

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра Вычислительной техники**

**ОТЧЕТ**  
**по курсовой работе**  
**по дисциплине «Программирование»**  
**Тема: Разработка электронной картотеки**

Студентка гр. 4316

\_\_\_\_\_

Бастамова М.Р.

Преподаватель

\_\_\_\_\_

Аббас С.А.

Санкт-Петербург

2025

### **Цель работы.**

Целью является создание электронной картотеки книг для управления коллекцией с возможностью их добавления, редактирования, удаления, сортировки, поиска и сохранения в файл.

### **Постановка задачи и описание решения**

Разработать программу для управления коллекцией книг, включающую следующие функции:

1. Ввод и сохранение информации о книгах.
2. Редактирование данных книги.
3. Удаление книги по выбранному критерию.
4. Поиск книг по различным параметрам.
5. Сортировка коллекции по выбранным критериям.
6. Сохранение и загрузка данных из файла.
7. Вывод всех книг на экран.

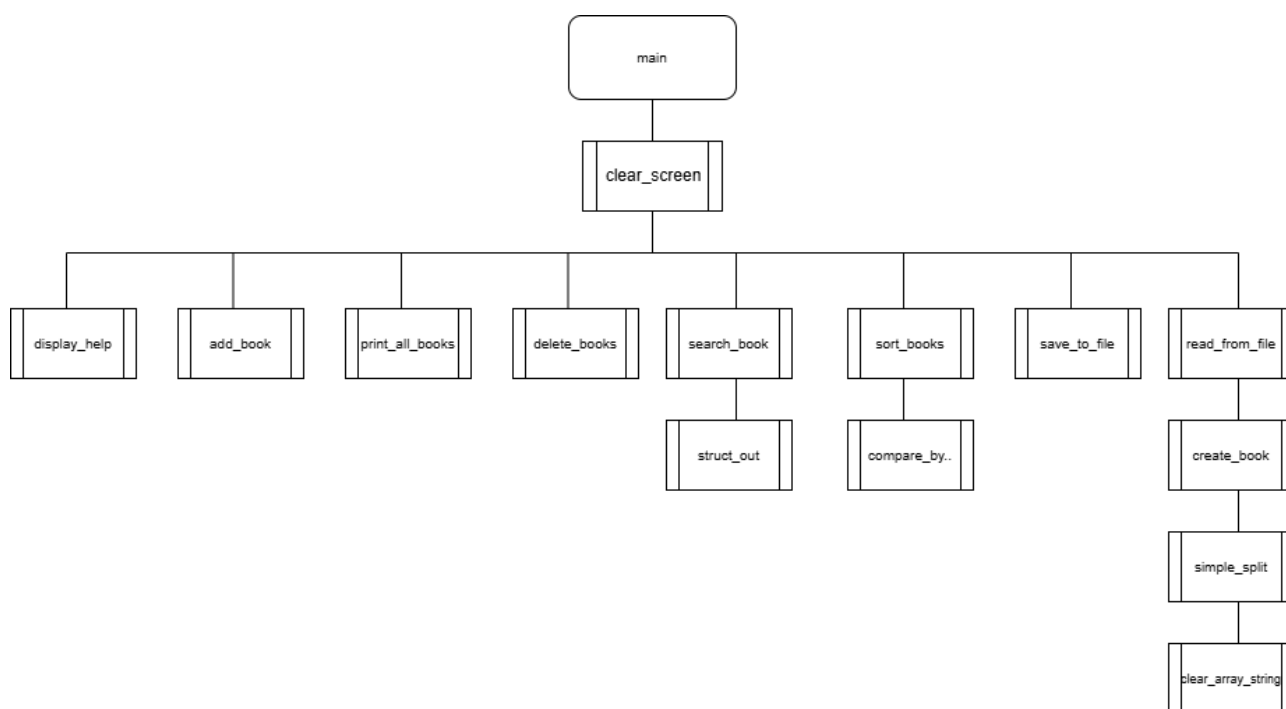
Для решения задачи используются: структуры данных для описания книги и организации списка; динамические массивы для хранения коллекции книг; функции для реализации каждой операции меню; функции для работы с файлами для сохранения и загрузки коллекции; меню для взаимодействия с пользователем.

### **Описание структур**

№	Поле	Тип	Назначение
<b>Book</b>			
1	name	char*	Название книги.
2	author	char*	Автор книги.
3	year	int	Год издания.
4	pages	int	Количество страниц.
5	average_rating	float	Средняя оценка книги.
6	price	float	Цена книги.
<b>Node</b>			

1	book	Book*	Указатель на структуру книги.
2	next	Node*	Указатель на следующий элемент списка.
<b>Head</b>			
1	cnt	int	Количество элементов в списке.
2	first	Node*	Указатель на первый элемент списка.
3	last	Node*	Указатель на последний элемент списка.

### Структура вызовов функций



### Описание функций

№	Наименование	Тип	Назначение
1	main	int	Главная функция программы. Управляет основным меню, вызывает другие функции в зависимости от выбора пользователя.
2	read_from_file	Book**	Читает данные о книгах из CSV-файла и возвращает массив указателей на структуры.

3	save_to_file	void	Сохраняет массив книг в CSV-файл.
4	make_head	Head*	Создает и инициализирует "голову" связного списка.
5	create_node	Node*	Создает новый узел списка с данными о книге.
6	create_book	Book*	Разбирает входную строку на отдельные компоненты, преобразует их в соответствующие типы данных и инициализирует поля структуры.
7	clear_string_array	void	Освобождает память, выделенную под массив строк.
8	simple_split	char**	Аналог strtok, разбивает строки.
9	print_header	void	Выводит на экран заголовок таблицы с книгами.
10	struct_out	void	Выводит данные одной книги в форматированной таблице.
11	add_book	Book*	Добавляет новую книгу в массив и возвращает указатель на неё.
12	sort_books	void	Сортирует книги по выбранному полю.
13	display_help	void	Выводит справочную информацию о командах программы.
14	print_all_books	void	Печатает список всех книг в виде таблицы.
15	edit_book	void	Редактирует выбранную книгу.
16	delete_books	void	Удаляет книги по выбранному полю.
17	search_books	void	Ищет книги по выбранному полю.
18	free_books	void	Освобождает память, выделенную под массив книг.
19	clear_screen	void	Очистка экрана.

## Описание переменных

№	Имя переменной	Тип	Назначение
<b>main()</b>			
1	books	Book**	массив указателей на книги
2	book_count	int	количество книг
3	choice	int	выбор действия пользователем
4	filename	char[256]	имя файла для чтения и сохранения
5	idx	int	индекс выбранной для редактирования книги
<b>read_from_file</b>			
1	filename	const char*	имя файла для чтения
2	sep	char	символ-разделитель
3	count	int*	указатель на количество загруженных книг
4	file	FILE*	для открытия файла
5	books	Book**	массив указателей на книги
6	line	char	строка для хранения текущей строки из файла
<b>create_book</b>			
1	string	char*	строка с данными книги
2	sep	char	символ-разделитель в строке
3	length	int	длина строки
4	fields	char**	массив строк — поля книги
5	b	Book*	указатель на новую структуру книги
<b>struct_out</b>			
1	b	Book*	указатель на книгу для вывода
2	id	int	идентификатор (номер) книги

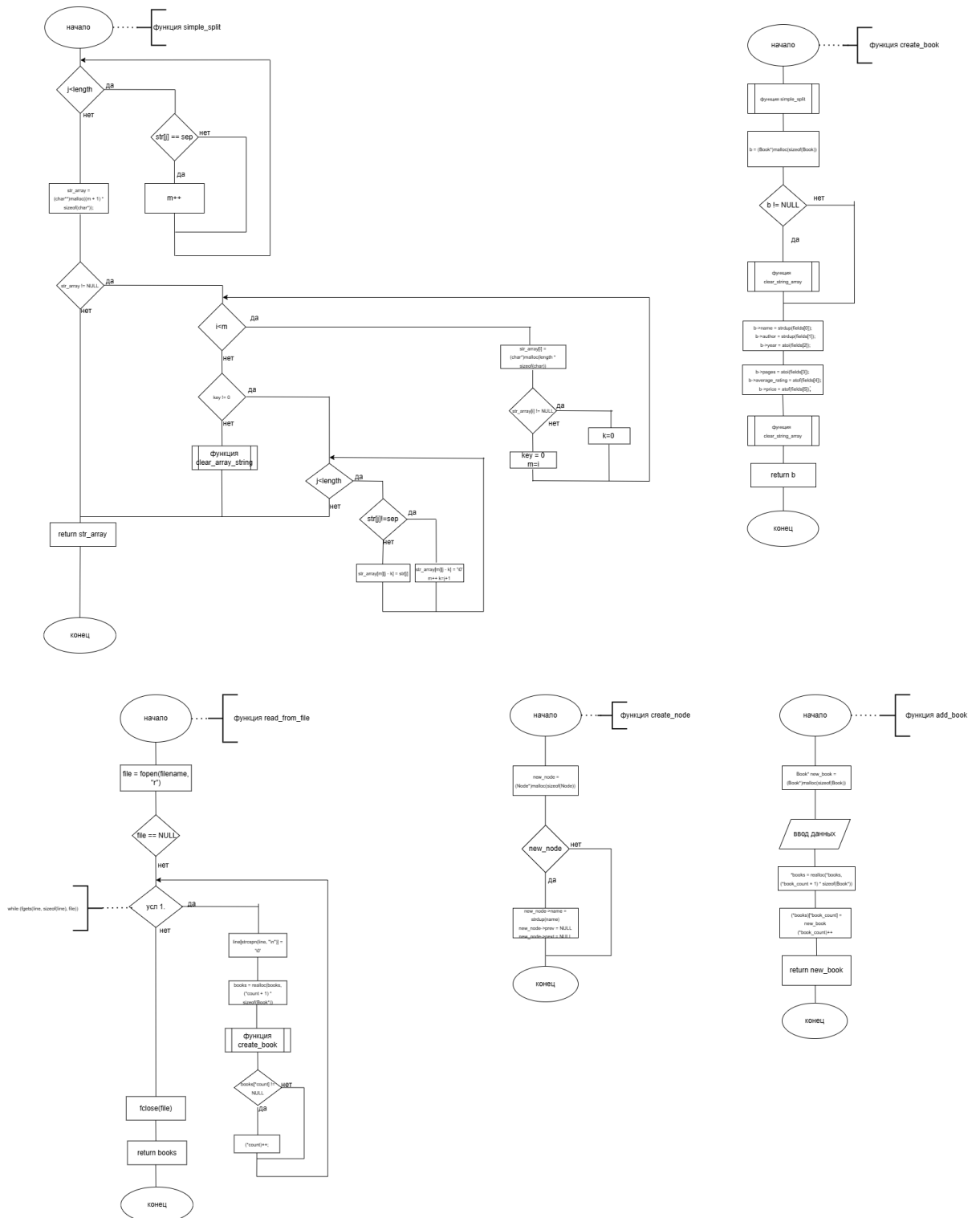
<b>add_book</b>			
1	books	Book***	указатель на массив книг
2	book_count	int*	указатель на количество книг
3	new_book	Book*	указатель на новую добавляемую книгу
4	buffer	char	буфер для временного хранения ввода
<b>edit_book</b>			
1	book	Book*	указатель на редактируемую книгу
2	buffer	char[256]	буфер для временного хранения ввода
3	temp_int	int	временная переменная для чисел
4	temp_float	float	временная переменная для чисел
<b>delete_books</b>			
1	books	Book***	указатель на массив книг
2	count	int*	указатель на количество книг
3	choice	int	выбор критерия удаления
4	str_val	char	строковое значение для сравнения при удалении
5	int_val	int	целочисленное значение для сравнения
6	i	int	индекс текущей книги в цикле
<b>search_books</b>			
1	books	Book**	массив указателей на книги
2	count	int	количество книг
3	choice	int	выбор критерия поиска
4	search_term	char	строковое значение для поиска
5	search_year	int	значение года для

			поиска
6	search_float	float	значение для поиска
7	i	int	индекс текущей книги в цикле
<b>compare_by</b>			
1	a	const void*	указатель на первый элемент сравнения
2	b	const void*	указатель на второй элемент сравнения
3	b1	const Book*	первый указатель на сравниваемую книгу
4	b2	const Book*	второй указатель на сравниваемую книгу
5	result	int	результат сравнения
<b>sort_books</b>			
1	books	Book**	массив указателей на книги
2	count	int	количество книг
3	choice	int	выбор критерия и направления сортировки
<b>save_to_file</b>			
1	books	Book**	массив указателей на книги
2	count	int	количество книг
3	filename	const char*	имя файла для сохранения
4	file	FILE*	файловый дескриптор для открытия файла
5	i	int	индекс текущей книги в цикле
<b>simple_split</b>			
1	str	char*	исходная строка для разделения
2	length	int	длина строки
3	sep	char	символ-разделитель
4	str_array	char**	массив строк — результат разделения

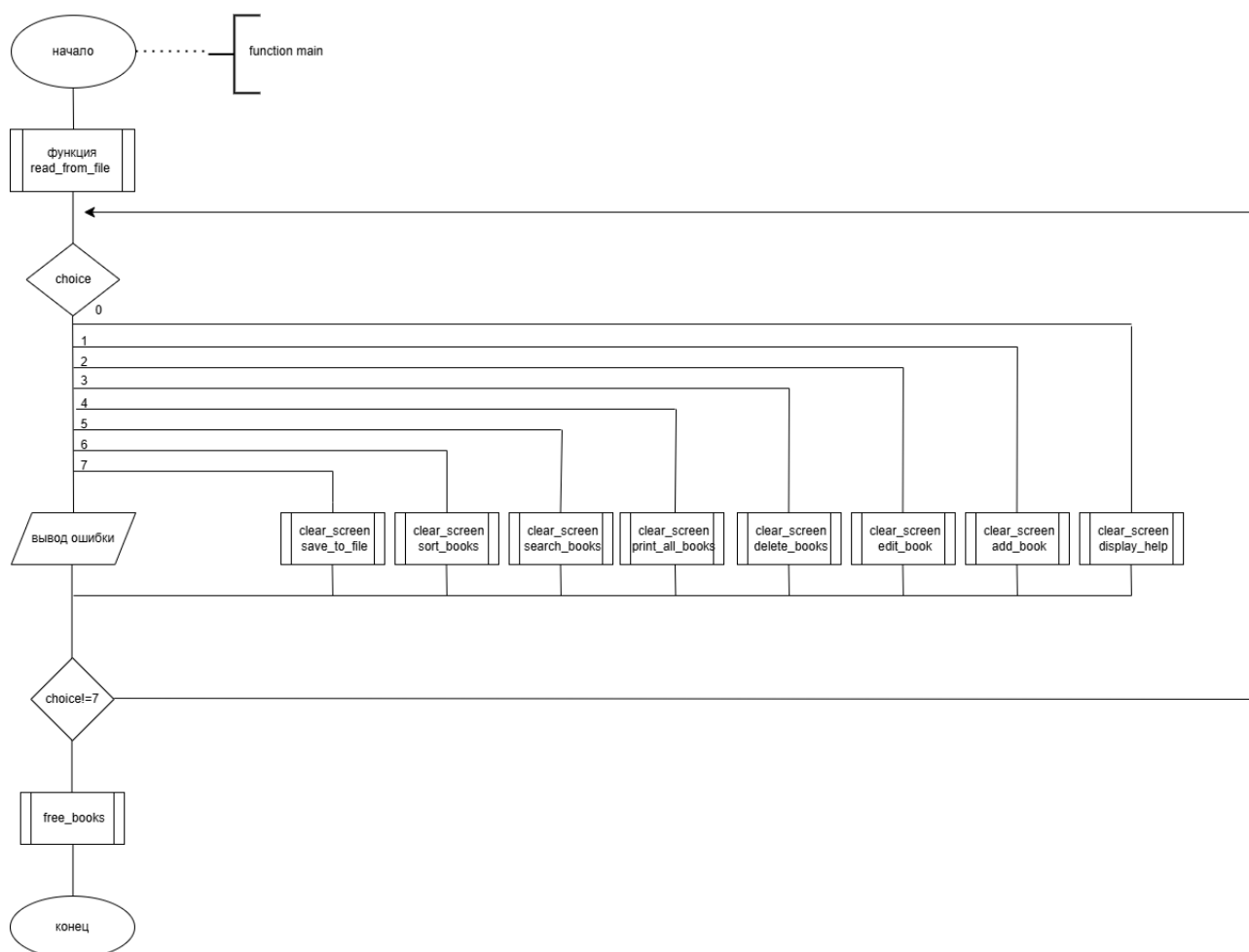
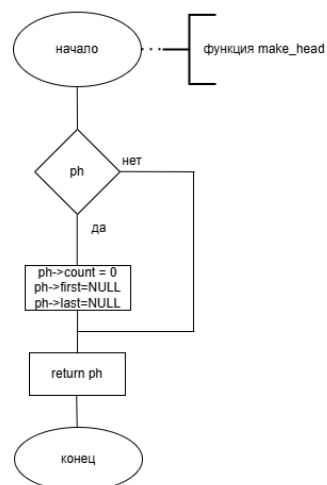
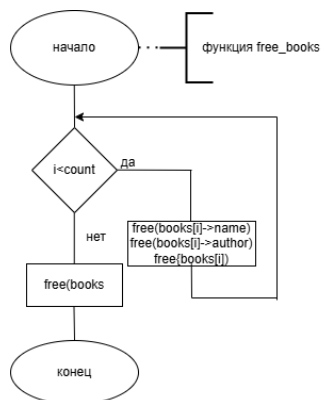
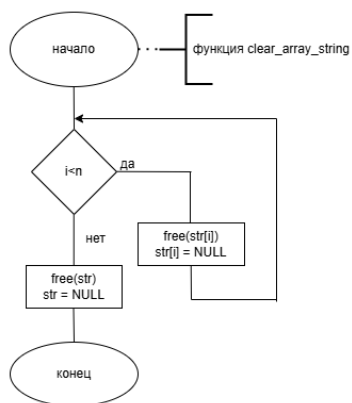
5	i, j, k, m	int	служебные индексы для цикла
6	key	int	флаг успешного выделения памяти
7	count	int	количество уже выделенных элементов
<b>free_books</b>			
1	books	Book**	массив указателей на книги
2	count	int	количество книг
3	i	int	индекс текущей книги в цикле
<b>create_node</b>			
1	new_book	Book*	указатель на книгу для вставки в узел
2	new_node	Node*	указатель на новый узел списка
<b>make_head</b>			
1	ph	Head*	указатель на новую голову списка

## Схемы алгоритмов

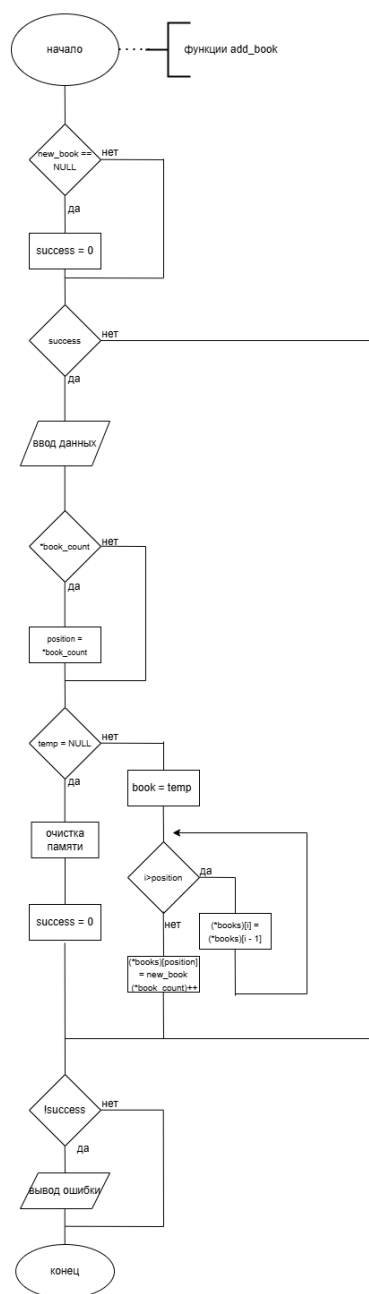
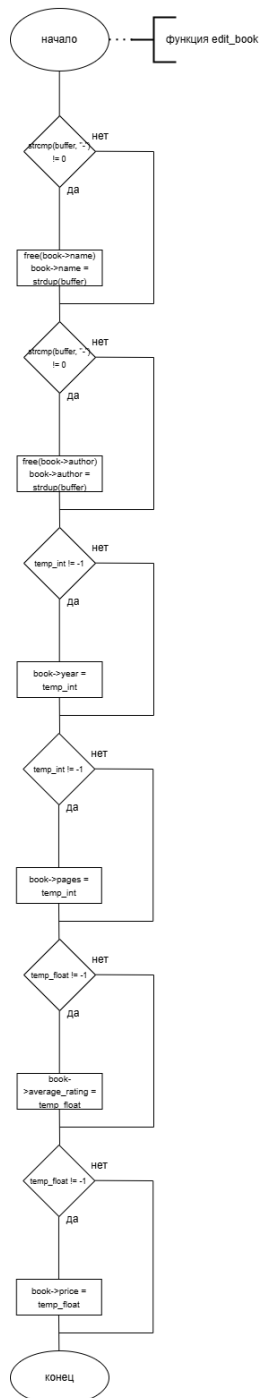
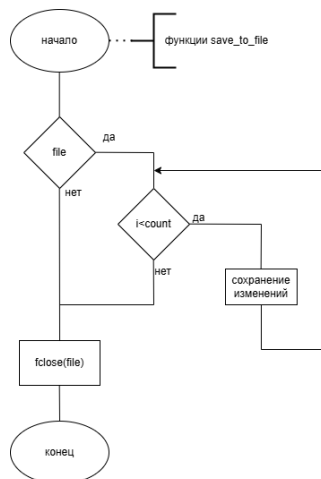
1. Функции **simple\_split**, **read\_from\_file**, **add\_book**, **create\_node**, **create\_book**



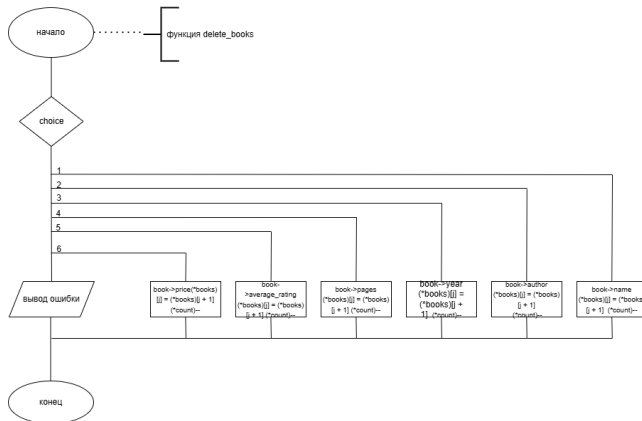
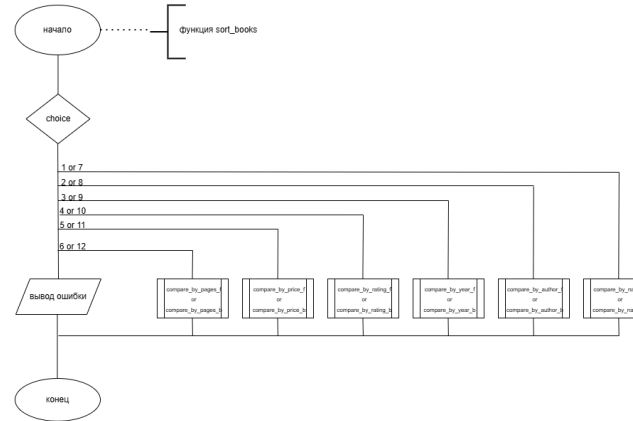
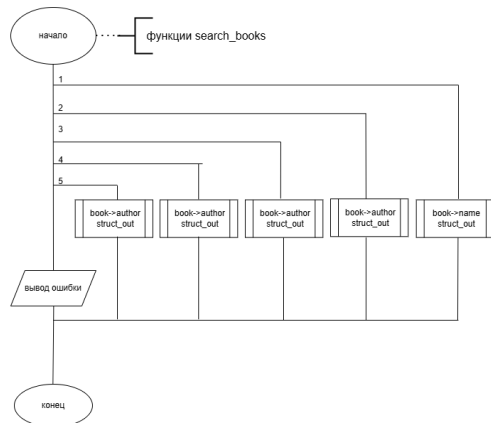
## 2. Функции free\_books, clear\_string\_array, make\_head, main



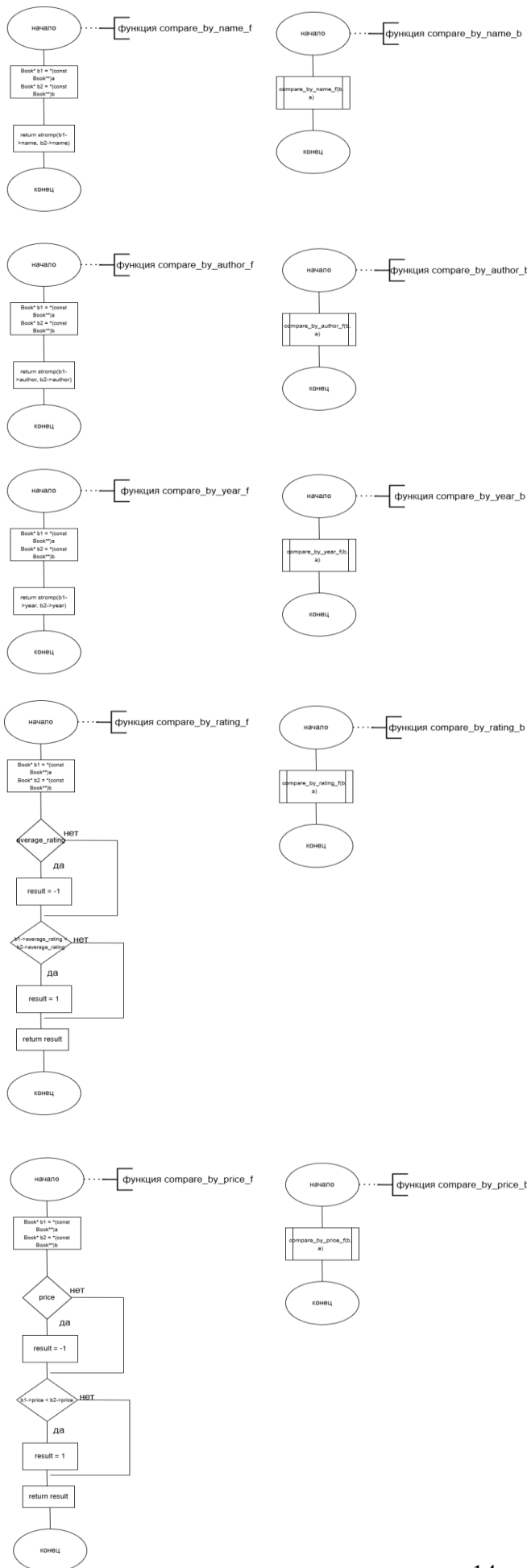
### 3. Функции `save_to_file`, `add_book`, `edit_book`



## 4. Функции sort\_books, delete\_books, search\_books



## 5. Функции compare\_by...



## Примеры работы программы, контрольные примеры:

### Пример № 1:

```
=== Help ===
This program is a console-based tool for managing book collections.
Quick commands:
1. Add    - Create new book entry
2. Edit   - Modify existing book
3. Delete - Remove by parameter
4. List   - Show all books
5. Search - Find by criteria
6. Sort   - Reorder catalog
7. Save   - Write changes to file & exit

Usage tips:
- All changes persist only after Save & Exit
- Press Enter after each command

Press Enter to continue...|
```

### Пример № 2:

```
Enter book name: All Quiet on the Western Front
Enter author: Erich Maria Remarque
Enter year: 1929
Enter pages: 250
Enter average rating: 4.9
Enter price: 10.00
Enter position to insert the book (0 to 24): 24
Book added successfully.

Press Enter to continue...|
```

ID	Book Name	Author	Year	Pages	Rating	Price
0	Animal Farm	George Orwell	1945	112	4.50	5.99
1	The Great Gatsby	F. Scott Fitzgerald	1925	180	4.30	10.99
2	Fahrenheit 451	Ray Bradbury	1953	194	4.20	8.20
3	The Catcher in the Rye	J.D. Salinger	1951	214	4.20	6.99
4	Sense and Sensibility	Jane Austen	1811	226	4.40	7.75
5	The Picture of Dorian Gray	Oscar Wilde	1890	254	4.10	6.50
6	Brave New World	Aldous Huxley	1932	268	4.20	9.00
7	Pride and Prejudice	Jane Austen	1813	279	4.50	9.50
8	Frankenstein	Mary Shelley	1818	280	4.10	6.99
9	To Kill a Mockingbird	Harper Lee	1960	281	4.80	7.99
10	The Hobbit	J.R.R. Tolkien	1937	310	4.60	8.50
11	1984	George Orwell	1949	328	4.70	8.99
12	Crime and Punishment	Fyodor Dostoevsky	1866	430	4.40	9.99
13	Little Women	Louisa May Alcott	1868	449	4.40	7.99
14	The Grapes of Wrath	John Steinbeck	1939	464	4.30	9.99
15	Great Expectations	Charles Dickens	1861	505	4.10	8.80
16	The Iliad	Homer	762	560	4.30	9.50
17	David Copperfield	Charles Dickens	1850	624	4.20	8.60
18	Moby-Dick	Herman Melville	1851	635	4.00	11.50
19	Don Quixote	Miguel de Cervantes	1605	863	4.20	12.50
20	The Lord of the Rings	J.R.R. Tolkien	1954	1178	4.90	15.99
21	War and Peace	Leo Tolstoy	1869	1225	4.10	12.99
22	The Count of Monte Cristo	Alexandre Dumas	1844	1276	4.50	13.00
23	Les Miserables	Victor Hugo	1862	1463	4.40	14.00
24	All Quiet on the Western Front	Erich Maria Remarque	1929	250	4.90	10.00

Press Enter to continue...

### Пример № 3:

5	The Picture of Dorian Gray	Oscar Wilde	1890	254	4.10	6.50
6	Brave New World	Aldous Huxley	1932	268	4.20	9.00
7	Pride and Prejudice	Jane Austen	1813	279	4.50	9.50
8	Frankenstein	Mary Shelley	1818	280	4.10	6.99
9	To Kill a Mockingbird	Harper Lee	1960	281	4.80	7.99
10	The Hobbit	J.R.R. Tolkien	1937	310	4.60	8.50
11	1984	George Orwell	1949	328	4.70	8.99
12	Crime and Punishment	Fyodor Dostoevsky	1866	430	4.40	9.99
13	Little Women	Louisa May Alcott	1868	449	4.40	7.99
14	The Grapes of Wrath	John Steinbeck	1939	464	4.30	9.99
15	Great Expectations	Charles Dickens	1861	505	4.10	8.80
16	The Iliad	Homer	762	560	4.30	9.50
17	David Copperfield	Charles Dickens	1850	624	4.20	8.60
18	Moby-Dick	Herman Melville	1851	635	4.00	11.50
19	Don Quixote	Miguel de Cervantes	1605	863	4.20	12.50
20	The Lord of the Rings	J.R.R. Tolkien	1954	1178	4.90	15.99
21	War and Peace	Leo Tolstoy	1869	1225	4.10	12.99
22	The Count of Monte Cristo	Alexandre Dumas	1844	1276	4.50	13.00
23	Les Miserables	Victor Hugo	1862	1463	4.40	14.00
24	All Quiet on the Western Front	Erich Maria Remarque	1929	250	4.90	10.00

Enter index of book to edit (0-based): 0  
Current book name: Animal Farm  
Enter new name (or '-' to skip): -  
Current author: George Orwell  
Enter new author (or '-' to skip): -  
Current year: 1945  
Enter new year (or -1 to skip): -  
Current pages: 112  
Enter new pages (or -1 to skip): 113  
Current average rating: 4.50  
Enter new average rating (or -1 to skip): 4.30  
Current price: 5.99  
Enter new price (or -1 to skip): 7.00

ID	Book Name	Author	Year	Pages	Rating	Price
0	Animal Farm	George Orwell	0	113	4.30	7.00
1	The Great Gatsby	F. Scott Fitzgerald	1925	180	4.30	10.99
2	Fahrenheit 451	Ray Bradbury	1953	194	4.20	8.20
3	The Catcher in the Rye	J.D. Salinger	1951	214	4.20	6.99
4	Sense and Sensibility	Jane Austen	1811	226	4.40	7.75
5	The Picture of Dorian Gray	Oscar Wilde	1890	254	4.10	6.50
6	Brave New World	Aldous Huxley	1932	268	4.20	9.00
7	Pride and Prejudice	Jane Austen	1813	279	4.50	9.50
8	Frankenstein	Mary Shelley	1818	280	4.10	6.99
9	To Kill a Mockingbird	Harper Lee	1960	281	4.80	7.99
10	The Hobbit	J.R.R. Tolkien	1937	310	4.60	8.50
11	1984	George Orwell	1949	328	4.70	8.99
12	Crime and Punishment	Fyodor Dostoevsky	1866	430	4.40	9.99
13	Little Women	Louisa May Alcott	1868	449	4.40	7.99
14	The Grapes of Wrath	John Steinbeck	1939	464	4.30	9.99
15	Great Expectations	Charles Dickens	1861	505	4.10	8.80
16	The Iliad	Homer	762	560	4.30	9.50
17	David Copperfield	Charles Dickens	1850	624	4.20	8.60
18	Moby-Dick	Herman Melville	1851	635	4.00	11.50
19	Don Quixote	Miguel de Cervantes	1605	863	4.20	12.50
20	The Lord of the Rings	J.R.R. Tolkien	1954	1178	4.90	15.99
21	War and Peace	Leo Tolstoy	1869	1225	4.10	12.99
22	The Count of Monte Cristo	Alexandre Dumas	1844	1276	4.50	13.00
23	Les Miserables	Victor Hugo	1862	1463	4.40	14.00
24	All Quiet on the Western Front	Erich Maria Remarque	1929	250	4.90	10.00

Press Enter to continue...

Пример №4:

```

Delete by:
1. Name
2. Author
3. Year
4. Pages
5. Rating
6. Price
Enter your choice: 3
Enter year: 1949
Book deleted.

Press Enter to continue...

```

ID	Book Name	Author	Year	Pages	Rating	Price
0	Animal Farm	George Orwell	0	113	4.30	7.00
1	The Great Gatsby	F. Scott Fitzgerald	1925	180	4.30	10.99
2	Fahrenheit 451	Ray Bradbury	1953	194	4.20	8.20
3	The Catcher in the Rye	J.D. Salinger	1951	214	4.20	6.99
4	Sense and Sensibility	Jane Austen	1811	226	4.40	7.75
5	The Picture of Dorian Gray	Oscar Wilde	1890	254	4.10	6.50
6	Brave New World	Aldous Huxley	1932	268	4.20	9.00
7	Pride and Prejudice	Jane Austen	1813	279	4.50	9.50
8	Frankenstein	Mary Shelley	1818	280	4.10	6.99
9	To Kill a Mockingbird	Harper Lee	1960	281	4.80	7.99
10	The Hobbit	J.R.R. Tolkien	1937	310	4.60	8.50
11	Crime and Punishment	Fyodor Dostoevsky	1866	430	4.40	9.99
12	Little Women	Louisa May Alcott	1868	449	4.40	7.99
13	The Grapes of Wrath	John Steinbeck	1939	464	4.30	9.99
14	Great Expectations	Charles Dickens	1861	505	4.10	8.80
15	The Iliad	Homer	762	560	4.30	9.50
16	David Copperfield	Charles Dickens	1850	624	4.20	8.60
17	Moby-Dick	Herman Melville	1851	635	4.00	11.50
18	Don Quixote	Miguel de Cervantes	1605	863	4.20	12.50
19	The Lord of the Rings	J.R.R. Tolkien	1954	1178	4.90	15.99
20	War and Peace	Leo Tolstoy	1869	1225	4.10	12.99
21	The Count of Monte Cristo	Alexandre Dumas	1844	1276	4.50	13.00
22	Les Miserables	Victor Hugo	1862	1463	4.40	14.00
23	All Quiet on the Western Front	Erich Maria Remarque	1929	250	4.90	10.00

Press Enter to continue...

### Пример № 5:

Search by:

1. Name
2. Author
3. Year
4. Average Rating
5. Price

Enter your choice: 5

Enter price to search: 9.99

11	Crime and Punishment	Fyodor Dostoevsky	1866	430	4.40	9.99
13	The Grapes of Wrath	John Steinbeck	1939	464	4.30	9.99

Press Enter to continue...

### Пример № 6:

Sort by:

Forward(1-5) or Backward(6-10):

- 1 or 7. Name
- 2 or 8. Author
- 3 or 9. Year
- 4 or 10. Average Rating
- 5 or 11. Price
- 6 or 12. Pages

Enter your choice: 7

Books sorted successfully.

Press Enter to continue...

ID	Book Name	Author	Year	Pages	Rating	Price
0	War and Peace	Leo Tolstoy	1869	1225	4.10	12.99
1	To Kill a Mockingbird	Harper Lee	1960	281	4.80	7.99
2	The Picture of Dorian Gray	Oscar Wilde	1890	254	4.10	6.50
3	The Lord of the Rings	J.R.R. Tolkien	1954	1178	4.90	15.99
4	The Iliad	Homer	762	560	4.30	9.50
5	The Hobbit	J.R.R. Tolkien	1937	310	4.60	8.50
6	The Great Gatsby	F. Scott Fitzgerald	1925	180	4.30	10.99
7	The Grapes of Wrath	John Steinbeck	1939	464	4.30	9.99
8	The Count of Monte Cristo	Alexandre Dumas	1844	1276	4.50	13.00
9	The Catcher in the Rye	J.D. Salinger	1951	214	4.20	6.99
10	Sense and Sensibility	Jane Austen	1811	226	4.40	7.75
11	Pride and Prejudice	Jane Austen	1813	279	4.50	9.50
12	Moby-Dick	Herman Melville	1851	635	4.00	11.50
13	Little Women	Louisa May Alcott	1868	449	4.40	7.99
14	Les Miserables	Victor Hugo	1862	1463	4.40	14.00
15	Great Expectations	Charles Dickens	1861	505	4.10	8.80
16	Frankenstein	Mary Shelley	1818	280	4.10	6.99
17	Fahrenheit 451	Ray Bradbury	1953	194	4.20	8.20
18	Don Quixote	Miguel de Cervantes	1605	863	4.20	12.50
19	David Copperfield	Charles Dickens	1850	624	4.20	8.60
20	Crime and Punishment	Fyodor Dostoevsky	1866	430	4.40	9.99
21	Brave New World	Aldous Huxley	1932	268	4.20	9.00
22	Animal Farm	George Orwell	0	113	4.30	7.00
23	All Quiet on the Western Front	Erich Maria Remarque	1929	250	4.90	10.00

Press Enter to continue...

Пример № 7 (ошибки):

Search by:

1. Name
2. Author
3. Year
4. Average Rating
5. Price

Enter your choice: 1

Enter book name: dsjdkj

Book not found.

Press Enter to continue...

Delete by:

1. Name
2. Author
3. Year
4. Pages
5. Rating
6. Price

Enter your choice: 3

Enter year: 199999

Book not found.

Press Enter to continue...

```

ID      Book Name      Author      Year      Pages      Rating      Price
-----
0      War and Peace      Leo Tolstoy      1869      1225      4.10      12.99
1      To Kill a Mockingbird      Harper Lee      1960      281      4.80      7.99
2      The Picture of Dorian Gray      Oscar Wilde      1890      254      4.10      6.50
3      The Lord of the Rings      J.R.R. Tolkien      1954      1178      4.90      15.99
4      The Iliad      Homer      762      560      4.30      9.50
5      The Hobbit      J.R.R. Tolkien      1937      310      4.60      8.50
6      The Great Gatsby      F. Scott Fitzgerald      1925      180      4.30      10.99
7      The Grapes of Wrath      John Steinbeck      1939      464      4.30      9.99
8      The Count of Monte Cristo      Alexandre Dumas      1844      1276      4.50      13.00
9      The Catcher in the Rye      J.D. Salinger      1951      214      4.20      6.99
10     Sense and Sensibility      Jane Austen      1811      226      4.40      7.75
11     Pride and Prejudice      Jane Austen      1813      279      4.50      9.50
12     Moby-Dick      Herman Melville      1851      635      4.00      11.50
13     Little Women      Louisa May Alcott      1868      449      4.40      7.99
14     Les Miserables      Victor Hugo      1862      1463      4.40      14.00
15     Great Expectations      Charles Dickens      1861      505      4.10      8.80
16     Frankenstein      Mary Shelley      1818      280      4.10      6.99
17     Fahrenheit 451      Ray Bradbury      1953      194      4.20      8.20
18     Don Quixote      Miguel de Cervantes      1605      863      4.20      12.50
19     David Copperfield      Charles Dickens      1850      624      4.20      8.60
20     Crime and Punishment      Fyodor Dostoevsky      1866      430      4.40      9.99
21     Brave New World      Aldous Huxley      1932      268      4.20      9.00
22     Animal Farm      George Orwell      0      113      4.30      7.00
23     All Quiet on the Western Front      Erich Maria Remarque      1929      250      4.90      10.00
Enter index of book to edit (0-based): 24
Invalid index.

Press Enter to continue...

```

## Текст программы

```

1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <windows.h>
5
6
7  struct book {
8      char* name;           /* name of the book */
9      char* author;         /* the author */
10     int year;              /* year of publication */
11     int pages;             /* number of pages */
12     float average_rating;  /* average book rating */
13     float price;          /* price of the book */
14 };
15
16 typedef struct book Book;
17
18 struct LNode {
19     Book* book; /* data area */
20     struct LNode* next;
21 };
22
23 struct LHead {
24     int cnt;
25     struct LNode* first;
26     struct LNode* last;
27 };
28
29 typedef struct LHead Head;
30 typedef struct LNode Node;
31
32 /* ----- Function prototypes ----- */
33
34 /* function to create a new list */
35 Head* make_head();
36
37 /* function to create a new node with a book */
38 Node* create_node(Book* new_book);
39
40 /* function to parse a CSV line into a Book structure */
41 Book* create_book(char* string, char sep);
42
43 /* function to clear array of string */
44 void clear_string_array(char **str, int n);
45
46 /* function to split string to array by separator */

```

```

46  /* function to split string to array by separator */
47  char **simple_split(char *str, int length, char sep);
48
49  /* function to print header string without data */
50  void print_header();
51
52  /* function to output structure fields on console */
53  void struct_out(Book *b, int id);
54
55  /* function to add a new book */
56  void add_book(Book*** books, int* book_count);
57
58  /* function to sort books */
59  void sort_books(Book** books, int count);
60
61  /* function to search for help */
62  void display_help();
63
64  /* function to display all books */
65  void print_all_books(Book** books, int count);
66
67  /* function to edit the information of a book */
68  void edit_book(Book* book);
69
70  /* function to delete a book from the list */
71  void delete_books(Book*** books, int* count);
72
73  /* function to search for books based on specific criteria */
74  void search_books(Book** books, int count);
75
76  /* function to save the list of books to a file */
77  void save_to_file(Book** books, int count, const char* filename);
78
79  /* function to load the list of books from a file */
80  Book** read_from_file(const char* filename, char sep, int* count);
81
82  /* function to free all allocated memory for the list */
83  void free_books(Book** books, int count);
84
85  /* function to clear screen */
86  void clear_screen();
87
88  Head* make_head() {
89      Head* ph = (Head*)malloc(sizeof(Head));
90      if (ph) {

```

```

90     if (ph) {
91         ph->cnt = 0;
92         ph->first = NULL;
93         ph->last = NULL;
94     }
95     return ph;
96 }
97
98 Node* create_node(Book* new_book) {
99     Node* new_node = (Node*)malloc(sizeof(Node));
100     if (new_node) {
101         new_node->book = new_book;
102         new_node->next = NULL;
103     }
104     return new_node;
105 }
106
107 Book* create_book(char* string, char sep) {
108     int length;
109     char **fields;
110     length = strlen(string);
111     fields = simple_split(string, length, sep);
112
113     Book* b = (Book*)malloc(sizeof(Book));
114
115     if (b) {
116         b->name = strdup(fields[0]);
117         b->author = strdup(fields[1]);
118         b->year = atoi(fields[2]);
119         b->pages = atoi(fields[3]);
120         b->average_rating = atof(fields[4]);
121         b->price = atof(fields[5]);
122     }
123     else
124     {
125         clear_string_array(fields, 6);
126     }
127     clear_string_array(fields, 6);
128     return b;
129 }
130
131 Book** read_from_file(const char* filename, char sep, int* count) {
132     FILE* f;
133     f = fopen(filename, "r");

```

```

132 FILE* file = fopen(filename, "r");
133 Book** books = NULL;
134 char line[512];
135 *count = 0;
136 if (file) {
137     while (fgets(line, sizeof(line), file)) {
138         line[strcspn(line, "\n")] = '\0'; /* remove newline character */
139         books = realloc(books, (*count + 1) * sizeof(Book*));
140         if ((books[*count] = create_book(line, sep))) {
141             (*count)++;
142         } else {
143             printf("Error creating book from line: %s\n", line);
144         }
145     }
146 } else {
147     printf("Error opening file!\n");
148 }
149 fclose(file);
150 return books;
151 }
152
153 void print_header() {
154     printf("| %-3s | %-30s | %-20s | %-4s | %-5s | %-6s | %-6s |\n",
155         "ID", "Book Name", "Author", "Year", "Pages", "Rating", "Price");
156     printf("|-----|-----|-----|-----|-----|-----|-----|\n");
157 }
158
159 void struct_out(Book *b, int id) {
160     printf("| %-3d | %-30s | %-20s | %-4d | %-5d | %-6.2f | %-6.2f |\n",
161         id, b->name, b->author, b->year, b->pages, b->average_rating, b->price);
162 }
163
164 void add_book(Book*** books, int* book_count) {
165     Book* new_book = NULL;
166     char buffer[256];
167     int position;
168     int success;
169     success = 1;
170
171     new_book = (Book*)malloc(sizeof(Book));
172     if (new_book == NULL) {
173         printf("Memory allocation failed.\n");
174         success = 0;
175     }
176 }

```

```

177     if (success) {
178         printf("Enter book name: ");
179         fgets(buffer, sizeof(buffer), stdin);
180         buffer[strcspn(buffer, "\n")] = '\0';
181         new_book->name = strdup(buffer);
182
183         printf("Enter author: ");
184         fgets(buffer, sizeof(buffer), stdin);
185         buffer[strcspn(buffer, "\n")] = '\0';
186         new_book->author = strdup(buffer);
187
188         printf("Enter year: ");
189         scanf("%d", &new_book->year);
190         getchar();
191
192         printf("Enter pages: ");
193         scanf("%d", &new_book->pages);
194         getchar();
195
196         printf("Enter average rating: ");
197         scanf("%f", &new_book->average_rating);
198         getchar();
199
200         printf("Enter price: ");
201         scanf("%f", &new_book->price);
202         getchar();
203
204         printf("Enter position to insert the book (0 to %d): ", *book_count);
205         scanf("%d", &position);
206         getchar();
207
208         if (position < 0 || position > *book_count) {
209             printf("Invalid position. Book will be added at the end.\n");
210             position = *book_count;
211         }
212
213         Book** temp = (Book**)realloc(*books, (*book_count + 1) * sizeof(Book*));
214         if (temp == NULL) {
215             printf("Memory reallocation failed.\n");
216             free(new_book->name);
217             free(new_book->author);
218             free(new_book);
219             success = 0;
220         } else {

```

```

221         *books = temp;
222
223         for (int i = *book_count; i > position; i--) {
224             (*books)[i] = (*books)[i - 1];
225         }
226
227         (*books)[position] = new_book;
228         (*book_count)++;
229         printf("Book added successfully.\n");
230     }
231 }
232
233 if (!success) {
234     printf("Failed to add the book.\n");
235 }
236 }
237
238
239 void display_help() {
240     printf("\n=== Help ===\n");
241     printf("This program is a console-based tool for managing book collections.\n");
242
243     printf("Quick commands:\n");
244     printf("1. Add    - Create new book entry\n");
245     printf("2. Edit   - Modify existing book\n");
246     printf("3. Delete - Remove by parameter\n");
247     printf("4. List   - Show all books\n");
248     printf("5. Search - Find by criteria\n");
249     printf("6. Sort   - Reorder catalog\n");
250     printf("7. Save   - Write changes to file & exit\n\n");
251
252     printf("Usage tips:\n");
253     printf("- All changes persist only after Save & Exit\n");
254     printf("- Press Enter after each command\n");
255 }
256
257 void print_all_books(Book** books, int count) {
258     print_header();
259     for (int i = 0; i < count; i++) {
260         struct_out(books[i], i);
261     }

```

```

262 }
263
264 void edit_book(Book* book) {
265     char buffer[256];
266     int temp_int;
267     float temp_float;
268
269     printf("Current book name: %s\n", book->name);
270     printf("Enter new name (or '-' to skip): ");
271     fgets(buffer, sizeof(buffer), stdin);
272     buffer[strcspn(buffer, "\n")] = '\0';
273     if (strcmp(buffer, "-") != 0) {
274         free(book->name);
275         book->name = strdup(buffer);
276     }
277
278     printf("Current author: %s\n", book->author);
279     printf("Enter new author (or '-' to skip): ");
280     fgets(buffer, sizeof(buffer), stdin);
281     buffer[strcspn(buffer, "\n")] = '\0';
282     if (strcmp(buffer, "-") != 0) {
283         free(book->author);
284         book->author = strdup(buffer);
285     }
286
287     printf("Current year: %d\n", book->year);
288     printf("Enter new year (or -1 to skip): ");
289     scanf("%d", &temp_int);
290     getchar();
291     if (temp_int != -1) book->year = temp_int;
292
293     printf("Current pages: %d\n", book->pages);
294     printf("Enter new pages (or -1 to skip): ");
295     scanf("%d", &temp_int);
296     getchar();
297     if (temp_int != -1) book->pages = temp_int;
298
299     printf("Current average rating: %.2f\n", book->average_rating);
300     printf("Enter new average rating (or -1 to skip): ");
301     scanf("%f", &temp_float);
302     getchar();
303     if (temp_float != -1) book->average_rating = temp_float;
304
305     printf("Current price: %.2f\n", book->price);

```

```

306     printf("Enter new price (or -1 to skip): ");
307     scanf("%f", &temp_float);
308     getchar();
309     if (temp_float != -1) book->price = temp_float;
310 }
311
312 void delete_books(Book ***books, int *count) {
313     int choice;
314     char str_val[256];
315     int int_val;
316     int i, found;
317     i = 0;
318     found = 0;
319
320     printf("Delete by:\n");
321     printf("1. Name\n");
322     printf("2. Author\n");
323     printf("3. Year\n");
324     printf("4. Pages\n");
325     printf("5. Rating\n");
326     printf("6. Price\n");
327     printf("Enter your choice: ");
328     scanf("%d", &choice);
329     getchar();
330
331     switch (choice) {
332         case 1:
333             printf("Enter book name: ");
334             fgets(str_val, sizeof(str_val), stdin);
335             str_val[strcspn(str_val, "\n")] = '\0';
336
337             while (i < *count) {
338                 if (strcmp((*books)[i]->name, str_val) == 0) {
339                     free((*books)[i]->name);
340                     free((*books)[i]->author);
341                     free((*books)[i]);
342                     for (int j = i; j < *count - 1; j++) {
343                         (*books)[j] = (*books)[j + 1];
344                     }
345                     (*count)--;
346                     *books = realloc(*books, (*count) * sizeof(Book*));
347                     found = 1;
348                     printf("Book deleted.\n");
349                 } else {

```

```

349         } else {
350             i++;
351         }
352     }
353     break;
354
355     case 2:
356         printf("Enter author: ");
357         fgets(str_val, sizeof(str_val), stdin);
358         str_val[strcspn(str_val, "\n")] = '\0';
359
360         while (i < *count) {
361             if (strcmp((*books)[i]->author, str_val) == 0) {
362                 free((*books)[i]->name);
363                 free((*books)[i]->author);
364                 free((*books)[i]);
365                 for (int j = i; j < *count - 1; j++) {
366                     (*books)[j] = (*books)[j + 1];
367                 }
368                 (*count)--;
369                 *books = realloc(*books, (*count) * sizeof(Book*));
370                 found = 1;
371                 printf("Book deleted.\n");
372             } else {
373                 i++;
374             }
375         }
376         break;
377
378     case 3:
379         printf("Enter year: ");
380         scanf("%d", &int_val);
381         getchar();
382
383         while (i < *count) {
384             if ((*books)[i]->year == int_val) {
385                 free((*books)[i]->name);
386                 free((*books)[i]->author);
387                 free((*books)[i]);
388                 for (int j = i; j < *count - 1; j++) {

```

```

384         if ((*books)[i]->year == int_val) {
388             for (int j = i; j < *count - 1; j++) {
389                 (*books)[j] = (*books)[j + 1];
390             }
391             (*count)--;
392             *books = realloc(*books, (*count) * sizeof(Book*));
393             found = 1;
394             printf("Book deleted.\n");
395         } else {
396             i++;
397         }
398     }
399     break;
400
401     case 4:
402         printf("Enter number of pages: ");
403         scanf("%d", &int_val);
404         getchar();
405
406         while (i < *count) {
407             if ((*books)[i]->pages == int_val) {
408                 free((*books)[i]->name);
409                 free((*books)[i]->author);
410                 free((*books)[i]);
411                 for (int j = i; j < *count - 1; j++) {
412                     (*books)[j] = (*books)[j + 1];
413                 }
414                 (*count)--;
415                 *books = realloc(*books, (*count) * sizeof(Book*));
416                 found = 1;
417                 printf("Book deleted.\n");
418             } else {
419                 i++;
420             }
421         }
422         break;
423
424     case 5:
425         printf("Enter rating: ");
426         scanf("%d", &int_val);
427         getchar();
428

```

```

429         while (i < *count) {
430             if ((*books)[i]->average_rating == int_val) {
431                 free((*books)[i]->name);
432                 free((*books)[i]->author);
433                 free((*books)[i]);
434                 for (int j = i; j < *count - 1; j++) {
435                     (*books)[j] = (*books)[j + 1];
436                 }
437                 (*count)--;
438                 *books = realloc(*books, (*count) * sizeof(Book*));
439                 found = 1;
440                 printf("Book deleted.\n");
441             } else {
442                 i++;
443             }
444         }
445         break;
446
447     case 6:
448         printf("Enter price: ");
449         scanf("%d", &int_val);
450         getchar();
451
452         while (i < *count) {
453             if ((*books)[i]->price == int_val) {
454                 free((*books)[i]->name);
455                 free((*books)[i]->author);
456                 free((*books)[i]);
457                 for (int j = i; j < *count - 1; j++) {
458                     (*books)[j] = (*books)[j + 1];
459                 }
460                 (*count)--;
461                 *books = realloc(*books, (*count) * sizeof(Book*));
462                 found = 1;
463                 printf("Book deleted.\n");
464             } else {
465                 i++;
466             }
467         }
468         break;
469     default: printf("Invalid choice.\n");

```

```

470         default: printf("Invalid choice.\n");
471     }
472     if (!found) {
473         printf("Book not found.\n");
474     }
475 }
476
477 void search_books(Book** books, int count) {
478     int choice;
479     char search_term[256];
480     int search_year;
481     float search_float;
482     int found;
483     found = 0;
484
485     printf("Search by:\n");
486     printf("1. Name\n");
487     printf("2. Author\n");
488     printf("3. Year\n");
489     printf("4. Average Rating\n");
490     printf("5. Price\n");
491     printf("Enter your choice: ");
492     scanf("%d", &choice);
493     getchar();
494
495     switch (choice) {
496     case 1:
497         printf("Enter book name: ");
498         fgets(search_term, sizeof(search_term), stdin);
499         search_term[strcspn(search_term, "\n")] = '\0';
500         for (int i = 0; i < count; i++) {
501             if (strstr(books[i]->name, search_term) != NULL) {
502                 struct_out(books[i], i);
503             }
504         }
505         found = 1;
506         break;
507     case 2:
508         printf("Enter an author: ");
509         fgets(search_term, sizeof(search_term), stdin);
510         search_term[strcspn(search_term, "\n")] = '\0';
511         for (int i = 0; i < count; i++) {

```

```

511         if (strstr(books[i]->author, search_term) != NULL) {
512             struct_out(books[i], i);
513         }
514         found = 1;
515     }
516     break;
517 case 3:
518     printf("Enter year to search: ");
519     scanf("%d", &search_year);
520     getchar();
521     for (int i = 0; i < count; i++) {
522         if (books[i]->year == search_year) {
523             struct_out(books[i], i);
524         }
525         found = 1;
526     }
527     break;
528 case 4:
529     printf("Enter rating to search: ");
530     fgets(search_term, sizeof(search_term), stdin);
531     search_term[strcspn(search_term, "\n")] = '\0';
532
533     float search_float = strtod(search_term, NULL);
534     for (int i = 0; i < count; i++) {
535         if (books[i]->average_rating == search_float) {
536             struct_out(books[i], i);
537         }
538         found = 1;
539     }
540     break;
541 case 5:
542     printf("Enter price to search: ");
543     fgets(search_term, sizeof(search_term), stdin);
544     search_term[strcspn(search_term, "\n")] = '\0';
545     search_float = strtod(search_term, NULL);
546     for (int i = 0; i < count; i++) {
547         if (books[i]->price == search_float) {
548             struct_out(books[i], i);
549         }
550         found = 1;
551     }

```

```

551     }
552     break;
553
554     default: printf("Invalid choice.\n");
555 }
556 if (found)
557 {
558     printf("Book not found.");
559 }
560 }
561
562 int compare_by_name_f(const void* a, const void* b) {
563     const Book* b1 = *(const Book**)a;
564     const Book* b2 = *(const Book**)b;
565     return strcmp(b1->name, b2->name);
566 }
567 int compare_by_name_b(const void* a, const void* b) {
568     return compare_by_name_f(b, a);
569 }
570
571 int compare_by_author_f(const void* a, const void* b) {
572     const Book* b1 = *(const Book**)a;
573     const Book* b2 = *(const Book**)b;
574     return strcmp(b1->author, b2->author);
575 }
576 int compare_by_author_b(const void* a, const void* b) {
577     return compare_by_author_f(b, a);
578 }
579
580 int compare_by_year_f(const void* a, const void* b) {
581     const Book* b1 = *(const Book**)a;
582     const Book* b2 = *(const Book**)b;
583     return b1->year - b2->year;
584 }
585 int compare_by_year_b(const void* a, const void* b) {
586     return compare_by_year_f(b, a);
587 }
588
589 int compare_by_rating_f(const void* a, const void* b) {
590     const Book* b1 = *(const Book**)a;
591     const Book* b2 = *(const Book**)b;
592     int result = 0;

```

```

593     if (b1->average_rating < b2->average_rating) {
594         result = -1;
595     }
596     if (b1->average_rating > b2->average_rating) {
597         result = 1;
598     }
599     return result;
600 }
601 int compare_by_rating_b(const void* a, const void* b) {
602     return compare_by_rating_f(b, a);
603 }
604
605 int compare_by_price_f(const void* a, const void* b) {
606     const Book* b1 = *(const Book**)a;
607     const Book* b2 = *(const Book**)b;
608     int result = 0;
609     if (b1->price < b2->price) {
610         result = -1;
611     }
612     if (b1->price > b2->price) {
613         result = 1;
614     }
615     return result;
616 }
617 int compare_by_price_b(const void* a, const void* b) {
618     return compare_by_price_f(b, a);
619 }
620
621 int compare_by_pages_b(const void* a, const void* b){
622     const Book* b1 = *(const Book**)a;
623     const Book* b2 = *(const Book**)b;
624     return b2->pages - b1->pages;
625 }
626
627 int compare_by_pages_f(const void* a, const void* b){
628     const Book* b1 = *(const Book**)a;
629     const Book* b2 = *(const Book**)b;
630     return b1->pages - b2->pages;
631 }
632
633 void sort_books(Book** books, int count) {
634     int choice;
635     printf("Sort by:\n");
636     printf("Forward(1-5) or Backward(6-10):\n");
637     printf("1 or 7. Name\n");

```

```

638     printf("2 or 8. Author\n");
639     printf("3 or 9. Year\n");
640     printf("4 or 10. Average Rating\n");
641     printf("5 or 11. Price\n");
642     printf("6 or 12. Pages\n");
643     printf("Enter your choice: ");
644     scanf("%d", &choice);
645     getchar();
646
647     switch (choice) {
648         case 1: qsort(books, count, sizeof(Book*), compare_by_name_f); break;
649         case 2: qsort(books, count, sizeof(Book*), compare_by_author_f); break;
650         case 3: qsort(books, count, sizeof(Book*), compare_by_year_f); break;
651         case 4: qsort(books, count, sizeof(Book*), compare_by_rating_f); break;
652         case 5: qsort(books, count, sizeof(Book*), compare_by_price_f); break;
653         case 6: qsort(books, count, sizeof(Book*), compare_by_pages_f); break;
654
655         case 7: qsort(books, count, sizeof(Book*), compare_by_name_b); break;
656         case 8: qsort(books, count, sizeof(Book*), compare_by_author_b); break;
657         case 9: qsort(books, count, sizeof(Book*), compare_by_year_b); break;
658         case 10: qsort(books, count, sizeof(Book*), compare_by_rating_b); break;
659         case 11: qsort(books, count, sizeof(Book*), compare_by_price_b); break;
660         case 12: qsort(books, count, sizeof(Book*), compare_by_pages_b); break;
661
662         default: printf("Invalid choice.\n");
663     }
664
665     printf("Books sorted successfully.\n");
666 }
667
668 void save_to_file(Book** books, int count, const char* filename) {
669     FILE* file = fopen(filename, "w");
670     if (file) {
671         for (int i = 0; i < count; i++) {
672             fprintf(file, "%s,%s,%d,%d,%.2f,%.2f\n",
673                 books[i]->name, books[i]->author, books[i]->year,
674                 books[i]->pages, books[i]->average_rating, books[i]->price);
675         }
676         printf("Books saved to file successfully.\n");
677     } else {
678         printf("Error opening file for writing.\n");
679     }
680     fclose(file);
681 }

```

```

682
683 void clear_string_array(char **str, int n) {
684     for (int i = 0; i < n; i++) {
685         free(str[i]);
686         str[i] = NULL;
687     }
688     free(str);
689     str = NULL;
690 }
691
692 char **simple_split(char *str, int length, char sep) {
693     char **str_array = NULL;
694     int i, j, k, m;
695     int key, count;
696
697     for (j = 0, m = 0; j < length; j++) {
698         if (str[j] == sep) m++;
699     }
700
701     key = 0;
702     str_array = (char**)malloc((m + 1) * sizeof(char*));
703     if (str_array != NULL) {
704         for (i = 0, count = 0; i <= m; i++, count++) {
705             str_array[i] = (char*)malloc(length * sizeof(char));
706             if (str_array[i] != NULL) key = 1;
707             else {
708                 key = 0;
709                 i = m;
710             }
711         }
712         if (key) {
713             k = 0;
714             for (i = 0; i <= m; i++) {
715                 for (j = 0; str[k] != sep && str[k] != '\0'; j++, k++) {
716                     str_array[i][j] = str[k];
717                 }
718                 str_array[i][j] = '\0';
719                 k++;
720             }
721         } else {
722             clear_string_array(str_array, count);
723         }
724     }
725     return str_array;

```

```

726 }
727
728 void free_books(Book** books, int count) {
729     for (int i = 0; i < count; i++) {
730         free(books[i]->name);
731         free(books[i]->author);
732         free(books[i]);
733     }
734     free(books);
735 }
736
737 void clear_screen() {
738     #ifdef _WIN32
739         system("cls");
740     #else
741         system("clear");
742     #endif
743 }
744
745 int main() {
746     Book** books = NULL;
747     int book_count = 0;
748     int choice;
749     char filename[256];
750
751     printf("Enter the filename: ");
752     scanf("%255s", filename);
753     getchar();
754
755     books = read_from_file(filename, ',', &book_count);
756
757     do {
758         clear_screen();
759         printf("\n--- Main Menu ---\n");
760         printf("0: Help\n");
761         printf("1: Add book\n");
762         printf("2: Edit book\n");
763         printf("3: Delete book\n");
764         printf("4: Print all books\n");
765         printf("5: Search books\n");
766         printf("6: Sort books\n");
767         printf("7: Save and exit\n");
768         printf("Enter your choice: ");
769         scanf("%d", &choice);

```

```

770     getchar();
771
772     switch (choice) {
773     case 0:
774         clear_screen();
775         display_help();
776         break;
777     case 1:
778         clear_screen();
779         add_book(&books, &book_count);
780         break;
781     case 2: {
782         clear_screen();
783         int idx;
784         print_all_books(books, book_count);
785         printf("Enter index of book to edit (0-based): ");
786         scanf("%d", &idx);
787         getchar();
788         if (idx >= 0 && idx < book_count) {
789             edit_book(books[idx]);
790         } else {
791             printf("Invalid index.\n");
792         }
793         break;
794     }
795     case 3: {
796         clear_screen();
797         delete_books(&books, &book_count);
798         break;
799     }
800     case 4:
801         clear_screen();
802         print_all_books(books, book_count);
803         break;
804     case 5:
805         clear_screen();
806         search_books(books, book_count);
807         break;
808     case 6:
809         clear_screen();
810         sort_books(books, book_count);

```

```

811         break;
812     case 7:
813         clear_screen();
814         save_to_file(books, book_count, filename);
815         break;
816     default:
817         printf("Invalid choice.\n");
818     }
819
820     if (choice != 7) {
821         printf("\nPress Enter to continue...");
822         while (getchar() != '\n');
823     }
824 } while (choice != 7);
825
826 free_books(books, book_count);
827 printf("Exiting program.\n");
828 return 0;
829 }

```

## Заключение

В ходе работы над задачей были использованы следующие функции и заголовочные файлы:

1. `#include <stdio.h>`
2. `#include <stdlib.h>`
3. `#include <windows.h>`

Разработан алгоритм с использованием функций:

- `read_from_file`, `save_to_file` – для работы с файлом (чтение и сохранение)
- `free_books` – для освобождения памяти
- `clear_screen`, `clear_string_array` – для очистки экрана и строк
- `search_books` – для поиска книг
- `delete_books` – для удаления книг
- `edit_book` – для изменения книг
- `print_all_books` – для вывода списка книг
- `display_help` – для вывода справки
- `sort_books` – для сортировки книг (в две стороны)
- `add_book` – для добавления книг в список
- `struct_out`, `print_header` – для вывода таблицы

- `simple_split` – аналог `strtok`
- `create_book`, `create_node`, `make_head` – для работы с односвязным списком

Реализована программа, включающая меню для выбора режима работы и обработку возможных ошибок (таких как ошибка чтения файла), а также проведено тестирование программы на различных входных данных, чтобы убедиться в корректной работе функций обработки текста и записи результатов.