|  |  |
| --- | --- |
| ­Изображение выглядит как герб, эмблема, символ, нашивка  Автоматически созданное описание | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

# КАФЕДРА «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2 ПО ДИСЦИПЛИНЕ:**

**ТИПЫ И СТРУКТУРЫ ДАННЫХ**

***«Обработка таблиц»***

Студент **Демирел Э.А.**

**Вариант 1**

Группа **ИУ7-31Б**

Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

|  |  |
| --- | --- |
| Студент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Демирел Э.А.** |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Барышникова М.Ю.** |

**2024**

**Условие задачи**

Создать таблицу, содержащую не менее 40-ка записей (тип – запись с вариантами (объединениями)). Упорядочить данные в ней по возрастанию ключей, двумя алгоритмами сортировки, где ключ – любое невариантное поле (по выбору программиста), используя: а) саму таблицу, б) массив ключей. Возможность добавления и удаления записей в ручном режиме, просмотр таблицы, просмотр таблицы в порядке расположения таблицы ключей обязательна. Осуществить поиск информации по варианту.

**Техническое задание**

Список книг, максимум 1000 записей, в программе уже содержится 40 записей в статическом массиве.

Информация о книге: фамилия автора, название книги, издательство, количество страниц, вид литературы:

1. Техническая:
2. Отрасль
3. Отечественная/переводная
4. Год издания
5. Художественная:
6. Тип литературы (роман, пьеса, поэзия)
7. Детская:
8. Минимальный возраст
9. Тип детской литературы (стихи, сказки)

Программа предоставляет пользователю интерактивное меню:

1. Добавить книгу

2. Удалить книгу по названию

3. Просмотреть список книг

4. Просмотреть таблицу ключей

5. Просмотреть книги, отсортированные по автору (обычная сортировка)

6. Измерить производительность сортировок

7. Просмотреть книги, отсортированные по автору (таблица авторов)

8. Показать все романы автора

9. Отсортировать массив индексов

10. Вывести обычную таблицу по таблице ключей

11. Выход

**Входные данные:**

Число, соответствующее пункту меню и желаемому действию, имя книги, автор книг, издательство книги, вид книги. При разных видах книг также вводятся дополнительные данные.

**Выходные данные:**

Таблица (список) книг, таблица ключей, отсортированные книги по автору с помощью разных реализаций, список романов конкретного автора, ошибки (при наличии).

**Способ обращения к программе**

Обращение к программе происходит через запуск исполняемого файла app.exe через терминал или в проводнике. После чего пользователем выбирается желаемое действие посредством ввода через клавиатуру.

**Аварийные ситуации**

1. Ошибки при взаимодействии с меню: нечисловой ввод, пустой ввод, числа, не входящие в диапазон [1;9].

2. Ошибки при добавлении книги (фамилии, названия, издательства): пустой ввод.

3. Ошибки при добавлении книги (количество страниц): пустой ввод, нечисловой ввод, отрицательное или нулевое количество страниц.

4. Ошибки при добавлении книги (тип литературы): пустой ввод, нечисловой ввод, число вне диапазона [0;2].

5. Ошибки при добавлении книги, если тип литературы – технический:

пустой ввод в поле отрасли; нечисловой ввод, пустой ввод или число вне диапазона [0;1] в поле происхождения.

6. Ошибки при добавлении книги, если тип литературы – художественный: нечисловой ввод, пустой ввод, число вне диапазона [1;3.

7. Ошибки при добавлении книги, если тип литературы – детский: пустой ввод, нечисловой ввод, отрицательное число в поле минимального возраста; пустой ввод, нечисловой ввод, число, не входящее в диапазон [1;2] в поле жанра.

8. Ошибки при добавлении книги в общем случае: массив уже переполнен.

9. Ошибки при удалении книги: не существующая книга.

10. Ошибки при выводе всех романов автора: у автора нет романов, такого автора не существует.

**Описание структур данных**

typedef enum

{

TECHNICAL, // техническая литература

FICTION, // художественная

CHILDREN // детская

} BookType;

Возможные типы литературы.

typedef struct

{

char branch[50]; // отрасль

char origin[50]; // отечественная / переводная

int year; // год издания

} Technical;

Структура для хранения данных технического типа литературы.

typedef struct

{

char genre[30]; // жанр

} Fiction;

Структура для хранения данных художественного типа литературы.

typedef struct

{

int min\_age; // мин. возраст

char genre[30]; // жанр

} Children;

Структура для хранения данных детского типа литературы.

typedef union

{

Technical tech;

Fiction fiction;

Children children;

} BookDetails;

Объединение, предназначенное для хранения типа литературы книги. Объединение позволяет экономить память, не выделяя её для каждого типа литературы.

typedef struct

{

char author[50]; // автор

char title[100]; // название

char publisher[50]; // издатель

int pages; // кол-во страниц

BookType type; // тип книги

BookDetails details; // детали

} Book;

Структура для хранения объекта книги.

typedef struct

{

int index;

char author[50];

} AuthorIndex;

Структура для хранения пары ключ – автор в таблице ключей.

**Алгоритм программы**

1. Вводится пункт меню.

2. Если выбран первый пункт:

* Проверяется свободное место в массиве.
* Вводится фамилия автора, название книги, издательство и количество страниц.
* Выбирается тип литературы.
* Вводится дополнительная информация в соответствие с типом литературы.
* Книга добавляется в конец массива.

3. Если выбран второй пункт:

* Вводится название книги для удаления
* Проверяется наличие этой книги
* Книга удаляется

4. Если выбран третий или четвертый пункт:

* Выводится список книг (или пар ключ – автор) построчно.

5. Если выбран пятый или седьмой пункт:

* Массив книг сортируется в соответствии с вариантом сортировки
* Отсортированный массив выводится на экран

6. Если выбран шестой пункт:

* Производится сортировка массива книг и ключей при разном количестве записей разными видами сортировок.
* Производится замер времени и подсчёт среднего.
* Результат для каждого количества записей и вида сортировок выводится на экран.

7. Если выбран восьмой пункт:

* Вводится фамилия искомого автора
* Ищутся все романы данного автора
* Результат поиска выводится на экран

8. Если выбран девятый пункт:

* Массив индексов сортируется
* Отсортированный массив индексов выводится на экран

9. Если выбран десятый пункт:

* Выводится список книг и их идентификатор в таблице ключей.

10. Если выбран одиннадцатый пункт:

* Программа завершается.

**Описание реализованных функций**

**generate\_random\_string**

Генерирует случайную строку заданной длины, используя буквы английского алфавита.

void generate\_random\_string(char \*str, size\_t size);

**generate\_random\_book**

Создаёт случайную книгу, заполняя её поля случайными данными.

void generate\_random\_book(Book \*book);

**add\_book**

Добавляет новую книгу в массив книг, запрашивая данные у пользователя.

void add\_book(Book books[], int \*book\_count);

**delete\_book\_by\_title**

Удаляет книгу из массива по её названию.

void delete\_book\_by\_title(Book books[], int \*book\_count, const char \*title);

**display\_books**

Выводит список всех книг в таблице.

void display\_books(Book books[], int book\_count);

**display\_sorted\_books\_by\_author**

Сортирует книги по имени автора (пузырьком и быстрой сортировкой) и выводит их.

void display\_sorted\_books\_by\_author(Book books[], int book\_count);

**create\_author\_index**

Создаёт таблицу индексов для книг на основе имени автора.

void create\_author\_index(const Book books[], AuthorIndex author\_table[], int book\_count);

**display\_sorted\_books\_using\_author\_table**

Сортирует книги с использованием таблицы авторов и выводит отсортированный список.

void display\_sorted\_books\_using\_author\_table(Book books[], int book\_count, AuthorIndex author\_table[]);

**display\_author\_fiction\_books**

Находит и выводит все художественные книги конкретного автора.

void display\_author\_fiction\_books(Book books[], int book\_count);

**quicksort\_books**

Выполняет быструю сортировку массива книг по автору.

void quicksort\_books(Book books[], int count);

**compare\_books\_by\_author**

Сравнивает книги по автору для использования в сортировке.

int compare\_books\_by\_author(const void \*a, const void \*b);

**quicksort\_author\_table**

Выполняет быструю сортировку таблицы авторов.

void quicksort\_author\_table(AuthorIndex author\_table[], int count);

**compare\_author\_index**

Сравнивает записи таблицы авторов по имени автора.

int compare\_author\_index(const void \*a, const void \*b);

**measure\_sorting\_performance**

Измеряет производительность разных методов сортировки массива книг.

void measure\_sorting\_performance(void);

**print\_author\_table**

Выводит таблицу авторов, содержащую ключи (индексы) и имена.

void print\_author\_table(const AuthorIndex author\_table[], int count);

**print\_table\_header**

Выводит заголовок таблицы для списка книг.

void print\_table\_header();

**print\_book\_table\_row**

Выводит строку таблицы для одной книги.

void print\_book\_table\_row(const Book \*book);

**print\_table\_header\_id**

Выводит заголовок таблицы для списка книг с индексами.

void print\_table\_header\_id();

**print\_book\_table\_row\_id**

Выводит строку таблицы для одной книги, добавляя индекс.

void print\_book\_table\_row\_id(const Book \*book, int id);

**bubble\_sort\_books\_by\_author**

Сортирует книги по автору методом пузырька.

void bubble\_sort\_books\_by\_author(Book books[], int book\_count);

**bubble\_sort\_books\_by\_author\_table**

Сортирует таблицу авторов методом пузырька.

void bubble\_sort\_books\_by\_author\_table(AuthorIndex author\_table[], int book\_count);

**show\_table\_by\_key\_table**

Выводит таблицу книг, используя таблицу ключей (индексов авторов).

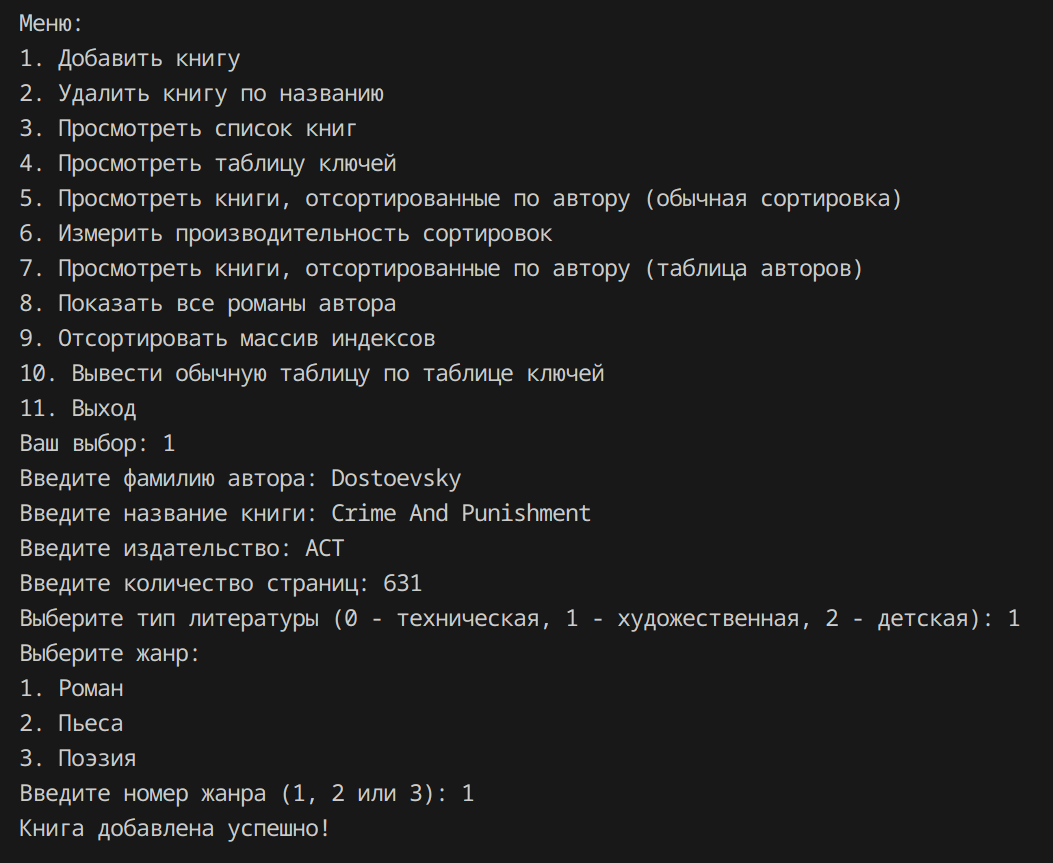
void show\_table\_by\_key\_table(AuthorIndex \*author\_table, Book \*books, int book\_count);

**Примеры работы программы**

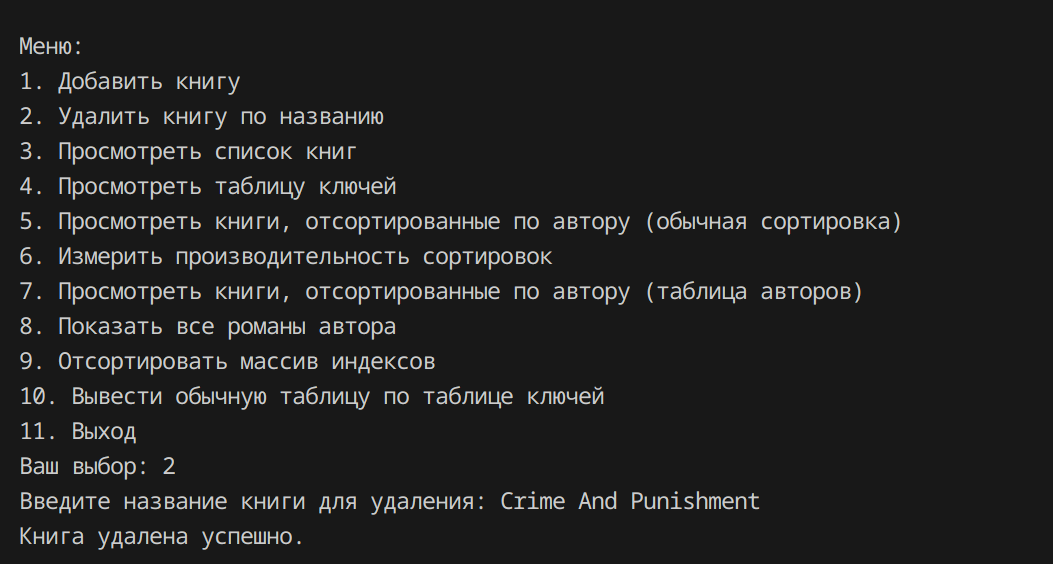
Примеры работы программы в разных ситуациях.

**Позитивные тесты:**

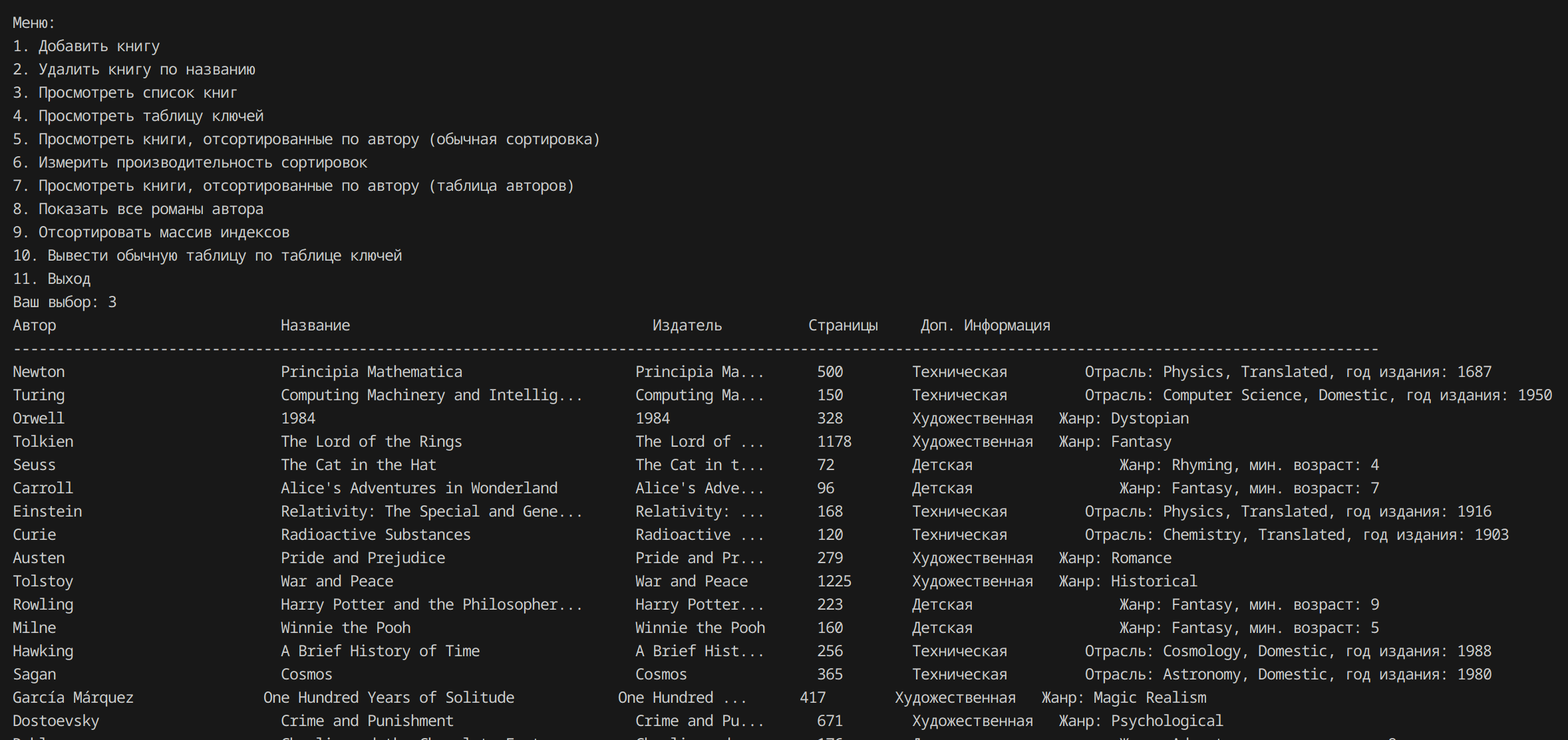
1. Добавление книги (массив не переполнен, данные вводятся верно).



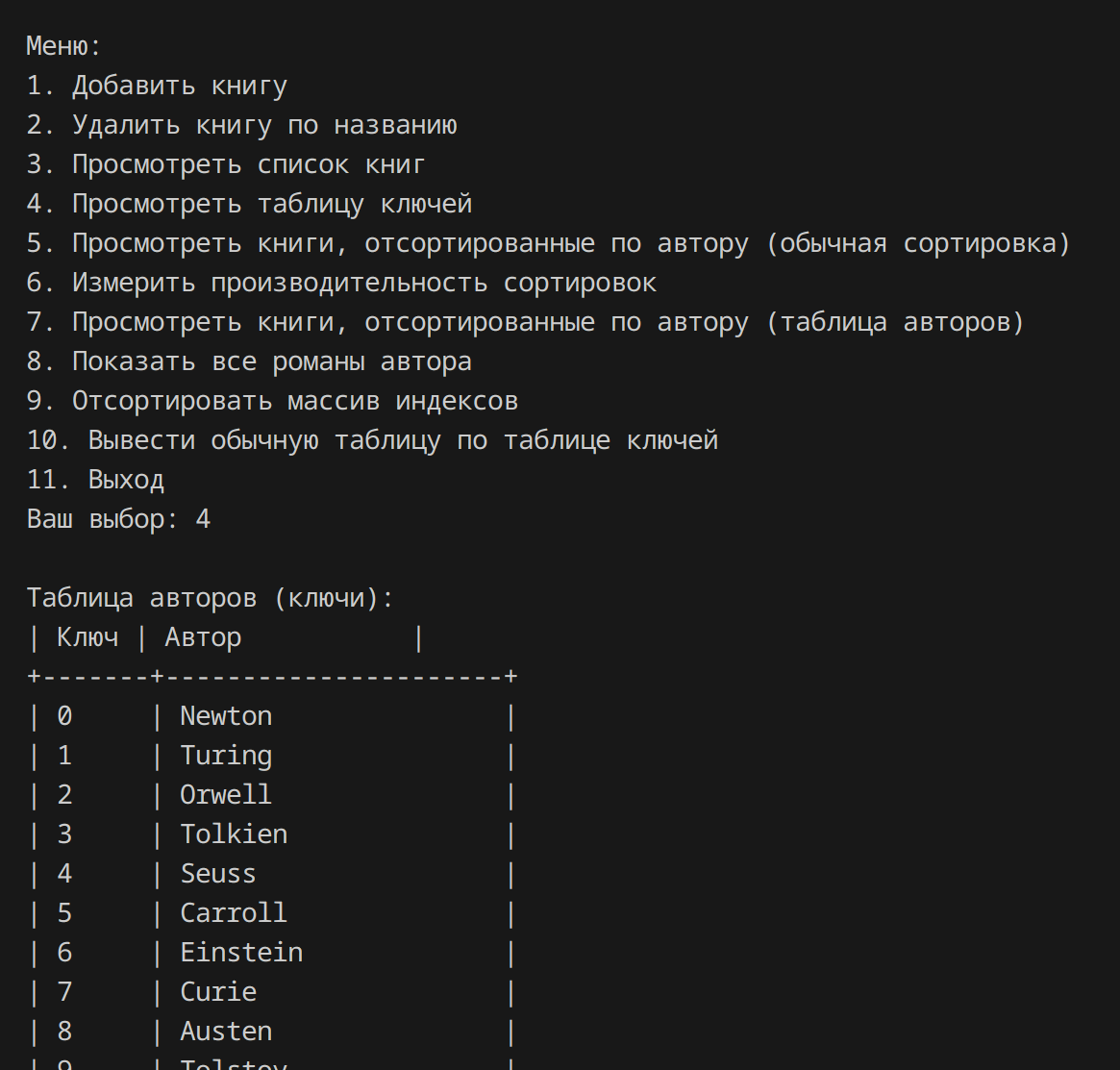
2. Удаление существующей книги, верный ввод



3. Просмотр списка книг



4. Просмотр таблицы ключей

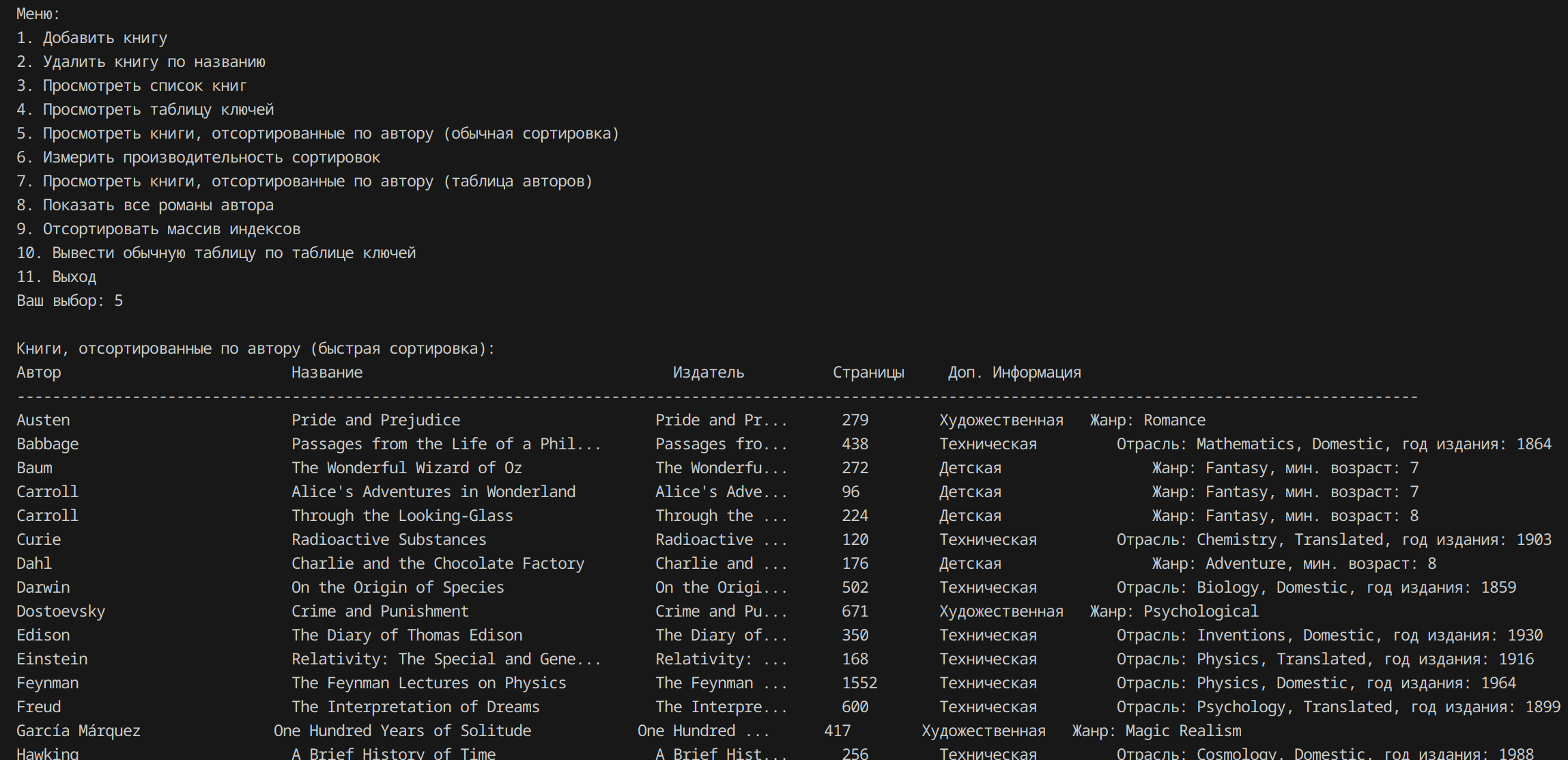


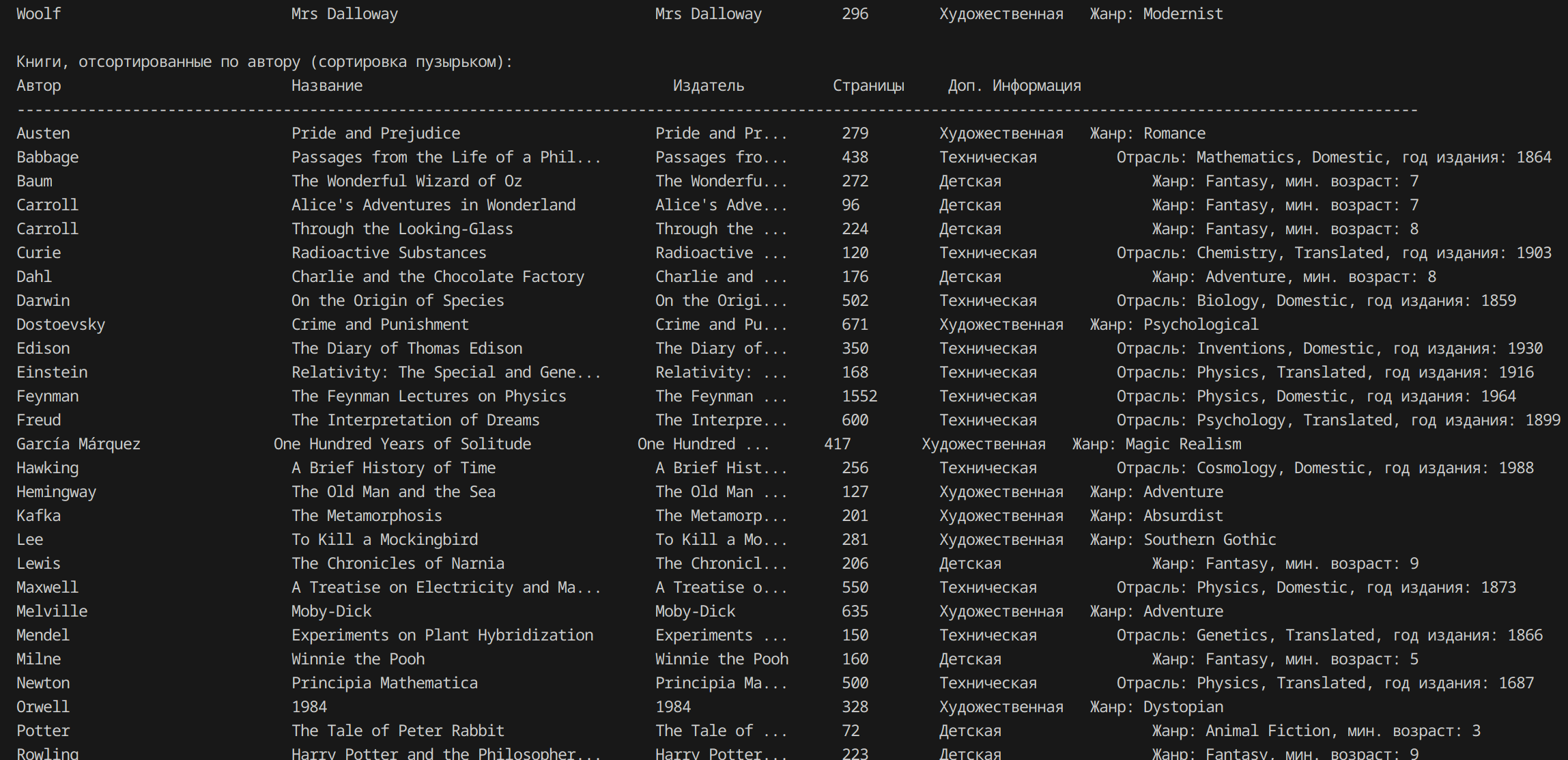
5. Просмотр книг, отсортированных по автору (сортировка массива книг).

Ваш выбор: 5

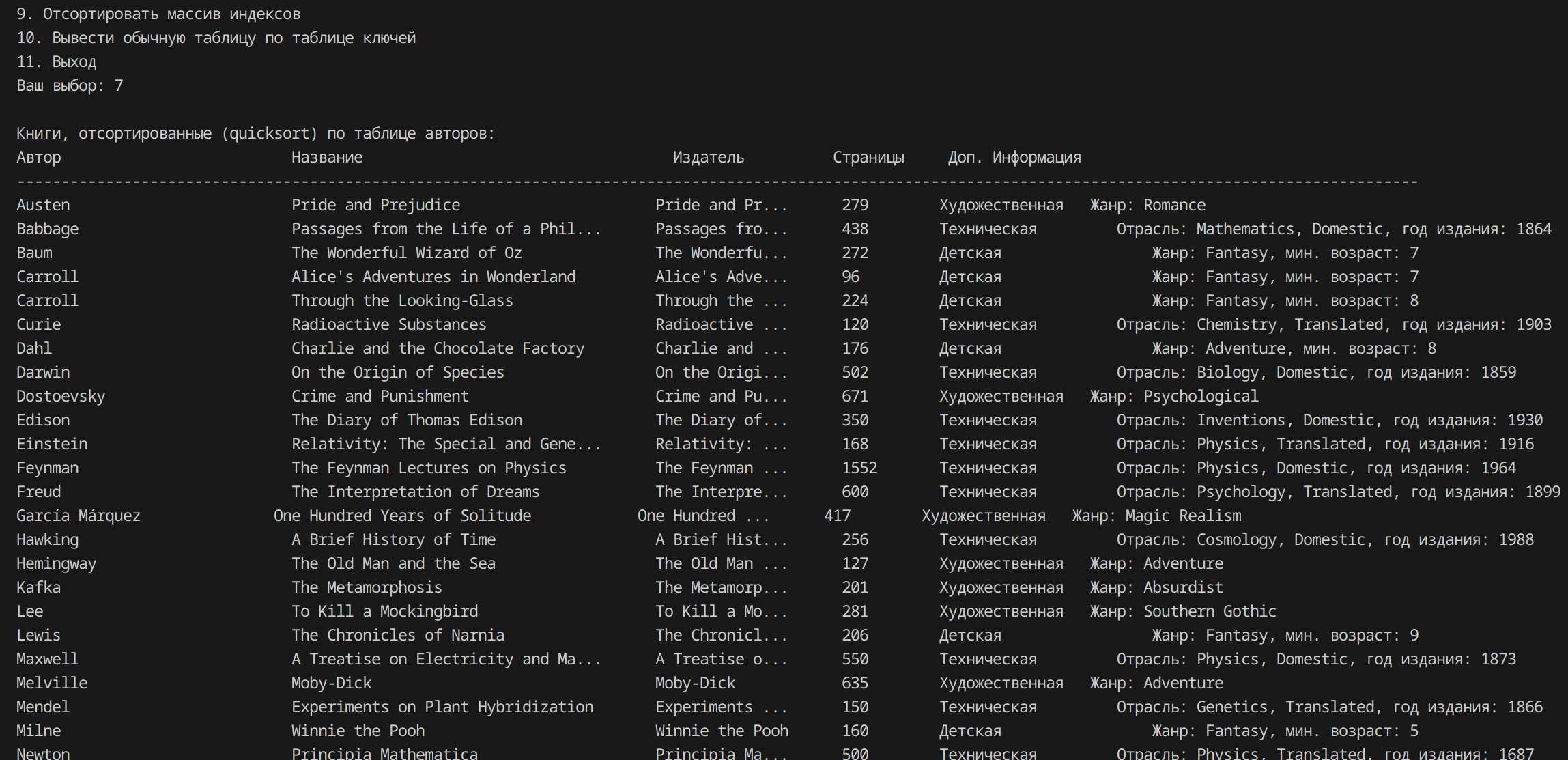
Книги, отсортированные по автору (быстрая сортировка):

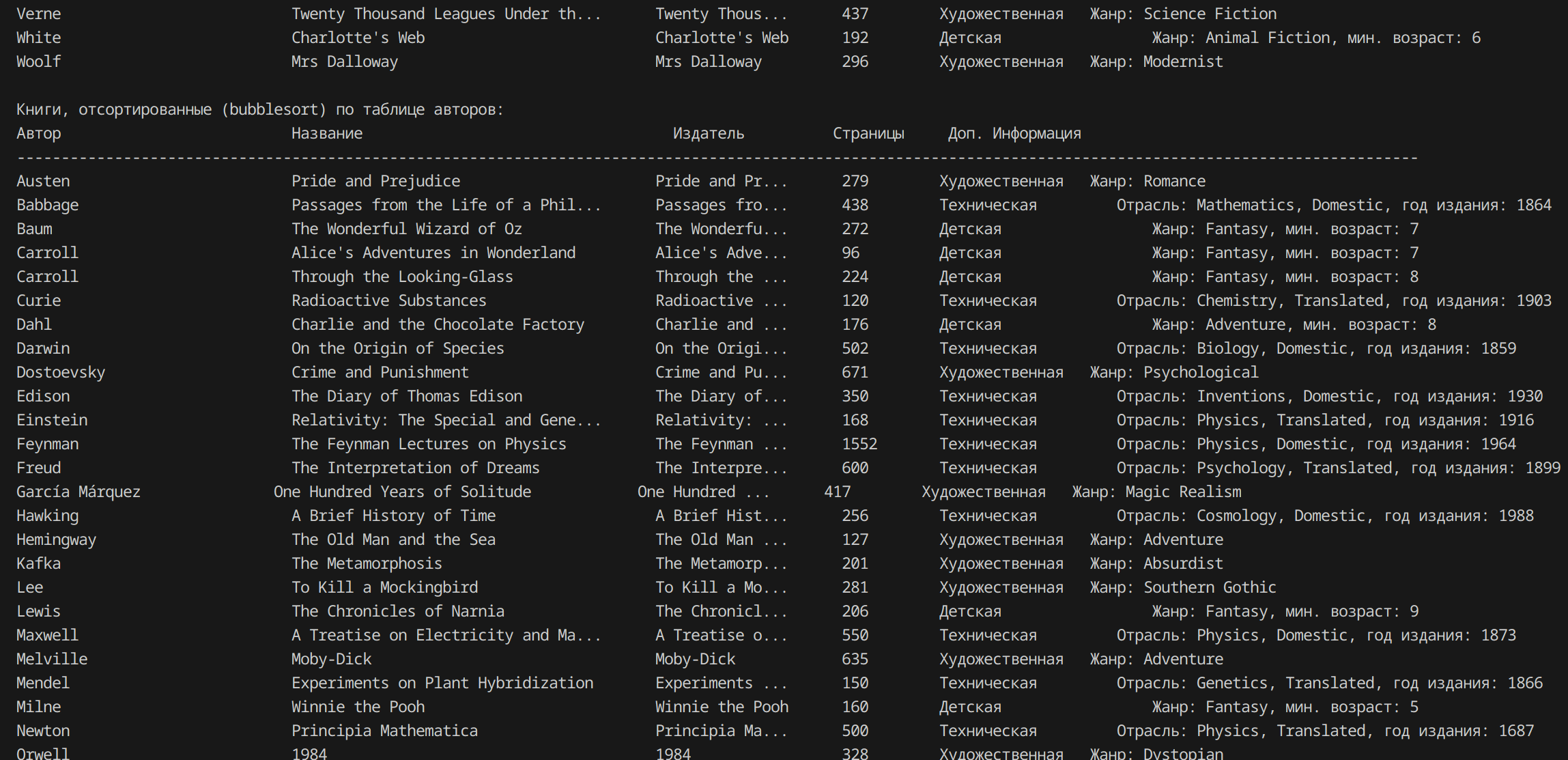
Автор Название Издатель Страницы



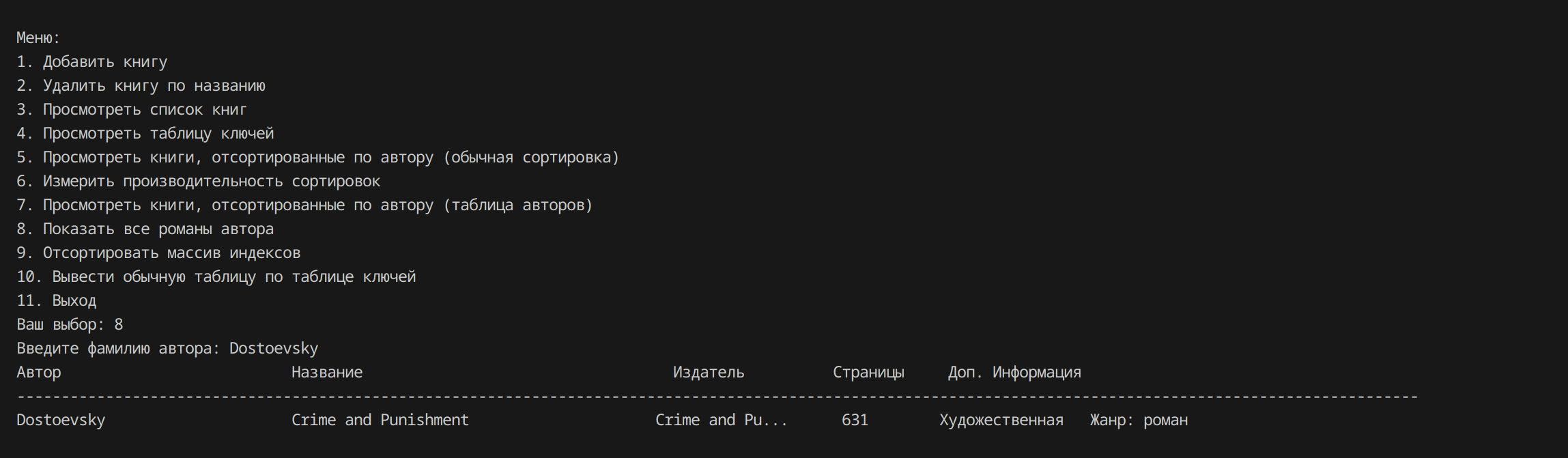


6. Просмотр книг, отсортированных по автору (сортировка таблицы ключей)





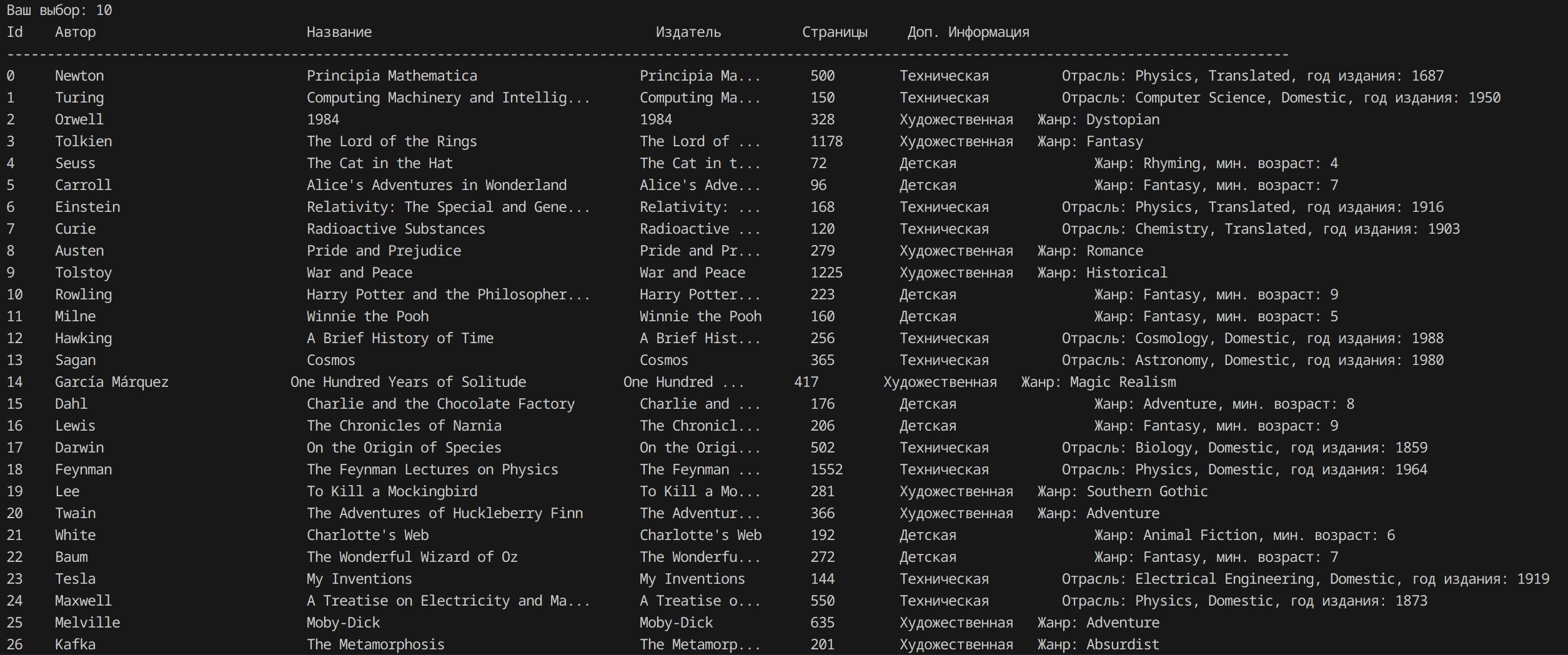
7. Поиск романов автора (Автор существует, роман(-ы) у автора тоже существует(-ют))



8. Сортировка массива индексов



9. Вывод обычной таблицы по таблице ключей



**Негативные тесты:**

1. Нечисловой выбор пункта меню

Ваш выбор: девять

Ошибка: Введите число.

2. Число в неверном диапазоне при выборе пункта меню

Ваш выбор: 0

Неверный выбор.

3. Пустой ввод при выборе пункта меню

Ваш выбор:

Ошибка: Введите число.

4. Добавление книги, когда массив переполнен.

Ваш выбор: 1

База данных переполнена.

5. Добавление книги, пустой ввод в любом из полей.

Ваш выбор: 1

Введите фамилию автора: Dostoevsky

Введите название книги: The Idiot

Введите издательство:

Ошибка: Издательство не может быть пустым.

6. Удаление несуществующей книги

Ваш выбор: 2

Введите название книги для удаления: Doesn't exist

Книга с названием "Doesn't exist" не найдена.

7. Поиск романов автора, но у него нет романов

Ваш выбор: 8

Введите фамилию автора: Mendel

Автор Название Издатель Страницы

------------------------------------------------------------Романы автора "Mendel" не найдены.

8. Поиск романов автора, такого автора нет

Ваш выбор: 8

Введите фамилию автора: Unknown

Автор Название Издатель Страницы

------------------------------------------------------------

Романы автора "Unknown" не найдены.

**Замерный эксперимент**

Проводились замеры для сортировки массива книг от 1000 до 10000 элементов в динамически выделяемом массиве с шагом в тысячу элементов с разными реализациями хранения элементов: массив книг и т.н. «таблица ключей», а также с различными сортировками: сортировкой пузырьком и быстрой сортировкой.

Сортировка проводилась по десять раз, после чего бралось среднее арифметическое из полученных измерений.

Результаты замеров следующие:

По времени:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Размер, шт** | **Сортировка пузырьком, с** | | **Быстрая сортировка, с** | | **Средняя скорость таблицы ключей от массива книг, %** |
| Массив книг | Таблица ключей | Массив книг | Таблица ключей |
| **1000** | 0,0155 | 0,0068 | 0,0007 | 0,0002 | 288,97 |
| **2000** | 0,0692 | 0,0275 | 0,0013 | 0,0005 | 255,82 |
| **3000** | 0,1466 | 0,0628 | 0,0022 | 0,0009 | 246,72 |
| **4000** | 0,2761 | 0,1257 | 0,003 | 0,0011 | 246,19 |
| **5000** | 0,4308 | 0,1888 | 0,0036 | 0,0015 | 234,09 |
| **6000** | 0,6932 | 0,2667 | 0,0049 | 0,0018 | 266,07 |
| **7000** | 1,0207 | 0,3672 | 0,0051 | 0,0019 | 273,19 |
| **8000** | 1,3113 | 0,4921 | 0,0062 | 0,0025 | 257,24 |
| **9000** | 1,7068 | 0,6142 | 0,0110 | 0,0025 | 358,94 |
| **10000** | 2,1114 | 0,7365 | 0,0078 | 0,0031 | 269,15 |

Средний прирост скорости от использования таблицы ключей составил **269,64%**

По памяти:

|  |  |  |
| --- | --- | --- |
| **Размер, шт** | **Массив книг, байт** | **Таблица ключей, байт** |
| **1000** | 312000 | 56000 |
| **2000** | 624000 | 112000 |
| **3000** | 936000 | 168000 |
| **4000** | 1248000 | 224000 |
| **5000** | 1560000 | 280000 |
| **6000** | 1872000 | 336000 |
| **7000** | 2184000 | 392000 |
| **8000** | 2496000 | 448000 |
| **9000** | 2808000 | 504000 |
| **10000** | 3120000 | 560000 |

Из таблицы выше видно, что использование таблицы ключей требует дополнительно **около 18%** памяти относительно исходного массива.

**Вывод**

В ходе проделанной лабораторной работы экспериментальным способом было установлено, что использование таблицы ключей для сортировки массива структур произвольных размеров (особенно больших размеров) позволяет значительно (более чем в 2,5 раза) ускорить сортировку. При этом дополнительное использование памяти таблицей ключей относительно исходного массива не слишком велика (< 20%). Соответственно, таблицу ключей лучше использовать, когда критична скорость сортировки, а использовать стандартную таблицу, когда важно эффективное использование памяти.

**Ответы на вопросы**

**Как выделяется память под вариантную часть записи?**

Память для вариантной части записи выделяется статически, размер будет равен наибольшему возможного размеру описанных в ней полей. Даже если используется только одно поле, в оперативной памяти резервируется место под всю возможную структуру, чтобы учесть наибольший размер данных.

**Что будет, если в вариантную часть ввести данные, несоответствующие описанным?**

Если ввести данные, которые не соответствуют описанным, то зачастую поведение программы непредсказуемо. В программе могут появиться ошибки, некорректные данные, сбои. Программа может аварийно завершиться.

**Кто должен следить за правильностью выполнения операций с вариантной частью записи?**

Ответственность за правильность работы с вариантной частью лежит на разработчике, ведь ему нужно убедиться, что данные корректно вводятся и используются в соответствии с указанными типами данных.

**Что представляет собой таблица ключей, зачем она нужна?**

Таблица ключей — это структура данных, в которой хранятся пары ключ - значение, позволяющие быстро находить записи в основной таблице. Ключи могут быть значениями определенных полей записи. Массив (таблица) ключей ускоряет поиск и сортировку данных, а также обеспечивает целостность и сохранность данных при работе с большими объемами.

**В каких случаях эффективнее обрабатывать данные в самой таблице, а когда использовать таблицу ключей?**

Обрабатывать данные эффективнее в самой таблице, когда она достаточно небольшая, а доступ к ней осуществляется зачастую напрямую. Таблица ключей эффективнее, когда таблица данных достаточно большая и в которой поиск занимает много времени, требуется быстрый доступ к данным по идентификаторам, и структура хранимых данных сложная, требующая частых добавлений, удалений, изменений, сортировки и выборки.

**Какие способы сортировки предпочтительнее для обработки таблиц и почему?**

Сортировка пузырьком часто работает лучше для частично или полностью отсортированных таблиц. Также сортировка пузырьком является устойчивой сортировкой, в то время как быстрая сортировка хоть и сортирует в разы быстрее за счёт особенностей реализации, но не обладает устойчивостью.