Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра интеллектуальных информационных технологий

Отчёт
по курсу «**Естественно-языковой интерфейс интеллектуальных систем**»

Лабораторная работа №1
«Разработка автоматизированной системы формирования словаря естественного языка»

Выполнили студенты группы 121701:	Чвилёв И.А. Воронцов Р.Г. Силибин С.
Проверил:	Крапивин Ю.Б.

Минск 2024

Цель работы: Освоить принципы разработки прикладных сервисных программ для решения задачи автоматического лексического и лексико-грамматического анализа текста естественного языка.

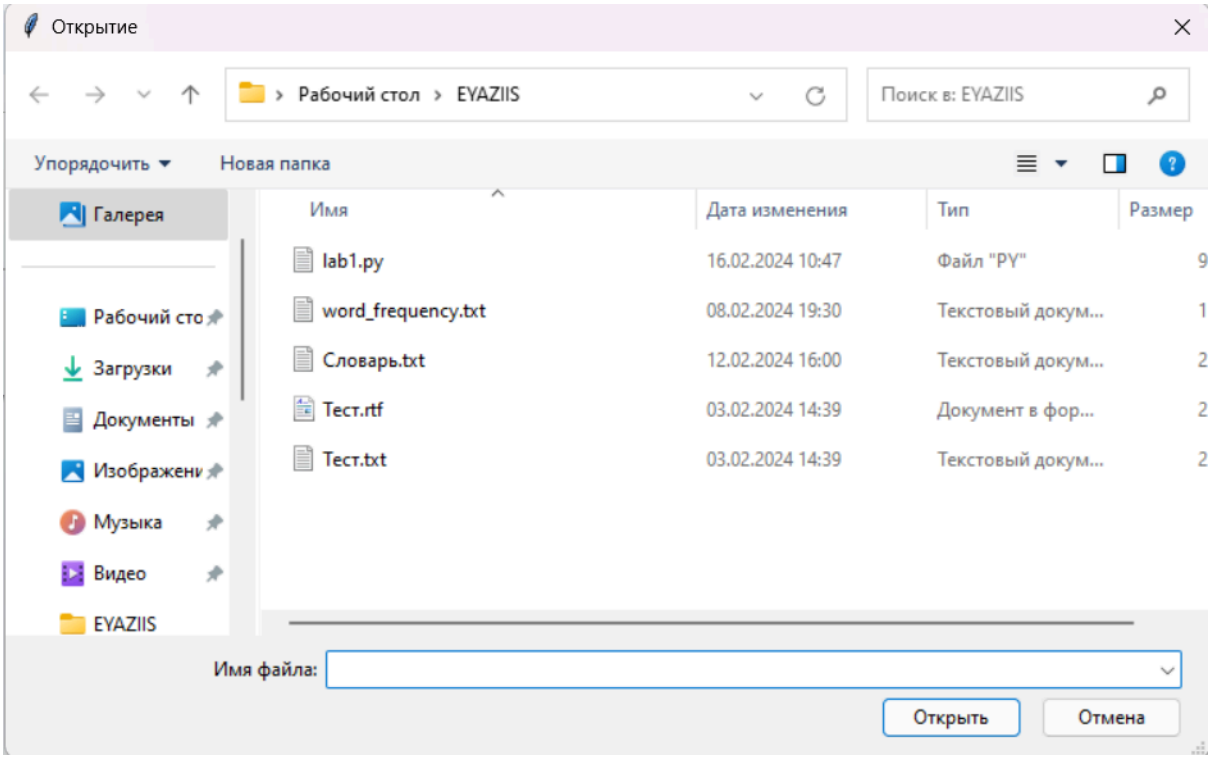
Задание: Список слов, упорядоченный по алфавиту и включающий как лексемы, так и словоформы, с указанием частоты встречаемости каждой из форм. Для словоформ пользователю должна быть предоставлена возможность вводить дополнительную морфологическую информацию, а именно, отнесение слова к соответствующей части речи, указание рода, числа, падежа и т.п. При этом морфологическая информация может быть оформлена как отдельная неформатированная запись, т.е. это просто текст, который пользователь может оформлять произвольным образом. Язык текста – русский, формат входного документа – TXT, RTF.

Алгоритм работы программы:

1) Открывается диалог выбора файла на диске

```
def start(self):
    filename = self.select_file()

    if filename:
        self.parse_text(filename)
        self.launch_app()
```

a) Пользователь выбирает текстовый файл, переход к шагу 2

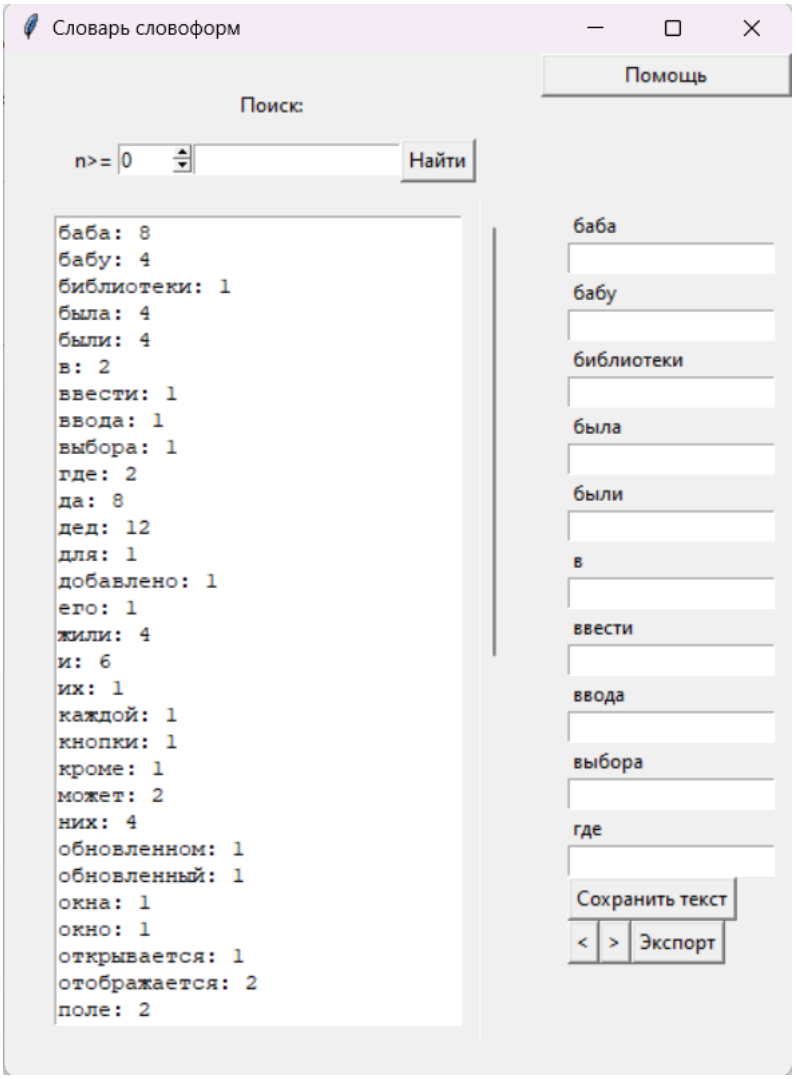


2) Программа проходит по всему тексту, получает все слова из файла, используя регулярное выражение, добавляет их в словарь, где ключ - уникальные слова в нижнем регистре, а значение - количество их вхождений в тексте, а затем сортирует.

```
def parse_text(self, filename):
    with open(filename, 'r', encoding='utf-8') as file:
        text = file.read()

    words = re.findall(r'\b\w+\b', text)
    word_freq = defaultdict(int)
    for word in words:
        word_freq[word.lower()] += 1
    sorted_word_freq = dict(sorted(word_freq.items(), key=lambda x: x[0]))
    self.word_freq = sorted_word_freq
```

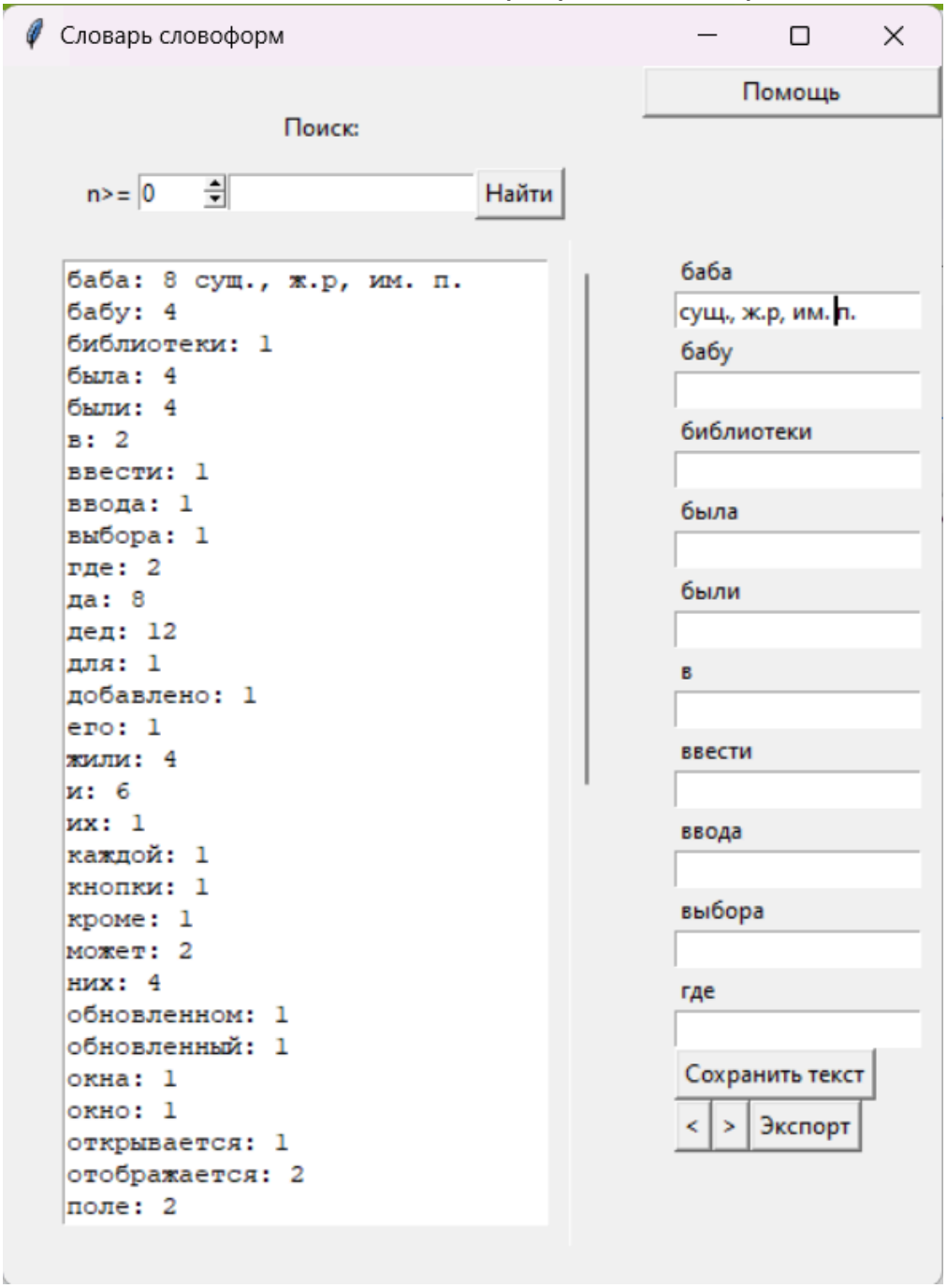
3) Программа формирует основное окно интерфейса, выводит сформированный словарь в список слева. Формирует список из первых десяти слов, выводит их справа, к каждому слову добавляется поле для ввода. Формируются поля для поиска, фильтрации по длине слова, вкладка помощи, кнопки для переключения страницы, сохранения текста, экспорта.

3.1) Программа ожидает действия пользователя.

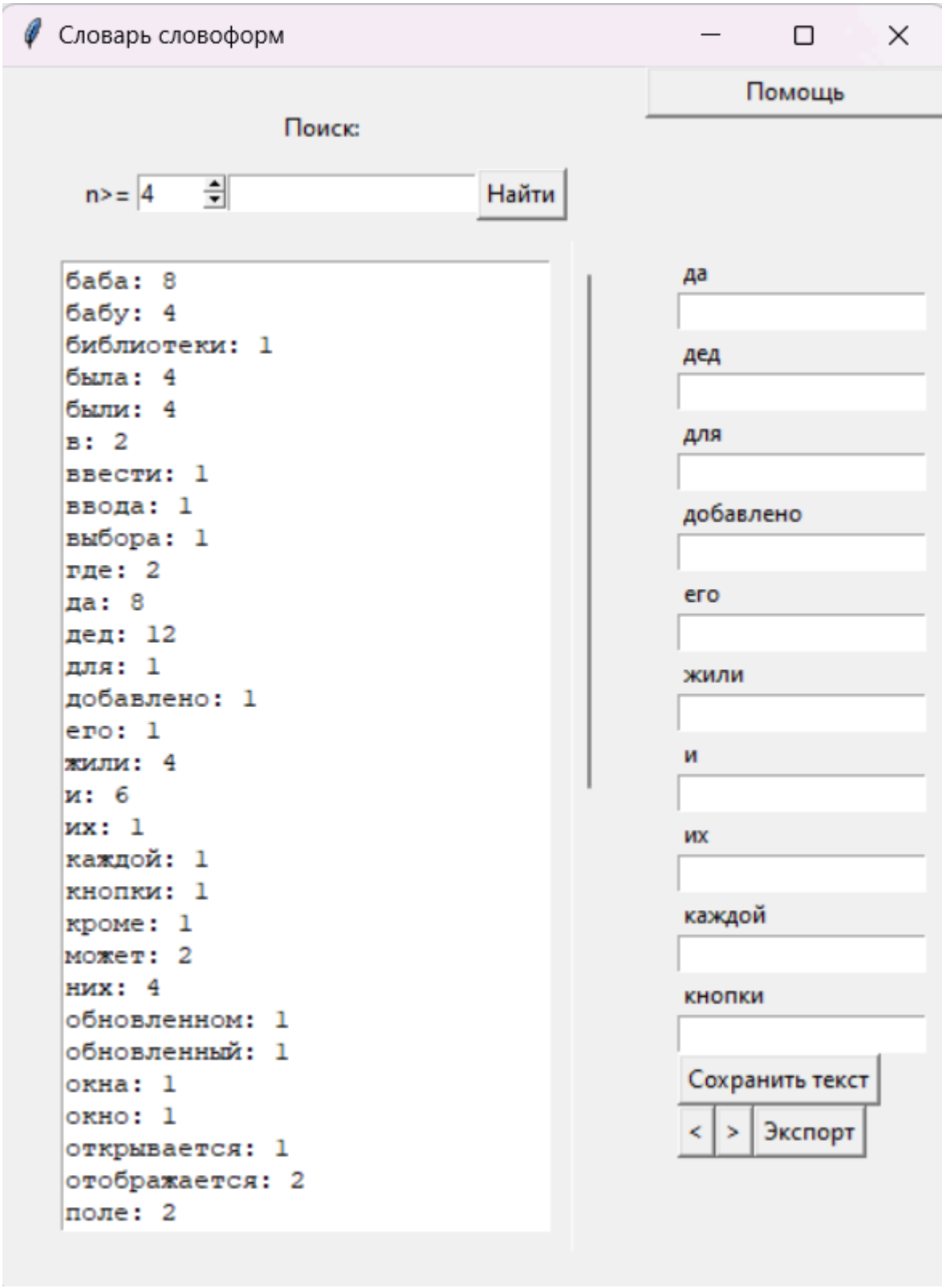


а) Пользователь вводит информацию в одном из полей справа и нажимает “Сохранить текст”. Программа переходит к шагу 4.
б) Пользователь нажимает на одну из кнопок “<”,”>”. Программа переходит к шагу 5.
в) Пользователь вводит запрос в строку поиска и нажимает “Найти”. Программа переходит к шагу 6.
г) Пользователь выбирает минимальное значение n, введя число в поле слева от строки поиска и нажимает “Найти”. Программа переходит к шагу 7.
д) Пользователь выбирает минимальное значение n, введя число в поле слева от строки поиска и нажимает “Найти”. Программа переходит к шагу 8.
е) Пользователь нажимает “Экспорт”. Программа переходит к шагу 9.
ж) Пользователь нажимает на кнопку “Помощь”. Программа переходит к шагу 10.
з) Пользователь закрывает программу. Программа переходит к шагу 11.

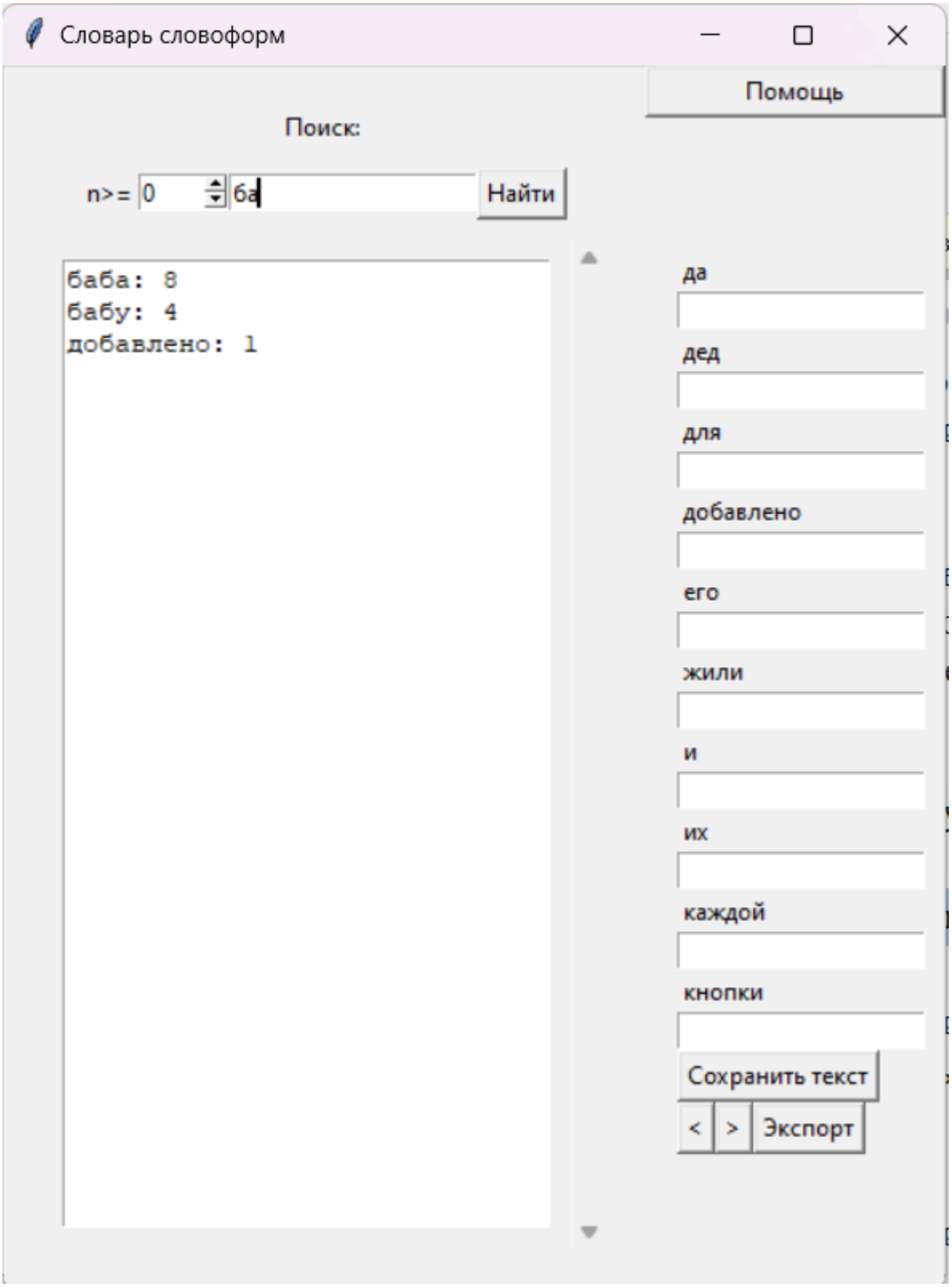
4) Введённая информация сохраняется в отдельном словаре и выводится в главном списке. Программа возвращается к шагу 3.1



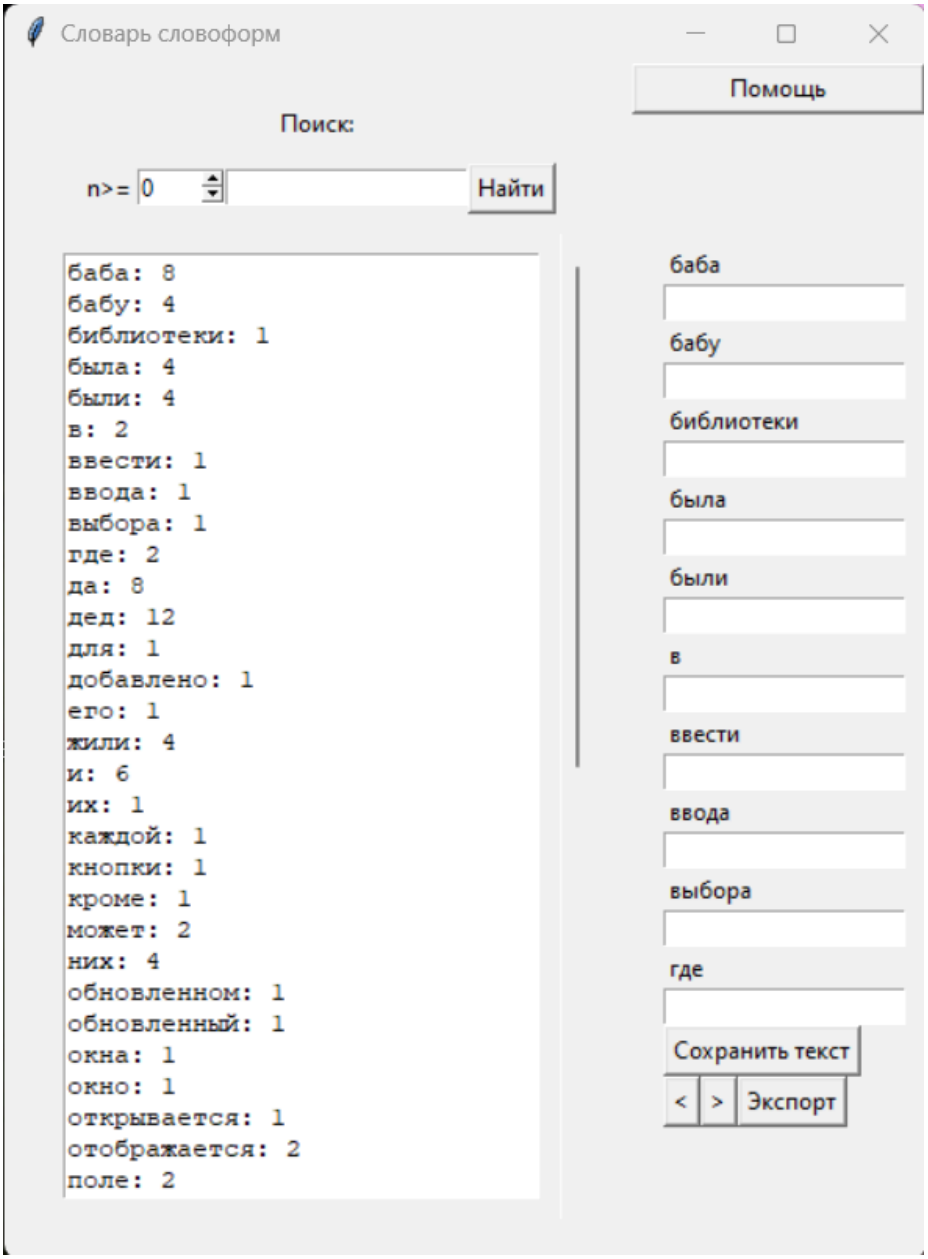
5) Список слов справа меняется на следующие или предыдущие 10 слов. Программа возвращается к шагу 3.1



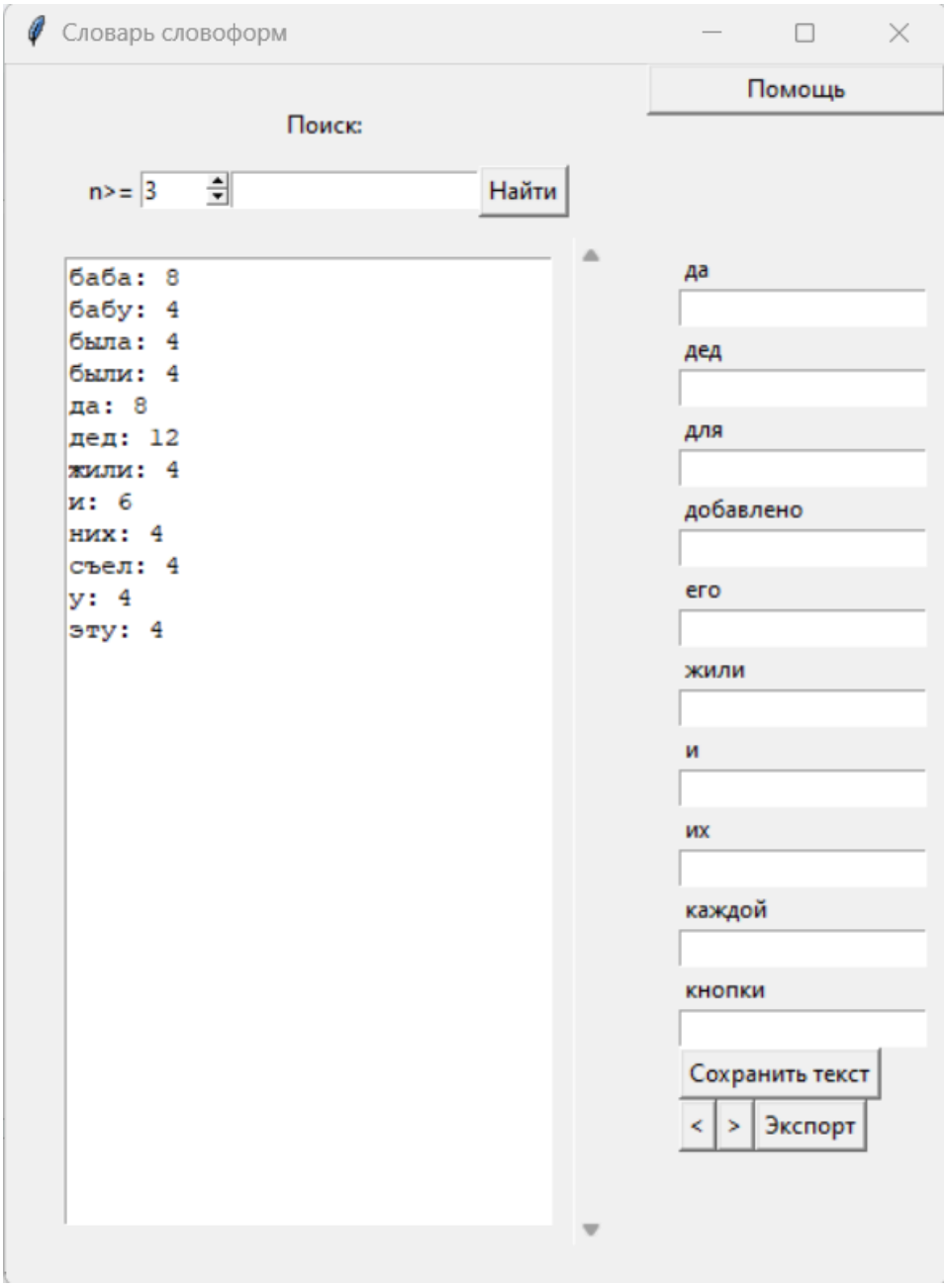
6) Список выведенных слов обновляется, включая только те, которые содержат значение поиска. Программа возвращается к шагу 3.1



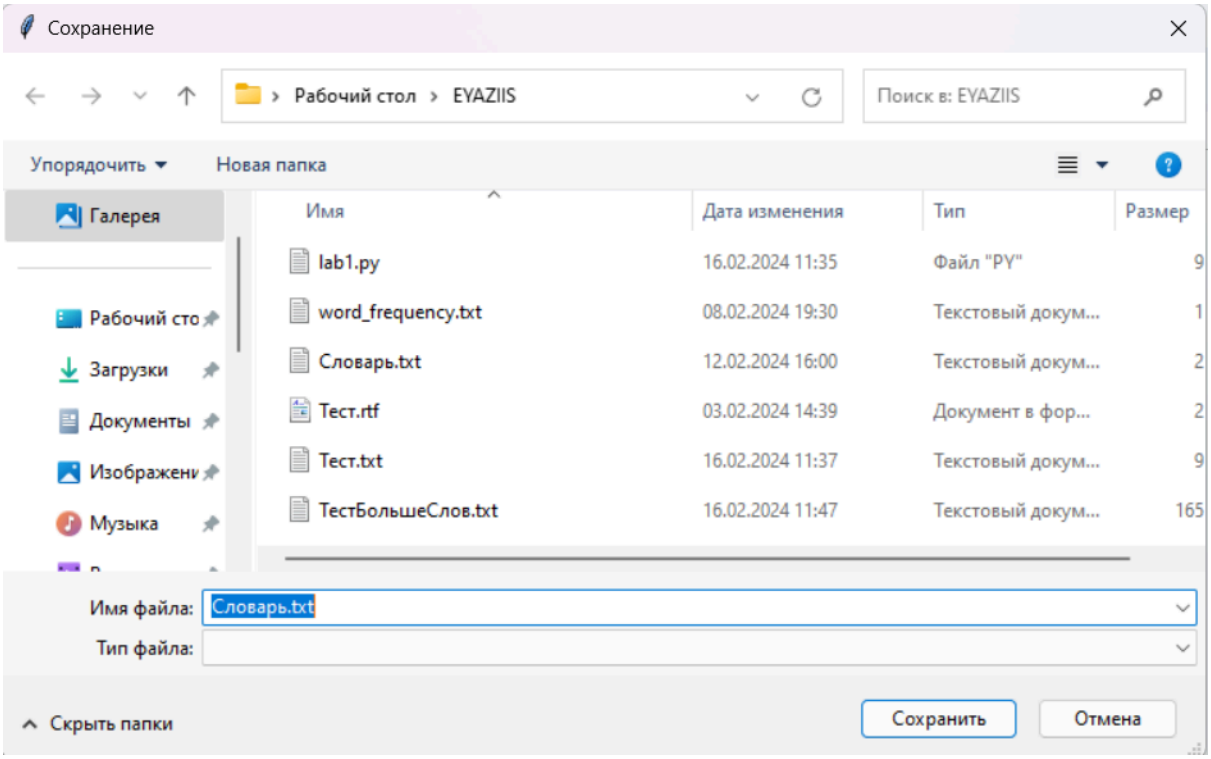
7) Список выведенных слов обновляется, включая только те, которые содержат значение поиска. Программа возвращается к шагу 3.1



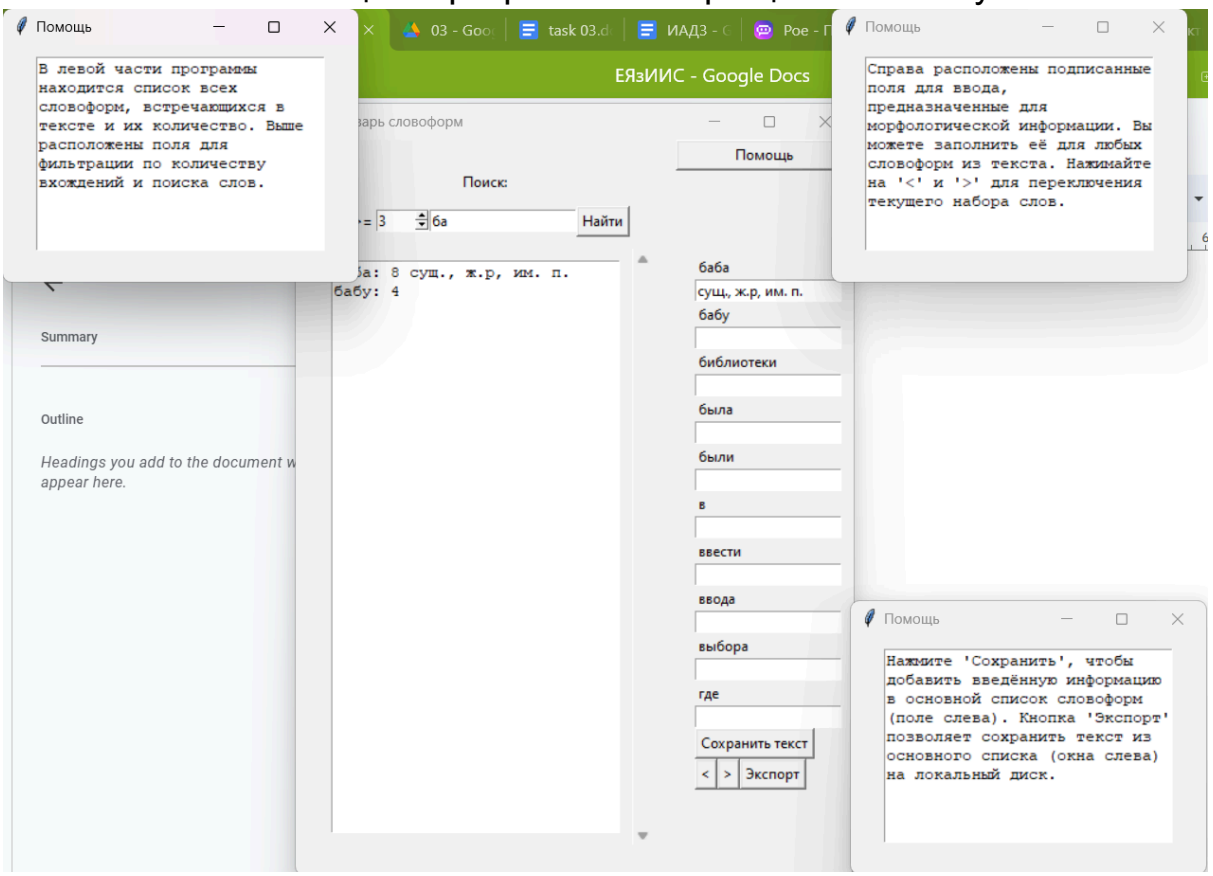
8) Список выведенных слов обновляется, включая только те, длина которых больше или равно значению в поле “n>=”. Программа возвращается к шагу 3.1



9) Программа предлагает пользователю выбрать имя файла и расположение на диске, куда этот файл будет сохранён. Затем сохраняет и возвращается к шагу 3.1

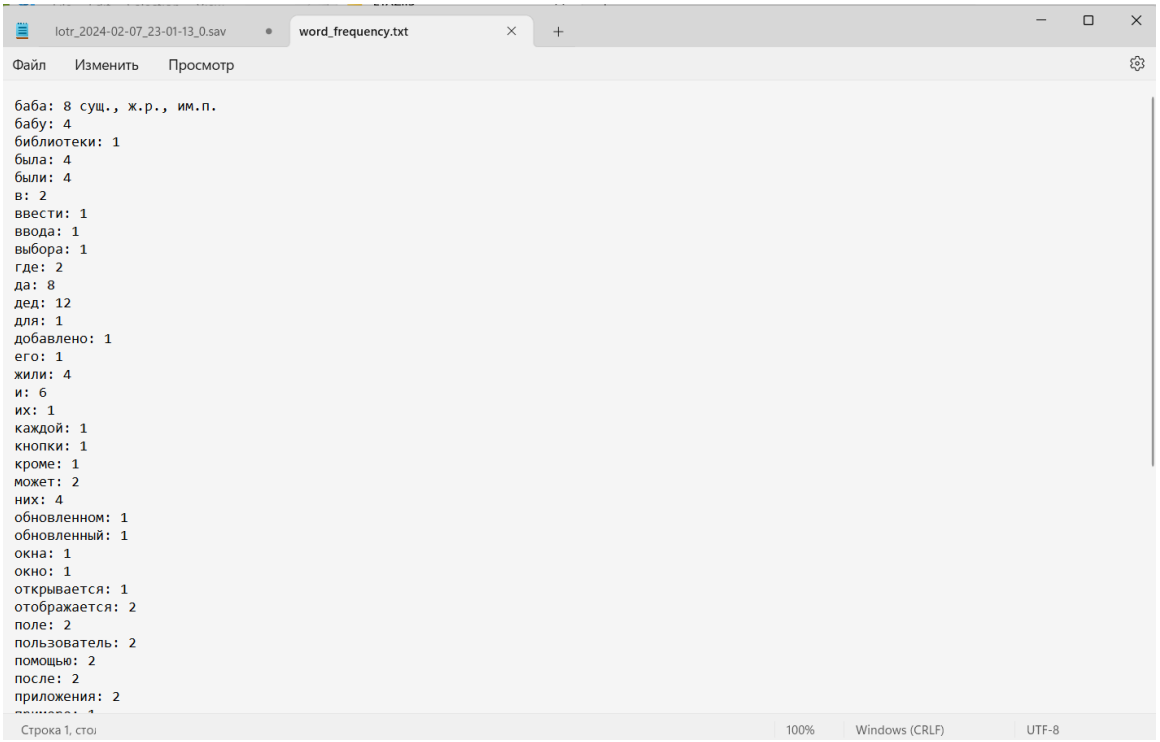


10) Открываются 3 окна-подсказки, объясняющие пользователю что, где и для чего расположено. Закрыть их можно повторным нажатием на “Помощь”. Программа возвращается к шагу 3.1



11) Программа завершает работу.

Пример экспортированного файла



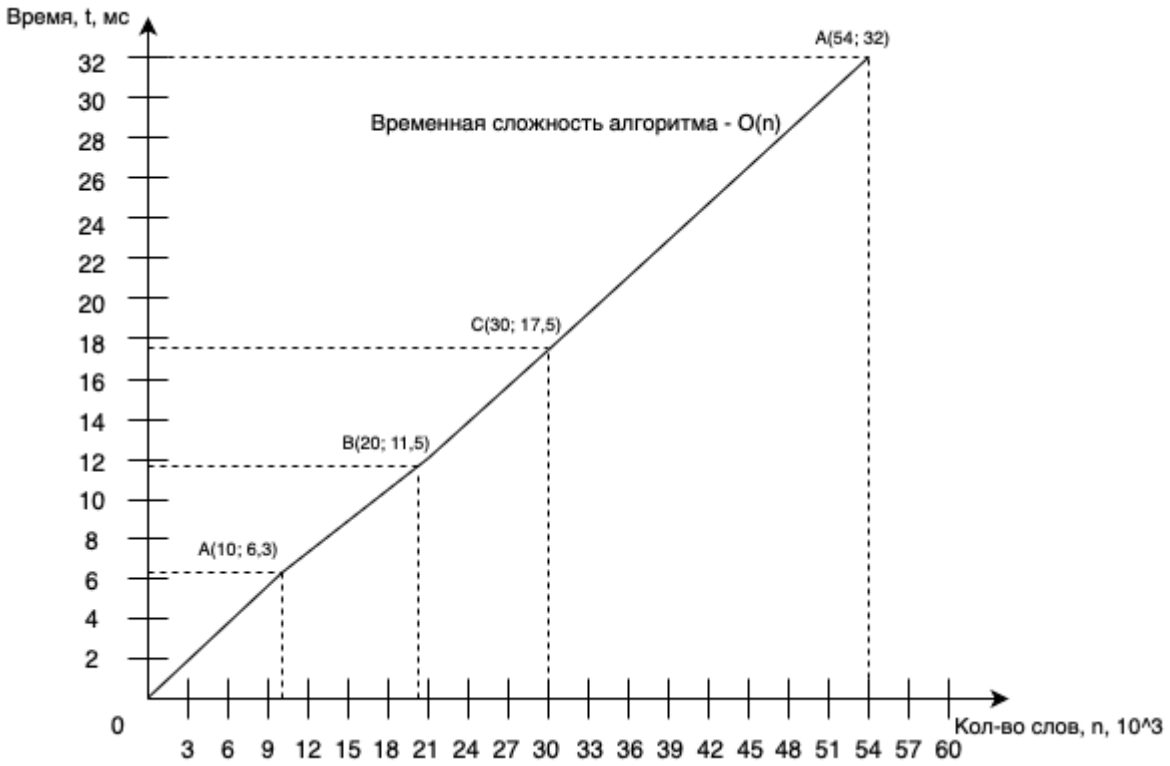
Использованные структуры хранения данных:
Хранение данных производится в самой программе на Python, в типе `defaultdict`, то есть словаре. Таких словарей 2:
word_freq:
ключ - слово из текста
значение - количество вхождений данного слова тексте
Заполняется после выбора текстового документа. Используется для отображения основного списка в поле слева.

```
def parse_text(self, filename):
    dt = datetime.datetime.now()
    with open(filename, 'r', encoding='utf-8') as file:
        text = file.read()

    words = re.findall(r'\b\w+\b', text)
    word_freq = defaultdict(int)
    for word in words:
        word_freq[word.lower()] += 1
    sorted_word_freq = dict(sorted(word_freq.items(), key=lambda x: x[0]))
    self.word_freq = sorted_word_freq

    print(datetime.datetime.now() - dt)
```

word_desc:
ключ - слово из текста

значение - введённая пользователем информация
Изначально пуст, пополняется при добавлении информации пользователем. Также добавляет информацию в поле слева. Влияет и на результат экспорта.

```
def save_user_texts(self):
    for i in range(self.input_frame_items_per_page):
        self.word_desc[self.input_frame_labels[i].cget("text")] = self.input_frame_texts[i].get()
    self.update_text_box()
```



Вывод: Освоены принципы разработки прикладных сервисных программ для решения задачи автоматического лексического и лексико-грамматического анализа текста естественного языка. Разработанное приложение полезно для составления статистики и расчета количества определенных слов в документах.