Programmation structurée, types de données, volumes de données

Le TD commence par une réflexion *commune* sur la création d'un petit logiciel.

L'énoncé du problème : Concevez (i.e. réfléchissez aux structures de données utiles *puis* aux fonctions utiles... sans programmer une ligne encore) une procédure `voirChromosome` qui prend en entrée un chromosome en tant que séquence sous la forme :

- soit (lorsque le cours sur la lecture de fichier aura eu lieu) d'un fichier de nom `f` qui contient une suite de caractères `A`, `T`, `G` ou `C`,
- soit directement d'une chaîne de caractères `c` ne contenant que des caractères `A`, `T`, `G` ou `C`,

a priori le nombre de caractères est donc très grand. La procédure `voirChromosome` affiche le chromosome sur un écran de 100 caractères de large avec :

- une première ligne indiquant le pourcentage en `GC` du chromosome,
- puis le chromosome lui même avec une numérotation des lignes commençant à 1,
- et le pourcentage en `GC` de chaque ligne du chromosome (indiqué en fin de ligne).

Voici un exemple de début de sortie d'écran :

```
Pourcentage en GC = 56 %

1: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG (50%)
2: ATATATATATATATATATATGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGCGC (75%)
3: .......etc.....
```

La démarche : on commence par évaluer les besoins des utilisateurs :

- le nombre approximatif de nucléotides par ligne
- le nombre de lignes incroyablement grand
- l'intérêt de calculer le pourcentage global à partir des pourcentages locaux en terme de temps de calcul
- le coût de mettre le pourcentage global de GC en premier plutôt qu'à la fin...
- l'intérêt de savoir si mono- ou multi-utilisateur sur une station
- les conséquences du choix mono-utilisateur avec beaucoup de mémoire en rabe

... et on en arrive au choix des structures de données :

- liste de chaînes de caractères pour stocker les lignes
- liste d'entiers pour stocker les pourcentages locaux
- entiers pour les numéros de ligne, le calcul des pourcentages, *etc.*

... qui implique les fonctions intermédiaires (et pas le contraire) :

- fonction de découpage de `c` en tranches de 80 caractères (et intérêt de définir une variable égale à 80) → fonction `coupe`
- fonction de calcul du pourcentage en GC d'un brin → fonction `pourcent`
- fonction de remplissage de la liste des pourcentages à partir de la liste des brins de 80 caractères → fonction `mapPourcent`
- fonction de moyenne de la liste des pourcentages → fonction `moyenne`
- procédure principale pour gérer l'IHM → procédure `voirChromosome`

Cela conduit à la suite du TD :

Exercice 1 : En supposant les fonctions précédentes déjà écrites, écrivez la procédure `voirChromosome`.

Réaliser les fonctions utiles :

Exercice 2 : Écrivez une fonction `coupe` qui prend en entrée un entier `n` et un nom de fichier `f` (ou une chaîne de caractères `c`) représentant un chromosome et fournit en sortie une liste dont les éléments :

- sont des chaînes de caractères de longueur exactement `n`, sauf le dernier qui peut être plus court
- qui représentent des brins consécutifs d'ADN tels que leur concaténation soit égale au chromosome `c`.

Exercice 3 : Écrivez la fonction `pourcent` qui prend en entrée un brin `b` (c'est-à-dire une chaîne de caractères contenant des `ATGC`) et retourne son pourcentage (entier) en `GC`.

Exercice 4 : Écrivez la fonction `mapPourcent` qui prend en entrée une liste de brins `l` et retourne la liste des pourcentages en `GC` correspondants (donc une liste de même taille).

Exercice 5 : Écrivez la fonction `moyenne` qui fournit la moyenne (entière) d'une liste d'entiers.