Оформление кода

Отступы

1. В качестве отступов используйте пробел. Табуляция не используйте.
2. Размер отступа 4 пробела.
3. Используйте пустые строки для логической группировки операторов, где это возможно
4. Всегда используйте одну пустую строку в качестве разделителя
5. Всегда используйте один пробел перед фигурной скобкой

```
// Wrong
if(foo){
}

// Correct
if (foo) {
}
```

6. Всегда ставьте один пробел после `*` или `&`, если они стоят перед описанием типов. Но никогда не ставьте пробелы после `*` или `&` и именем переменной
7. Бинарные операции отделяются пробелами с 2-х строн
8. После преобразования типов не ставьте пробелов

Именование

1. Структуры, перечисления, объединения именуются `CamelCase` стиле, без использования `_`

```
struct Camel {

};
typedef struct Camel Camel;
```

2. Переменные и функции именуются в `under_score` стиле

```
int simple_output_format_id = 0;
```

Определение переменных

1. Объявляйте по одной переменной в строке
2. Избегайте, если это возможно, коротких и запутанных названий переменных
3. Односимвольные имена переменных подходят только для итераторов циклов, небольшого локального контекста и временных переменных. В остальных случаях имя переменной должно отражать ее назначение
4. Заводите переменные только по мере необходимости
5. Избегайте аббревиатур

Скобки

1. Возьмите за основу расстановку открывающих фигурных скобок

на одной строке с выражением, которому они предшествуют

```
// Wrong
if (codec)
{
}

// Correct
if (codec) {
}
```

2. Используйте фигурные скобки в условиях, если тело условия в размере превышает одну линию, или тело условия достаточное сложное и выделение скобками действительно необходимо

```
// Correct
if (address == 0) {
  return false;
}

for (i = 0; i < 10; ++i) {
  fprintf(stdout, "%i", i);
}

// Correct
if (address != 0)
  return true;
```

```
for (i = 0; i < 10; ++i)
  fprintf(stdout, "%i", i);
```

3. Используйте фигурные скобки, когда тела ветвлений `if-then-else` занимают несколько строчек

```
// Wrong
if (address == 0)
  return false;
else {
  fprintf(stdout, "%s", address);
  ++it;
}

// Correct
if (address == 0) {
  return false;
} else {
  fprintf(stdout, "%s", address);
  ++it;
}

// Wrong
if (a)
  if (b)
    ...
  else
    ...
```

```
// Wrong
if (a) {
  if (b)
    ...
  else
    ...
}
```

4. Используйте фигурные скобки для обозначения пустого тела условия

```
// Wrong
while (a);

// Correct
while (a) {}
```

5. Используйте круглые скобки для группировки выражений

```
// Wrong
if (a && b || c)

// Correct
if ((a && b) || c)

// Wrong
a + b & c
```

```
// Correct
(a + b) & c
```

Использование конструкции `switch`

1. Операторы `case` должны быть в одном столбце со `switch`
2. Каждый оператор `case` должен иметь закрывающий `break` (или `return`) или комментарий, котрой предполагает намеренное отсутсвие `break` или `return`

```
switch (myEnum) {
case Value1:
  do_somthing();
  break;
case Value2:
  do_somthing_else();
  // continue
default:
  default_handling();
  break;
}
```

Разрыв строк

1. Длина строки кода не должна превышать 100 символов. Если надо - используйте разрыв строки
2. Запятые помещаются в конец разорванной линии; операторы

помещаются в начало новой строки. В зависимости от используемой вами `IDE`, оператор на конце разорванной строки можно проглядеть

```
// Correct
if (longExpression
    + otherLongExpression
    + otherOtherLongExpression) {
}

// Wrong
if (longExpression +
    otherLongExpression +
    otherOtherLongExpression) {
}
```