TP 3

Les conteneurs associatifs

Exercice 1 Compter le nombre d'occurrences d'un mot

Utiliser un conteneur `map` pour compter le nombre de fois qu'un mot apparaît dans une phrase entrée par l'utilisateur.

Affichez ensuite le résultat (mot, nombre de fois vu).

Exercice 2 Découper une phrase en mots

Le but est d'écire une fonction qui lit une chaine de caractères qui peut contenir des espaces, et la coupe en mots. A la fin on a un pour chaque ligne un vecteur de mots.

Pour réaliser cela nous allons nous aider de la fonction `find_if` de la bibliothèque `<algorithm>` et de la fonction `isspace`.

```
iterator find_if ( iterator first, iterator last, Predicate pred );
//Returns an iterator to the first element in the range [first,last)
//for which applying pred to it, is true.

int isspace ( char c );

// un autre constructeur de string
string (iterator begin, iterator end)
```

Ecrire la fonction `split` qui décompose une ligne en un vecteur de mots. L'idée est de déterminer le début et la fin de chaque mot puis de les ranger dans un vecteur.

Exercice 3 Générer une table de cross-reference

Le but est d'écrire un programme qui génère une table de *cross-reference* qui indique à quelles lignes apparaissent chacuns des mots d'un texte.

Nous allons maintenant écrire la fonction `xref` qui construit la table.

1. Quels types pour la table ?
2. Nous voudrions faire en sorte de pouvoir changer la façon de trouver les mots, comment faire ?
3. Allons y ! Testons ensuite la fonction en l'affichant.