МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ  
ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ (МАДИ)

Кафедра  
 «Автоматизированные системы управления»

**Курсовая работа по дисциплине   
«Алгоритмы и структуры данных»**

Вариант № 37

Выполнил: студент группы 2бИТС2

Лобосов Н.В.

Проверил: доцент каф. АСУ Куфтинова Н.Г.

Москва 2025

**Оглавление**

[**Введение** 3](#_Toc192623142)

[**1.** **Обоснование выбора ЯП** 5](#_Toc192623143)

[**2.** **Описание алгоритма на псевдокоде** 7](#_Toc192623144)

[**3.** **Описание выполнения алгоритма** 9](#_Toc192623145)

[**Заключение** 19](#_Toc192623146)

[**Список источников** 20](#_Toc192623147)

[**Приложение** 21](#_Toc192623148)

# **Введение**

В современных условиях информационных технологий обработка текстовых данных играет важную роль. Одним из ключевых аспектов работы с текстовыми файлами является возможность редактирования и упорядочивания их содержимого, что делает актуальной разработку программных инструментов для эффективного управления текстовой информацией.

Целью данной курсовой работы является разработка программного приложения, позволяющего загружать текст из файла, отображать его в графическом интерфейсе и выполнять выбор отдельных строк, для следующей группировки в таблицу.

Работа с текстовыми файлами является неотъемлемой частью деятельности пользователей различных сфер – от программирования и анализа данных до ведения документации и работы с базами данных. Во многих случаях возникает необходимость реорганизации текстовой информации, что требует удобного пользовательского интерфейса и эффективных алгоритмов обработки данных. Предлагаемое программное решение обеспечит пользователю удобные механизмы перемещения строк между окнами, предоставляя возможность организовывать текстовые данные наиболее удобным способом. Разработка подобного инструмента является актуальной задачей, так как существующие текстовые редакторы не всегда обладают необходимой функциональностью для удобного перемещения строк между областями отображения.

Для достижения поставленной цели необходимо решить следующие задачи:

* Создание окна для загрузки строк с текстом.
* Обеспечение возможности выбора отдельных строк в окне.
* Реализация составления таблицы из раздельных строк.
* Реализации возможности скачивания таблицы.

Для реализации проекта будет использоваться язык программирования Python с графической библиотекой, что обеспечит удобство создания пользовательского интерфейса. Будут применены алгоритмы работы с текстовыми файлами и структурой данных для удобного хранения и обработки строк.

# **Обоснование выбора ЯП**

Для реализации данной курсовой работы был выбран язык программирования Python, а также библиотека Tkinter для создания графического интерфейса. Такой выбор обусловлен рядом преимуществ, которые делают Python оптимальным вариантом для решения поставленных задач.

1. Удобная разработка приложений

Python – это высокоуровневый язык программирования, известный своей лаконичностью и понятным синтаксисом. Благодаря этому разработка приложения становится более удобной, а код – читаемым и легко поддерживаемым. Это особенно важно при создании пользовательских интерфейсов, где требуется обработка событий и взаимодействие с графическими элементами.

1. Встроенные библиотеки и кроссплатформенность

Python имеет мощную стандартную библиотеку, включающую Tkinter – встроенный инструмент для создания оконных приложений. Использование Tkinter позволяет избежать необходимости установки сторонних библиотек. Легкость разработки – упрощенная работа с виджетами (кнопки, текстовые поля, списки) позволяет быстро создать удобный интерфейс. Кроссплатформенность – приложения на Tkinter работают на всех операционных системах, поддерживающих Python (Windows, macOS, Linux). Небольшой размер и низкие системные требования – программа, написанная с использованием Tkinter, не требует больших ресурсов и может работать даже на слабых компьютерах. Существуют другие библиотеки для создания графических интерфейсов, такие как PyQt и Kivy, однако они имеют более сложный синтаксис, требуют установки дополнительных компонентов и занимают больше ресурсов. Для данной курсовой работы Tkinter является оптимальным выбором благодаря своей простоте, легкости и встроенной поддержке в Python

1. Гибкость и расширяемость

Данный язык поддерживает большое количество библиотек для работы с файлами, обработки текста и создания графических интерфейсов. В рамках данной курсовой работы можно расширять функциональность программы, добавляя, например: поддержку других форматов файлов (CSV, JSON); дополнительные стили оформления интерфейса; функции поиска и сортировки строк.

1. Поддержка работы с текстовыми файлами

Язык предоставляет удобные механизмы для работы с текстовыми файлами, что является ключевым аспектом данной работы. Встроенные функции позволяют легко загружать данные, изменять их и сохранять обратно в файл, используя всего несколько строк кода.

1. Доступность документации

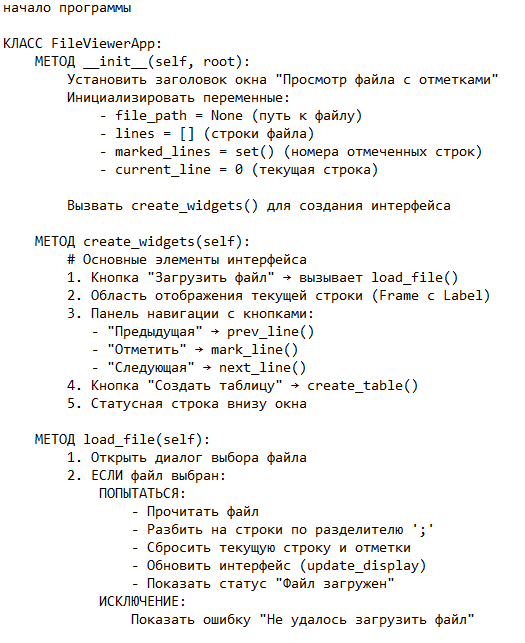
Python – один из самых популярных языков программирования в мире, что означает наличие обширной документации и активного сообщества. Это облегчает поиск решений в случае возникновения сложностей при разработке.

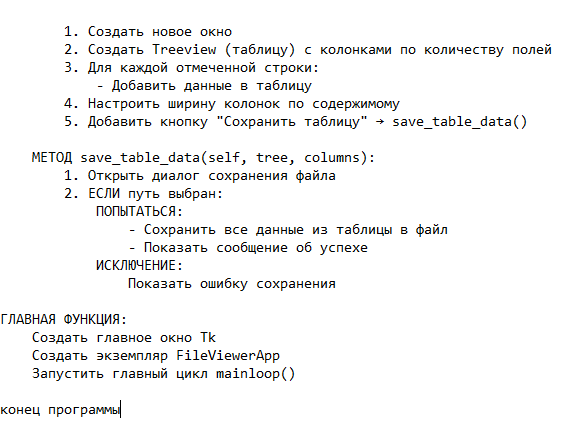
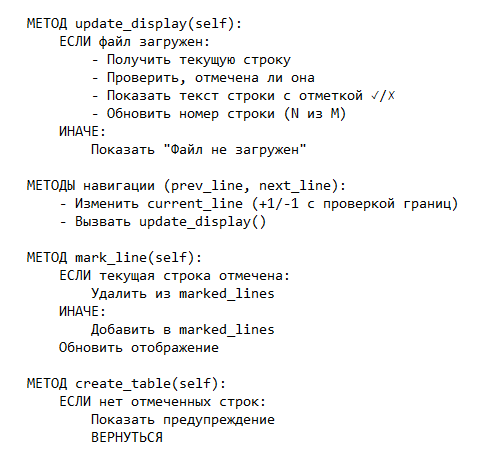
Выбор Python и Tkinter позволяет создать удобное, функциональное и кроссплатформенное приложение для работы с текстовыми файлами. Такой подход обеспечивает легкость разработки, простоту в использовании и возможность дальнейшего расширения функционала программы.

# **Описание алгоритма на псевдокоде**

**Псевдокод** — компактный язык описания алгоритмов, использующий ключевые слова императивных языков программирования, но опускающий несущественные для понимания алгоритма подробности и специфический синтаксис.

Ниже представлен алгоритм программы, написанный на псевдокоде:





# **Описание выполнения алгоритма**

Код представляет собой приложение на языке Python с использованием библиотеки**Tkinter,** которая предназначена для создания графических интерфейсов пользователя (GUI). В данном коде реализована программа с двумя окнами, которые позволяют перемещать и копировать строки текста между ними, а также сохранять текст в новый файл. **filedialog** — подмодуль библиотеки tkinter, используемый для работы с файловыми диалогами (открытие и сохранение файлов).

Рассмотрим использованные методы, функции, классы и библиотеки:

**Tkinter** — это стандартная библиотека Python для создания графических пользовательских интерфейсов (GUI). Она предоставляет все необходимые инструменты для создания окон, кнопок, меток, текстовых полей и других элементов управления в приложении.

***Классы***

**tk.Tk()** — создаёт главное окно приложения.

**tk.Frame()** — создаёт фрейм (контейнер для других элементов) внутри главного окна.

**tk.Text()** — создаёт многострочное текстовое поле, которое используется для отображения и редактирования текста.

**tk.Button()** — создаёт кнопку для взаимодействия с пользователем.

**tk.Label()** — создаёт метку для отображения текста.

***Методы и функции***

**pack(), grid()** — методы, определяющие, как элементы размещаются внутри контейнера (фрейма или главного окна).

**bind()** — метод для привязки событий (например, кликов) к действиям.

1. \_\_init\_\_(self, root)

Конструктор класса FileViewerApp:

• Инициализирует главное окно приложения с заголовком "Просмотр файла с отметками"

• Создает начальные переменные состояния:

- file\_path - путь к текущему файлу

- lines - список строк файла

- marked\_lines - множество отмеченных строк

- current\_line - индекс текущей отображаемой строки

• Вызывает метод create\_widgets() для построения интерфейса

2. create\_widgets(self)

Метод создания графического интерфейса:

• Создает кнопку "Загрузить файл" (вызывает load\_file())

• Создает область отображения текущей строки с рамкой

• Добавляет панель навигации с кнопками:

- "Предыдущая" (prev\_line())

- "Отметить" (mark\_line())

- "Следующая" (next\_line())

• Добавляет кнопку "Создать таблицу" (create\_table())

• Создает статусную строку внизу окна

3. load\_file(self)

Метод загрузки файла:

• Открывает диалоговое окно выбора файла

• Читает содержимое файла с кодировкой UTF-8

• Разбивает содержимое на строки по разделителю ';'

• Сбрасывает состояние текущей строки и отметок

• Обновляет отображение и статусную строку

• В случае ошибки выводит сообщение

4. update\_display(self)

Метод обновления отображения:

• Проверяет, загружен ли файл

• Форматирует текущую строку с отметкой (✓/✗)

• Обновляет текст в области отображения

• Показывает номер текущей строки и общее количество строк

5. prev\_line(self)

Метод навигации:

• Уменьшает индекс текущей строки на 1 (не ниже 0)

• Вызывает обновление отображения

6. next\_line(self)

Метод навигации:

• Увеличивает индекс текущей строки на 1 (не выше количества строк)

• Вызывает обновление отображения

7. mark\_line(self)

Метод отметки строк:

• Добавляет/удаляет текущую строку в множество отмеченных

• Обновляет отображение для показа изменения отметки

8. create\_table(self)

Метод создания таблицы:

• Проверяет наличие отмеченных строк

• Создает новое окно с таблицей (Treeview)

• Настраивает столбцы таблицы по количеству полей в данных

• Заполняет таблицу отмеченными строками

• Автоматически подбирает ширину столбцов

• Добавляет кнопку сохранения таблицы

9. save\_table\_data(self, tree, columns)

Метод сохранения таблицы:

• Открывает диалоговое окно сохранения файла

• Сохраняет данные таблицы в выбранный файл

• Поддерживает форматы TXT и CSV

• Выводит сообщение об успешном сохранении или ошибке

**Метод open() –** используется для открытия файла и возвращает объект файла, с которым можно работать (чтение, запись и т. д.).

**Метод read() –** используется для чтения содержимого файла. Он считывает весь файл и возвращает его в виде строки.

**Метод readline() –** используется для чтения файла построчно. Каждый вызов метода читает одну строку.

**Метод readlines() –** считывает все строки файла и возвращает их в виде списка.

**Метод write() –** используется для записи текста в файл. Он записывает строку в файл, начиная с текущей позиции указателя (если файл открыт в режиме записи или добавления).

**Метод writelines() –** записывает все строки из переданного списка в файл.

**Метод close() –** используется для закрытия файла после завершения работы с ним. Очень важно закрывать файл после чтения или записи, чтобы освободить системные ресурсы.

**with (Контекстный менеджер)** использование конструкции**`with`** является более безопасным способом работы с файлами, так как она автоматически закрывает файл, даже если произошла ошибка во время работы с ним. Это избавляет от необходимости вручную вызывать close().

**Метод index() –** используется для поиска индекса (положения) первого вхождения элемента в коллекцию (например, в строку или список). Он возвращает индекс первого элемента, который соответствует заданному значению. Если элемент не найден, вызывается исключение ValueError.

**Метод get() –** используется в**словарях (dict)** для извлечения значения по ключу. В отличие от обычного обращения через [], метод**`get()`** безопасен: если ключ не найден, он возвращает None или заданное значение по умолчанию, вместо того чтобы выбрасывать исключение KeyError.

**Метод split() –** используется для разделения строки на части (подстроки) по указанному разделителю (по умолчанию — по пробелам). Этот метод возвращает список подстрок.

**Метод join() –** используется для объединения элементов коллекции (например, списка) в строку с заданным разделителем.

**Метод replace() –** используется для замены части строки на другую подстроку.

**Метод strip() –** используется для удаления ведущих и завершающих пробелов или других символов в строке.

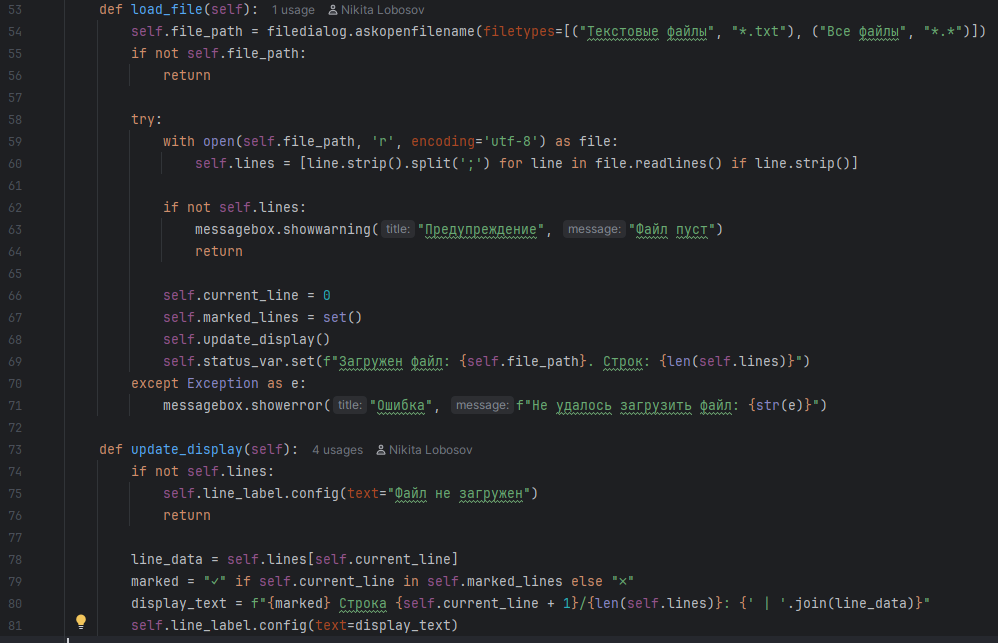
**Метод find() –** используется для поиска первого вхождения подстроки в строке. Он возвращает индекс первого вхождения или -1, если подстрока не найдена (в отличие от index(), который вызывает исключение).

**Метод count()** – используется для подсчета количества вхождений подстроки в строку.

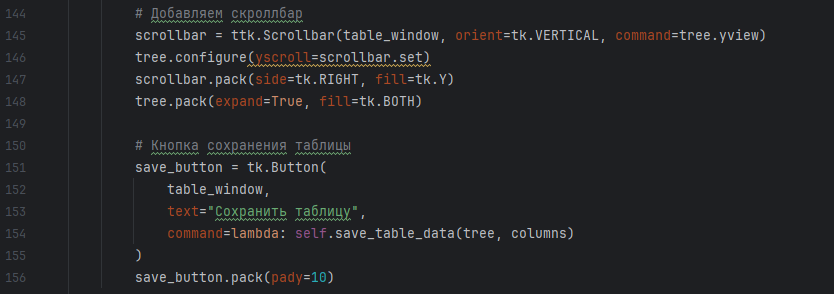
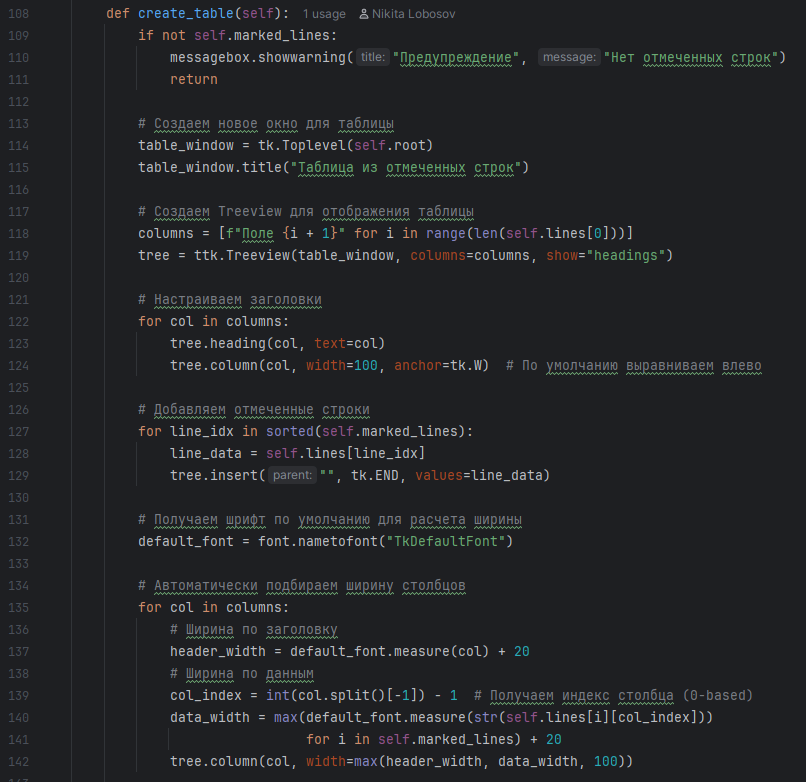
**Метод enumerate() –** используется для получения индексов элементов в итерабельных коллекциях, таких как списки или строки. Этот метод полезен, когда нужно одновременно получать и индекс, и значение.

**Метод zip() –** используется для объединения нескольких коллекций в кортежи (например, для объединения двух списков).

На рисунке 1-2 показан фрагмент кода, реализующий основные функции и методы.



*Рисунок 1*



*Рисунок 2*

На рисунке 3 показаны окна, которое отображаются при запуске программы.

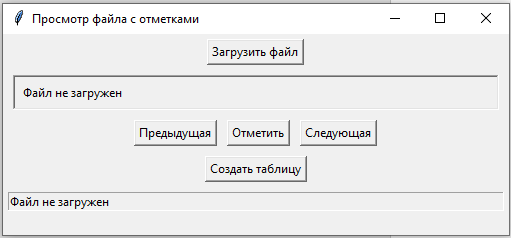


Рисунок 3 – основные окна программы, которое выводятся на экран при запуске программы

На рисунке 4-9 показан основной интерфейс программы.

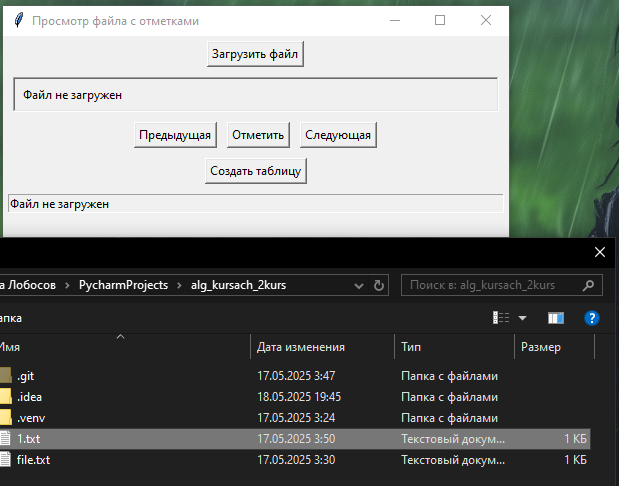


Рисунок 4 – осуществление выбора файла для загрузки строк

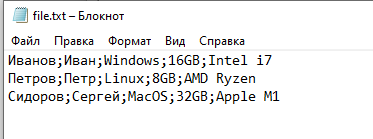


Рисунок 5 – информация в выбранном файле

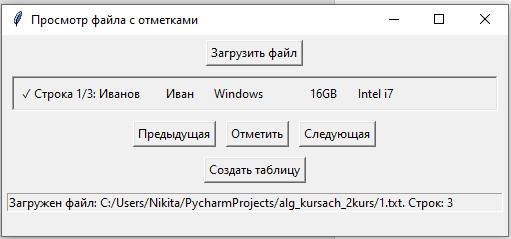


Рисунок 6 – выбор нужной строки (кнопка “отметить”)

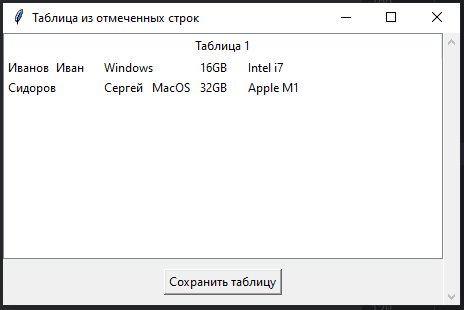


Рисунок 7 – создание таблицы (выбраны 1 и 3 строки)

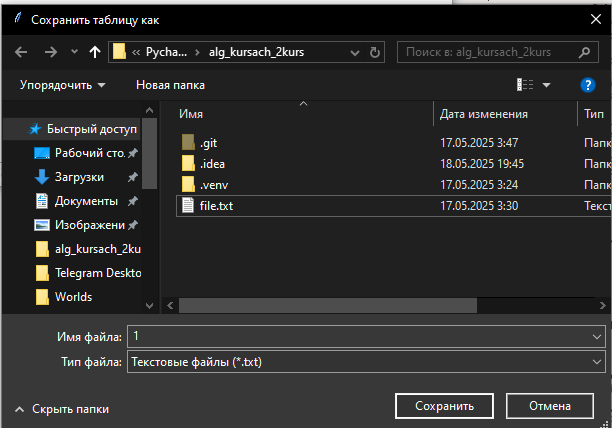


Рисунок 8 – сохранение таблицы в отдельном файле

На рисунке 9 показано сообщение, которое будет выведено на экран, когда таблица успешно сохранена.

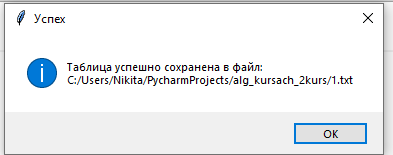


Рисунок 9 – сообщение для пользователя

# **Заключение**

В процессе разработки программного приложения для работы с текстовыми данными была решена задача создания удобного интерфейса, который позволяет пользователю создавать таблицу данных из выбранных строк. Задача заключалась в реализации функционала, обеспечивающего возможность загрузки текста из файла, отображения его в графическом интерфейсе, а также выбора нужных строк. В результате работы было создано решение, которое соответствует заявленным требованиям и эффективно выполняет все основные функции. Также задачи, поставленные в введении были успешно решены, что подтверждает функциональность и удобство разработанного решения.

Для реализации данного приложения был выбран язык программирования Python, который идеально подходит для поставленной задачи. В качестве графической библиотеки использовалась Tkinter, что позволило создать простое, но эффективное графическое окружение, не перегружая интерфейс лишними элементами и обеспечив легкость в использовании. Tkinter предоставляет все необходимые компоненты для реализации таких функций, как создание окон, кнопок, меток, списков и других элементов управления, которые значительно упрощают процесс создания приложений.

Одним из основных функциональных элементов приложения является возможность загрузки текстовых данных из файла. Загрузка данных в приложение позволяет пользователю работать с любыми текстовыми файлами, содержащими необходимую информацию. Важно отметить, что при разработке была учтена необходимость обработки возможных ошибок, таких как отсутствие доступа к файлу или его повреждение, что помогает улучшить стабильность и надежность программы.

# **Список источников**

1. ["ГОСТ Р 59853-2021 Информационные технологии (ИТ). Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения" (Источник: ИСС "ТЕХЭКСПЕРТ")](http://192.168.11.90:8080/docs/d?nd=1200181819&mark=0H86H5P36F7AA619U8HK83RQVUMH00000041T70CNV39NQQDR084VBDD)
2. ["Об информации, информационных технологиях и о защите информации (с изменениями на 23 ноября 2024 года) (редакция, действующая с 1 января 2025 года)" (Источник: ИСС "ТЕХЭКСПЕРТ")](kodeks://link/d?nd=901990051&mark=000000000000000000000000000000000000000000000000007D60K4)
3. Алгоритмы: введение в разработку и анализ / Левитин А. В.

: Пер. с англ. – М. : Издательский дои «Вильямс», 2006. – 576 c.

1. Алгоритмы и структуры данных: учебное пособие / О. Б. Фофанов; Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2014. – 126с.

# **Приложение**

