Міністерство освіти та науки України

Національний аерокосмічний університет ім. М.Є. Жуковського

«Харківський авіаційний інститут»

Кафедра комп’ютерних систем, мереж і кібербезпеки

**Звіт**

| з | *Навчальна практика* |
| --- | --- |
|  | (назва дисципліни) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Виконав: | здобувач(ка) | *1* | курсу групи № | *515і* |
|  |  |  |  | (№ групи) |
| *12.07.2023* | |  | *Михайло КОСТЯНЮК* | |
| (дата, підпис) | |  | (ім’я та прізвище) | |
| Перевірив: | *к.т.н., доцент, доцент закладу вищої освіти* | | | |
|  | (науковий ступінь, вчене звання, посада) | | | |
|  | |  | *Євгеній Бабєшко* | |
| (дата, підпис) | |  | (ім’я та прізвище) | |

Харків – 2023

Зміст

[1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ 3](#_Toc140097555)

[1.1 Завдання 3](#_Toc140097556)

[1.2 Аналіз предметної області 4](#_Toc140097557)

[1.3 Постановка задачі 6](#_Toc140097558)

[2. ПРОЄКТУВАННЯ 8](#_Toc140097559)

[2.1 Структура програми 8](#_Toc140097560)

[3. РОЗРОБЛЕННЯ 12](#_Toc140097561)

[3.1 Прототипи функцій 12](#_Toc140097562)

[3.2 Схеми алгоритмів 14](#_Toc140097563)

[4. ТЕСТУВАННЯ 19](#_Toc140097564)

[4.1 Модульне тестування 19](#_Toc140097565)

[4.2 Тестові сценарії 19](#_Toc140097566)

[Висновки 29](#_Toc140097567)

[Список використаних джерел 31](#_Toc140097568)

[ДОДАТОК А. ПОСИЛАННЯ НА ГІЛКУ РЕПОЗИТОРІЮ ТА ВИХІДНІ КОДИ ПРОГРАМИ 32](#_Toc140097569)

[ДОДАТОК Б. СЕРТИФІКАТ ПРО ПРОХОДЖЕННЯ НАВЧАЛЬНОГО КУРСУ 48](#_Toc140097570)

# 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1 Завдання

Розробити програму для ведення бази даних, організованої на файлах. Програма має використовувати конфігураційний файл (текстовий) та файл з даними (дані зберігаються у двійковому вигляді) та надавати через меню набір функцій для керування цими даними.

Реалізовувані програмою функції поділяються на три групи:

* основні функції керування даними;
* спеціальні функції (визначаються варіантом завдання);
* додаткові функції (не є обов’язковими для реалізації, але дозволяють отримати додаткові бали).

Основні функції:

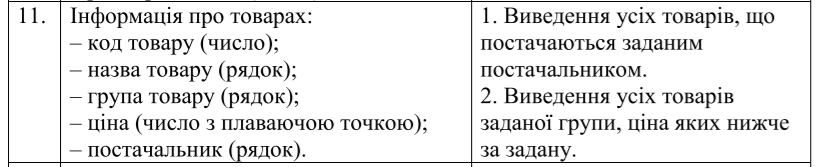
* додавання запису (користувач вводить дані, а програма автоматично задає ідентифікаційний номер – порядковий номер запису у файлі);
* видалення запису, перегляд записів, змінювання запису (змінюється одне або декілька полів даних для заданого ідентифікаційним номером запису).

Пам’ять під зберігання даних має виділятися динамічно (використання масивів з фіксованим розміром не допускається, необхідне використання зв’язних списків). Додаткові функції:

* зберігання бази даних у файлі з заданим ім’ям;
* експорт бази даних у текстовий файл формату CSV з заданим ім’ям (кожен запис бази з нового рядка, значення розділити крапками з комою);
* змінювання будь-якого з налаштувань файлу налаштувань через інтерфейс програми.

Варіант завдання визначає дані, які необхідно зберігати в базі, та перелік спеціальних функцій. Розробити звіт [1].

Варіант 11



## 1.2 Аналіз предметної області

Структуры в языке С використовуються для організації інформації, яка складається з різних типів даних. Вони дозволяють створювати складніші об'єкти, які містять поля різних типів і можуть бути оброблені як єдиний об'єкт [1].

Для оголошення структури використовується ключове слово **struct**, за яким слідує ім'я структури та оголошення її полів. Поля структури можуть бути будь-якого типу даних, включаючи примітивні типи (наприклад, цілі числа, дійсні числа) і навіть інші структури [1, 2].

Основні переваги використання структур включають [2]:

* Організація складних об'єктів з різними типами даних.
* Логічна групування пов'язаних даних.
* Зручний доступ до полів структури за допомогою оператора крапка (**.**).
* Передача структур як аргументів функцій або їх повернення з функцій.

Для роботи з файлами в мові С існують різні функції та операції. Для відкриття файлу використовується функція **fopen**, яка приймає ім'я файлу та режим відкриття. Режими відкриття можуть бути "r" (для читання), "w" (для запису) або "a" (для додавання) [3, 4].

Після відкриття файлу можна здійснювати різні операції [4]:

* Зчитування та запис даних за допомогою функцій **fread** та **fwrite**.
* Зчитування рядків з файлу за допомогою функції **fgets**.
* Запис рядків у файл за допомогою функції **fputs**.
* Переміщення покажчика у файлі за допомогою функції **fseek**.
* Закриття файлу за допомогою функції **fclose**.

Одним з плюсів роботи з файлами є можливість зберігати дані між сеансами роботи програми. Файли можна використовувати для зберігання постійних даних або результатів обчислень, щоб їх можна було використовувати пізніше [5].

Однак, робота з файлами також має свої мінуси [6]:

* Необхідність відкривати та закривати файли, що може створювати проблеми, якщо це забуто або зроблено неправильно.
* Читання та запис у файл можуть бути повільними, особливо при роботі з великими обсягами даних.
* Ризик пошкодження або втрати даних, якщо файл не збережено або втрачено.

Символьне та бінарне збереження інформації у файлі мають свої відмінності.

Символьне збереження використовує текстовий формат, де дані представлені у вигляді символів. Це зручно для читання та редагування файлів вручну, оскільки дані можуть бути розуміними людиною. Однак, символьне збереження займає більше місця у файлі та може бути повільним для обробки великих обсягів даних [5, 6].

Бінарне збереження використовує двійковий формат, де дані представлені у вигляді бінарного коду. Це ефективно для збереження великих обсягів даних, оскільки займає менше місця у файлі та дозволяє швидше зчитування та запис. Однак, бінарні файли не можуть бути легко читані або редаговані вручну, і для цього потрібні спеціальні інструменти або програми [5, 6].

У випадку символьного збереження, якщо дані мають бути представлені у текстовій формі або потребують редагування, символьне збереження є кращим варіантом. Наприклад, якщо дані є читабельними для користувача або мають бути легко редагованими в текстовому редакторі.

У випадку бінарного збереження, коли швидкість та ефективність мають важливе значення, а читабельність або редагування не є пріоритетом, бінарне збереження може бути кращим варіантом. Наприклад, коли потрібно зберігати великі масиви чисел або структури з великою кількістю полів [5, 6].

В загальному, вибір між символьним та бінарним збереженням залежить від конкретних вимог проекту і компромісів між зручністю, швидкістю та розміром файлів.

## 1.3 Постановка задачі

Вивчити особливості роботи зі структурами, текстовими та двійковими файлами, динамічним виділенням пам'яті. Закріпити знання, здобуті у рамках дисципліни.

Необхідно розробити такі завдання для реалізації програми управління базою даних на С мові:

1. Створити конфігураційний файл: реалізувати функцію, яка читатиме конфігураційний файл, наприклад, за допомогою читання текстового файлу, та завантажувати налаштування програми, такі як ім'я файлу з даними та інші параметри.
2. Створити файл даних: реалізувати функцію для створення файлу даних, де зберігатимуться записи бази даних. Ви можете використовувати бінарний формат для зберігання даних.
3. Додавання запису: реалізувати функцію, яка дозволить користувачеві вводити дані та автоматично надавати ідентифікаційний номер (наприклад, порядковий номер запису у файлі). Доданий запис необхідно зберегти у файлі даних.
4. Видалення запису: реалізувати функцію, яка дозволить користувачеві видаляти записи за ідентифікаційним номером. При видаленні запису позначте його як віддалений або видаліть фактично з файлу даних.
5. Перегляд записів: реалізувати функцію, щоб переглянути всі записи бази даних. Функція повинна відображати інформацію про кожен запис, включаючи ідентифікаційний номер, дані та інші відомості.
6. Зміна запису: реалізувати функцію, яка дозволить користувачеві змінювати одну або кілька полів даних для ідентифікаційного номера запису. Змінені дані мають бути збережені у файлі даних.
7. Додаткові функції: реалізувати додаткові функції, якщо вказано у варіанті завдання. Наприклад, функція збереження бази даних у файлі, експорту бази даних у текстовий файл формату CSV або зміни налаштувань програми через інтерфейс програми.

# 2. ПРОЄКТУВАННЯ

## 2.1 Структура програми

У програмі використовуються такі структури:

* AuctionProduct - структура, що представляє інформацію про товар у базі даних.
* Config - структура, що містить інформацію з файлу конфігурації.

Програма також використовує макроси для визначення максимальної довжини імені, імені конфігураційного файлу та максимальної довжини ліцензійного ключа.

Основна логіка програми знаходиться в циклі do-while у функції main, де користувач вибирає операції з меню та відповідні функції викликаються для виконання завдань.

Діаграма класів для даної програми буде виглядати наступним чином:

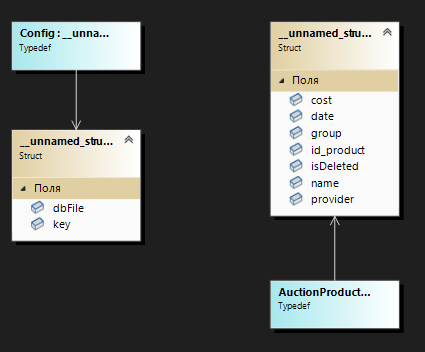


Рисунок 2.1 – Діаграма класів програми

Пояснення:

* **AuctionProduct**: клас, що представляє товар на аукціоні. Містить поля, такі як ідентифікаційний номер (**id\_product**), назва (**name**), група (**group**), ціна (**cost**), постачальник (**provider**), дата та час (**date**) і прапорець видалення (**isDeleted**).
* **Config**: клас, що представляє конфігураційні налаштування програми. Містить поля, такі як назва файлу бази даних (**dbFile**) і ліцензійний ключ (**key**).
* **Database**: клас, що представляє базу даних. Містить поле **products**, яке є масивом об'єктів **AuctionProduct**, і поле **count**, яке відстежує кількість записів у базі.
* **Menu**: клас, що представляє меню програми. Містить поля **products** і **count**, які посилаються на об'єкт **Database**, і поле **isValidKey**, яке вказує на валідність ліцензійного ключа.
* **Program**: головний клас програми, який включає точку входу **main()**. Виконує ініціалізацію конфігураційних налаштувань, викликає функції управління даними і взаємодіє з користувачем через меню. Звертається до об'єктів **Database** і **Menu**.

Детальний опис модулів програми були продемонстровані у таблиці 2.1, де детально розписані як вони функціонують і для чого необхідні.

Таблиця 2.1 − Модулі програми

|  |  |
| --- | --- |
| **Модуль** | **Опис** |
| **getConfig()** | Функція для отримання налаштувань конфігураційного файлу. Зчитує дані з файлу конфігурації та повертає структуру **Config**, що містить інформацію про файл бази даних та ключ доступу. |
| **addRecord()** | Функція додавання запису до бази даних. Користувач вводить дані, а функція автоматично визначає ідентифікаційний номер запису. Повертає покажчик на оновлений масив записів **products** та оновлену кількість **count**. |

Продовження таблиці 2.1

|  |  |
| --- | --- |
| **deleteRecord()** | Функція видалення запису з бази даних. Приймає масив записів **products**, кількість записів **count** та ідентифікаційний номер запису **id**\_**product** для видалення. Позначає запис як віддалений або видаляє фактично з масиву. |
| **viewRecords()** | Функція, щоб переглянути всі записи бази даних. Приймає масив записів **products** та кількість записів **count**. Відображає інформацію про кожен запис, включаючи ідентифікаційний номер, дані та інші відомості. |
| **modifyRecord()** | Функція зміни запису в базі даних. Приймає масив записів **products**, кількість записів **count** та ідентифікаційний номер запису **id**\_**product** для зміни. Користувач може ввести нові значення для одного або кількох полів даних запису. |
| **printProductsInRange()** | Функція для виведення всіх товарів заданої групи, ціна яких нижча за задану. Приймає масив записів products, кількість записів **count**, назву групи товарів **group** та максимальну вартість **maxCost**. Виводить інформацію про кожен товар, який відповідає умовам. |
| **saveDatabase()** | Функція збереження бази даних у файлі. Приймає масив записів **products**, кількість записів **count** та запитує у користувача ім'я файлу для збереження. Записує дані у вказаний файл у двійковому форматі. |

Продовження таблиці 2.1

|  |  |
| --- | --- |
| **Модуль** | **Опис** |
| **printProductsByProvider()** | Функція виведення всіх товарів, поставляемых заданим постачальником. Приймає масив записів продукції, кількість записів **count** та назву постачальника **purveyor**. Виводить інформацію про кожен товар, який постачається вказаним постачальником. |
| **exportCSV()** | Функція експорту бази даних до текстового файла формату **CSV**. Приймає масив записів **products**, кількість записів **count** та ім'я файлу **filename** для експорту. Експортує дані у вказаний файл у форматі **CSV**. |
| **open\_input\_database()** | Функція для відкриття бази даних із файлу. Приймає вказівник на масив записів **products**, вказівник на кількість записів **count**. Запитує у користувача ім'я файлу бази даних та зчитує дані з файлу, завантажуючи їх у масив записів. |
| **createDatabaseFile()** | Функція створення файлу бази даних. Приймає ім'я файлу бази даних **filename**. Якщо файл не існує, він створює його. |
| **main()** | Основна функція програми. У ній реалізовано головне меню, у якому користувач може вибрати з функцій управління базою даних. Тут також міститься обробка логіки доступу до функцій із використанням ліцензійного ключа. |

# 3. РОЗРОБЛЕННЯ

## 3.1 Прототипи функцій

Було розроблено такі функції:

1. Функція **getConfig()**:

* Параметри: відсутні.
* Повертане значення: структура **Config**, яка містить інформацію з конфігураційного файлу (назва файлу бази даних та ліцензійний ключ).

1. Функція **addRecord(AuctionProduct\*\* products, int\* count)**:
   * Параметри: вказівник на вказівник на масив структур **AuctionProduct** (масив товарів) та вказівник на змінну **count** (кількість записів).
   * Повертане значення: вказівник на масив структур **AuctionProduct** (масив товарів після додавання нового запису).
2. Функція **deleteRecord(AuctionProduct\* products, int\* count, int id\_product)**:
   * Параметри: масив структур **AuctionProduct** (масив товарів), вказівник на змінну **count** (кількість записів) та ідентифікаційний номер запису.
   * Повертане значення: відсутнє.
3. Функція **viewRecords(AuctionProduct\* products, int count)**:
   * Параметри: масив структур **AuctionProduct** (масив товарів) та кількість записів.
   * Повертане значення: відсутнє.
4. Функція **modifyRecord(AuctionProduct\* products, int count, int id\_product)**:
   * Параметри: масив структур **AuctionProduct** (масив товарів), кількість записів та ідентифікаційний номер запису
   * Повертане значення: відсутнє
5. Функція **printProductsInRange(AuctionProduct\* products, int count, const char\* group, float maxCost)**:
   * Параметри: масив структур **AuctionProduct** (масив товарів), кількість записів, назва групи товарів та максимальна ціна.
   * Повертане значення: відсутнє.
6. Функція **printProductsByProvider(AuctionProduct\* products, int count, const char\* purveyor)**:
   * Параметри: масив структур **AuctionProduct** (масив товарів), кількість записів та назва постачальника.
   * Повертане значення: відсутнє.
7. Функція **saveDatabase(AuctionProduct\* products, int count)**:
   * Параметри: масив структур **AuctionProduct** (масив товарів) та кількість записів.
   * Повертане значення: відсутнє.
8. Функція **exportCSV(AuctionProduct\* products, int count, const char\* filename)**:
   * Параметри: масив структур **AuctionProduct** (масив товарів), кількість записів та назва файлу для експорту.
   * Повертане значення: відсутнє.
9. Функція **open\_input\_database(AuctionProduct\*\* products, int\* count)**:
   * Параметри: вказівник на вказівник на масив структур **AuctionProduct** (масив товарів) та вказівник на змінну **count** (кількість записів).
   * Повертане значення: відсутнє.
10. Функція **createDatabaseFile(const char\* filename)**:
    * Параметри: назва файлу бази даних
    * Повертане значення: відсутнє

В основній функції **main()** реалізовано меню для взаємодії з користувачем та обробки вибраних опцій.

## 3.2 Схеми алгоритмів

На рисунку 3.1 наведено схему алгоритму функції getConfig():

* Відкрийте конфігураційний файл для читання.
* Якщо файл не існує, створити новий файл і записати значення за промовчанням.
* Прочитати конфігураційний файл та завантажити налаштування до структури Config.
* Закрити файл та повернути структуру Config.

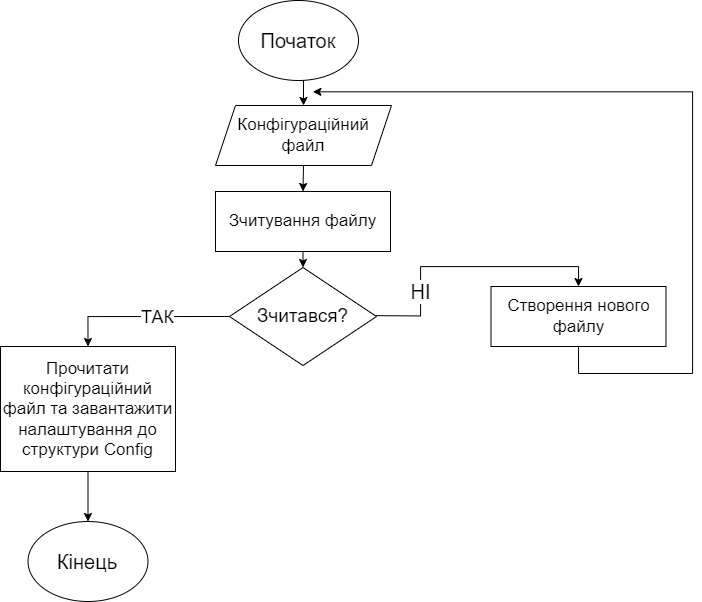


Рисунок 3.1 – Схема алгоритму функції getConfig()

На рисунку 3.2 наведено схему алгоритму функції addRecord():

* Створити тимчасову структуру AuctionProduct для зберігання даних про новий товар.
* Запросити у користувача дані про товар: назву, групу, ціну, постачальника, дату та час.
* Виділити пам'ять для нового запису в масиві продукції з використанням realloc.
* Зберегти новий запис у масиві та збільшити лічильник count.
* Повернути оновлений масив products.

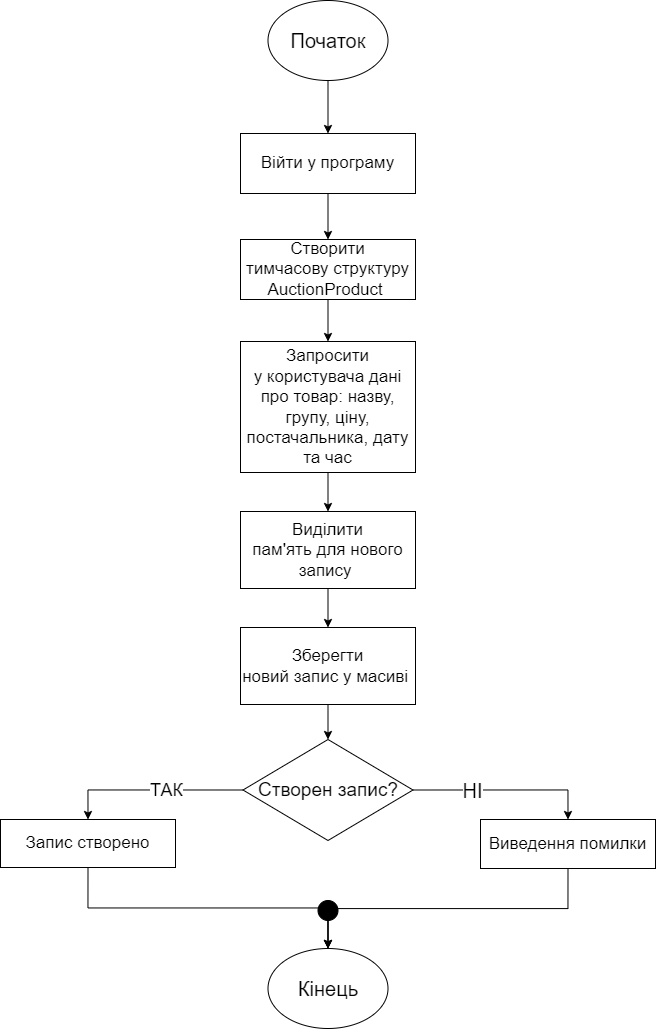


Рисунок 3.2 – Схема алгоритму функції addRecord()

На рисунку 3.3 наведено схему алгоритму функції printProductsInRange():

* Вивести заголовок таблиці з іменами полів.
* Для кожного товару в масиві products (не поміченого як віддалений), що збігається із заданою групою і має ціну нижче за задане значення, вивести дані про товар.

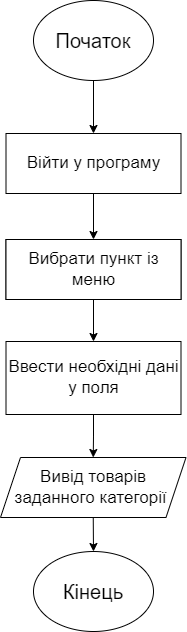


Рисунок 3.3 – Схема алгоритму функції printProductsInRange()

На рисунку 3.4 наведено схему алгоритму функції printProductsByProvider():

* Вивести заголовок таблиці з іменами полів.
* Для кожного товару в масиві products (не поміченого як віддалений), що належить заданому постачальнику, вивести дані про товар.

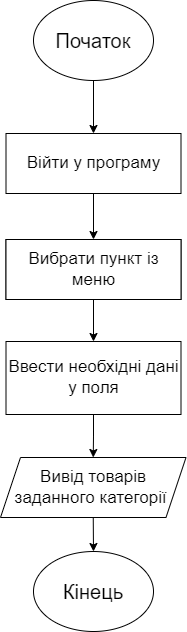


Рисунок 3.4 – Схема алгоритму функції printProductsByProvider()

На рисунку 3.5 наведено схему алгоритму основної функції main():

* Отримати конфігурацію з файлу конфігурації.
* Запросити ліцензійний ключ користувача.
* Перевірити дійсність ключа.
* Якщо ключ дійсний, вивести повідомлення про успішну перевірку.
* Якщо ключ недійсний, вивести повідомлення про недійсний ключ.
* Створити масив products та ініціалізувати лічильник count.
* Відкрити файл бази даних, якщо він існує, і завантажити дані в масив products.
* Вивести меню та обробляти вибір користувача з використанням різних функцій залежно від вибраного пункту.
* Повторювати до вибору пункту "0" (вихід).
* Звільнити пам'ять, виділену для масиву products.
* Завершити програму.

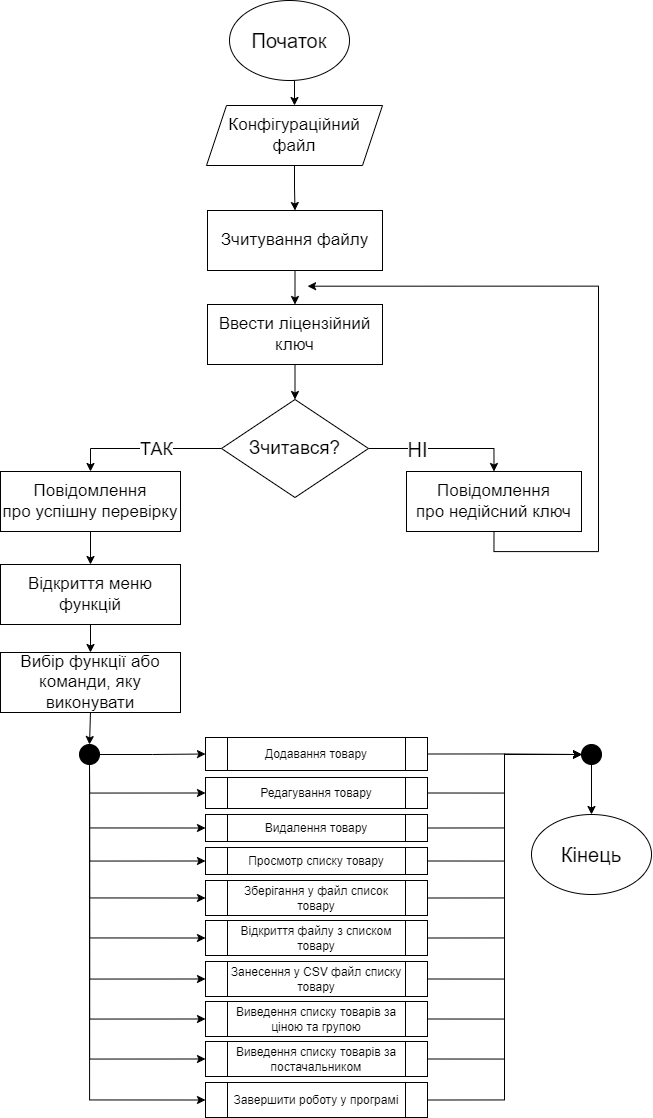


Рисунок 3.5 – Схема алгоритму функції printProductsByProvider()

# 4. ТЕСТУВАННЯ

## 4.1 Модульне тестування

Було розроблено такий перелік модульних тестів:

* додавання запису;
* видалення запису;
* перегляду записів;
* зміни запису;
* виведення товарів за групою та ціновим діапазоном;
* виведення товарів за постачальником;
* збереження бази даних у файлі;
* експорту бази даних у текстовий файл формату CSV;
* виведення бази даних з файлу;
* введення правильного ліцензійного ключа.

## 4.2 Тестові сценарії

Таблиця 4.1 – Тестовий сценарій 1

|  |  |  |
| --- | --- | --- |
| Мета | | Перевірити введення правильного ліцензійного ключа |
| Передумови | | Запустити програму |
| Кроки | | |
| № | Дія | |
| 1 | Запустити програму | |
| 2 | Ввести ключ | |
| Очікуваний результат для кроків | | |
| № | Результат кроку | |
| 3 | Покаже повідомлення про успішне введення | |

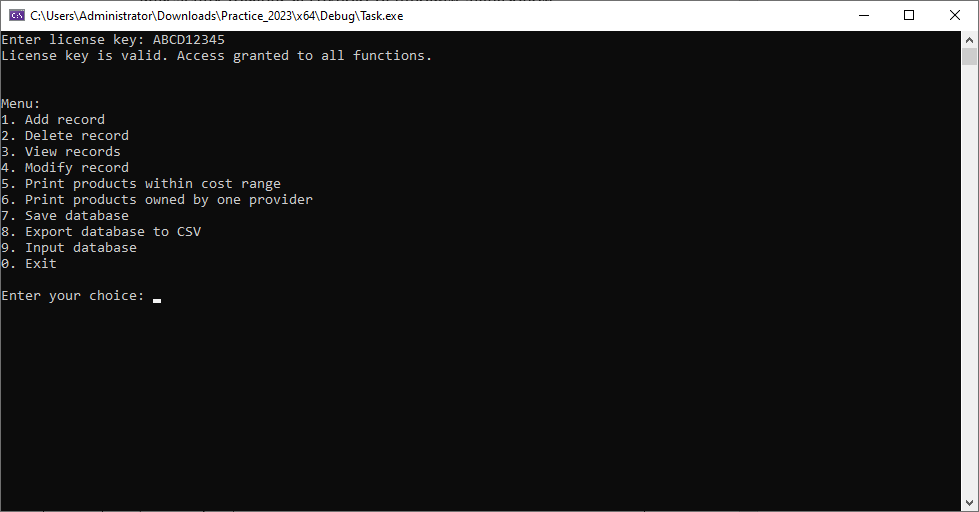


Рисунок 4.1 – Скриншот проходження тесту

Таблиця 4.2 – Тестовий сценарій 2

|  |  |  |
| --- | --- | --- |
| Мета | | Перевірити виведення бази даних з файлу |
| Передумови | | Запустити програму та вибрати пункт 9 |
| Кроки | | |
| № | Дія | |
| 1 | Запустити програму | |
| 2 | Ввести ключ | |
| 3 | Введення існуючого файлу бази даних | |
| 4 | Перевірка, чи були коректно зчитані всі записи з файлу | |
| 5 | Перевірка, чи відповідає кількість зчитаних записів кількості записів у пам'яті | |
| Очікуваний результат для кроків | | |
| № | Результат кроку | |
| 6 | Покаже результати у вигляді повідомлення | |

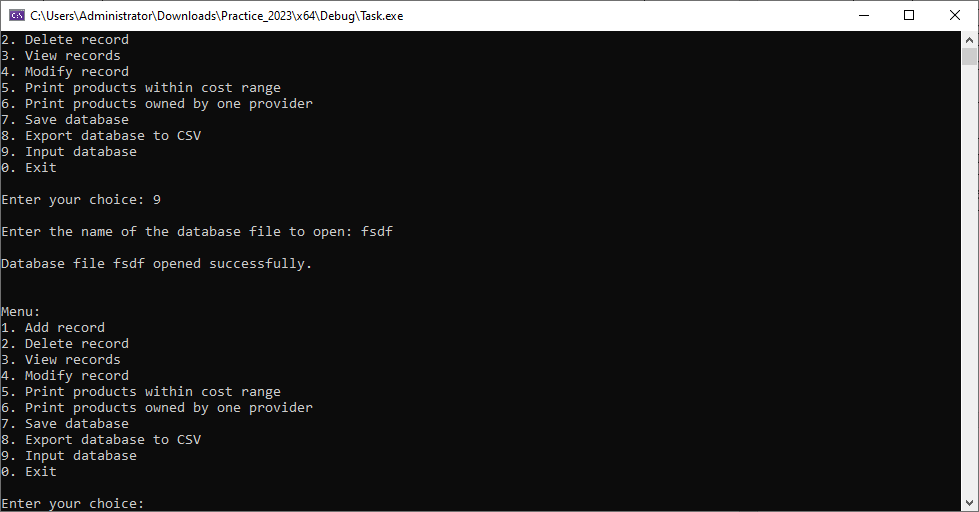


Рисунок 4.2 – Скриншот проходження тесту

Таблиця 4.3 – Тестовий сценарій 3

|  |  |  |
| --- | --- | --- |
| Мета | | Перевірити функцію перегляду записів |
| Передумови | | Запустити програму та вибрати пункт 3 |
| Кроки | | |
| № | Дія | |
| 1 | Запустити програму | |
| 2 | Ввести ключ | |
| 3 | Перевірка виведення всіх записів бази даних | |
| 4 | Перевірка правильності виведення інформації про кожен запис | |
| Очікуваний результат для кроків | | |
| № | Результат кроку | |
| 5 | Покаже результати у вигляді таблиці | |

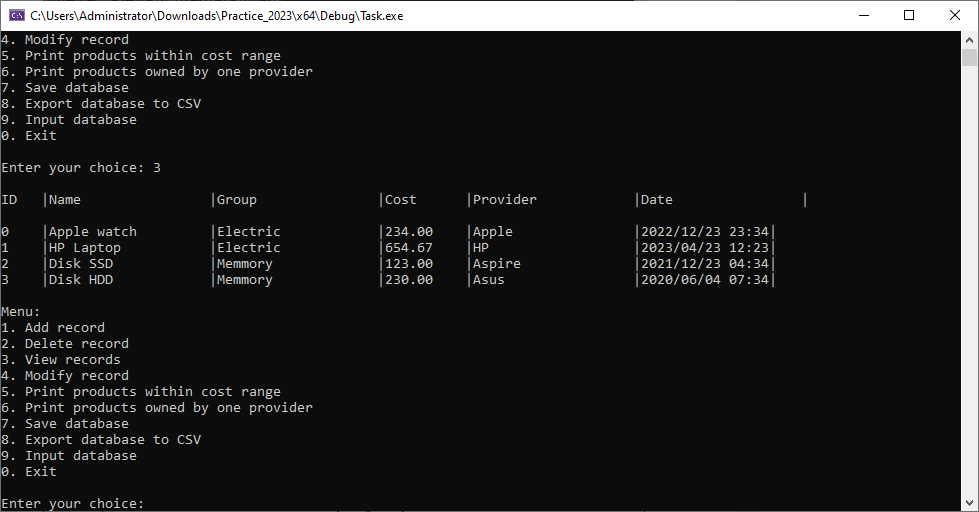


Рисунок 4.3 – Скриншот проходження тесту

Таблиця 4.4 – Тестовий сценарій 4

|  |  |  |
| --- | --- | --- |
| Мета | | Перевірити функцію експорту бази даних у текстовий файл формату CSV |
| Передумови | | Запустити програму та вибрати пункт 8 |
| Кроки | | |
| № | Дія | |
| 1 | Запустити програму | |
| 2 | Ввести ключ | |
| 3 | Запуск функції експорту бази даних у CSV-файл | |
| 4 | Перевірка, чи був створений CSV-файл з відповідною назвою | |
| 5 | Перевірка, чи були експортовані всі записи бази даних у CSV-файл | |
| 6 | Запуск функції експорту бази даних у CSV-файл | |
| Очікуваний результат для кроків | | |
| № | Результат кроку | |
| 7 | Покаже результати у вигляді повідомлення | |

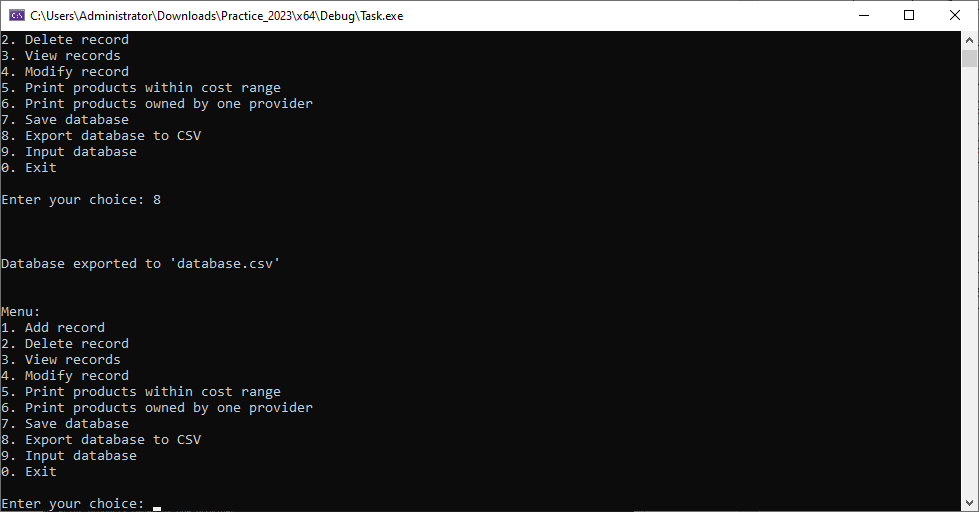


Рисунок 4.4 – Скриншот проходження тесту

Таблиця 4.5 – Тестовий сценарій 5

|  |  |  |
| --- | --- | --- |
| Мета | | Перевірити функцію зміни запису |
| Передумови | | Запустити програму та вибрати пункт 4 |
| Кроки | | |
| № | Дія | |
| 1 | Запустити програму | |
| 2 | Ввести ключ | |
| 3 | Введення коректного ідентифікаційного номера запису для зміни | |
| 4 | Введення нових значень для полів запису | |
| 5 | Перевірка, чи були змінені відповідні поля запису в базі даних | |
| Очікуваний результат для кроків | | |
| № | Результат кроку | |
| 6 | Покаже результати у вигляді повідомлення | |

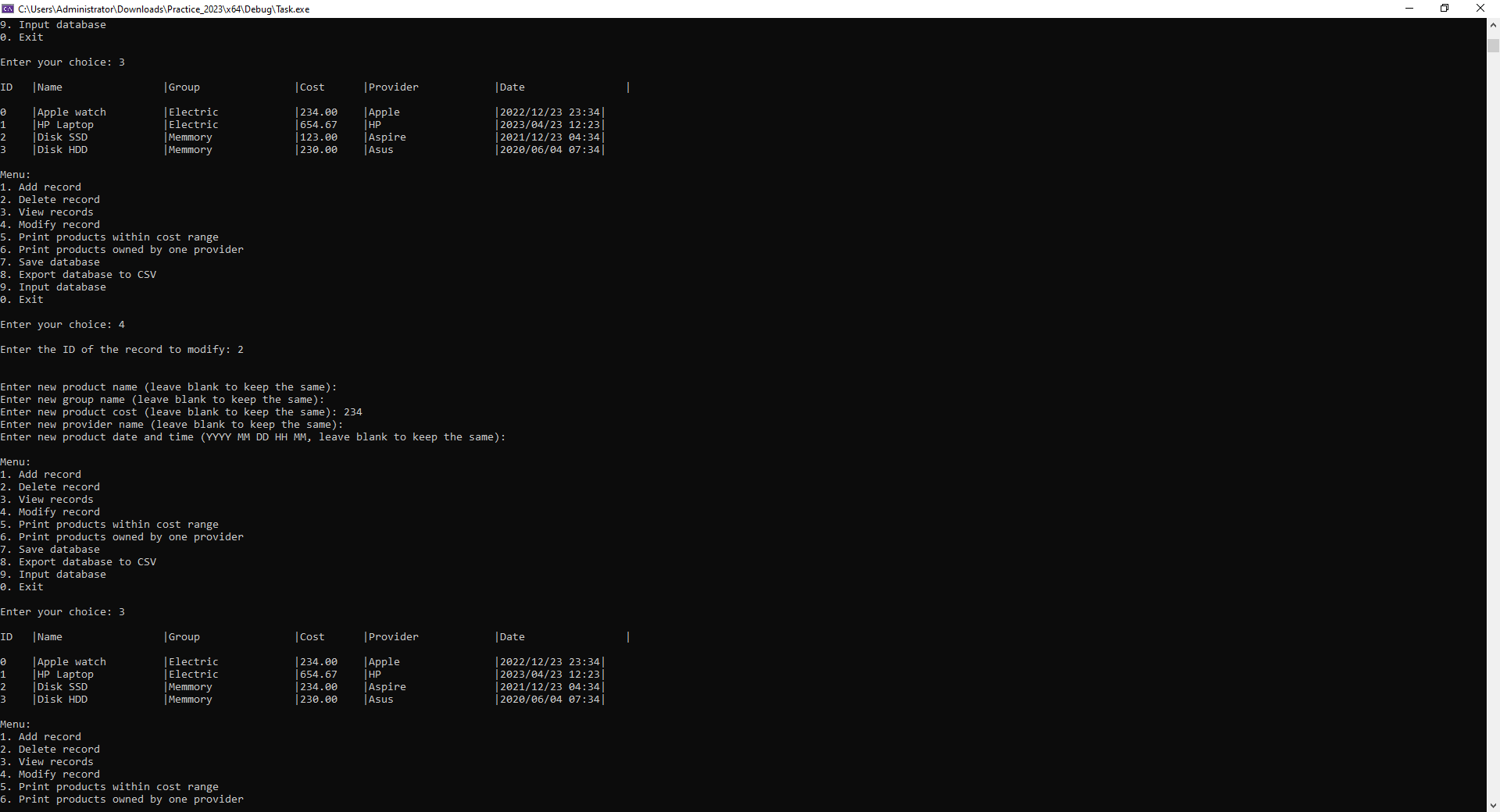


Рисунок 4.5 – Скриншот проходження тесту

Таблиця 4.6 – Тестовий сценарій 6

|  |  |  |
| --- | --- | --- |
| Мета | | Перевірити функцію додавання запису |
| Передумови | | Запустити програму та вибрати пункт 1 |
| Кроки | | |
| № | Дія | |
| 1 | Запустити програму | |
| 2 | Ввести ключ | |
| 3 | Введення коректних даних для додавання запису | |
| 4 | Перевірка, чи був доданий запис до бази даних | |
| 5 | Перевірка збільшення кількості записів у базі даних | |
| Очікуваний результат для кроків | | |
| № | Результат кроку | |
| 6 | Покаже результати у вигляді повідомлення | |

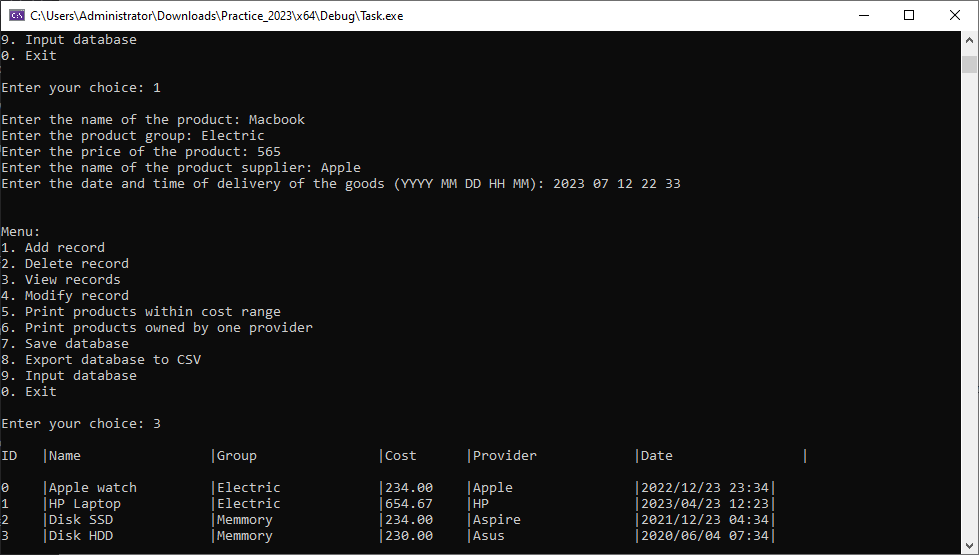


Рисунок 4.6 – Скриншот проходження тесту

Таблиця 4.7 – Тестовий сценарій 7

|  |  |  |
| --- | --- | --- |
| Мета | | Перевірити виведення товарів за групою та ціновим діапазоном |
| Передумови | | Запустити програму та вибрати пункт 5 |
| Кроки | | |
| № | Дія | |
| 1 | Запустити програму | |
| 2 | Ввести ключ | |
| 3 | Введення коректної групи та максимальної ціни | |
| 4 | Перевірка, чи були виведені товари з відповідною групою та ціновим діапазоном | |
| 5 | Перевірка правильності виведення інформації про кожен товар | |
| Очікуваний результат для кроків | | |
| № | Результат кроку | |
| 6 | Покаже результати у вигляді таблиці | |

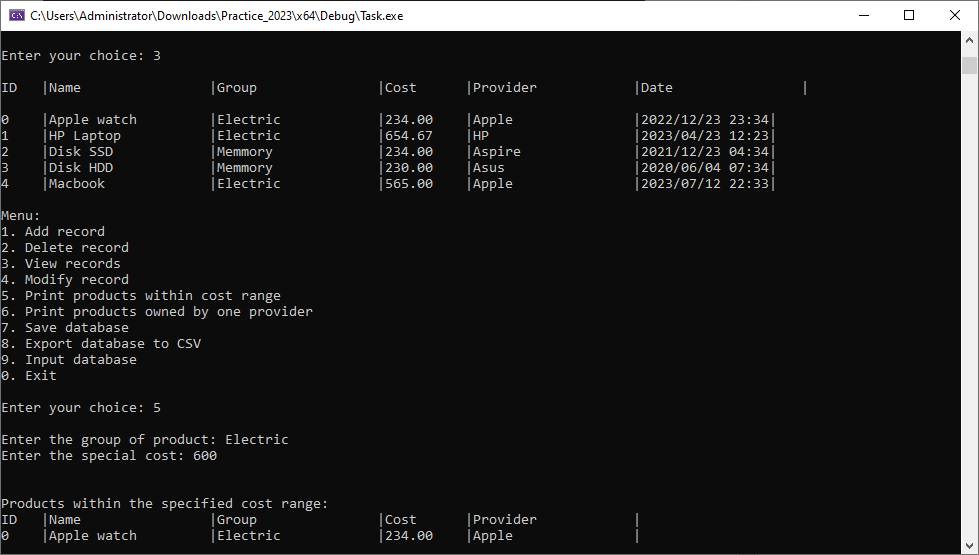


Рисунок 4.7 – Скриншот проходження тесту

Таблиця 4.8 – Тестовий сценарій 8

|  |  |  |
| --- | --- | --- |
| Мета | | Перевірити виведення товарів за постачальником |
| Передумови | | Запустити програму та вибрати пункт 6 |
| Кроки | | |
| № | Дія | |
| 1 | Запустити програму | |
| 2 | Ввести ключ | |
| 3 | Введення коректного постачальника | |
| 4 | Перевірка, чи були виведені товари, які постачаються відповідним постачальником | |
| 5 | Перевірка правильності виведення інформації про кожен товар | |
| Очікуваний результат для кроків | | |
| № | Результат кроку | |
| 6 | Покаже результати у вигляді таблиці | |

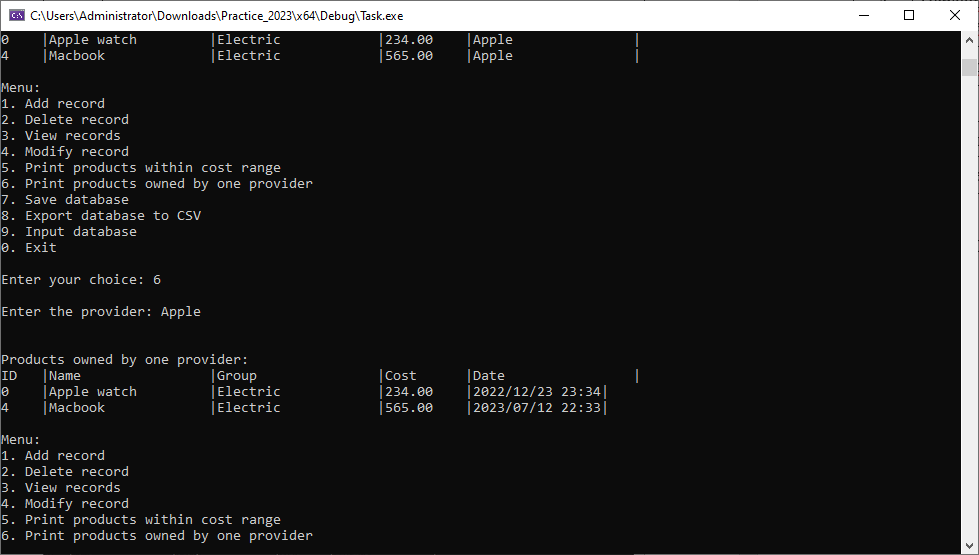


Рисунок 4.8 – Скриншот проходження тесту

Таблиця 4.9 – Тестовий сценарій 9

|  |  |  |
| --- | --- | --- |
| Мета | | Перевірити видалення запису |
| Передумови | | Запустити програму та вибрати пункт 2 |
| Кроки | | |
| № | Дія | |
| 1 | Запустити програму | |
| 2 | Ввести ключ | |
| 3 | Введення коректного ідентифікаційного номера запису для видалення | |
| 4 | Перевірка, чи був видалений запис з бази даних | |
| 5 | Перевірка зменшення кількості записів у базі даних | |
| Очікуваний результат для кроків | | |
| № | Результат кроку | |
| 6 | Покаже результати у вигляді повідомлення | |

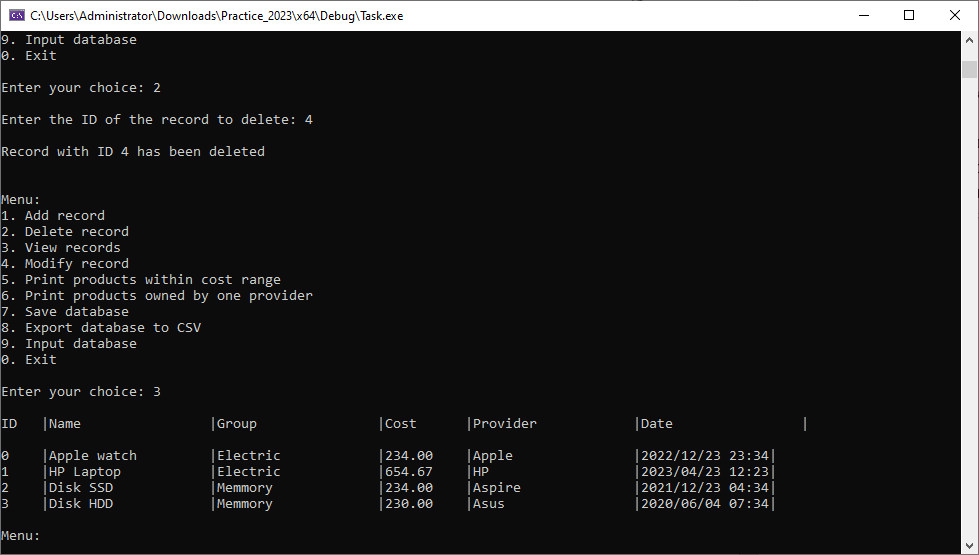


Рисунок 4.9 – Скриншот проходження тесту

Таблиця 4.10 – Тестовий сценарій 10

|  |  |  |
| --- | --- | --- |
| Мета | | Перевірити збереження бази даних у файл |
| Передумови | | Запустити програму та вибрати пункт 7 |
| Кроки | | |
| № | Дія | |
| 1 | Запустити програму | |
| 2 | Ввести ключ | |
| 3 | Запуск функції збереження бази даних | |
| 4 | Перевірка, чи був створений файл з відповідною назвою | |
| 5 | Перевірка, чи були збережені всі записи бази даних у файл | |
| Очікуваний результат для кроків | | |
| № | Результат кроку | |
| 6 | Покаже результати у вигляді повідомлення | |

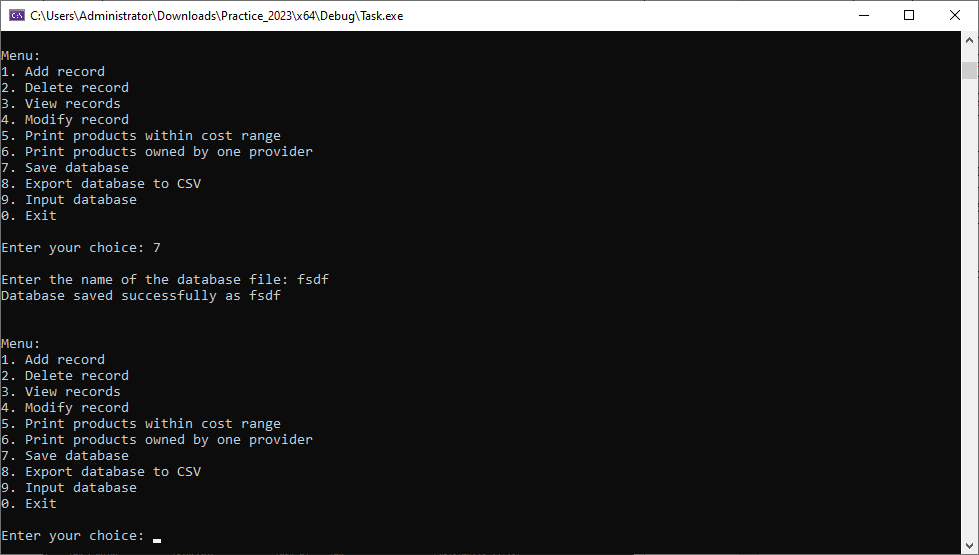


Рисунок 4.10 – Скриншот проходження тесту

# Висновки

Після виконання завдань навчальної практики було отримано такі результати:

* на основі навчальної практики було розроблено програму для ведення бази даних товарів. Програма дозволяє додавати нові записи, видаляти та змінювати існуючі записи, а також переглядати всі записи бази даних. Додатково, програма надає функції для виведення товарів за заданою групою та ціновим діапазоном, а також за постачальником. Програма також підтримує збереження бази даних у файлі та експорт бази даних у текстовий файл формату CSV.
* у процесі розробки програми було проведено аналіз предметної області, поставлено задачу і спроектовано структуру програми. Були розроблені прототипи функцій та створені схеми алгоритмів для виконання основних функцій програми. Для перевірки коректності роботи програми було проведено модульне тестування та розроблено тестові сценарії.
* отримані результати демонструють успішне виконання поставленої задачі і реалізацію основних функцій програми для ведення бази даних товарів.
* завершена програма та вихідні коди можуть бути знайдені у відповідній гілці репозиторію. Додатково було отримано сертифікат про проходження навчального курсу.

Подальшим розвитком програми може бути:

1. Реалізація додаткових функцій:

* Пошук записів за певним критерієм (наприклад, назва товару, група, постачальник).
* Сортування записів за певним полем (наприклад, за назвою товару, ціною).
* Видалення всіх записів з бази даних.
* Відновлення видалених записів з бази даних.
* Експорт бази даних у інші формати (наприклад, JSON, XML).

2. Покращення інтерфейсу користувача:

* Додавання можливості вибору функцій з графічного меню.
* Покращення форматування виведення записів для більшої зручності користувача.
* Додавання можливості введення дати та часу у зручному форматі.

3. Вдосконалення системи збереження даних:

* Використання бази даних (наприклад, SQLite) замість файлів для кращої швидкості та ефективності роботи з даними.
* Застосування індексації для прискорення пошуку та сортування даних.
* Забезпечення резервного копіювання бази даних для запобігання втрати даних.

4. Додаткова обробка та перевірка введених даних:

* Перевірка коректності введених даних, таких як числові обмеження, формат дати та часу.
* Забезпечення валідації даних перед додаванням до бази.

5. Забезпечення безпеки даних:

* Шифрування бази даних або окремих полів для захисту конфіденційної інформації.
* Встановлення автентифікації та авторизації користувачів для обмеження доступу до функцій та даних.

6. Оптимізація продуктивності програми:

* Використання ефективних алгоритмів для операцій з даними, таких як пошук та сортування.
* Масштабування програми для роботи з великими обсягами даних.

# Список використаних джерел

1. Правила оформлення навчальних і науково-дослідних документів: навч. посіб. / Ю. А. Воробйов, Ю. О. Сисоєв. 4-те вид. URL: http://library.khai.edu/library/fulltexts/metod/Vorobjov\_Pravila.pdf (дата звернення: 01.07.2023).
2. Презентація. Структура в мові С. URL: https://naurok.com.ua/prezentaciya-struktura-v-movi-s-298021.html (дата звернення: 11.07.2023).
3. Використання структур. URL: http://cpp.dp.ua/vykorystannya-struktur/ (дата звернення: 11.07.2023).
4. Робота зі структурами в мові програмування Сі + +. URL: https://ua-referat.com/Робота\_зі\_структурами\_в\_мові\_програмування\_Сі (дата звернення: 11.07.2023).
5. Поняття файла. URL: http://cpp.dp.ua/ponyattya-fajla/ (дата звернення: 12.07.2023).
6. Файли в мові С. URL: https://vseosvita.ua/library/embed/0100dxkb-78a3.docx.html (дата звернення: 12.07.2023).

# ДОДАТОК А. ПОСИЛАННЯ НА ГІЛКУ РЕПОЗИТОРІЮ ТА ВИХІДНІ КОДИ ПРОГРАМИ

Посилання на гілку репозиторію:

Вихідний код програми:

/\*\*

\* @file TAsk.cpp

\* @author Костянюк Михайло Віталійович, гр. 515і, варіант 11

\* @date 11.07.2023

\* @brief Практика

\*

\* База даних студентів

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_NAME\_LENGTH 50

#define CONFIG\_FILE "configurating.txt"

#define MAX\_KEY\_LENGTH 10

typedef struct {

int id\_product; /\* Код товару \*/

char name[MAX\_NAME\_LENGTH]; /\* Назва товару \*/

char group[MAX\_NAME\_LENGTH]; /\* Група товару \*/

float cost; /\* Ціна \*/

char provider[MAX\_NAME\_LENGTH]; /\* Постачальник товару \*/

int date[6]; /\* Дата та час \*/

int isDeleted;

} AuctionProduct;

typedef struct {

char dbFile[MAX\_NAME\_LENGTH];

char key[MAX\_KEY\_LENGTH];

} Config;

// Функція для отримання налаштувань з конфігураційного файлу

Config getConfig() {

Config config = { 0 };;

FILE\* configFile = fopen(CONFIG\_FILE, "r");

if (configFile == NULL) {

// Створення нового конфігураційного файлу, якщо він не існує

configFile = fopen(CONFIG\_FILE, "w");

if (configFile == NULL) {

perror("Failed to create configuration file"); // Не вдалося створити конфігураційний файл

exit(EXIT\_FAILURE);

}

fprintf(configFile, "DB = database.db\n");

fprintf(configFile, "KEY = ABCD12345\n");

fclose(configFile);

// Повторне відкриття щойно створеного файлу

configFile = fopen(CONFIG\_FILE, "r");

if (configFile == NULL) {

perror("Failed to open configuration file"); // Не вдалося відкрити конфігураційний файл

exit(EXIT\_FAILURE);

}

}

char strin[100];

while (fgets(strin, sizeof(strin), configFile)) {

if (strncmp(strin, "DB =", 4) == 0) {

int result = sscanf(strin, "DB = %s", config.dbFile);

if (result != 1) {

// Обробка помилки при розборі рядка

}

}

else if (strncmp(strin, "KEY =", 5) == 0) {

int result = sscanf(strin, "KEY = %s", config.key);

if (result != 1) {

}

}

}

fclose(configFile);

return config;

}

// Функція для додавання запису до бази даних

AuctionProduct\* addRecord(AuctionProduct\*\* products, int\* count) {

AuctionProduct product;

printf("\n");

printf("Enter the name of the product: ");

fgets(product.name, sizeof(product.name), stdin);

product.name[strcspn(product.name, "\n")] = '\0'; // Удаление символа новой строки

printf("Enter the product group: ");

fgets(product.group, sizeof(product.group), stdin);

product.group[strcspn(product.group, "\n")] = '\0'; // Удаление символа новой строки

printf("Enter the price of the product: ");

scanf("%f", &product.cost);

getchar();

printf("Enter the name of the product supplier: ");

fgets(product.provider, sizeof(product.provider), stdin);

product.provider[strcspn(product.provider, "\n")] = '\0'; // Удаление символа новой строки

printf("Enter the date and time of delivery of the goods (YYYY MM DD HH MM): ");

scanf("%d %d %d %d %d", &product.date[0], &product.date[1], &product.date[2], &product.date[3], &product.date[4]);

printf("\n");

getchar();

product.isDeleted = 0;

product.id\_product = \*count;

AuctionProduct\* newProducts = (AuctionProduct\*)realloc(\*products, (\*count + 1) \* sizeof(AuctionProduct));

if (newProducts == NULL) {

perror("Failed to reallocate memory");

exit(EXIT\_FAILURE);

}

else {

\*products = newProducts;

(\*products)[\*count] = product;

(\*count)++;

}

return \*products;

}

// Функція для видалення запису з бази даних

void deleteRecord(AuctionProduct\* products, int\* count, int id\_product) {

if (id\_product < 0 || id\_product >= \*count) {

printf("Invalid record ID\n");

return;

}

if (products[id\_product].isDeleted) {

printf("Record has already been deleted\n");

return;

}

products[id\_product].isDeleted = 1;

printf("Record with ID %d has been deleted\n", id\_product);

printf("\n");

// Зсуваємо товари після віддаленого елемента

for (int i = id\_product + 1; i < \*count; i++) {

products[i - 1] = products[i];

products[i - 1].id\_product = i - 1;

}

(\*count)--;

}

// Функція для перегляду записів бази даних

void viewRecords(AuctionProduct\* products, int count) {

printf("\n");

printf("%-5s|%-20s|%-20s|%-10s|%-20s|%-20s|\n", "ID", "Name", "Group", "Cost", "Provider", "Date");

printf("\n");

for (int i = 0; i < count; i++) {

if (!products[i].isDeleted) {

printf("%-5d|%-20s|%-20s|%-10.2f|%-20s|%.2d/%.2d/%.2d %.2d:%.2d|\n",

products[i].id\_product, products[i].name, products[i].group, products[i].cost, products[i].provider,

products[i].date[0], products[i].date[1], products[i].date[2],

products[i].date[3], products[i].date[4]);

}

}

}

// Функція для зміни запису в базі даних

void modifyRecord(AuctionProduct\* products, int count, int id\_product) {

if (id\_product < 0 || id\_product >= count) {

printf("Invalid record ID\n");

return;

}

if (products[id\_product].isDeleted) {

printf("Record has been deleted\n");

return;

}

AuctionProduct modifiedProduct = products[id\_product];

printf("\n");

printf("Enter new product name (leave blank to keep the same): ");

fgets(modifiedProduct.name, sizeof(modifiedProduct.name), stdin);

modifiedProduct.name[strcspn(modifiedProduct.name, "\n")] = '\0'; // Удаляем символ новой строки

printf("Enter new group name (leave blank to keep the same): ");

fgets(modifiedProduct.group, sizeof(modifiedProduct.group), stdin);

modifiedProduct.group[strcspn(modifiedProduct.group, "\n")] = '\0'; // Удаляем символ новой строки

printf("Enter new product cost (leave blank to keep the same): ");

char costInput[10];

fgets(costInput, sizeof(costInput), stdin);

if (costInput[0] != '\n') {

sscanf(costInput, "%f", &modifiedProduct.cost);

}

printf("Enter new provider name (leave blank to keep the same): ");

fgets(modifiedProduct.provider, sizeof(modifiedProduct.provider), stdin);

modifiedProduct.provider[strcspn(modifiedProduct.provider, "\n")] = '\0'; // Удаляем символ новой строки

printf("Enter new product date and time (YYYY MM DD HH MM, leave blank to keep the same): ");

char dateInput[50];

fgets(dateInput, sizeof(dateInput), stdin);

if (dateInput[0] != '\n') {

sscanf(dateInput, "%d %d %d %d %d",

&modifiedProduct.date[0], &modifiedProduct.date[1], &modifiedProduct.date[2],

&modifiedProduct.date[3], &modifiedProduct.date[4]);

}

// Проверяем, были ли введены новые значения, и сохраняем предыдущие значения при отсутствии ввода

if (modifiedProduct.name[0] == '\0') {

strcpy(modifiedProduct.name, products[id\_product].name);

}

if (modifiedProduct.group[0] == '\0') {

strcpy(modifiedProduct.group, products[id\_product].group);

}

if (costInput[0] == '\n') {

modifiedProduct.cost = products[id\_product].cost;

}

if (modifiedProduct.provider[0] == '\0') {

strcpy(modifiedProduct.provider, products[id\_product].provider);

}

if (dateInput[0] == '\n') {

memcpy(modifiedProduct.date, products[id\_product].date, sizeof(modifiedProduct.date));

}

products[id\_product] = modifiedProduct;

}

// Функция для виведення усіх товарів заданої групи, ціна яких нижче за задану

void printProductsInRange(AuctionProduct\* products, int count, const char\* group, float maxCost) {

printf("\n");

printf("Products within the specified cost range:\n");

printf("%-5s|%-20s|%-20s|%-10s|%-20s|\n", "ID", "Name", "Group", "Cost", "Provider");

for (int i = 0; i < count; i++) {

if (!products[i].isDeleted && strcmp(products[i].group, group) == 0 && products[i].cost < maxCost) {

printf("%-5d|%-20s|%-20s|%-10.2f|%-20s|\n", products[i].id\_product, products[i].name, products[i].group, products[i].cost, products[i].provider);

}

}

}

// Функція для виведення усіх товарів, що постачаються заданим постачальником

void printProductsByProvider(AuctionProduct\* products, int count, const char\* purveyor) {

printf("\n");

printf("Products owned by one provider:\n");

printf("%-5s|%-20s|%-20s|%-10s|%-20s|\n", "ID", "Name", "Group", "Cost", "Date");

for (int i = 0; i < count; i++) {

if (!products[i].isDeleted && strcmp(products[i].provider, purveyor) == 0) {

printf("%-5d|%-20s|%-20s|%-10.2f|%.2d/%.2d/%.2d %.2d:%.2d|\n",

products[i].id\_product, products[i].name, products[i].group, products[i].cost,

products[i].date[0], products[i].date[1], products[i].date[2],

products[i].date[3], products[i].date[4]);

}

}

}

// Функція для збереження бази даних у файлі

void saveDatabase(AuctionProduct\* products, int count) {

char dbFileName[MAX\_NAME\_LENGTH];

printf("\n");

printf("Enter the name of the database file: ");

fgets(dbFileName, sizeof(dbFileName), stdin);

dbFileName[strcspn(dbFileName, "\n")] = '\0'; // Удаление символа новой строки

FILE\* file = fopen(dbFileName, "wb");

if (file == NULL) {

perror("File opening failed");

exit(EXIT\_FAILURE);

}

fwrite(products, sizeof(AuctionProduct), count, file);

fclose(file);

printf("Database saved successfully as %s\n", dbFileName);

printf("\n");

}

// Функція для експорту бази даних у текстовий файл формату CSV

void exportCSV(AuctionProduct\* products, int count, const char\* filename) {

FILE\* file = fopen(filename, "w");

if (file == NULL) {

perror("File opening failed");

exit(EXIT\_FAILURE);

}

printf("\n");

fprintf(file, "ID;Name;Group;Cost;Provider;Date\n");

printf("\n");

for (int i = 0; i < count; i++) {

if (!products[i].isDeleted) {

fprintf(file, "%d;%s;%s;%.2f;%s; %d.%d.%d %02d:%02d\n",

products[i].id\_product, products[i].name, products[i].group, products[i].cost, products[i].provider,

products[i].date[0], products[i].date[1], products[i].date[2],

products[i].date[3], products[i].date[4]);

}

}

fclose(file);

}

void open\_input\_database(AuctionProduct\*\* products, int\* count) {

if (\*products != NULL) {

free(\*products);

\*products = NULL;

\*count = 0;

}

printf("\n");

char db\_file\_name[MAX\_NAME\_LENGTH];

printf("Enter the name of the database file to open: ");

fgets(db\_file\_name, sizeof(db\_file\_name), stdin);

db\_file\_name[strcspn(db\_file\_name, "\n")] = '\0'; // Удаление символа новой строки

FILE\* db\_file;

if (fopen\_s(&db\_file, db\_file\_name, "rb") != 0) {

printf("Error opening the database file.\n");

return;

}

AuctionProduct product;

while (fread(&product, sizeof(AuctionProduct), 1, db\_file)) {

AuctionProduct\* temp = (AuctionProduct\*)realloc(\*products, (\*count + 1) \* sizeof(AuctionProduct));

if (temp == NULL) {

perror("Memory reallocation failed");

fclose(db\_file);

exit(EXIT\_FAILURE);

}

\*products = temp;

(\*products)[\*count] = product;

(\*count)++;

}

fclose(db\_file);

printf("\n");

printf("Database file %s opened successfully.\n", db\_file\_name);

printf("\n");

}

void createDatabaseFile(const char\* filename) {

FILE\* file = fopen(filename, "rb");

if (file == NULL) {

file = fopen(filename, "wb");

if (file == NULL) {

printf("\n");

perror("Database file creation failed");

exit(EXIT\_FAILURE);

}

fclose(file);

printf("\n");

printf("Database file created: %s\n", filename);

}

else {

fclose(file);

}

}

int main() {

Config config = getConfig();

// Запрос ліцензійного ключа у користувача

char licenseKey[MAX\_KEY\_LENGTH];

printf("Enter license key: ");

fgets(licenseKey, sizeof(licenseKey), stdin);

licenseKey[strcspn(licenseKey, "\n")] = '\0'; // Видалення символу нового рядка

// Перевірка ліцензійного ключа

int isValidKey = 0;

if (atoi(licenseKey) % 1024 == 0) {

isValidKey = 1;

}

// Перевірка доступу до функцій

if (isValidKey) {

printf("License key is valid. Access granted to all functions.\n");

printf("\n");

}

else {

printf("Invalid license key. Access restricted.\n");

printf("\n");

}

AuctionProduct\* products = NULL;

int count = 0;

FILE\* file = fopen(config.dbFile, "rb");

if (file != NULL) {

AuctionProduct product;

while (fread(&product, sizeof(AuctionProduct), 1, file)) {

AuctionProduct\* temp = (AuctionProduct\*)realloc(products, (count + 1) \* sizeof(AuctionProduct));

if (temp == NULL) {

perror("Memory reallocation failed");

free(products); // Звільняємо пам'ять, виділену раніше

exit(EXIT\_FAILURE);

}

products = temp;

products[count] = product;

count++;

}

fclose(file);

}

int choice;

do {

printf("\n");

printf("Menu:\n");

printf("1. Add record\n");

printf("2. Delete record\n");

printf("3. View records\n");

printf("4. Modify record\n");

printf("5. Print products within cost range\n");

printf("6. Print products owned by one provider\n");

printf("7. Save database\n");

printf("8. Export database to CSV\n");

printf("9. Input database\n");

printf("0. Exit\n");

printf("\n");

printf("Enter your choice: ");

scanf\_s("%d", &choice);

(void)getchar(); // Читання символу нового рядка після введення вибору

switch (choice) {

case 1:

if (isValidKey) {

products = addRecord(&products, &count);

}

else {

printf("\n");

printf("Access denied. Invalid license key.\n");

printf("\n");

}

break;

case 2:

if (isValidKey) {

int id;

printf("\n");

printf("Enter the ID of the record to delete: ");

scanf\_s("%d", &id);

printf("\n");

deleteRecord(products, &count, id);

}

else {

printf("\n");

printf("Access denied. Invalid license key.\n");

printf("\n");

}

break;

case 3:

viewRecords(products, count);

break;

case 4: {

if (isValidKey) {

int id;

printf("\n");

printf("Enter the ID of the record to modify: ");

scanf\_s("%d", &id);

printf("\n");

(void)getchar(); // Читання символу нового рядка після введення ID

modifyRecord(products, count, id);

break;

}

else {

printf("\n");

printf("Access denied. Invalid license key.\n");

printf("\n");

}

break;

}

case 5: {

if (isValidKey) {

char group[100];

float maxCost;

printf("\n");

printf("Enter the group of product: ");

fgets(group, sizeof(group), stdin);

group[strcspn(group, "\n")] = '\0'; // Удаление символа новой строки

printf("Enter the special cost: ");

scanf("%f", &maxCost);

printf("\n");

getchar(); // Читання нового рядка після введення вартості

printProductsInRange(products, count, group, maxCost);

}

else {

printf("\n");

printf("Access denied. Invalid license key.\n");

printf("\n");

}

break;

}

case 6: {

if (isValidKey) {

char purveyor[50];

printf("\n");

printf("Enter the provider: ");

scanf(" %[^\n]", purveyor);

printf("\n");

(void)getchar(); // Читання символу нового рядка після введення дати

printProductsByProvider(products, count, purveyor);

break;

}

else {

printf("\n");

printf("Access denied. Invalid license key.\n");

printf("\n");

}

break;

}

case 7:

{if (isValidKey) {

saveDatabase(products, count);

break;

}

else {

printf("\n");

printf("Access denied. Invalid license key.\n");

printf("\n");

}

break;

}

case 8: if (isValidKey) {

exportCSV(products, count, "database.csv");

printf("\n");

printf("Database exported to 'database.csv'\n");

printf("\n");

}

else {

printf("\n");

printf("Access denied. Invalid license key.\n");

printf("\n");

}

break;

case 9:

open\_input\_database(&products, &count);

break;

case 0:

printf("\n");

printf("Goodbye!\n");

break;

default:

printf("\n");

printf("Invalid choice\n");

break;

}

} while (choice != 0);

free(products);

return EXIT\_SUCCESS;

}

# ДОДАТОК Б. СЕРТИФІКАТ ПРО ПРОХОДЖЕННЯ НАВЧАЛЬНОГО КУРСУ



Рисунок Б.1 − Скриншот сертифікату с платформи Sololearn

Посилання на сертифікат: <https://www.sololearn.com/certificates/CC-VED10SQI>