Правила стиля кода

**1. Организация заголовочных файлов**

* **Соответствие**: Каждый .c-файл должен иметь соответствующий .h-файл (кроме тестов и main()).
* **Защита от повторного включения**:

#ifndef PROJECT\_MODULE\_HEADER\_H\_

#define PROJECT\_MODULE\_HEADER\_H\_

*// Код...*

#endif *// PROJECT\_MODULE\_HEADER\_H\_*

* **Порядок включения** (с пустыми строками между группами):
  1. Соответствующий .h
  2. Системные библиотеки (#include <stdio.h>)
  3. Сторонние библиотеки
  4. Заголовки проекта

Между группами - пустая строка

**2. Область видимости и переменные**

* **Локальные переменные**:
  + Объявлять в максимально узкой области видимости.
  + Инициализировать при объявлении:

int count = 0;

**3. Функции**

* **Принцип**: Одна функция = одна ответственность (одно четкое действие
* **Размер**: Не более 40 строк.
* **Выход**: Одна точка выхода (один return).

**4. Особенности языка C**

* **Инкремент**: ++i вместо i++, если вам не нужна постфиксная семантика.
* **Константность**:
  + Использовать const везде, где это имеет смысл.
  + Порядок: const int\* ptr (не int const \*ptr).
* switch: Всегда включать default case.

**5. Правила именования**

| **Объект** | **Стиль** | **Пример** |
| --- | --- | --- |
| Файлы | lowercase\_with\_underscores | network\_utils.c |
| Типы (struct, enum) | UpperCamelCase | typedef struct {…} TreeNode; |
| Переменные | snake\_case | user\_count |
| Макросы | UPPER\_CASE | MAX\_BUFFER\_SIZE |
| Константы | kUpperCamelCase | const int kMaxUsers = 100; |

**Дополнительно**:

* Английские названия (без транслитерации).
* Имена должны быть **понятными** и отражать суть переменной/функции
* Избегать сокращений, кроме общепринятых (idx, tmp, max, min и т. д.).
* Исключение из правил: i, j в циклах.

**6. Комментарии**

* **Комментарии функций** перед объявлением (**Doxygen-стиль**):

*/\*\**

*\* @brief Converts string to lowercase.*

*\* @param str Input string (must be null-terminated).*

*\* @return New allocated string in lowercase, or NULL on error.*

*\*/*

char \*str\_to\_lower(const char \*str);

* **Переменные**: Комментарий не заменяет название, лучше понятное название, чем понятный комментарий.
* **Сложные блоки**: Сложные или запутанные блоки кода должны иметь комментарии перед ними (Но лучше потратьте время на переписывание кода, а не на написание комментария).
* **TODO**: с указанием контекста (временное, краткосрочное решение):

*// TODO(dev): Заменить хэш-таблицей*

*// TODO(bug-45): Исправить утечку памяти*

**7. Прочие рекомендации**

1. **Магические числа**: Заменять на именованные константы.

const int kMaxRetries = 3; *// Вместо числа 3 в коде.*

1. **Проверка указателей** на NULL перед использованием:

if (ptr == NULL) { */\* Обработка ошибки \*/* }

1. **Указатели vs массивы**:
   * Для массивов: char[].
   * Для одиночных указателей: char\*.
   * **Размеры**: Предпочитать size\_t для размеров