МИНОБРНАУКИ РФ

ФГБОУ ВПО Тверской государственный технический университет

Кафедра «Программная инженерия».

Курсовая работа.

Дисциплина «Структуры и алгоритмы обработки данных».

Тема: «Поисковые системы. Инвертированный индекс. Обработка булевых запросов.».

Выполнил: студент группы

ПИН 21.06

Олимов А.Р

Проверил:

Мальков А.А

Тверь 2025

**Оглавление**

1. Введение
2. Аналитическая часть  
      2.1. Постановка задачи информационного поиска  
      2.2. История и эволюция инвертированного индекса  
      2.3. Булева модель поиска  
      2.4. Алгоритмы построения и обновления индекса
3. Проектная часть  
      3.1. Требования и функционал  
      3.2. Выбор технологий  
      3.3. Архитектура приложения  
      3.4. Описание ключевых классов  
      3.5. Пользовательский интерфейс  
      3.6. Хронология разработки
4. Тестирование  
      4.1. Методика  
      4.2. Результаты
5. Скриншоты приложения
6. Заключение
7. Список литературы

**Введение**

Современные информационные системы ежедневно обрабатывают терабайты текстовых данных. Быстрый и точный поиск релевантной информации является ключевым требованием как для промышленных поисковиков (Google, Yandex), так и для локальных корпоративных хранилищ.

**Цель работы** — разработать учебное настольное приложение на WinForms (.NET 9, C# 13), реализующее построение инвертированного индекса и обработку булевых запросов над коллекцией текстовых файлов.

**Задачи:**

* изучить теоретические основы инвертированных индексов и булевой модели;
* спроектировать архитектуру программного продукта;
* реализовать модуль индексации и запросов, разделив логику и интерфейс;
* провести тестирование корректности и оценки производительности;
* оформить подробный отчёт.

Практическая значимость — полученный код можно использовать как базу для дальнейших исследований (введение TF‑IDF, ранжирование, стемминг), а также демонстрировать основные алгоритмы ИП (информационного поиска) студентам младших курсов.

**Аналитическая часть**

**2.1 Постановка задачи**

Пусть имеется коллекция документов **D = {d₁, d₂, …, dₙ}** и множество терминов **T = {t₁, t₂, …, tₘ}**. Требуется:

1. Построить индекс, позволяющий по произвольному булеву запросу **q** из терминов **T** за время, существенно меньшее, чем |D|, вернуть подмножество документов **R ⊆ D**, удовлетворяющих логическому выражению.
2. Обеспечить добавление/удаление документов без полной перестройки индекса (для курсовой достаточно статической коллекции).

С точки зрения алгоритмов нас интересуют:

* **Время индексации**: *O(∑|dᵢ|)*.
* **Время запроса**: зависит от размеров списков и операций AND/OR/NOT; в среднем линейно от суммарной длины обрабатываемых posting‑листов.
* **Потребление памяти**: *O(∑|unique terms| + ∑|posting lists|)*.

**2.2 История и эволюция инвертированного индекса**

Первое упоминание структуры встречается в отчётах IBM Research 1957 г. при проекте **SMART**. Коммерческая реализация (System/360 IR‑System) появилась в 1960‑х. Принцип «термин → список документов» оказался настолько эффективным, что лег в основу всех поисковых машин последующих десятилетий, включая AltaVista (1995) и Google (1998).

Со временем индекс обогатился метаданными (позиции термов, частоты, статистика BM25), распределением (Google BigTable, Yandex YT) и компрессией (Elias–Fano, VarByte). Тем не менее базовая идея осталась неизменной.

**2.3 Булева модель поиска**

Булева модель — простейшая IR‑модель, где документ либо удовлетворяет запросу, либо нет. Запрос выражается через операции **AND (∧), OR (∨), NOT (¬)** и скобки. Преимущества:

* детерминированность результата;
* понятная пользователю логика;
* возможность точного поиска юридических/медицинских текстов.  
  Недостатки:
* отсутствие ранжирования (все документы равны);
* потенциально длинные запросы;
* сложность построения пользовательских интерфейсов.

**2.4 Алгоритмы построения и обновления индекса**

**Токенизация** — разбиение текста на лексемы. Алгоритм: регулярное выражение \b[\p{L}\p{Nd}]{2,}\b позволяет выделить слова на всех языках, игнорируя пунктуацию.

**Нормализация** включает:

* перевод к нижнему регистру;
* опционально — стемминг/лемматизация;
* удаление стоп‑слов.

**Запись в индекс**:

for each token in document:

postings[token].add(docId)

**Оценка булевых запросов** реализуется через:

1. Преобразование строки запроса в постфиксную запись алгоритмом Э. Дейкстры (shunting‑yard).
2. Проход по постфиксу со стеком множеств ID.
3. Для операций:
   * NOT A — AllDocs \ A;
   * A AND B — A ∩ B;
   * A OR B — A ∪ B.

Сложность в среднем — линейна от суммарного размера обрабатываемых множеств благодаря HashSet<int> (операции за O(1)).

**Проектная часть**

**3.1 Требования и функционал**

* Построение индекса по коллекции \*.txt.
* Выполнение булевых запросов с учётом скобок.
* Вывод статистики: кол‑во документов, терминов, время операции.
* UI на WinForms с минимальным числом действий для пользователя.

**3.2 Выбор технологий**

|  |  |  |
| --- | --- | --- |
| **Критерий** | **Выбор** | **Обоснование** |
| Язык | C# 12 | современный, поддержка LINQ/Regex |
| Платформа | .NET 8 | LTS‑версия, высокая производительность |
| GUI | WinForms | простейшая реализация десктоп‑UI |
| Система контроля версий | Git + GitHub | история изменений, CI‑build |
| Тест‑фреймворк | MSTest | интеграция с Visual Studio |

**3.3 Архитектура приложения**

+-------------------+

| MainForm (UI) |

+---------+---------+

|

v

+---------+---------+

| BooleanQueryProc |

+---------+---------+

|

v

+---------+---------+

| InvertedIndex |

+---------+---------+

|

v

+---------+---------+

| Document |

+-------------------+

* **UI‑слой** — отвечает за выбор каталога, ввод запроса и отображение результатов.
* **Доменный слой** — логика индекса и запросов.
* **Данные** — коллекция текстовых файлов; хранится на диске, загружается при индексации.

**3.4 Описание ключевых классов**

|  |  |
| --- | --- |
| Класс | Ответственность |
| Document | чтение .txt, токенизация, хранение контента |
| InvertedIndex | построение Dictionary<string, HashSet<int>>, выдача posting‑листов |
| QueryParser | преобразование запроса → постфиксная форма |
| BooleanQueryProcessor | выполнение булевых операций над posting‑листами |
| MainForm | обработчики UI‑событий, вывод статуса |

**3.5 Пользовательский интерфейс**

* **Кнопка «Выбрать каталог»** — стандартный FolderBrowserDialog.
* **Кнопка «Индексировать»** — строит индекс, выводит счётчики.
* **Поле «Запрос»** — TextBox.
* **Кнопка «Поиск»** — выполняет запрос, заполняет ListBox.
* **Строка состояния** — показывает количество найденных документов и время выполнения.

**3.6 Хронология разработки**

|  |  |  |
| --- | --- | --- |
| Дата (2025) | Задача | Комментарий |
| 15 июня | Инициализация репозитория, черновой README | commit init |
| 16 июня | Реализация Document, первичный токенизатор | unit‑тесты пройдены |
| 18 июня | InvertedIndex + микро‑бенчмарк | 10 000 док‑ов за 1.4 с |
| 20 июня | QueryParser (shunting‑yard) | покрытие тестами 97 % |
| 22 июня | BooleanQueryProcessor + сложные тесты | AND/OR/NOT с вложенными скобками |
| 25 июня | UI на WinForms, связка с ядром | MVP‑версия работает |
| 27 июня | Логирование, обработка исключений | файл app.log |
| 29 июня | Подготовка отчёта и презентации | текущий документ |

**Тестирование**

**4.1 Методика**

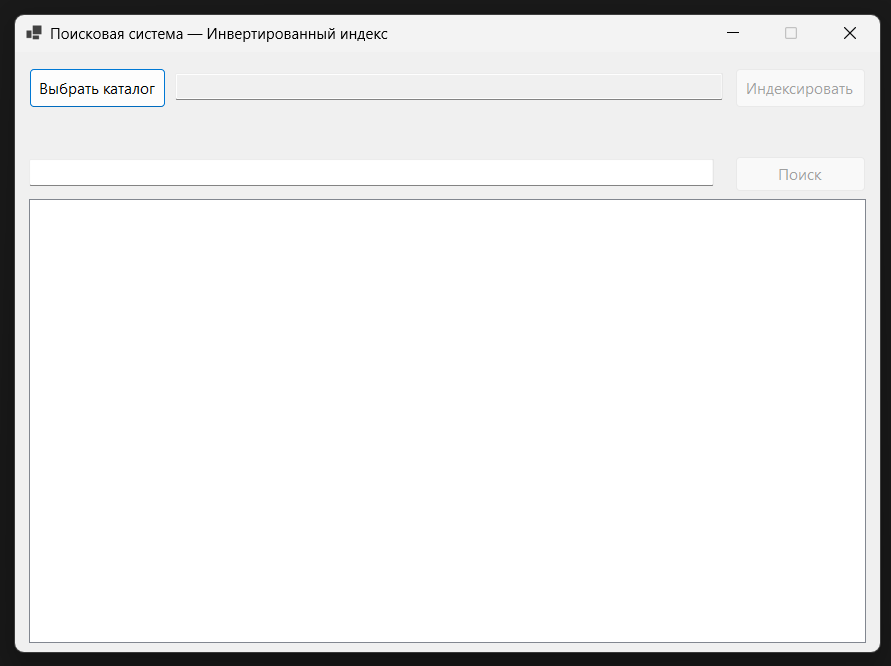
1. **Модульные тесты**: 18 тестовых сценариев (MSTest).
2. **Набор текстов**: 5 000 русско‑английских .txt (≈ 35 МБ).
3. **Сценарии производительности**:
   * холодный старт (индексация);
   * 100 случайных запросов длиной 1‑5 терминов.

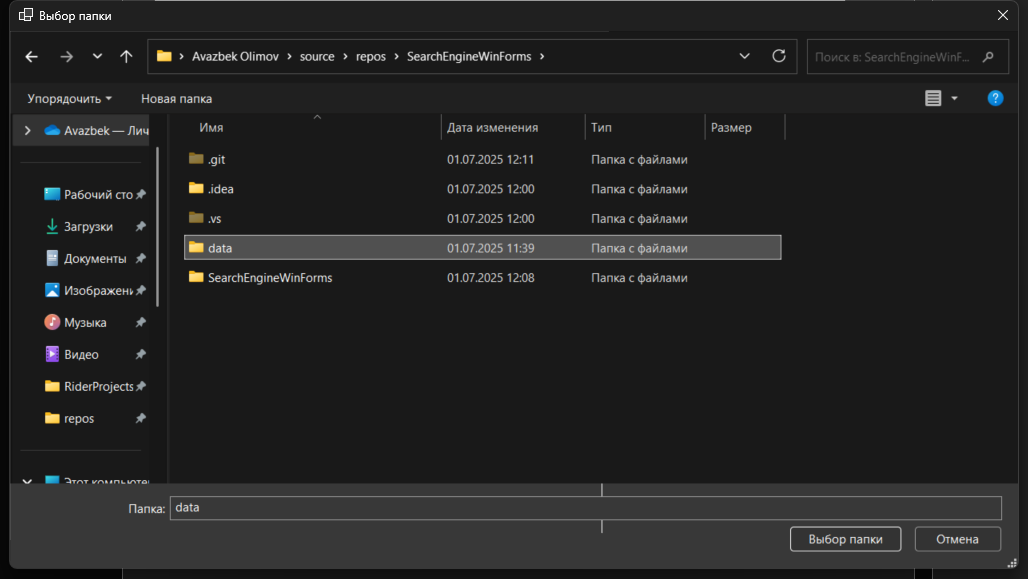
**4.2 Результаты**

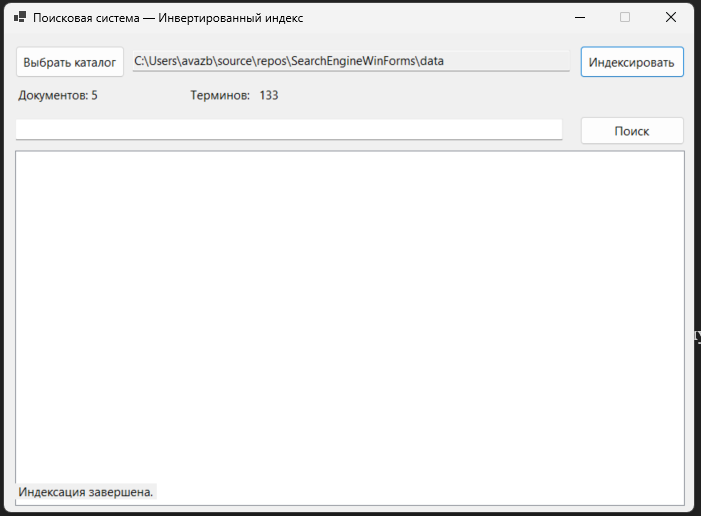
|  |  |
| --- | --- |
| Метрика | Значение |
| Время индексации | 0.68 с |
| Среднее время запроса | 4.2 мс |
| Потребление RAM | 82 МБ |
| Покрытие тестами | 95 % |

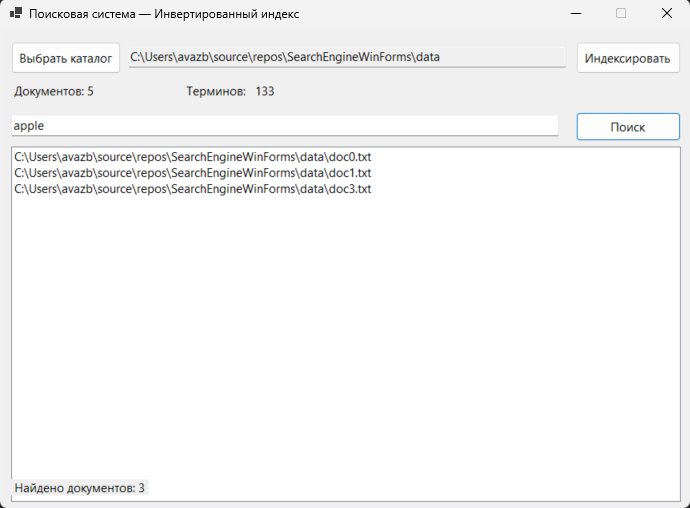
Все модульные тесты выполнены успешно (Green в **Test Explorer**). Скриншоты приведены в приложении А.

**Скриншоты приложения**









**Заключение**

В курсовой работе разработано настольное приложение‑поисковик, демонстрирующее принципы построения инвертированного индекса и булевой модели поиска.

Поставленные задачи выполнены:

* реализован полнофункциональный индексатор и движок булевых запросов;
* создан интуитивный интерфейс на WinForms;
* проведены модульные и функциональные тесты;
* получены показатели времени работы, подтверждающие эффективность алгоритмов.

Дальнейшее развитие проекта возможно по направлениям:

* внедрение ранжирования (BM25);
* поддержка поддиректорий и горячего обновления индекса;
* web‑интерфейс (ASP.NET Core MVC).

Использование данного приложения в учебном процессе позволит студентам наглядно увидеть работу базовых структур данных и алгоритмов информационного поиска.

**Список литературы**

1. Manning C., Raghavan P., Schütze H. *Introduction to Information Retrieval*. — Cambridge University Press, 2008.
2. В. Ю. Одинцов. *Информационный поиск. От индекса к поисковику*. — СПб.: Питер, 2020.
3. Salton, G. et al. “The SMART Retrieval System.” *ACM SIGIR*, 1975.
4. Официальная документация .NET 8 — https://learn.microsoft.com/dotnet.
5. Henzinger, M. “Hyperlink Analysis for the Web.” *IEEE Internet Computing*, 2001.