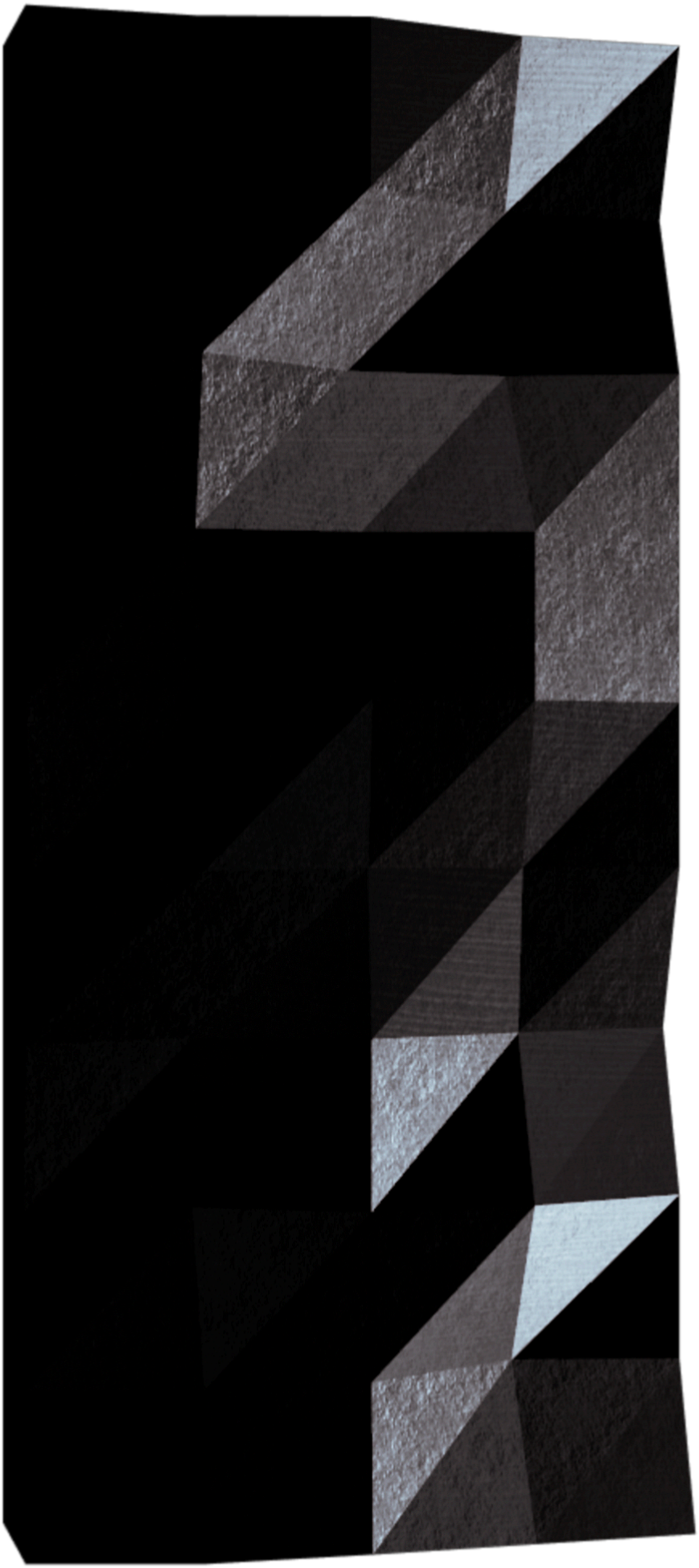

Implémentation de la norme de 42
Version 2.0.1

Formatage

- Un nom de structure doit commencer par s_.
- Un nom de typedef doit commencer par t_.
- Un nom d'union doit commencer par u_.
- Un nom d'enum doit commencer par e_.
- Un nom de globale doit commencer par g_.
- Les noms de variables, de fonctions doivent être composés exclusivement de mi-
nuscules, de chiffres et de '_' (Unix Case).
- Les noms de fichiers et de répertoires doivent être composés exclusivement de minuscules, de chiffres et de '_' (Unix Case).
- Le fichier doit être compilable.
- Les caractère ne faisant pas partie de la table ascii standart ne sont pas autorisés.
- Tous vos fichiers devront commencer par le header standard de 42 dès la première ligne. Ce header est disponible par defaut dans les éditeurs emacs et vim sur les dumps.
- Vous devez indenter votre code avec des tabulations de la taille de 4 espaces. Ce n'est pas équivalent à 4 espace, ce sont bien des tabulations.
- Chaque fonction doit faire au maximum 25 lignes sans compter les accolades du bloc de la fonction.

- Chaque ligne ne peut faire plus de 80 colonnes, commentaires compris. Attention : une tabulation ne compte pas pour une colonne mais bien pour les n espaces qu'elle represente.
- Une seule instruction par ligne.
- Une ligne vide ne doit pas contenir d'espace ou de tabulation.
- Une ligne ne devrait jamais se terminer par des espaces ou des tabulations.
- Quand vous rencontrez une accolade ouvrante ou fermante ou une fin de structure de controle, vous devez retourner à la ligne.

- Chaque virgule ou point-virgule doit être suivi d'un espace si nous ne sommes pas en fin de ligne.
- Chaque opérateur (binaire ou ternaire) et opérandes doivent être séparés par un espace et seulement un.
- Chaque mot-clé du C doit être suivi d'un espace, sauf pour les mot-clefs de type (comme int, char, float, etc.) ainsi que sizeof.
- Chaque déclaration de variable doit être indentée sur la même colonne.
- Les étoiles des pointeurs doivent être collés au nom de la variable.
- Une seule déclaration de variable par ligne.
- On ne peut faire une déclaration et une initialisation sur une même ligne, à l'ex- ception des variables globales et des variables statiques.
- Les déclarations doivent être en début de fonction et doivent être séparées de l'implémentation par une ligne vide.
- Aucune ligne vide ne doit être présente au milieu des déclarations ou de l'implé- mentation.

- La multiple assignation est interdite.
- Vous pouvez retourner à la ligne lors d'une même instruction ou structure de contrôle, mais vous devez rajouter une indentation par parenthèse ou opérateur d'affectation. Les opérateurs doivent être en début de ligne.
- Une fonction prend au maximum 4 paramètres nommés.
- Une fonction qui ne prend pas d'argument doit explicitement être prototypée avec le mot void comme argument.

Fonctions

- Les paramètres des prototypes de fonctions doivent être nommés
- Chaque définition de fonction doit être séparée par une ligne vide de la suivante.
- Vous ne pouvez déclarer que 5 variables par bloc au maximum.

• Vos identifiants de fonctions doivent être alignés dans un même fichier. Cela s'ap- plique aux headers C.

Typedef, struct, enum et union

- Vous devez mettre une tabulation lorsque vous déclarez une struct, enum ou union.
- Lors de la déclaration d'une variable de type struct, enum ou union, vous ne
  mettrez qu'un espace dans le type.
- Vous devez utiliser une tabulation entre les deux paramètres d'un typedef.
- Lorsque vous déclarez une struct, union ou enum avec un typedef, toutes les règles s'appliquent et vous devez aligner le nom du typedef avec le nom de la struct, union ou enum.

- Vous ne pouvez pas déclarer une structure dans un fichier .c.

Headers

- Seuls les inclusions de headers (système ou non), les déclarations, les defines, les prototypes et les macros sont autorisés dans les fichiers headers.
- Tous les includes de .h doivent se faire au début du fichier (.c ou .h).
- On protégera les headers contre la double inclusion. Si le fichier est ft_foo.h, la
  macro témoin est FT_FOO_H.
- Les prototypes de fonctions doivent se trouver exclusivement dans des fichiers .h.
- Une inclusion de header (.h) dont on ne se sert pas est interdite.

Macros et Pré-processeur

- Les defines définissant du code sont interdits.
- Les macros multilignes sont interdites.
- Seuls les noms de macros sont en majuscules
- Il faut indenter les caractères qui suivent un #if , #ifdef ou #ifndef

Choses Interdites !

• Vous n’avez pas le droit d’utiliser : ○ for

○ do...while ○ switch
○ case
○ goto

• Les opérateurs ternaires ‘ ?’ imbriqués
• Les tableaux à taille variable (VLA - Variable Length Array)

Commentaires

- Les commentaires peuvent se trouver dans tous les fichiers source.
- Il ne doit pas y avoir de commentaires dans le corps des fonctions.
- Toutes les lignes intermédiaires s’alignent sur elles, et commencent par ‘**’.
- Les zones de commentaires commencant par // sont interdits.

Les fichiers

• Vous ne pouvez pas inclure un .c.
• Vous ne pouvez pas avoir plus de 5 définitions de fonctions dans un .c.

Makefile

* Les règles $(NAME), clean, fclean, re et all sont obligatoires.
- Le projet est considéré comme non fonctionnel si le Makefile relink.
- Dans le cas d’un projet multibinaire, en plus des règles précédentes, vous devez avoir une règle all compilant les deux binaires ainsi qu’une règle spécifique à chaque binaire compilé.
- Dans le cas d’un projet faisant appel à une bibliothèque de fonctions (par exemple une libft), votre makefile doit compiler automatiquement cette bibliothèque.
- Les sources nécessaires à la compilation de votre programme doivent être explicitement cités dans votre Makefile.