**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

Інститут **ІКНІ**

Кафедра **ПЗ**



**ЗВІТ**

До лабораторної роботи №2

**На тему: “Етапи проектування та кодування програми”**

**З дисципліни:** *“Вступ до інженерії програмного забезпечення”*

**Лектор:**

к. т. н. доцент каф. ПЗ

Левус Є. В.

**Виконав:**

ст. гр. ПЗ-16

Кабачок Т. О.

**Прийняв:**

асист. каф. ПЗ

Легкий М. В.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2025 р.

Львів – 2025

**Тема:** Етапи проектування та кодування програми.

**Мета:** Отримати розуміння змісту етапів проектування та кодування. Навчитися документувати основні результати етапів проектування та кодування найпростіших програм.

**Теоретичні відомості**

**12, 18, 28**

**12. Які є відомі способи представлення алгоритмів?**

* словесний – містить занумеровані пункти у вигляді окремих команд для виконання певних дій
* формульний – порядок здійснення обчислення для здобуття числового результату
* розрахункова таблиця – для серії розрахунків за однаковими формулами
* блок-схема – сукупність геометричних фігур зʼєднаних напрямними лініями

**18. Які вимоги до запису коментарів у тексті програми?**

* чіткість, читабельність, інформативність
* однострічковий коментар через //, для коментування великих фрагментів /\* \*/.

**28. Як можна підвищити продуктивність роботи на етапі кодування?**

* Використання стандартів оформлення коду – сукупності вказівок, які встановлюють правила стильового оформлення коду. Метою прийняття та використання стандарту є спрощення сприйняття програмного коду людиною, мінімізація навантаження на пам’ять та зір при читанні програми.

**Постановка завдання**

**Частина I.** У розробленій раніше програмі до лабораторної роботи з  дисципліни «Основи програмування» внести зміни – привести її до модульної структури, де модуль – окрема функція-підпрограма. У якості таких функцій запрограмувати алгоритми зчитування та запису у файл, сортування, пошуку, редагування, видалення елементів та решта подібних функцій згідно з варіантом.

**Частина II.** Перевірити роботу програми згідно з технічним завданням (п.3, п.4.1). У разі розбіжності внести корективи у програму.

**Частина III.** Сформувати пакет документів до розробленої раніше власної програми:

1. Схематичне зображення структур даних, які використовуються для збереження інформації.
2. Блок-схема алгоритмів – основної функції і двох окремих функцій- підпрограм.
3. Схематичне зображення модульної структури програми у формі набору функцій.
4. Текст програми з коментарями, оформлений згідно з наведеними у методичних вказівках рекомендаціями щодо читабельності й зрозумілості.

Для схематичного зображення структур даних, блок-схем алгоритму, структури програми можна використати редактор MS-Visio або інший редактор інженерної та ділової графіки.

**Варіант завдання**

З текстового файлу зчитати послідовність записів, які містять дані про книгу: <Автор>, <Назва книги>, <Рік видання>, <Кількість сторінок>, <Вартість>. Роздрукувати введені дані у вигляді таблиці, а також подати інформацію згідно варіанту. Відсортувавати дані за прізвищами авторів в алфавітному порядку. Реалізувати операцію вставки нового елемента у відсортований список і операцію вилучення зі списку даних, які відповідають одній з наступних умов: про студентів, які не мають оцінки 5.

**Отримані результати**

**Схематичне зображення структур даних**

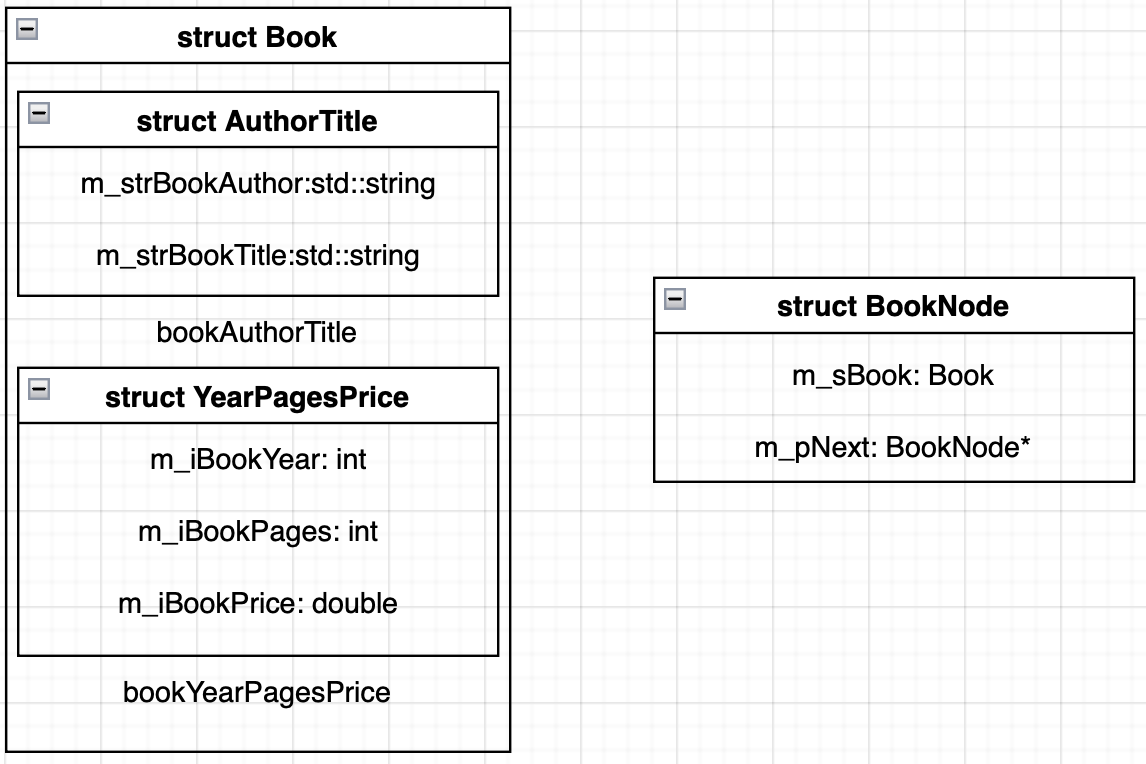


Рис. 1. Структури Book, AuthorTitle, YearPagesPrice, BookNode.

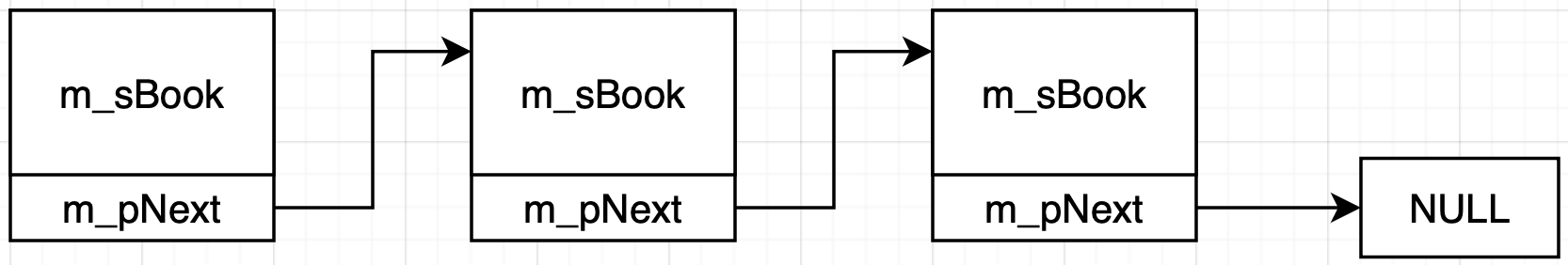


Рис. 2. Схематичне зображення однозв’язного списку Book.

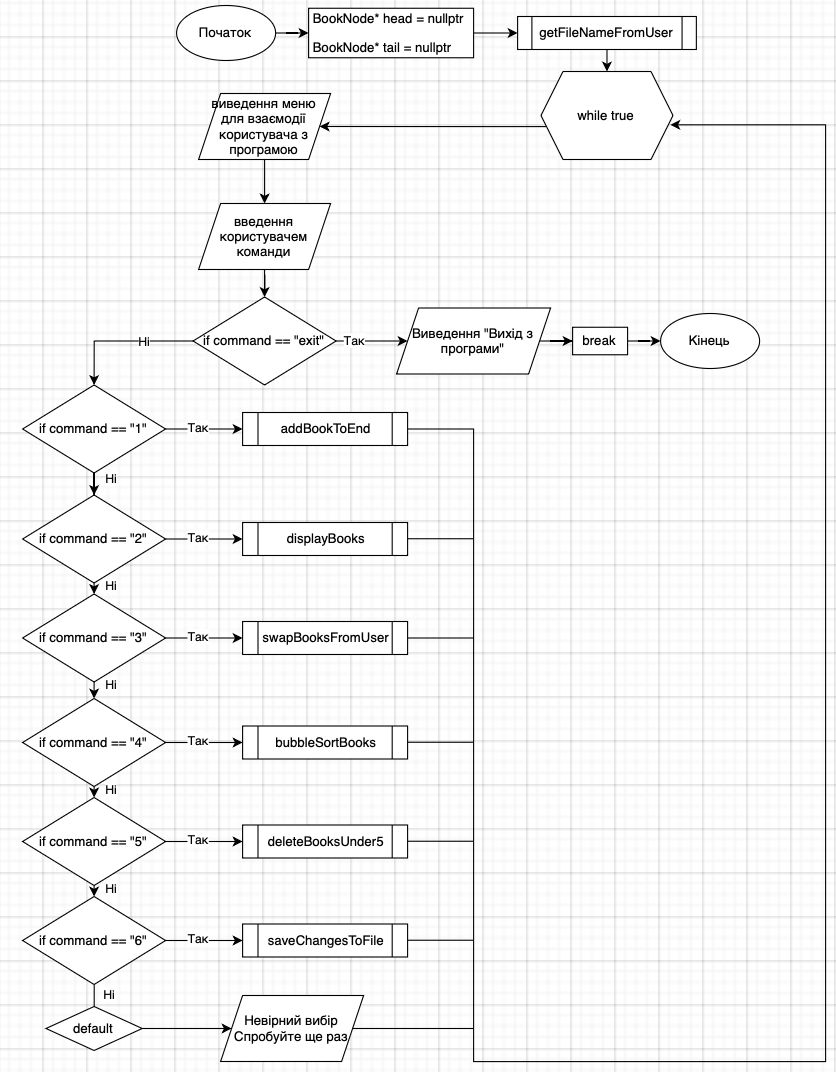


Рис. 3. Блок-схема основної функції.

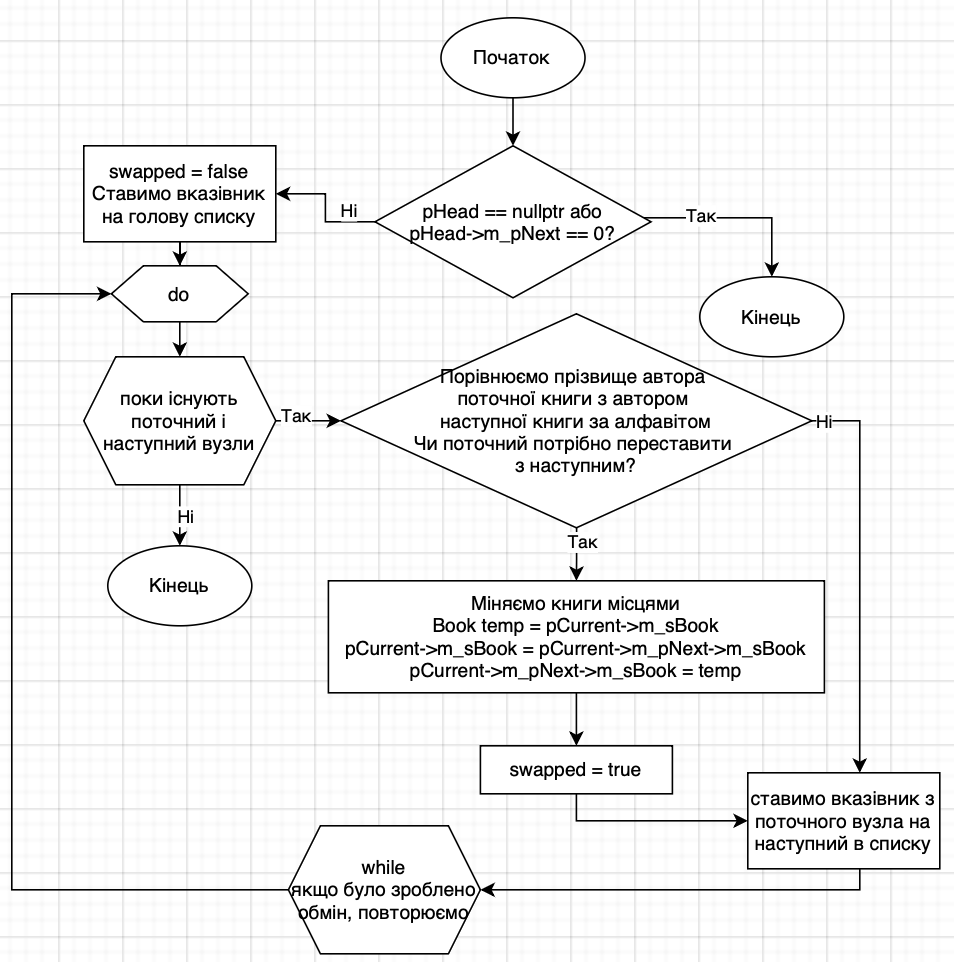


Рис. 4. Блок-схема алгоритму сортування списку книг.

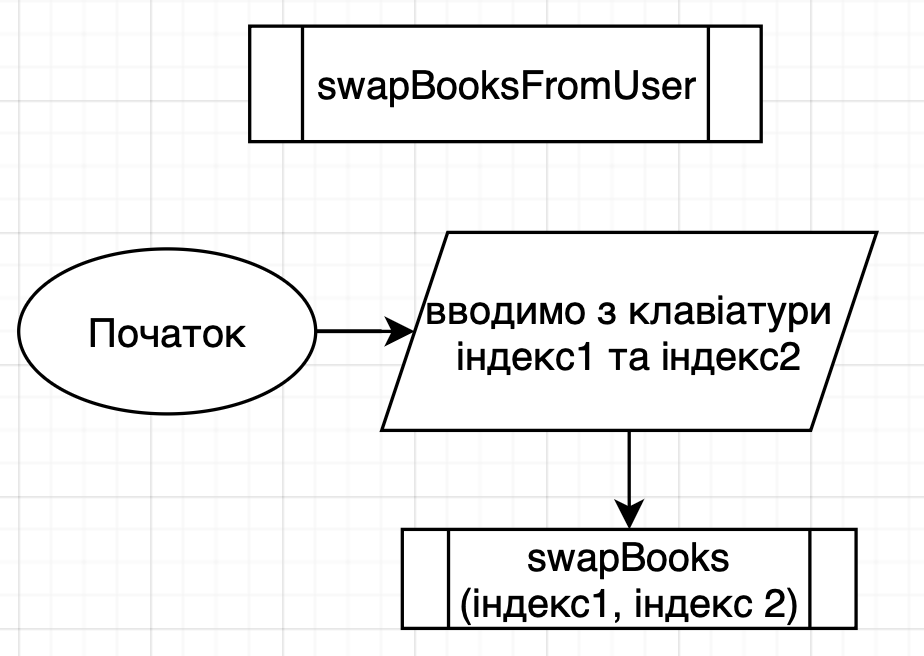


Рис. 5. Блок-схема функції swapBooksFromUser.

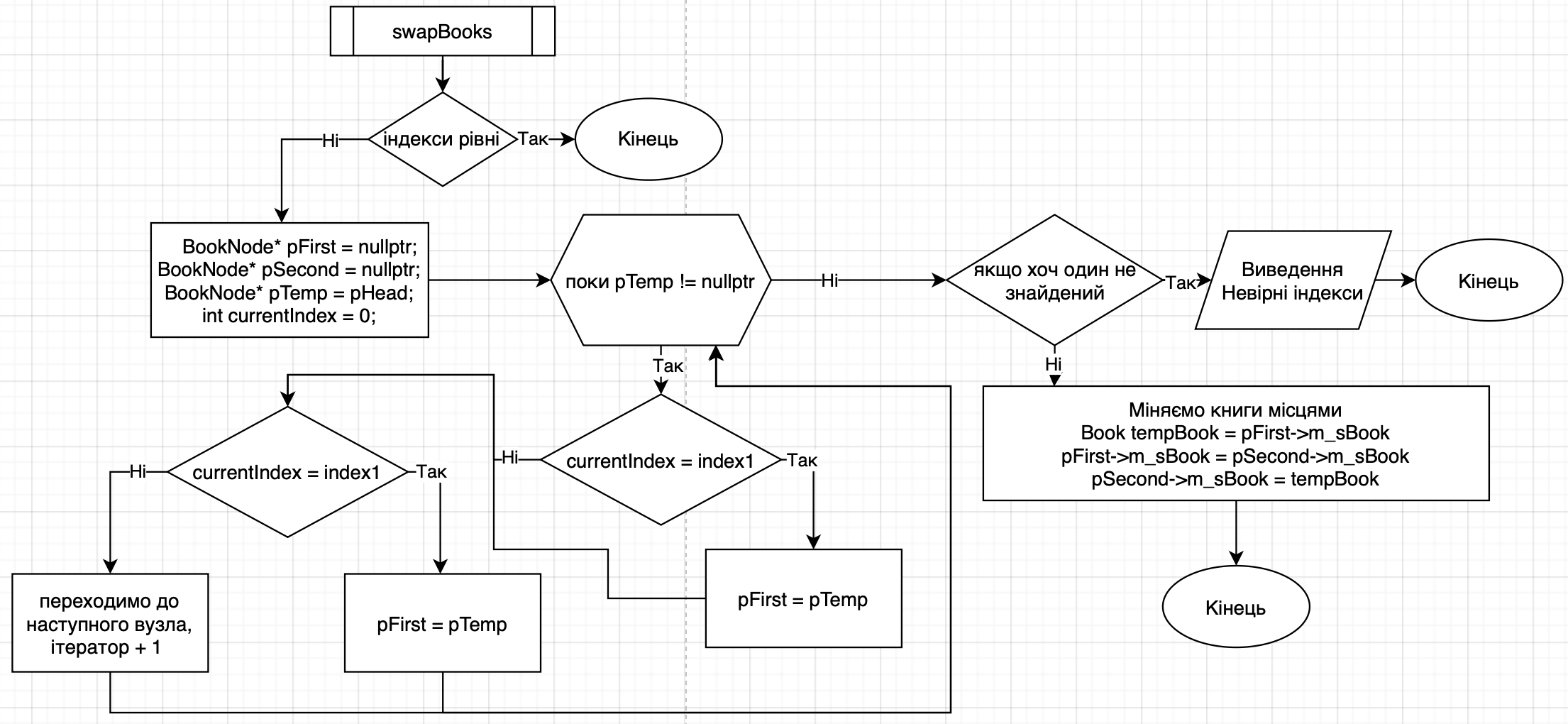


Рис. 6. Блок-схема функції swapBooks

**Текст програми**

**Файл func.h**

#pragma once

#include <string>

// Структура для зберігання інформації про книгу.

// Включає автора та назву книги, а також рік видання, кількість сторінок і ціну.

// Приклад створення об'єкта та використання:

//

// Book myBook;

// myBook.bookAuthorTitle.m\_strBookAuthor = "Автор книги";

// myBook.bookAuthorTitle.m\_strBookTitle = "Назва книги";

// myBook.bookYearPagesPrice.m\_iBookYear = 2025;

// myBook.bookYearPagesPrice.m\_iBookPages = 200;

// myBook.bookYearPagesPrice.m\_dBookPrice = 250.0;

//

// В результаті об'єкт myBook міститиме інформацію про книгу.

struct Book {

// Структура для зберігання автора та назви книги.

// Містить поля для збереження імені автора та назви книги.

struct AuthorTitle {

std::string m\_strBookAuthor; // Автор книги

std::string m\_strBookTitle; // Назва книги

} bookAuthorTitle;

// Структура для зберігання року видання, кількості сторінок і ціни книги.

// Містить поля для зберігання року видання, кількості сторінок і ціни книги.

struct YearPagesPrice {

int m\_iBookYear; // Рік видання книги

int m\_iBookPages; // Кількість сторінок у книзі

double m\_dBookPrice; // Ціна книги

} bookYearPagesPrice;

};

// Структура для елемента списку книжок.

// Кожен елемент містить книгу та вказівник на наступний елемент списку.

// Приклад використання:

// BookNode\* head = new BookNode; // Створюємо новий елемент списку

// head->m\_sBook.bookAuthorTitle.m\_strBookAuthor = "Автор 1";

// head->m\_sBook.bookAuthorTitle.m\_strBookTitle = "Книга 1";

// head->m\_pNext = nullptr; // Ініціалізуємо наступний елемент як nullptr

struct BookNode {

Book m\_sBook; // Книга

BookNode\* m\_pNext; // Вказівник на наступний елемент списку

};

// функція для зчитування даних з файлу та додавання книг до списку

void loadBooksFromFile(const std::string& filename, BookNode\*& head, BookNode\*& tail);

// функція для запиту назви файлу у користувача та зчитування книг

void getFileNameFromUser(BookNode\*& head, BookNode\*& tail);

// функція для додавання нової книги в кінець списку

void addBookToEnd(BookNode\*& head, BookNode\*& tail);

// функція для виведення списку книг на екран

void displayBooks(BookNode\* head);

// функція для виведення повідомлення про помилку при відкритті файлу

void displayFileError();

// функція для зміни місцями двох книжок за їх індексами

void swapBooks(BookNode\* head, int index1, int index2);

// функція для запиту у користувача індексів двох книг та їх зміни місцями

void swapBooksFromUser(BookNode\* head);

// функція для сортування списку книг за алфавітним порядком авторів

void bubbleSortBooks(BookNode\*& head);

// функція для видалення книг з ціною нижче 5 грн

void deleteBooksUnder5(BookNode\*& head);

// функція для збереження змін у файлі

void saveChangesToFile(BookNode\* head);

**Файл func.cpp**

#include "func.h"

#include <iostream>

#include <fstream>

#include <limits>

#include <cstdlib>

using namespace std;

//--------------------------------------------------------------------------------

/\* Функція завантажує книги з файлу та додає їх в список

Функція приймає:

- filename: шлях до файлу, з якого будуть завантажені книги;

- head та tail: вказівники на голову та кінець списку книг.

Під час виконання функція відкриває файл, читає дані про кожну книгу,

створює новий вузол для книги, заповнює його даними та додає до списку.

Якщо файл не відкривається, виводиться повідомлення про помилку. \*/

void loadBooksFromFile(const string& strFilename, BookNode\*& pHead, BookNode\*& pTail) {

ifstream ifsFile(strFilename);

if (!ifsFile.is\_open()) {

displayFileError(); // Виведення повідомлення про помилку при відкритті файлу

return;

}

string strAuthor, strTitle, strYear, strPages, strPrice;

while (ifsFile) {

BookNode\* pNewBookNode = new BookNode;

// Читання даних з файлу

getline(ifsFile, strAuthor, '"');

getline(ifsFile, strAuthor, '"');

getline(ifsFile, strTitle, '"');

getline(ifsFile, strTitle, '"');

getline(ifsFile, strYear, '"');

getline(ifsFile, strYear, '"');

getline(ifsFile, strPages, '"');

getline(ifsFile, strPages, '"');

getline(ifsFile, strPrice, '"');

getline(ifsFile, strPrice, '"');

// Перевірка на порожні значення

if (strAuthor.empty() || strTitle.empty() || strYear.empty() || strPages.empty() || strPrice.empty()) {

delete pNewBookNode;

continue;

}

// Присвоєння значень

pNewBookNode->m\_sBook.bookAuthorTitle.m\_strBookAuthor = strAuthor;

pNewBookNode->m\_sBook.bookAuthorTitle.m\_strBookTitle = strTitle;

pNewBookNode->m\_sBook.bookYearPagesPrice.m\_iBookYear = stoi(strYear);

pNewBookNode->m\_sBook.bookYearPagesPrice.m\_iBookPages = stoi(strPages);

pNewBookNode->m\_sBook.bookYearPagesPrice.m\_dBookPrice = stod(strPrice);

pNewBookNode->m\_pNext = nullptr;

if (!pHead) {

pHead = pNewBookNode; // Якщо список порожній, новий елемент стає головою

pTail = pHead; // Кінець списку також вказує на цей елемент

} else {

pTail->m\_pNext = pNewBookNode; // Додаємо нову книгу в кінець списку

pTail = pNewBookNode; // Оновлюємо вказівник на кінець

}

}

ifsFile.close();

displayBooks(pHead); // Виведення всіх книг

}

//--------------------------------------------------------------------------------

/\* Функція запитує у користувача назву файлу та намагається його відкрити

Функція приймає:

- head та tail: вказівники на голову та кінець списку книг.

Якщо файл відкривається, завантажує книги з цього файлу. Якщо неможливо відкрити файл,

виводиться повідомлення про помилку. \*/

void getFileNameFromUser(BookNode\*& pHead, BookNode\*& pTail) {

string strFilename;

cout << "Введіть назву файлу: ";

getline(cin, strFilename);

string strFullPath = "../data/" + strFilename;

ifstream ifsFile(strFullPath);

if (ifsFile.is\_open()) {

loadBooksFromFile(strFullPath, pHead, pTail); // Завантаження книг з файлу

ifsFile.close();

} else {

displayFileError(); // Виведення помилки при неможливості відкрити файл

}

}

//--------------------------------------------------------------------------------

/\* Функція додає нову книгу в кінець списку

Функція приймає:

- head та tail: вказівники на голову та кінець списку книг.

Під час виконання користувач вводить всі необхідні дані для нової книги,

яка потім додається в кінець списку. \*/

void addBookToEnd(BookNode\*& pHead, BookNode\*& pTail) {

BookNode\* pNewBookNode = new BookNode;

cout << "Додайте нову книгу в кінець списку" << endl;

cout << "Введіть прізвище автора: ";

getline(cin, pNewBookNode->m\_sBook.bookAuthorTitle.m\_strBookAuthor);

cout << "Введіть назву книги: ";

getline(cin, pNewBookNode->m\_sBook.bookAuthorTitle.m\_strBookTitle);

cout << "Введіть рік видання: ";

cin >> pNewBookNode->m\_sBook.bookYearPagesPrice.m\_iBookYear;

cout << "Введіть кількість сторінок: ";

cin >> pNewBookNode->m\_sBook.bookYearPagesPrice.m\_iBookPages;

cout << "Введіть ціну книги (грн): ";

cin >> pNewBookNode->m\_sBook.bookYearPagesPrice.m\_dBookPrice;

cin.ignore(numeric\_limits<streamsize>::max(), '\n'); // очищення буфера

pNewBookNode->m\_pNext = nullptr;

if (pHead == nullptr) {

pHead = pNewBookNode;

pTail = pNewBookNode;

} else {

BookNode\* pTemp = pHead;

while (pTemp->m\_pNext != nullptr) {

pTemp = pTemp->m\_pNext;

}

pTemp->m\_pNext = pNewBookNode;

pTail = pNewBookNode; // оновлюємо tail

}

cout << "Книга успішно додана!\n";

}

//--------------------------------------------------------------------------------

/\* Функція відображає всі книги в списку

Функція приймає:

- head: вказівник на голову списку книг.

Під час виконання функція виводить інформацію про всі книги в списку.

Якщо список порожній, виводиться відповідне повідомлення. \*/

void displayBooks(BookNode\* pHead) {

if (!pHead) {

cout << "Немає книг для відображення.\n";

return;

}

BookNode\* pCurrent = pHead;

int iIndex = 0; // Змінна для індексації книг

while (pCurrent) {

cout << "Індекс: " << iIndex + 1 << "\n";

cout << "Автор: " << pCurrent->m\_sBook.bookAuthorTitle.m\_strBookAuthor << "\n";

cout << "Назва книги: " << pCurrent->m\_sBook.bookAuthorTitle.m\_strBookTitle << "\n";

cout << "Рік: " << pCurrent->m\_sBook.bookYearPagesPrice.m\_iBookYear << "\n";

cout << "Сторінки: " << pCurrent->m\_sBook.bookYearPagesPrice.m\_iBookPages << "\n";

cout << "Ціна: " << pCurrent->m\_sBook.bookYearPagesPrice.m\_dBookPrice << " грн\n";

cout << "----------------------------\n";

// Переходимо до наступного елементу

pCurrent = pCurrent->m\_pNext;

iIndex++; // Збільшуємо індекс

}

}

//--------------------------------------------------------------------------------

/\* Функція відображає повідомлення про помилку при відкритті файлу \*/

void displayFileError() {

cout << "Помилка: не вдалося відкрити файл.\n";

}

/\* Функція запитує у користувача індекси двох книг і викликає функцію для їх обміну місцями

Функція приймає параметр:

head - голова списку книг

Під час виконання, функція запитує у користувача два індекси і передає їх функції swapBooks

для здійснення обміну між книгами за вказаними індексами \*/

void swapBooksFromUser(BookNode\* pHead) {

int index1, index2;

// Запитуємо у користувача індекси

cout << "Введіть індекс першої книги для зміни місцями: ";

cin >> index1;

cout << "Введіть індекс другої книги для зміни місцями: ";

cin >> index2;

cin.ignore(numeric\_limits<streamsize>::max(), '\n'); // очищаємо буфер вводу

// Викликаємо функцію для зміни місцями

swapBooks(pHead, index1 - 1, index2 - 1);

}

/\* Функція здійснює обмін двох книг у списку за заданими індексами

Функція приймає параметри:

pHead - голова списку книг

index1 - індекс першої книги для обміну

index2 - індекс другої книги для обміну

Під час виконання, функція знаходить книги за вказаними індексами і міняє їх місцями \*/

void swapBooks(BookNode\* pHead, int index1, int index2) {

if (index1 == index2) return; // Якщо індекси однакові, нічого не робимо.

BookNode\* pFirst = nullptr;

BookNode\* pSecond = nullptr;

BookNode\* pTemp = pHead;

int currentIndex = 0;

// Шукаємо елементи з відповідними індексами

while (pTemp != nullptr) {

if (currentIndex == index1) pFirst = pTemp;

if (currentIndex == index2) pSecond = pTemp;

pTemp = pTemp->m\_pNext;

currentIndex++;

}

// Якщо хоча б один з елементів не знайдений

if (!pFirst || !pSecond) {

cout << "Невірні індекси.\n";

return;

}

// Міняємо місцями книжки

Book tempBook = pFirst->m\_sBook;

pFirst->m\_sBook = pSecond->m\_sBook;

pSecond->m\_sBook = tempBook;

cout << "Книжки успішно поміняні місцями!\n";

}

/\* Функція сортує список книг за автором в алфавітному порядку (метод бульбашки)

Функція приймає параметр:

pHead - голова списку книг

Під час виконання, функція порівнює авторів книг і здійснює їх перестановку при необхідності \*/

void bubbleSortBooks(BookNode\*& pHead) {

if (!pHead || !pHead->m\_pNext) {

return; // Якщо список порожній або містить лише один елемент

}

bool swapped;

do {

swapped = false;

BookNode\* pCurrent = pHead;

while (pCurrent && pCurrent->m\_pNext) {

if (pCurrent->m\_sBook.bookAuthorTitle.m\_strBookAuthor > pCurrent->m\_pNext->m\_sBook.bookAuthorTitle.m\_strBookAuthor) {

// Міняємо місцями книги, якщо автор наступної книги має менше значення

Book temp = pCurrent->m\_sBook;

pCurrent->m\_sBook = pCurrent->m\_pNext->m\_sBook;

pCurrent->m\_pNext->m\_sBook = temp;

swapped = true;

}

pCurrent = pCurrent->m\_pNext;

}

} while (swapped); // Якщо було зроблено обмін, повторюємо

}

/\* Функція видаляє книги, ціна яких менше 5 грн, з запитом у користувача на підтвердження

Функція приймає параметр:

pHead - голова списку книг

Під час виконання, функція перевіряє ціну кожної книги і запитує у користувача

чи хоче він її видалити, якщо ціна менша за 5 грн \*/

void deleteBooksUnder5(BookNode\*& pHead) {

if (!pHead) {

cout << "Список порожній.\n";

return;

}

BookNode\* pCurrent = pHead;

BookNode\* pPrev = nullptr;

bool anyDeleted = false;

while (pCurrent) {

if (pCurrent->m\_sBook.bookYearPagesPrice.m\_dBookPrice < 5) {

cout << "Книга \"" << pCurrent->m\_sBook.bookAuthorTitle.m\_strBookTitle << "\" автор: "

<< pCurrent->m\_sBook.bookAuthorTitle.m\_strBookAuthor << " має ціну "

<< pCurrent->m\_sBook.bookYearPagesPrice.m\_dBookPrice << " грн.\n";

cout << "Чи хочете ви її видалити? (1 - так, 0 - ні): ";

int choice;

cin >> choice;

// Очищаємо буфер вводу після cin

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

if (choice == 1) {

// Видалення книги з початку списку або з середини/кінця

if (pPrev) {

pPrev->m\_pNext = pCurrent->m\_pNext;

} else {

pHead = pCurrent->m\_pNext; // Якщо це перший елемент, змінюємо head

}

BookNode\* pTemp = pCurrent;

pCurrent = pCurrent->m\_pNext;

delete pTemp;

anyDeleted = true;

} else {

pPrev = pCurrent;

pCurrent = pCurrent->m\_pNext;

}

} else {

pPrev = pCurrent;

pCurrent = pCurrent->m\_pNext;

}

}

if (!anyDeleted) {

cout << "Немає книг з ціною менше 5 грн.\n";

} else {

cout << "Видалено книги з ціною менше 5 грн.\n";

}

}

/\* Функція зберігає зміни списку книг у файл

Функція приймає параметр:

pHead - голова списку книг

Під час виконання, функція запитує у користувача назву файлу для збереження

і зберігає інформацію про всі книги у файл \*/

void saveChangesToFile(BookNode\* pHead) {

// Запитуємо назву файлу для збереження

string filename;

cout << "Введіть назву файлу для збереження: ";

getline(cin, filename);

// Додаємо шлях до папки data

string fullPath = "../data/" + filename;

// Запитуємо чи хоче користувач зберегти зміни

cout << "Чи хочете ви зберегти зміни в файл? (1 - так, 0 - ні): ";

int choice;

cin >> choice;

cin.ignore(numeric\_limits<streamsize>::max(), '\n'); // Очищаємо буфер вводу

// Якщо вибір "1", записуємо зміни у файл

if (choice == 1) {

ofstream file(fullPath);

if (!file.is\_open()) {

cout << "Не вдалося відкрити файл для запису.\n";

return;

}

BookNode\* pCurrent = pHead;

while (pCurrent) {

file << "\"" << pCurrent->m\_sBook.bookAuthorTitle.m\_strBookAuthor << "\" "

<< "\"" << pCurrent->m\_sBook.bookAuthorTitle.m\_strBookTitle << "\" "

<< "\"" << pCurrent->m\_sBook.bookYearPagesPrice.m\_iBookYear << "\" "

<< "\"" << pCurrent->m\_sBook.bookYearPagesPrice.m\_iBookPages << "\" "

<< "\"" << pCurrent->m\_sBook.bookYearPagesPrice.m\_dBookPrice << "\"\n";

pCurrent = pCurrent->m\_pNext;

}

file.close();

cout << "Зміни успішно збережено у файл: " << fullPath << "\n";

} else {

cout << "Зміни не були збережені.\n";

}

}

**Файл main.cpp**

#include <func.h>

#include <iostream>

#include <limits>

int main() {

BookNode\* head = nullptr;

BookNode\* tail = nullptr;

getFileNameFromUser(head, tail); // Зчитуємо назву файлу і перевіряємо його існування

std::string command;

while (true) {

// Виводимо меню

std::cout << "\nОберіть дію:\n";

std::cout << "1. Додати нову книгу\n";

std::cout << "2. Вивести список книг\n";

std::cout << "3. Змінити місцями книги\n";

std::cout << "4. Відсортувати книги за автором\n";

std::cout << "5. Видалити книги з ціною менше 5 грн\n";

std::cout << "6. Зберегти зміни в файл\n";

std::cout << "Введіть 'exit' для виходу.\n";

std::cout << "Ваш вибір: ";

std::getline(std::cin, command);

// Перевірка на вихід

if (command == "exit") {

std::cout << "Вихід з програми.\n";

break;

}

// Виконання вибраної операції

switch (std::stoi(command)) {

case 1:

addBookToEnd(head, tail); // Додаємо нову книгу

break;

case 2:

displayBooks(head); // Виводимо список книг

break;

case 3:

swapBooksFromUser(head); // Змінюємо книжки місцями

break;

case 4:

bubbleSortBooks(head); // Сортуємо книги

break;

case 5:

deleteBooksUnder5(head); // Видаляємо книги з ціною менше 5 грн

break;

case 6:

saveChangesToFile(head); // Зберігаємо зміни в файл

break;

default:

std::cout << "Невірний вибір. Спробуйте ще раз.\n";

break;

}

}

return 0;

}

**Результат виконання програми**

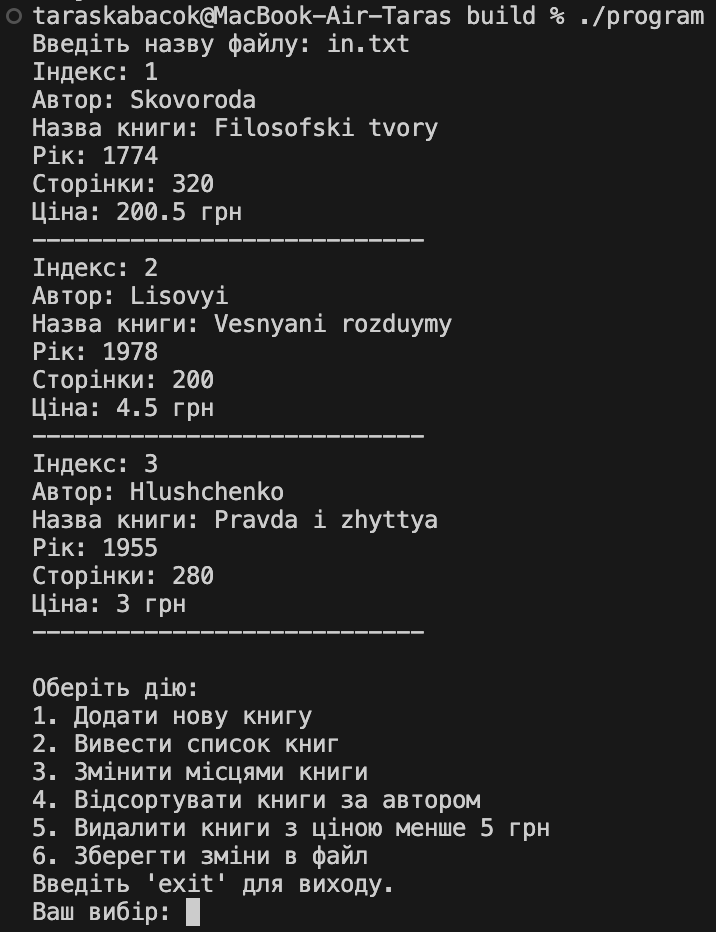


Рис. 7. Головне меню програми після відкриття вхідного файлу.

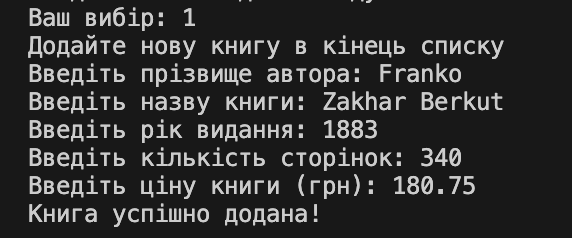


Рис. 8. Додавання нової книги в список.

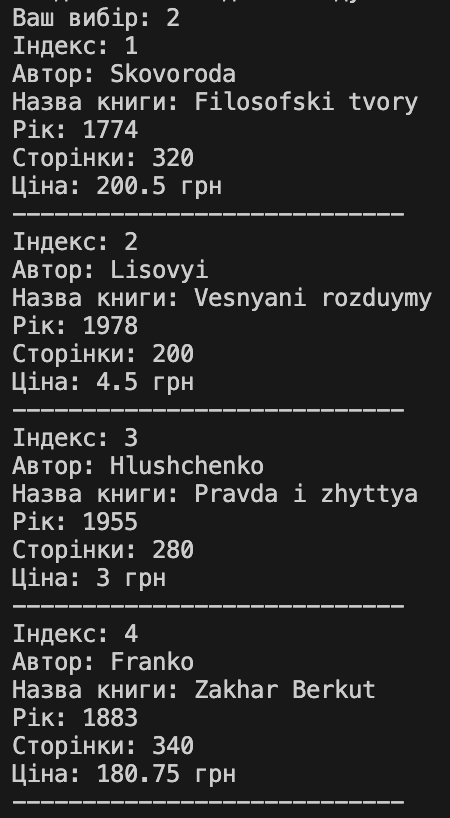


Рис. 9. Список після додавання книги в кінець.

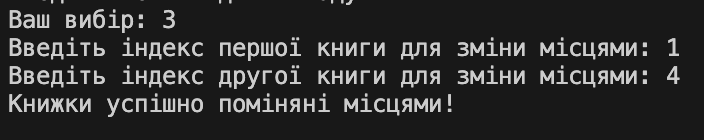


Рис. 10. Заміна книг місцями.

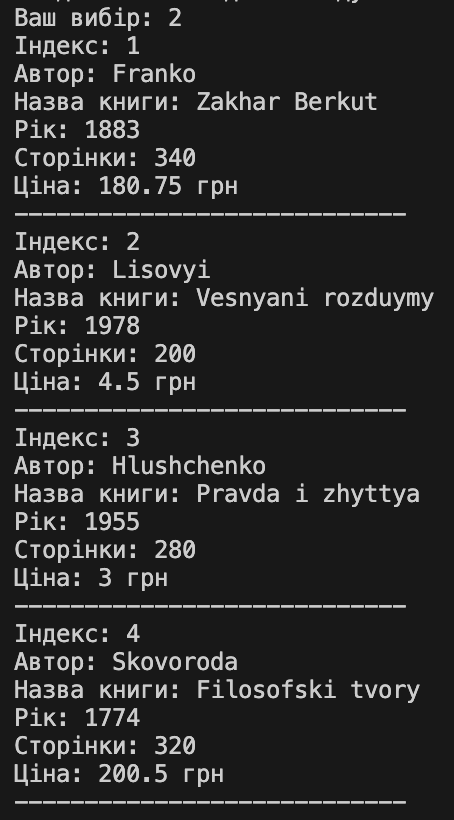


Рис. 11. Список після заміни 1 та 4 книг місцями.

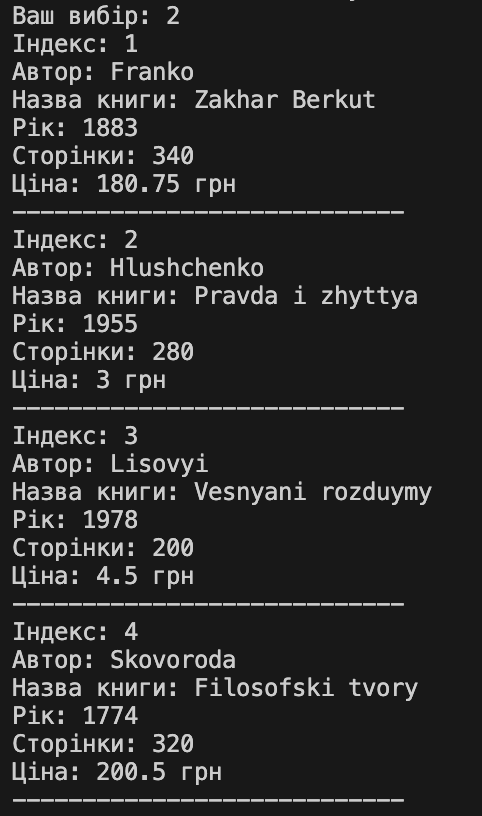


Рис. 12. Список після сортування.

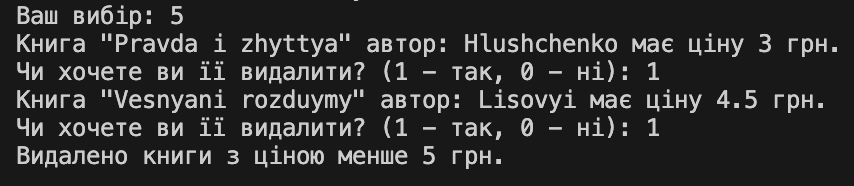


Рис. 13. Процес видалення книг з ціною менше 5 грн.

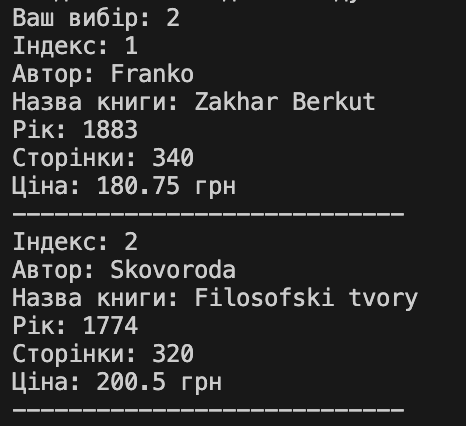


Рис. 14. Список після видалення книг з вартістю менше 5 грн.

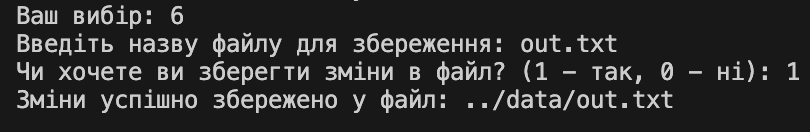


Рис. 15. Виведення програми про успішний запис даних в файл.

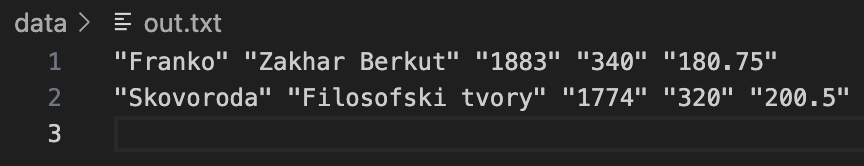
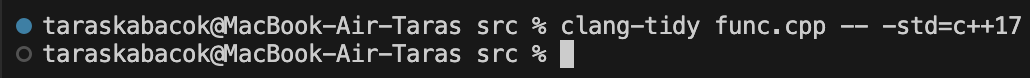


Рис. 16. Файл out.txt

Рис. 17. Розширення clang-tidy не вивело попереджень під час аналізу файлу func.cpp, що свідчить про відсутність виявлених проблем відповідно до обраних правил. Таким чином, вважається, що код є чистим і відповідає встановленим стандартам.

**Висновки**

Під час виконання лабораторної роботи я ознайомився з етапами проєктування та конструювання програмного забезпечення. Розробив блок-схеми для головної функції та підпрограм, зобразив структури даних, включно з вкладеними структурами та зв’язним списком, і побудував модульну схему програми. Також я розібрався з реалізацією багатьох функцій у зв’язному списку і принципами роботи з вказівниками. Дана лабораторна робота покращила моє розуміння структурованого програмування та візуалізації алгоритмів.