Изображение выглядит как символ, эмблема, герб, нашивка

Контент, сгенерированный ИИ, может содержать ошибки.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Отчет по выполнению практического задания №5.2

**Тема:** РАБОТА С ДАННЫМИ ИЗ ФАЙЛА

Дисциплина: Структуры и алгоритмы обработки данных

Выполнил студент Жижикин Л.С.

группа ИКБО-50-24

**Москва 2025**

Содержание

[Задание 1 2](#__RefHeading___Toc208_977610627)

[Постановка задачи. 2](#__RefHeading___Toc210_977610627)

[Структура файла 2](#__RefHeading___Toc212_977610627)

[Размер файла 2](#__RefHeading___Toc214_977610627)

[Составление бинарного файла 3](#__RefHeading___Toc216_977610627)

[Результат отработки системы 5](#__RefHeading___Toc218_977610627)

[Задание 2 6](#__RefHeading___Toc220_977610627)

[Постановка задачи 6](#__RefHeading___Toc225_977610627)

[Алгоритм 6](#__RefHeading___Toc223_977610627)

[Код линейного поиска 7](#__RefHeading___Toc227_977610627)

[Таблица с замерами 8](#__RefHeading___Toc229_977610627)

[Задание 3 8](#__RefHeading___Toc231_977610627)

[Постановка задачи 8](#__RefHeading___Toc233_977610627)

[Таблица данных 8](#__RefHeading___Toc485_2664379012)

[Код вспомогательных функций 9](#__RefHeading___Toc493_2664379012)

[Код однородного бинарного поиска 10](#__RefHeading___Toc495_2664379012)

[Таблица с замерами 11](#__RefHeading___Toc497_2664379012)

**Цель работы**: Получить практический опыт по применению алгоритмов поиска в таблицах данных

**Задание:** Разработать программу поиска записей с заданным ключом в двоичном файле с применением различных алгоритмов.

# Задание 1

## Постановка задачи.

Создать двоичный файл из записей (структура записи определена вариантом). Поле ключа записи в задании варианта подчеркнуто. Заполнить файл данными, используя для поля ключа датчик случайных чисел. Ключи записей в файле уникальны

## Структура файла

Файл состоит из записей следующего вида:

Число – unsigned int\_64

Автор – строка вида “X.X.Xxxxxx…” от 10 до 15 символов

Escape character – символ ‘\0’

Название книги – строка вида “Xxxxxx…” от 10 до 25 символов

Escape character – символ ‘\0’

## Размер файла

8 - число u\_i64;

20 - 40 байт на автора + название;

2 байта - символ выхода из последовательности;

В итоге, 30-50 байт на 1 книгу.

## Составление бинарного файла

Структура для описания книги:

A computer screen with many colorful text

AI-generated content may be incorrect.

Функция для случайной генерации одной книги:

A computer screen with many colorful text

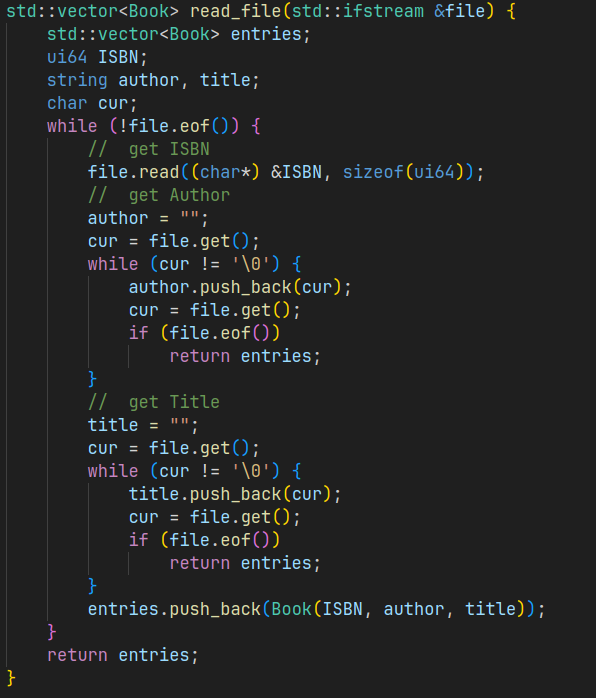
AI-generated content may be incorrect.

Функция для создания бинарного файла с данными:

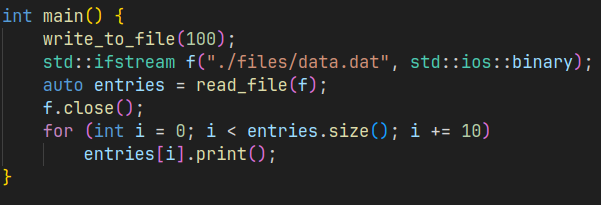
A screen shot of a computer code

AI-generated content may be incorrect.

Функция считывания информации о всех книгах в файле:

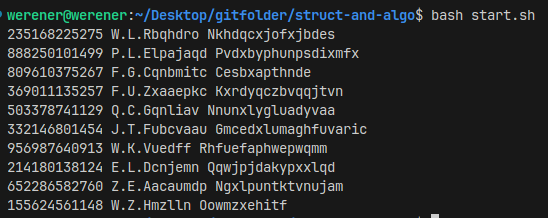


Функция main:



## Результат отработки системы

Вывод в консоль:

Содержимое файла:

# Задание 2

## Постановка задачи

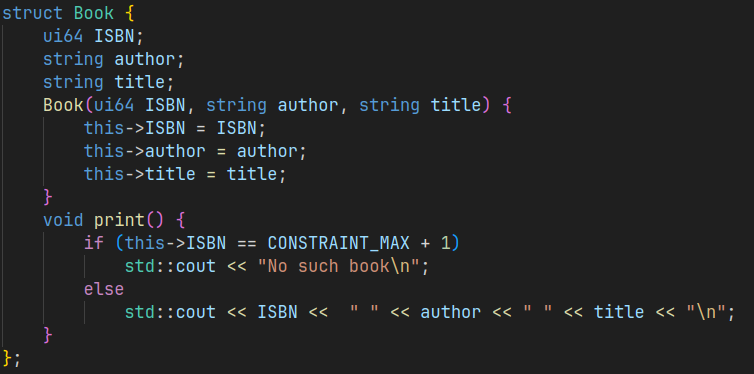
1. Разработать программу поиска записи по ключу в бинарном файле с применением алгоритма линейного поиска.
2. Провести практическую оценку времени выполнения поиска на файле объемом 100, 1000, 10 000 записей.
3. Составить таблицу с указанием результатов замера времени

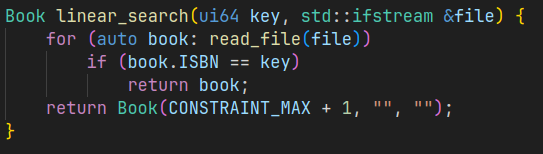
## Алгоритм

|  |
| --- |
| ***алг*** *Линейный поиск (****арг*** *цел key,* ***рез*** *лит s)*  ***нач***  ***пока*** *файл не закончился*  ***нач***  ***цел*** *reg\_num*  ***ввод*** *reg\_num*  ***лит*** *name*  ***ввод*** *name*  ***если*** *reg\_num = key*  ***нач***  *s:= name*  ***вернуть*** *s*  ***кон***  ***кон***  *s:= ""*  ***вернуть*** *s*  ***кон*** |

## Код линейного поиска

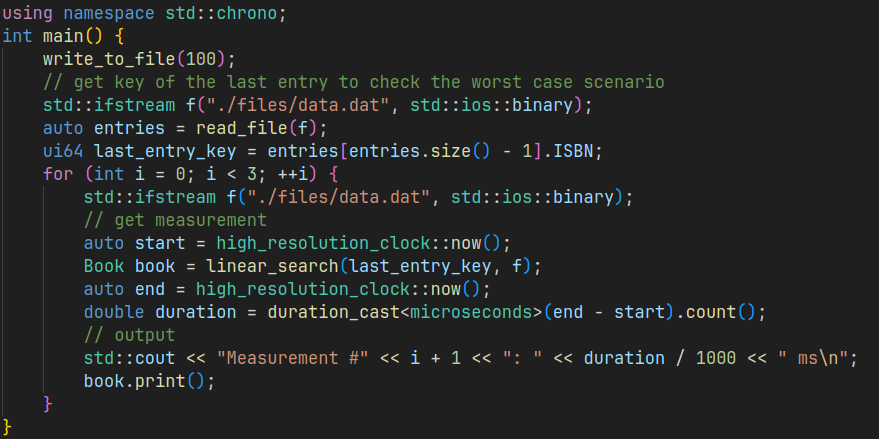
Для начала, изменим структуру Book, чтобы ее метод print поддерживал «несуществующую» книгу:

Функция линейного поиска:

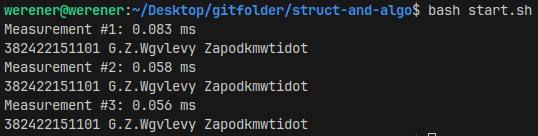


## Замеры

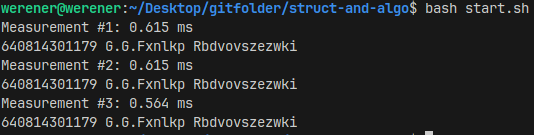
Функция main:



Вывод в консоль для 100 книг:



Вывод в консоль для 1000 книг:

Вывод в консоль для 10000 книг:

## Таблица с замерами

Для худшего случая:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Кол-во книг | Замеры (ms) | | | В среднем(ms) |
| 100 | 0.083 | 0.058 | 0.056 | 0.066 |
| 1000 | 0.615 | 0.615 | 0.564 | 0.598 |
| 10000 | 6.963 | 6.669 | 6.889 | 6.840 |

# Задание 3

## Постановка задачи

Поиск записи в файле с применением дополнительной структуры данных, сформированной в оперативной памяти.

## Таблица данных

Таблица данных реализована как вектор «ячеек». Ячейка, в свою очередь, является структурой Cell с двумя полями — номер ISBN, выполняющий роль ключа, и поле offset, хранящее номер байта, с которого начинаются автор и название книги по этому ключу.

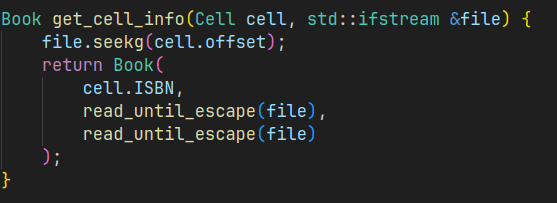
Доступ к записи будет происходить через std::seekg(offset), который перенесет указатель на offset-ый байт в файле.

В структуре Cell также хранится статический метод compare (для более простой сортировки функцией sort() из библиотеки algorithm) и метод вывода информации о текущей ячейке.

## Код вспомогательных функций

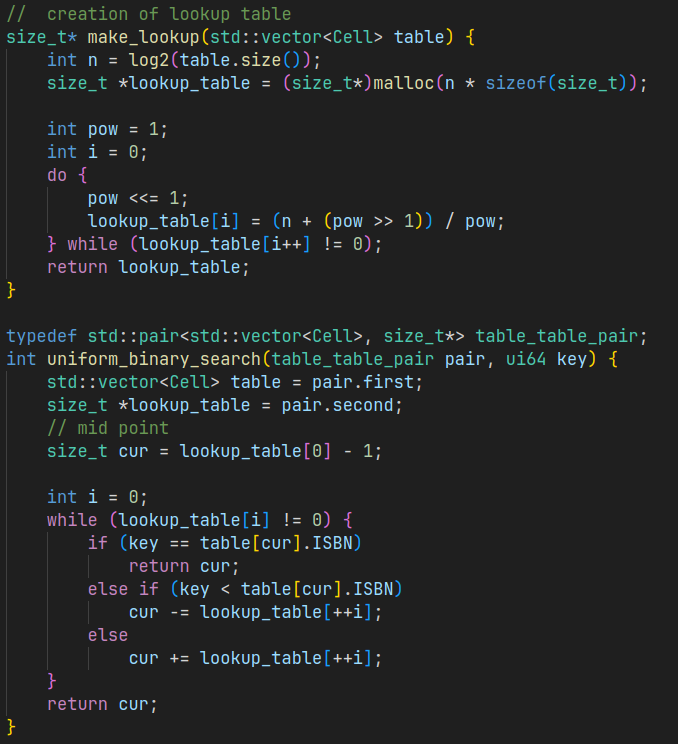
Функция для считывания данных в таблицу:

Функция получения книги по ячейке в таблице:



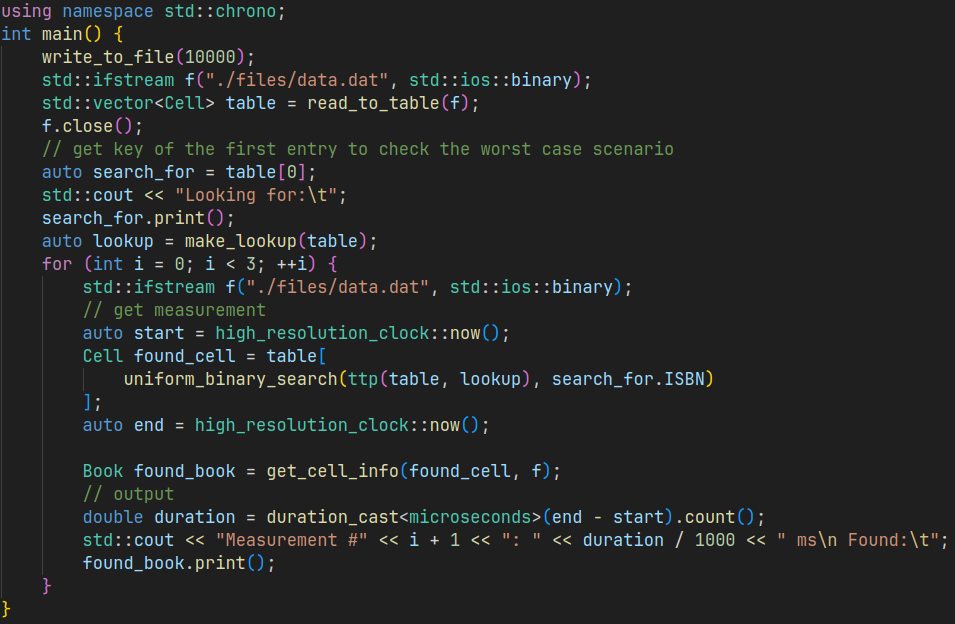
## Код однородного бинарного поиска

В функцию подается пара «таблица данных — таблица смещений» (table\_table\_pair — ttp) и искомый ключ:

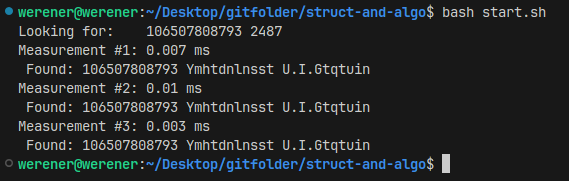


## Замеры

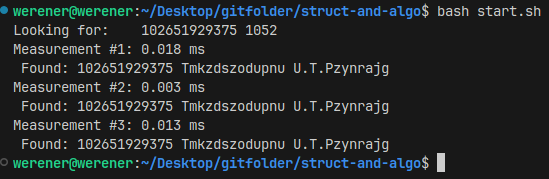
Функция main:



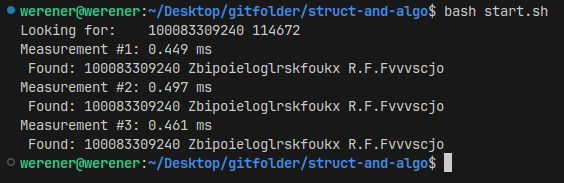
Вывод для 100 книг:



Вывод для 1000 книг:



Вывод для 10000 книг:



## Таблица с замерами

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Кол-во книг | Замеры (ms) | | | В среднем(ms) |
| 100 | 0.007 | 0.01 | 0.003 | 0.004 |
| 1000 | 0.018 | 0.003 | 0.013 | 0.011 |
| 10000 | 0.449 | 0.497 | 0.461 | 0.469 |