Edition et compilation des programmes

1. Sauter des lignes pour structurer les déclarations

* laisser toujours une ligne blanche pour séparer les déclarations de variables en tête de bloc des instructions proprement dites
* laisser aussi une ligne blanche entre les#include et les #define, et le reste des déclarations.
* même chose entre les déclarations de variables globales et les déclarations de fonctions
* même chose entre les définition de fonctions différentes

De manière générale, aérer et structurer le code en sautant des lignes :
* pour séparer des déclarations de variables attachées à un même concept.
* pour séparer des groupes d'instructions utilisées pour accomplir une même tâche

Ces sauts de ligne permettront souvent de se passer de commentaires.

2. Améliorer la lisibilité en ajoutant des blancs autour des opérateurs

* utiliser des blancs autour des opérateurs. En particulier autour de l'affectation: c'est plus lisible.
* utiliser des blancs pour rendre lisible les expressions conditionnelles complexes dans les if ou les boucles while

```
if  (   ( (a > b) && (a > c) )   ||   (c > d)  )
```

même chose pour être sûr de la priorité des opérateurs.

Mais n'utilisez surtout pas de blancs pour réaliser l'indentation: n'utilisez que les caractères de tabulation. En effet, l'introduction de blancs pour l'indentation rend difficile la réutilisation ou modification par couper/coller, et cela fait perdre beaucoup de temps lors de l'édition de programmes.

3. Indentation et présentation du code des instructions

* n'utiliser que des caractères tabs pour réaliser l'indentation (jamais de caractères blancs)
* mettre une seule instruction par ligne (une même instruction peut figurer sur plusieurs lignes, mais une ligne ne doit pas contenir plusieurs instructions).
* matérialiser l'indentation des blocs entre accolades, en alignant toujours l'accolade fermante sur le début de l'instruction qu'elle termine.

On recommande généralement la disposition suivante :

```
while ( a > b )
{
        instr;
        instr;
}
```

qui a l'avantage de la redondance et la clarté (ici alignement du mot while et des deux accolades).

Remarque: mais la position de l'ouvrante est moins importante que celle de la fermante. On peut placer l'accolade ouvrante en bout de ligne de début d'instruction : cela permet de sauver une ligne sans inconvénient majeur. Par contre, le mot "while" doit être bien aligné sur l'accolade fermante, et les instruction du bloc doivent être en retrait d'un niveau de tabulation :

Ainsi, on acceptera la variante :

```
while ( a > b )  {
        instr;
        instr;
}
```

* Même disposition pour les blocs de définitions des fonctions. Il faut en outre indenter les déclarations de variables locales, et indenter le corps de définition :

```
long toto (int qdf, unsigned long truc)
{
        int i, j;

        /* premières instructions */
        etc;
}
```

Mais on pourra ici aussi accepter que l'accolade ouvrante figure en bout de ligne de déclaration de la fonction pour économiser une ligne:

```
long toto (int qdf, unsigned long truc)  {
        int i, j;

        ...
}
```

* Bien indenter les instructions des boucles, surtout quand elles sont réduites à une seule instruction:

```
        while (fghkfjg)
                instruction;
        for (i=1;  i< 10;  i++)
                x = x + i;
```

* Mettre une instruction sur plusieurs lignes si nécessaire, en particulier dans les appels de fonctions qu'on peut toujours scinder entre les arguments d'appels. Néanmoins, il est alors recommandé d'indenter le reste des arguments au moins à hauteur de la parenthèse ouvrante de l'appel, comme ici, pour que l'appel de fonction soit bien lisible :

```
        for ( ; ; ) {
                printf("impression%d de %f truc %d machin%echose\n",
                        x, y, z, t);
                FonctionLongue (long_argument, encore_un_tres_long,
                                        fgh, fgjh);
        }
```

Remarque: pour couper une chaîne de caractères trop longue (dans un printf par exemple) on utilise le caractère backslash \ dans l'éditeur de texte du fichier source pour qu'il annule (ce traitement est effectué par le pré-compilateur) le caractère de retour à la ligne qui suit dans l'éditeur de texte quand on va à la ligne. Mais on doit alors ensuite venir placer les caractères restants en tout début de ligne sur la ligne suivante de l'éditeur (sinon la chaîne entre guillemets contiendra les blancs du début de la nouvelle ligne). Exemple :

```
                printf ("cette chaine entre guillemets ne sera pas coupee\
mais continue ici en tout debut de ligne dans l'editeur de programme %d %d %d\n", x, y, z);
```

4. Convention de noms pour les constantes et les variables

On note en majuscules (séparées par des caractères soulignés s'il y a plusieurs mots) les noms des constantes introduites par #define. Cette convention est aussi appliquée aux constantes d'un type énumération.

Par contre on commence par une minuscule les noms de fonctions ou de variables. On peut utiliser le caractère souligné, ou l'alternance majuscule/minuscule pour séparer des mots dans ces identificateurs. Mais on évitera de faire commencer un identificateur de fonction par le

caractère souligné, car c'est la pratique des programmeurs système (pour éviter des confusions possibles de noms entre les fonctions de l'utilisateur et celles du système ou des bibliothèques).

```
#define MAX 800
#define MINIMUM_VITAL     100


int racineCarreeDe( int x)
{
        ...
}
```

5. Noms des programmes

Le nom d'un programme sera en minuscule pour être semblable à celui d'une commande Linux. Les noms de répertoires de fichiers commenceront par contre par une majuscule, et les fichiers par des minuscules. Si le programme fait des définitions de constantes ou de fonctions, on créera un fichier de déclarations (externes) pouvant être utilisé comme en-tête dans d'autres fichiers implémentant d'autres unité de compilation du programme. Dans ce cas, si le fichier contenant le main s'appelle toto.c, on fera un fichier d'entête toto.h, et on compilera l'exécutable du programme en toto.