МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра Інженерії програмного забезпечення

**Звіт**

**з навчальної практики**

студента І курсу групи ВТ-21-1

спеціальності 121 «Інженерія програмного забезпечення.

Бабушка Андрія Сергійовича

(прізвище, ім’я та по-батькові)

Керівник ст. викладач кафедри КН

, Фуріхата Д. В. ,

Дата захисту: " \_\_\_ " \_\_\_\_\_\_\_\_\_\_\_\_ 2022 р.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

Житомир – 2022

**Зміст**

[**ВСТУП** 3](#_Toc93861870)

[**РОЗДІЛ 1. АНАЛІЗ ЗАДАЧІ** 5](#_Toc93861871)

[**РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ** 7](#_Toc93861872)

[**РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ** 17](#_Toc93861873)

[**ВИСНОВКИ** 25](#_Toc93861874)

[**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ** 26](#_Toc93861875)

[**ДОДАТКИ** 28](#_Toc93861876)

# **ВСТУП**

***Актуальність:*** «Сі» – мова системного програмування, що дозволяє виконувати стандартні високорівневі підпрограми.

Мова «Сі» являє собою вдалий компроміс між бажанням мати у своєму розпорядженні ті можливості, які зазвичай надають програмісту настільки зрозумілі та зручні мови високого рівня, і прагненням ефективно використовувати особливості комп'ютера. Крім набору засобів, властивих сучасним мовам програмування високого рівня: структурність, модульність, типи даних, що визначаються. До нього включені засоби для програмування «майже» на рівні асемблера: використання покажчиків, побітові операції, операції зсуву. Великий набір операторів дозволяє писати компактні та ефективні програми.

Декілька фактів про мову програмування «Сі»:

* «Сі» був створений Денісом Рітчі, як інструмент для розробки операційної системи «UNIX» та реалізований у рамках цієї операційної системи. Назва мови має випадкове походження: «С» - третя буква англійського алфавіту. Це найменування свідчить про почуття гумору у творців мови - його попередником був мова «B» («B» - друга літера англійського алфавіту).
* В даний час є велика кількість систем програмування на «Сі» для різних типів комп'ютерів. Розроблено багато бібліотек модулів, інструментальних засобів розробки та налагодження, що полегшують створення нових програм. Програми на «Сі» мають високу мобільність, без будь-яких змін вони переносяться, транслюються і виконуються на машинах різних типів.
* Мова «Сі» - компілюючого типу. Текст програми на «Сі», підготовлений за допомогою текстового редактора, для отримання об'єктного модуля обробляється компілятором, причому компіляція виконується у два проходи. При першому проході обробляються рядки-директиви, що починаються зі знака «#», при другому - транслюється текст програми та створюється машинний код. Для отримання виконуваного модуля необхідно відредагувати зовнішні зв'язки об'єктного модуля - приєднати до нього відповідні бібліотечні модулі.

Актуальність принципів роботи з масивами структур, створенню файлів, запису та читання файлів допоможуть у виконанні тяжчих робіт і при роботі з базами даних. Також важливим моментом є розвиток логічного мислення при написанні програми і, звичайно, побудова плану роботи.

***Предмет дослідження:*** робота та створення програм з використанням файлів, як масивів даних.

***Об’єкт дослідження:*** технологія роботи з файлами та базами даних.

***Мета роботи:*** вивчення та закріплення знать роботи з файлами та масивами структур.

# **РОЗДІЛ 1. АНАЛІЗ ЗАДАЧІ**

Перед тим, як почати писати код, потрібно зробити аналіз та визначити порядок виконання завдання.

Задача на практичну роботу полягає у створенні програмного додатку, що реалізує обробку даних. Обов’язкові умови, або функції, які має містити додаток наступні:

* Введення даних з клавіатури.
* Дані мають зберігатися у вигляді файлів (бінарних, текстових).
* Виведення даних на екран як всіх так і за параметрами (кількість параметрів -3).
* Пошук даних за параметрами (кількість параметрів - 3).
* Редагування даних. Перед редагуванням знайти потрібний запис.
* Видалення даних. Перед видаленням знайти потрібний запис.
* Впорядкування записів за зростанням або спаданням;
* Програма має містити за необхідності декілька рівнів меню та здійснювати перехід між різними рівнями меню.

Для розробки проекту було видано 1-ший варіант: «Особиста бібліотека». У роботі потрібно реалізувати 2-ві структури: «Дані книги» та «Розділ бібліотеки». Головна структура є «Розділ бібліотеки», в свою чергу вкладена структура це «Дані книги». В структуру «Дані книги» входять: автор, назва, видавництво, рік створення, а в структуру «Розділ бібліотеки» входять: жанр, походження, наявність, особиста оцінка книги. Всі змінні представляють рядки. Також, у програмі потрібно реалізувати специфічну функцію для 1-ого варіанту під назвою «Інвентаризація». Вигляд структур подано наступним чином:

struct library\_book\_information {

char authors[50];

char name[50];

char edition[50];

char year[50];

} \*book = NULL;

struct personal\_library {

struct library\_book\_information book;

char section[50];

char book\_origin[50];

char availability\_of\_a\_book[50];

char book\_evaluation[50];

} \*library = NULL;

Також було ініціалізовано назви функцій глобально та об’явлено їх як «NULL» для подальшого динамічного виділення пам’яті.

Наступним кроком після аналізу потрібних типів даних для проекту буде проектування, в якому створено функції для запису та виводу таблицею структур у консоль, виконана робота з файлом(читання/запис), написання відповідних функцій, як пошук за 3 параметрами, редагування, видалення, сортування та інвентаризація, а також логічний опис їх роботи.

# **РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

Для початку визначимо означення. Проектування програмного забезпечення (ПЗ) – процес визначення архітектури, компонентів, інтерфейсів, інших характеристик системи й кінцевого результату. Проектування ПЗ як інженерний процес входить зараз як складова дисципліна загального інженерного процесу розроблення програм – програмної інженерії.

Еталонну модель програмної інженерії можна визначити як взаємодію трьох факторів:

* процесів;
* продуктів;
* ресурсів.

Кожна програмна система протягом свого існування проходить з певною послідовністю фази або стадії від задуму до його втілення в програми, експлуатацію та вилучення. Така послідовність фаз називається життєвим циклом розробки. На кожній фазі відбувається певна сукупність процесів, кожен з яких породжує певний продукт, використовуючи певні ресурси. Усі продукти всіх процесів програмної інженерії являють собою певні описи — тексти вимог до розробки, погодження домовленостей, документацію, тексти програм, інструкції з експлуатації тощо. Головними ресурсами програмної інженерії, які визначають ефективність її розробок, є час і вартість цих розробок.

Отже, повернемось до проектування нашого проекту. Для початку, побудуємо структуру роботи програми.

Користувач, після запуску програми, бачить у консолі перед собою меню, яке буде складатись із заголовків: «Введення даних», «Виведення даних», «Пошук», «Редагування», «Видалення», «Впорядкування» (сортування), «Інвентаризація бібліотеки», «Вихід з програми». Це все буде містити в собі функція меню. Відповідно з натисканням потрібної клавіши на клавіатурі буде виконана потрібна користувачеві функція.

Ось показана показано роботу функції (меню):

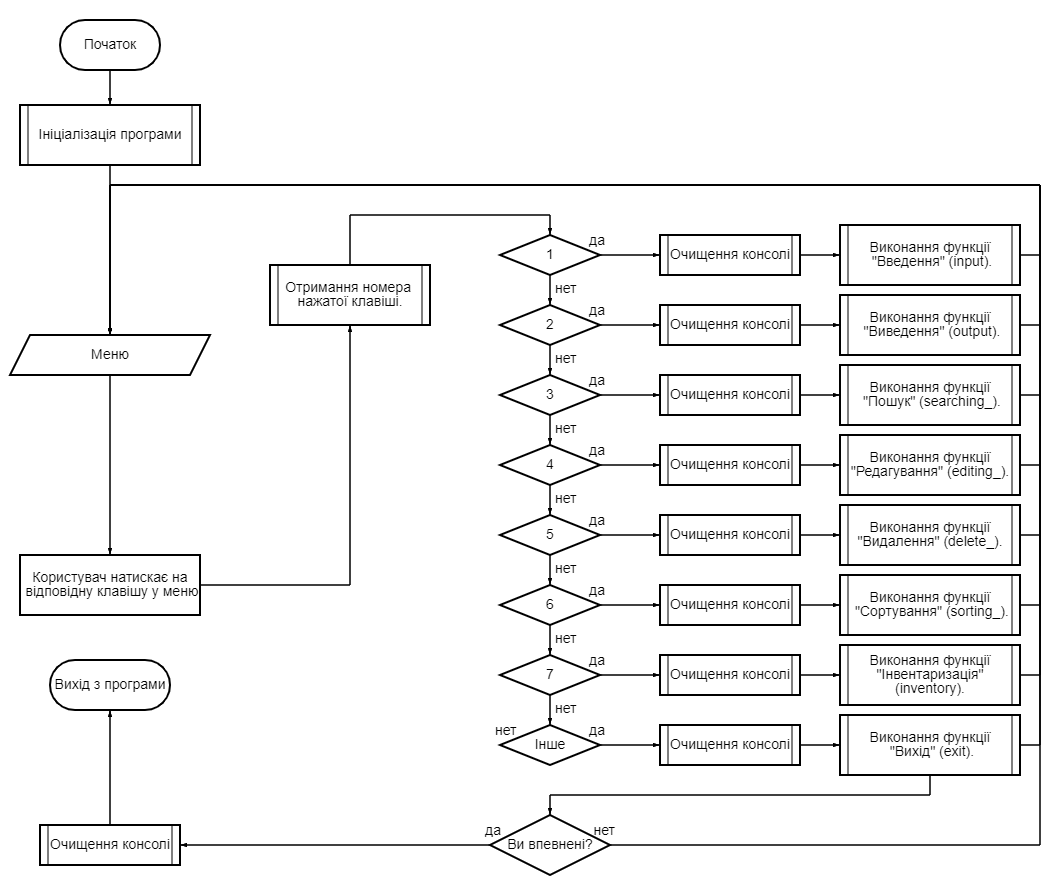


Рис. 2.1. Структура роботи програми(меню)

Після остаточного визначення роботи меню, одразу розпочато роботу над наступною функцією «exit».

Її структура полягає у тому, що ми питаємо у користувача чи впевнений він, що хоче саме вийти з програми, бо бувають такі моменти, що користувач може випадково натиснути не на потрібну клавішу.

Щоб надалі не виникало питань, надано лістинг функції «main»:

SetConsoleCP(1251); SetConsoleOutputCP(1251);

binary = fopen("binary.bin", "ab"); fclose(binary);

remove("output.txt");

outputfile = fopen("output.txt", "a"); fclose(outputfile);

library = (struct personal\_library\*)calloc(sizeof(struct personal\_library), 15);

book = (struct library\_book\_information\*)calloc(sizeof(struct library\_book\_information), 15);

menu();

Було створено 2 файли: бінарний та текстовий. Бінарний слугує сховищем для даних, а текстовий лише архівування виведених даних у консоль. Також було виділено пам’ять під структури.

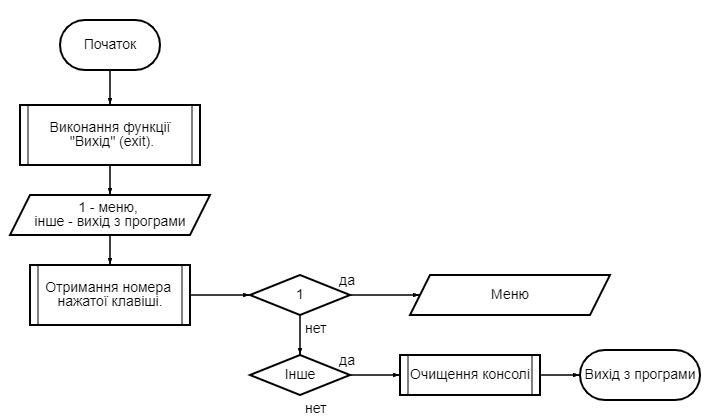
Повернемось до функції «exit». Ось її блок-схема:

Рис. 2.2. Структура роботи функції «exit»

Лістинг функції «exit» наступний:

char m;

while (!\_kbhit) {}

printf("\nВибір:\n| 1 - повернення до меню | інше - вихід з програми | --> "); m = \_getche();

switch (m) {

case 49:

system("cls");

menu();

break;

default:

system("cls");

return 1;

break;

}

Перше завдання виконано. Далі було розпочато роботу послідовно над кожним номером меню.

Наступна функція це «Введення» - окрема функція для запису інформації у файл. Сутність роботи функції «Введення» полягає у зчитуванні даних з клавіатури. Спочатку відбувається запис даних у відповідні структури, а потім виклик функції для запису даних з структур у файл. Було вирішено використовувати для запису даних бінарний файл. Ось наглядний рисунок роботи функції введення:

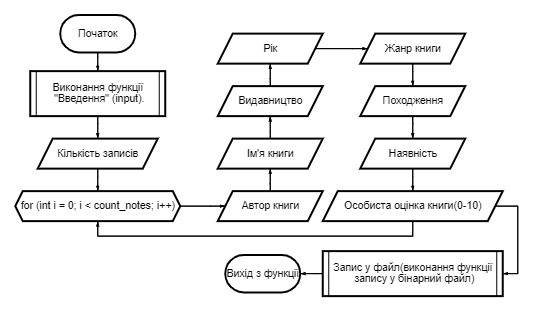


Рис. 2.3. Структура роботи функції «Введення»

В додаток до функції «Введення» приведено лістинг запису у файл

Лістинг функції:

FILE\* binary = fopen("binary.bin", "wb");

if (binary == NULL) {

printf("Error opening file");

\_getch();

exit(-3);

}

fwrite(&count\_notes, sizeof(int), 1, binary);

fwrite(&library, sizeof(personal\_library), count\_notes, binary);

fclose(binary);

Сутність роботи коду функції полягає у відкритті файлу на читання(створення функції було здійснено у «мейні»). Перевірка слугує знаком, що файл не було відкрито для запису. Через «fwrite» ми записуємо адреси змінних, які є в структурах.

Далі буде здійснено опис наступних функцій: «Виведення» та читання записів з бінарного файлу. Сутність роботи цих функцій в наступному.

Вона приймає запит на виведення чи введення, бо інколи програма може не мати в собі ніяких даних для виведення. Після вирішення користувачем вводити чи виводити він обирає між повним виведенням, за параметрами чи виходом з програми. Далі відбувається певна функція яка відповідає за той чи інший вивід інформації у табличку. Також для читання даних структур потрібно виконати читання інформації з бінарного файлу. Тому надано лістинг функції читання:

FILE\* binary = fopen("binary.bin", "ab");

if (binary == NULL) {

printf("Error opening file");

\_getch();

exit(-3);

}

fread(&count\_notes, sizeof(int), 1, binary);

fread(&library, sizeof(personal\_library), 1, binary);

Читання файлу працює майже так само, як і запис, але використовується функція «fread».

Приведемо блок-схему роботи функції «Виведення»:

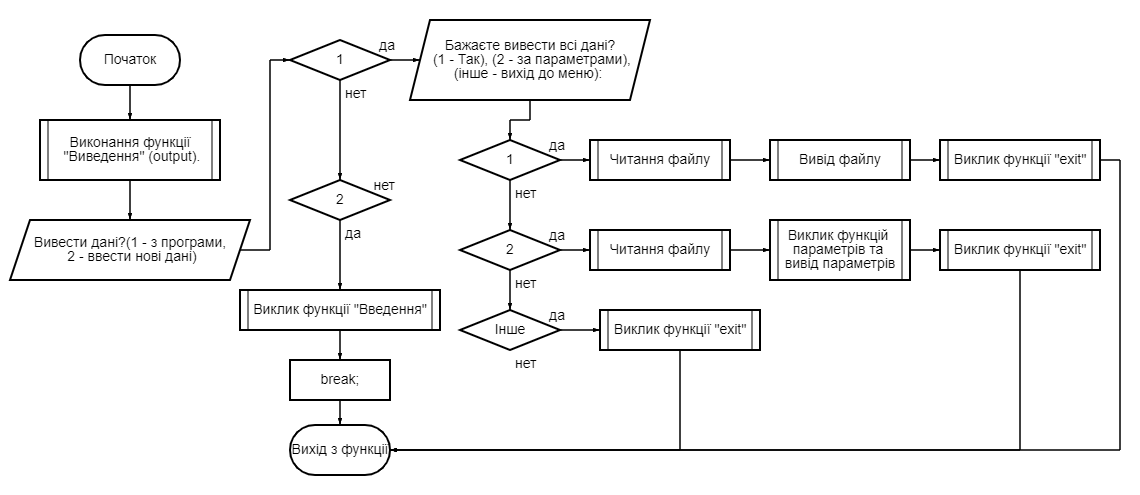


Рис. 2.4. Структура роботи функції «Виведення»

1 етап роботи виконано – це організація функції вводу/виводу інформації та підключення до їх роботи читання/запис структур з файлу.

Наступним етапом буде написання остаточних функцій, які дозволять користувачеві шукати, редагувати, видаляти, сортувати інформацію та проводити інвентаризацію наявних чи ненаявних книг у бібліотеці. Першою функцією буде «Пошук». Опишемо роботу цієї функції у вигляді блок-схеми:

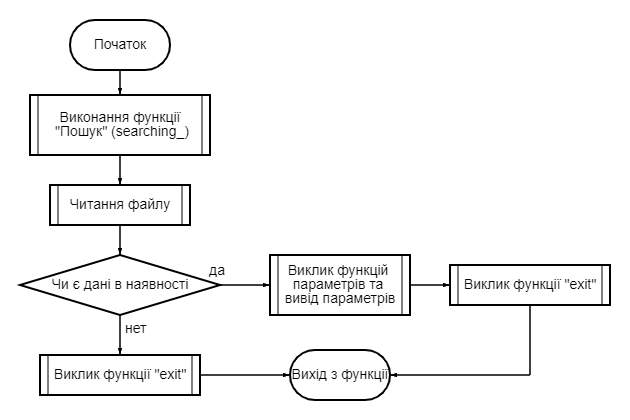


Рис. 2.5. Структура роботи функції «Пошук»

Для початку потрібно провести зчитування даних з відповідного файлу у системі. Далі йде перевірка на наявність даних у системі, чи були вони у файлі. Далі у заключному результаті цієї перевірки ми здійснюємо чи виклик функції параметрів для конкретного вибору користувачем запису за параметром та подальший його вивід, чи вихід з функції у меню або з програми.

Також для наглядного прикладу наведено лістинг цієї функції:

readBinary();

printf("\nЗапит на пошук.\n");

if (count\_notes == 0) {

printf("В базі даних 0 записів.\n");

exit();

}

parameters\_();

for (int b = 0; b < count\_parameters; b++) {

output\_parameters(b);

}

fclose(binary);

exit();

Наступна функція має назву «Редагування». Давайте зробимо опис функції за допомогою блок-схеми:

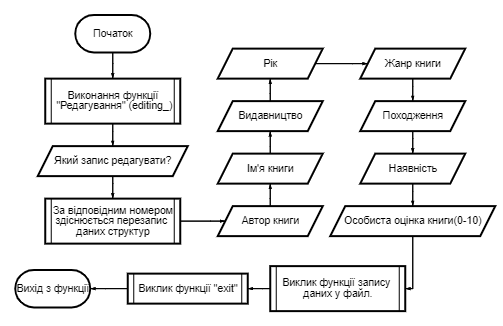


Рис. 2.6. Структура роботи функції «Редагування»

Працює функція так: для початку програма питає у користувача який саме запис редагувати за його номером. Програма визначає номер та виводить у консоль відповідні дані структур для подальшого запису їх користувачем. Після всіх операцій, програма записує всі дані у файл та виходить у головне меню.

Лістинг «Редагування»:

printf("\nЗапит на редагування.");

if (count\_notes == 0) {

printf("В базі даних 0 записів.\n");

exit();

}

int i;

while(!\_kbhit){}

printf("\nЯкий запис редагувати? ---> "); scanf("%d", &i); i = i - 1;

printf("\tЗапис %d:\n", i + 1);

функція введення даних… scanf("%s", library[i].book\_evaluation); fwrite(library[i].book\_evaluation, sizeof(char), 50, binary);

writeBinary();

exit();

Вже друга функція виконана. Вирушаємо до наступної функції – «Видалення». Її блок-схема подана наступним чином:

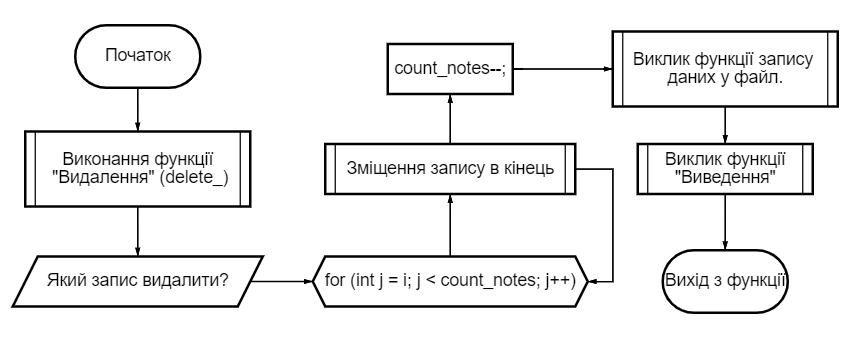


Рис. 2.7. Структура роботи функції «Видалення».

Також додамо її лістинг для наглядності:

readBinary();

printf("\nЗапит на видалення.\n");

if (count\_notes == 0) {

printf("В базі даних 0 записів.\n");

exit();

}

int i;

while (!\_kbhit) {}

printf("\nЯкий запис видалити? ---> "); scanf("%d", &i); i = i - 1;

for (int j = i; j < count\_notes; j++) {

library[j] = library[j + 1];

book[j] = book[j + 1];

}

count\_notes--; writeBinary();

fclose(binary); output();

Робота функції така: проходить читання даних з файлу, далі програма питає у користувача який номер із існуючих записів видалити і виконує цю дію за допомогою зсуву потрібного запису структури в її кінець. Останній крок це зменшити кількість існуючих записів на 1, щоб виводилось без видаленого.

Ще 1 функція позаду. Йдемо далі. Функція «Сортування» працює на основі порівняння потрібного запису за параметром з іншими. Сутність її роботи полягає в наступному. Спочатку ми зчитуємо дані з файлу, затим питаємо у користувача за яким саме записом потрібно сортувати і за яким способом сортувати. Наприклад, 1 вибір користувача за автором книги, наступний за зростанням. Будуть прочитані всі дані, оцінено довжину рядків та виведено за вибраним параметром: зростання чи спадання. Для збереження даних буде здійснено запис даних у файл у відповідній функції «Виведення».

Ось її блок-схема:

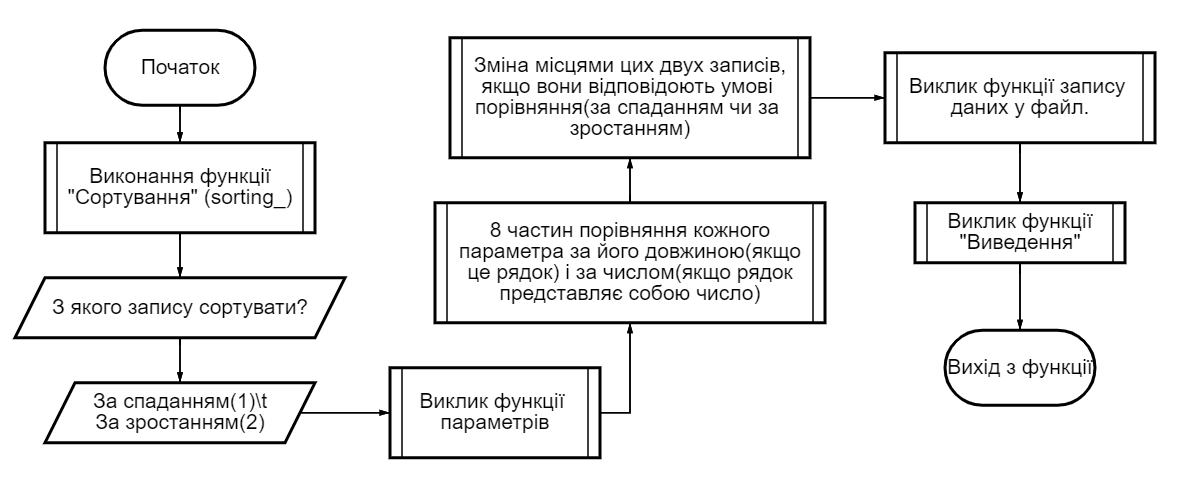


Рис 2.8. Структура роботи функції «Сортування»

Залишилась остання функція програми, це «Інвентаризація». Сутність роботи полягає у знаходженні записів за наявністю товару, чи його відсутністю. У коді це є пошук книг за записом «в наявності» чи «не має в наявності». Її блок-схема подана наступним чином:

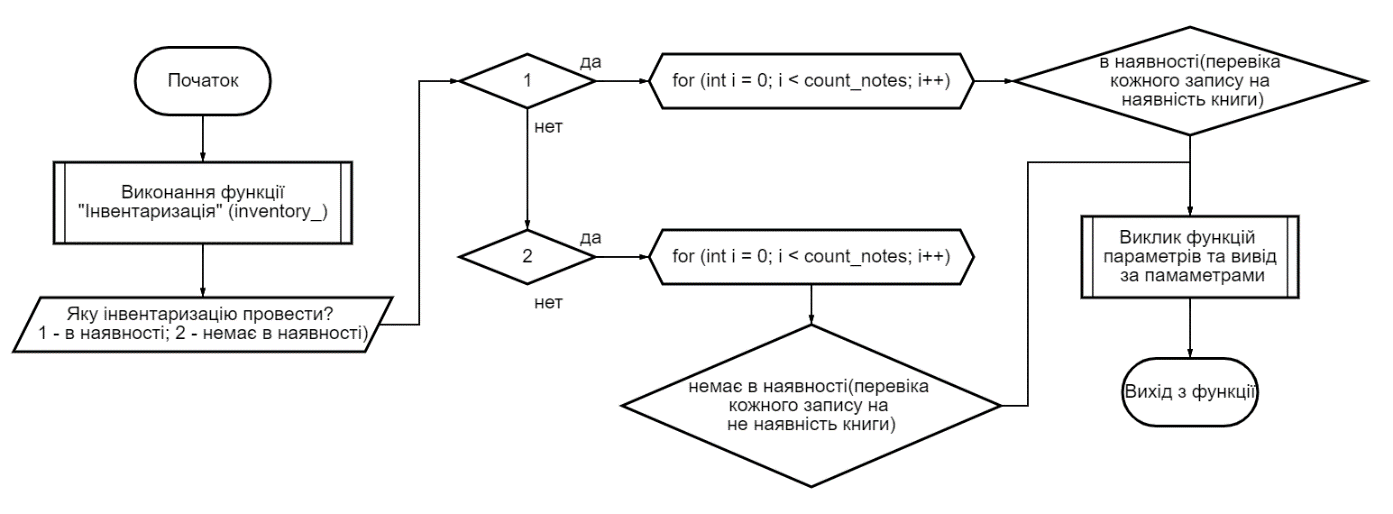


Рис. 2.9. Структура роботи функції «Інвентаризація»

Лістинг функції «Інвентаризація»:

printf("\nЗапит на інвентаризацію.\n");

if (count\_notes == 0) {

printf("В базі даних 0 записів.\n");

exit();

}

readBinary(); int inventory\_book;

printf("\nЯку інвентаризацію провести? (1 - в наявності; 2 - немає в наявності)---> "); scanf("%d", &inventory\_book);

switch (inventory\_book) {

case 1:

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

if (strcmp("Есть", library[i].availability\_of\_a\_book) == 0) {

output\_header();

output\_parameters2\_0(i);

}

break;

case 2:

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

if (strcmp("Нет", library[i].availability\_of\_a\_book) == 0) {

output\_header();

output\_parameters2\_0(i);

}

break;

}

exit();

Принцип роботи функції такий: для початку її роботи потрібно прочитати всі існуючі дані, далі ми запитуємо у користувача яку саме інвентаризацію провести: за наявністю чи ні. Далі відбираємо за номером потрібну частину коду та виводимо всі записи, які мають ту чи іншу оцінку наявності товару.

# **РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ**

3.1. Опис роботи з програмним дотатком

Початок роботи програми починається з її запуску та входу у меню.

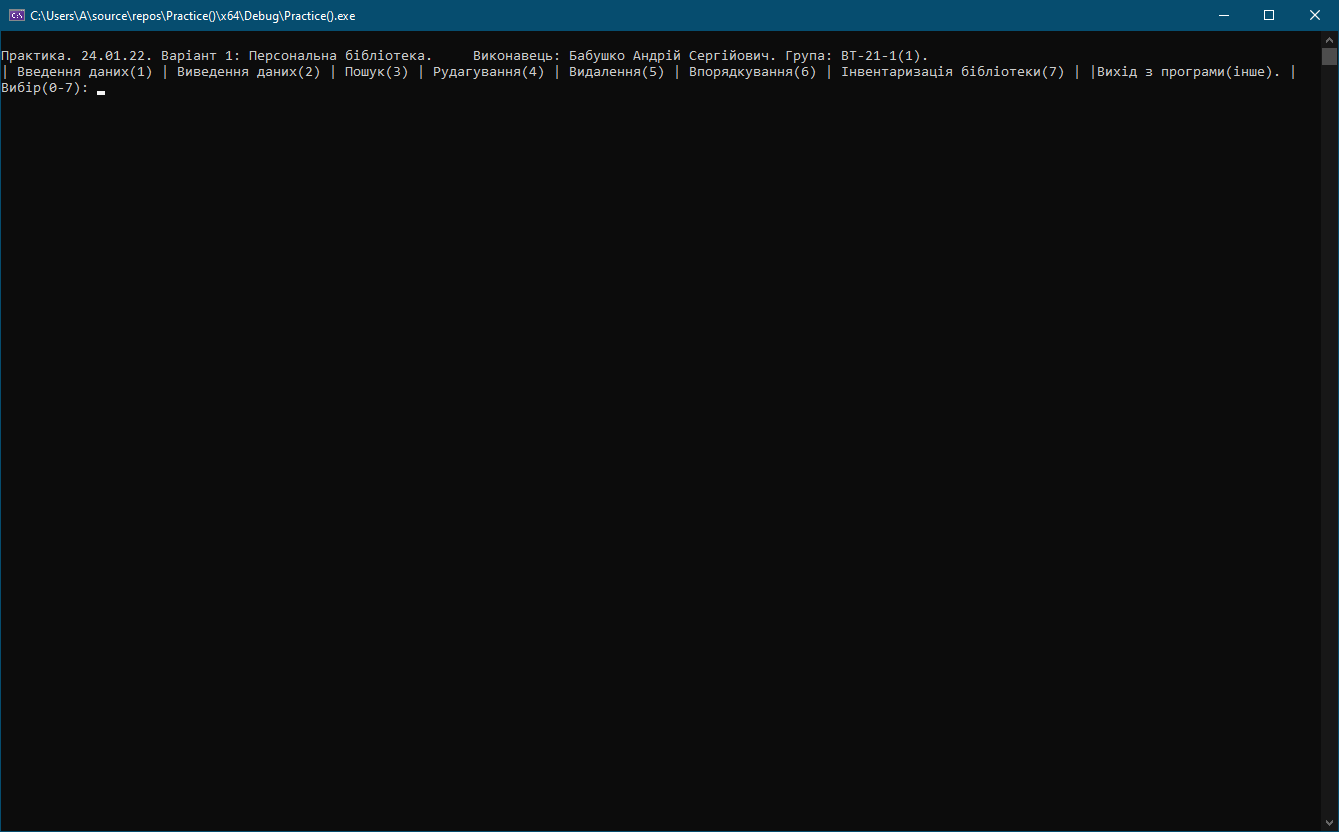
Ось такий вигляд має меню програми:

Рис. 3.1. Вигляд за реалізація меню у консолі

При запуску додатка відбувається створення бінарного файлу, якщо його не існує, заходження у функцію меню і виведення відповідної інформації на екран. У меню користувач, за допомогою клавіатури, у відповідне поле може вписати необхідну йому цифру для виконання певної функції, номер яких записаний у «()» після назви функції. Якщо користувач натисне будь-яку клавішу окрім 1-7, тоді буде запропоновано вихід з програми, чи повернення до меню додатка.

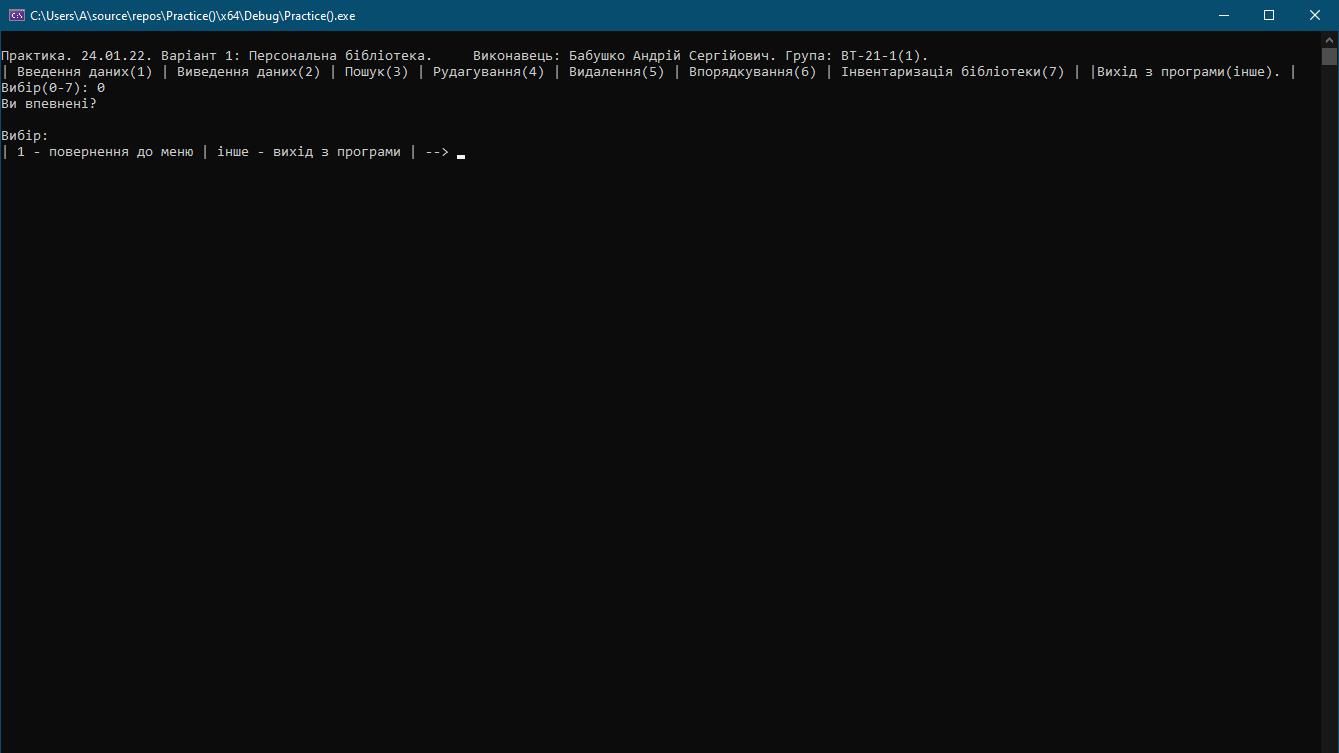
Меню після натискання будь-якої клавіші, окрім 1-7: 

Рис. 3.2. Виконання у консолі функції «Вихід»

3.2. Тестування додатку

Давайте пояснимо, що собою представляє поняття «Тестування ПЗ».

Тестування програмного забезпечення — це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому його мають використовувати. Техніка тестування також включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових із метою оцінки. Тестування програмного забезпечення — процес перевірки відповідності заявлених до продукту вимог і реально реалізованої функціональності, який здійснюють шляхом спостереження за його роботою в штучно створених ситуаціях і на обмеженому наборі тестів, обраних певним чином. Можуть оцінювати: відповідність вимогам, якими керувалися проектувальники та розробники, правильність відповіді для всіх можливих вхідних даних, виконання функцій за прийнятний час, практичність, сумісність із програмним забезпеченням та операційними системами, відповідність задачам замовника. Оскільки число можливих тестів навіть для нескладних програмних компонентів практично нескінченне, тому стратегія тестування полягає в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів. Як результат програмне забезпечення (ПЗ) тестують стандартним виконанням програми з метою виявлення багів (помилок або інших дефектів). Тестування ПЗ може надавати об'єктивну, незалежну інформацію про якість ПЗ, ризики відмови, як для користувачів, так і для замовників. Тестування можна проводити, як тільки створено виконуваний код (навіть частково завершений). Процес розробки зазвичай передбачає, коли та як буде відбуватися тестування. Наприклад, при поетапному процесі більшість тестів відбувається після визначення системних вимог і тоді вони реалізуються в тестових програмах. На противагу цьому, відповідно до вимог гнучкої розробки ПЗ, програмування і тестування часто відбувається одночасно.

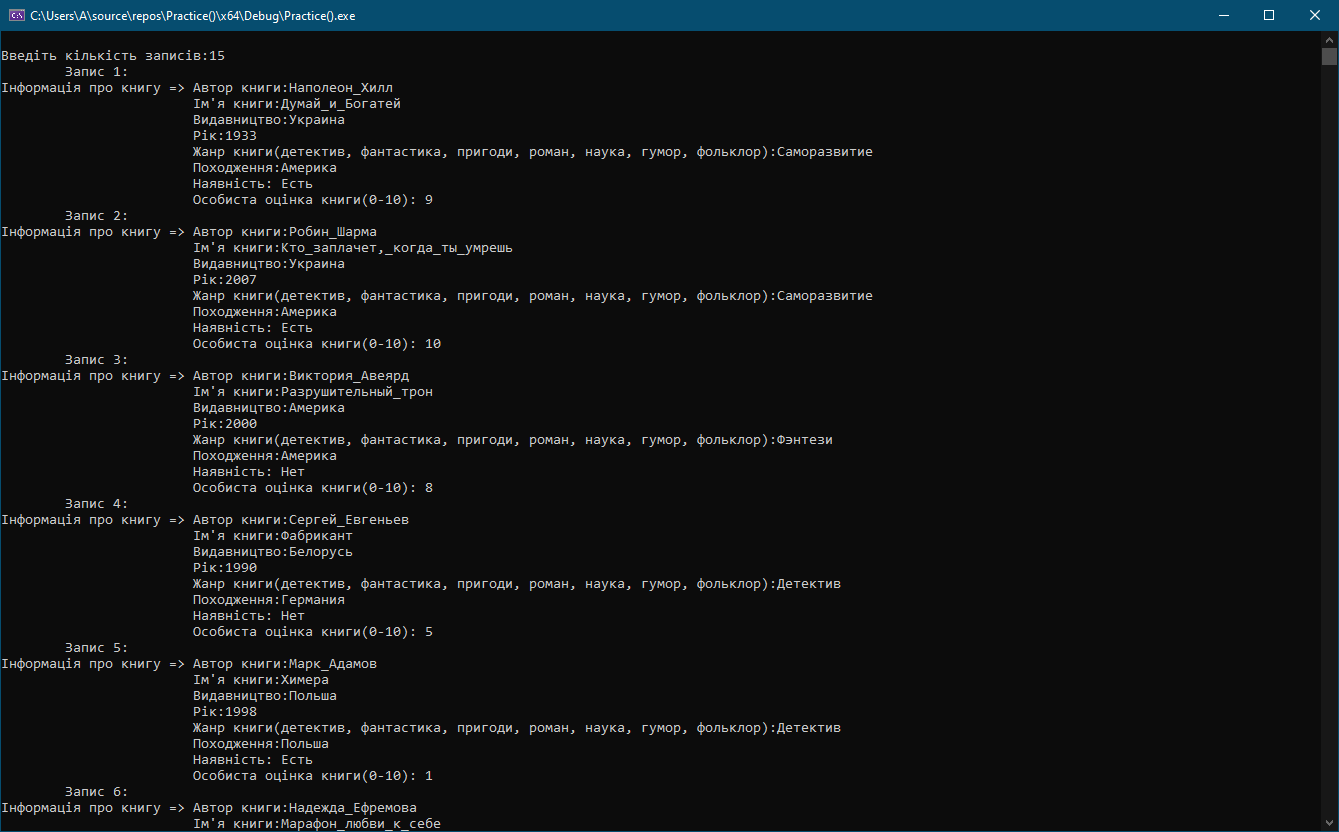
Отже, перша функція додатку, яку побачить користувач буде «Введення даних(1). Вигляд цього підменю наступний: 

Рис. 3.3. Вигляд функції додатка «Введення» у консолі

У першому полі користувач має вписати кількість загальних записів, які він хоче додати до бібліотеки. Далі у полі «Інформація про книгу» потрібно записати послідовно усі дані про книгу: її автора, назва книги, видавництво, рік и т. д.. Відповідно до кількості записів програма пропонує таку ж кількість записів ввести. Всі дані записуються у файл після остаточного введення всіх записів користувачем. Після закінчення введення потрібних даних, консоль пропонує повернутися до меню чи вийти з програми. Далі вона очищається та повертається до головного меню.

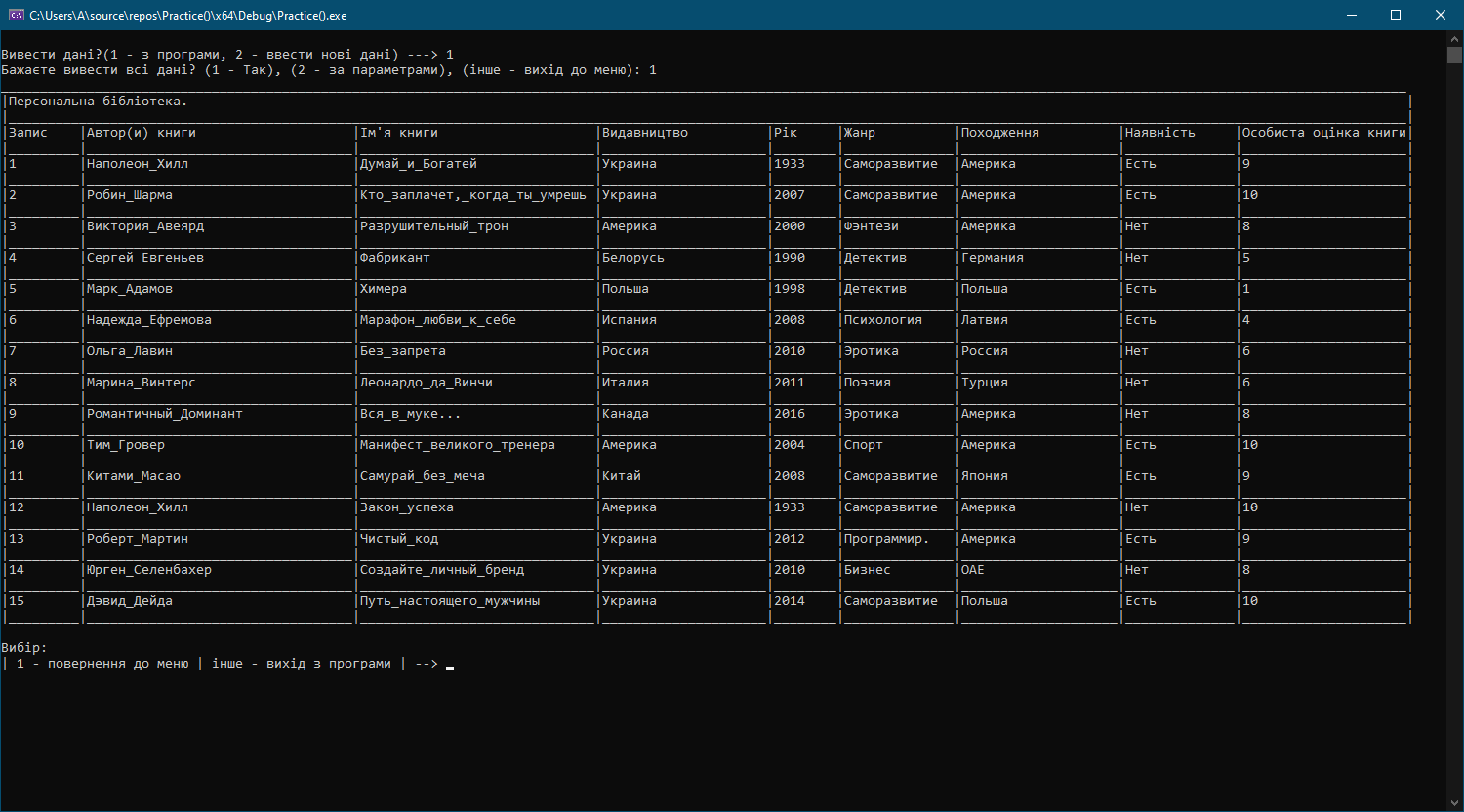
Введення користувач здійснив. Тепер потрібно побачити всі дані, які він записав у базу даних. Для цього йому потрібно виконати функцію додатку під назвою «Виведення даних(2)». Під час виконання функції буде запропоновано виведення даних, чи введення нових даних, бо інколи даних у програмі може не бути. Після цього потрібно вибрати як саме виводи всі дані: всі, чи за параметрами. Також є варіант повернення до меню, бо є випадки раптового натискання не потрібної клавіші. При виведенні повної інформації консоль буде виглядати так: 

Рис. 3.4. Вигляд консолі після виведення повної інформації

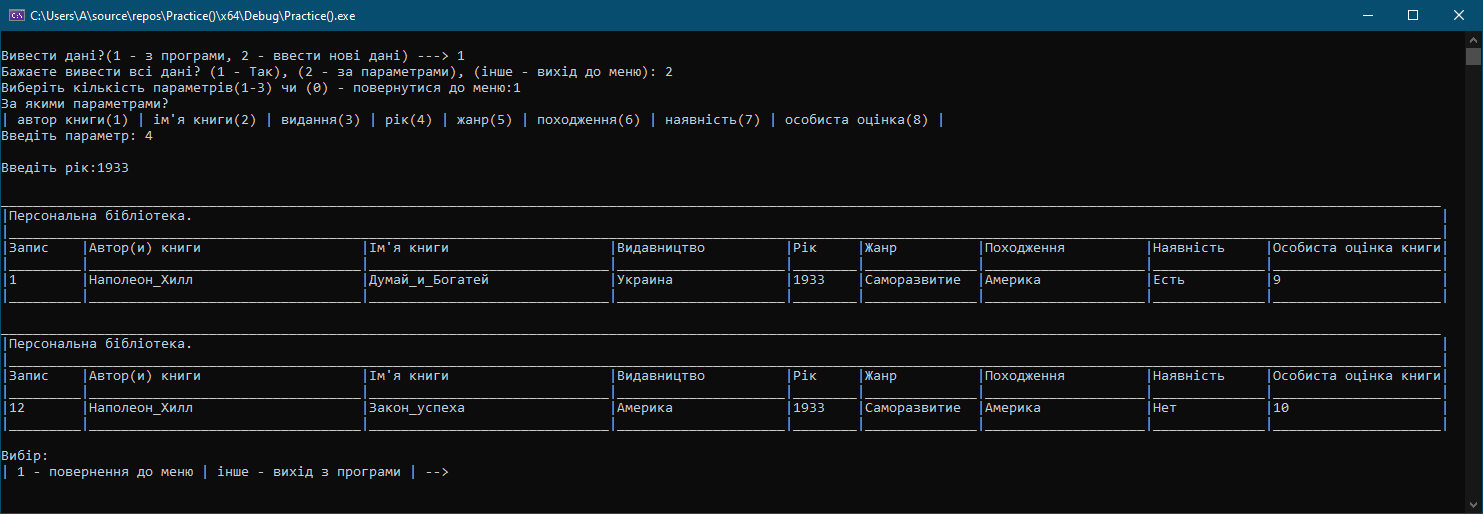
Якщо ж користувач захоче виводити тільки ті записи, які йому потрібні, тоді йому потрібно у відповідні поля, такі як «Вивести всі дані?» і «Бажаєте вивести всі дані?» вписати потрібні номера цифр після чого програма виконає виведення за параметрами. Так виглядає додаток після виконання цих дій: 

Рис. 3.5. Вигляд консолі після виведення інформації за параметрами

Також додамо, що користувач може вибрати кількість параметрів, за якими він хоче здійснити виведення. Наприклад, можна вибрати за роком, оцінкою та за наявністю(послідовність параметрів не важлива). Наприклад, можна вибрати за роком, оцінкою та за наявністю(послідовність параметрів не важлива). Взагалі, кількість параметрів до 8, але за проектом визначено кількість параметрів 3.

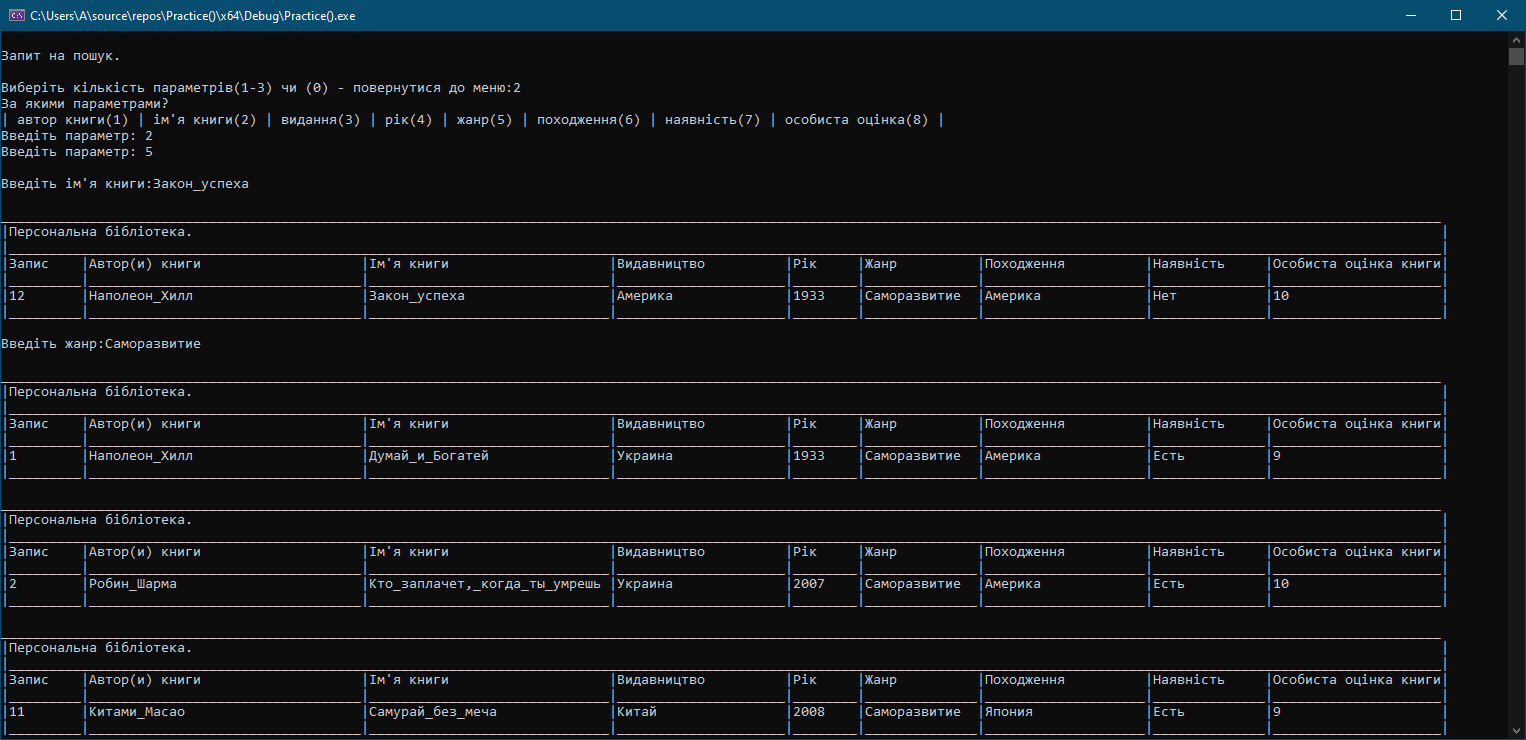
Прямуємо далі. Наступна функція додатку має назву «Пошук(3)». Ось наочний приклад вигляду цієї функції: 

Рис. 3.6. Вигляд консолі після виконання функції «Пошук»

Тут користувачу потрібно записати кількість параметрів, номери параметрів та самі параметри за якими буде відбуватись подальший пошук потрібних записів у базі даних. Ця функція працює майже так само, як і виведення за параметрами. Єдина відмінність це те, що ця функція описана окремо від виведення за параметрами. Також після виведення записів буде запропоновано повернення до меню чи вихід з програми.

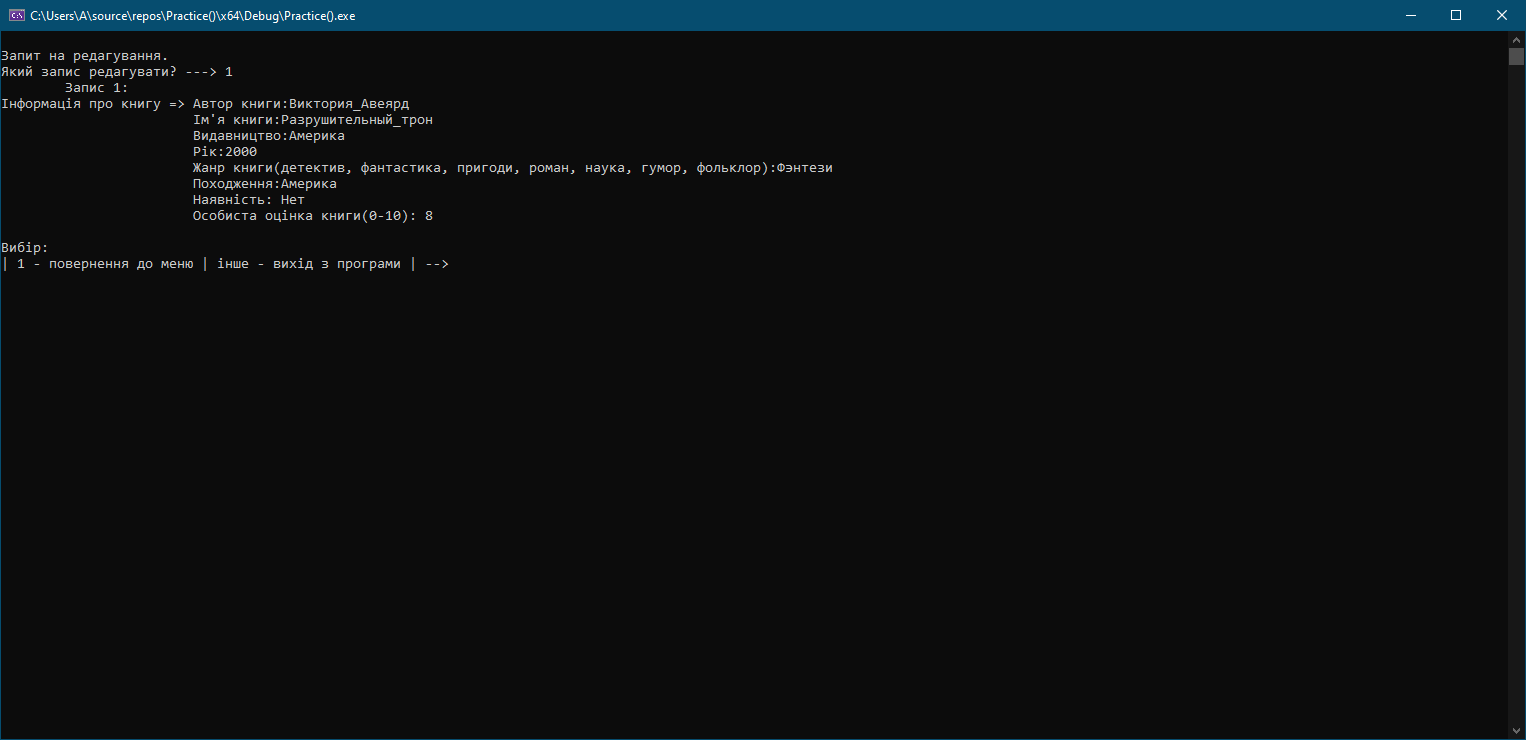
Маємо надію, що все було зрозуміло. Йдемо далі. Наступна функція це «Редагування(4)». Для початку її роботи потрібно вибрати номер запису який потрібно редагувати. Єдиний недолік цього способу редагування це те, що користувачеві потрібно вводити потрібний йому запис повністю. У функції не має можливості вибору конкретної колонки яку потрібно редагувати. Наприклад, запис правильний, але автор книги був записаний неправильно, чи випадково була натиснула зайва літера. Але загалом, ця функція працює правильно. Вигляд консолі наступний: 

Рис. 3.7. Вигляд консолі після виконання функції «Редагування»

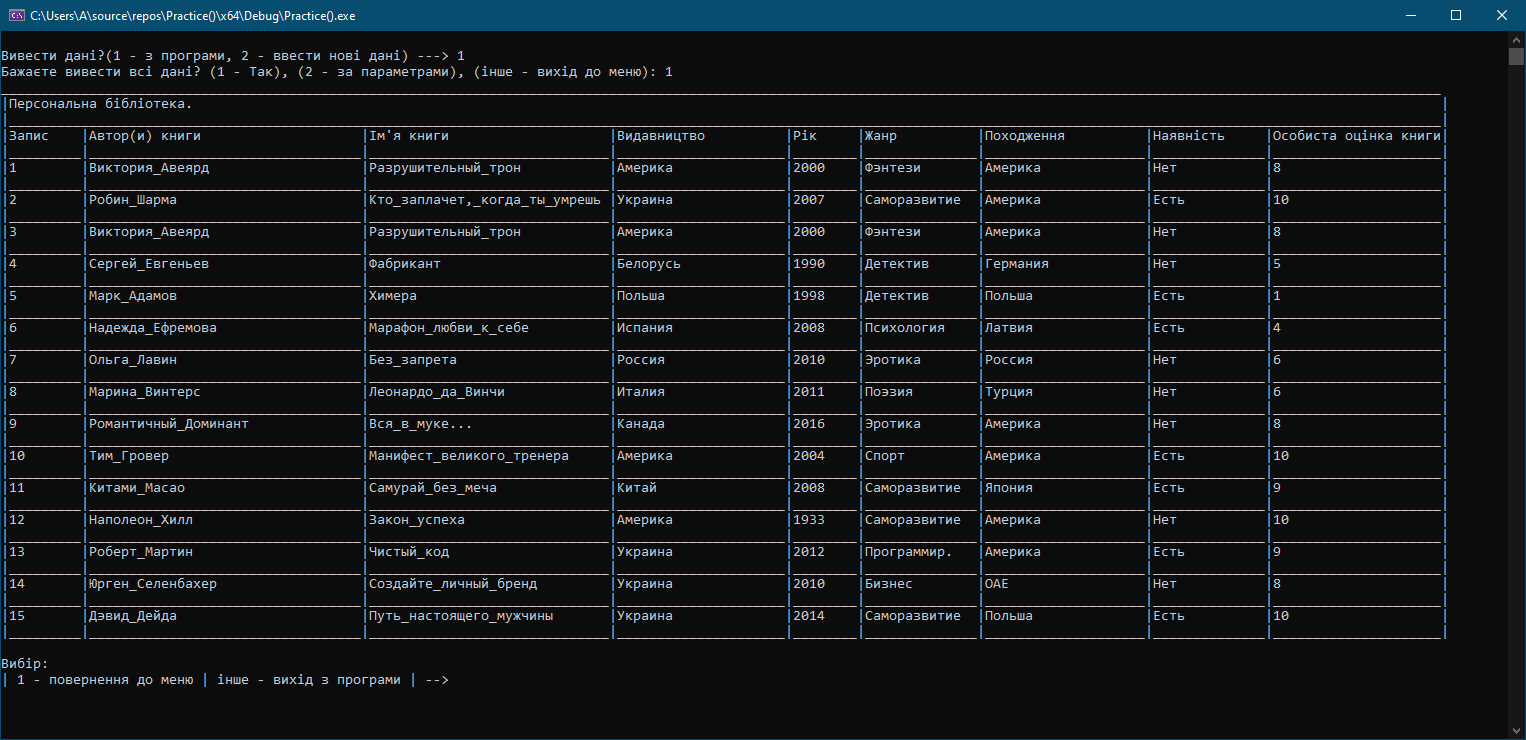
Щоб переконатися, що ця функція працює, ось виведення після її виконання: 

Рис. 3.8. Доведення роботи функції «Редагування»

Як можна побачити, 1 та 3 записи співпадають. Функція працює вірно.

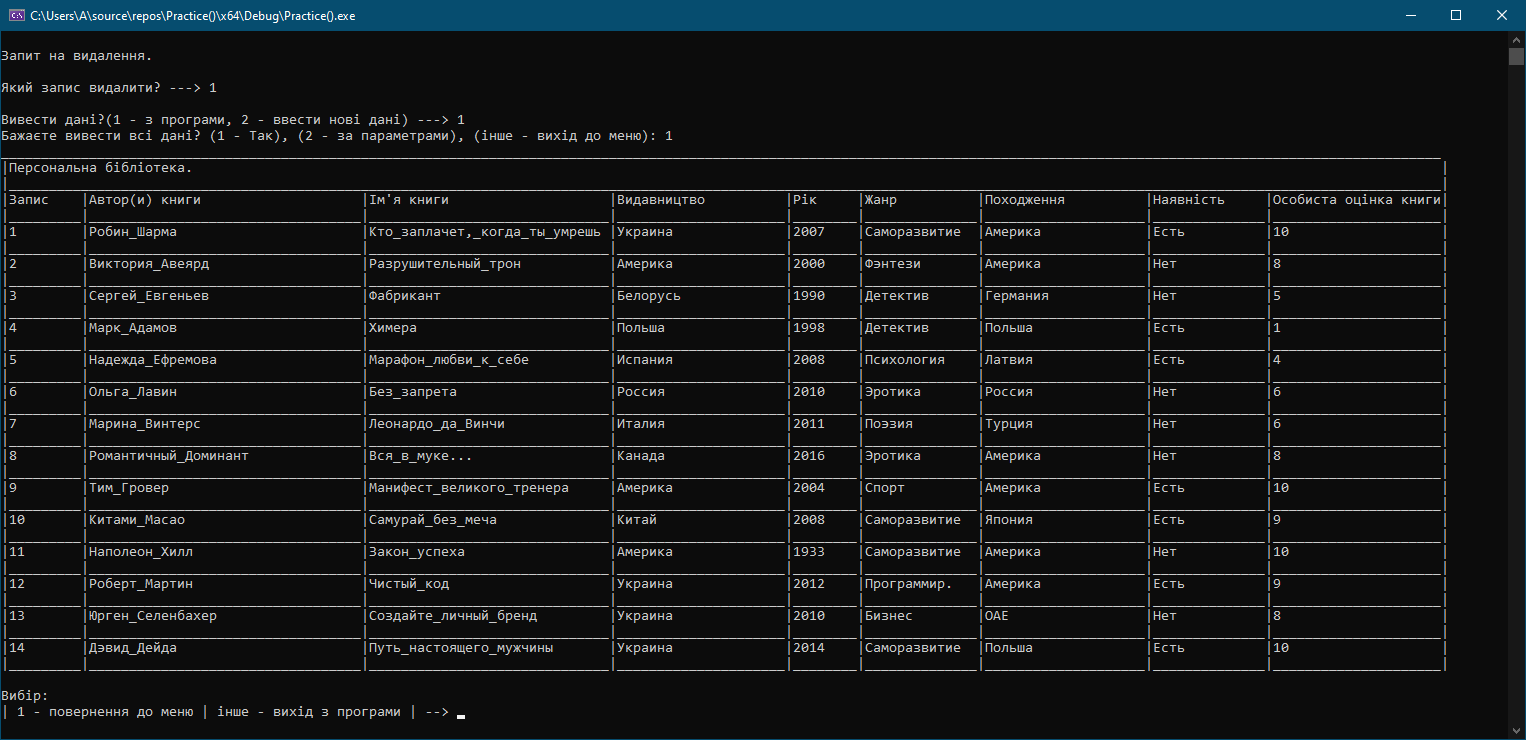
Наступним у меню йде «Видалення(5)». Давайте покажемо та опишемо її роботу: 

Рис. 3.9. Вигляд консолі при виконанні функції «Видалення» та доведення її роботи

Для початку треба ввести номер запису який потрібно видалити. Тут той самий недолік, який був у функції «Редагування» - немає можливості видалення номера запису за параметром. Все ж таки, в завданні проекту не було вказано як саме проводити видалення і, також, чи використовувати параметри при видаленні запису з файлу. Звичайно, для правильної роботи з даними перед початком кожної функції проводиться зчитування даних з файлу та запис їх для подальшого використання. Ніяких проблем з написанням функції не виникало, багів виявлено не було.

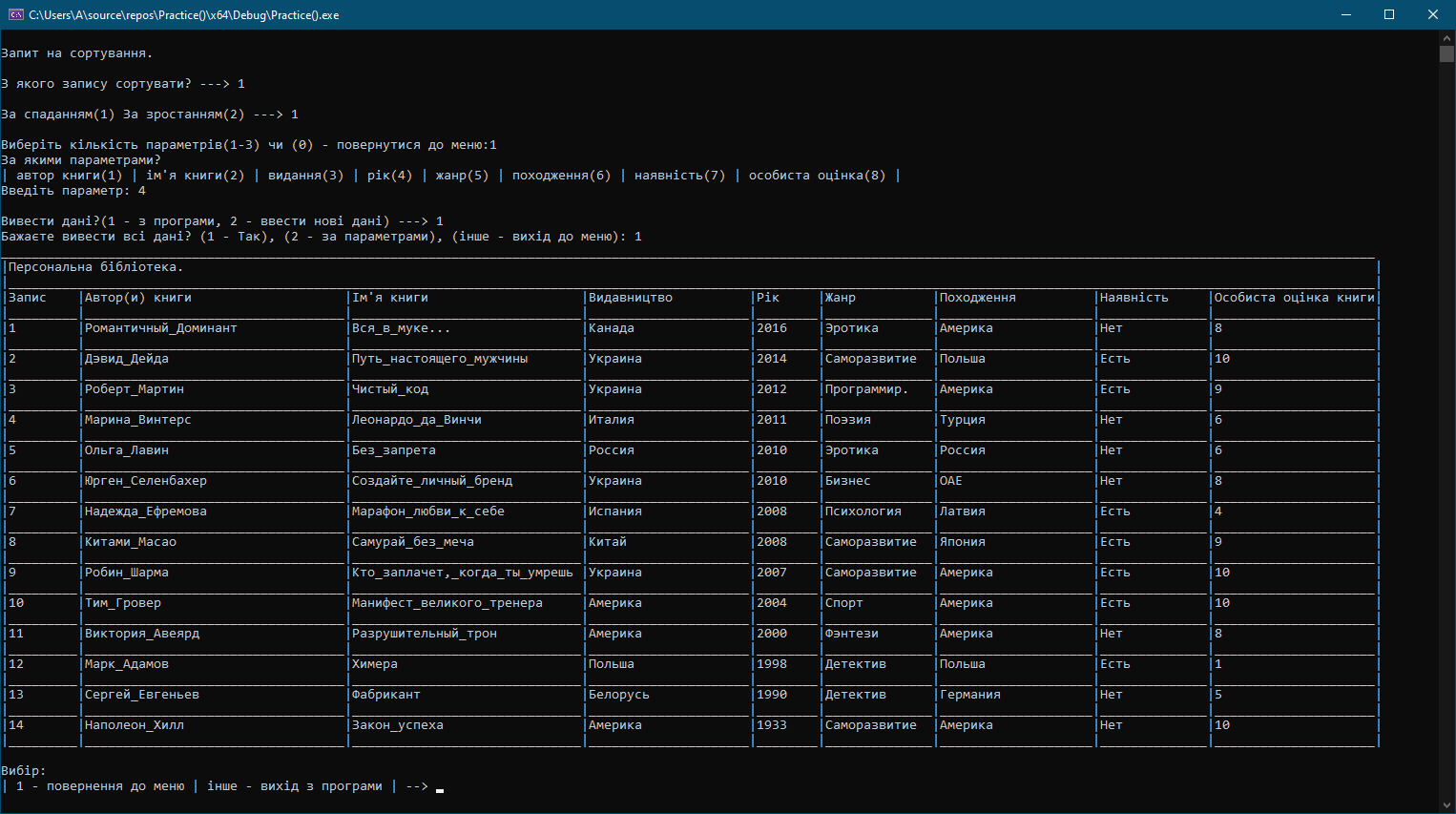
Рушаємо далі. Наступну функцію, яку ми будемо тестувати - «Впорядкування(6)». Ось її загальний вигляд у консолі: 

Рис. 3.10. Вигляд консолі при виконанні функції «Впорядкування» та доведення її роботи

Спочатку, користувачу потрібно ввести номер запису з якого буде відбуватися сортування, далі яким способом сортувати: за спаданням(1), чи за зростанням(2). Потім користувач має ввести кількість потрібних йому параметрів та їх номери. Програма, відповідно, вирішує за допомогою перевірки яким чином сортувати та виконує поставлене завдання. На кінець вона викликає функцію «Виведення» та виводить відсортовані дані з файлів та записує новий порядок у бінарний файл.

Для остаточної перевірки працездатності функції подивимось на верхній рисунок ще раз. Всі дані в колонці «Рік» були виставлені по спаданню, як і хотіли.

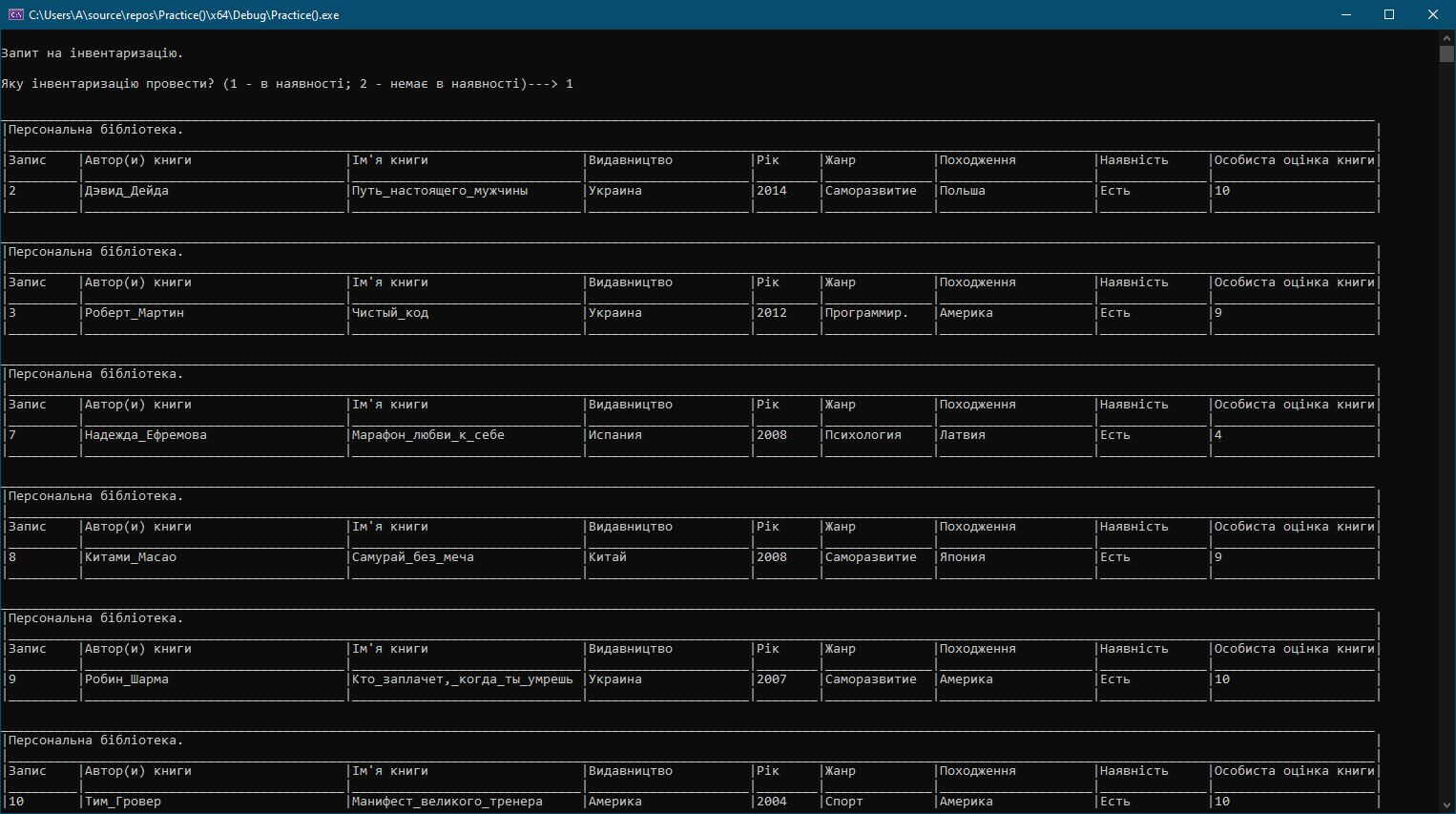
І на кінець, визначимо та проведемо тестування останньої функції додатку, специфічної для 1-ого варіанту, це «Інвентаризація бібліотеки(7)». Ось її наочний вигляд у консолі: 

Рис. 3.11. Вигляд консолі при виконанні функції «Інвентаризація бібліотеки» та доведення її роботи

Працює вона дуже просто. Для початку її роботи лише потрібно ввести за яким параметром проводити інвентаризацію: чи за наявністю той чи іншої книги у бібліотеці, чи за відсутністю, і програма сама виконає виведення потрібної інформації, яка наявна у базі даних.

# **ВИСНОВКИ**

Під час виконання поставленого завдання на практику були отримані практичні навички роботи з масивами структур та файлами, також було продемонстровано ці навички при написанні додатку, який працює правильно та без будь-яких проблем.

В першому розділі «Аналіз задачі» було проаналізовано поставлену задачу, а саме які дії потрібно виконати для повної розробки програми. Стало зрозуміло, що потрібно за чим робити, а також як повинен працювати додаток у її фінальній версії.

В другому розділі було узагальнено попередні задачі та описано їх логічну роботу та проілюстровано за допомогою блок-схем. Цей етап роботи та розробки програмного забезпечення є одним з найважливіших, оскільки саме від проектування та логічної реалізації всіх функцій програмного додатку залежить надійність його роботи.

В третьому розділі було описано повну роботу готової програми і виправлено всі його недоліки.

Підсумовуючи вище сказане, можем сказати, що ця робота дала нам змогу закріпити раніше вивчений матеріал, написавши для цього додаток. Тобто, задача на практику була успішно виконана та досягнута!

# **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

# Функція fprintf - http://www.c-cpp.ru/books/fprintf-i-fscanf

# getch, getche - <http://www.c-cpp.ru/content/getch-getche>

# Редактор блок-схем - <https://programforyou.ru/block-diagram-redactor>

# Загальна характеристика мови «Сі» - <https://sites.google.com/site/azykmetodprog/vvedenie-v-azyk-programmirovania-si/obsaa-harakteristika-azyka-i-primer-programmy-na-si>

# Правопис різних слів та словосполучень.

# Ресурси програм з <https://learn.ztu.edu.ua/course/view.php?id=2972>

# ЛІТЕРАТУРА

# Вінник В.Ю. Алгоритмічні мови та основи програмування: мова С - Житомир: ЖДТУ, 2007. – 328 с.

# Войтенко В. В., Морозов А. В. С\С++ Практика програмування. Навчально методичний посібник - Житомир: ЖДТУ, 2003. – 324 с.

# Юркин А.Г. Задачник по программированию. – СПб.: Питер, 2002. – 192 с.

# ДСТУ 3008-2015 "Документація. Звіти в сфері науки і техніки. Структура і правила оформлення" – Державний стандарт України.

# Bhattacharya Debabrata. C Programming: The Complete Guide. MTech, 2018. — 239 p.

# Bichkar R.S. Programming with C. Universities Pres, 2017. — 1184p.

# C Notes for Professionals. GoalKicker.com, 2018. — 341 p.

# Kochan Stephen G. Programming in C. 4th edition. — Pearson Education, Inc., 2015. — 1285 p.о

# Gabriel T. C Programming: The Tutorial. Amazon Digital Services, 2016. — 1341 p.

# Zambon G. Practical C. Apress, 2016. — 477 p. Universities Pres, 2017. — 1184 p.

# Доун Г. зучаем программирование на C / Гриффитс Доун. – Москва: Эксмо, 2019. – 624 с.

# Перри, Г. Программирование на C для начинающих / Г. Перри, Д. Миллер. - М.: Эксмо, 2015. - 368 c

# Прата С. Язык программирования C (Си). Лекции и упражнения / Стивен Прата. – Питер: Вильямс, 2012. – 960 с. – (5-е).

# Павловская Т.А. Програмирование на языке высокого уровня / Т.А. Павловская – СПб.: Питер, 2011, – 461с.: ил.

# **ДОДАТКИ**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <windows.h>

#include <conio.h>

#include <ctype.h>

//глобальные переменные

FILE\* binary = NULL; FILE\* outputfile = NULL; FILE\* outputparameters = NULL;

int count\_parameters = 0, parameters[3] = { 0 }, count\_notes = 0;

// глобальные функции

void readBinary();

void writeBinary();

void menu();

int exit();

void input();

void output\_header();

void parameters\_();

void output\_parameters(int q);

void output\_parameters2\_0(int z);

void output();

void searching\_();

void editing\_();

void delete\_();

void sorting\_();

void inventory();

// структуры

struct library\_book\_information {

char authors[50];

char name[50];

char edition[50];

char year[50];

} \*book = NULL;

struct personal\_library {

struct library\_book\_information book;

char section[50];

char book\_origin[50];

char availability\_of\_a\_book[50];

char book\_evaluation[50];

} \*library = NULL;

int main() {

SetConsoleCP(1251); SetConsoleOutputCP(1251);

binary = fopen("binary.bin", "ab"); fclose(binary);

remove("output.txt");

outputfile = fopen("output.txt", "a"); fclose(outputfile);

library = (struct personal\_library\*)calloc(sizeof(struct personal\_library), 15);

book = (struct library\_book\_information\*)calloc(sizeof(struct library\_book\_information), 15);

menu();

}

void readBinary() {

FILE\* binary = fopen("binary.bin", "ab");

if (binary == NULL) {

printf("Error opening file");

\_getch();

exit(-3);

}

fread(&count\_notes, sizeof(int), 1, binary);

fread(&library, sizeof(personal\_library), 1, binary);

}

void writeBinary() {

FILE\* binary = fopen("binary.bin", "wb");

if (binary == NULL) {

printf("Error opening file");

\_getch();

exit(-3);

}

fwrite(&count\_notes, sizeof(int), 1, binary);

fwrite(&library, sizeof(personal\_library), count\_notes, binary);

fclose(binary);

}

void menu() {

printf("\nПрактика. 24.01.22. Варіант 1: Персональна бібліотека. Виконавець: Бабушко Андрій Сергійович. Група: ВТ-21-1(1).\n| Введення даних(1) | Виведення даних(2) | Пошук(3) | Рудагування(4) | Видалення(5) | Впорядкування(6) | Інвентаризація бібліотеки(7) | |Вихід з програми(інше). |");

int i;

while (!\_kbhit) {}

printf("\nВибір(0-7): "); i = \_getche();

switch (i) {

case 49:

system("cls");

input();

break;

case 50:

system("cls");

output();

break;

case 51:

system("cls");

searching\_();

break;

case 52:

system("cls");

editing\_();

break;

case 53:

system("cls");

delete\_();

break;

case 54:

system("cls");

sorting\_();

break;

case 55:

system("cls");

inventory();

break;

default:

printf("\nВи впевнені?\n");

exit();

break;

}

}

int exit() {

char m;

while (!\_kbhit) {}

printf("\nВибір:\n| 1 - повернення до меню | інше - вихід з програми | --> "); m = \_getche();

switch (m) {

case 49:

system("cls");

menu();

break;

default:

system("cls");

return 1;

break;

}

}

void input()

{

printf("\nВведіть кількість записів:"); scanf("%d", &count\_notes);

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

printf("\tЗапис %d:\n", i + 1);

printf("Інформація про книгу =>");

printf(" Автор книги:"); scanf("%s", &library[i].book.authors);

printf("\t\t\tIм'я книги:"); scanf("%s", &library[i].book.name);

printf("\t\t\tВидавництво:"); scanf("%s", &library[i].book.edition);

printf("\t\t\tРік:"); scanf("%s", &library[i].book.year);

printf("\t\t\tЖанр книги(детектив, фантастика, пригоди, роман, наука, гумор, фольклор):"); scanf("%s", &library[i].section);

printf("\t\t\tПоходження:"); scanf("%s", &library[i].book\_origin);

printf("\t\t\tНаявність: "); scanf("%s", &library[i].availability\_of\_a\_book);

printf("\t\t\tОсобиста оцінка книги(0-10): "); scanf("%s", &library[i].book\_evaluation);

}

writeBinary();

exit();

}

void output\_header() {

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("|Персональна бібліотека. |\n");

printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

printf("|Запис |Автор(и) книги |Iм'я книги |Видавництво |Рік |Жанр |Походження |Наявність |Особиста оцінка книги|\n");

printf("|\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

// запис таблиці у файл

FILE\* outputfile = fopen("output.txt", "w");

fprintf(outputfile, "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

fprintf(outputfile, "|Персональна бібліотека. |\n");

fprintf(outputfile, "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

fprintf(outputfile, "|Запис |Автор(и) книги |Iм'я книги |Видавництво |Рік |Жанр |Походження |Наявність |Особиста оцінка книги|\n");

fprintf(outputfile, "|\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

fclose(outputfile);

fclose(binary);

}

void parameters\_() {

printf("\nВиберіть кількість параметрів(1-3) чи (0) - повернутися до меню:"); scanf("%d", &count\_parameters);

if (count\_parameters == 0) {

system("cls");

menu();

}

printf("За якими параметрами?\n| автор книги(1) | ім'я книги(2) | видання(3) | рік(4) | жанр(5) | походження(6) | наявність(7) | особиста оцінка(8) |\n");

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

printf("Введіть параметр: "); scanf("%d", &parameters[i]);

}

}

void output\_parameters(int q) {

if (parameters[q] == 1) {

char author[50];

printf("\nВведіть автора книги:"); scanf("%s", author);

for (int z = 0; z < count\_notes; z++) {

if (strcmp(author, library[z].book.authors) == 0) {

output\_header();

output\_parameters2\_0(z);

}

}

}

else {

if (parameters[q] == 2) {

char book\_name[50];

printf("\nВведіть ім'я книги:"); scanf("%s", book\_name);

for (int z = 0; z < count\_notes; z++) {

if (strcmp(book\_name, library[z].book.name) == 0) {

output\_header();

output\_parameters2\_0(z);

}

}

}

else {

if (parameters[q] == 3) {

char book\_edition[50];

printf("\nВведіть видання книги:"); scanf("%s", book\_edition);

for (int z = 0; z < count\_notes; z++) {

if (strcmp(book\_edition, library[z].book.edition) == 0) {

output\_header();

output\_parameters2\_0(z);

}

}

}

else {

if (parameters[q] == 4) {

char book\_year[50];

printf("\nВведіть рік:"); scanf("%s", book\_year);

for (int z = 0; z < count\_notes; z++) {

if (strcmp(book\_year, library[z].book.year) == 0) {

output\_header();

output\_parameters2\_0(z);

}

}

}

else {

if (parameters[q] == 5) {

char section[50];

printf("\nВведіть жанр:"); scanf("%s", section);

for (int z = 0; z < count\_notes; z++) {

if (strcmp(section, library[z].section) == 0) {

output\_header();

output\_parameters2\_0(z);

}

}

}

else {

if (parameters[q] == 6) {

char book\_origin[50];

printf("\nВведіть походження:"); scanf("%s", book\_origin);

for (int z = 0; z < count\_notes; z++) {

if (strcmp(book\_origin, library[z].book\_origin) == 0) {

output\_header();

output\_parameters2\_0(z);

}

}

}

else {

if (parameters[q] == 7) {

char availability\_of\_a\_book[50];

printf("\nВведіть наявність:"); scanf("%s", availability\_of\_a\_book);

for (int z = 0; z < count\_notes; z++) {

if (strcmp(availability\_of\_a\_book, library[z].availability\_of\_a\_book) == 0) {

output\_header();

output\_parameters2\_0(z);

}

}

}

else {

if (parameters[q] == 8) {

char book\_evaluation[50];

printf("\nОцінку книги:"); scanf("%s", &book\_evaluation);

for (int z = 0; z < count\_notes; z++) {

if (strcmp(book\_evaluation, library[z].book\_evaluation) == 0) {

output\_header();

output\_parameters2\_0(z);

}

}

}

else {

printf("\nЗапис не знайдено.\n");

}

}

}

}

}

}

}

}

}

void output\_parameters2\_0(int z) {

int temp;

char spaces[9][50] = { " ", "Автор книги ", "Iм'я книги ", "Видавництво ", "Рік ","Жанр ","Походження ","Наявність ","Особиста оцінка книги" };

for (int i = 0; i < 9; i++) {

printf("|");

fprintf(outputfile, "|");

for (int j = 0; j <= strlen(spaces[i]); j++) {

if (i == j) {

printf("%d", z + 1);

fprintf(outputfile, "%d", z + 1);

int x = 1;

if (z > 8) {

x++;

}

temp = strlen(spaces[i]) - x;

for (int k = 0; k < temp; k++) {

printf(" ");

fprintf(outputfile, " ");

}

temp = 0;

break;

}

if (i == 1) {

fread(library[z].book.authors, sizeof(char), 50, binary);

printf("%s", library[z].book.authors);

fprintf(outputfile, "%s", library[z].book.authors);

temp = strlen(spaces[i]) - strlen(library[z].book.authors);

for (int k = 0; k < temp; k++) {

printf(" ");

fprintf(outputfile, " ");

}

break;

}

if (i == 2) {

printf("%s", library[z].book.name);

fprintf(outputfile, "%s", library[z].book.name);

temp = strlen(spaces[i]) - strlen(library[z].book.name);

for (int k = 0; k < temp; k++) {

printf(" ");

fprintf(outputfile, " ");

}

break;

}

if (i == 3) {

printf("%s", library[z].book.edition);

fprintf(outputfile, "%s", library[z].book.edition);

temp = strlen(spaces[i]) - strlen(library[z].book.edition);

for (int k = 0; k < temp; k++) {

printf(" ");

fprintf(outputfile, " ");

}

break;

}

if (i == 4) {

printf("%s", library[z].book.year);

fprintf(outputfile, "%s", library[z].book.year);

temp = strlen(spaces[i]) - strlen(library[z].book.year);

for (int k = 0; k < temp; k++) {

printf(" ");

fprintf(outputfile, " ");

}

break;

}

if (i == 5) {

printf("%s", library[z].section);

fprintf(outputfile, "%s", library[z].section);

temp = strlen(spaces[i]) - strlen(library[z].section);

for (int k = 0; k < temp; k++) {

printf(" ");

fprintf(outputfile, " ");

}

break;

}

if (i == 6) {

printf("%s", library[z].book\_origin);

fprintf(outputfile, "%s", library[z].book\_origin);

temp = strlen(spaces[i]) - strlen(library[z].book\_origin);

for (int k = 0; k < temp; k++) {

printf(" ");

fprintf(outputfile, " ");

}

break;

}

if (i == 7) {

printf("%s", library[z].availability\_of\_a\_book);

fprintf(outputfile, "%s", library[z].availability\_of\_a\_book);

temp = strlen(spaces[i]) - strlen(library[z].availability\_of\_a\_book);

for (int k = 0; k < temp; k++) {

printf(" ");

fprintf(outputfile, " ");

}

break;

}

if (i == 8) {

printf("%s", library[z].book\_evaluation);

fprintf(outputfile, "%s", library[z].book\_evaluation);

temp = strlen(spaces[i]) - strlen(library[z].book\_evaluation);

for (int k = 0; k < temp; k++) {

printf(" ");

fprintf(outputfile, " ");

}

break;

}

}

}

printf("|\n");

fprintf(outputfile, "|\n");

printf("|\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

fprintf(outputfile, "|\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

}

void output()

{

char data\_;

while (!\_kbhit) {}

printf("\nВивести дані?(1 - з програми, 2 - ввести нові дані) ---> "); data\_ = \_getche();

switch (data\_) {

case 49: {

char i;

while (!\_kbhit) {}

printf("\nБажаєте вивести всі дані? (1 - Так), (2 - за параметрами), (інше - вихід до меню): "); i = \_getche();

switch (i) {

case 49: {

readBinary();

output\_header();

for (int z = 0; z < count\_notes; z++) {

output\_parameters2\_0(z);

}

fclose(binary);

fclose(outputfile);

exit();

break;

}

case 50: {

readBinary();

parameters\_();

for (int q = 0; q < count\_parameters; q++) {

output\_parameters(q);

}

fclose(binary);

exit();

break;

}

default: {

exit();

break;

}

}

break;

}

case 50: {

input();

break;

}

}

}

void searching\_() { //функція сортування

readBinary();

printf("\nЗапит на пошук.\n");

if (count\_notes == 0) {

printf("В базі даних 0 записів.\n");

exit();

}

parameters\_();

for (int b = 0; b < count\_parameters; b++) {

output\_parameters(b);

}

fclose(binary);

exit();

}

void editing\_() {

printf("\nЗапит на редагування.");

if (count\_notes == 0) {

printf("В базі даних 0 записів.\n");

exit();

}

int i;

while(!\_kbhit){}

printf("\nЯкий запис редагувати? ---> "); scanf("%d", &i); i = i - 1;

printf("\tЗапис %d:\n", i + 1);

printf("Інформація про книгу =>");

printf(" Автор книги:"); scanf("%s", library[i].book.authors); fwrite(library[i].book.authors, sizeof(char), 50, binary);

printf("\t\t\tIм'я книги:"); scanf("%s", library[i].book.name); fwrite(library[i].book.name, sizeof(char), 50, binary);

printf("\t\t\tВидавництво:"); scanf("%s", library[i].book.edition); fwrite(library[i].book.edition, sizeof(char), 50, binary);

printf("\t\t\tРік:"); scanf("%s", library[i].book.year); fwrite(library[i].book.year, sizeof(char), 50, binary);

printf("\t\t\tЖанр книги(детектив, фантастика, пригоди, роман, наука, гумор, фольклор):"); scanf("%s", library[i].section); fwrite(library[i].section, sizeof(char), 50, binary);

printf("\t\t\tПоходження:"); scanf("%s", library[i].book\_origin); fwrite(library[i].book\_origin, sizeof(char), 50, binary);

printf("\t\t\tНаявність: "); scanf("%s", library[i].availability\_of\_a\_book); fwrite(library[i].availability\_of\_a\_book, sizeof(char), 50, binary);

printf("\t\t\tОсобиста оцінка книги(0-10): "); scanf("%s", library[i].book\_evaluation); fwrite(library[i].book\_evaluation, sizeof(char), 50, binary);

writeBinary();

exit();

}

void delete\_() {

readBinary();

printf("\nЗапит на видалення.\n");

if (count\_notes == 0) {

printf("В базі даних 0 записів.\n");

exit();

}

int i;

while (!\_kbhit) {}

printf("\nЯкий запис видалити? ---> "); scanf("%d", &i); i = i - 1;

for (int j = i; j < count\_notes; j++) {

library[j] = library[j + 1];

book[j] = book[j + 1];

}

count\_notes--; writeBinary();

fclose(binary); output();

}

void sorting\_() {

printf("\nЗапит на сортування.\n");

if (count\_notes == 0) {

printf("В базі даних 0 записів.\n");

exit();

}

int j;

printf("\nЗ якого запису сортувати? ---> "); scanf("%d", &j); j = j - 1;

int sort;

printf("\nЗа спаданням(1)\tЗа зростанням(2) ---> "); scanf("%d", &sort);

readBinary();

parameters\_();

switch (sort) {

case 1: {

int fl;

do {

fl = 0;

for (int z = 0; z < count\_parameters; z++) {

for (int i = j; i < count\_notes; i++) {

if (parameters[z] == 1) {

if (strcmp(library[i].book.authors, library[i + 1].book.authors) < 0) {

library[count\_notes] = library[i];

library[i] = library[i + 1];

library[i + 1] = library[count\_notes];

book[count\_notes] = book[i];

book[i] = book[i + 1];

book[i + 1] = book[count\_notes];

fl = 1;

}

}

if (parameters[z] == 2) {

if (strcmp(library[i].book.name, library[i + 1].book.name) < 0) {

library[count\_notes] = library[i];

library[i] = library[i + 1];

library[i + 1] = library[count\_notes];

book[count\_notes] = book[i];

book[i] = book[i + 1];

book[i + 1] = book[count\_notes];

fl = 1;

}

}

if (parameters[z] == 3) {

if (strcmp(library[i].book.edition, library[i + 1].book.edition) < 0) {

library[count\_notes] = library[i];

library[i] = library[i + 1];

library[i + 1] = library[count\_notes];

book[count\_notes] = book[i];

book[i] = book[i + 1];

book[i + 1] = book[count\_notes];

fl = 1;

}

}

if (parameters[z] == 4) {

if (atoi(library[i].book.year) < atoi(library[i + 1].book.year)) {

library[count\_notes] = library[i];

library[i] = library[i + 1];

library[i + 1] = library[count\_notes];

book[count\_notes] = book[i];

book[i] = book[i + 1];

book[i + 1] = book[count\_notes];

fl = 1;

}

}

if (parameters[z] == 5) {

if (strcmp(library[i].section, library[i + 1].section) < 0) {

library[count\_notes] = library[i];

library[i] = library[i + 1];

library[i + 1] = library[count\_notes];

book[count\_notes] = book[i];

book[i] = book[i + 1];

book[i + 1] = book[count\_notes];

fl = 1;

}

}

if (parameters[z] == 6) {

if (strcmp(library[i].book\_origin, library[i + 1].book\_origin) < 0) {

library[count\_notes] = library[i];

library[i] = library[i + 1];

library[i + 1] = library[count\_notes];

book[count\_notes] = book[i];

book[i] = book[i + 1];

book[i + 1] = book[count\_notes];

fl = 1;

}

}

if (parameters[z] == 7) {

if (strcmp(library[i].availability\_of\_a\_book, library[i + 1].availability\_of\_a\_book) < 0) {

library[count\_notes] = library[i];

library[i] = library[i + 1];

library[i + 1] = library[count\_notes];

book[count\_notes] = book[i];

book[i] = book[i + 1];

book[i + 1] = book[count\_notes];

fl = 1;

}

}

if (parameters[z] == 8) {

if (atoi(library[i].book\_evaluation) < atoi(library[i + 1].book\_evaluation)) {

library[count\_notes] = library[i];

library[i] = library[i + 1];

library[i + 1] = library[count\_notes];

book[count\_notes] = book[i];

book[i] = book[i + 1];

book[i + 1] = book[count\_notes];

fl = 1;

}

}

}

}

} while (fl);

break;

}

case 2: {

int fl;

do {

fl = 0;

for (int z = 0; z < count\_parameters; z++) {

for (int i = j - 1; i < count\_notes; i++) {

if (parameters[z] == 1) {

if (strcmp(library[i].book.authors, library[i + 1].book.authors) > 0) {

library[count\_notes] = library[i];

library[i] = library[i + 1];

library[i + 1] = library[count\_notes];

book[count\_notes] = book[i];

book[i] = book[i + 1];

book[i + 1] = book[count\_notes];

fl = 1;

}

}

if (parameters[z] == 2) {

if (strcmp(library[i].book.name, library[i + 1].book.name) > 0) {

library[count\_notes] = library[i];

library[i] = library[i + 1];

library[i + 1] = library[count\_notes];

book[count\_notes] = book[i];

book[i] = book[i + 1];

book[i + 1] = book[count\_notes];

fl = 1;

}

}

if (parameters[z] == 3) {

if (strcmp(library[i].book.edition, library[i + 1].book.edition) > 0) {

library[count\_notes] = library[i];

library[i] = library[i + 1];

library[i + 1] = library[count\_notes];

book[count\_notes] = book[i];

book[i] = book[i + 1];

book[i + 1] = book[count\_notes];

fl = 1;

}

}

if (parameters[z] == 4) {

if (atoi(library[i].book.year) > atoi(library[i + 1].book.year)) {

library[count\_notes] = library[i];

library[i] = library[i + 1];

library[i + 1] = library[count\_notes];

book[count\_notes] = book[i];

book[i] = book[i + 1];

book[i + 1] = book[count\_notes];

fl = 1;

}

}

if (parameters[z] == 5) {

if (strcmp(library[i].section, library[i + 1].section) > 0) {

library[count\_notes] = library[i];

library[i] = library[i + 1];

library[i + 1] = library[count\_notes];

book[count\_notes] = book[i];

book[i] = book[i + 1];

book[i + 1] = book[count\_notes];

fl = 1;

}

}

if (parameters[z] == 6) {

if (strcmp(library[i].book\_origin, library[i + 1].book\_origin) > 0) {

library[count\_notes] = library[i];

library[i] = library[i + 1];

library[i + 1] = library[count\_notes];

book[count\_notes] = book[i];

book[i] = book[i + 1];

book[i + 1] = book[count\_notes];

fl = 1;

}

}

if (parameters[z] == 7) {

if (strcmp(library[i].availability\_of\_a\_book, library[i + 1].availability\_of\_a\_book) > 0) {

library[count\_notes] = library[i];

library[i] = library[i + 1];

library[i + 1] = library[count\_notes];

book[count\_notes] = book[i];

book[i] = book[i + 1];

book[i + 1] = book[count\_notes];

fl = 1;

}

}

if (parameters[z] == 8) {

if (atoi(library[i].book\_evaluation) > atoi(library[i + 1].book\_evaluation)) {

library[count\_notes] = library[i];

library[i] = library[i + 1];

library[i + 1] = library[count\_notes];

book[count\_notes] = book[i];

book[i] = book[i + 1];

book[i + 1] = book[count\_notes];

fl = 1;

}

}

}

}

} while (fl);

break;

}

}

writeBinary();

output();

}

void inventory() {

printf("\nЗапит на інвентаризацію.\n");

if (count\_notes == 0) {

printf("В базі даних 0 записів.\n");

exit();

}

readBinary();

int inventory\_book;

printf("\nЯку інвентаризацію провести? (1 - в наявності; 2 - немає в наявності)---> "); scanf("%d", &inventory\_book);

switch (inventory\_book) {

case 1:

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

if (strcmp("Есть", library[i].availability\_of\_a\_book) == 0) {

output\_header();

output\_parameters2\_0(i);

}

}

break;

case 2:

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

if (strcmp("Нет", library[i].availability\_of\_a\_book) == 0) {

output\_header();

output\_parameters2\_0(i);

}

}

break;

}

exit();

}