ДИСЦИПЛИНА **Конфигурационное управление**

(полное наименование дисциплины без сокращений)

ИНСТИТУТ **Информационных технологий**

КАФЕДРА **Корпоративных информационных систем**

(полное наименование кафедры)

ВИД УЧЕБНОГО МАТЕРИАЛА **Задание для текущего контроля**

(в соответствии с пп.1-11)

ПРЕПОДАВАТЕЛЬ **П.Н. Советов**

(фамилия, имя, отчество)

СЕМЕСТР **3 семестр (осенний) 2025/2026 учебного года**

(указать семестр обучения, учебный год)

# Конфигурационное управление

## Сборник практических работ № 2

ИКБО-40-24

РТУ МИРЭА – 2025

## Оглавление

## О практических работах

Практические работы (ПР) состоят из нескольких этапов. ПР выполняются очно и защита каждого этапа происходит на семинарских занятиях. Этапы работы над ПР сохраняются в публично доступном git-репозитории. Каждый этап разработки ПР должен быть отражен в истории коммитов с детальными сообщениями. Студент самостоятельно выбирает язык реализации.

Документация по ПР оформляется в виде readme.md, который содержит:

1. Общее описание.
2. Описание всех функций и настроек.
3. Описание команд для сборки проекта и запуска тестов.
4. Примеры использования.

Список публичных git-сервисов для репозиториев ПР:

1. github.com
2. gitea.com
3. gitlab.com
4. gitflic.ru
5. hub.mos.ru
6. gitverse.ru
7. gitee.com

## Вариант №1

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров являются опции командной строки.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Версия пакета.
   - Максимальная глубина анализа зависимостей.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Python (pip).
2. Информацию необходимо получить для заданной пользователем версии пакета.
3. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
4. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.

5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS с рекурсией.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран порядка загрузки зависимостей для заданного пакета. Сравнить результаты с реальным менеджером пакетов. Если есть расхождения в результатах, объяснить их наличие.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм PlantUML.
2. Вывести на экран изображение графа.

3. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
4. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №2

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров являются опции командной строки.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Версия пакета.
   - Имя сгенерированного файла с изображением графа.
   - Максимальная глубина анализа зависимостей.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Ubuntu (apt).
2. Информацию необходимо получить для заданной пользователем версии пакета.

3. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
4. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом BFS с рекурсией.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
4. Корректно обработать случаи наличия циклических зависимостей.
5. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран порядка загрузки зависимостей для заданного пакета. Сравнить результаты с реальным менеджером пакетов. Если есть расхождения в результатах, объяснить их наличие.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Graphviz.
2. Сохранить изображение графа в файле формата SVG.
3. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
4. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №3

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата XML.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Режим вывода зависимостей в формате ASCII-дерева.
   - Максимальная глубина анализа зависимостей.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Alpine Linux (apk).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS с рекурсией.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран порядка загрузки зависимостей для заданного пакета. Сравнить результаты с реальным менеджером пакетов. Если есть расхождения в результатах, объяснить их наличие.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Graphviz.
2. Вывести на экран изображение графа.
3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.

4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №4

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата TOML.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Версия пакета.
   - Режим вывода зависимостей в формате ASCII-дерева.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Java (Maven).
2. Информацию необходимо получить для заданной пользователем версии пакета.
3. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.

4. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS с рекурсией.
2. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран порядка загрузки зависимостей для заданного пакета. Сравнить результаты с реальным менеджером пакетов. Если есть расхождения в результатах, объяснить их наличие.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм D2.
2. Вывести на экран описание графа на языке диаграмм.
3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.
4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №5

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров являются опции командной строки.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Версия пакета.
   - Режим вывода зависимостей в формате ASCII-дерева.
   - Максимальная глубина анализа зависимостей.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Rust (Cargo).
2. Информацию необходимо получить для заданной пользователем версии пакета.

3. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
4. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом BFS без рекурсии.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
4. Корректно обработать случаи наличия циклических зависимостей.
5. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран порядка загрузки зависимостей для заданного пакета. Сравнить результаты с реальным менеджером пакетов. Если есть расхождения в результатах, объяснить их наличие.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Mermaid.
2. Вывести на экран описание графа на языке диаграмм.
3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.
4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №6

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата CSV.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Имя сгенерированного файла с изображением графа.
   - Режим вывода зависимостей в формате ASCII-дерева.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Python (pip).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS с рекурсией.
2. Корректно обработать случаи наличия циклических зависимостей.
3. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Mermaid.
2. Сохранить изображение графа в файле формата SVG.
3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.
4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.

5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №7

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров являются опции командной строки.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Версия пакета.
   - Имя сгенерированного файла с изображением графа.
   - Максимальная глубина анализа зависимостей.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Ubuntu (apt).
2. Информацию необходимо получить для заданной пользователем версии пакета.

3. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
4. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS без рекурсии.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
4. Корректно обработать случаи наличия циклических зависимостей.
5. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм D2.
2. Сохранить изображение графа в файле формата SVG.
3. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
4. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №8

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата XML.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Версия пакета.
   - Режим вывода зависимостей в формате ASCII-дерева.
   - Максимальная глубина анализа зависимостей.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Rust (Cargo).
2. Информацию необходимо получить для заданной пользователем версии пакета.
3. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.

4. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS с рекурсией.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран порядка загрузки зависимостей для заданного пакета. Сравнить результаты с реальным менеджером пакетов. Если есть расхождения в результатах, объяснить их наличие.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Mermaid.
2. Вывести на экран изображение графа.
3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.
4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №9

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров являются опции командной строки.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Версия пакета.
   - Имя сгенерированного файла с изображением графа.
   - Режим вывода зависимостей в формате ASCII-дерева.
   - Максимальная глубина анализа зависимостей.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Ubuntu (apt).
2. Информацию необходимо получить для заданной пользователем версии пакета.

3. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
4. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS с рекурсией.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
4. Корректно обработать случаи наличия циклических зависимостей.
5. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран порядка загрузки зависимостей для заданного пакета. Сравнить результаты с реальным менеджером пакетов. Если есть расхождения в результатах, объяснить их наличие.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Graphviz.
2. Сохранить изображение графа в файле формата SVG.
3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.
4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №10

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата XML.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Имя сгенерированного файла с изображением графа.
   - Режим вывода зависимостей в формате ASCII-дерева.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Rust (Cargo).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом BFS с рекурсией.
2. Корректно обработать случаи наличия циклических зависимостей.
3. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Graphviz.
2. Сохранить изображение графа в файле формата PNG.
3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.
4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.

5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №11

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров являются опции командной строки.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Версия пакета.
   - Имя сгенерированного файла с изображением графа.
   - Режим вывода зависимостей в формате ASCII-дерева.
   - Максимальная глубина анализа зависимостей.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Ubuntu (apt).
2. Информацию необходимо получить для заданной пользователем версии пакета.

3. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
4. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом BFS с рекурсией.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
4. Корректно обработать случаи наличия циклических зависимостей.
5. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран порядка загрузки зависимостей для заданного пакета. Сравнить результаты с реальным менеджером пакетов. Если есть расхождения в результатах, объяснить их наличие.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм D2.
2. Сохранить изображение графа в файле формата SVG.
3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.
4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №12

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров являются опции командной строки.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Имя сгенерированного файла с изображением графа.
   - Режим вывода зависимостей в формате ASCII-дерева.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Java (Maven).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.

4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом BFS без рекурсии.
2. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм PlantUML.
2. Сохранить изображение графа в файле формата PNG.

3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.
4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №13

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата TOML.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Имя сгенерированного файла с изображением графа.
   - Режим вывода зависимостей в формате ASCII-дерева.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов .NET (NuGet).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.

4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS с рекурсией.
2. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран порядка загрузки зависимостей для заданного пакета. Сравнить результаты с реальным менеджером пакетов. Если есть расхождения в результатах, объяснить их наличие.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Mermaid.
2. Сохранить изображение графа в файле формата SVG.

3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.
4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №14

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров являются опции командной строки.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Rust (Cargo).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS без рекурсии.
2. Корректно обработать случаи наличия циклических зависимостей.
3. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм D2.
2. Вывести на экран изображение графа.
3. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
4. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.

5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №15

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров являются опции командной строки.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Имя сгенерированного файла с изображением графа.
   - Максимальная глубина анализа зависимостей.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Alpine Linux (apk).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом BFS без рекурсии.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран порядка загрузки зависимостей для заданного пакета. Сравнить результаты с реальным менеджером пакетов. Если есть расхождения в результатах, объяснить их наличие.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Graphviz.
2. Сохранить изображение графа в файле формата PNG.
3. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.

4. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №16

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров являются опции командной строки.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Версия пакета.
   - Максимальная глубина анализа зависимостей.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Python (pip).
2. Информацию необходимо получить для заданной пользователем версии пакета.
3. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.

4. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS без рекурсии.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
4. Корректно обработать случаи наличия циклических зависимостей.
5. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм D2.
2. Вывести на экран описание графа на языке диаграмм.
3. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
4. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №17

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров являются опции командной строки.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Имя сгенерированного файла с изображением графа.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Alpine Linux (apk).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS с рекурсией.
2. Корректно обработать случаи наличия циклических зависимостей.
3. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран порядка загрузки зависимостей для заданного пакета. Сравнить результаты с реальным менеджером пакетов. Если есть расхождения в результатах, объяснить их наличие.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Mermaid.
2. Сохранить изображение графа в файле формата SVG.
3. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
4. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.

5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №18

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата TOML.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Версия пакета.
   - Имя сгенерированного файла с изображением графа.
   - Максимальная глубина анализа зависимостей.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Rust (Cargo).
2. Информацию необходимо получить для заданной пользователем версии пакета.

3. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
4. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом BFS с рекурсией.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
4. Корректно обработать случаи наличия циклических зависимостей.
5. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Mermaid.
2. Сохранить изображение графа в файле формата SVG.
3. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
4. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №19

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата TOML.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Версия пакета.
   - Режим вывода зависимостей в формате ASCII-дерева.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Rust (Cargo).
2. Информацию необходимо получить для заданной пользователем версии пакета.
3. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.

4. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS с рекурсией.
2. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Graphviz.

2. Вывести на экран изображение графа.
3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.
4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №20

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров являются опции командной строки.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Версия пакета.
   - Имя сгенерированного файла с изображением графа.
   - Максимальная глубина анализа зависимостей.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов JavaScript (npm).
2. Информацию необходимо получить для заданной пользователем версии пакета.
3. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.

4. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом BFS с рекурсией.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран порядка загрузки зависимостей для заданного пакета. Сравнить результаты с реальным менеджером пакетов. Если есть расхождения в результатах, объяснить их наличие.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм D2.
2. Сохранить изображение графа в файле формата PNG.
3. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
4. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №21

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров являются опции командной строки.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Версия пакета.
   - Имя сгенерированного файла с изображением графа.
   - Максимальная глубина анализа зависимостей.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Java (Maven).
2. Информацию необходимо получить для заданной пользователем версии пакета.
3. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.

4. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом BFS без рекурсии.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Graphviz.

2. Сохранить изображение графа в файле формата SVG.
3. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
4. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №22

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата CSV.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Режим вывода зависимостей в формате ASCII-дерева.
   - Максимальная глубина анализа зависимостей.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов JavaScript (npm).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом BFS с рекурсией.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Mermaid.
2. Вывести на экран описание графа на языке диаграмм.
3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.

4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №23

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата TOML.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Имя сгенерированного файла с изображением графа.
   - Режим вывода зависимостей в формате ASCII-дерева.
   - Максимальная глубина анализа зависимостей.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Rust (Cargo).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.

4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS без рекурсии.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Mermaid.
2. Сохранить изображение графа в файле формата SVG.

3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.
4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №24

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата INI.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Версия пакета.
   - Имя сгенерированного файла с изображением графа.
   - Режим вывода зависимостей в формате ASCII-дерева.
   - Максимальная глубина анализа зависимостей.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Ubuntu (apt).
2. Информацию необходимо получить для заданной пользователем версии пакета.

3. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
4. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом BFS с рекурсией.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
4. Корректно обработать случаи наличия циклических зависимостей.
5. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм PlantUML.
2. Сохранить изображение графа в файле формата SVG.
3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.
4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №25

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров являются опции командной строки.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Имя сгенерированного файла с изображением графа.
   - Режим вывода зависимостей в формате ASCII-дерева.
   - Максимальная глубина анализа зависимостей.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов JavaScript (npm).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.

4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS с рекурсией.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Mermaid.
2. Сохранить изображение графа в файле формата PNG.

3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.
4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №26

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата CSV.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Имя сгенерированного файла с изображением графа.
   - Максимальная глубина анализа зависимостей.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Java (Maven).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS без рекурсии.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран порядка загрузки зависимостей для заданного пакета. Сравнить результаты с реальным менеджером пакетов. Если есть расхождения в результатах, объяснить их наличие.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм PlantUML.
2. Сохранить изображение графа в файле формата SVG.
3. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.

4. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №27

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров являются опции командной строки.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Режим вывода зависимостей в формате ASCII-дерева.
   - Максимальная глубина анализа зависимостей.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Python (pip).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом BFS без рекурсии.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран порядка загрузки зависимостей для заданного пакета. Сравнить результаты с реальным менеджером пакетов. Если есть расхождения в результатах, объяснить их наличие.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм D2.
2. Вывести на экран изображение графа.
3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.

4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №28

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата TOML.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Версия пакета.
   - Максимальная глубина анализа зависимостей.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Rust (Cargo).
2. Информацию необходимо получить для заданной пользователем версии пакета.
3. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
4. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.

5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS с рекурсией.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран порядка загрузки зависимостей для заданного пакета. Сравнить результаты с реальным менеджером пакетов. Если есть расхождения в результатах, объяснить их наличие.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм PlantUML.
2. Вывести на экран изображение графа.

3. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
4. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №29

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата YAML.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Версия пакета.
   - Режим вывода зависимостей в формате ASCII-дерева.
   - Максимальная глубина анализа зависимостей.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов .NET (NuGet).
2. Информацию необходимо получить для заданной пользователем версии пакета.
3. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.

4. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS без рекурсии.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Graphviz.

2. Вывести на экран описание графа на языке диаграмм.
3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.
4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №30

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров являются опции командной строки.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Режим вывода зависимостей в формате ASCII-дерева.
   - Максимальная глубина анализа зависимостей.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Python (pip).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.

4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом BFS без рекурсии.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
4. Корректно обработать случаи наличия циклических зависимостей.
5. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Mermaid.

2. Вывести на экран описание графа на языке диаграмм.
3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.
4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №31

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата XML.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов .NET (NuGet).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом BFS с рекурсией.
2. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Mermaid.
2. Вывести на экран изображение графа.
3. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.

4. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №32

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата YAML.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Максимальная глубина анализа зависимостей.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов .NET (NuGet).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом BFS с рекурсией.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Mermaid.
2. Вывести на экран описание графа на языке диаграмм.
3. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.

4. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №33

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров являются опции командной строки.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Версия пакета.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов JavaScript (npm).
2. Информацию необходимо получить для заданной пользователем версии пакета.
3. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
4. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.

5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом BFS без рекурсии.
2. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран порядка загрузки зависимостей для заданного пакета. Сравнить результаты с реальным менеджером пакетов. Если есть расхождения в результатах, объяснить их наличие.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм PlantUML.
2. Вывести на экран изображение графа.

3. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
4. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №34

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата TOML.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Режим вывода зависимостей в формате ASCII-дерева.
   - Максимальная глубина анализа зависимостей.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Alpine Linux (apk).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.

4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS без рекурсии.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
4. Корректно обработать случаи наличия циклических зависимостей.
5. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран порядка загрузки зависимостей для заданного пакета. Сравнить результаты с реальным менеджером пакетов. Если есть расхождения в результатах, объяснить их наличие.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм D2.
2. Вывести на экран описание графа на языке диаграмм.
3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.
4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №35

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата TOML.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Версия пакета.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Rust (Cargo).
2. Информацию необходимо получить для заданной пользователем версии пакета.
3. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
4. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.

5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом BFS с рекурсией.
2. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран порядка загрузки зависимостей для заданного пакета. Сравнить результаты с реальным менеджером пакетов. Если есть расхождения в результатах, объяснить их наличие.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Mermaid.
2. Вывести на экран описание графа на языке диаграмм.

3. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
4. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №36

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров являются опции командной строки.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Режим вывода зависимостей в формате ASCII-дерева.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Alpine Linux (apk).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS с рекурсией.
2. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм PlantUML.
2. Вывести на экран изображение графа.
3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.

4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №37

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров являются опции командной строки.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Режим вывода зависимостей в формате ASCII-дерева.
   - Максимальная глубина анализа зависимостей.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Rust (Cargo).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS с рекурсией.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм D2.
2. Вывести на экран изображение графа.
3. Если задан соответствующий параметр, вывести на экран зависимости в виде ASCII-дерева.

4. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
5. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №38

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата CSV.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Alpine Linux (apk).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом BFS с рекурсией.
2. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
3. Корректно обработать случаи наличия циклических зависимостей.
4. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Mermaid.
2. Вывести на экран описание графа на языке диаграмм.
3. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.

4. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №39

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата TOML.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
   - Версия пакета.
   - Максимальная глубина анализа зависимостей.
   - Подстрока для фильтрации пакетов.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Ubuntu (apt).
2. Информацию необходимо получить для заданной пользователем версии пакета.
3. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.

4. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS без рекурсии.
2. Проводить анализ с учетом максимальной глубины, заданной пользователем.
3. Не учитывать при анализе пакеты, имя которых содержит заданную пользователем подстроку.
4. Корректно обработать случаи наличия циклических зависимостей.
5. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
6. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран обратных зависимостей для заданного пакета. Это те пакеты, которые зависят от данного пакета. Использовать алгоритм обхода из предыдущего этапа.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Graphviz.
2. Вывести на экран описание графа на языке диаграмм.
3. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
4. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

## Вариант №40

Разработать инструмент визуализации графа зависимостей для менеджера пакетов. Готовые средства (менеджеры пакетов, библиотеки) для получения зависимостей использовать нельзя.

**Этап 1. Минимальный прототип с конфигурацией**

Цель: создать минимальное CLI-приложение и сделать его настраиваемым.

Требования:

1. Источником настраиваемых пользователем параметров является конфигурационный файл формата INI.
2. К настраиваемым параметрам относятся:
   - Имя анализируемого пакета.
   - URL-адрес репозитория или путь к файлу тестового репозитория.
   - Режим работы с тестовым репозиторием.
3. (только для этого этапа) При запуске приложения вывести все параметры, настраиваемые пользователем, в формате ключ-значение.
4. Реализовать и продемонстрировать обработку ошибок для всех параметров.
5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 2. Сбор данных**

Цель: реализовать основную логику получения данных о зависимостях для их дальнейшего анализа и визуализации. Запрещено пользоваться менеджерами пакетов и сторонними библиотеками для получения информации о зависимостях пакетов.

Требования:

1. Использовать формат пакетов Ubuntu (apt).
2. Извлечь информацию о прямых зависимостях заданного пользователем пакета, используя URL-адрес репозитория.
3. (только для этого этапа) Вывести на экран все прямые зависимости заданного пользователем пакета.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 3. Основные операции**

Цель: построить граф зависимостей (с учетом транзитивности) и выполнить основные операции над ним.

Требования:

1. Получение графа зависимостей реализовать алгоритмом DFS с рекурсией.
2. Корректно обработать случаи наличия циклических зависимостей.
3. Поддержать режим тестирования. Вместо URL реального репозитория, дать возможность пользователю указать путь к файлу описания графа репозитория, где пакеты называются большими латинскими буквами. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
4. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 4. Дополнительные операции**

Цель: выполнить дополнительные операции над графом зависимостей.

Требования:

1. (только для этого этапа) Поддержать режим вывода на экран порядка загрузки зависимостей для заданного пакета. Сравнить результаты с реальным менеджером пакетов. Если есть расхождения в результатах, объяснить их наличие.
2. Продемонстрировать функциональность этого этапа на различных случаях работы с тестовым репозиторием.
3. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.

**Этап 5. Визуализация**

Цель: получить графическое представление графа зависимостей.

Требования:

1. Сформировать текстовое представление графа зависимостей на языке диаграмм Mermaid.
2. Вывести на экран описание графа на языке диаграмм.
3. Продемонстрировать примеры визуализации зависимостей для трех различных пакетов.
4. Сравнить результаты с выводом штатных инструментов визуализации для выбранного менеджера пакетов. Если есть расхождения в результатах, объяснить их наличие.

5. Результат выполнения этапа сохранить в репозиторий стандартно оформленным коммитом.