Міністерство освіти і науки України

Відокремлений структурний підрозділ

«Зборівський фаховий коледж

Тернопільського національного технічного університету імені Івана Пулюя»

Циклова комісія комп’ютерно

інформаційних технологій

Курсова робота

з дисципліни

«Об’єктно-орієнтоване програмування» на тему:

«Дослідити можливості Python

при роботі з текстовими і

табличними даними»

Розробила студентка

групи 31-ІПЗ

Гуртовенко Дарія

Керівник проєкту

Баран М. П.

Зборів, 2025 р.

# ****ЗМІСТ****

**Завдання на курсову роботу** ................................................................................ 2

**Перелік умовних позначень** ............................................................................... 4

**Вступ** ..................................................................................................................... 5

1. **Аналіз предметної області, методів та засобів** ............................................. 6
   1. Поняття та значення списку покупок .................................................. 6
   2. **Методи та засоби вирішення задачі** .................................................. 10

**1.2.1 Алгоритми для обробки списків покупок ................................ 10**

**1.2.2 Використання програмних засобів для створення списків покупок …............................................................................................. 11**

1. **Реалізація програмного забезпечення** ......................................................... 13

2.1. **Постановка задачі** ............................................................................... 13

2.1.1 Завдання до КР .......................................................................... 15

2.1.2 Вхідна інформація ..................................................................... 18

2.1.3 Вихідна інформація ................................................................... 22  
 2.2 **Опис мови програмування** ................................................................. 25  
 2.3 **Опис алгоритму та ідентифікатори** ................................................... 28  
 2.4. **Тестування програми** ......................................................................... 31

**Висновки** .......................................................................................................... 35

**Перелік використаної літератури** ................................................................... 37

**Додатки** ............................................................................................................. 38

**ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ**

CSV (Comma Separated Values) — формат файлу для зберігання табличних даних.

TXT — текстовий формат файлу для зберігання даних.

GUI (Graphical User Interface) — графічний інтерфейс користувача.

CLI (Command Line Interface) — інтерфейс командного рядка.

JSON — формат обміну даними, який може використовуватися в програмах для роботи з даними.

API (Application Programming Interface) — інтерфейс для взаємодії між різними компонентами програмного забезпечення.

ООП (Об'єктно-орієнтоване програмування) — парадигма програмування, яка використовується в роботі.

**ВСТУП**

У сучасному світі, де цифрові технології стали невід'ємною частиною нашого повсякденного життя, необхідність у простих і ефективних рішеннях для управління фінансами та іншими аспектами побуту зростає. Однією з таких необхідностей є планування та ведення списку покупок. Традиційно цей процес здійснюється вручну, але завдяки розвитку інформаційних технологій з'являються нові можливості для автоматизації і спрощення подібних завдань.

Метою цієї курсової роботи є розробка програми для ведення списку покупок, яка дозволяє користувачеві додавати нові покупки, переглядати наявний список і підраховувати загальну суму витрат. Застосування Python для реалізації цього проєкту дозволяє ефективно працювати з текстовими і табличними даними, зберігати інформацію у файлах форматів .txt та .csv, що є зручними для подальшої обробки або аналізу. Програма дає можливість автоматизувати процес введення і зберігання даних, а також забезпечує простий доступ до них.

Завданням цієї роботи є не тільки реалізація описаної програми, але й дослідження можливостей Python при роботі з файлами, текстовими і табличними даними. Використання бібліотек Python для роботи з CSV-файлами дозволяє зберігати покупки у табличному вигляді, що робить дані більш структурованими і доступними для аналізу. Цей проєкт є хорошим прикладом застосування основних концепцій програмування на Python, що дозволяє зрозуміти, як мови програмування можуть спростити повсякденні завдання.

**1** **АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ,**

**МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАВДАННЯ**

#### 1.1 Поняття та значення списку покупок

**Список покупок** — це інструмент, що служить для організації процесу покупок, дозволяючи зберігати необхідну інформацію про товари, які потрібно придбати. З самого початку свого існування список покупок був простим способом уникнути забудькуватості при покупках. Однак з часом цей інструмент набув більше значення, ставши важливим елементом планування витрат і контролю над бюджетом.

##### ****Історія розвитку списків покупок****

Спочатку списки покупок були суто фізичними: люди записували на аркуші паперу або вигравіювали на камені необхідні товари для покупки. Протягом століть цей процес не зазнав значних змін, однак із розвитком людської цивілізації та зростанням кількості товарів на ринку важливість ведення таких списків значно зросла. Із ростом споживчого попиту на продукти і товари, зокрема в індустріальну епоху, люди почали все більше покладатися на списки покупок для ефективного планування витрат.

У наші дні, завдяки технологічним досягненням, списки покупок значно змінилися: від простого запису на папері до складних цифрових інструментів. Цифрові програми для створення списків покупок дозволяють користувачам синхронізувати свої списки між пристроями, робити додаткові нотатки про кожен товар, підраховувати витрати в реальному часі, а також нагадувати про необхідність зробити покупку.

##### **Основні функції та значення списку покупок:**

1. Найбільш очевидною функцією списку покупок є організація процесу покупки товарів. Він дозволяє користувачам скласти чіткий перелік необхідних предметів, що робить похід до магазину менш хаотичним і більш продуктивним. Замість того щоб покладатися на свою пам'ять, людина може просто переглянути список, щоб не забути про важливі товари. Це особливо важливо при великих покупках, наприклад, під час підготовки до свят, коли необхідно придбати безліч різних товарів.
2. Список покупок також виступає інструментом для контролю за витратами. Завдяки тому, що більшість додатків для списків покупок дають змогу додавати ціни на товари, користувач може заздалегідь розрахувати приблизну суму, яку він витратить. Це дозволяє не перевищувати бюджет і навіть порівнювати ціни на однакові продукти в різних магазинах. У свою чергу, це допомагає уникнути імпульсивних покупок та приймати більш обґрунтовані рішення при здійсненні покупок.
3. Списки покупок дозволяють чітко планувати свій бюджет. Наприклад, коли користувач складає список покупок, він може додати до нього орієнтовну вартість товарів. Це дозволяє користувачу відразу побачити, скільки грошей буде витрачено, і коригувати список у разі необхідності. Також деякі додатки для списків покупок дають змогу враховувати знижки або акції, що дозволяє скоротити витрати.
4. Використання списку покупок значно збільшує ефективність процесу. Завдяки чітко структурованому переліку користувач може швидше пройти по магазинах, зекономивши час. Цей аспект особливо важливий для людей, які ведуть активний спосіб життя і мають обмежену кількість часу на покупки. Список допомагає уникнути повернення в магазин для покупок забутих товарів, тим самим економлячи час і ресурси.

##### **Типи списків покупок:**

1. **Індивідуальні списки покупок -** це найбільш поширений тип списків покупок. Користувач складає список лише для своїх особистих потреб. Такий список може включати в себе продукти для дому, медикаменти, побутову техніку, засоби гігієни і так далі. Індивідуальні списки можуть бути досить простими або складними, залежно від потреб користувача. Наприклад, деякі користувачі створюють списки на основі рецептур, що дозволяє їм чітко планувати покупки продуктів для приготування їжі.
2. **Сімейні списки покупок -** у випадку, коли покупками займається не одна людина, а ціла родина, часто створюються спільні списки покупок. У таких списках можуть бути зазначені предмети для всього господарства — від продуктів харчування до побутової хімії. Крім того, у таких списках можна зазначити, хто з членів родини відповідає за придбання конкретних товарів, що дозволяє уникнути непорозумінь і розподілити обов'язки.
3. **Спеціалізовані списки покупок -** окрім стандартних списків покупок для дому, існують списки, що складаються для конкретних подій, таких як свята, дні народження, корпоративи, весілля або інші урочистості. Ці списки включають спеціальні товари, які можуть бути потрібні для організації святкових заходів, таких як подарунки, декорації, їжа для банкетів тощо. Такі списки часто мають детальний розподіл за категоріями, що дозволяє організувати закупівлю заздалегідь.
4. **Списки покупок для бізнесу -** не лише приватні особи користуються списками покупок. У бізнесі також існує потреба в їх використанні. Наприклад, компанії можуть створювати списки для закупівлі офісних товарів, техніки, канцелярії, запасів для виробництва та інше. Такі списки часто мають більшу складність і деталізацію, оскільки включають конкретні бренди товарів, кількість одиниць, постачальників і інші важливі дані.

##### **Психологічний аспект використання списків покупок:**

1. **Зменшення стресу** - використання списку покупок також має психологічний ефект. У ситуаціях, коли покупок багато, або коли людина має обмежений час, список дозволяє зменшити стрес і відчуття перевантаженості. Користувачі, які ведуть списки, відчувають більше контролю над ситуацією і впевненість у своїх діях.
2. **Забезпечення структури -** списки покупок дають структуру і організованість у процесі здійснення покупок. Це дає змогу уникнути хаосу та непотрібних кроків, таких як пошук товарів без чіткої мети. Структурованість дозволяє точно виконати завдання, без відхилень від запланованого маршруту.
3. **Формування корисних звичок -** ведення списку покупок також є частиною формування корисних фінансових звичок. Люди, які використовують списки покупок регулярно, зазвичай вміють краще управляти своїми витратами, оскільки процес планування покупок зазвичай передбачає обмеження бюджету і обрання лише найнеобхідніших товарів.

##### ****Технічні аспекти створення списку покупок****

Враховуючи, що багато людей використовують цифрові платформи для ведення списків покупок, варто звернути увагу на різні технічні аспекти цього процесу:

* **Програмне забезпечення та додатки -** існує безліч програм та мобільних додатків для створення списків покупок. Від простих текстових редакторів, таких як Google Keep, до спеціалізованих програм типу Any.do та Todolist. Кожен додаток має свої особливості: одні пропонують зручний інтерфейс для швидкого введення даних, інші мають додаткові функції, такі як синхронізація на кількох пристроях, можливість голосового введення або навіть сканування штрих-кодів.
* **Інтеграція з магазинами -** деякі додатки для списків покупок дозволяють інтегруватися з онлайн-магазинами та переглядати ціни на товари, що значно спрощує процес вибору та покупки товарів.

#### ****1.2 Методи та засоби вирішення задачі****

У сучасному світі існує безліч методів і засобів для вирішення проблем, пов'язаних із створенням та управлінням списками покупок. Для ефективного розв’язання задачі з розробки програми для ведення списку покупок необхідно застосовувати різноманітні програмні методи, алгоритми та інструменти. Це дозволить забезпечити зручний і швидкий доступ до інформації, а також можливість для користувачів легко організовувати свої покупки, планувати бюджети та здійснювати контроль витрат.

##### ****1.2.1 Алгоритми для обробки списків покупок****

Одним із важливих етапів вирішення задачі є розробка алгоритмів для обробки списків покупок. Оскільки списки покупок можуть бути досить великими та складними, необхідно застосовувати ефективні методи зберігання, сортування та маніпулювання даними[6]. Основні задачі, які необхідно вирішити під час розробки алгоритмів, включають:

1. **Додавання елементів до списку - о**сновна задача полягає в тому, щоб користувач міг швидко додавати нові елементи до списку покупок. Алгоритм додавання має бути простим і швидким. Зазвичай додається новий товар в кінець списку, але в деяких випадках можна реалізувати функціональність для додавання товару в середину списку в залежності від пріоритету покупки (наприклад, найбільш термінові товари).
2. **Видалення елементів із списку** - ще однією поширеною операцією є видалення елементів зі списку. Користувач може захотіти видалити товар, якщо він більше не є необхідним або був придбаний раніше. Алгоритм видалення має забезпечувати зручний доступ до кожного елемента, а також перевірку на існування товару в списку перед його видаленням.
3. **Сортування елементів -** важливим аспектом є сортування товарів за різними критеріями, такими як назва, категорія, ціна або пріоритет покупки. Це дозволяє користувачам краще орієнтуватися в списку і швидше знаходити потрібні товари. Алгоритми сортування, такі як сортування злиттям або швидке сортування, можуть бути застосовані для оптимізації цього процесу.
4. **Фільтрація та пошук -** оскільки список покупок може включати велику кількість елементів, важливо мати можливість швидко знаходити необхідний товар. Алгоритм пошуку має забезпечувати ефективний пошук товарів за певними характеристиками, такими як назва, категорія або ціна. Використання бінарного пошуку або хеш-таблиць дозволяє прискорити пошук в списку.
5. **Підрахунок вартості покупок -** додавання цін до кожного товару дозволяє підрахувати загальну вартість покупок. Це можна зробити шляхом проходження через всі елементи списку і підсумовування їх вартостей. Алгоритм має перевіряти коректність введених даних і забезпечувати правильний облік всіх товарів.

##### ****1.2.2 Використання програмних засобів для створення списків покупок****

Для розробки програм, що працюють з покупками, використовуються різноманітні програмні засоби[1]. Вибір технології безпосередньо впливає на продуктивність, функціональність і зручність користування програмою.

1. **Мови програмування: Python та JavaScript є** одним з найбільш популярних мов програмування для створення простих і потужних програм є **Python**. Його простота синтаксису та багатий набір бібліотек дозволяють швидко створювати ефективні додатки для роботи з даними. У контексті списку покупок Python можна використовувати для обробки текстових файлів (.txt), CSV-файлів та навіть інтеграції з базами даних для збереження інформації. Іншою популярною мовою є **JavaScript**, яка забезпечує можливість створення веб-додатків для ведення списку покупок через браузер.
2. Для роботи з даними та створення програм для списків покупок можна використовувати такі бібліотеки Python, як **Pandas** для обробки табличних даних і **csv** для роботи з CSV-файлами. За допомогою **Tkinter** можна створити простий графічний інтерфейс користувача (GUI) для програми, що дозволить зробити її більш інтерактивною. Для складніших рішень можна використовувати **Flask** або **Django** для розробки веб-додатків.
3. Якщо список покупок має включати велику кількість товарів або потребує зберігання історії покупок, можна використовувати бази даних. Простою та швидкою базою даних для невеликих проектів є **SQLite**, яка дозволяє зберігати дані в локальних файлах без необхідності в сервері бази даних. Для більших систем можна використовувати такі бази, як **MySQL** або **PostgreSQL**, що дозволяють працювати з великими обсягами даних.
4. У наш час багато користувачів хочуть мати доступ до своїх списків покупок на мобільних пристроях. Для створення мобільних додатків використовуються інструменти, такі як **React Native** для створення крос-платформених мобільних додатків, що підтримують Android і iOS. Це дозволяє користувачам зручно керувати своїми списками покупок на ходу.
5. Створення інтерфейсу користувача є важливим етапом розробки програм для списків покупок. Для простих програм часто використовується **консольний інтерфейс** (CLI), де користувач взаємодіє з програмою через командний рядок. Для більш складних програм створюються **графічні інтерфейси користувача** (GUI), які роблять програму більш доступною для людей без технічних навичок. Окрім стандартних кнопок і текстових полів, можна реалізувати такі елементи як таблиці, спливаючі вікна для додавання нових товарів, вибіркові фільтри для сортування та пошуку товарів.

**2 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

**2.1 Постановка задачі**

Сучасний світ з кожним роком стає все більш цифровим, і в результаті кількість даних, яку ми генеруємо та обробляємо, зростає з величезною швидкістю. У цій ситуації важливу роль відіграють інструменти та технології, що дозволяють ефективно працювати з інформацією, забезпечуючи її збереження, обробку та аналіз. Одним із найбільш поширених способів зберігання даних є текстові та табличні формати, зокрема CSV (Comma Separated Values), а також прості текстові файли. Ці формати використовуються в багатьох сферах: від ведення списків покупок до бухгалтерії, фінансів та наукових досліджень. Важливою задачею є розробка програм, що дозволяють автоматизувати обробку таких даних, що забезпечує ефективність, точність і зручність роботи.

У багатьох випадках люди стикаються з необхідністю ведення різноманітних списків або таблиць для персональних або професійних цілей. Наприклад, списки покупок, що містять назву товару та його ціну, чи витрати на певні категорії товарів або послуг. Введення і зберігання таких даних можуть бути трудомісткими без відповідних інструментів. Важливою метою є створення простих і ефективних програм, які дозволяють зберігати ці дані в зручному форматі, надаючи можливість переглядати, аналізувати їх і підраховувати витрати. Без таких інструментів обробка таких даних може бути не лише зайвою, але й схильною до помилок.

Постановка задачі цієї курсової роботи полягає в розробці програми, яка дозволяє автоматизувати процеси роботи з текстовими та табличними даними. Конкретно, задача полягає у створенні програми для ведення списку покупок, де користувач може додавати елементи списку, зберігати їх у текстовому файлі, а також зберігати у табличному форматі CSV для подальшого аналізу. Також програма повинна надавати можливість переглядати список покупок і підраховувати загальну суму витрат, що значно спрощує роботу з даними, економить час користувача і зменшує ймовірність помилок при обчисленнях.

Програма повинна бути інтуїтивно зрозумілою, зручнішою для користувачів, які не мають великого досвіду в програмуванні або роботі з базами даних. Користувачі зможуть вводити назву товару та його ціну, а програма автоматично зберігатиме ці дані в обох форматах — текстовому та CSV. Це дозволить забезпечити зручність обробки даних для подальшого аналізу.

Важливою складовою цієї задачі є використання мови програмування Python. Python обраний через свою простоту у вивченні та багатий набір бібліотек, що дозволяють ефективно працювати з текстовими і табличними даними. Бібліотеки, такі як csv для роботи з табличними даними та стандартні методи для роботи з текстовими файлами, дозволяють швидко і ефективно реалізувати основні функції програми, не вдаючись до складних структур або великих обсягів коду.

Крім того, важливим аспектом є обробка помилок та забезпечення коректності даних. Програма повинна мати функціональність для перевірки введених даних (наприклад, перевірка правильності формату введення ціни) та повідомлення користувача про помилки. Це забезпечить більшу надійність і точність результатів.

Завданням цієї курсової роботи є створення такого інструменту, який дозволяє без труднощів автоматизувати обробку даних, зробити її більш ефективною, швидкою та зручною для користувача. Програма повинна стати корисним інструментом для тих, хто хоче оптимізувати процеси обробки інформації у повсякденному житті або в роботі, що є важливим кроком у напрямку розвитку автоматизації та зручності використання інформаційних технологій.

Таким чином, результатом цієї роботи стане програма, яка дозволяє користувачам швидко і зручно вести облік покупок, контролювати витрати та зберігати дані в зручних форматах для подальшого аналізу, що робить її корисною в багатьох повсякденних і професійних ситуаціях.

**2.1.1 Завдання до КР**

Основним завданням цієї курсової роботи є розробка програмного продукту, що дозволяє ефективно працювати з текстовими та табличними даними, зокрема для ведення списку покупок. Програма повинна забезпечити можливість введення даних, їх збереження у різних форматах (текстовий файл і CSV), перегляд списку покупок, а також підрахунок загальної суми витрат. Користувачі повинні мати можливість здійснювати всі ці операції через інтуїтивно зрозумілий інтерфейс, що не потребує спеціальних знань у програмуванні[6].

#### Основні функції програми

Програма повинна реалізувати наступні функції:

1. **Додавання товару до списку покупок**:
   * Користувач повинен мати можливість вводити назву товару та його ціну.
   * Дані мають бути збережені у двох форматах: текстовому файлі та CSV-файлі.
   * Програма повинна перевіряти коректність введених даних (наприклад, ціна повинна бути числовим значенням).
2. **Перегляд списку покупок**:
   * Користувач повинен мати можливість переглядати всі додані покупки. Це можна зробити через текстовий файл, що містить дані у форматі "Назва товару: ціна".
   * Список має бути відображений у зручному для користувача вигляді.
3. **Підрахунок загальної суми витрат**:
   * Програма повинна мати функцію для підрахунку загальної суми витрат за всіма товарами, що були додані до списку.
   * Підрахунок має бути здійснений на основі даних, що зберігаються у CSV-файлі, де кожен товар має свою ціну.
   * В результаті підрахунку користувач повинен отримати точну суму витрат.
4. **Обробка помилок**:
   * Програма повинна коректно обробляти помилки вводу (наприклад, якщо ціна введена не як число).
   * Повідомлення про помилки повинні бути чіткими та зрозумілими для користувача.
5. **Виведення повідомлень для користувача**:
   * Програма повинна надавати користувачеві зрозумілі повідомлення про виконання операцій (наприклад, "Покупку додано", "Невірний формат ціни", "Загальна сума витрат: X грн").

#### Вимоги до інтерфейсу

Програма повинна мати простий текстовий інтерфейс. Користувач буде взаємодіяти з програмою через командний рядок[8]. Потрібно забезпечити можливість вибору операцій через меню. Важливими вимогами є:

1. **Простота використання**:
   * Користувач повинен легко зрозуміти, як додавати покупку, переглядати список покупок та отримувати загальну суму витрат.
   * Усі необхідні інструкції щодо введення даних повинні бути надані у вигляді підказок на екрані.
2. **Зрозумілі повідомлення про помилки**:
   * Програма повинна інформувати користувача про некоректні дії (наприклад, при введенні тексту замість числа для ціни), пропонуючи виправити помилку.
   * Повідомлення повинні бути простими і зрозумілими.
3. **Гнучкість і масштабованість** - програма повинна бути легко модифікована для розширення її функціональності в майбутньому (наприклад, додавання можливості зберігати дані в інших форматах або додавання нових операцій).

#### Технічні вимоги:

1. **Мова програмування** - програма буде написана мовою Python, оскільки Python є однією з найбільш популярних мов для обробки текстових і табличних даних завдяки простоті використання та великій кількості бібліотек для роботи з файлами.
2. **Бібліотеки**:
   * Для роботи з текстовими файлами використовуються стандартні методи Python.
   * Для роботи з CSV-файлами буде використана бібліотека csv, яка є частиною стандартної бібліотеки Python.
3. **Формат даних** - для збереження покупок використовуються два формати: текстовий файл для простоти перегляду та CSV-файл для можливості подальшої обробки даних.
4. **Перевірка наявності файлів** - програма повинна перевіряти, чи існують файли для збереження даних перед виконанням операцій. Якщо файли не існують, програма повинна їх створювати.
5. **Операційна система** - програма повинна бути сумісною з основними операційними системами, такими як Windows, Linux та macOS.

#### Оцінка ефективності програми

Оцінка ефективності програми буде проводитися на основі таких критеріїв:

1. **Швидкість виконання операцій -** час, що витрачається на додавання товару в список, збереження даних у файлі, перегляд списку та підрахунок витрат, не повинен бути надмірним навіть при великій кількості покупок.
2. **Зручність користування -** легкість у використанні програми, зрозумілість інтерфейсу та простота виконання основних операцій.
3. **Точність підрахунку витрат -** програма повинна правильно підраховувати загальну суму витрат без помилок.

#### 2.1.2 Вхідна інформація

У даному підрозділі буде розглянуто типи вхідної інформації, яку програма повинна обробляти, а також специфікації щодо формату введення даних. Для правильної роботи програми необхідно чітко визначити типи даних, які користувач вводить у систему, а також формат збереження цих даних у файлах.

Вхідна інформація складається з наступних основних елементів:

1. **Назва товару**:
   * Користувач вводить текстову інформацію, що є назвою товару, яку необхідно зберігати в списку покупок.
   * Назва товару може містити літери, пробіли, цифри та деякі спеціальні символи (наприклад, дефіси, апострофи). Важливо, щоб програма правильно обробляла введення без обмежень щодо використання символів, окрім некоректних символів, таких як неприпустимі знаки для файлів.
2. **Ціна товару**:
   * Користувач вводить числове значення, що відображає ціну товару. Це значення має бути числовим типом, причому підтримуються як ціле число, так і дробові значення.
   * Програма повинна перевіряти коректність введеного значення, перевіряючи на наявність символів, які не є частиною числа, таких як літери чи спеціальні знаки.
   * Ціна товару повинна бути записана у гривнях (грн). При введенні може бути використано десятковий роздільник у вигляді крапки або коми.
3. **Формат збереження даних**:

Дані про покупку будуть зберігатися в двох форматах:

* + - **Текстовий файл** - кожен запис містить назву товару і його ціну, записані у форматі: "Назва товару: ціна грн".
    - **CSV-файл** - дані про покупку будуть збережені в табличному вигляді, де кожен рядок містить два стовпці: назву товару і ціну товару. Формат CSV є більш структурованим, що дозволяє згодом здійснювати аналіз витрат, використовуючи табличні програми (наприклад, Microsoft Excel або Google Sheets).

### Лістинги вхідних даних:

#### Текстовий формат

У цьому форматі дані зберігаються у вигляді простого тексту. Кожен товар містить назву та ціну, записану через двокрапку і пробіл, за яким слідує ціна в гривнях[2].

**Приклад вхідних даних у текстовому файлі**:

Молоко: 20 грн

Хліб: 12.50 грн

Яблука: 30 грн

Картопля: 15 грн

Програма повинна додавати нові покупки в кінець цього списку, враховуючи введену користувачем назву та ціну.

#### CSV формат

У CSV-файлі дані зберігаються у вигляді таблиці, де кожен рядок містить два елементи: назву товару та ціну товару. Кожен елемент у файлі розділений комою, що дозволяє легко експортувати і імпортувати дані[2].

**Приклад вхідних даних у CSV-файлі:**

Молоко, 20

Хліб, 12.50

Яблука, 30

Картопля, 15

Тут кожен рядок — це окремий товар із його ціною, яка розділена комою.

#### Введення користувача

Для того щоб програма могла коректно обробляти дані, користувач має вводити їх через інтерфейс командного рядка. Користувач вводить два значення: назву товару та ціну[3].

**Приклад введення даних через командний рядок**:

Введіть назву товару: Молоко

Введіть ціну: 20

Програма повинна перевірити правильність введеного числа для ціни та дозволяти введення лише числових значень.

#### Помилки вводу

У разі некоректного введення даних програма повинна повідомляти користувача про помилку та просити повторити введення[4].

**Приклад помилки**:

Введіть назву товару: Молоко

Введіть ціну: двадцять

Помилка! Ціна повинна бути числом.

Або інший варіант:

Введіть назву товару: Хліб

Введіть ціну: -12

Помилка! Ціна не може бути від'ємною.

Програма має забезпечити перевірку вводу та зворотний зв'язок для коректного виконання операції.

#### Структура файлів

Для збереження даних у текстовому файлі або CSV-файлі необхідно правильно форматувати ці файли. При цьому програма повинна автоматично створювати файли при першому запуску, якщо вони ще не існують, та додавати нові записи до існуючих файлів[5].

**Приклад коду для збереження даних у текстовому файлі**  (Лістинг 2.1):

**Лістинг 2.1 Додавання нової покупки**

def add\_item():

item = input("Введіть назву товару: ")

while True:

price = input("Введіть ціну: ")

try:

price = float(price)

if price < 0:

print("Ціна не може бути від'ємною! Спробуйте ще раз.")

continue

break

except ValueError:

print("Помилка! Ціна повинна бути числом. Спробуйте ще раз.")

with open(txt\_file, "a", encoding="utf-8") as txt:

txt.write(f"{item}: {price} грн\n")

with open(csv\_file, "a", newline="", encoding="utf-8") as csvfile:

writer = csv.writer(csvfile)

writer.writerow([item, price])

print("✅ Покупку додано!")

У цьому коді перевіряється, чи введено коректне значення для ціни. Якщо ціна введена як текст або значення, що не може бути перетворене в число, програма виводить повідомлення про помилку і не додає запис у файл.

#### Перегляд списку покупок

При перегляді списку покупок програма повинна вивести всі товари з файлу. У разі, якщо файл порожній, програма повинна повідомити про це користувача.

**Приклад перегляду даних з текстового файлу** (Лістинг 2.2):

Лістинг 2.2 Перегляд списку покупок

def view\_list():

if not os.path.exists(txt\_file):

print("Список покупок порожній.")

return

with open(txt\_file, "r", encoding="utf-8") as txt:

print("\nСписок покупок:")

print(txt.read())

Цей код дозволяє вивести список покупок, збережений у текстовому файлі. Якщо файл порожній, програма повідомить користувача про відсутність даних.

#### 2.1.3 Вихідна інформація

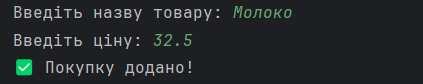
Вихідна інформація є кінцевим результатом виконання операцій у програмі, і вона відображає всі зміни, що були зроблені з даними користувача, а також включає підсумки виконаних обчислень. Для кожної операції програма надає візуальний відгук через текстове виведення на екран, що надає користувачеві необхідну інформацію.

У цій частині курсової роботи ми розглянемо різні варіанти вихідної інформації, що генерується під час виконання програми для списку покупок. Зокрема, ми зупинимося на таких операціях, як перегляд списку покупок, підрахунок витрат, видалення та редагування товарів, а також на тому, як ця інформація відображається у програмі та на екрані.

Основні операції:

#### Додавання товару

Першою операцією в програмі є додавання товару до списку покупок. Після того, як користувач вводить назву товару та його ціну, програма додає цей товар до текстового та CSV файлів. Після успішного додавання програма виводить повідомлення, яке підтверджує, що товар був доданий до списку (Рисунок 2.1):



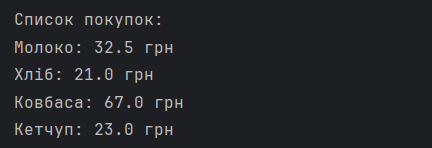
**Рисунок 2.1** — Додавання товару до списку покупок

Цей процес здійснюється як в текстовому форматі (txt\_file), так і в форматі CSV для зручності обчислень. Після додавання, програма не тільки виводить повідомлення на екран, але й оновлює відповідні файли з новими даними.

#### Перегляд списку покупок

Однією з основних функцій програми є перегляд списку покупок. Користувач може побачити всі товари, які були додані до списку разом з їх ціною. Коли програма виконує операцію перегляду, вона зчитує інформацію з текстового файлу (txt\_file), де кожен товар записаний у вигляді рядка з його назвою та ціною.

Вихідна інформація, яка виводиться на екран, містить усі товари, що були записані в текстовому файлі. Це може виглядати так (Рисунок 2.2):



**Рисунок 2.2** — Виведення списку покупок із текстового файлу

Цей список також зберігається у CSV-файлі для подальших обчислень та зручності. Однак для цього етапу програма лише виводить на екран дані з текстового файлу, що є більш доступним для перегляду.

#### 3. Підрахунок витрат

Іншою важливою операцією є підрахунок загальної суми витрат. Ця операція здійснюється на основі даних, що зберігаються в CSV-файлі. Кожен товар записується як рядок, що складається з двох елементів: назви товару та його ціни. Програма зчитує ці дані, додає ціни всіх товарів та виводить загальну суму витрат.

При виконанні цієї операції на екрані відображається повідомлення про загальну суму витрат (Рисунок 2.3):



**Рисунок 2.3** — Підрахунок витрат та виведення результату

Цей результат дозволяє користувачу швидко оцінити, скільки грошей він витратив на покупки, і базується на даних, що зберігаються в програмі. Зверніть увагу, що якщо дані в CSV-файлі змінюються, то загальна сума витрат також буде оновлена відповідно.

#### 4. Видалення товару

Користувач може також видаляти товари зі списку покупок. Для цього програма надає відповідну опцію, і після введення назви товару програма видаляє його з обох файлів: текстового та CSV. Коли товар успішно видалено, програма виводить повідомлення про це, яке підтверджує виконану операцію (Рисунок 2.4):



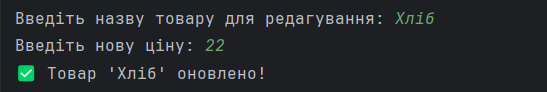
**Рисунок 2.4** — Видалення товару зі списку покупок

Видалення товару з двох файлів дозволяє уникнути дублювання даних і гарантує, що усі дані актуальні. Після видалення товару сума витрат також оновлюється, і це підтверджується виведенням нового підсумку.

#### 5. Редагування товару

Ще однією важливою функцією є можливість редагування товарів. Користувач може змінити ціну певного товару. Для цього програма виводить запит на введення нового значення ціни, після чого вона оновлює записи в обох файлах.

Після успішного редагування програма виводить на екран підтвердження про оновлення ціни (Рисунок 2.5):

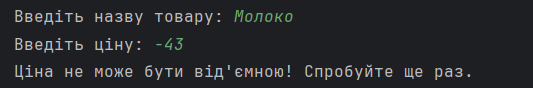


**Рисунок 2.5** — Редагування ціни товару

Редагування товару зберігає цілісність даних, і всі зміни вносяться одразу в обидва файли. Таким чином, користувач має можливість коригувати інформацію про покупки на етапі їх введення або після певного часу, коли ціни або інші умови змінюються.

#### 6. Сповіщення про помилки

Програма також генерує сповіщення в разі помилок. Наприклад, якщо користувач намагається ввести некоректну ціну, програма виведе повідомлення про помилку (Рисунок 2.6):



**Рисунок 2.6** — Повідомлення про помилку введення

Такі сповіщення допомагають користувачеві правильно вводити дані і уникати помилок під час роботи з програмою.

### Опис мови програмування

Мова програмування Python є однією з найпопулярніших у сучасному світі ІТ. Вона широко застосовується в різних галузях: від веброзробки й автоматизації до аналізу даних, машинного навчання та створення штучного інтелекту. Простий синтаксис, гнучкість і потужна екосистема бібліотек роблять Python ефективним інструментом як для новачків, так і для досвідчених програмістів.

Python є інтерпретованою, динамічно типізованою мовою високого рівня з автоматичним управлінням пам’яттю. Вона підтримує кілька парадигм програмування: об’єктно-орієнтоване, процедурне, функціональне тощо. Однією з головних особливостей є її синтаксис, який максимально наближений до природної мови, що робить код легким для читання та супроводу[8].

### Сфера застосування

Python активно використовується у таких напрямах:

* **Веброзробка -** фреймворки Django, Flask, FastAPI дозволяють швидко створювати вебсайти, REST API та серверні застосунки.
* **Наука та аналітика -** за допомогою бібліотек Pandas, NumPy, Matplotlib, SciPy, Seaborn та ін. Python є одним із найкращих інструментів для обробки, візуалізації та аналізу даних.
* **Машинне навчання** - популярні бібліотеки TensorFlow, PyTorch, scikit-learn дають змогу створювати моделі штучного інтелекту.
* **Автоматизація -** python часто застосовується для написання скриптів, автоматизації рутинних дій, роботи з файлами, браузерами та операційною системою.
* **Розробка десктопних застосунків -** за допомогою бібліотек Tkinter, PyQt чи Kivy можна створювати GUI-програми.
* **Ігрова індустрія -** хоча Python не є основною мовою в цій сфері, його використовують для розробки ігор або прототипів (наприклад, Pygame).
* **Освіта -** через свою простоту Python часто використовується для навчання програмуванню.

### Особливості синтаксису

Однією з найбільш помітних рис мови Python є її синтаксис. Замість фігурних дужок для визначення блоків коду використовуються **відступи**, що не лише забезпечує чистоту структури, але й змушує програмістів писати акуратний код. Змінні оголошуються без вказання типу (Лістинг 2.3)[7].

Лістинг 2.3 Приклад змінних в python

x = 10

message = "Привіт, світ!"

Функції створюються за допомогою ключового слова def (Лістинг 2.4).

Лістинг 2.4 Приклад функції в python

def привітання(ім’я):

return f"Привіт, {ім’я}!"

Мова підтримує **анотації типів**, які і не є обов’язковими, але покращують читабельність і сумісність із засобами статичної перевірки (Лістинг 2.5)

Лістинг 2.5 Приклад анотації типів в python

def додати(a: int, b: int) -> int:

return a + b

Також доступні всі базові конструкції: if-elif-else, цикли for і while, обробка винятків через try-except, генератори списків, lambda-функції тощо.

Python дозволяє ефективно працювати з файлами (Лістинг 2.6).

Лістинг 2.6 Приклад читання файлу data.txt

with open("data.txt", "r", encoding="utf-8") as file:

content = file.read()

Такий синтаксис забезпечує автоматичне закриття файлу навіть у разі виникнення винятку.

### Бібліотеки та інструменти

Python має потужну екосистему бібліотек і фреймворків. Менеджер пакетів pip дозволяє швидко встановлювати зовнішні бібліотеки з офіційного репозиторію PyPI. Наприклад:

* requests – для HTTP-запитів;
* pandas – для роботи з табличними даними;
* matplotlib, seaborn – для побудови графіків;
* re – для обробки тексту за допомогою регулярних виразів;
* openpyxl, xlsxwriter – для роботи з Excel-файлами;
* sqlite3, sqlalchemy – для роботи з базами даних.

Ці інструменти особливо корисні в задачах, пов’язаних з обробкою текстових та табличних даних, які розглядаються в межах цієї курсової роботи.

### Причини вибору мови програмування Python

Мова програмування Python була обрана для реалізації програмної частини курсової роботи з кількох причин[5]:

1. **Простота синтаксису -** python дозволяє писати зрозумілий та лаконічний код без надмірної кількості символів і конструкцій. Це особливо корисно при роботі з обробкою даних, де важливо сфокусуватись на логіці, а не на синтаксичних деталях.
2. **Готові інструменти для обробки текстових і табличних даних -** бібліотеки Pandas, csv, re, openpyxl надають всі необхідні можливості для реалізації вимог курсової. Вони дозволяють читати, обробляти, фільтрувати, сортувати та зберігати інформацію в зручному вигляді.
3. **Доступна документація та спільнота -** в разі виникнення труднощів легко знайти приклади, відповіді на форумах або в офіційній документації.
4. **Масштабованість - у** разі потреби розширити програму новим функціоналом Python дозволяє легко реалізувати додаткові можливості без необхідності переписувати вже написаний код.
5. **Переносимість -** програми, написані на Python, можуть запускатись у різних операційних системах без змін у коді (за умови дотримання базових стандартів).

Таким чином, Python є найбільш доцільним вибором для реалізації задач, поставлених у межах цієї курсової роботи.

### Опис алгоритму та ідентифікатори

У цьому розділі представлено опис алгоритму роботи програми для ведення списку покупок, яка реалізована за допомогою мови програмування Python. Програма дозволяє користувачу додавати нові покупки, переглядати список покупок, редагувати або видаляти окремі товари, а також підраховувати загальну суму витрат. Усі дані зберігаються у двох форматах: текстовому (.txt) і табличному (.csv).

### Загальна логіка роботи програми

Програма має консольний інтерфейс, що побудований на основі головного меню, в якому користувач обирає одну з доступних дій. Кожна дія виконується за допомогою окремої функції[6].

Основні етапи роботи програми:

1. Виведення головного меню користувачу.
2. Обробка вибору дії.
3. Виконання відповідної функції (додавання, перегляд, редагування тощо).
4. Повернення до головного меню або завершення роботи програми.

В **Додатку Б** представлено блок-схему алгоритму роботи програми. Вона демонструє послідовність дій, яку виконує користувач та програмна логіка при обробці запитів.

### Алгоритм дій користувача:

1. Користувач запускає програму.
2. Програма виводить список доступних дій:
   * Додати покупку
   * Переглянути список
   * Підрахувати витрати
   * Видалити покупку
   * Редагувати покупку
   * Вийти
3. Користувач обирає дію за номером.
4. Програма виконує відповідну функцію.
5. Після завершення дії програма повертається до головного меню.

### Опис основних функцій:

**1. Додавання нового товару**

Функція add\_item() запитує у користувача назву товару та ціну. Ціна проходить валідацію — вона має бути числовою та не від’ємною. Після цього дані записуються у два файли: shopping\_list.txt та shopping\_list.csv.

**2. Перегляд списку покупок**

Функція view\_list() читає текстовий файл і виводить його вміст. Якщо файл порожній або відсутній, виводиться відповідне повідомлення.

**3. Підрахунок витрат**

Функція total\_spent() відкриває .csv файл і сумує значення другого стовпчика — ціни товарів. У випадку, якщо трапляється некоректне значення, воно ігнорується.

**4. Видалення покупки**

Функція remove\_item() видаляє конкретний товар зі списку за назвою. Для цього зчитуються всі рядки з файлів, фільтруються ті, що не збігаються з введеною назвою, і зберігаються назад.

**5. Редагування покупки**

Функція edit\_item() дозволяє змінити ціну наявного товару. Програма шукає збіг у файлі та замінює стару ціну новою.

### Формат збереження даних

У текстовому файлі shopping\_list.txt кожен товар зберігається у вигляді:

Молоко: 39.99 грн

Хліб: 25 грн

У CSV-файлі shopping\_list.csv рядки мають такий формат:

Молоко,39.99

Хліб,25.00

Це дозволяє легко імпортувати дані у табличні редактори або бази даних.

### Обробка помилок

У програмі реалізовано обробку типових помилок, зокрема:

* введення нечислового значення для ціни;
* відсутність файлів або порожній список;
* спроба редагувати або видалити неіснуючий товар.

Завдяки цьому програма не завершується аварійно та надає зрозумілі повідомлення для користувача.

### Ідентифікатори програми

Ідентифікатори в програмі мають інтуїтивно зрозумілі назви, що відповідають їх функціональному призначенню[5]. Наприклад:

* add\_item() — функція додавання товару;
* view\_list() — функція перегляду списку;
* csv\_file, txt\_file — шляхи до відповідних файлів;
* item, price, total — змінні, що відповідають товарам, цінам і загальній сумі.

### Висновок

Алгоритм реалізовано таким чином, щоб користування програмою було інтуїтивно зрозумілим. Кожна функція виконує чітко визначену дію. Завдяки модульній структурі програму легко доповнювати новими функціональними можливостями, наприклад, додаванням графічного інтерфейсу, сортуванням покупок або експортом у різні формати. Програма має високу стабільність за рахунок продуманого контролю помилок і перевірок введених даних.

Повний програмний код наведено у **Додатку А**.

**2.4 Тестування програми**

У даному підрозділі проведено тестування програми для ведення списку покупок. Метою тестування є перевірка правильності роботи основних функцій, відповідності виводу очікуваним результатам, а також перевірка на обробку помилок користувача.

Тестування проводилося вручну шляхом введення даних у консольний інтерфейс. Кожен тест містить опис вхідних даних, очікуваний результат і фактичний результат, підтверджений скріншотом виводу. Усі функції перевірялись як на коректні, так і на некоректні введення:

### 1. Додавання покупки (коректні дані)

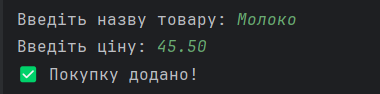
**Вхідні дані:**

Назва товару: Молоко

Ціна: 45.50

**Очікуваний результат –** успішне додавання покупки, запис у файли shopping\_list.txt і shopping\_list.csv;

**Фактичний результат – «**Покупку додано!» (Рисунок 2.7).

****

**Рисунок 2.7** – Додавання покупки з коректними даними

### 2. Додавання покупки (некоректна ціна)

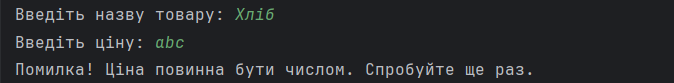
**Вхідні дані:**

Назва товару: Хліб

Ціна: abc

**Очікуваний результат –** повідомлення про помилку та запит повторного введення;

**Фактичний результат – «**Помилка! Ціна повинна бути числом. Спробуйте ще раз.» (Рисунок 2.8).

****

**Рисунок 2.8** – Введення некоректної ціни

### 3. Перегляд списку покупок

**Умова –** у файлі вже наявні товари;

**Очікуваний результат –** виведення списку товарів з цінами у форматі "Назва: Ціна грн";

**Фактичний результат** (Рисунок 2.9).

****

**Рисунок 2.9** – Виведення поточного списку покупок

### 4. Підрахунок витрат

**Умова –** у файлі shopping\_list.csv присутні 2 товари:

* Молоко – 45.5 грн
* Хліб – 25 грн

**Очікуваний результат** - «Загальна сума витрат: 70.5 грн»;

**Фактичний результат - «**Загальна сума витрат: 70.5 грн» (Рисунок 2.10).

****

**Рисунок 2.10** – Результат підрахунку загальної суми

### 5. Видалення товару (існуючий товар)

**Вхідні дані –** назва товару для видалення «Хліб»;

**Очікуваний результат –** товар успішно видалений з обох файлів;

**Фактичний результат – «**Товар 'Хліб' видалено!» (Рисунок 2.11).



**Рисунок 2.11** – Видалення товару зі списку

### 6. Видалення товару (неіснуючий товар)

**Вхідні дані –** назва товару «Сир»;

**Очікуваний результат** **–** нічого не видаляється, програма не завершується аварійно;

**Фактичний результат – «**Товар 'Сир' не знайдено у списку.» (Рисунок 2.12).

****

**Рисунок 2.12** – Спроба видалення неіснуючого товару

### 7. Редагування товару (зміна ціни)

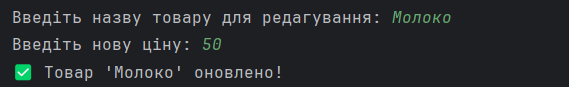
**Вхідні дані:**

Назва товару: Молоко

Нова ціна: 50

**Очікуваний результат** **–** оновлення ціни товару в обох файлах;

**Фактичний результат –** «Товар 'Молоко' оновлено!» (Рисунок 2.13).



**Рисунок 2.13** – Результат редагування ціни товару

### Висновки за результатами тестування

Усі функції програми пройшли тестування успішно. Програма стабільно працює як з коректними, так і з некоректними вхідними даними, не допускає аварійного завершення. Завдяки обробці винятків, користувач отримує зрозумілі повідомлення про помилки та підказки для повторного введення. Це свідчить про надійність і зручність програмного продукту.

Усі результати тестування підтверджено виводом у консоль, який представлено на рисунках 2.7–2.13.

## ВИСНОВКИ

У процесі виконання курсової роботи на тему "Розробка програми для ведення списку покупок" було створено програмне забезпечення, яке забезпечує основні функції для користувачів: додавання товарів до списку, редагування їх вартості, видалення товарів, а також підрахунок загальної суми витрат. Дані зберігаються як у текстовому форматі (.txt), так і в CSV форматі, що дозволяє забезпечити зручний доступ та обробку інформації.

Для розробки програми була обрана мова Python, оскільки вона має простий синтаксис та потужні бібліотеки для роботи з файлами, зокрема csv для роботи з таблицями та os для маніпуляцій з файлами. Завдяки цьому процес розробки став більш ефективним, і програму вдалося створити за короткий термін.

У програмі реалізовані механізми обробки помилок і валідації введених даних. Це важливо для уникнення аварійних ситуацій, особливо коли користувач вводить некоректні значення або намагається працювати з порожніми файлами. Кожен етап взаємодії з програмою перевіряється на коректність, що підвищує стабільність роботи програми.

Під час тестування програма продемонструвала свою працездатність у всіх передбачених сценаріях. Усі основні функції працюють коректно: можна додавати, редагувати, видаляти товар, а також підраховувати загальну суму витрат. В процесі тестування були виправлені кілька незначних помилок, що підвищило зручність користування програмою.

Перспективами розвитку програми є додавання нових функцій, таких як сортування товарів, пошук по назві або ціні, а також створення графічного інтерфейсу користувача для більш зручної роботи з програмою. Можна також розглянути інтеграцію з базами даних для зберігання великих обсягів даних або синхронізацію з мобільними пристроями.

У загальному підсумку, курсова робота успішно виконана, програма виконує всі поставлені завдання і відповідає вимогам. Це корисний інструмент для ведення обліку покупок і витрат, який може бути покращений і адаптований для більш складних потреб у майбутньому.

## ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. **Лутц, Марк. Вивчення Python. 5-е видання. – М.: Вільямс, 2016. – 1504 с.**
2. **Гомозова, Н. М. Основи програмування на мові Python. – К.: Наукова думка, 2018. – 320 с.**
3. **Сучков, В. П. Вступ до програмування на Python. – Харків: Факт, 2017. – 256 с.**
4. **Python Documentation. Python Software Foundation. [Електронний ресурс]. URL: https://docs.python.org/3/ - Дата звернення: 12.04.2025.**
5. **Чарлі Б. Поглиблене програмування на Python: практичні підходи до вирішення задач. – М.: Олімп, 2019. – 320 с.**
6. **Розробка програм для зберігання даних. Основи обробки інформації та програмування. – К.: Вища школа, 2019. – 450 с.**
7. **Роберт С. Лафор. Основи програмування: сучасний підхід. – М.: ДМК Пресс, 2019. – 384 с.**
8. **Докука, І. О. Програмування в Python для початківців. – Одеса: ОНУ, 2017. – 200 с.**
9. **Шумейко, М. І. Практичне застосування Python в програмуванні. – К.: Техніка, 2021. – 275 с.**

ДОДАТКИ

**Додаток А**

Лістинг - Повний код програми

import csv

import os

txt\_file = "shopping\_list.txt"

csv\_file = "shopping\_list.csv"

def add\_item():

item = input("Введіть назву товару: ")

while True:

price = input("Введіть ціну: ")

try:

price = float(price)

if price < 0:

print("Ціна не може бути від'ємною! Спробуйте ще раз.")

continue

break

except ValueError:

print("Помилка! Ціна повинна бути числом. Спробуйте ще раз.")

with open(txt\_file, "a", encoding="utf-8") as txt:

txt.write(f"{item}: {price} грн\n")

with open(csv\_file, "a", newline="", encoding="utf-8") as csvfile:

writer = csv.writer(csvfile)

writer.writerow([item, price])

print("✅ Покупку додано!")

def view\_list():

if not os.path.exists(txt\_file):

print("Список покупок порожній.")

return

with open(txt\_file, "r", encoding="utf-8") as txt:

print("\nСписок покупок:")

print(txt.read())

def total\_spent():

if not os.path.exists(csv\_file):

print("Список покупок порожній.")

return

total = 0

with open(csv\_file, "r", encoding="utf-8") as csvfile:

reader = csv.reader(csvfile)

for row in reader:

try:

total += float(row[1])

except ValueError:

continue

print(f"\nЗагальна сума витрат: {total} грн")

def remove\_item():

item\_to\_remove = input("Введіть назву товару для видалення: ")

if not os.path.exists(txt\_file) or not os.path.exists(csv\_file):

print("Список порожній або не існує.")

return

with open(txt\_file, "r", encoding="utf-8") as txt:

lines = txt.readlines()

item\_found = any(item\_to\_remove in line for line in lines)

if not item\_found:

print(f"❌ Товар '{item\_to\_remove}' не знайдено у списку.")

return

with open(txt\_file, "w", encoding="utf-8") as txt:

for line in lines:

if item\_to\_remove not in line:

txt.write(line)

with open(csv\_file, "r", encoding="utf-8") as csvfile:

reader = csv.reader(csvfile)

rows = list(reader)

with open(csv\_file, "w", newline="", encoding="utf-8") as csvfile:

writer = csv.writer(csvfile)

for row in rows:

if item\_to\_remove != row[0]:

writer.writerow(row)

print(f"✅ Товар '{item\_to\_remove}' успішно видалено!")

def edit\_item():

item\_to\_edit = input("Введіть назву товару для редагування: ")

if not os.path.exists(txt\_file) or not os.path.exists(csv\_file):

print("Список порожній або не існує.")

return

new\_price = input("Введіть нову ціну: ")

try:

new\_price = float(new\_price)

if new\_price < 0:

print("Ціна не може бути від'ємною!")

return

except ValueError:

print("Помилка! Ціна повинна бути числом.")

return

with open(txt\_file, "r", encoding="utf-8") as txt:

lines = txt.readlines()

with open(txt\_file, "w", encoding="utf-8") as txt:

for line in lines:

if item\_to\_edit in line:

txt.write(f"{item\_to\_edit}: {new\_price} грн\n")

else:

txt.write(line)

rows = []

with open(csv\_file, "r", encoding="utf-8") as csvfile:

reader = csv.reader(csvfile)

rows = list(reader)

with open(csv\_file, "w", newline="", encoding="utf-8") as csvfile:

writer = csv.writer(csvfile)

for row in rows:

if item\_to\_edit == row[0]:

writer.writerow([item\_to\_edit, new\_price])

else:

writer.writerow(row)

print(f"✅ Товар '{item\_to\_edit}' оновлено!")

def main():

while True:

print("\n1. Додати покупку")

print("2. Переглянути список")

print("3. Підрахувати витрати")

print("4. Видалити покупку")

print("5. Редагувати покупку")

print("6. Вийти")

choice = input("Оберіть дію: ")

if choice == "1":

add\_item()

elif choice == "2":

view\_list()

elif choice == "3":

total\_spent()

elif choice == "4":

remove\_item()

elif choice == "5":

edit\_item()

elif choice == "6":

break

else:

print("Невідома команда. Спробуйте ще раз.")

main()

**Додаток Б**

Рисунок – Блок-схема роботи програми

