



# Практическое задание

## Тема

Узлы (Nodes) в Mermaid: идентификаторы, текст, формы, повторное использование

## Цель

Закрепить понимание:

- различия между **внутренним ID** и **отображаемым текстом**,
- корректного объявления узлов в `flowchart`,
- семантического использования форм,
- повторного использования одного и того же узла,
- аккуратной работы с многострочным текстом,
- соблюдения структурной дисциплины (тип диаграммы, порядок, читаемость).

Задание направлено не на “красоту”, а на **структурную корректность и логическую ясность**.

## Задание (шаги)

Создайте диаграмму `flowchart TD`, которая моделирует **упрощённый процесс обработки пользовательского запроса в системе**.

### Шаг 1. Объявите тип диаграммы

Первая строка — строго:

```
flowchart TD
```

Никаких пробелов, комментариев или текста выше.

## Шаг 2. Создайте узлы со следующей логикой

В диаграмме должны присутствовать:

1. Начало процесса
2. Получение запроса
3. Проверка корректности запроса (условие)
4. Два варианта исхода проверки:
  - Запрос некорректен → возврат ошибки
  - Запрос корректен → обработка запроса
5. Завершение процесса

## Шаг 3. Используйте разные формы узлов

Обязательно:

- Начало/конец — скруглённая форма
- Проверка — ромб
- Обычные действия — прямоугольники

Формы должны использоваться **семантически**, а не случайно.

## Шаг 4. Разделите ID и отображаемый текст

- ID должны быть короткими латинскими именами ( `start` , `validate` , `process` , `error` и т.п.)
- Отображаемый текст должен быть человекочитаемым
- Не используйте длинные предложения в качестве ID

## Шаг 5. Используйте один узел повторно

После обработки запроса поток должен возвращаться к узлу “Получение запроса” (имитация цикла).

Важно:

Используйте **тот же ID**, а не создавайте новый узел с похожим текстом.

## Шаг 6. Добавьте комментарии

Добавьте минимум 3 комментария `%%`, которые объясняют:

- почему выбран определённый тип формы,
- где используется повторный ID,
- где реализовано условное ветвление.

Комментарии должны быть строковыми (отдельной строкой).

## Подсказки по ключевым частям

### 1. Узел создаётся при первом упоминании ID

Если вы написали:

```
A --> B
```

то узлы `A` и `B` уже существуют.

### 2. Формы задаются скобками

- `A["текст"]` — прямоугольник
- `A("текст")` — скруглённый
- `A{"текст"}` — ромб

Следите, чтобы скобки были корректно закрыты.

### 3. Подписи на стрелках (для условия)

В `flowchart` можно писать:

```
validate -->|Да| process  
validate -->|Нет| error
```

Текст между `| |` — это подпись связи.

## 4. Многострочный текст

Если хотите перенос строки внутри узла, используйте:

```
<br/>
```

Не используйте `\n`.

## 5. Не допускайте

- пробелов в ID
- одинаковых ID для разных сущностей
- смещения `sequenceDiagram` синтаксиса

## Что проверить перед отправкой (чек-лист)

Перед тем как считать задание выполненным, проверьте:

- ☐ Первая строка — `flowchart TD`
- ☐ Все ID уникальны
- ☐ Нет пробелов в ID
- ☐ Формы узлов соответствуют смыслу
- ☐ Ромб используется только для условия
- ☐ Один узел действительно повторно используется (не дубликат)
- ☐ Все скобки корректно закрыты
- ☐ Нет inline-комментариев (только `%%` отдельной строкой)
- ☐ Диаграмма компилируется без ошибок
- ☐ Текст читается за 2–3 секунды

## Советы по улучшению работы

1. **Сначала нарисуйте логику на бумаге.**  
Только потом переводите в Mermaid.
2. **Сначала напишите ID без текста.**  
Проверьте структуру. Затем добавьте подписи.

3. **Не перегружайте текст.**

Диаграмма — это навигация, а не документация.

4. **Проверьте повторное использование узла.**

Убедитесь, что вы действительно ссылаетесь на тот же ID.

5. **Смотрите на код как на программу.**

Mermaid — это формальный язык. Если вы мыслите структурой, а не “картинкой”, ошибок почти не будет.