Федеральное агентство связи

Сибирский государственный университет телекоммуникаций и информатики

**Межрегиональный учебный центр переподготовки специалистов**

**Курсовая работа**

по дисциплине: Структуры и алгоритмы обработки данных (часть 2)

**Выполнил**: Коростелин А. В.

**Группа:** ПБТ-11

**Вариант:** 3/3.3/1

**Проверила**: Мачикина Е. П.

Новосибирск, 2022

**Задание**

1. Хранящуюся в файле базу данных "Обманутые вкладчики", состоящую из 4000 записей загрузить в оперативную память компьютера и построить АВЛ-дерево поиска для решения задачи по поиску записей (Вывести на экран количество и фамилии всех адвокатов из базы данных; Вывести упорядоченный (по полю ФИО вкладчика) список вкладчиков, которые имеют заданного адвоката, и их фамилии начинаются на заданную подстроку). Из найденных записей организовать список (очередь) и вывести их на экран.

2. Созданное приложение должно иметь дружественный интерфейс, выводить подсказки при вводе и выводе данных, проверять корректность вводимых данных и пр. Операции, выражающие логически завершенные действия, рекомендуется оформлять в виде функций, грамотно выбирая параметры функций. Имена переменных подпрограмм, параметры подпрограмм, используемые языковые конструкции должны способствовать удобочитаемости программы.

3. Протестировать созданное приложение, скриншоты приложения включить в отчет.

Структура записи БД:

1. ФИО вкладчика: текстовое поле 32 символа

формат <Фамилия>\_<Имя>\_<Отчество>

2. Сумма вклада: целое 16-pазpядное положительное число

3. Дата вклада: текстовое поле 8 символов, фоpмат дд-мм-гг

4. ФИО адвоката: текстовое поле 22 символа

фоpмат <Фамилия>\_<буква>\_<буква>

**Содержание**

[1. Формирование требований 4](#_Toc96366965)

[2. Проектирование 5](#_Toc96366966)

[3. Реализация 11](#_Toc96366967)

[4. Список литературы 15](#_Toc96366968)

# 1. Формирование требований

Требуется разработать программу DbView для доступа к файлу БД в режиме «только чтение», без внесения изменений, в соответствии с заданием.

Программа выполняется в режиме «Командной строки» на IBM PC-совместимом компьютере под управлением ОС Microsoft Windows версий, актуальных на момент составления настоящих требований, разрядность x32/x64.

Определим следующие требования к функционалу приложения:

1. Просмотр содержащихся в БД записей. Поскольку число записей значительно превышает размер экрана, предусматривается механизм прокрутки (скроллинг).
2. Поиск в БД всех записей об адвокатах и отображение их в виде списка.
3. Реализация поиска записей по вводимой с клавиатуры подстроке.
4. Интуитивно понятный текстовый интерфейс с элементами псевдографики. Управление осуществляется стрелками на клавиатуре и клавишами Fx. Для управления прокруткой используются стрелки вверх/вниз, кнопки PageUp/PageDown. Завершение работы программы по нажатию кнопки Escape.

Программа рассчитана на широкий круг пользователей, обладающих стандартными навыками работы с ПК. Каких-либо режимов разделения прав доступа, средств надзора, диагностических и административных инструментов не предусматривается.

Программа поставляется в скомпилированном виде (файлы EXE, DLL), готовом для исполнения в целевой рабочей среде. Инструменты установки и удаления не предусматриваются.

Для разработки программы используется язык Си и свободная кроссплатформенная среда разработки [Code::Blocks IDE](https://www.codeblocks.org/) v.20. Приложение имеет псевдографическое меню, реализованное с применением библиотеки управления терминалом [PDCurses](https://pdcurses.org/) (лицензия без ограничений – public domain).

К данной работе приложена скомпилированная версия приложения (исполняемый файл DbView.exe) и полный исходный код проекта (каталог src).

# 2. Проектирование

Определим внутреннюю логику приложения DbView.

Работа приложения начинается с открытия файла БД и загрузки содержимого в память. Поскольку каждая строка файла представляет собой отдельную запись БД, файл считывается построчно, и каждая строка разбивается на составляющие в соответствии определённому в задании формату. Для каждой записи выделяется пространство в памяти ПК. Выполняется функцией:

static int ReadDbFile(const char\* fname, struct List \*dbcontents);

Единожды считанные из файла и размещённые в памяти записи более никуда не перемещаются и никак не изменяются. Доступ к записям осуществляется посредством списка, хранящего указатели на размещённые в памяти ПК структуры. При необходимости (например, при поиске имён адвокатов) создаётся копия списка, над которой производятся операции сортировки и удаления нерелевантных для выполняемой задачи указателей.

После загрузки информации, файл БД закрывается, и производится построение АВЛ-дерева записей. Ключевым полем является ФИО вкладчика. Код построения дерева в общем аналогичен Лабораторной работе №2 части 2 курса, за исключением того, что вершина дерева хранит указатель на запись, а не целое число. Процесс построения сопровождается прогресс-баром для наглядности.

После построения дерева, на экран выводится список всех содержащихся в БД записей. Пользователь может вызвать желаемую функцию приложения нажатием клавиш Fx (F2 – показать все записи БД, F3 – показать перечень адвокатов, F4 – выполнить поиск вкладчика). Клавиши считываются в бесконечном цикле, прерываемом нажатием Escape, после чего программа завершает работу.

Для формирования перечня адвокатов, создаётся новый список указателей. Затем, для каждой записи БД выполняется проверка поля «ФИО адвоката» – содержит ли какая-либо из записей результирующего списка это поле. Если совпадений не найдено, в результирующий список добавляется указатель на соответствующую запись. Наконец, для вывода списка на экран создаётся массив строк размера, соответствующего длине получившегося списка, в который помещаются найденные ФИО, и передаётся функции, осуществляющей вывод на экран. Перечень адвокатов формируется функцией:

static wndresult\_t PrintLawyers(const struct List \*dbcontents);

Для поиска вкладчика запрашивается ввод начальной строки ФИО вкладчика и начальной строки ФИО адвоката. Затем производится поиск в построенном АВЛ-дереве. Код поиска в дереве в общем аналогичен Лабораторной работе №2 части 2 курса, с двумя особенностями. Во-первых, вместо проверки ключей на строгое равенство проверяется, начинается ли запись с заданной подстроки. Во-вторых, очевидно, что записей, начинающихся на заданную подстроку, может быть несколько – поэтому после нахождения соответствия функция поиска не останавливается, а продолжается далее в обоих поддеревьях найденной вершины. Поиск выполняется функцией:

static wndresult\_t SearchName(struct Vertex \*treeroot);

Для сравнения строк применена стандартная функция strncmp. Особенность её состоит в том, что, если искомая подстрока имеет нулевую длину, функция возвращает 0, как и в случае равенства строк. Эта особенность для нас полезна, поскольку позволяет пользователю ввести пустую строку для поиска всех элементов, что является интуитивно-понятным поведением.

Описание структур данных и объявления функций, оперирующих данными, содержатся в заголовочном файле datastruct.h:

/\*

\* Структуры данных

\*/

struct DbRecord {char name[DBREC\_NAME\_LENGTH+1]; int sum; char date[DBREC\_DATE\_LENGTH+1]; char lawyer[DBREC\_LAYR\_LENGTH+1];};

//запись из БД, размещаемая в памяти

struct Vertex {struct DbRecord \*data; struct Vertex \*left; struct Vertex \*right; int bal;};

//вершина дерева (хранит указатель на запись из БД)

struct Node {struct DbRecord \*data; struct Node \*previous; struct Node \*next;};

//узел списка (хранит указатель на запись из БД)

struct List {struct Node \*head; struct Node \*tail; size\_t length;}; //список: указатели на заглавный и конечный узлы, общее кол-во узлов

/\*

\* Функции работы с данными

\*/

struct Vertex\* TreeInsert(struct DbRecord \*data, struct Vertex \*root); //добавляет запись из БД в АВЛ-дерево, возвращает вершину дерева

void TreeDestruct(struct Vertex \*root); //удаляет все вершины дерева, высвобождая занятую память

void TreeSearch(const char\* key, struct Vertex \*root, struct List \*result);

//поиск записей в дереве, для которых поле name начинается с key, и добавление ссылок на найденное в список

void ListAppend(struct DbRecord \*data, struct List \*lst); //добавление узла в конец списка

void ListRemove(struct Node \*n, struct List \*lst); //удаление узла из списка

void ListClear(struct List \*lst); //удаление всех узлов списка

Описанные функции, относящиеся к работе со списками, реализованы в файле list.c, а относящиеся к работе с деревом поиска – в файле avltree.c. Содержимое файлов не приводится во избежание избыточного цитирования т.к. полностью код приложен к настоящей работе.

Интерфейс приложения с элементами псевдографики реализован с использованием подключаемой библиотеки PDCurses. Функции пользовательского интерфейса определены в заголовочном файле uicurs.h:

typedef enum WndResult {ERROR=-1, EXIT=0, VOID, DOSEARCH, SHOWLR, SHOWDB} wndresult\_t; //коды возврата функций, оперирующих окнами интерфейса; совместимы с ERR/OK, определёнными PDCurses

//возвращается флаг ошибки ERROR, нормального завершения EXIT, или признак требуемого следующего действия

int UIInit(void); //инициализация интерфейса и PDCurses; возвращает OK/ERR

wndresult\_t UIDisplayDbrecords(const struct List \*lst, const char \*dbname);

//отображение списка, содержащего записи БД, в окне с полосой прокрутки

wndresult\_t UIDisplayStrings(int argc, char \*argv[]); //отображение массива с заданным кол-вом строк

wndresult\_t UIProgressbarDisplay(int progress, char \*title); //вывод полосы - индикатора прогресса (задаётся значение от 0 до 100 и заголовок)

int UIProgressbarDestroy(void); //деинициализировать индикатор прогресса после использования

int UIDestroy(void); //деинициализировать PDCurses после использования

Описанные функции реализованы в файле uicurs.c, содержимое которого не приводится во избежание избыточного цитирования т.к. полностью код приложен к настоящей работе.

Точка входа приложения реализована следующим образом:

int main()

{

//загрузить содержимое файла в ОЗУ

struct List \*dbcontents = calloc(1, sizeof(struct List));

if (ReadDbFile(DBFILENAME, dbcontents)==ERR){

UIDestroy();

return ERR;

}

//инициализация пользовательского интерфейса

if (UIInit() == ERR) return ERR;

//построить из загруженных данных дерево поиска,

отображая индикатор прогресса

struct Vertex \*treeroot = NULL;

size\_t counter = 0;

for (struct Node \*n=dbcontents->head; n!=NULL; n=n->next, counter++){

UIProgressbarDisplay(counter\*100/dbcontents->length,

"Построение АВЛ дерева");

treeroot = TreeInsert(n->data,treeroot);

}

UIProgressbarDestroy();

//отображение окон пользовательского интерфейса в цикле

до завершения программы

wndresult\_t retcode = UIDisplayDbrecords(dbcontents, DBFILENAME);

bool finish = FALSE;

for(;!finish;){

switch(retcode){

case ERROR:

case EXIT:

finish = TRUE;

break;

case DOSEARCH: //запустить поиск записи

retcode = SearchName(treeroot);

break;

case SHOWDB: //вывести содержимое БД

retcode = UIDisplayDbrecords(dbcontents, DBFILENAME);

break;

case SHOWLR: //вывести список адвокатов

retcode = PrintLawyers(dbcontents);

break;

}

}

//по завершению программы деинициализируем интерфейс

UIDestroy();

TreeDestruct(treeroot);

ListClear(dbcontents);

free(dbcontents);

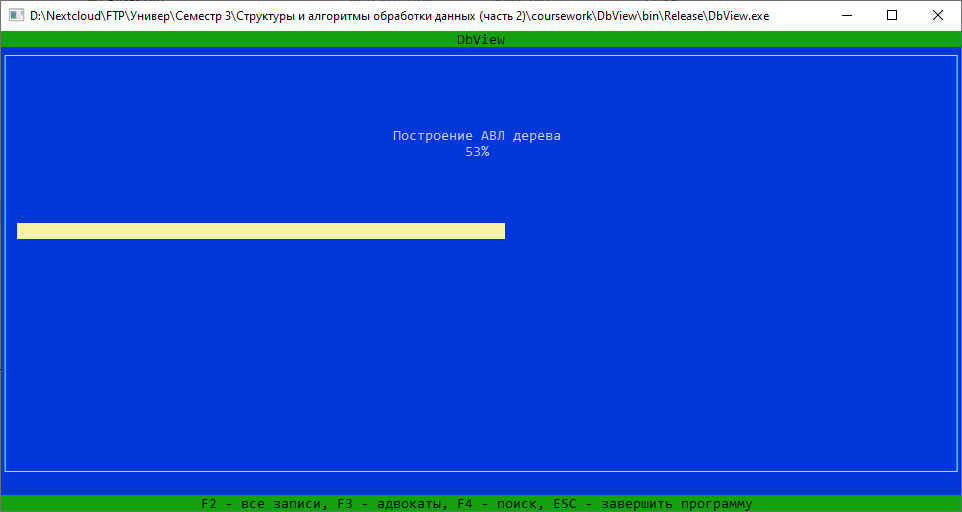
if (retcode == ERROR) return ERR;

else return OK;

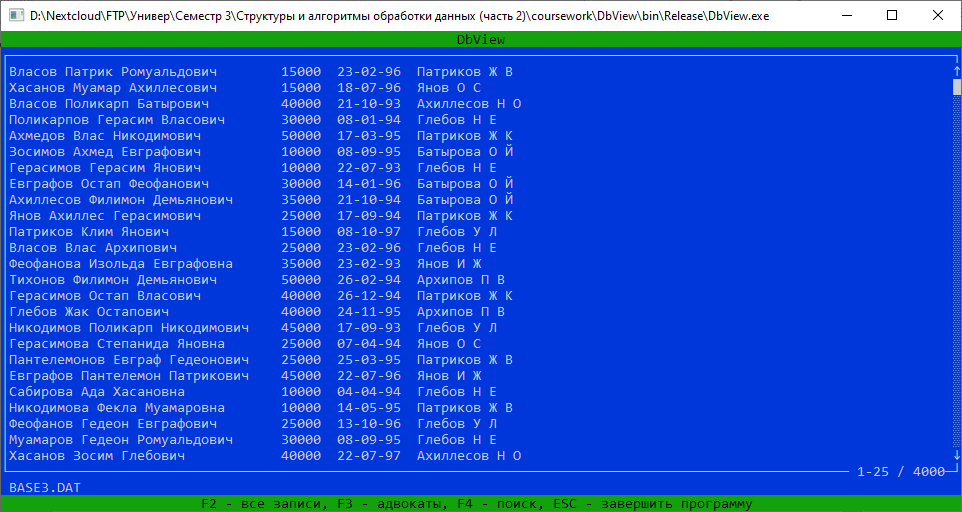
}

# 3. Реализация

При запуске программы производится загрузка файла БД и построение дерева поиска с отображением прогресса:



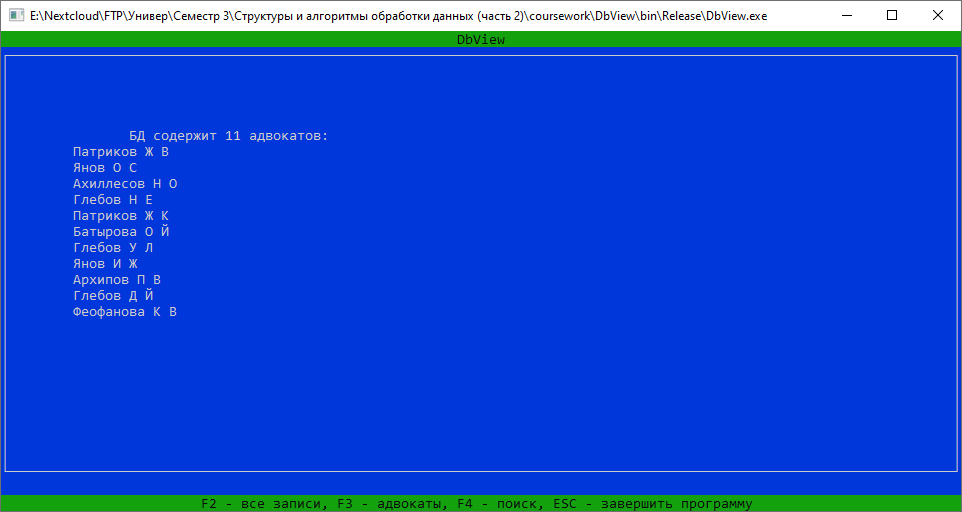
После загрузки данных открывается окно просмотра содержимого БД:



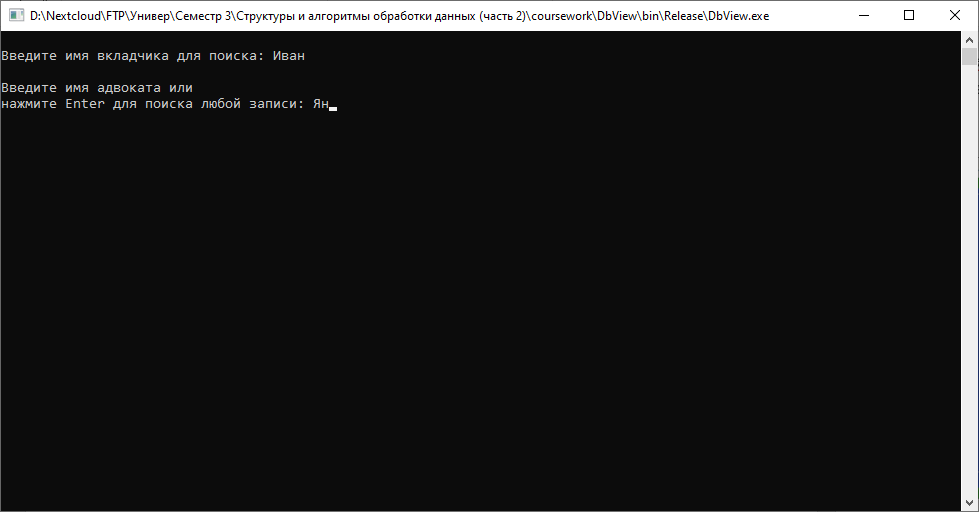
В правой нижней части окна расположен счётчик, показывающий номера отображаемых на экране строк и общее их количество. Расположение ползунка прокрутки справа также примерно соответствует расположению выводимого фрагмента в общем списке строк. В левой нижней части окна отображается имя файла БД.

Для прокрутки по одной строке следует нажимать стрелки вверх/вниз на клавиатуре. Для прокрутки по одному экрану следует нажимать PageUp/PageDown или стрелки вверх/вниз с нажатым Ctrl. Переход к концу или началу списка возможен сочетанием Ctrl+Home/Ctrl+End.

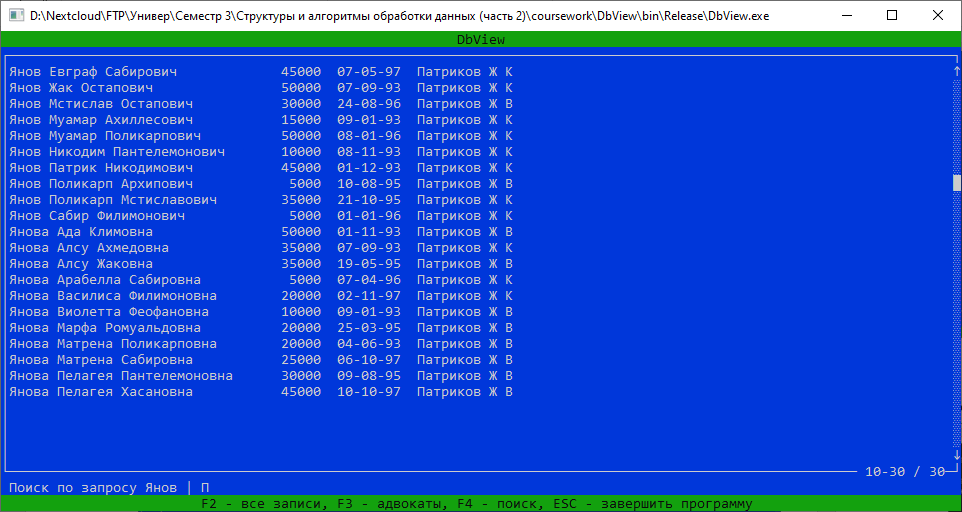
Для отображения информации об адвокатах следует нажать F3:

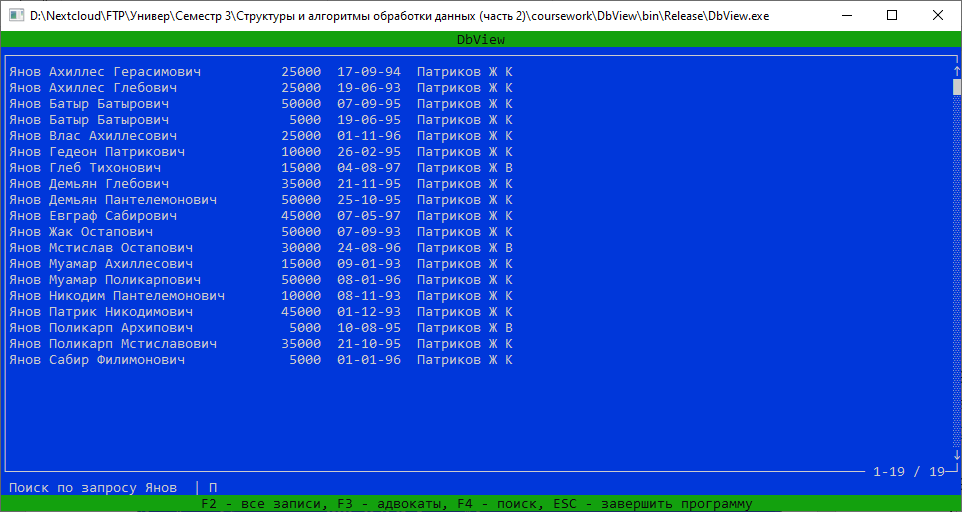


Для поиска следует нажать F4 и ввести начальные символы искомого ФИО вкладчика и адвоката (поиск регистро-чувствительный). Ввод завершается по нажатию Enter. Поисковая выдача сортируется по ФИО вкладчика. Введённые подстроки будут отображены в левом нижнем углу окна с результатами поиска.

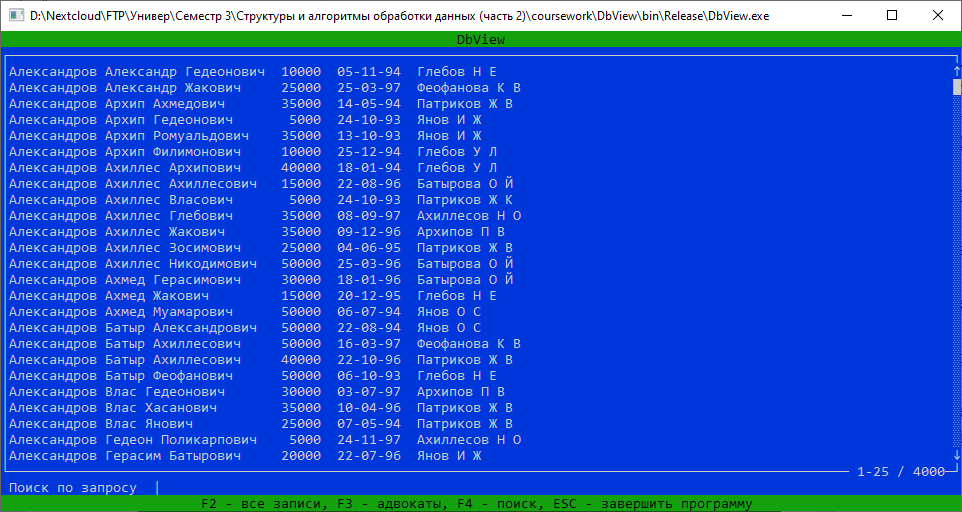


Допустимы пробелы во вводимых строках (например, если ввести «Янов» – будут выведены все вкладчики с ФИО Янов\*, в том числе и «Янова»; а если ввести «Янов » с пробелом в конце – «Янова» перестанет быть подходящим совпадением):

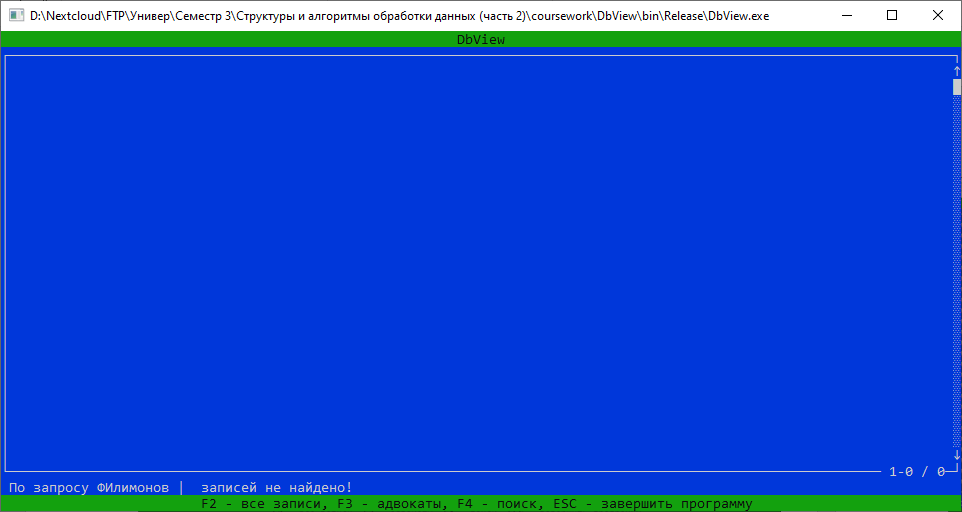




Вводимая строка может быть пустой, тогда будут выведены любые записи. Например, если ввести пустую строку ФИО вкладчика, но задать ФИО адвоката – будут выведены все вкладчики, для которых указан данный адвокат, и наоборот, если ввести ФИО вкладчика и пустую строку ФИО адвоката – будут выведены все подходящие вкладчики с любыми адвокатами. Крайний случай, когда обе строки пустые – выведет все записи БД, отсортированные по алфавиту:



Если релевантные записи найти не удалось, в левом углу окна будет выведено соответствующее сообщение:



Из любого окна можно переходить в любое нажатием клавиши Fx. Завершается работа программы нажатием клавиши Escape.

Для успешного запуска в одном каталоге с программой должна быть размещена подключаемая библиотека pdcurses.dll и файл базы данных BASE3.DAT.

# 4. Список литературы

1. Самуйлов С.В. Алгоритмы и структуры обработки данных [Электронный ресурс]: учебное пособие/ Самуйлов С.В.— Электрон. текстовые данные.— Саратов: Вузовское образование, 2016.— 132 c.— Режим доступа: <http://www.iprbookshop.ru/47275> по паролю— ЭБС «IPRbooks».

2. Курапова Е.В. Структуры и алгоритмы обработки данных [Электронный ресурс]: лабораторный практикум/ Курапова Е.В., Мачикина Е.П.— Электрон. текстовые данные.— Новосибирск: Сибирский государственный университет телекоммуникаций и информатики, 2015.— 23 c.— Режим доступа: <http://www.iprbookshop.ru/55501> по паролю — ЭБС «IPRbooks».

3. Miller B., Ranum D. Problem Solving With Algorithms and Data Structures [Электронный ресурс] – Режим доступа: <https://github.com/aliev/runestone>

3. Керниган, Брайан У., Ритчи, Деннис М. Язык программирования C, 2-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2009. – 304 с.

4. Standard C and C++ complete online reference [Электронный ресурс] – Режим доступа: <https://en.cppreference.com/w/>

5. PDCurses Documentation [Электронный ресурс] – Режим доступа: <https://pdcurses.org/docs/>

6. Curses library – IBM AIX 7.2 Documentation [Электронный ресурс] – Режим доступа: <https://www.ibm.com/docs/en/aix/7.2?topic=concepts-curses-library>