Séance 1

Module séquence Le module *séquence* stocke la séquence des $N+1$ mots qui ont été saisis précédemment. Les N mots les plus anciens constituent un N-gramme et le plus récent est un mot qui suit immédiatement ce N-gramme. Le stockage est géré sous la forme d'un tableau circulaire : si l'indice de parcours du tableau avance au delà de sa fin, il est alors ré-initialiser pour désigner la première case du tableau.

Lorsqu'un nouveau mot est saisi, il est écrit dans le tableau à une position "courante" gérée dans le module. Une fonction du module permet de faire progresser la séquence en intégrant le nouveau mot dans le N-gramme. En conséquence la case qui stockait le mot le plus ancien du N-gramme précédant est libérée, et peut être utilisée pour accueillir le mot suivant. La figure 2 illustre ce processus.

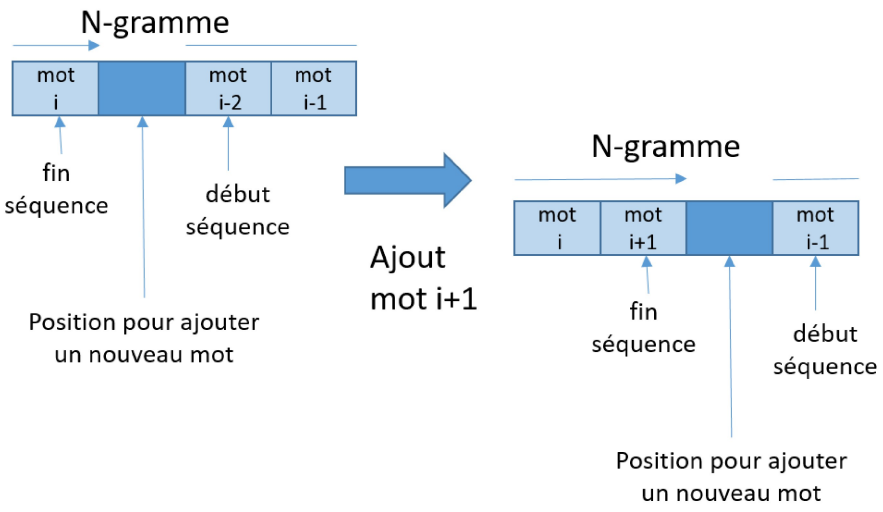

FIGURE 2 – Stockage et mise à jour d'une séquence de mots

L'interface du module séquence est donnée ci-dessous.

```
// nombre de mots dans les N-grammes
enum { Lg_N_gramme=3 };
// initialise le N-gramme avec des mots vides
void sequence_initialize( struct strhash * ht );
// Gestion de l'iterateur permettant de parcourir le N-gramme
// initialise l'iterateur sur le premier mot du N-gramme courant
void sequence_itStart( void );
// retourne le mot correspondant a la position de l'iterateur
// avance la position de l'iterateur
const char * sequence_itNext( void );
// test si l'iterateur est a la fin du N-gramme
int sequence_itHasNext( void );

/ Nouveau mot de fin du prochain N-gramme
// definit le nouveau mot qui entrera dans le N-gramme
void sequence_addWord( const char * wordi, struct strhash *ht );
// retourne le nouveau mot qui entrera dans le N-gramme
const char *  sequence_nextWord( void );
// avance le N gramme pour integrer le nouveau mot ecrit
void sequence_progress( void );

// Debug
// affiche le N-gramme courant, les mots sont separes par des /
void sequence_print( void );
// sequence sous forme d une chaine de caracteres
char *  sequence_printInTab( void );
```

Pour la mise en œuvre du module, il est imposé :

1. De respecter l'interface qui précède,
2. D'implanter les variables de stockage et de gestion du tableau sous la forme de variables externes privées (`static`). Une seule instance de séquence peut être gérée à la fois.
3. De stocker les mots de la séquence dans **une table de hachage**. La mise en œuvre de la table de hachage garantit l'absence de mots identiques au sein de celle-ci. En effet, dans le cas où l'on tente d'insérer un mot déjà présent dans le table, l'adresse du mot identique qui s'y trouve est retournée. Cette implantation permettra dans la suite du projet de tester si 2 mots sont identiques par comparaison de pointeurs, sans avoir à faire appel à la fonction *strcmp()*.

Les fichiers objets `hash.o` et `list.o`, qui mettent en œuvre la table de hachage, et son interface publique (.h), vous sont fournis. Ne pas les ré-écrire !

Travail à réaliser

1. Implanter les fonctions du module séquence.
2. Écrire le test unitaire du module.