# Our Coding Guidelines

Version 0.9.0

# Введение

Документ перечисляет правила и соглашения, строгое следование которым позволяет улучшить читаемость кода.

Данные правила должны применяться в любом C/C++ коде студентов и преподавателей-семинаристов БК №536.

Документ может изменяться общим решением преподавателей-программистов.

## 1. Форматирование

Вложенный код (тела циклов, функций, условных конструкций, объявлений структур) отбивается 4 пробелами (предпочтительно) или одним табом.

*Почти всегда* **запрещено** писать несколько выражений на одной строке.

Рекомендуемая длина строки (с учетом отступов) — меньше 80 символов.

**Запрещено** создавать строки длиной больше 119 символов (с учетом отступов)!

Операторы и операнды **обязательно** отделяются пробелами (исключение — унарные операторы).

```
// Good
char* findCharPosition(const char* str, char target) {
    while (*str) {
        if (*str == target) {
            return str;
        }
        ++str;
    }
    return NULL;
}

// Bad
char* findCharPosition(const char* str, char target) {
while (*str) {
if (*str == target) {
    return str;
}
++str;
}
return NULL;
}

// Bad (evil)
char* findCharPosition(const char* str, char target) {
int index = 0;
while (*str)
    {
    if (*str == target) {
return str;
    }
    ++str;
}
return NULL;
}
```

## 2. Нейминг

### 2.1.Общие требования

Имена функций, пользовательских типов, констант и переменных должны раскрывать свое назначение, желательно в емкой форме. Избегайте как коротких и немногословных названий, так и слишком длинных имен переменных. Избегайте использования чисел в именах для обозначения копий, перегрузок функций, «похожести» объектов.

Короткие имена допускаются для индексов, но только те, что принято использовать в качестве индексов (i, j, k…), а также для объектов с коротким

временем жизни и очевидным назначением (n в качестве длины массива, но size предпочтительнее).

Для имен функций старайтесь использовать глаголы.

```
// Good
char*   studentName;
size_t  studentId;

void strip(char* source);
char* getStripped(const char* source);

struct Student;

// Bad
char*   n;
size_t  i;

void func(char* s);
char* getfunc(const char* s1);

struct A;

// Also bad
char*   theNameOfStudentFromInput;

void eraseAllSpacesFromBeginAndEnd(char* source);

void toupper(char* source);
char* toupper1(const char* cs);
```

## 2.2.Нотация

Нотация — способ записи имен объектов.

| | |
|---|---|
| Переменная | camelCase (lowerCamelCase) |
| Функция | camelCase (lowerCamelCase) |
| Глобальная константа | PascalCase (UpperCamelCase) |
| Глобальная переменная | g_PascalCase (на обсуждении) |
| Поле структуры/объединения | camelCase |
| Константа в enum | PascalCase (UpperCamelCase) |
| Имя пользовательского типа | PascalCase (UpperCamelCase) |

| | |
|---|---|
| Синоним типа | PascalCase (UpperCamelCase) |
| Макросы | UPPER_CASE |

## 3. Скобки

### 3.1. Фигурные скобки

**Обязательно** оборачивайте фигурными скобками тела циклов и условий (даже если тело состоит из одного стейтмента).

Мотивация: помимо улучшения читаемости, упрощает отладку и дальнейшую поддержку кода. Вполне вероятно, что в будущем вам придется добавить в тело еще одну строку, по итогу скобки все равно будут проставлены.

(следующий абзац — на обсуждении, но вероятно он войдет в стандарт)

Левая фигурная скобка находится на той же строчке, что и оператор.

Исключение: если в условной конструкции длинное условие и закрывающая круглая скобка находится на другой строке.

```
// Good
for (int i = 0; i < n; ++i) {
    printf("%d\n", i * i);
}

if (condition) {
    approximate();
}

if (condition && otherCondition
    && oneMoreCondition && longLongLongCondition)
{
    reproximate();
}

// Bad
for (int i = 0; i < n; ++i)
    printf("%d\n", i * i);

if (condition)
{
    approximate();
}

if (condition && otherCondition
    && oneMoreCondition && longLongLongCondition) {
    reproximate();
}
```

### 3.2. Круглые скобки

Выражения в условиях следует группировать с помощью круглых скобок.

Объекты внутри скобки не отбиваются от скобки пробелом: (1 + 1).

## 4. Указатели

**Запрещено** объявлять несколько указателей на одной строке.

Знак указателя находится рядом с типом.

```
// Good
int* x;
int* y;

// Bad
int *x, *y;
```

5. Разное

Набор общих правил и рекомендаций.

- warning’и компилятора недопустимы;
- документируйте свой код;
- удаляйте неиспользуемые хедеры и объекты из исходного кода;
- выполнение операций в if нежелательно;
- предпочитайте явные преобразования типов неявным;
- используйте префиксный инкремент/декремент (если задача не требует использования постфиксного инкремента/декремента);
- используйте typedef при декларации структуры в языке C;
- при разрыве строки операторы переносятся на следующую строку;
- если аргумент функции можно пометить const, он должен быть const;
- можно отделять логические блоки пустой строкой.