**ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ОЛЕСЯ ГОНЧАРА**

Кафедра електронних обчислювальних машин

**КУРСОВА РОБОТА**

з дисципліни «Програмування»

Студента I курсу групи КІ-21-2

Спеціальності 123 – «Комп’ютерна інженерія»

Шейко Р.О

(прізвище та ініціали)

Керівник:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів

Національна шкала

Члени комісії:

Карпенко Н.В.

(підпис)

(прізвище та ініціали)

Спірінцева О.В.

(підпис)

(підпис)

(прізвище та ініціали)

(прізвище та ініціали)

Гниленко О.Б.

м. Дніпро

2022

# **ЗАВДАННЯ НА КУРСОВУ РОБОТУ**

**Мета:** поглиблення і закріплення теоретичних знань та набуття практичних навичок роботи з розробки інформаційних систем.

**Постановка задачі:** Дані зберігаються в файлі на диску. Будь-яка інформаційна система (ІС) повинна мати інтерфейс користувача, де можна обирати потрібні дії. Функції, які повинна підтримувати ІС:

* додавання даних;
* редагування даних;
* видалення даних;
* сортування даних за обраним критерієм;
* пошук даних за обраним критерієм;
* виконання інших дій з даними (передбачається специфікою інформаційної системи).

**Програма повинна виконувати будь-яку дію без перезапуску*.***

**При виконанні завдання необхідно:**

1. Ім’я файлу задається в командному рядку. Якщо воно там не було задано, то після відповідного запиту вводиться користувачем з клавіатури.
2. Використовувати динамічне виділення пам’яті (розмір масиву задається користувачем після відповідного запиту). Після використання масиву слід обов’язково звільнити пам’ять.
3. Програма повинна мати зручне меню, в якому реалізовані всі функції, наведені у постановці завдання.
4. Інформацію для пошуку слід вводити з клавіатури.

***Домашня бібліотека***. Є інформація про автора книги, назву книги, розділ, до якого належить література даного виду (фантастика, пригоди, наукова, кулінарія тощо), місце, де стоїть книга, а також про те, чи знаходиться книга вдома. Якщо книга була комусь позичена, то слід передбачити поле, де буде вказано прізвище позичальника. Для перегляду книг слід передбачити їх сортування за вибором — або за прізвищем автора або за назвою книги. Пошук слід здійснювати за прізвищем автора, назвою книги (у тому разі і за окремими літерами) та за видом. Обробити ситуацію, коли під час пошуку за введеними даними нічого не знайдено. Передбачити ситуацію, коли позичена книга буде повернута або книгу побажають позичити.

**ЗМІСТ**

[**ЗАВДАННЯ НА КУРСОВУ РОБОТУ** 2](#_Toc105625182)

[**ВСТУП** 6](#_Toc105625183)

[**1.** **ЗБІР ТА АНАЛІЗ ВИМОГ** 7](#_Toc105625184)

[**1.1.** **Умова завдання** 7](#_Toc105625185)

[1.2. **Збір та аналіз вимог** 7](#_Toc105625186)

[**2.** **РЕАЛІЗАЦІЯ** 10](#_Toc105625187)

[**2.1.** **Опис логічної структури** 10](#_Toc105625188)

[**2.2.** **Крок 1. Створення функції main()** 11](#_Toc105625189)

[***2.2.1.*** ***Опис алгоритму функції main()*** 11](#_Toc105625194)

[**2.3.** **Крок 2. Створення функції redactor() та її гілка з функціями** 12](#_Toc105625203)

[***2.3.1 Функція redactor(). Опис її алгоритму*** 12](#_Toc105625204)

[***2.3.2 Функція redactorBook(). Опис її алгоритму*** 13](#_Toc105625205)

[***2.3.3 Функція redactorBookMenu(). Опис її алгоритму*** 13](#_Toc105625213)

[***2.3.4 Функція redactorBookNumMesto (). Опис її алгоритму*** 14](#_Toc105625214)

[***2.3.5 Функція redactorBookStatusMenu (). Опис її алгоритму*** 15](#_Toc105625224)

[***2.3.6 Функція redactorBookStatusAdd (). Опис її алгоритму*** 16](#_Toc105625235)

[***2.3.7 Функція redactorBookStatusDelete (). Опис її алгоритму*** 16](#_Toc105625247)

[**2.3.8** ***Функція redactorBookStatusZaemshik (). Опис її алгоритму*** 17](#_Toc105625260)

[***2.3.9 Функція redactorBookName(). Опис її алгоритму*** 18](#_Toc105625274)

[***2.3.10 Функція redactorBookAutor(). Опис її алгоритму*** 18](#_Toc105625289)

[***2.3.11 Функція redactorBookRozdil(). Опис її алгоритму*** 19](#_Toc105625305)

[***2.4.*** ***Крок 3. Створення функцій initArray(), initArrayStart(), check(), initKniga(), initKnigaFileR()*** 19](#_Toc105625311)

[***2.4.1 Функція initArray(). Опис її алгоритму*** 19](#_Toc105625314)

[***2.4.2 Функція initArrayStart(). Опис її алгоритму*** 20](#_Toc105625315)

[***2.4.3 Функція check (). Опис її алгоритму*** 21](#_Toc105625316)

[***2.4.4 Функція initKniga (). Опис її алгоритму*** 21](#_Toc105625317)

[***2.4.5 Функція initKnigaFileR (). Опис її алгоритму*** 22](#_Toc105625318)

[**2.5.** **Крок 4. Створення функцій initWrite(), write()** 23](#_Toc105625325)

[***2.5.1 Функція initWrite (). Опис її алгоритму*** 23](#_Toc105625326)

[**2.5.2 Функція write (). Опис її алгоритму** 23](#_Toc105625327)

[**2.6.** **Крок 5. Створення функцій addElement(), writeElementInFile ()** 24](#_Toc105625335)

[***2.6.1 Функція addElement (). Опис її алгоритму*** 24](#_Toc105625336)

[***2.6.2 Функція writeElementInFile (). Опис її алгоритму*** 24](#_Toc105625337)

[**2.7.** **Крок 6. Створення функцій deleteElement (), deleteElementChoosed ()** 25](#_Toc105625346)

[***2.7.1 Функція deleteElement (). Опис її алгоритму*** 25](#_Toc105625347)

[***2.7.2 Функція deleteElementChoosed (). Опис її алгоритму*** 25](#_Toc105625348)

[**2.8.** **Крок 7. Створення функцій displayArray (), displayKniga ()** 25](#_Toc105625358)

[***2.8.1 Функція displayArray (). Опис її алгоритму*** 25](#_Toc105625359)

[***2.8.2 Функція displayKniga (). Опис її алгоритму*** 25](#_Toc105625360)

[**2.9.** **Крок 8. Створення функцій sort(), sortNomerMesta(), sortName(), sortNazva()** 25](#_Toc105625361)

[***2.9.1 Функція sort (). Опис її алгоритму*** 25](#_Toc105625362)

[***2.9.2 Функція sortNomerMesta (). Опис її алгоритму*** 25](#_Toc105625363)

[***2.9.3 Функція sortName (). Опис її алгоритму*** 25](#_Toc105625364)

[***2.9.4 Функція sortNazva (). Опис її алгоритму*** 25](#_Toc105625365)

[**2.10.** **Крок 9. Створення функцій displayMenu (), displayСonsensus (), displayChoise (), displayChoiseRozdil (), displayChoiseAutor (), displayChoiseNazva ()** 25](#_Toc105625377)

[***2.10.1 Функція displayChoise (). Опис її алгоритму*** 25](#_Toc105625378)

[***2.10.2 Функція displayMenu (). Опис її алгоритму*** 25](#_Toc105625379)

[***2.10.3 Функція displayСonsensus (). Опис її алгоритму*** 25](#_Toc105625380)

[***2.10.4 Функція displayChoiseRozdil (). Опис її алгоритму*** 25](#_Toc105625381)

[***2.10.5 Функція displayChoiseAutor (). Опис її алгоритму*** 25](#_Toc105625382)

[**3.** **ТЕСТУВАННЯ** 26](#_Toc105625383)

[**3.1 Крок 1. Тестування меню програми(функції main)** 26](#_Toc105625387)

[**3.2 Крок 2. Тестування гілки редагування** 27](#_Toc105625392)

[**3.3 Крок 3. Тестування гілки створення та додавання нової книги** 31](#_Toc105625398)

[**3.4 Крок 4. Тестування гілки сортування** 31](#_Toc105625405)

[**3.5 Крок 5. Тестування гілки видалення** 32](#_Toc105625413)

[**3.6 Крок 6. Тестування гілки пошуку** 33](#_Toc105625429)

[**ВИСНОВОК** 35](#_Toc105625430)

[**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ** 36](#_Toc105625431)

[**ДОДАТКИ** 37](#_Toc105625432)

# **ВСТУП**

В ХХI столітті ІТ сфера ще більш, і більш пронизує наше повсякденне життя. Вранці ми використовуємо смартфон з додатками, що написані програмістами. Йдучи до місця роботи, ми беремо кофе, з кофе-машини, щоб підбадьоритися. А у вечері ми беремо книгу в бібліотеці й працівник відмічає у БД, що книгу взяли ми. Тож вивчення способів програмування дуже важливо на даному етапі для розвитку суспільства.

В цій роботі висвітлено дослідження в області програмування, й стосується вивченню та засвоєнню навичок з мови програмування «С». Вона є яскравим прикладом низькорівневої мови програмування, що може бути використана хоч у кофе-машині, хоч при написанні інформаційної системи у бібліотеці хоч у операційних системах, вбудованих системах та ін.

Мета курсової роботи концентрована на аналізі засобів мови С для реалізації задачі, що пов’язана зі створенням інформаційної системи «Домашня бібліотека». ІС повинні бути реалізовані такі функції: додавання даних; редагування даних; видалення даних; сортування даних за обраним критерієм; пошук даних за обраним критерієм; виконання інших дій з даними (передбачається специфікою інформаційної системи) [2].

Для досягнення мети роботи треба виконати наступні задачі:

1. Вивчення теоретичної частини за допомогою навчальних матеріалів та книг.
2. Навчитися використовувати та розуміти логіку використовуваних засобів мови.
3. Виконати наведені задачі за визначеними темами.
4. Додати описання використаних методів та отриманих даних.
5. Зробити висновки щодо виконаної роботи, закріпити матеріал.

Об’єктом дослідження є процес розробки програмного забезпечення.

# **ЗБІР ТА АНАЛІЗ ВИМОГ**

* 1. **Умова завдання**

Скласти інформаційну систему «*Домашня бібліотека*» для дослідження мови програмування «С».

***Домашня бібліотека***. Є інформація про автора книги, назву книги, розділ, до якого належить література даного виду (фантастика, пригоди, наукова, кулінарія тощо), місце, де стоїть книга, а також про те, чи знаходиться книга вдома.

Якщо книга була комусь позичена, то слід передбачити поле, де буде вказано прізвище позичальника.

Для перегляду книг слід передбачити їх сортування за вибором — або за прізвищем автора або за назвою книги.

Пошук слід здійснювати за прізвищем автора, назвою книги (у тому разі і за окремими літерами) та за видом.

Обробити ситуацію, коли під час пошуку за введеними даними нічого не знайдено.

Передбачити ситуацію, коли позичена книга буде повернута або книгу побажають позичити.

* 1. **Збір та аналіз вимог**

ІС «Домашня бібліотека» повинна надавати можливість користувачу:

• додавати книгу в бібліотеку;

• додавати в бібліотеку автора, назву, розділ, номер місця, статус та позичальника ;

• зберігати інформацію в файлі;

• відображати назву книги, автора, розділ, номер місця, статус та позичальника(якщо є);

• виконувати пошук за назвою, автором і видом;

• виконувати пошук за частиною слова;

• редагувати інформацію з бібліотеки (назву книги, автора, розділ, номер місця, статус та позичальника);

• видаляти непотрібні книгу;

• сортувати за автором, за назвою, номером місця.

* 1. **Моделювання вимог**

На рис. 1.1 наведено моделювання вимог користувача до ІС «Домашня бібліотека» за допомогою діаграми варіантів використання (use-case diagram). Оскільки кожній людині, яка буде користуватись домашньої бібліотекої буде надано однаковий функціонал, то у нашому випадку буде лише один актор. Він може виконувати дії (прецеденти): додати, редагувати чи видаляти контакти, переглядати (перелік книг, результати пошуку тощо), сортувати за обраним критерієм, виконувати пошук.

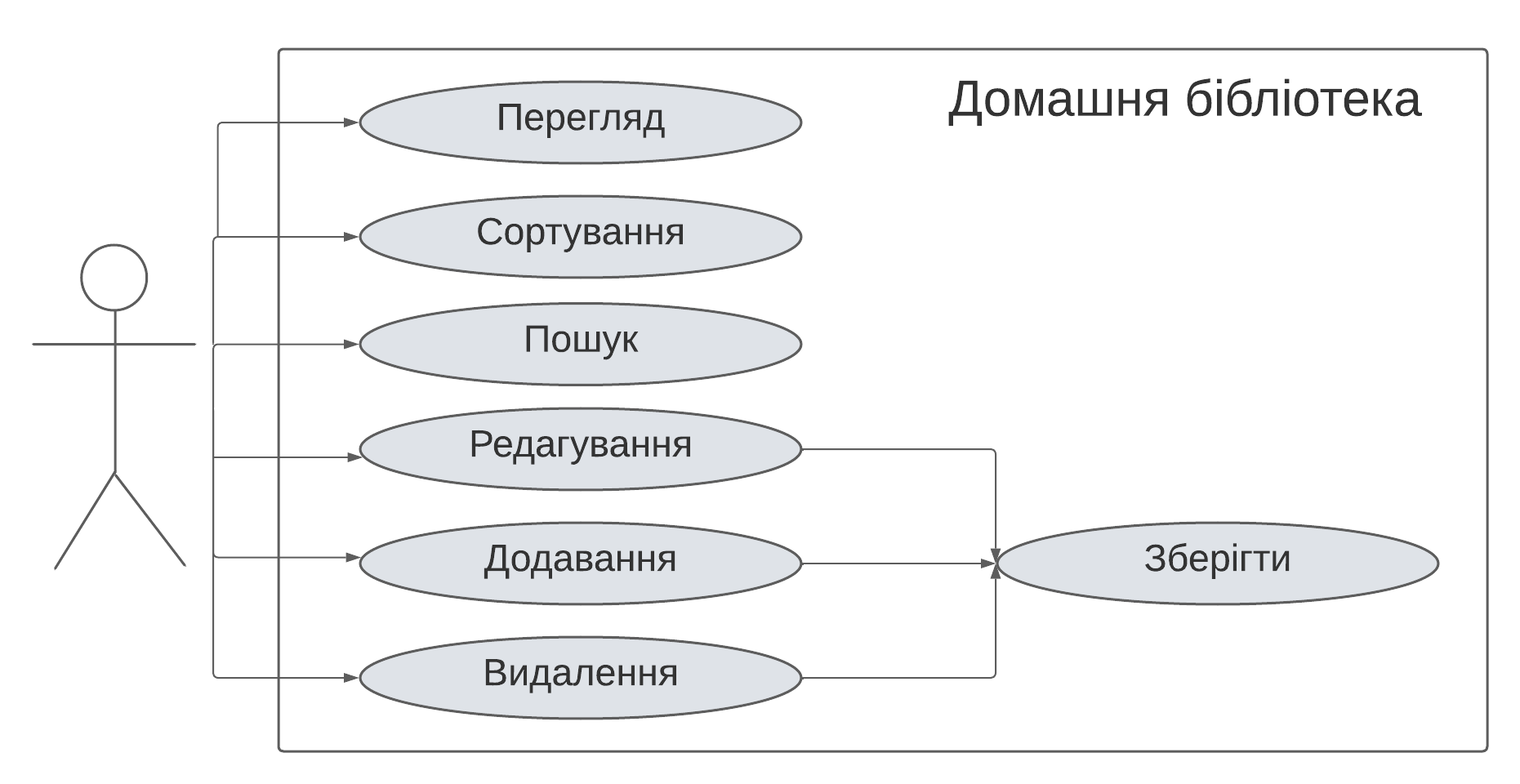


Рисунок 1.1. – Моделювання вимог користувача за допомогою діаграми use-case

Кожен прецедент – це окрема функція в системі, яка складається з послідовності виконуваних дій. Поведінку не тільки системи, але й окремої функції можна змоделювати за допомогою діаграми активності (Activity diagram). Зовні вона нагадує блок-схему, в якій замість блоків «Початок» та «Кінець» використовують спеціальні символи – чорний круг та «котяче око». Ця діаграма підтримує малювання оператору switch (множинний вибір) та розпаралелювання потоків (використовується під час паралельних та розподілених обчислень). Наприклад, на рис. 1.2 показано моделювання прецеденту «редактор» за допомогою діаграми активностей.

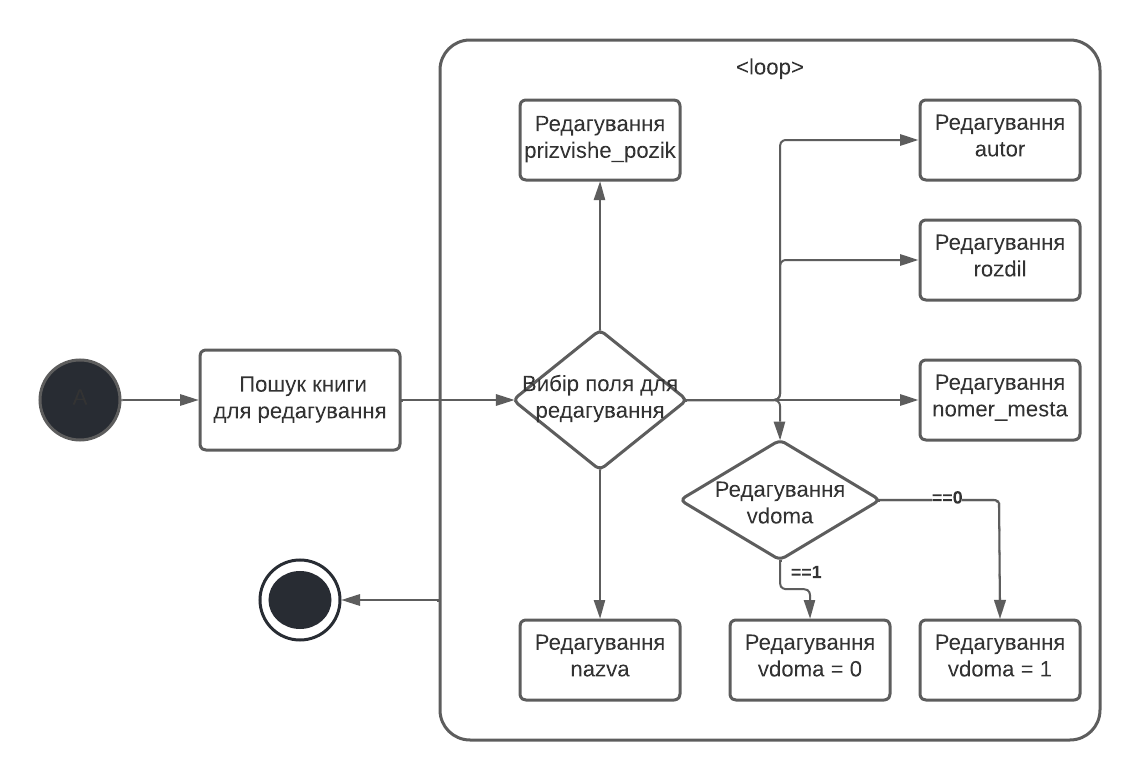


Рисунок 1.2. – Моделювання прецеденту «редактор» за допомогою Activity diagram

1. **РЕАЛІЗАЦІЯ**
   1. **Опис логічної структури**

Програма складається з двох програмних модулів - файлу **main.c** та **source.h**. До складу модулів входять такі функції:

**main()** – головна функція програми**;**

**check() -** функція перевірки кінця структури у файлу(шукає таку строчку- - - 0 0 -);

**sort()** – функція вибору сортування**;**

**sortName()** – функція сортування по імені**;**

**sortNazva()** – функція сортування по назві**;**

**sortNomerMesta()** – функція сортування по номеру місця**;**

**displayChoiseNazva()** -функція пошуку по назві;

**displayChoiseAutor ()** – функція пошуку по автору;

**displayChoiseRozdil()** -функція пошуку по розділу;

**displayChoise ()** –функція пошуку по чіткому введенню;

**displayСonsensus ()** -функція пошуку по збігу;

**displayMenu ()** - функція вибору пошуку;

**displayKniga()** –функція відображення інформації про структуру;

**displayArray ()** - функція передачi структури в функцію для друку інформації;

**initKniga()** -функція ініціалізації структури користувачем;

**initKnigaFileR ()** – функція ініціалізації структури через файл;

**deleteElement ()** -функція вибору для видалення елементу; **deleteElementChoosed ()** -функція видалення обраного елементу;

**addElement()** -функція добавлення елементу;

**writeElementInFile ()** –функція створення та вписування нового елементу у файл;

**initWrite ()** -функція ініціалізації запису;

**write ()** -функція запису у файл;

**initArrayStart ()** – функція створення динамічного масиву структур при старту програми;

**initArray ()** – функція створення динамічного масиву структур;

**redactor()** -функція підтвердження редагування;

**redactorBook()** -функція вибору книги для редагування;

**redactorBookMenu()** -функція вибору параметру для редагування;

**redactorBookZaemshik ()** -функція зміни позичальника;

**redactorBookNumMesto ()** -функція зміни номера місця;

**redactorBookName ()** -функція зміни ім’я;

**redactorBookAutor ()** -функція зміни автора;

**redactorBookRozdil ()** -функція зміни розділу;

**redactorBookStatusDelete ()** -функція зміни статусу книги на бібліотека;

**redactorBookStatusAdd ()** -функція зміни статусу книги на дома;

**redactorBookStatusMenu ()** -функція зміни статусу книги на дома;

## **Крок 1. Створення функції main()**



### ***Опис алгоритму функції main()***

На першому кроці ми створюємо функцію **main().** Вона призначена для взаємодії користувача з інтерфейсом. Функція ініціалізує масив структур, видає запит користувачу на вибір в меню.

Для виклику функцій потрібно увести одну з представлених цифр. Меню(інтерфейс) представлено на рис 2.1.

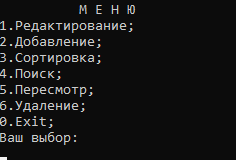


Рис 2.1.- Ілюстрація меню

При введені 1, запускається функція **redactor(),** починає роботу гілка редагування.

При введені 2, запускається функція **addElement(),** починає роботу гілка створення і додавання нової книги до масиву книг.

При введені 3, запускається функція **sort(),** починає роботу гілка сортування.

При введені 4, запускається функція **displayMenu(),** починає роботу гілка пошуку.

При введені 5, запускається функція **displayArray (),** виводяться усі книги.

При введені 6, запускається функція **deleteElement (),** що видаляє книги.

У блок-схемі функції **main()** приведено алгоритм роботи , що знаходиться у **додатку В.**



## **Крок 2. Створення функції redactor() та її гілка з функціями**

### ***2.3.1 Функція redactor(). Опис її алгоритму***

У цій функції при виборі 1 ми виводимо список усіх книг, на 2 запускається функція вибору книги, а на 0 організований вихід з функції й гілки редагування. На рис 2.2 ілюстрована робота функції.

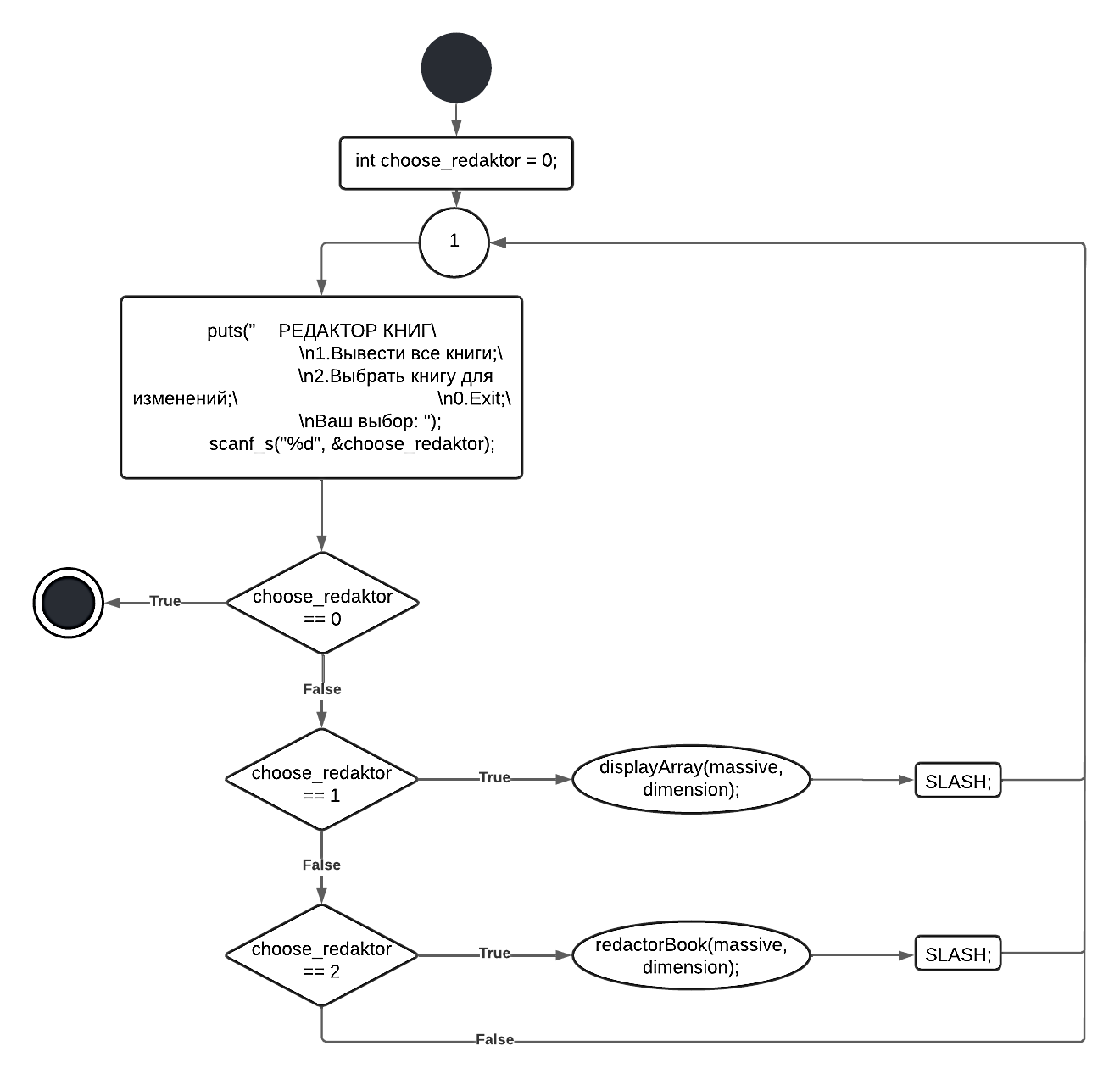


Рис 2.2. – Блок схема функції redactor()

***2.3.2 Функція redactorBook(). Опис її алгоритму***

У цій функції ми вибираємо книгу для редагування. На рис.2.3 проілюстрована її робота.

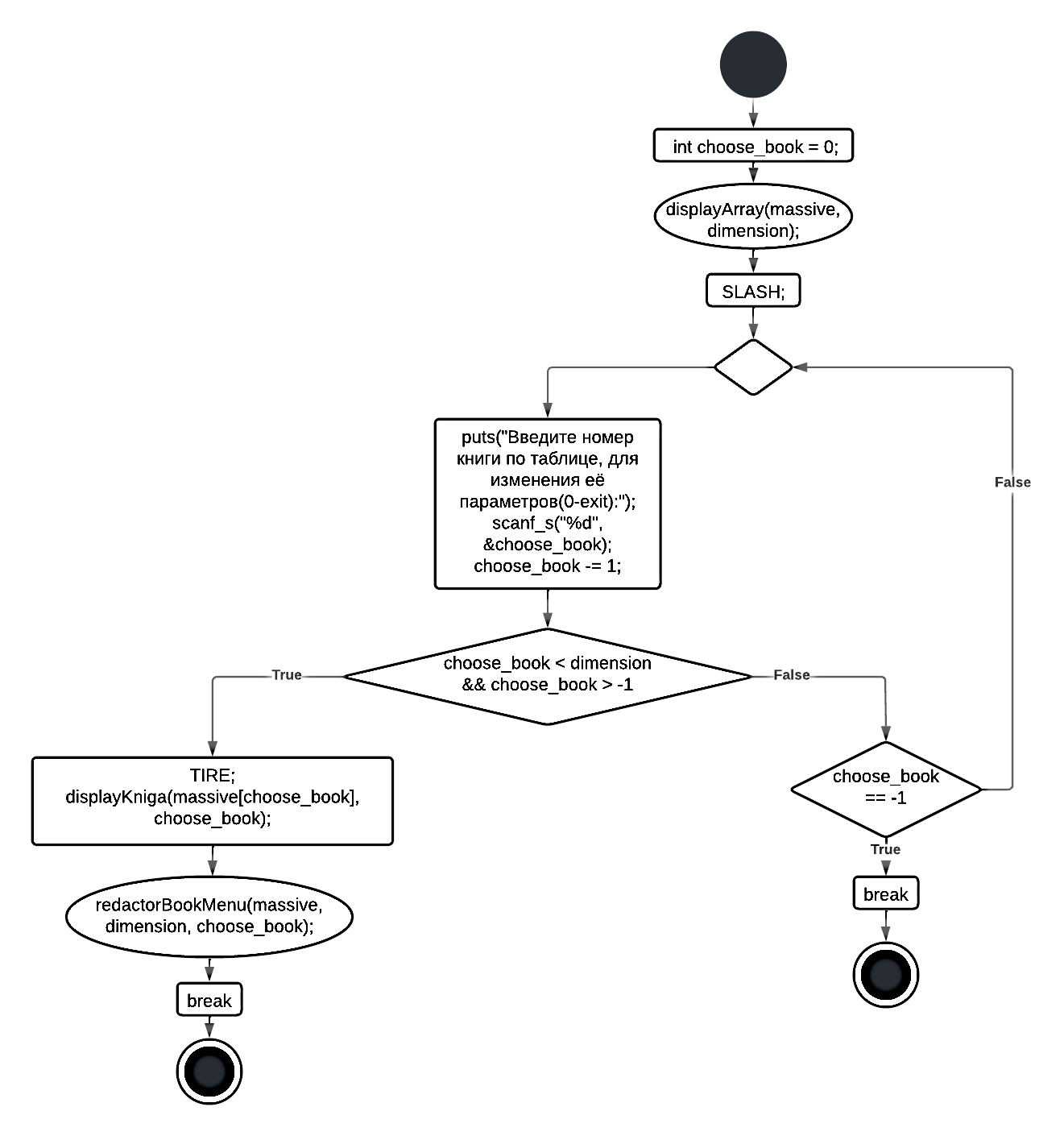


Рис 2.3. – Блок схема функції redactorBook()



### ***2.3.3 Функція redactorBookMenu(). Опис її алгоритму***

Після вибору книги, ми маємо можливість редагувати її параметри(рис.2.4).

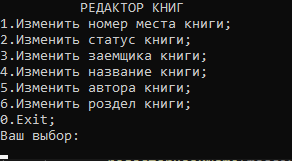


Рис 2.4. – Меню редагування

Для редагування доступні такі параметри: номер місця, статус, позичальник, назва, автор, розділ. Для вибору одного з цих параметрів треба вибирати одну з цифр, що приложені до Рис 2.4. Також, у блок-схемі функції **redactorBookMenu(),** що знаходиться у **додатку Г**, проілюстровано роботу функції.

### ***2.3.4 Функція redactorBookNumMesto (). Опис її алгоритму***

Функція займається редагуванням номера місця книги. Для цього потрібно увести новий номер й підтвердити зміну. Вона має цикл перевірки на введення. Уся логічна структура відображена на блок-схемі **redactorBookNumMesto ()** у **додатку Ґ.**



### ***2.3.5 Функція redactorBookStatusMenu (). Опис її алгоритму***

Функція слугує для виведення статусу знаходження книги. Алгоритм зображений на рисунку 2.5.

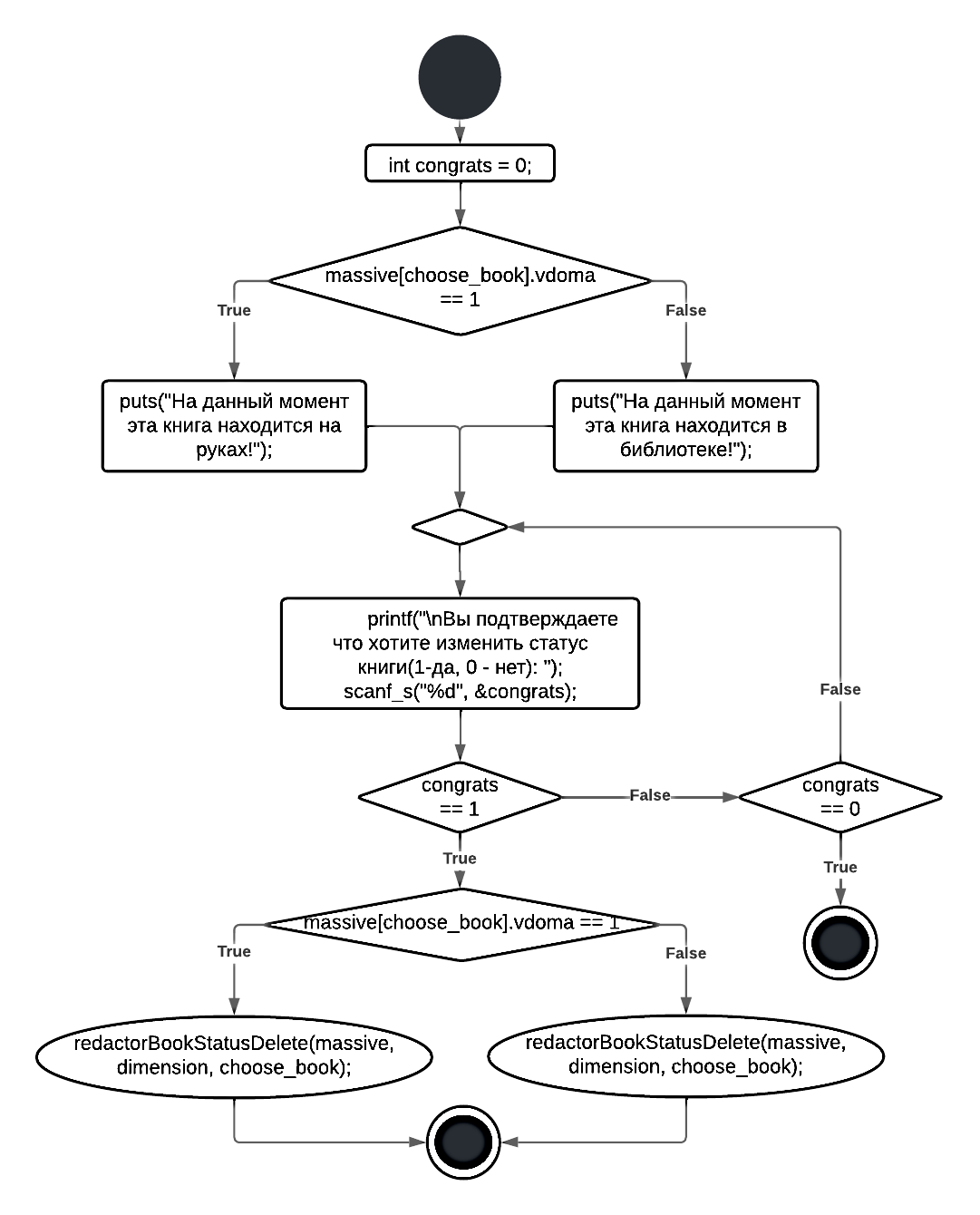


Рис 2.5. – Блок схема функції redactorBookStatusMesto()



### ***2.3.6 Функція redactorBookStatusAdd (). Опис її алгоритму***

Спочатку спрацювання цієї функції потрібно увести нове прізвище позичальника. Потім, після підтвердження йде опрацьовування даних, в наслідок чого змінюється статус на «vdoma = 1» та додається прізвище позичальника. На рис. 2.6 зображено блок-схему за допомогою мови UML. Вона ілюструє роботу функції **redactorBookStatusAdd().**

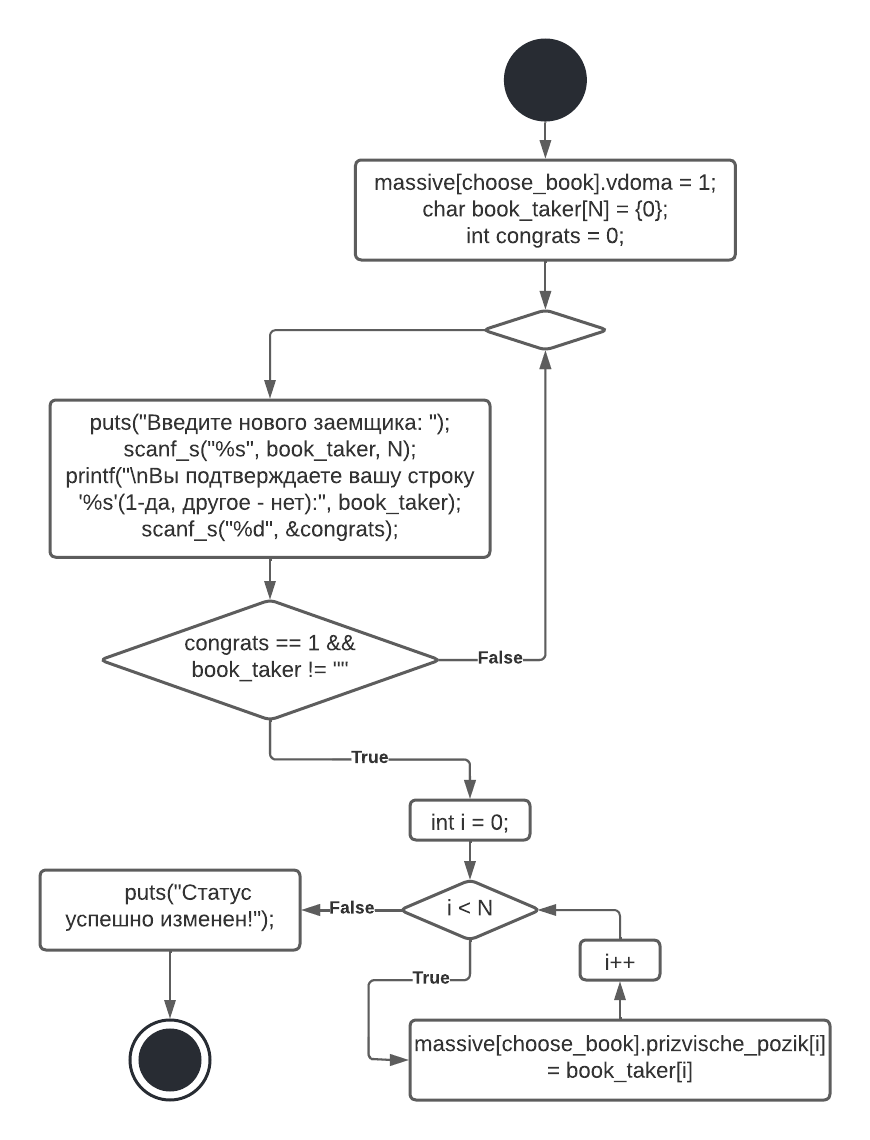


Рис 2.6. – Блок схема функції **redactorBookStatusAdd()**



### ***2.3.7 Функція redactorBookStatusDelete (). Опис її алгоритму***

Спочатку спрацювання цієї функції потрібно увести підтвердження повернення книги у бібліотеку. Потім, після підтвердження йде опрацьовування даних, в наслідок чого змінюється статус на «vdoma = 0» та на місці прізвища позичальника пишеться «-». На рис 2.7 зображено алгоритм функції **redactorBookStatusDelete()**

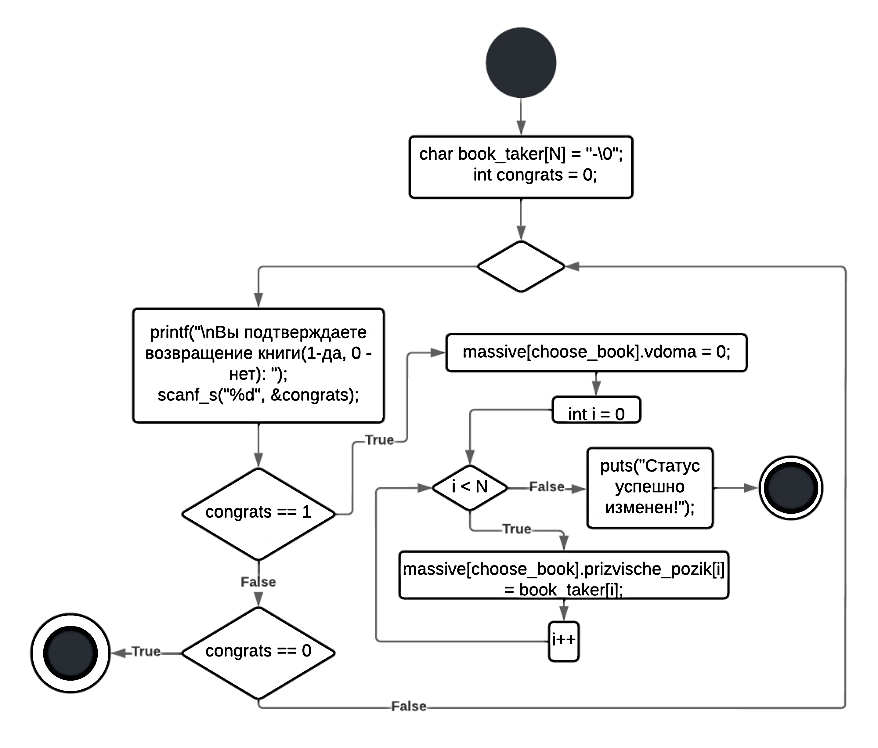


Рис 2.7. – Блок-схема функції **redactorBookStatusDelete()**



### ***Функція redactorBookStatusZaemshik (). Опис її алгоритму***

Спочатку спрацювання цієї функції алгоритм з’ясовує, де знаходиться обрана книга. Якщо книга знаходиться у бібліотеці, функція виводить на екран сповіщення про знаходження книги у бібліотеці. Якщо книга знаходиться на руках, то зміна позичальника стає можливою. Програма просить увести нового позичальника, а потім підтвердити його.

Алгоритм й блок-схема спрацьовування функції знаходиться у рис 2.8.

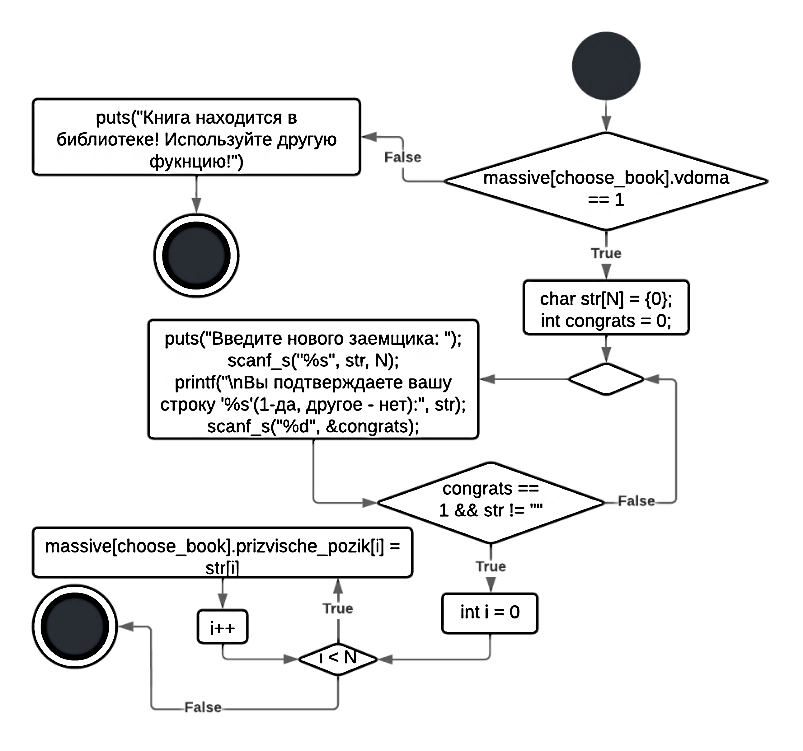


Рис 2.8. – Блок схема функції **redactorBookZaemshik()**



### ***2.3.9 Функція redactorBookName(). Опис її алгоритму***

Спочатку спрацювання цієї функції програма просить увести нову назву книги, а потім підтвердити зміни. Якщо назва не була введена або не було підтвердження, то назва не змінюється. Алгоритм дії функції зображено на рис 2.9.

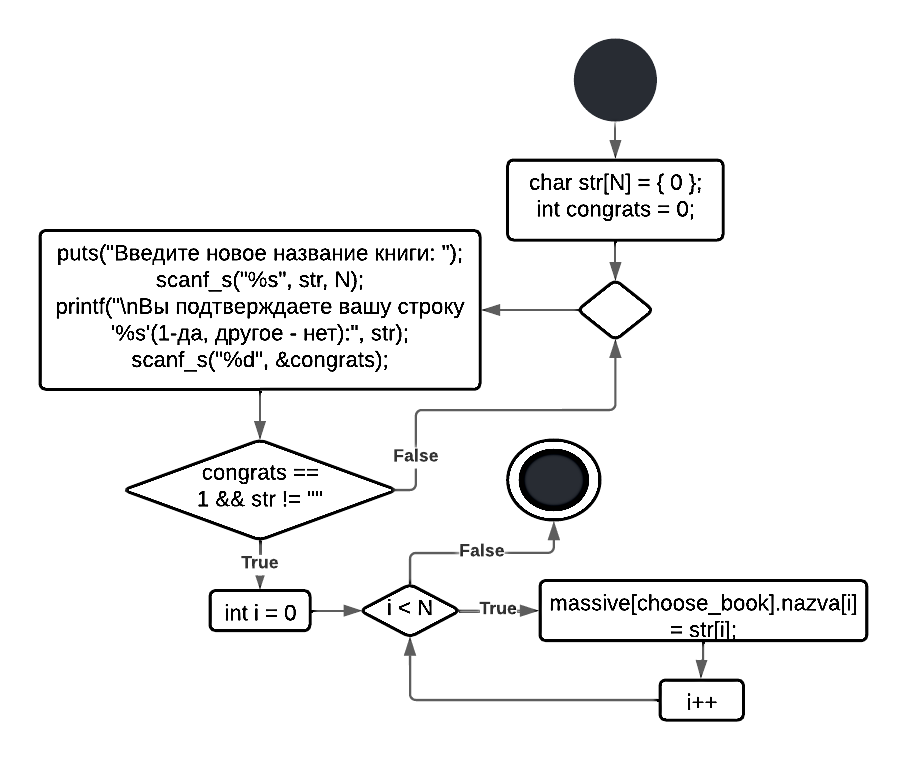


Рис 2.9. – Блок схема функції **redactorBookName** ()



### ***2.3.10 Функція redactorBookAutor(). Опис її алгоритму***

Спочатку спрацювання цієї функції програма просить увести нового автора книги, а потім підтвердити зміни. Якщо назва не була введена або не було підтвердження, то автор не змінюється. Алгоритм дії зображено на рис 2.10.

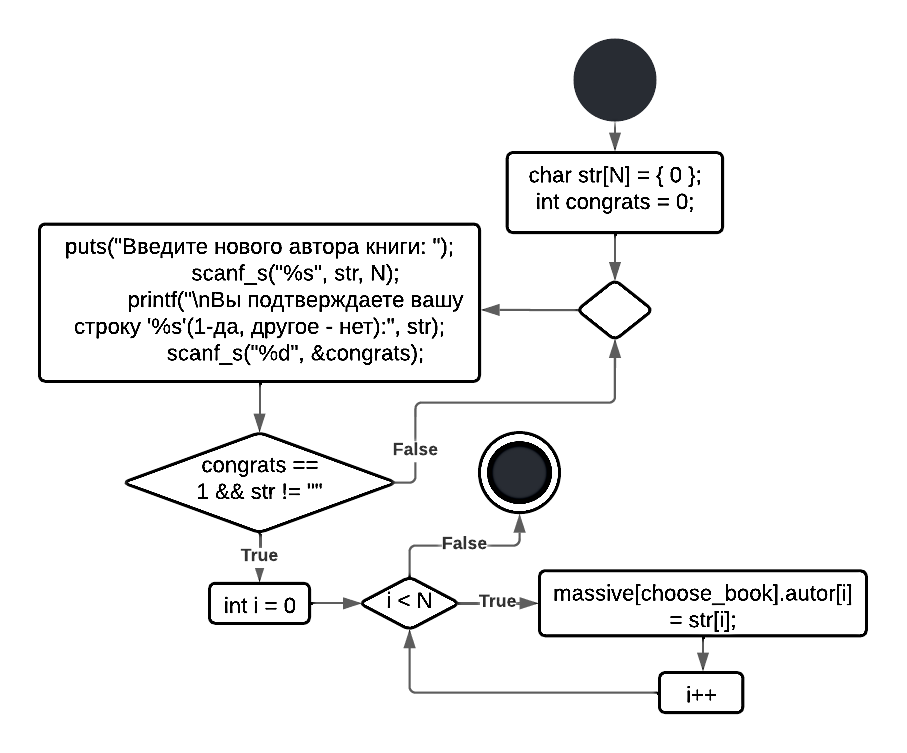


Рис 2.10. – Блок схема функції **redactorBookAutor ()**



### ***2.3.11 Функція redactorBookRozdil(). Опис її алгоритму***

Спочатку спрацювання цієї функції програма просить увести новий розділ книги, а потім підтвердити зміни. Якщо розділ не був введений або не було підтвердження, то розділ не змінюється. Більше подробиць на рис 2.11.

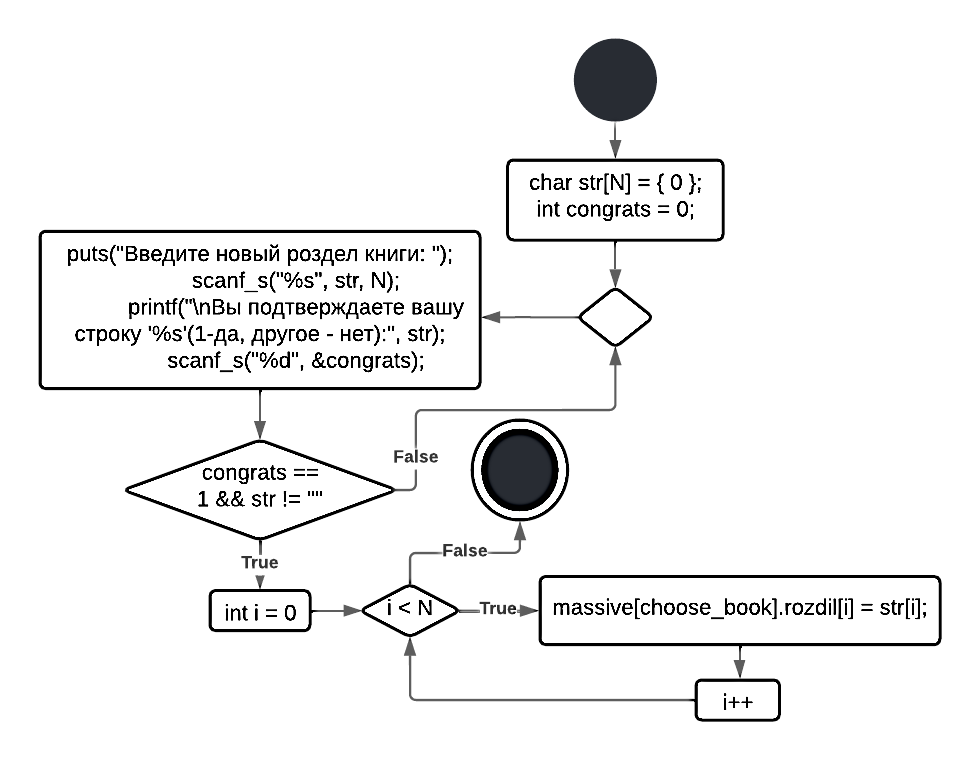


Рис 2.11. – Блок схема функції **redactorBookRozdil ()**



## ***Крок 3. Створення функцій initArray(), initArrayStart(), check(), initKniga(), initKnigaFileR()***



### ***2.4.1 Функція initArray(). Опис її алгоритму***

Данна функція створює масив за переданим числом dimension.

Потім, якщо створюваний масив дорівнює NULL, то функція повертає значення NULL.

Якщо створюваний масив не дорівнює нулю, то відкривається файл у режимі «читання».

Якщо файл не відкрився, програма завершується з кодом 1, а якщо відкрився, то запускається цикл, в якому функція зчитує кожну структуру даних я записує у масив.

Алгоритм дії функції повністю зображено на блок-схемі 2.12.

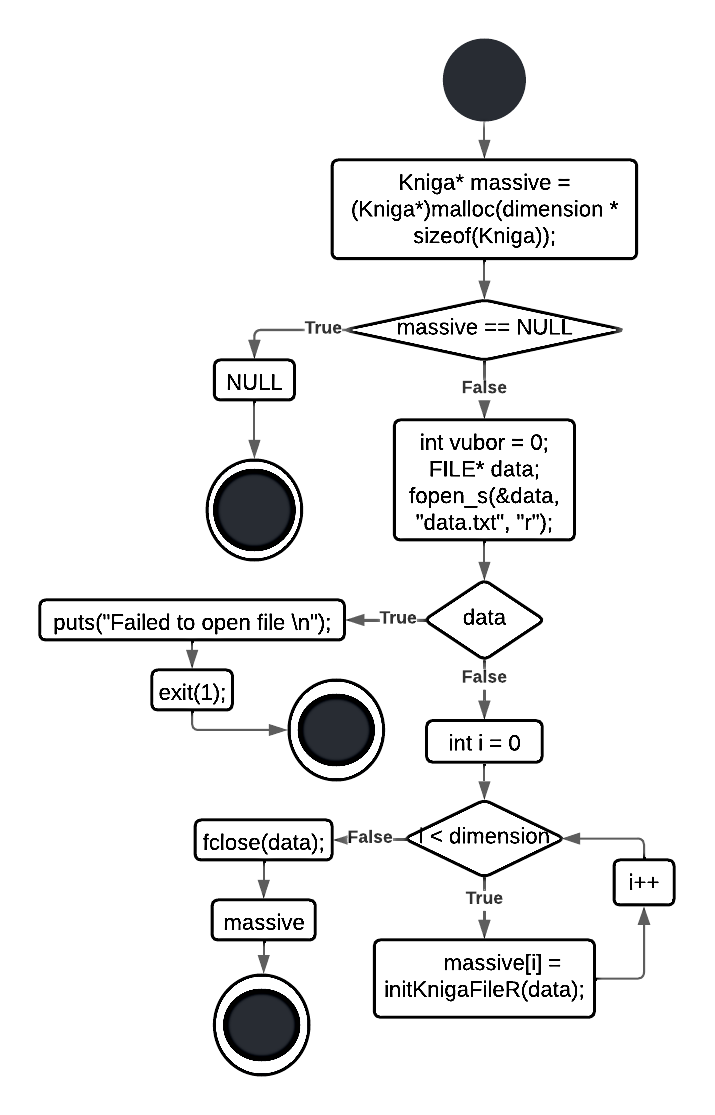


Рис 2.12. – Блок схема функції initArray ()

### ***2.4.2 Функція initArrayStart(). Опис її алгоритму***

Данна функція створює масив за переданим числом dimension. Потім, якщо створюваний масив дорівнює NULL, то функція повертає значення NULL. Якщо створюваний масив не дорівнює нулю, то запускається цикл, в якому функція зчитує кожну структуру даних, що створює user у консолі, записує у масив. Блок-схему представлено на рис 2.13.

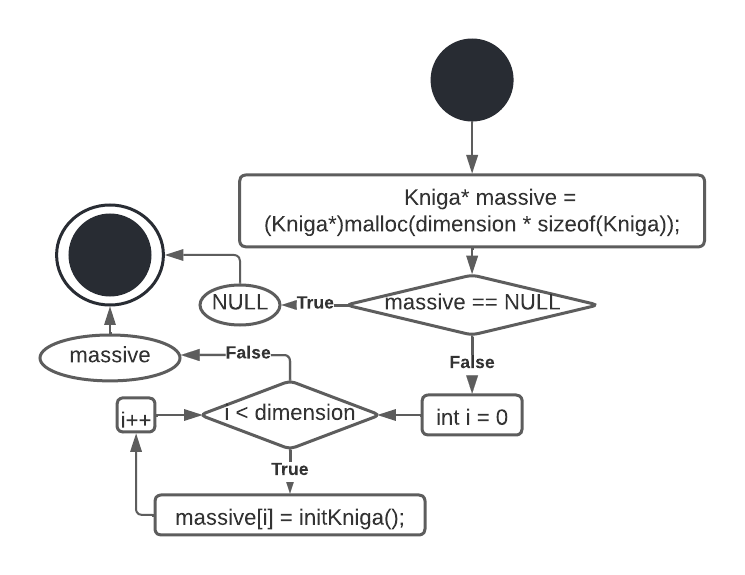


Рис 2.13. – Блок схема функції **initArrayStart ()**

### ***2.4.3 Функція check (). Опис її алгоритму***

Данна функція з початку спрацьовування відкриває файл з даними, а потім перевіряє, чи відкрився файл. Якщо файл не відкрився, то програма завершується з кодом 1. Якщо файл відкрився, то запускається цикл, в якому ціль програми зчитати номер місця = 0 та вдома 0. Якщо це відбувається, то файл закривається та з функції повертається значення знаходження «флажку». Алгоритм даної функції зображений на рис 2.14.

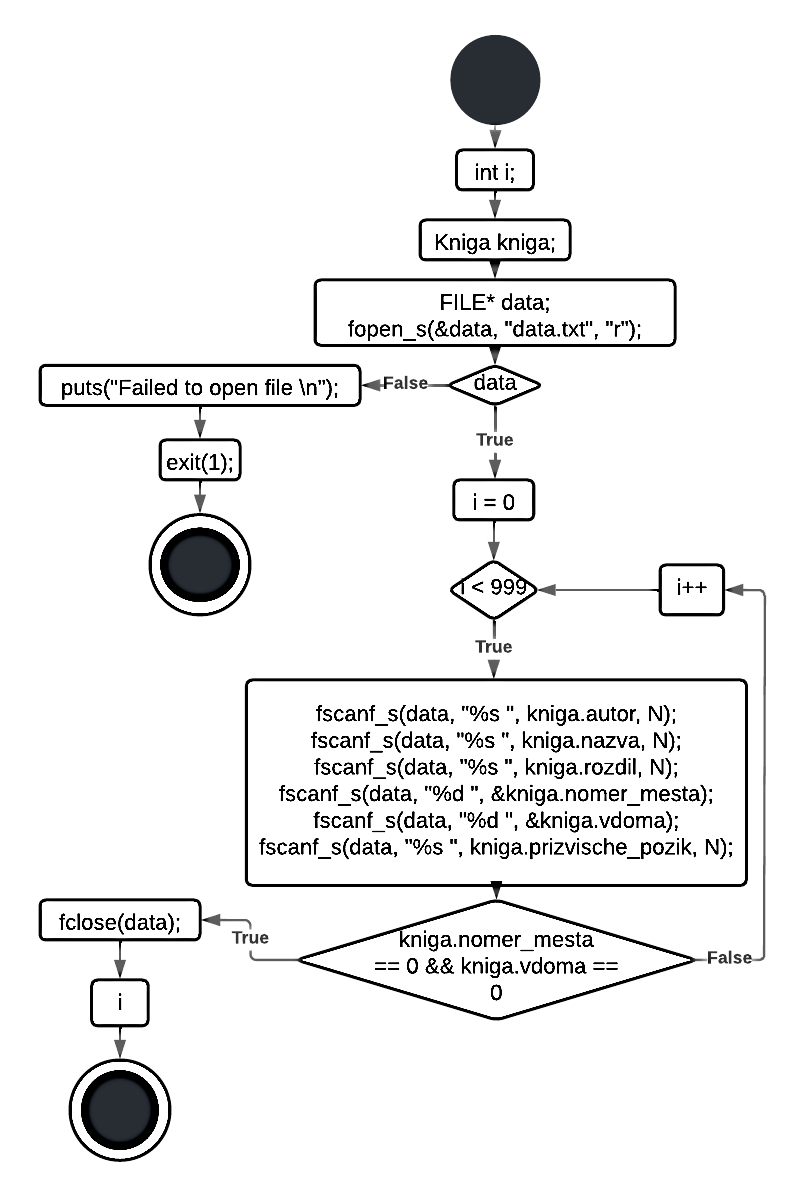


Рис 2.14. – Блок схема функції check ()

### ***2.4.4 Функція initKniga (). Опис її алгоритму***

Дана функція створює структуру книги. Потім, йде запис у цю структуру: автора, назви, розділу, номер місця. Потім, уводимо статус книги. Якщо статус дорівнює 1, то ми записуємо прізвище позичальника. Якщо статус дорівнює 0, то записується дана книга у бібліотеку. Алгоритм функції зображений на рис 2.15.

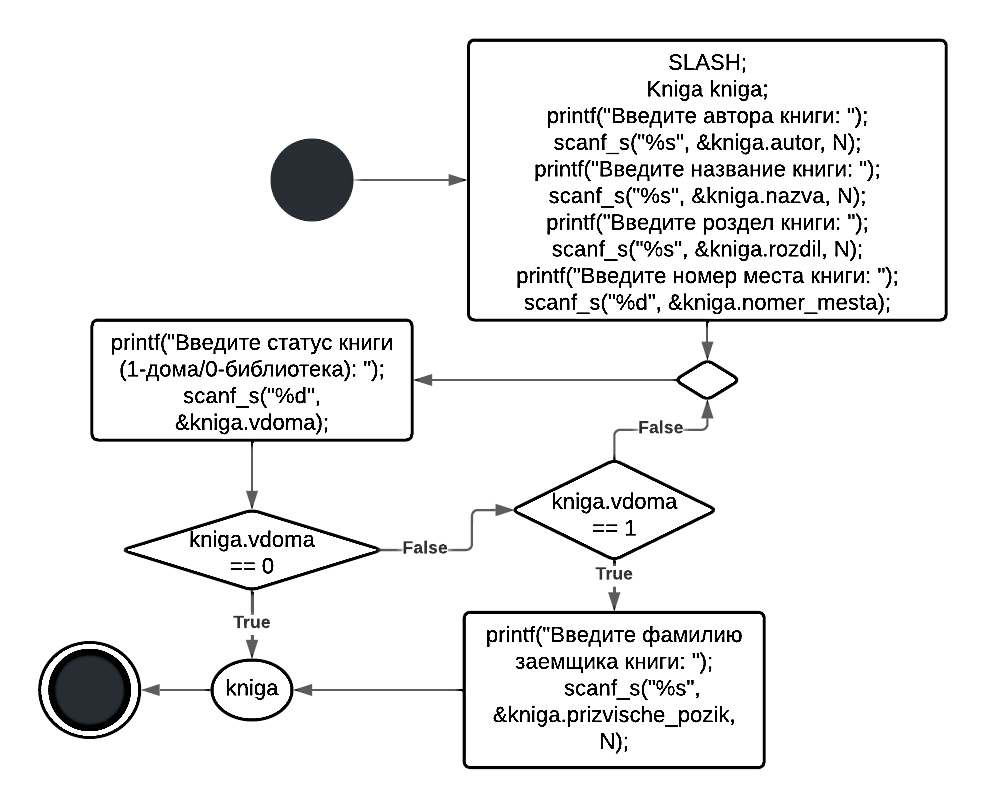


Рис 2.15. – Блок схема функції initKniga ()

### ***2.4.5 Функція initKnigaFileR (). Опис її алгоритму***

З початку роботи функції, створюється структура Kniga. Потім, данні зчитуються з файлу й записуються у структуру.

Функція повертає структуру. На рис 2.16 показана робота алгоритму.

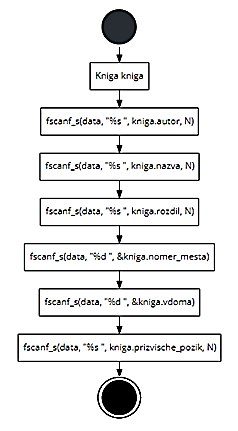


Рис 2.16. – Блок схема функції initKnigaFileR ()



## **Крок 4. Створення функцій initWrite(), write()**

### ***2.5.1 Функція initWrite (). Опис її алгоритму***

Спочатку, йде перевірка, у якому режимі працює функція.

Відповідно до переданого режиму – якщо 2 – то створюється новий файл для запису, якщо ні, то відкривається у режимі читання та змін старий файл.

Далі, якщо файл не відкрився, то виводиться відповідне сповіщення, про проблему з відкриттям, а також програма завершується з кодом 1.

Далі, коли файл був відкритий, запускається цикл з функцією запису у файл. Після завершення йде перевірка на режим.

Якщо він дорівнює 2, то у кінець файлу записується «флажок», а потім закривається файл й спрацьовує вихід з функції.

А якщо режим не дорівнює 2, то файл файл й спрацьовує вихід з функції. На рис 2.17 зображена блок-схема функції **initWrite ().**

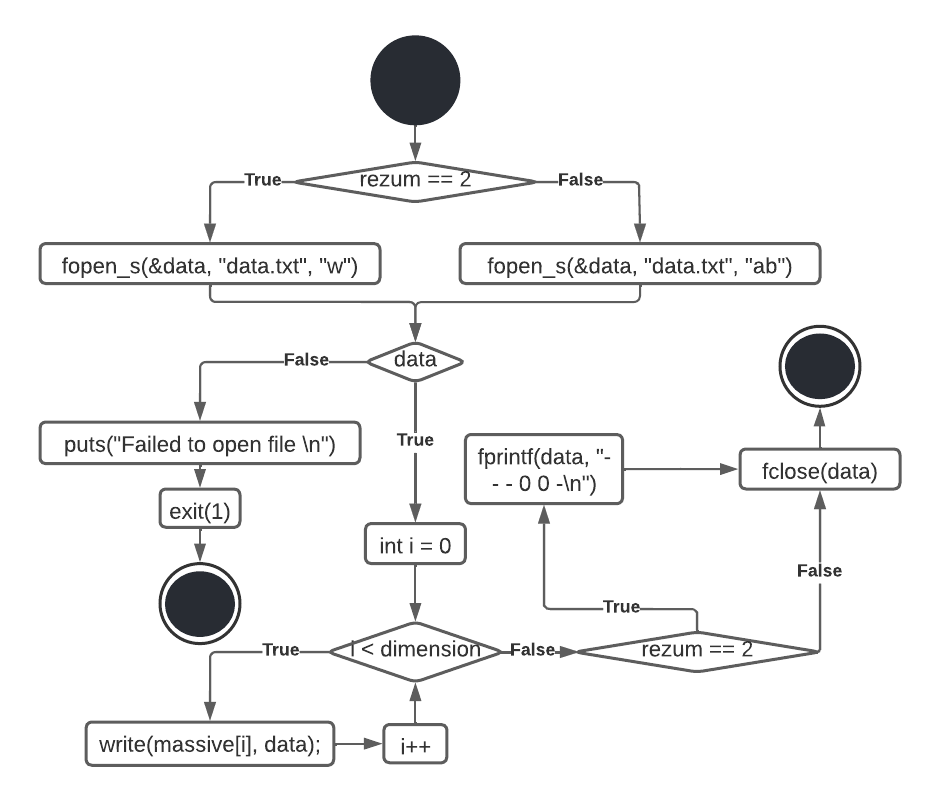


Рис 2.17. – Блок схема функції **initWrite ()**

### **2.5.2 Функція write (). Опис її алгоритму**

Спочатку роботи цієї функції, вже відкритий файл, що переданий в агргументі. Потім починається записування даних у файл. Після запису у файл, відповідно функція закінчує свою дію. Блок-схема наведена у Рис 2.18.

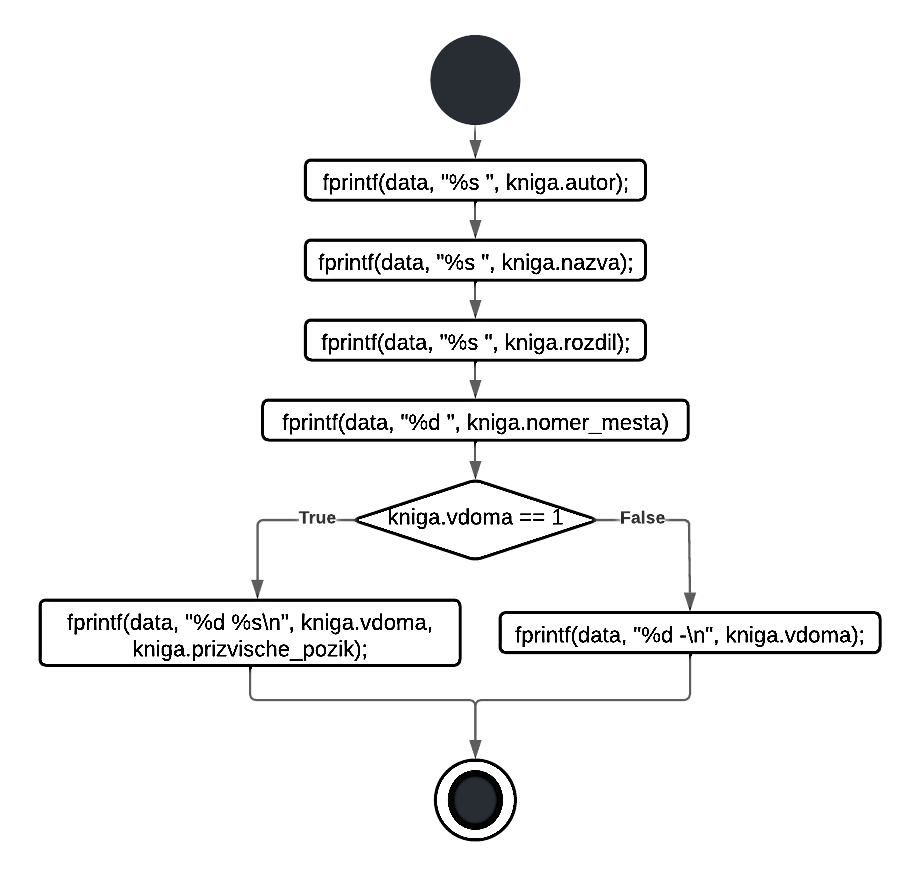


Рис 2.18. – Блок схема функції write()



## **Крок 5. Створення функцій addElement(), writeElementInFile ()**

### ***2.6.1 Функція addElement (). Опис її алгоритму***

При спрацьовуванні цієї функції створюється та відкривається новий файл(або БД).

Якщо файл не відкрився, то програма завершується кодом 1.

Якщо відкривається, то активується цикл з функцією запису масиву у файл.

Потім, заходимо у функцію запису нового елемента, а по виходу з неї завершуємо запис «флажком» й закриваємо файл.

Блок-схема до функції зображена на малюнку Рис 2.19.

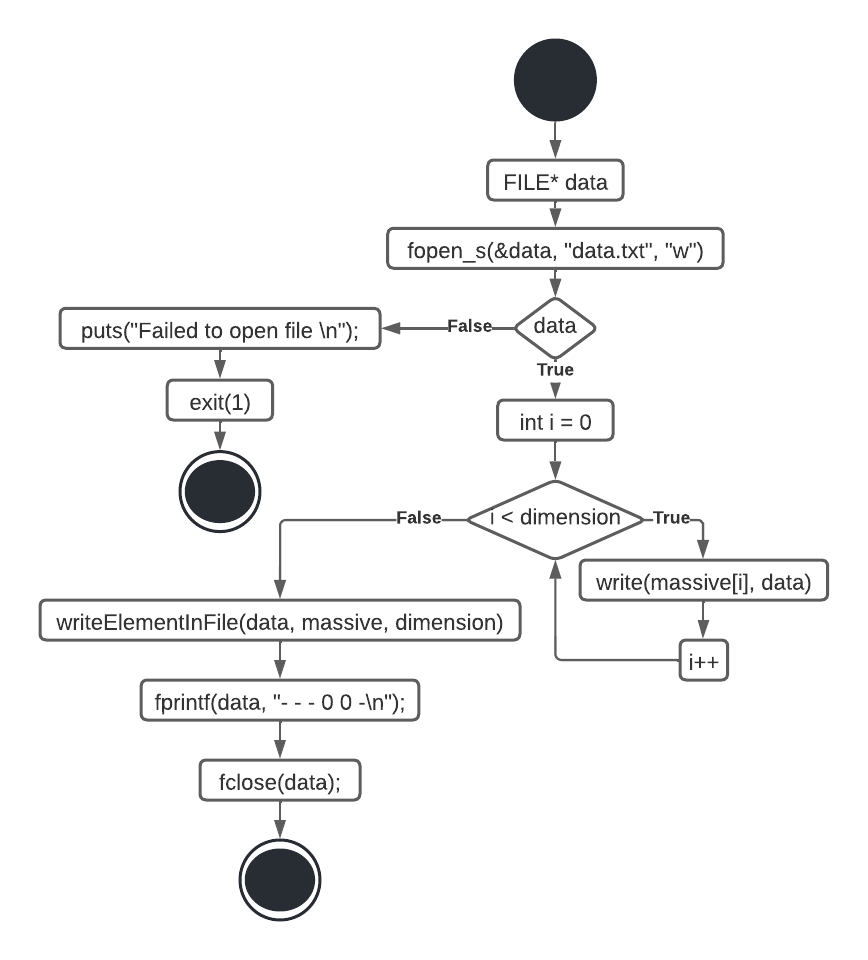


Рис 2.19. - Блок схема функції addElement()

### ***2.6.2 Функція writeElementInFile (). Опис її алгоритму***

У цій функції спочатку вводиться назва, автора, розділ. Потім запускається цикл з уведенням нового номеру. Потім, запускається цикл перевірки. Якщо місце занято, то програма просить увести по новому номер місця. Якщо номер місця не зайнятий, то уводиться статус книги. Якщо він прирівнюється до 0, то одразу починається друк даних у файл й завершення функції. Якщо прирівнюється до 1, то потрібно увести прізвище позичальника й потім йде друк у файл, завершення функції. Якщо число уведено неправильно, то друку немає. Блок-схема знаходиться у **Додатку Д.**



## **Крок 6. Створення функцій deleteElement (), deleteElementChoosed ()**

### ***2.7.1 Функція deleteElement (). Опис її алгоритму***

При спрацьовуванні функції виводяться книги, а потім йде вибір книги. Якщо увести 0, то спрацьовує вихід. В іншому випадку вибирається книга. Потім йде підтвердження вашого вибору. Після цього запускається функція видалення книги з масиву й кінець спрацьовування функції.

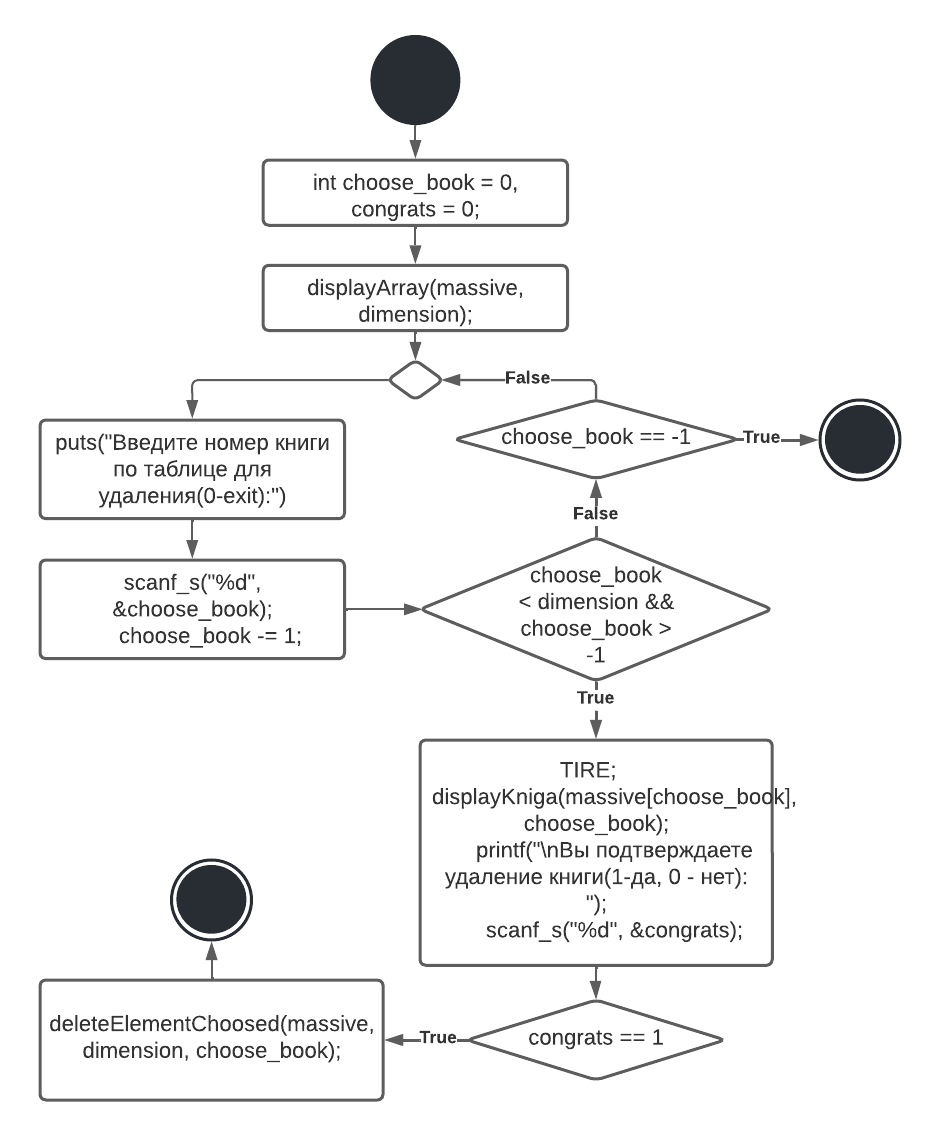


Рис 2.20. - Блок схема функції **deleteElement()**

### ***2.7.2 Функція deleteElementChoosed (). Опис її алгоритму***

При спрацьовуванні функції створюється і відкривається файл. Якщо файл не був відкритим, то програма завершується з кодом 1. Якщо файл відкрився, то починає роботу цикл, що вписує усі книги в файл, окрім вибранної, завдяки функції write(). Після завершення запису масива книг, записується «флажок» й файл закривається. Функція теж завершає свою роботу. Блок-схема зображена на Рис 2.21.

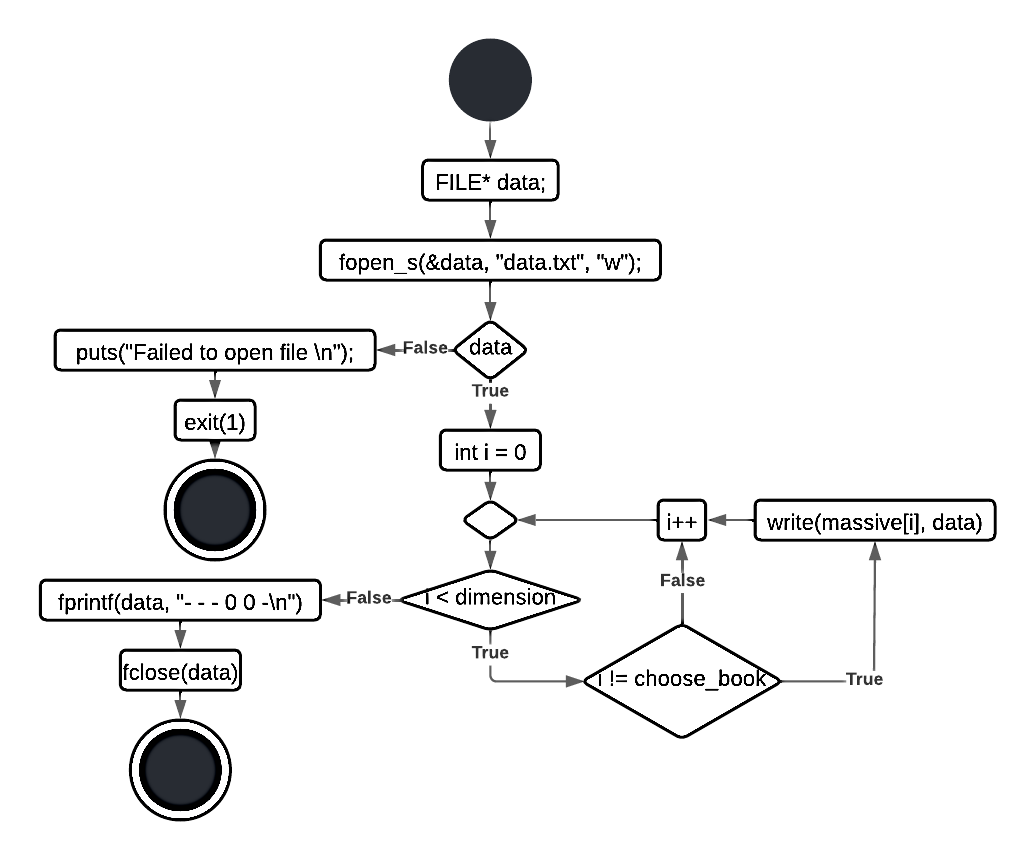


Рис 2.21. - Блок схема функції **deleteElementChoosed()**



## **Крок 7. Створення функцій displayArray (), displayKniga ()**

### ***2.8.1 Функція displayArray (). Опис її алгоритму***

При спрацьовуванні функції виводиться заголовок таблиці(рис 2.22):



Рис 2.22. – ілюстрація виводу функції **displayArray ()**

Також, потім запускається цикл з функцієї виводу усіх книг. У кінці всього виходить табличка.

Блок-схема знаходиться нижче на рис. 2.23.

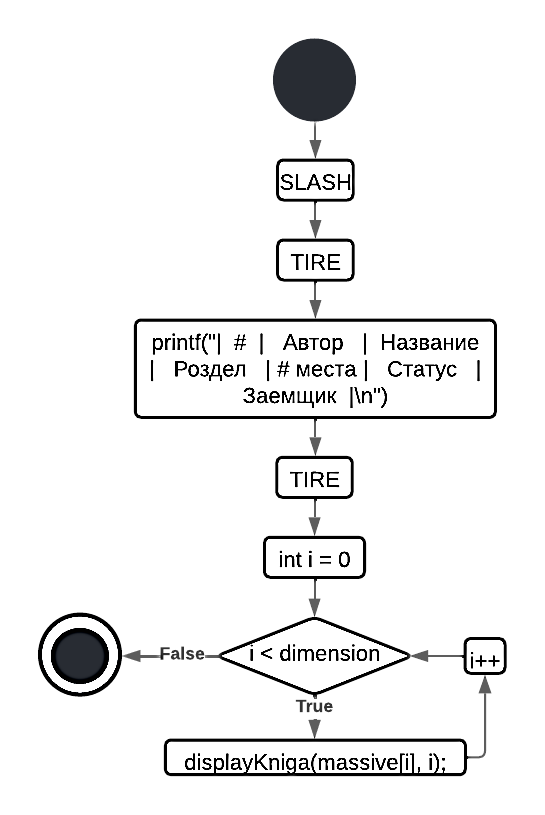


Рис 2.23. - Блок схема функції displayArray()

### ***2.8.2 Функція displayKniga (). Опис її алгоритму***

Функція виводить значення масиву. Блок схема зображена на малюнку 2.24.

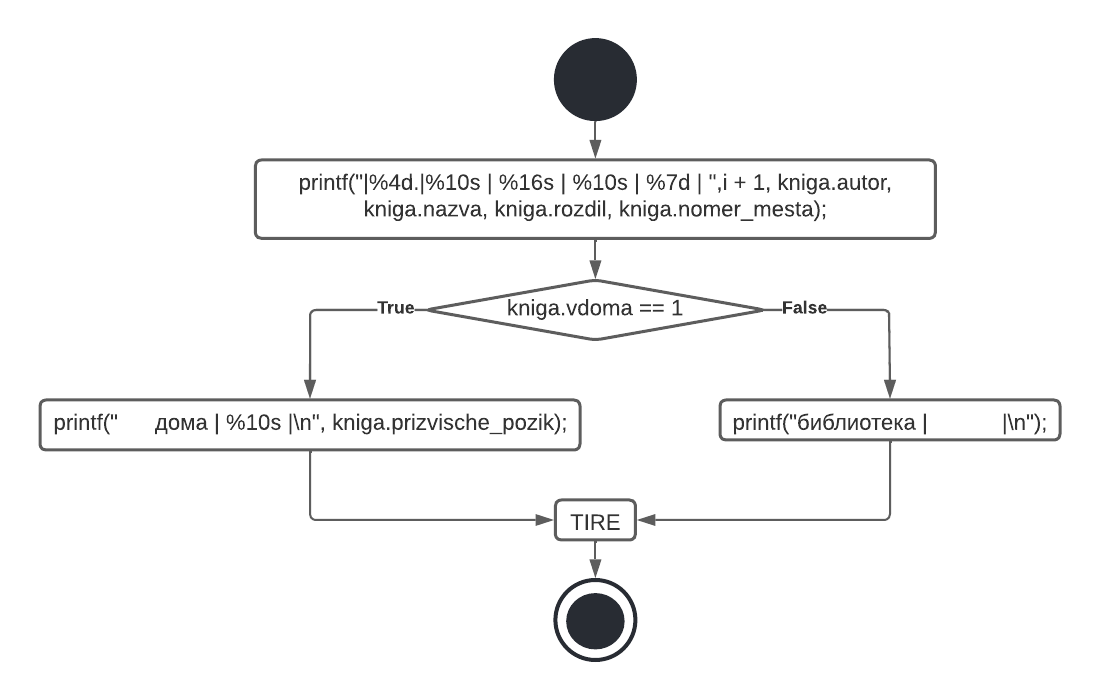


Рис 2.24. - Блок схема функції displayKniga()

## **Крок 8. Створення функцій sort(), sortNomerMesta(), sortName(), sortNazva()**

### ***2.9.1 Функція sort (). Опис її алгоритму***

Функція надає вибір для сортування:

* за назвою;
* за автора;
* за номером місця;

Для здійснення вибору потрібно увести значення з клавіатури.

Блок-схема зображена на Рис 2.25

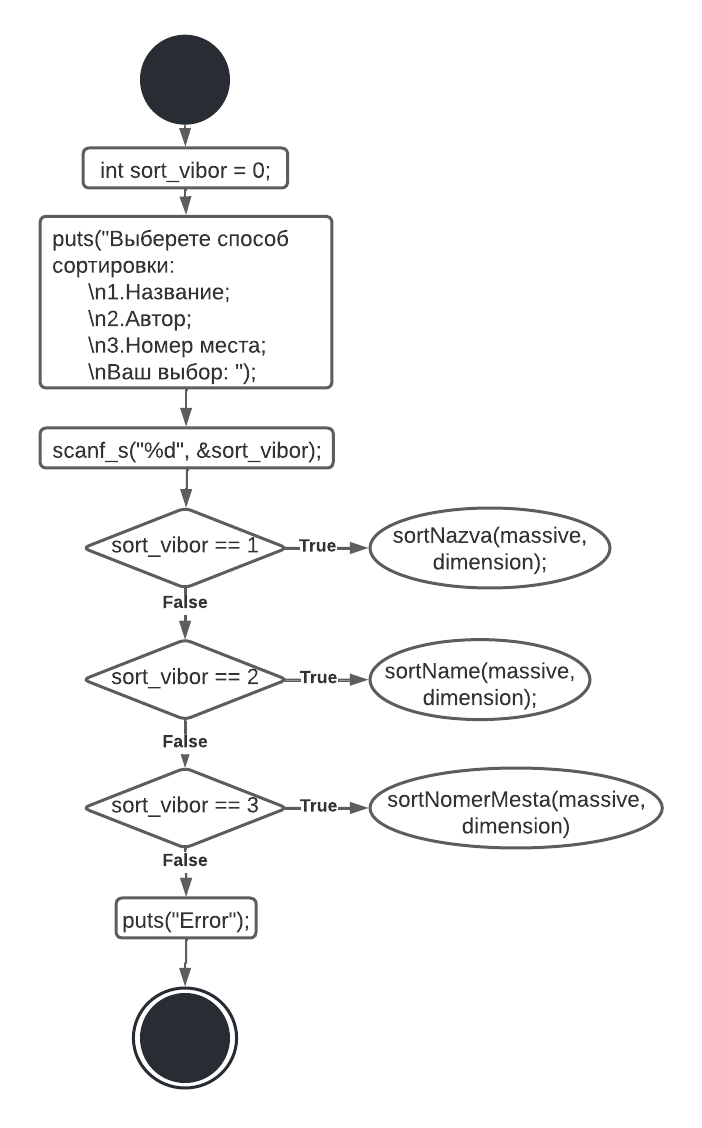


Рис 2.25. - Блок схема функції **sort ()**

### ***2.9.2 Функція sortNomerMesta (). Опис її алгоритму***

Функція виконує сортування за номером місця.

Блок-схема зображена на малюнку Рис. 2.26.

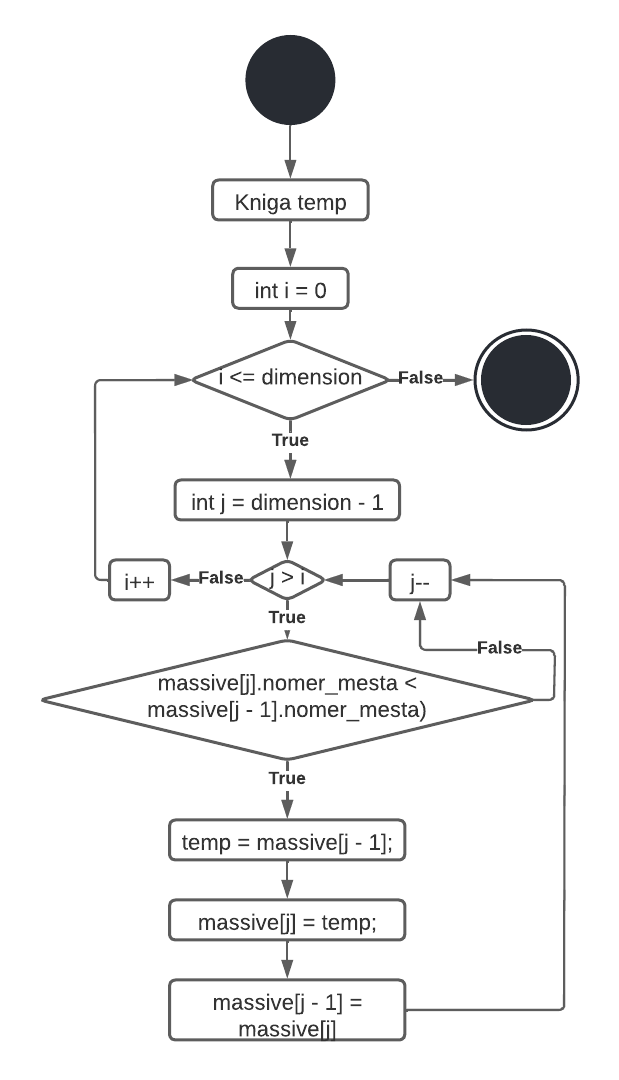


Рис 2.26. - Блок схема функції **sortNomerMesta ()**

### ***2.9.3 Функція sortName (). Опис її алгоритму***

Функція виконує сортування за автором.

Блок-схема зображена на малюнку Рис. 2.27.

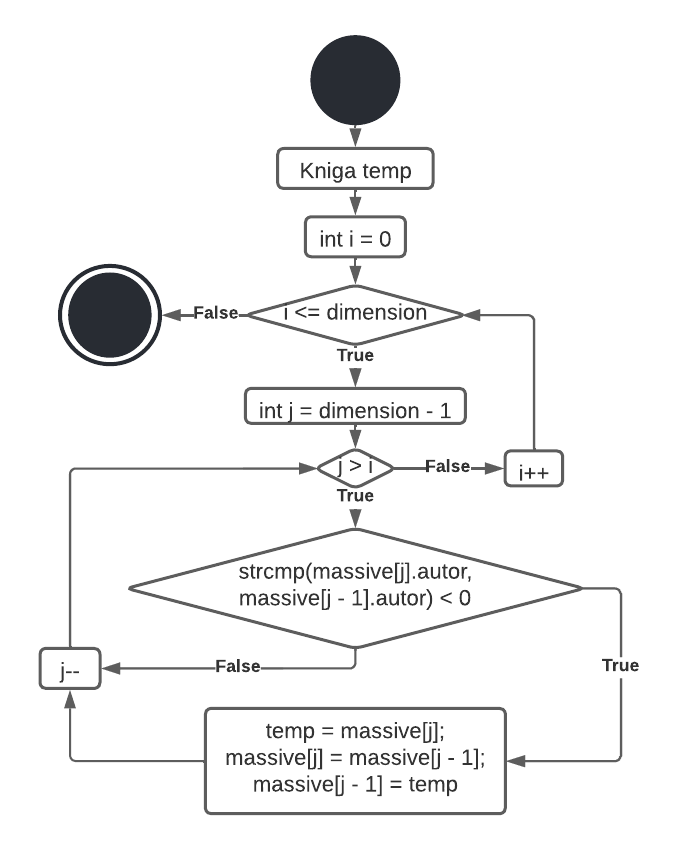


Рис 2.27. - Блок схема функції **sortName ()**

### ***2.9.4 Функція sortNazva (). Опис її алгоритму***

Функція виконує сортування за назвою.

Блок-схема функції зображена на Рис. 2.28.



Рис 2.28. - Блок схема функції sortNazva()



## **Крок 9. Створення функцій displayMenu (), displayСonsensus (), displayChoise (), displayChoiseRozdil (), displayChoiseAutor (), displayChoiseNazva ()**

### ***2.10.1 Функція displayChoise (). Опис її алгоритму***

При спрацьовуванні цієї функції спочатку починається запит на вибір пунктів для пошуку. Після вибору користувачем здійснюється розтасування пунктів у визначенні змінні завдяки циклу.

Потім, починає свою роботу цикл з введенням обраного пункту у консоль. Далі, починає свою роботу цикл з case, де йде перевірка. Якщо нічого не знайдено, то програма повідомить про це. Якщо щось знайде, то масив виведеться на екран.

Блок-схема функції знаходиться у **Додатку Е.**

### ***2.10.2 Функція displayMenu (). Опис її алгоритму***

При активації цієї функції починає роботу цикл, що відповідає за вибір пошука. При виборі 1, ми заходимо до докладного пошуку, а при 2 – збіговий.

Блок-схема функції знаходиться у зображена на Рис 2.29.

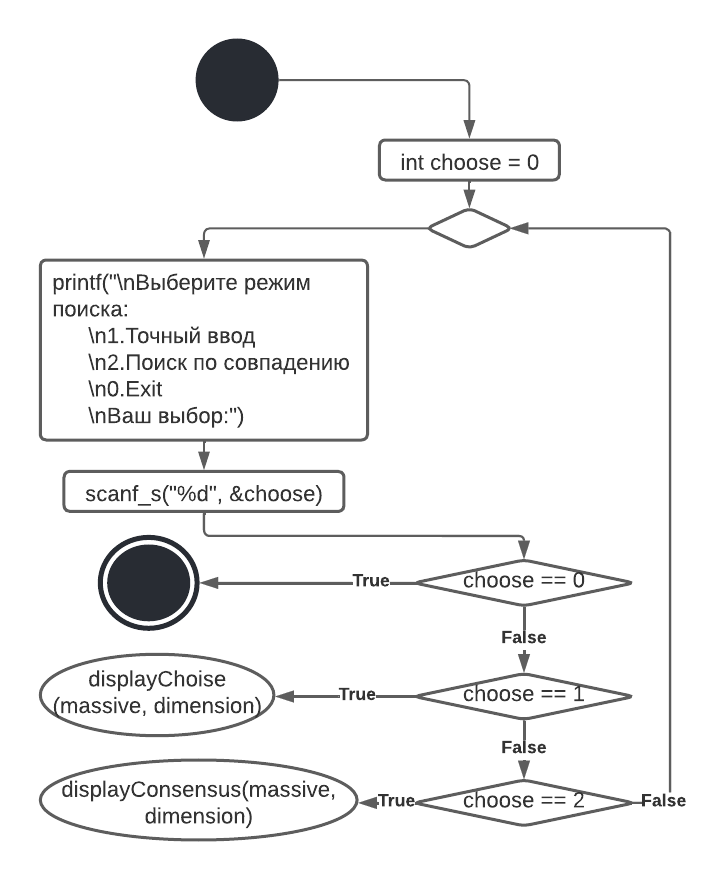


Рис 2.29. - Блок схема функції **displayMenu ()**

### ***2.10.3 Функція displayСonsensus (). Опис її алгоритму***

Спочатку пошуку по першому збігу, потрібно увести ключове слово. Потім, йде перебирання на знаходження подібного. Після цього, якщо знайдено збіг, то виводиться ця книга зі збігом. Блок-схема знаходиться у **Додатку Є.**

### ***2.10.4 Функція displayChoiseRozdil (). Опис її алгоритму***

При активації функції йде на пошук на збіг. Якщо збігу не має, то повертається значення 0. В іншому разі, повертається значення 1.

Блок-схема зображена на Рис 2.30.

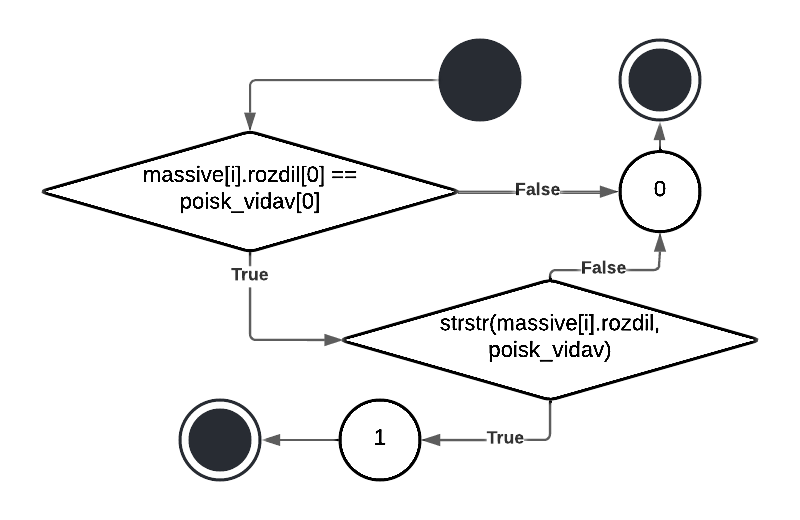


Рис 2.30. - Блок схема функції displayChoiseRozdil()

### ***2.10.5 Функція displayChoiseAutor (). Опис її алгоритму***

При активації функції йде на пошук на збіг (рис. 2.36, лістинг 2.10). Якщо збігу не має, то повертається значення 0. В іншому разі, повертається значення 1.

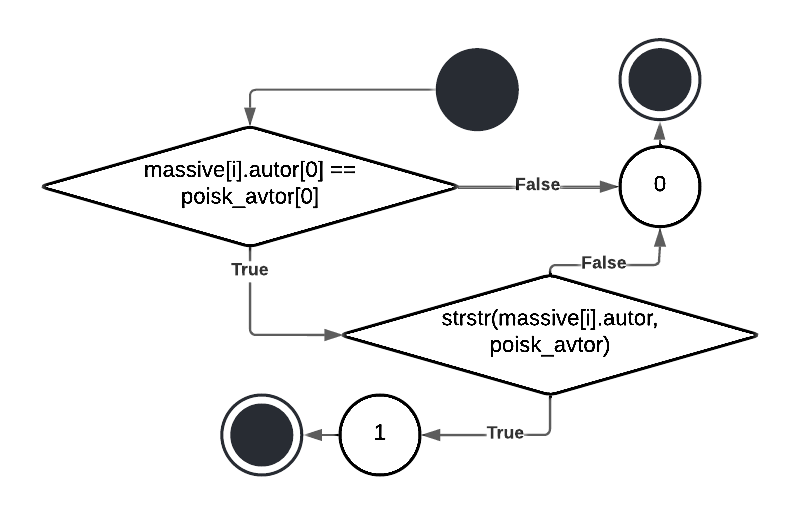


Рис 2.31. - Блок схема функції displayChoiseAutor()

***2.10.6 Функція displayChoiseNazva (). Опис її алгоритму***

При активації функції йде на пошук на збіг. Якщо збігу не має, то повертається значення 0. В іншому разі, повертається значення 1.

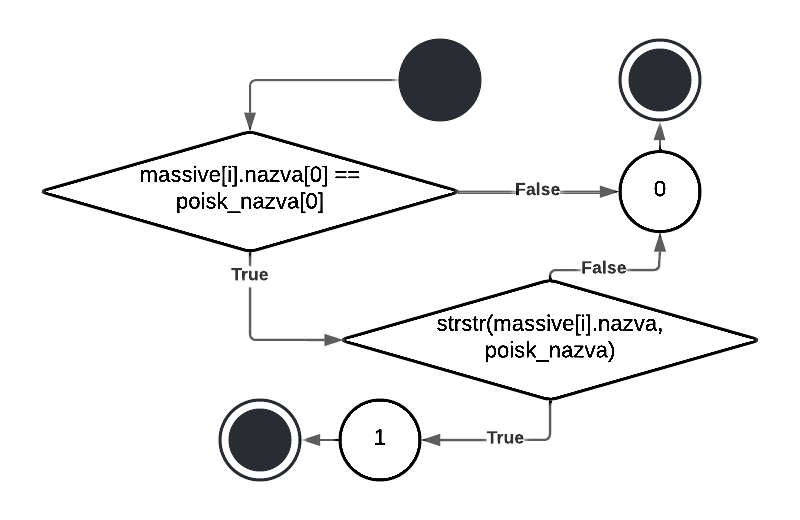


Рис 2.37. - Блок схема функції displayChoiseNazva()

1. **ТЕСТУВАННЯ**

## **3.1 Крок 1. Тестування меню програми(функції main)**

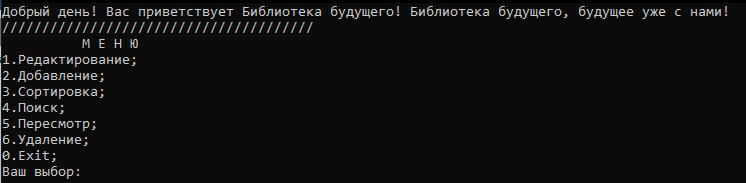


Рис. 3.1.1 – Меню програми

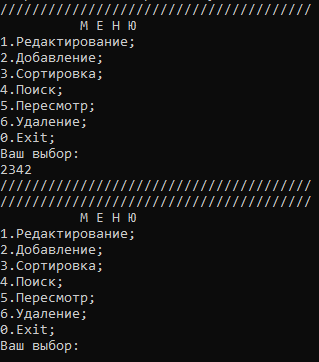


Рис. 3.1.2 – Результати тестування при введені некоректного пункту меню

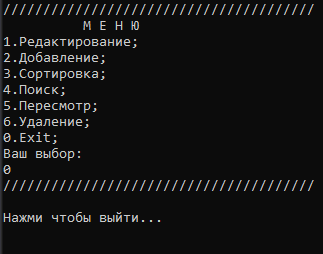


Рис. 3.1.3 – Результати тестування при введені 0 пункту меню

При введені одного з пунктів вибору, здійснюється перехід до цієї гілки програми. *Результати тестування гілок наведений нижче.*



## **3.2 Крок 2. Тестування гілки редагування**

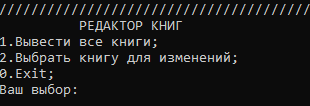


Рис. 3.2.1 – Меню гілки редагування

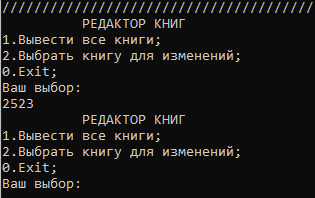


Рис. 3.2.2 – Результати тестування при введені некоректного пункту меню

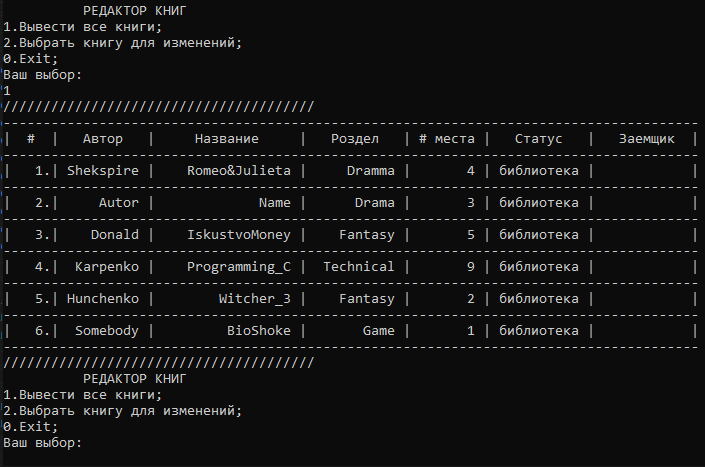


Рис. 3.2.3 – Результати тестування при виборі 1(«Вивести книги»)

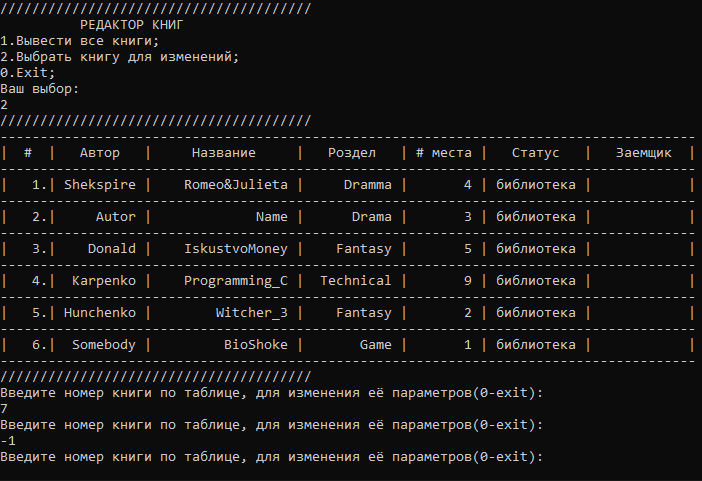


Рис. 3.2.3 – Результати тестування при виборі 2(«Вибір книги для змін»)

й обробка некоректного вибору книги

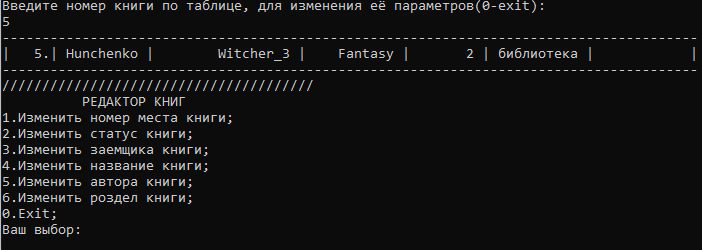


Рис. 3.2.4 – Результати тестування при виборі існуючої книги

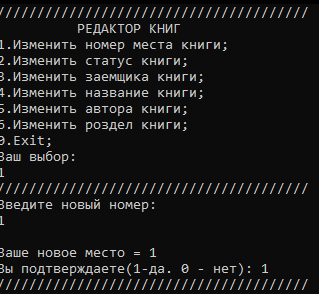


Рис. 3.2.4 а– Результати тестування при виборі зайнятого місця книги

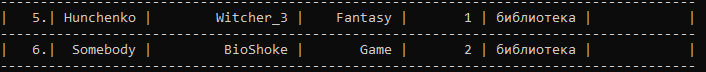


Рис. 3.2.4 б – Результати тестування при виборі зайнятого місця книги

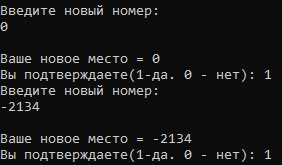


Рис. 3.2.4 б – Результати тестування при виборі некоректного місця книги



Рис. 3.2.5 – Результати тестування при виборі некоректного статусу книги

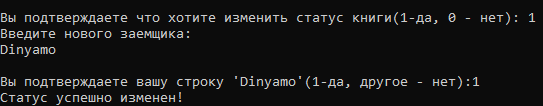


Рис. 3.2.6 а – Результати тестування при виборі коректного статусу книги



Рис. 3.2.6 б – Результати тестування при виборі коректного статусу книги

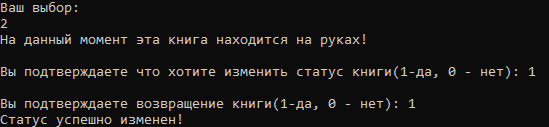


Рис. 3.2.7 а – Результати тестування при виборі коректного статусу книги



Рис. 3.2.7 б – Результати тестування при виборі коректного статусу книги

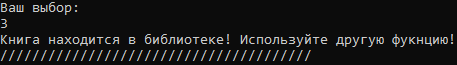


Рис. 3.2.8 – Результати тестування при спробі зміни позичальника при знаходженні книги у бібліотеці

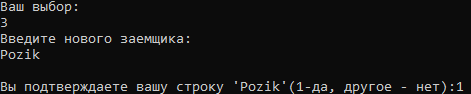


Рис. 3.2.9 а – Результати тестування змінa позичальника



Рис. 3.2.9 б – Результати тестування змінa позичальника

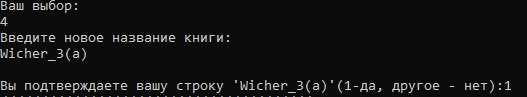


Рис. 3.2.10 a – Результати тестування змінa назви



Рис. 3.2.10 б – Результати тестування змінa назви

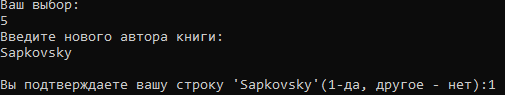


Рис. 3.2.11 a – Результати тестування змінa автора



Рис. 3.2.11 б – Результати тестування змінa автора

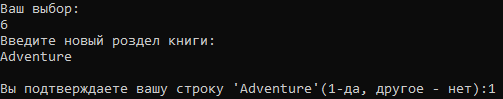


Рис. 3.2.12 a – Результати тестування змінa розділу книги



Рис. 3.2.12 б – Результати тестування змінa розділу книги

Гілка редагування працює стабільно.



## **3.3 Крок 3. Тестування гілки створення та додавання нової книги**

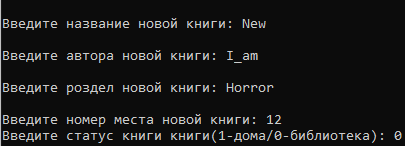


Рис. 3.3.1 а – Результати тестування коректного додавання книги



Рис. 3.3.1 б – Результати тестування коректного додавання книги

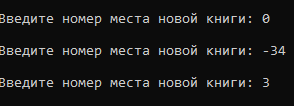


Рис. 3.3.2 – Результати тестування некоректного додавання книги(уведено номер місця: неможливе/зайняте місце)



## **3.4 Крок 4. Тестування гілки сортування**

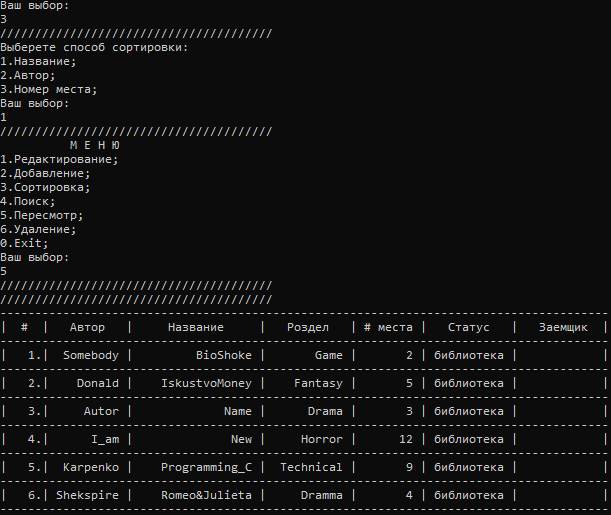


Рис. 3.4.1 – Результати тестування коректного сортування за назвою

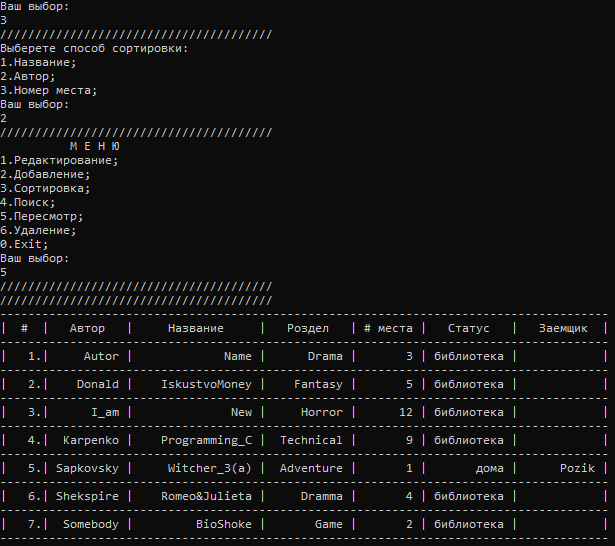


Рис. 3.4.2 – Результати тестування коректного сортування за автором

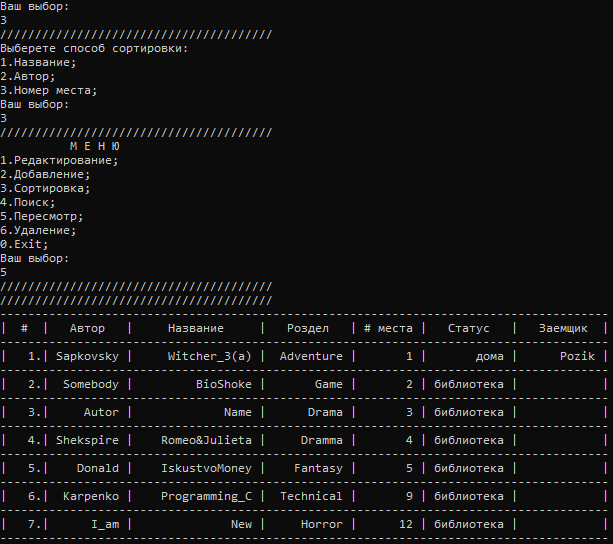


Рис. 3.4.3 – Результати тестування коректного сортування за автором



## **3.5 Крок 5. Тестування гілки видалення**

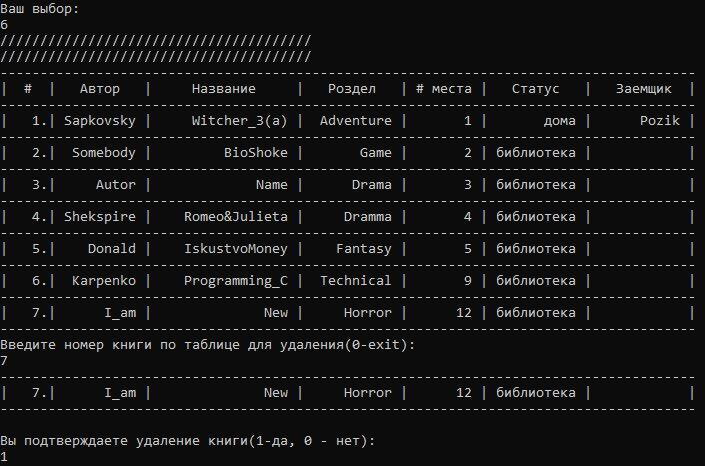


Рис. 3.5.1 а – Результати тестування коректного видалення

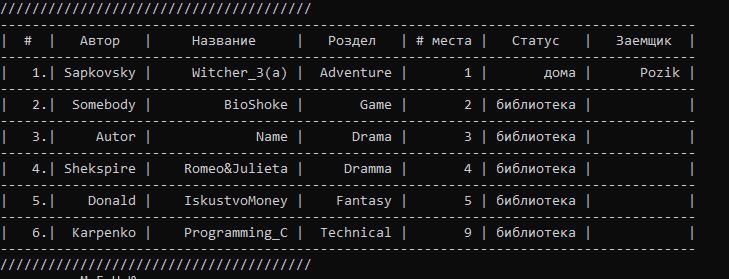


Рис. 3.5.1 б – Результати тестування коректного видалення

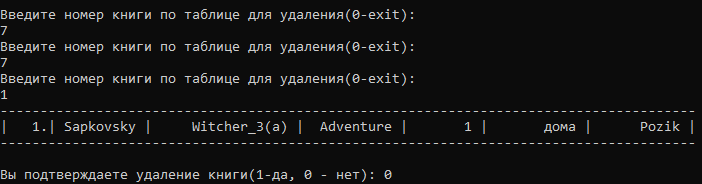


Рис. 3.5.2 а – Результати тестування гілки видалення. Обробка тестових ситуацій. Введення некоректного номеру книги або помилка при виборі книги.



## **3.6 Крок 6. Тестування гілки пошуку**

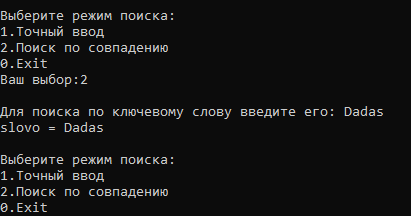


Рис. 3.6.1– Результати тестування некоректного пошуку по збігу

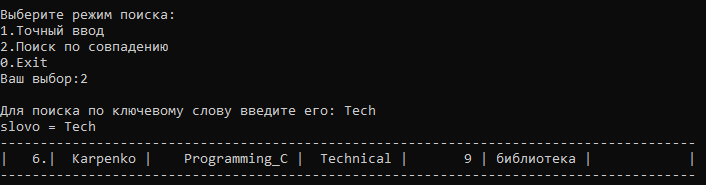


Рис. 3.6.2 – Результати тестування коректного пошуку по збігу

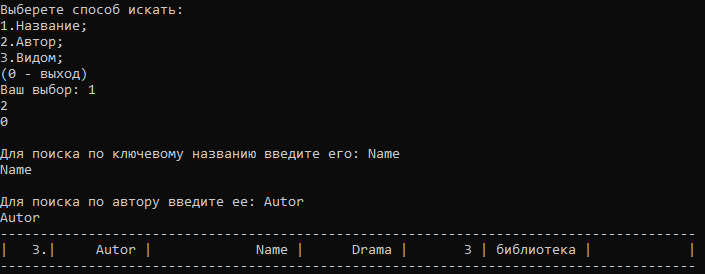


Рис. 3.6.3 – Результати тестування коректного пошуку по «точному введенню»

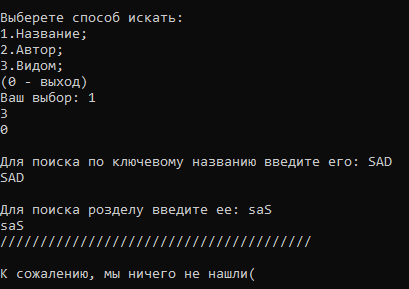


Рис. 3.6.3 – Результати тестування некоректного пошуку по «точному введенню»

# **ВИСНОВОК**

Під час виконання курсової роботи ми створили інформаційну систему. Ми на практиці пройшли всі етапи розробки програмного забезпечення, починаючи від збору та аналізу вимог, закінчуючи тестуванням програми. Програма «Домашня бібліотека» була розроблена на мові С.

При було пройдено при створенні програми такі 5 етапів:

* + - 1. Збір вимог;
      2. Аналіз вимог;
      3. Моделювання;
      4. Кодування;
      5. Тестування;

При зборі та аналізу вимог ми сформували список завдань бізнесу. Кожна з даних вимог була реалізована в наступних пунктах.

При моделюванні функцій, їх блок-схем використовувалася мова UML. Блок-схеми ілюструють роботу системи. Також, був сформований прецендент, у якому була сформована ілюстрація роботи функції редагування.

Під час кодування ми зустрілися з низкою проблем, що потім ми успішно вирішили. Також, під час написання коду для уникнення помилок було створено низку перевірок, що роблять систему більш стабільною та прийнятною для користувача.

Далі, при тестуванні, ми обробили усі виключні випадки й перевірили тестові випадки. Була сформована табличка з обробкою тестових випадків.

У результаті була розроблена програма «Домашня бібліотека».

# **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

**1.** Язык программирования [Електронний ресурс]. Режим до-ступу: https://ru.wikipedia.org/wiki/Язык\_программирования — Загол. з екрана.

**2**. Наталья Березовская. Cat, Mouse и Brainfuck: сколько язы-ков программирования существует в мире. [Електронний ресурс]. Режим доступу: https://skillbox.ru/media/code/cat\_mouse\_i\_brainfuck/ — Загол. з екрана.

**3**. Литвинов О. А., Герасимов В. В., Карпенко Н. В. Об’єктно-орієнтована розробка інформаційних систем. Монографія. / О. А. Литвинов, В. В. Герасимов, Н. В. Карпенко – Д.: Ліра. – 2018. –448 с.

**4**. Programming languages — C. ISO/IEC 9899:2017 [Електрон-ний ресурс]. Режим доступу: https://web.archive.org/web/20181230041359if\_/http://www.open-std.org/jtc1/sc22/wg14/www/abq/c17\_updated\_proposed\_fdis.pdf — Загол. з екрана.

**5**. Методичні вказівки до виконання курсових робіт з дисципліни «Програмування» / Н. В. Карпенко, В. В. Герасимов — Д.: Ліра, 2020. — 52 с.

**6**. Карпенко Н. В., Герасимов В. В. Сучасний підхід до програмування на мові С від нульового до просунутого рівня : навч. посіб. / Н. В. Карпенко, В. В. Герасимов – Д.: Ліра, 2022. – 340 с.

# **ДОДАТКИ**

**Додаток А**

*Лістинг «sourse.h»*

#pragma once

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <locale.h>

#define SLASH puts("///////////////////////////////////////")

#define TIRE puts("---------------------------------------------------------------------------------------")

#define N 30

#define E 7

/\*

Домашня бібліотека.

//Є інформація:

//автора книги

//назву книги

//розділ, до якого належить література даного виду(фантастика, пригоди, наукова, кулінарія тощо)

//місце, де стоїть книга

//чи знаходиться книга вдома.

//Якщо книга була комусь позичена, то слід передбачити поле, де буде вказано прізвище позичальника.

//Для перегляду книг слід передбачити їх сортування за вибором ;

// за прізвищем автора

//за назвою книги.

//Пошук слід здійснювати

//за прізвищем автора

//назвою книги (у тому разі і за окремими літерами)

//за видом.

//Обробити ситуацію, коли під час пошуку за введеними даними нічого не знайдено.

Передбачити ситуацію, коли позичена книга буде повернута або книгу побажають позичити.

\*/

typedef struct {

char autor[N];

char nazva[N];

char rozdil[N];

int nomer\_mesta;

int vdoma;

char prizvische\_pozik[N];

} Kniga;

/\* редактор для массива структур \*/

void redactor(Kniga\*, int);

void redactorBook(Kniga\*, int);

void redactorBookMenu(Kniga\*, int, int);

void redactorBookZaemshik(Kniga\*, int, int);

void redactorBookNumMesto(Kniga\*, int, int);

void redactorBookStatusMenu(Kniga\*, int, int);

void redactorBookStatusAdd(Kniga\*, int, int);

void redactorBookStatusDelete(Kniga\*, int, int);

void redactorBookName(Kniga\*, int, int);

void redactorBookAutor(Kniga\*, int, int);

void redactorBookRozdil(Kniga\*, int, int);

/\* створення динамічного масиву структур \*/

Kniga\* initArrayStart(int);

Kniga\* initArray(int);

/\*Запись в файл\*/

void initWrite(Kniga\*, int, int);

void write(Kniga, FILE\*);

/\*Добавление нового елемента массива\*/

void writeElementInFile(FILE\*, Kniga\*, int );

void addElement(Kniga\*, int);

/\*Удалить елемент из массива\*/

void deleteElementChoosed(Kniga\*, int, int);

void deleteElement(Kniga\*, int);

/\* ініціалізація структур \*/

Kniga initKniga();

Kniga initKnigaFileR(FILE\*);

/\* відображення структур \*/

void displayArray(Kniga\*, int);

/\* відображення інформації про структуру \*/

void displayKniga(Kniga, int);

/\* сортування структур \*/

void sort(Kniga\*, int);

void sortName(Kniga\*, int);

void sortNazva(Kniga\*, int);

void sortNomerMesta(Kniga\*, int);

/\* поиск структур \*/

int displayChoiseNazva(Kniga\*, int, int, char\*);

int displayChoiseAutor(Kniga\*, int, int, char\*);

int displayChoiseRozdil(Kniga\*, int, int, char\*);

void displayChoise(Kniga\*, int);

/\*Считываем конец структуры в файле\*/

int check();

**Додаток Б**

*Лістинг «main.c»*

#include "source.h"

/\*///////////////////////////////\*/

/\* редактор для массива структур \*/

/\*///////////////////////////////\*/

void redactorBookRozdil(Kniga\* massive, int dimension, int choose\_book) {

char str[N] = { 0 };

int congrats = 0;

do {

puts("Введите новый роздел книги: ");

scanf\_s("%s", str, N);

printf("\nВы подтверждаете вашу строку '%s'(1-да, другое - нет):", str);

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

if (congrats == 1 && str != "")

{

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

massive[choose\_book].rozdil[i] = str[i];

}

break;

}

} while (1);

}

void redactorBookAutor(Kniga\* massive, int dimension, int choose\_book) {

char str[N] = { 0 };

int congrats = 0;

do {

puts("Введите нового автора книги: ");

scanf\_s("%s", str, N);

printf("\nВы подтверждаете вашу строку '%s'(1-да, другое - нет):", str);

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

if (congrats == 1 && str != "")

{

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

massive[choose\_book].autor[i] = str[i];

}

break;

}

} while (1);

}

void redactorBookName(Kniga\* massive, int dimension, int choose\_book) {

char str[N] = { 0 };

int congrats = 0;

do {

puts("Введите новоe название книги: ");

scanf\_s("%s", str, N);

printf("\nВы подтверждаете вашу строку '%s'(1-да, другое - нет):", str);

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

if (congrats == 1 && str != "")

{

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

massive[choose\_book].nazva[i] = str[i];

}

break;

}

} while (1);

}

void redactorBookStatusDelete(Kniga\* massive, int dimension, int choose\_book) {

char book\_taker[N] = "-\0";

int congrats = 0;

do {

printf("\nВы подтверждаете возвращение книги(1-да, 0 - нет): ");

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

if (congrats == 1)

{

massive[choose\_book].vdoma = 0;

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

massive[choose\_book].prizvische\_pozik[i] = book\_taker[i];

}

puts("Статус успешно изменен!");

break;

}

else if (congrats == 0)

break;

} while (1);

}

void redactorBookStatusAdd(Kniga\* massive, int dimension, int choose\_book) {

massive[choose\_book].vdoma = 1;

char book\_taker[N] = {0};

int congrats = 0;

do {

puts("Введите нового заемщика: ");

scanf\_s("%s", book\_taker, N);

printf("\nВы подтверждаете вашу строку '%s'(1-да, другое - нет):", book\_taker);

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

if (congrats == 1 && book\_taker != "")

{

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

massive[choose\_book].prizvische\_pozik[i] = book\_taker[i];

}

puts("Статус успешно изменен!");

break;

}

} while (1);

}

void redactorBookStatusMenu(Kniga\* massive, int dimension, int choose\_book) {

int congrats = 0;

if (massive[choose\_book].vdoma == 1)

puts("На данный момент эта книга находится на руках!");

else puts("На данный момент эта книга находится в библиотеке!");

do {

printf("\nВы подтверждаете что хoтите изменить статус книги(1-да, 0 - нет): ");

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

if (congrats == 1)

{

if (massive[choose\_book].vdoma == 1) {

redactorBookStatusDelete(massive, dimension, choose\_book);

break;

}

else {

redactorBookStatusAdd(massive, dimension, choose\_book);

break;

}

}

else if (congrats == 0)

break;

} while (1);

}

void redactorBookNumMesto(Kniga\* massive, int dimension, int choose\_book) {

int new\_num\_mesto = 0, yes = 0, temp = 0, zanyato = 0;

SLASH;

do {

puts("Введите новый номер:");

scanf\_s("%d", &new\_num\_mesto);

printf("\nВаше новое место = %d\

\nВы подтверждаете(1-да. 0 - нет): ", new\_num\_mesto);

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

if (yes == 1 && new\_num\_mesto > 0)

break;

} while (1);

for (int i = 0; i < dimension; i++)

if (massive[i].nomer\_mesta == new\_num\_mesto)

zanyato = i;

if (zanyato != 0 ) {

temp = massive[choose\_book].nomer\_mesta;

massive[zanyato].nomer\_mesta = 99999;

massive[choose\_book].nomer\_mesta = new\_num\_mesto;

massive[zanyato].nomer\_mesta = temp;

}

else massive[choose\_book].nomer\_mesta = new\_num\_mesto;

}

void redactorBookZaemshik(Kniga\* massive, int dimension, int choose\_book) {

if (massive[choose\_book].vdoma == 1)

{

char str[N] = {0};

int congrats = 0;

do {

puts("Введите нового заемщика: ");

scanf\_s("%s", str, N);

printf("\nВы подтверждаете вашу строку '%s'(1-да, другое - нет):", str);

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

if (congrats == 1 && str != "")

{

for (int i = 0; i < N; i++)

massive[choose\_book].prizvische\_pozik[i] = str[i];

break;

}

} while (1);

}

else puts("Книга находится в библиотеке! Используйте другую фукнцию!");

}

void redactorBookMenu(Kniga\* massive, int dimension, int choose\_book) {

int choose\_element = 0;

do {

SLASH;

puts(" РЕДАКТОР КНИГ\

\n1.Изменить номер места книги;\

\n2.Изменить статус книги;\

\n3.Изменить заемщика книги;\

\n4.Изменить название книги;\

\n5.Изменить автора книги;\

\n6.Изменить роздел книги;\

\n0.Exit;\

\nВаш выбор: ");

scanf\_s("%d", &choose\_element);

if (choose\_element == 0)

break;

else if (choose\_element == 1)

redactorBookNumMesto(massive, dimension, choose\_book);

else if (choose\_element == 2)

redactorBookStatusMenu(massive, dimension, choose\_book);

else if (choose\_element == 3)

redactorBookZaemshik(massive, dimension, choose\_book);

else if (choose\_element == 4)

redactorBookName(massive, dimension, choose\_book);

else if (choose\_element == 5)

redactorBookAutor(massive, dimension, choose\_book);

else if (choose\_element == 6)

redactorBookRozdil(massive, dimension, choose\_book);

} while (1);

}

void redactorBook(Kniga\* massive, int dimension) {

int choose\_book = 0;

displayArray(massive, dimension);

SLASH;

do {

puts("Введите номер книги по таблице, для изменения её параметров(0-exit):");

scanf\_s("%d", &choose\_book);

choose\_book -= 1;

if (choose\_book < dimension && choose\_book > -1) {

TIRE;

displayKniga(massive[choose\_book], choose\_book);

redactorBookMenu(massive, dimension, choose\_book);

break;

}

if (choose\_book == -1)

break;

} while (1);

}

void redactor(Kniga\* massive, int dimension){

int choose\_redaktor = 0;

do {

puts(" РЕДАКТОР КНИГ\

\n1.Вывести все книги;\

\n2.Выбрать книгу для изменений;\

\n0.Exit;\

\nВаш выбор: ");

scanf\_s("%d", &choose\_redaktor);

if (choose\_redaktor == 0)

break;

else if (choose\_redaktor == 1){

displayArray(massive, dimension);

SLASH;

}

else if (choose\_redaktor == 2) {

redactorBook(massive, dimension);

SLASH;

}

} while (1);

}

/\*///////////////////////////////////////\*/

/\* створення динамічного масиву структур \*/

/\*///////////////////////////////////////\*/

Kniga\* initArray(int dimension) {

Kniga\* massive = (Kniga\*)malloc(dimension \* sizeof(Kniga));

if (massive == NULL)

return NULL;

int vubor = 0;

FILE\* data;

fopen\_s(&data, "data.txt", "r");

if (!data)

{

puts("Failed to open file \n");

exit(1);

}

for (int i = 0; i < dimension; i++)

massive[i] = initKnigaFileR(data);

fclose(data);

return massive;

}

Kniga\* initArrayStart(int dimension) {

Kniga\* massive = (Kniga\*)malloc(dimension \* sizeof(Kniga));

if (massive == NULL)

return NULL;

for (int i = 0; i < dimension; i++)

massive[i] = initKniga();

return massive;

}

/\*////////////////////////\*/

/\* ініціалізація структур \*/

/\*////////////////////////\*/

Kniga initKniga() {

SLASH;

Kniga kniga;

printf("Введите автора книги: ");

scanf\_s("%s", &kniga.autor, N);

printf("Введите название книги: ");

scanf\_s("%s", &kniga.nazva, N);

printf("Введите роздел книги: ");

scanf\_s("%s", &kniga.rozdil, N);

printf("Введите номер места книги: ");

scanf\_s("%d", &kniga.nomer\_mesta);

do {

printf("Введите статус книги(1-дома/0-библиотека): ");

scanf\_s("%d", &kniga.vdoma);

if (kniga.vdoma == 0)

break;

else if (kniga.vdoma == 1)

{

printf("Введите фамилию заемщика книги: ");

scanf\_s("%s", &kniga.prizvische\_pozik, N);

break;

}

} while (1);

return kniga;

}

Kniga initKnigaFileR(FILE\* data) {

Kniga kniga;

fscanf\_s(data, "%s ", kniga.autor, N);

fscanf\_s(data, "%s ", kniga.nazva, N);

fscanf\_s(data, "%s ", kniga.rozdil, N);

fscanf\_s(data, "%d ", &kniga.nomer\_mesta);

fscanf\_s(data, "%d ", &kniga.vdoma);

fscanf\_s(data, "%s ", kniga.prizvische\_pozik, N);

return kniga;

}

/\*/////////////\*/

/\*Запись в файл\*/

/\*/////////////\*/

void initWrite(Kniga\* massive, int dimension, int rezum) {

FILE\* data;

if (rezum == 2)

fopen\_s(&data, "data.txt", "w");//запись в новый файл

else fopen\_s(&data, "data.txt", "ab");//дозапись и чтение

if (!data) {

puts("Failed to open file \n");

exit(1);

}

for (int i = 0; i < dimension; i++)

write(massive[i], data);

if (rezum == 2)

fprintf(data, "- - - 0 0 -\n");

fclose(data);

//puts("Книги в библиотеке!");

}

void write(Kniga kniga, FILE\* data) {

fprintf(data, "%s ", kniga.autor);

fprintf(data, "%s ", kniga.nazva);

fprintf(data, "%s ", kniga.rozdil);

fprintf(data, "%d ", kniga.nomer\_mesta);

if (kniga.vdoma == 1)

fprintf(data, "%d %s\n", kniga.vdoma, kniga.prizvische\_pozik);

else //(massive.vdoma == 0)

fprintf(data, "%d -\n", kniga.vdoma);

}

/\*//////////////////////////////////\*/

/\*Добавление нового елемента массива\*/

/\*//////////////////////////////////\*/

void writeElementInFile(FILE\* data, Kniga\* massive, int dimension) {

int nomer\_mesta = 0, vdoma = 0, marker = 0;

char autor[N] = { 0 }, nazva[N] = { 0 }, rozdil[N] = { 0 }, prizvische\_pozik[N] = { 0 };

printf("\nВведите название новой книги: ");

scanf\_s("%s", &nazva, N);

printf("\nВведите автора новой книги: ");

scanf\_s("%s", &autor, N);

printf("\nВведите роздел новой книги: ");

scanf\_s("%s", &rozdil, N);

do {

printf("\nВведите номер места новой книги: ");

scanf\_s("%d", &nomer\_mesta);

for (int i = 0; i < dimension;i++)

if ((massive[i].nomer\_mesta == nomer\_mesta) || nomer\_mesta <= 0)

marker = 1;

if (marker == 0)

break;

else marker = 0;

}while(1);

do {

printf("Введите статус книги книги(1-дома/0-библиотека): ");

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

if (vdoma == 0)

break;

else if (vdoma == 1)

{

printf("Введите фамилию заемщика книги: ");

scanf\_s("%s", &prizvische\_pozik, N);

break;

}

} while (1);

//Write in File Now

fprintf(data, "%s ", autor);

fprintf(data, "%s ", nazva);

fprintf(data, "%s ", rozdil);

fprintf(data, "%d ", nomer\_mesta);

if (vdoma == 1)

fprintf(data, "%d %s\n", vdoma, prizvische\_pozik);

else //(massive.vdoma == 0)

fprintf(data, "%d -\n", vdoma);

}

void addElement(Kniga\* massive, int dimension) {

FILE\* data;

fopen\_s(&data, "data.txt", "w");//запись в новый файл

if (!data)

{

puts("Failed to open file \n");

exit(1);

}

for (int i = 0; i < dimension; i++)

write(massive[i], data);

writeElementInFile(data, massive, dimension);

fprintf(data, "- - - 0 0 -\n");

fclose(data);

}

/\*//////////////////////////\*/

/\*Удалить елемент из массива\*/

/\*//////////////////////////\*/

void deleteElementChoosed(Kniga\* massive, int dimension, int choose\_book){

FILE\* data;

fopen\_s(&data, "data.txt", "w");//запись в новый файл

if (!data)

{

puts("Failed to open file \n");

exit(1);

}

for (int i = 0; i < dimension; i++)

if (i != choose\_book)

write(massive[i], data);

fprintf(data, "- - - 0 0 -\n");

fclose(data);

}

void deleteElement(Kniga\* massive, int dimension) {

int choose\_book = 0, congrats = 0;

displayArray(massive, dimension);

do {

puts("Введите номер книги по таблице для удаления(0-exit):");

scanf\_s("%d", &choose\_book);

choose\_book -= 1;

if (choose\_book < dimension && choose\_book > -1) {

TIRE;

displayKniga(massive[choose\_book], choose\_book);

printf("\nВы подтверждаете удаление книги(1-да, 0 - нет): ");

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

if (congrats == 1)

deleteElementChoosed(massive, dimension, choose\_book);

break;

}

if (choose\_book == -1)

break;

} while (1);

}

/\*///////////////////////////////////////////////////\*/

/\* передача структури в функцію для друку інформації \*/

/\*///////////////////////////////////////////////////\*/

void displayArray(Kniga\* massive, int dimension){

SLASH;

TIRE;

printf("| # | Автор | Название | Роздел | # места | Статус | Заемщик |\n");

TIRE;

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

displayKniga(massive[i], i);

}

}

/\*///////////////////////////////////////\*/

/\* відображення інформації про структуру \*/

/\*///////////////////////////////////////\*/

void displayKniga(Kniga kniga, int i) {

printf("|%4d.|%10s | %16s | %10s | %7d | ",i + 1, kniga.autor, kniga.nazva, kniga.rozdil, kniga.nomer\_mesta);

if(kniga.vdoma == 1)

printf(" дома | %10s |\n", kniga.prizvische\_pozik);

else printf("библиотека | |\n");

TIRE;

}

/\*/////////////////////\*/

/\* сортування структур \*/

/\*/////////////////////\*/

void sortName(Kniga\* massive, int dimension){

Kniga temp;

for (int i = 0; i <= dimension; i++)

for (int j = dimension - 1; j > i; j--)

if (strcmp(massive[j].autor, massive[j - 1].autor) < 0) {

temp = massive[j];

massive[j] = massive[j - 1];

massive[j - 1] = temp;

}

}

void sortNazva(Kniga\* massive, int dimension) {

Kniga temp;

for (int i = 0; i <= dimension; i++)

for (int j = dimension - 1; j > i; j--)

if (strcmp(massive[j].nazva, massive[j - 1].nazva) < 0) {

temp = massive[j];

massive[j] = massive[j - 1];

massive[j - 1] = temp;

}

}

void sortNomerMesta(Kniga\* massive, int dimension) {

Kniga temp;

for (int i = 0; i <= dimension; i++)

for (int j = dimension - 1; j > i; j--)

if (massive[j].nomer\_mesta < massive[j - 1].nomer\_mesta){

temp = massive[j - 1];

massive[j - 1] = massive[j];

massive[j] = temp;

}

}

void sort(Kniga\* massive, int dimension) {

int sort\_vibor = 0;

puts("Выберете способ сортировки:\n1.Название;\n2.Автор;\n3.Номер места;\nВаш выбор: ");

scanf\_s("%d", &sort\_vibor);

if (sort\_vibor == 1)

sortNazva(massive, dimension);

else if (sort\_vibor == 2)

sortName(massive, dimension);

else if (sort\_vibor == 3)

sortNomerMesta(massive, dimension);

else puts("Error");

}

/\*////////////////\*/

/\* поиск структур \*/

/\*////////////////\*/

int displayChoiseNazva(Kniga\* massive, int dimension, int i, char\* poisk\_nazva)

{

if (massive[i].nazva[0] == poisk\_nazva[0])

if (strstr(massive[i].nazva, poisk\_nazva))

return 1;

else return 0;

else return 0;

}

int displayChoiseAutor(Kniga\* massive, int dimension, int i, char\* poisk\_avtor)

{

if (massive[i].autor[0] == poisk\_avtor[0])

if (strstr(massive[i].autor, poisk\_avtor))

return 1;

else return 0;

else return 0;

}

int displayChoiseRozdil(Kniga\* massive, int dimension, int i, char\* poisk\_vidav)

{

if (massive[i].rozdil[0] == poisk\_vidav[0])

if (strstr(massive[i].rozdil, poisk\_vidav))

return 1;

else return 0;

else return 0;

}

void displayChoise(Kniga\* massive, int dimension)

{

int sort\_vibor[E] = { 0 }, vibor1 = 0, vibor2 = 0, vibor3 = 0;// sort = 0;

printf("\nВыберете способ искать:\n1.Название;\n2.Автор;\n3.Видом;\n(0 - выход)\nВаш выбор: ");

for (int i = 0; i < E; i++)

{

scanf\_s("%d", &sort\_vibor[i]);

if (sort\_vibor[i] == 0)

break;

}

for (int i = 0; i < E; i++)

{

if (sort\_vibor[i] == 0)

break;

else if (sort\_vibor[i] == 1)

vibor1 = 1;

else if (sort\_vibor[i] == 2)

vibor2 = 1;

else if (sort\_vibor[i] == 3)

vibor3 = 1;

else;

}

char poisk\_nazva[N] = { 0 }, poisk\_rozdil[N] = { 0 }, poisk\_avtor[N] = { 0 };

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

if (vibor1 == 1) {

printf("\nДля поиска по ключевому названию введите его: ");

scanf\_s("%s", poisk\_nazva, N);

puts(poisk\_nazva);

vibor1 = 2;

}

else if (vibor2 == 1)

{

printf("\nДля поиска по автору введите ее: ");

scanf\_s("%s", poisk\_avtor, N);

puts(poisk\_avtor);

vibor2 = 2;

}

else if (vibor3 == 1)

{

printf("\nДля поиска розделу введите ее: ");

scanf\_s("%s", poisk\_rozdil, N);

puts(poisk\_rozdil);

vibor3 = 2;

}

}

int kostil = 1, oshibka\_poiska = 0;

for (int i = 0, truue = 1; i < dimension; i++, truue = 1)

{

switch (kostil)

{

case 1:

{

if (vibor1 == 2 && truue == 1)

truue = displayChoiseNazva(massive, dimension, i, poisk\_nazva);

}

case 2:

{

if (vibor2 == 2 && truue == 1)

truue = displayChoiseAutor(massive, dimension, i, poisk\_avtor);

}

case 3:

{

if (vibor3 == 2 && truue == 1)

truue = displayChoiseRozdil(massive, dimension, i, poisk\_rozdil);

}

case 4:

{

if (oshibka\_poiska == 0 && (i == dimension - 1) && truue == 0) {

SLASH;

puts("\nК сожалению, мы ничего не нашли(");

}

if (truue == 0)

break;

TIRE;

displayKniga(massive[i], i);

oshibka\_poiska++;

}

}

}

}

void displayСonsensus(Kniga\* massive, int dimension) {

char slovo[N] = {0};

int n = 0;

printf("\nДля поиска по ключевому слову введите его: ");

scanf\_s("%s", slovo, N -1);

printf("slovo = %s\n", slovo);

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

if (n == 0)

n = strstr(massive[i].autor, slovo);

if (n == 0)

n = strstr(massive[i].nazva, slovo);

if (n == 0)

n = strstr(massive[i].rozdil, slovo);

if (n != 0) {

TIRE;

displayKniga(massive[i], i);

n = 0;

}

}

}

void displayMenu(Kniga\* massive, int dimension) {

int choose = 0;

do {

printf("\nВыберите режим поиска:\n1.Точный ввод\n2.Поиск по совпадению\n0.Exit\nВаш выбор:");

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

if (choose == 0)

break;

else if (choose == 1)

displayChoise(massive, dimension);

else if (choose == 2)

displayСonsensus(massive, dimension);

} while (1);

}

/\*/////////////////////////////////\*/

/\*Считываем конец структуры в файле\*/

/\*/////////////////////////////////\*/

int check() {

int i;

Kniga kniga;

FILE\* data;

fopen\_s(&data, "data.txt", "r");//чтение

if (!data){

puts("Failed to open file \n");

exit(1);

}

for (i = 0; i < 999; i++) {

fscanf\_s(data, "%s ", kniga.autor, N);

fscanf\_s(data, "%s ", kniga.nazva, N);

fscanf\_s(data, "%s ", kniga.rozdil, N);

fscanf\_s(data, "%d ", &kniga.nomer\_mesta);

fscanf\_s(data, "%d ", &kniga.vdoma);

fscanf\_s(data, "%s ", kniga.prizvische\_pozik, N);

if (kniga.nomer\_mesta == 0 && kniga.vdoma == 0)

break;

}

fclose(data);

return i;

}

/\*////////////////////\*/

/\*Головна ф-я програми\*/

/\*////////////////////\*/

int main(void)

{

setlocale(LC\_ALL, "Rus");

int choose\_menu = 0, dimension = 0, marker\_of\_changes = 0;

Kniga\* massiveStruct = initArrayStart(0);

puts("Добрый день! Вас приветствует Библиотека будущего! Библиотека будущего, будущее уже с нами!");

do {

if (marker\_of\_changes == 0) {

dimension = check();

massiveStruct = initArray(dimension);

marker\_of\_changes = 1;

}

SLASH;

puts(" М Е Н Ю\

\n1.Редактирование;\

\n2.Добавление;\

\n3.Сортировка;\

\n4.Поиск;\

\n5.Пересмотр;\

\n6.Удаление;\

\n0.Exit;\

\nВаш выбор: ");

scanf\_s("%d", &choose\_menu);

SLASH;

if (choose\_menu == 0)

break;

else if (choose\_menu == 1) {

redactor(massiveStruct, dimension);

marker\_of\_changes == 0;

}

else if (choose\_menu == 2) {

addElement(massiveStruct, dimension);

marker\_of\_changes = 0;

}

else if (choose\_menu == 3)

sort(massiveStruct, dimension);

else if (choose\_menu == 4)

displayMenu(massiveStruct, dimension);

else if (choose\_menu == 5)

displayArray(massiveStruct, dimension);

else if (choose\_menu == 6) {

deleteElement(massiveStruct, dimension);

marker\_of\_changes = 0;

}

} while (1);

initWrite(massiveStruct, dimension, 2);

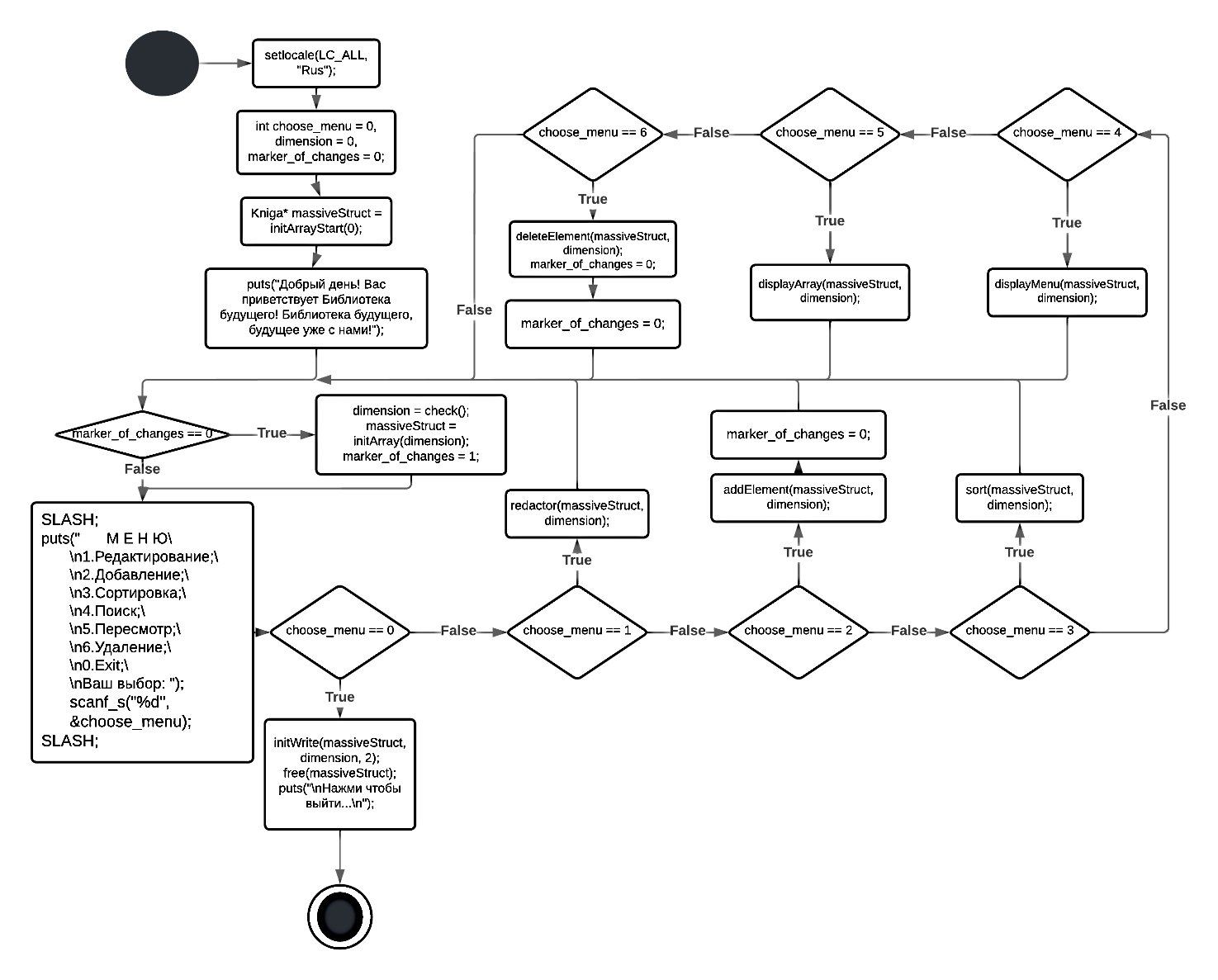
free(massiveStruct);

puts("\nНажми чтобы выйти...\n");

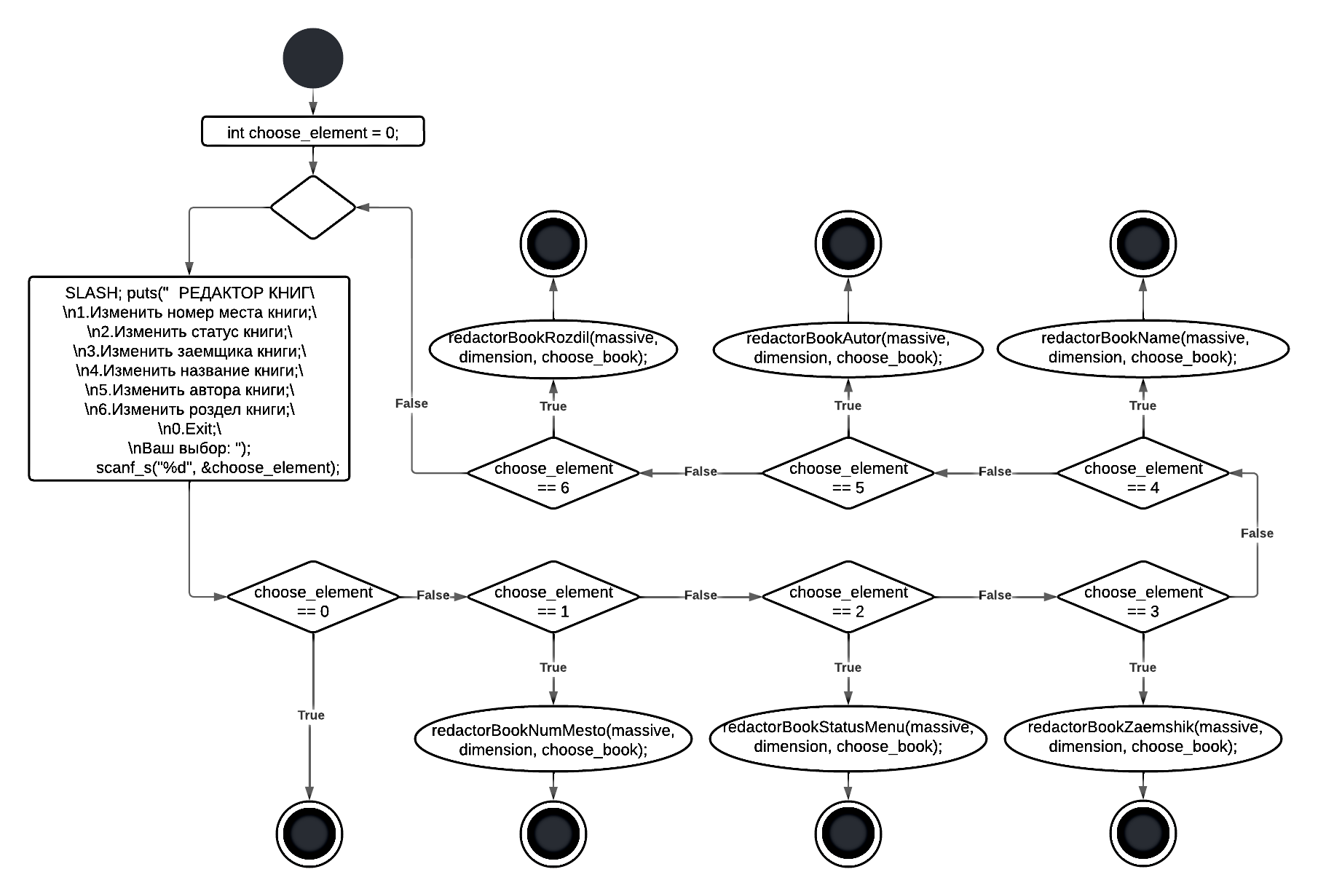
return 0;

}

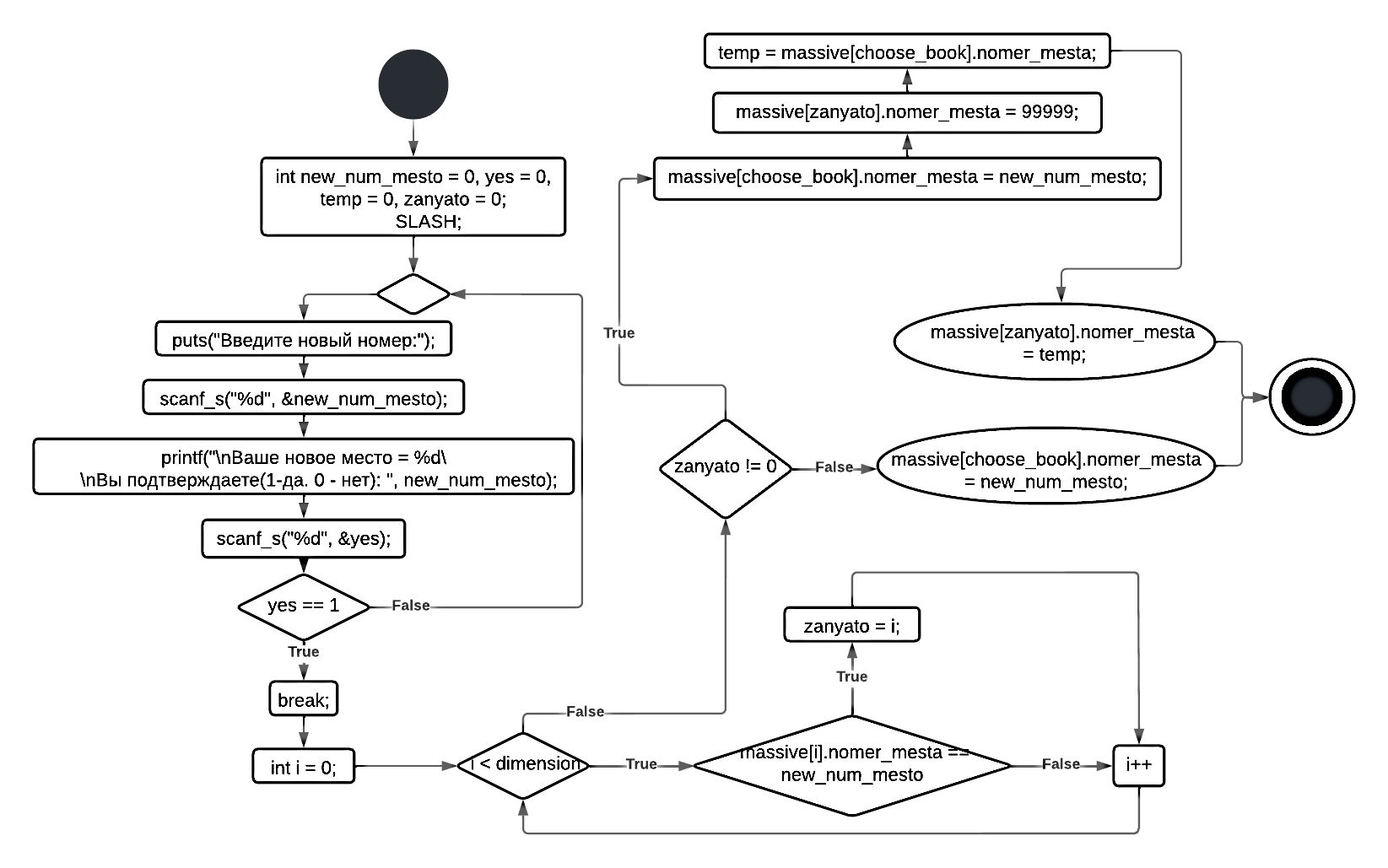
**Додаток В**



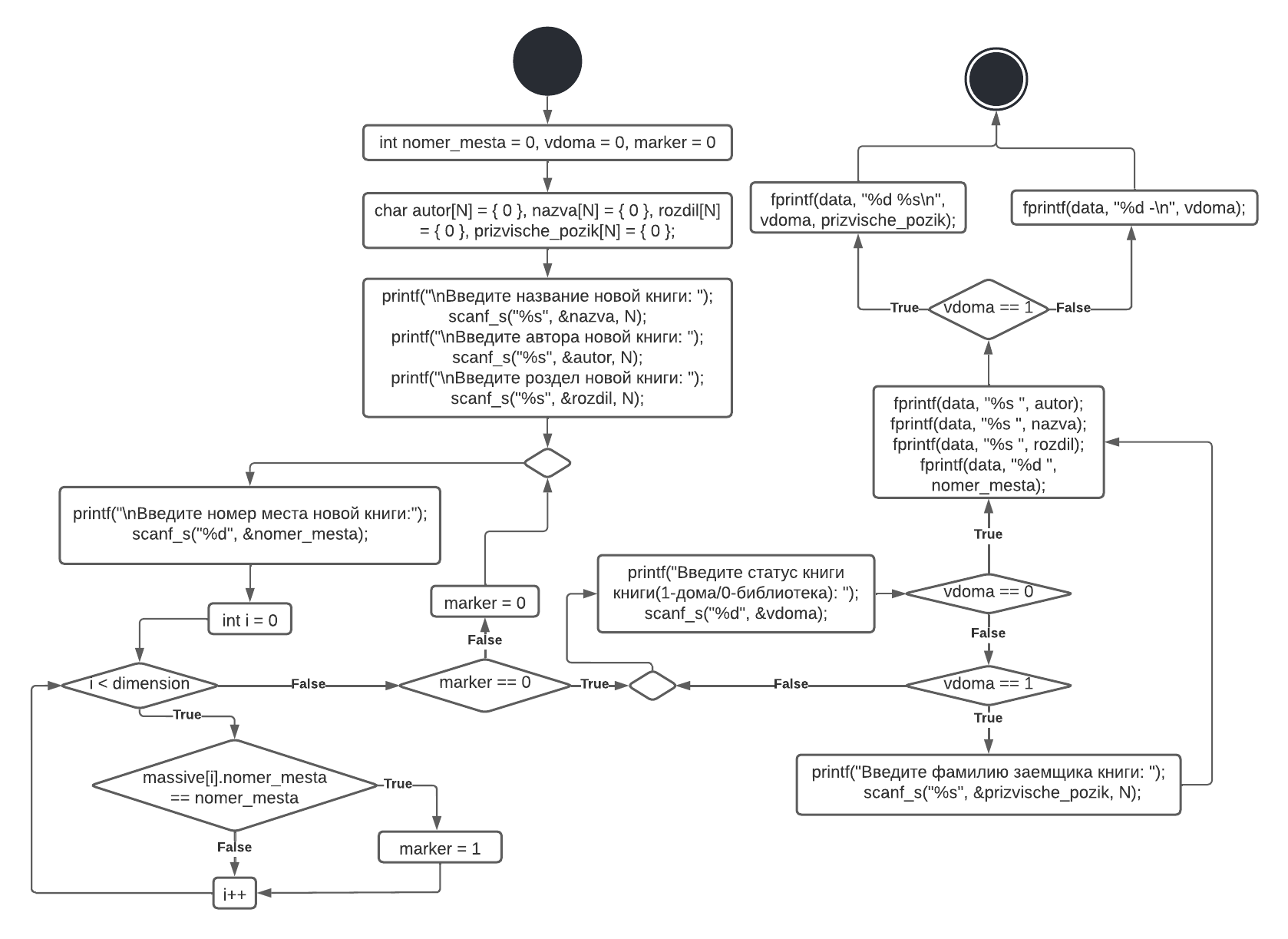
**Додаток Г**



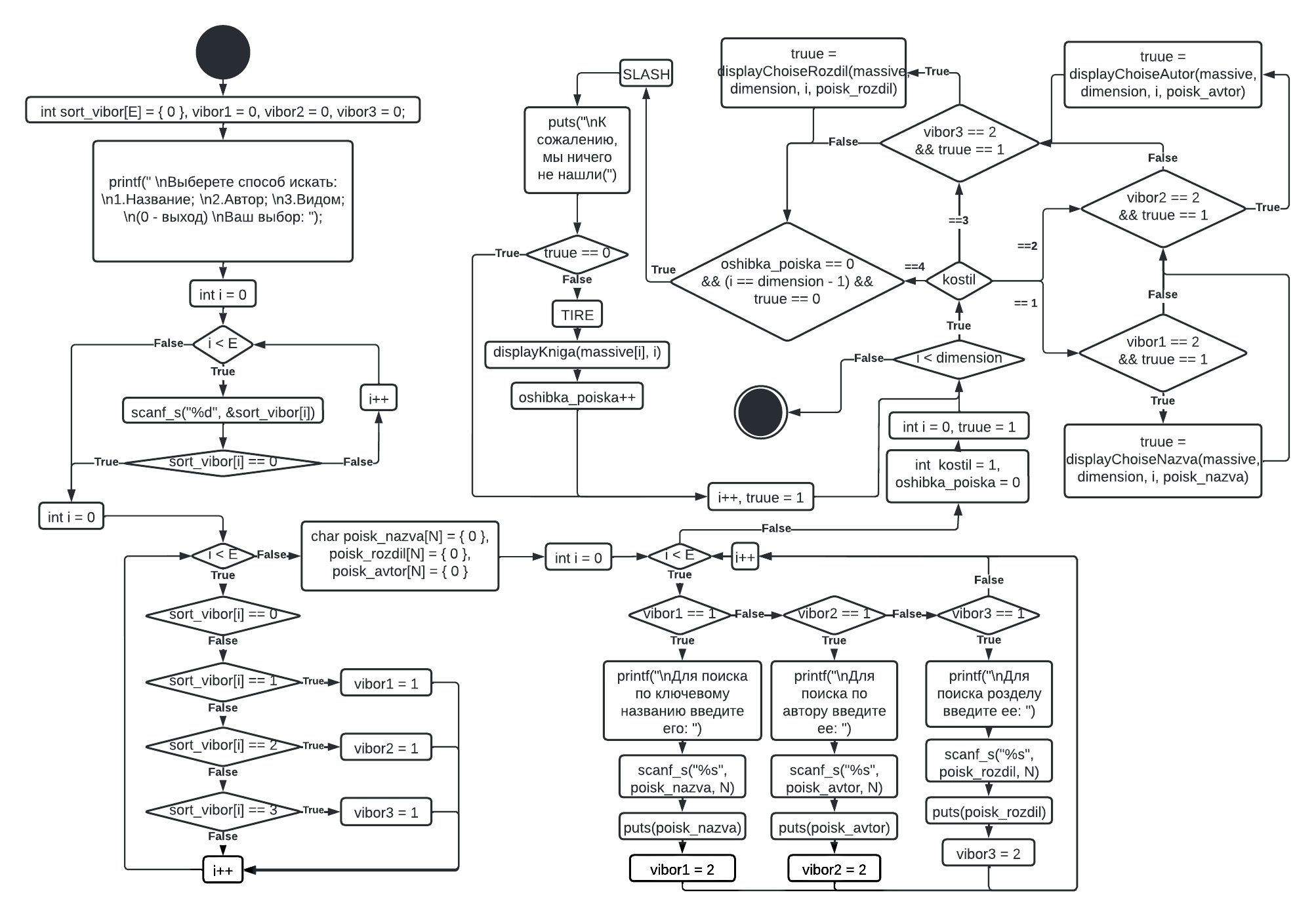
**Додаток Ґ**



**Додаток Д**



**Додаток Е**



**Додаток Є**

