**1. Подход к расширяемости**

1. **Добавление новых связей**
   * Новые связи можно добавлять в JSON-файл, следуя существующему формату.
   * Для добавления потребуется лишь описание связи в виде комбинации родственных уровней, таких как LIK(M)/ELDRE(W).
   * Программа должна быть спроектирована так, чтобы при обновлении JSON-файла автоматически учитывать новые связи без изменения логики обработки.
2. **Редактирование существующих связей**
   * Можно изменить определения родственных отношений, уточняя или добавляя новые комбинации.
   * Например, если потребуется уточнение термина "Сноха", можно обновить существующую запись без влияния на другие части системы.
3. **Гибкая система поиска**
   * Поиск связей строится на основе анализа цепочек родства, описанных в JSON.
   * Программа должна уметь интерпретировать новые связи по заданным шаблонам.

**2. Расшифровка формата JSON и интерпретация**

Каждый тип родства в JSON представлен через комбинации сокращений:

* **Типы людей:**
  + LIK(M) — супруг (LIK = партнер, M = мужской пол).
  + LIK(W) — супруга (W = женский пол).
  + UNG(M) — сын (UNG = младшее поколение, M = мужской пол).
  + ELDRE(W) — мать (ELDRE = старшее поколение, W = женский пол).
* **Комбинации:**
  + / — обозначает родственную связь. Например, LIK(W)/ELDRE(M) означает «тесть» (мужчина из старшего поколения супруги).
  + && — альтернативные пути, например, «внук» может быть как по отцовской, так и по материнской линии.

Примеры интерпретации:

* "Сноха": "UNG(M)/LIK(W)" — жена сына (невестка).
* "Тёща": "LIK(W)/ELDRE(W)" — мать супруги.
* "Дедушка": "ELDRE(W)/ELDRE(M)&&ELDRE(M)/ELDRE(M)" — возможны два варианта дедушки (по линии отца и по линии матери).

**3. Добавление новых родственных связей**

Добавление новых типов родства требует:

1. Определения шаблона для новой связи.
   * Например, если необходимо добавить понятие "Кузен" (двоюродный брат), оно может быть представлено как:

"Кузен": "ELDRE(M)/UNG(M)/UNG(M)"

* + Здесь:
    - ELDRE(M) — отец,
    - UNG(M) — его ребёнок,
    - UNG(M) — сын его брата (двоюродный брат).

1. Обновления JSON-файла с новыми значениями.
2. Автоматического распознавания программы новых связей без необходимости обновления кода.

**Пример новых записей:**

{

"Кузен": "ELDRE(M)/UNG(M)/UNG(M)",

"Троюродный брат": "ELDRE(M)/ELDRE(M)/UNG(M)/UNG(M)",

"Пасынок": "LIK(W)/UNG(M)",

"Падчерица": "LIK(W)/UNG(W)"

}

**4. Масштабируемость системы**

Для того чтобы обеспечить удобное расширение и поддержку системы, важно учитывать:

1. **Разделение логики обработки:**
   * JSON-файл должен хранить только правила и структуру связей, а интерпретация должна выполняться отдельным модулем.
   * Это позволяет легко обновлять данные и расширять функциональность.
2. **Гибкость обработки данных:**
   * Использование рекурсивного алгоритма для поиска связей по JSON.
   * Возможность добавления пользовательских правил через интерфейс.
3. **Локализация терминов:**
   * Связи могут быть представлены на разных языках, например, добавить вариант "Brother-in-law" для "Шурин".
   * Для этого можно организовать словарь терминов, например:

{

"Шурин": { "en": "Brother-in-law"}

}

**Вывод**

Использование JSON для хранения родственных связей обеспечивает:

* **Гибкость и лёгкость обновления**, позволяя без изменения кода добавлять новые виды отношений.
* **Простоту масштабирования**, так как система адаптируется к новым данным автоматически.
* **Поддержку многозначных связей**, таких как альтернативные пути родства (&&).

Такой подход делает систему удобной для дальнейшего развития и локализации.