**II.1 - Название (Название переменных)**

1. Название structure (структур) должно начинаться с префикса s\_.

2. Название typedef (создания типов) должны начинаться с префикса t\_.

3. Название union (объединения) должны начинаться с префикса u\_.

4. Название enum (перечисления) должны начинаться с префикса e\_.

5. Название глобальных переменных должны начинаться с префикса g\_.

6. Название переменных и функций должны содержать строчные (мал.) буквы, цифры и симв. ’\\_’.

7. Название файлов и каталогов должны содержать только строчные (мал.) буквы, цифры и ’\\_’.

8. Файл должен скомпилироваться.

9. Запрещены символы, которых нет в стандартной ASCII таблице.

**Рекомендации / Advice part**

1. Названия объектов (переменных, функций, макросов, типов, файлов и директории) должны быть понятны и запоминающимися. Только ‘counters’ (переменные-счетчики) могут быть названы так, как вам хочется.

2. Допускается использование аббревиатур только в том случае, если названия все равно остаются понятными. Если название содержит в себе более чем одно слово, то слова должны разделяться символом ’\\_’ .

3. Все идентификаторы (функции, макросы, типы, переменные и т. д.) должны быть на английском языке.

4. Любое использование глобальной переменной должно быть оправданным (обоснованным).

**II.2 - Оформление / Formatting**

1. Все ваши файлы должны начинаться с заголовка школы 42 (начиная с первой строки вашего файла). Этот заголовок по умолчанию доступен в редакторах emacs и vim в дампах.

2. В качестве отступа в вашем коде вы должны использовать 4-пробельную табуляцию. Это не тоже самое, что 4 пробела, мы говорим о настоящей табуляции (кнопка TAB).

3. Каждая функция должна содержать не более 25 строк, не считая строк с фигурными скобками этой функции.

4. Максимальная длина строки вместе с комментариями - 80 знаков. Внимание: табы считаются не как знаки, а как количество пробелов, которые они заменяют (1 Tab = 4 пробела).

5. Одна инструкция на строку.

6. Пустая строка должна быть пустой: никаких TAB’ов или пробелов.

7. Строка никогда не должна заканчиваться пробелом или TAB’ом.

8. Вам нужно начинать новую строку после каждой фигурной скобки или конца контрольной структуры.

9. Все запятые и точки с запятой должны иметь после себя пробел, только если это не конец строки.

10. Каждый оператор (бинарныйдвоичный или тернарныйтроичный) или операндаргумент-операции должны быть разделены одним и только одним пробелом.

11. За каждым ключевым словом С должен идти пробел, за исключением ключевых слов для типов (таких как int, char, float, и т.д.), а также sizeof.

12. Каждое объявление переменной должно иметь отступ в том же столбце.

13. Звездочки указателей должны быть вплотную к названиям переменных.

14. Одно объявление переменной на строку.

15. Нельзя делать объявление и инициализацию в одной строке, за исключением глобальных и статических переменных.

16. Объявления переменных должны быть в начале функции и должны быть отделены пустой строкой.

17. Не должно быть пустой строки между объявлениями или реализациями.

18. Многократные назначения строго запрещены. :🚩некор.\_перевод🚩:

19. Вы можете добавить новую строку после инструкции или управляющей структуры, но вам придется добавить отступ в скобках или оператор присваивания. Операторы должны быть в начале строки.

**II.3 - Параметры передаваемые в функцию / Functions parameters**

1. Функция может принимать максимум 4 именованных параметров.

2. Функция, которая ничего не возвращает аргументы, должна начинаться со слова (должна быть явно прототипирована со слова) “void”, в качестве аргумента.

**II.4 - Функции / Functions**

1. Параметры в прототипах функций должны быть названы.

2. Каждая функция должна быть отделена от следующей пустой строкой.

3. Вы не можете объявлять больше 5 переменных в одном блоке.

4. Возвращение функции должно быть между скобками.

Рекомендации / Advice part

1. Идентификаторы ваших функций должны быть выровнены в пределах одного и того же файла. То же самое касается заголовочных файлов.

**II.5 - Определение имени, структура, перечисление и объединение / Typedef, struct, enum and union**

1. Добавляйте табуляцию при объявлении struct(структуры), enum(перечисления) or union(объединения).

2. При объявлении переменной типа (struct, enum или union) добавьте один пробел в типе.

3. Добавляйте табуляцию между двумя параметрами typedef(определителя имени).

4. При объявлении (struct, union или enum) с typedef, применяются все правила. Новое имя typedef должно соотноситься с именем struct / union / enum.

5. Нельзя объявлять struct в .с файле.

**II.6 - Заголовки / Headers**

1. В заголовочных файлах разрешено следующие : включения(#include) заголовка (системные или нет), объявления, определения, прототипы и макросы.

2. Все включения ( .c или .h) должны быть в начале файла.

3. Мы будем защищать заголовки от двойных включений. Если имя файла ft\_foo.h, его визуальный макрос это FT\_FOO\_H.

4. Прототипы функций должны быть только в файлах формата .h

5. Неиспользованное включение заголовков ( .h) недопустимы.

**Рекомендации / Advice part**

1. Все включения заголовка должны быть обоснованы, как в .c файлах, так и в .h .

**II.7 - Макрос и препроцессоры / Macros and Pre-processors**

1. Константы препроцессора (или #define), которые вы создаете, должны использоваться только для связывания литеральных и константных значений.

2. Запрещены все #define, созданные для обхода нормального и / или запутанного кода. Этот момент должен быть проверен человеком.

3. Вы можете использовать макросы, доступные в стандартных библиотеках, только если они разрешены в рамках данного проекта.

4. Многострочные макросы запрещены.

5. В верхнем регистре должны содержаться только имена макросов.

6. После #if, #ifdef или #ifndef вы должны сделать отступ для символов.

**II.8 - Запрещенные штуки ! / Forbidden stuﬀ !**

1. Вам запрещено использовать: / You’re not allowed to use:

\* цикл for

\* цикл do...while

\* оператор switch

\* оператор case

\* оператор goto

2. Запрещены вложенные тернарныетроичные операторы, такие как ‘?’.

3. Запрещено пользоваться массивы переменной длины (динамическими массивами) :🚩некор.\_перевод🚩:

**II.9 - Комментарии / Comments**

1. Вам разрешено комментировать свой код в исходных файлах.

2. Комментарии не могут быть внутри(inside) тела функции.

3. Комментарии должны начинаться и заканчиваться на одной строке. Все промежуточные строки должны быть выравнены и начинаться с ‘\*\*’.

4. Никаких комментариев вида //.

**Рекомендации / Advice part**

1. Ваши комментарии должны быть на английском языке. И они должны быть полезными.

2. Комментарий не может оправдать «ублюдочную» функцию.

**II.10 - Files**

1. Вы не можете включить(#include) .c файл.

2. Вы не можете иметь более 5 определений функций в .c файле.

**II.11 - Makeﬁle**

1. $(NAME), clean, fclean, re и все правила являются обязательными.

2. Если makefile relink'ается, проект будет считаться нефункциональным.(Работа будет не засчитана).

3. В случае мультибинарного проекта, в дополнение к правилам, которые вы видели, у вас должно быть правило, которое компилирует оба бинарных(двоичных) файла, а также определенное правило для каждого скомпилированного бинарного файла.

4. В случае, если ваш проект использует системную библиотеку функий (например: libft), ваш makefile должен автоматически скомпилировать ее эту библиотеку.

5. Все файлы исходники, которые вам нужны для компиляции вашего проекта, должны быть явно названы (указаны) в вашем Makefile.

**III.1 - Norminette errors**

**1. Error: global scope bad aligned**

Найдите в своей функции строчку с объявлением функции вида int main.

Удалите пробел после int и вставьте после него два Tab.

2. Error: 42 header not at top of the ^le

Вставьте заголовок 42 школы (fn+F1 в vim). Если он есть - попробуйте сделать его заново. (Ваш файл должен иметь расширение .с!)

3. Error: must not begin by spaces/tabulations

Уберите пробелы или дабы из начала.

4. Error: bad indentation

Проверьте отступы перед командами в функции! Должно быть как на примере. Только ТАБы, длиной в 4 пробела.

5. Error: Space before FUNCTION\_NAME

Замените пробел перед именем функции на Tab.

6. Error: bad spacing after ft\_print\_comb

После конца функции (после последней ‘}’) не правильно задано количество пустых строк: должна быть одна.

7. Error: function XXXXXXXXX has XX lines

Функция содержит слишком много строк! Максимум 25 строк!

8. Error: declarations must be followed by one empty line in XXXXXXXXX

Проверьте, как вы объявляете переменные! Расстановка пустых строк должна быть как на примере, и объявлять можно по одной переменной за раз! После объявления переменных должна быть пустая строка(смотри пример!).

7. Error: missing space around <

Не `“a<b”`, а `“a < b”`.

8. Error: global scope bad aligned

проверьте дабы перед именами функций. Названия функций должны быть одна над другой, строго, по всей программе! Для выравнивания используйте Tab.

9. declarations in XXXXXXXXX are bad aligned

Проблема в объявлении переменных в функции. С помощью ТАБов выровняйте их так, чтобы они располагались одна над другой, красивым столбиком. Если есть типы данных с длинным названием, то перед короткими (типа int) прийдется ставить несколько Tab.

10. Warning: /Users/XXXXXXXX may not compile or is invalid for some reasons.

Проверьте базовый синтаксис С. Это можно сделать следующим образом: добавьте int main(){} в конец программы и скомпилируйте с помощью gcc - и он укажет на ошибки в программе.

11. Error : declaration of ft\_putchar forbidden in .c

Вы запускаете norminette без флагов. Используйте norminette -R CheckForbiddenSourceHeader (Moulinette будет использовать такие же флаги!)

12. Error: missing void in function main

Если ваша функция не принимает на вход никаких аргументов, то это необходимо указать явно: int main(void){}.

13. Error: multiple empty lines

Просто удалите лишние пустые строки.

14. Error : wrong number of spaces in preprocessor indentation

Перед de^ne должен быть пробел: # de^ne