие функции:

- 1) Действия с балансом
- 2) Действия со списком продуктов
- 3) Действия с базой продуктов
- 4) Действия с историей покупок
- 5) Изменить имя пользователя в приветствии
- 6) Назад
- 7) Выход из программы

Введите номер действия: 3

Действия с базой продуктов:

- 1) Посмотреть базу продуктов
- 2) Добавить продукт в базу
- 3) Удалить продукт из базы
- 4) Назад
- 5) Выход из программы

Введите номер действия:

Рисунок 38 – Демонстрация перемещения по различным меню


```
def main():
    print('-----\n'
          'Быстрый доступ:\n'
          '1) Добавить продукт(ы) в список покупок\n'
          '2) Сделать новую запись о покупке\n'
          '3) Просмотреть записи о покупках\n'
          '4) Просмотреть текущий баланс\n'
          '5) Другие функции\n'
          '6) Выход из программы\n')
    ans = input('Введите номер действия: ')
    if ans != '':
        for k in ans:
            if k in alf:
                print('Ошибка! Неправильный ввод.')
                main()
                break
        if int(ans) == 1:
            add_product(cart)
        elif int(ans) == 2:
            buy(cart)
        elif int(ans) == 3:
            view_bought()
        elif int(ans) == 4:
            balance()
        elif int(ans) == 5:
            dop_func()
        elif int(ans) == 6:
            out()
        else:
            print('Ошибка! Неправильный ввод.')
```

Рисунок 39 – Функция перемещения по меню

Когда пользователь закончил работу, он выбирает соответствующий пункт, после чего программа запускает соответствующую функцию (рисунок 40), прощается с пользователем и заканчивает свою работу (рисунок 41).

```
def out():
    print('-----\nДо свидания!')
    exit(0)
```

Рисунок 40 – Функция окончания работы программы

Быстрый доступ:

- 1) Добавить продукт(ы) в список покупок
- 2) Сделать новую запись о покупке
- 3) Просмотреть записи о покупках
- 4) Просмотреть текущий баланс
- 5) Другие функции
- 6) Выход из программы

Введите номер действия: 6

До свидания!

Рисунок 41 – Демонстрация окончания работы программы

Вывод:

В процессе выполнения домашнего задания был получен опыт создания программного обеспечения системы обработки данных на языке Python, что в дальнейшем поможет в создании более сложного программного обеспечения.