# Московский авиационный институт (национальный исследовательский университет)

# Институт информационные технологии и прикладной математики

Кафедра вычислительной математики и программирования

# Лабораторная работа №5 по курсу «Информационный поиск»

 Студент:
 Е.М. Стифеев

 Преподаватель:
 А.А. Кухтичев

 Группа:
 M8O-109M-21

Дата: 25.11.21

Оценка: Подпись:

# Лабораторная работа №4 «Поиск цитат, координатный индекс»

В этом задании необходимо расширить язык запросов булева поиска новым элементом – поиском цитат. Синтаксис этого элемента следующий:

- [ «что где когда» ] кавычки, включают режим цитатного поиска для терминов внутри кавычек. Этому запросу удовлетворяют документы, содержащие в себе все термины *что*, *где* и *когда*, причём они должны встретиться внутри документа ровно в этой последовательности, без какихлибо вкраплений других терминов.
- [ «что где когда» / 5 ] аналогично предыдущему пункту, но допускаются вкрапления других терминов так, чтобы расстояние от первого термина цитаты до последнего не превышало бы 5.

Новый элемент может комбинироваться с другими стандартными средствами булева поиска, например:

- [ «что где когда» && друзь ]
- [ «что где когда» || квн ]
- [ «что где когда» &&! «хрустальная сова» ]

Для реализации цитатного поиска нужно использовать координатный индекс, т.е. для каждого вхождения термина в документ построить и сохранить список позиций внутри документа, где этот термин встречался.

В отчёте нужно описать формат координатного индекса. Привести статистические данные:

Размер получившегося индекса.

- Время построения индекса.
- Общее количество позиций. Среднее количество позиций на термин и на пару термин-документ.
- Скорость индексации (кб входных данных в секунду)
- Время выполнения поисковых запросов.
- Примеры долго выполняющихся запросов.

Кроме того, нужно привести примеры запросов и результаты их выполнения. В выводах должны быть указаны недостатки работы, приведены примеры их решения. Что можно сделать, чтобы ускорить «долгие» запросы?

#### 1. Описание

# Корпус

Напомню, что корпус документов имеет следующую структуру, полученную по результатам  $\Pi P1$  (доступен по ссылке <a href="https://cloud.mail.ru/public/ZfkX/gccM7hnDR">https://cloud.mail.ru/public/ZfkX/gccM7hnDR</a>):

- Корпус документов
  - o films1.txt (94 Мб, 15000 документов)
  - o films2.txt (96 Мб, 15000 документов)
  - o films3.txt (184 Мб, 15000 документов)
  - o films4.txt (219 Мб, 15000 документов)
  - o films5.txt (322 Мб, 15000 документов)
  - o films6.txt (711 Мб, 15000 документов)
  - o films7.txt (823 Мб, 15000 документов)
  - o films8.txt (226 Мб, 15000 документов)
  - o films9.txt (67 Мб, 15000 документов)
  - o films10.txt (75 Мб, 15000 документов)
  - o films11.txt (99 Мб, 15000 документов)
  - o films12.txt (78 Мб, 15000 документов)
  - o films13.txt (41 Мб, 6109 документов)

$$\Sigma_{Gb} = 2,899 \text{ Gb}, \Sigma_{docs} = 186109$$

Также, напомню, что получение одного документа могло включать проход по нескольким html-страницам и обработку динамически подгружаемых страниц, поэтому общее количество обработанных страниц было >800'000.

В каждом файле \*.txt документы хранятся следующим образом:

- 1 строка 1 документ {....}
- 2 строка 2 документ {....}
- n строка n документ  $\{....\}$

Каждый документ снабжён прямой ссылкой на источник, откуда был скачен, и хранит только выделенный из html-кода текст в кодировке UTF-8. Например, 234 строка файла films1.txt выглядит так:

{"page\_url": "https://www.kinopoisk.ru/media/article/1773537/", "title": "Артур Смольянинов: «Я сомневался, что смогу сыграть ангела»", "body": "2 января в российский прокат вышла романтическая комедия Веры Сторожевой "Мой

# Индекс

Готовый индекс хранится в четырёх файлах:

# • **docs\_id.data** (42 M6)

Файл служит для отображения индекса документа (doc\_id) в его текстовое представление в файлах \*txt. Поддерживается переменная длина пути до файлов с документами.

# • terms.data (54 M6)

Файл служит для хранения словаря с терминами и ссылок (смещений) на файл с словопозициями и координатами. Поддерживается переменная длина термина. Термины упорядочены в лексикографическом порядке.

# • postings\_list.data (2.68 Γδ)

Файл служит для хранения словопозиций и координат терминов в документе. Слопозиции упорядочены по возрастанию идентификаторов документов.

# Структура

n_terms						
n_docs[0]	docdoc	freqfreq	offsetoffset	beginbegin	endend	begin
n_docs[1]	docdoc	freqfreq	offsetoffset	beginbegin	endend	begin
•••						

#### Описание полей

Название	Тип	Назначение
n_terms	uint	Число терминов в корпусе/ число
		списков словопозиций и координат
n_docs[0],,	uint	Число словопозиций для
n_docs [n_terms -1]		конкретного терма

doc[i][0],,	*int	Вектор идентификаторов
doc[i][n_docs[i]-1],		документов, в которых встречается
$i = 0n_{terms-1}$		термин (слопозиции)
freq[i][0],,	*int	Вектор частот вхождений терма в
freq [i][n_docs[i]-1],		документы
$i = 0n_{terms-1}$		
offset[i][0],,	*uint	Относительные смещения до
offset[i][n_docs[i]-1],		координат. Таким образом, если
$i = 0n_{terms-1}$		понадобятся координаты і-терма в ј-
		м документе, то сначала выполнится
		смещение до нужной строки в
		таблице posting_list.data с помощью
		смещений в словаре. Затем, зная
		число документов n_docs[i], можно
		быстро считать freq и offset, не читая
		остальные данные. Далее, с
		помощью offset выполняется
		смещение до блока, в котором
		находятся координаты beginbegin,
		endend терма в документе. Их
		количество равно значению freq.
begin begin	*int	Координаты начал термина в
		документе. Координаты измеряются
		в символах от начала документа
endend	*int	Координаты концов термина в
		документе. Координаты измеряются
		в символах от начала документа

# • **tf.data** (939 M6)

Файл служит для быстрого получения компонент документа, как вектора в пространстве терминов. Он нужен для быстрого ранжирования на основе косинуса между вектором запроса и вектором документа. См. подробности в ЛР по ранжированию.

# Алгоритм цитатного поиска

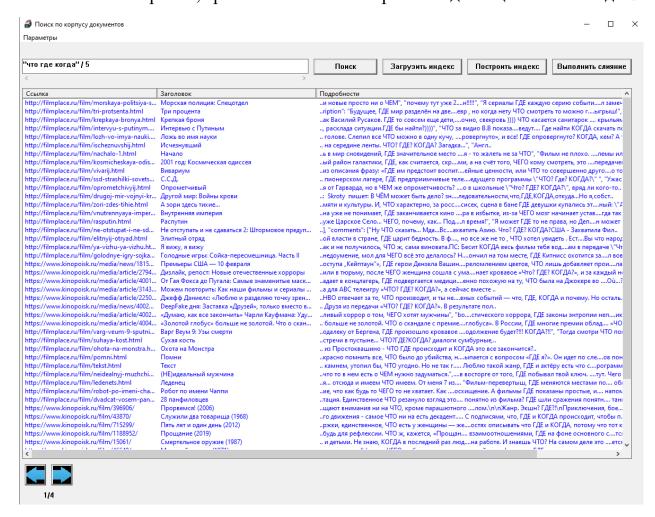
Рассмотрим алгоритм на примере поиска "что где когда" \ 5. Сначала переведём расстояние в словах в расстояние в символах. Для этого сделаем предположение, что среднее число пробельных символов составляет 2

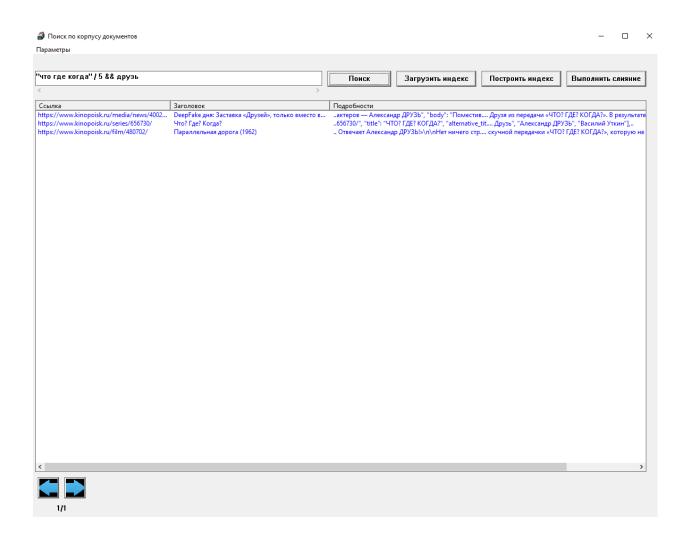
символа, а средняя длина слова – 7 символов. Тогда 5 требуемых слов переведётся в:

2 (пробел) + 3 (где) + 2 (пробел) + 5 (когда) + 2 + 7 + 2 + 7 + 2 + 7 = 32 символа. Расстояние между словами считается как расстояние между их концами в

символах.

Далее, получим словопозиции и координаты для терминов "что" и "где" из файлов индекса. Запустим обычный алгоритм пересечения списков словопозиций, но будем дополнительно отсекать документы, если они не удовлетворяют требованиям близости терминов. Для этого создадим цикл по всем координатам терма "что" и, пользуясь отсортированностью координат терма "где", с помощью бинарного поиска найдём ближайшие вхождения и проверим их на близость. Сохраним координат цитаты ("что где"). Далее, пересекаем получившийся список словопозиций со списком для терма "когда" аналогичным образом, сравнивая близость терма "когда" и цитаты "что где".

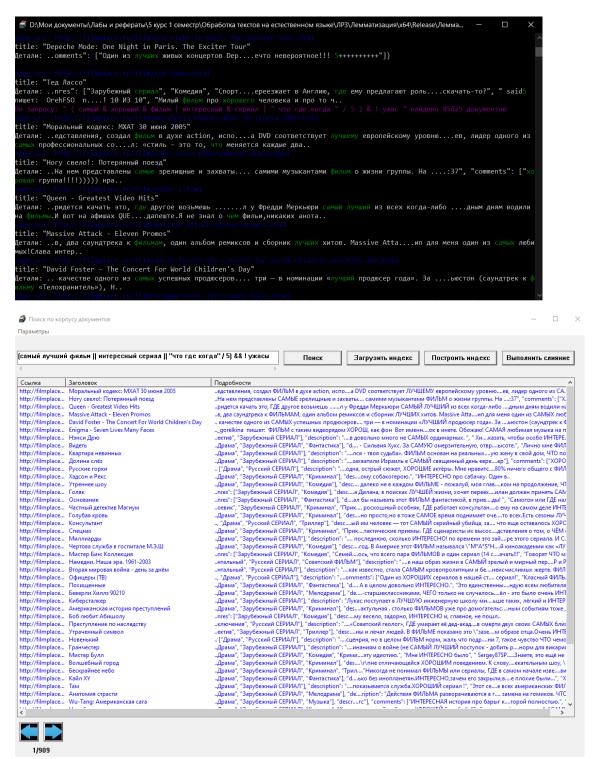




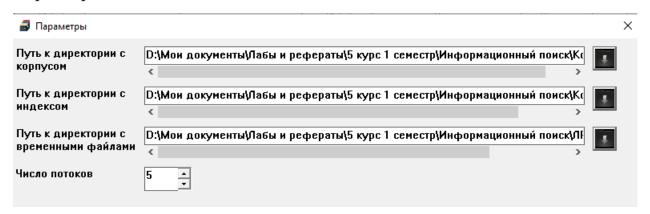
# Интерфейс

Программа была реализована в двух интерфейсах: консольном и оконном.

Для последнего использовалась библиотека Windows.h (winapi). Примеры ниже:



# Параметры поиска:



*Примечание*. Внимательный читатель мог заметить наличие лемматизации и сниппетов. Подробности реализации см. в соответсвующих ЛР по курсу.

# 2. Исходный код

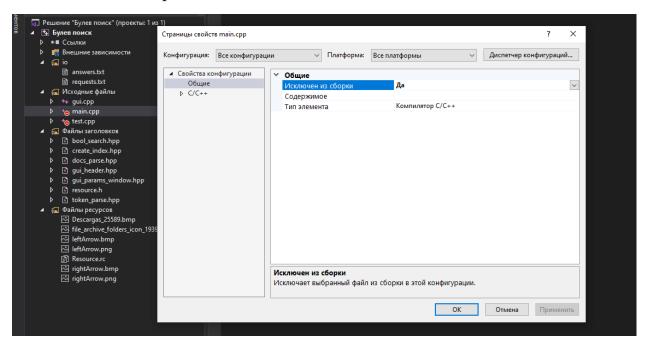
# Структура проекта

- include
  - o bool\_search.hpp (булев поиск)
  - о create\_index.hpp (создание, чтение индекса)
  - о defs.hpp (подключение внешних библиотек, макросы)
  - о docs\_parse.hpp (извлечение полей из корпуса)
  - о gui\_defs.hpp (подключение внешних библиотек, макросы, глобальные переменные)
  - о gui\_params\_window.hpp (окно с выбором параметров)
  - о quote\_search.hpp (реализация цитатного поиска)
  - о resource.h (подключение изображений, иконок и прочего)
  - о token\_parse.hpp (функции для преобразования токенов в термы)
- python
  - о lemmatizator.py (лемматизация документа)
  - о lemmatizator\_setup.py (компиляция lemmatizator.py в ехе-файл)
  - o request\_parse.py (лемматизация запроса)
  - o request\_parse\_setup.py (компиляция request\_parse.py в ехе-файл)
- io
- o anwers.txt
- o requests.txt
- src
  - о gui.cpp (точка входа в оконный интерфейс)
  - о main.cpp (точка входа в консольный интерфейс)
  - о test.cpp (утилита для тестирования программы)
- resourses (файлы ресурсов для оконного приложения)

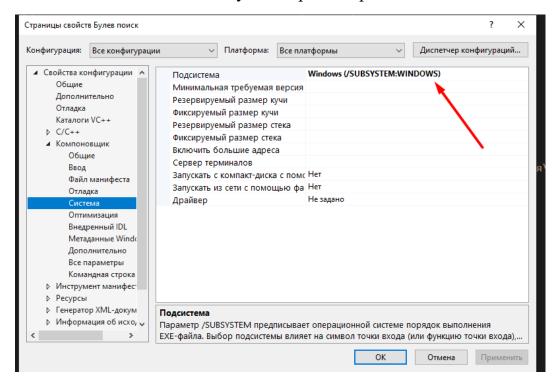
Проект был написан с помощью Microsoft Visual Studio 2019 эксклюзивно для ОС семейства Windows.

# Запуск

Переключение между тремя точками входа осуществляется с помощью флага «исключить из сборки»:



Не забудь при переключении между консольными и оконными приложениями менять подсистему в настройке проекта:



### Консольное приложение поддерживает флаги запуска:

- -і 'абс. путь к корпусу'
- -о 'абс. путь к индексу'
- -t 'абс. путь к директории с блочным индексом'
- -р кол-во процессов для распараллеливания
- -create : создать блочный индекс
- -merge : выполнить слияние блочного индекса
- -clear : очистить папку с временными файлами после слияния
- -search : выполнить поиск

# Утилита тестирования поддерживает следующие ключи:

- -і 'абс. путь к корпусу'
- -о \'абс. путь к индексу'
- -n1 число запросов
- -n2 длина запросы в термах

# Пример создания индекса из корпуса:

\$ ./Булев индекс.exe -merge -create -clear -i "D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус" -о "D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\_index" -t "D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\ЛР3\Булев индекс\tmp"

[INFO] Создание индекса для блоков

[INFO] Thread 0 processing block 1/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films13.txt

[INFO] Thread 1 processing block 2/13 : D:\Mou документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films9.txt

[INFO] Thread 2 processing block 3/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films10.txt

[INFO] Thread 3 processing block 4/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films12.txt

[INFO] Block 1 has 232216 terms

<...>

[INFO] Block 10 has 597482 terms

```
[INFO] Block 11 has 669497 terms
```

[INFO] Block 12 has 921883 terms

[INFO] Block 13 has 1383148 terms

[INFO] Создание очередей термов: 13 блок из 13

[INFO] Слияние docs id: 13 блок из 13

[INFO] Слияние слопозиций термов

[INFO] Осталось термов: 0

[INFO] Очистка временных файлов

[INFO] Общее число термов в словаре = 2809203

Время выполнения = 145,5 sec, размер корпуса = 2,899 Gb, документов = 186109

Средняя скорость на документ = 0,782 ms

Средняя скорость на килобайт = 0,048 ms

# Пример работы тестировщика:

\$ ./Булев индекс.exe -n1 200 -n2 5 -search -i "..\..\Корпус" -o "..\..\Корпус index" -t "tmp" <io/requests.txt >io/answers.txt

# 3. Выводы

Размер индекса	3.69 Гб	
Время построения индекса	4 часа (отечественный лемматизатор	
	natasha работает очень медленно!)	
Скорость индексации	268.7 Кб / сек	
Время выполнения поисковых	≈2 сек. Одну из них занимает вызов	
запросов	natash'и для лемматизации терминов	
	из запроса.	

Пример долго выполняющегося запроса (30 сек):

[фильм || сериал || мультфильм]

```
No запросу: "фильм | сериал | мультфильм " найдено 162370 документов раде_url: http://filmplace.ru/film/jimmy-радe-and-robert-plant-no-quarter-unledded.html title: "Jimmy Page & Robert Plant - No Quarter - Unledded"
Детали: ..Некоторые фрагменты фильма были сняты в национ..

ваде_url: http://filmplace.ru/film/slava-bobkov-i-gruppa-altaj-xxi-pereplavlennyij-v-kolokol.html title: "Cnama Бобков и группа Алтай XXI: Переплавленный в колокол"
Детали: ..в Санкт-Петербурге. Фильм писали поти 7 часо..

раде_url: http://filmplace.ru/film/primal-fear.html

title: "Primal Fear - 16.6 All Over The World"
Детали: ..нным художественным фильмом, который незадолго ..

раде_url: http://filmplace.ru/film/madonna-the-re-invention-world-tour.html

title: "Madonna - The Re-Invention World tour"
Детали: ..ачный опыт. Музыка, фильмы, танцы — все, что я..

раде_url: http://filmplace.ru/film/sergej-lyubavin-nash-esenin.html

title: "Сергей Любавин - Наш Есенин"
Детали: .."], "description": "Фильм был снят 17 февраля..

раде_url: http://filmplace.ru/film/the-rolling-stones-the-stones-in-the-park.html

title: "The Rolling Stones: The Stones In The Park"

Детали: ..просмотра трудно. В фильме есть нечто гипнотич..

раде_url: http://filmplace.ru/film/the-rolling-stones-live-at-the-max.html

title: "The Rolling Stones: Live At The Max"
Детали: ..зный проект — сняли фильм, посвященный гастро..

раде_url: http://filmplace.ru/film/moralnyij-kodeks-mhat-30-iyunya-2005.html

title: "Моральный кодекс: МХАТ 30 мюня 2005"
Детали: ..едставления, создал фильм в духе action, испо..

раде_url: http://filmplace.ru/film/kontsert-eltona-dzhona-v-sssr-v-1979-godu.html
```

Причины: длинный список словопозиций-документов, ранжирование по методу косинусов (20 сек), создание сниппетов (8 сек).

В ходе выполнения лабораторной работы я научился выполнять координатный поиск для коллекции документов. Познакомился с winapi. Научился разрабатывать оконный интерфейс на языке C/C++. Научился выполнять python-скрипты внутри C++.

# Литература

[1] Кристофер Д.Маннинг, Прабхакар Рагхаван, Хайнрих Шютце. Введение в информационный поиск. 2020, изд. Вильямс.