Travaux Pratiques d'algorithmique n°8

Cours d'Informatique de Deuxième Année

—L2.1—

Listes chaînées (suite)

Ce TP ne comporte pas de notions nouvelles. C'est un "prolongement" du TP précédent. Vous le terminerez en "incluant dans un menù" toutes les fonctions écrites sur les listes chaînées.

Nous reprenons les définitions de type du TP précédent permettant de représenter une liste chaînée d'entiers (que nous appellerons simplement une liste) :

```
typedef struct _cellule
{
   int Valeur;                      /* donnee stockee : un entier.       */
   struct _cellule * Suivant;       /* pointeur sur la cellule suivante. */
} Cellule;                          /* definition d'un nouveau type.     */
typedef Cellule * Liste;            /* definition d'un nouveau type.     */
```

▸ **Exercice 1.** *La fonction* `AfficheListe(Liste p)` *du TP précédent permet d'afficher, dans l'ordre du chaînage, les éléments de la liste passée en paramètre. Déduire de cette fonction une autre fonction* `SauveListe(Liste p, FILE *destination)` *qui sauvegarde les valeurs de la liste passée en premier paramètre dans le fichier passé en deuxième paramètre (on supposera que le fichier a été correctement ouvert).*

▸ **Exercice 2.** *La fonction* `Liste LireListe(Liste *L)` *du TP précédent permet de créer une liste chaînée d'entiers non nuls en partant de la liste chaînée vide et en insérant successivement les entiers lus, depuis l'entrée standard, en tête de liste. Déduire de cette fonction une autre fonction* `Liste ChargeListe(FILE *source)` *de telle sorte que la liste soit créée à partir d'entiers lus dans le fichier passé en paramètre (on supposera que le fichier a été correctement ouvert).*

▸ **Exercice 3. (Crible d'Eratosthène)**

- *Écrire une fonction* `Construct` *prenant en argument un entier* `n` *et qui retourne la liste des entiers compris entre 2 et* `n` *dans l'ordre croissant.*

- *Écrire une fonction* `ElimineMultiples` *prenant en argument une liste d'entiers* `l` *et un entier* `a` *et qui supprime tous les multiples de* `a` *dans la liste* `l` *;*

On veut maintenant implémenter, en utilisant des listes chaînées, l'algorithme du crible d'Eratosthène qui permet de calculer tous les nombres premiers inférieurs ou égaux à un entier donné `n` *: partir de la liste de tous les entiers de* 2 *à* n*, puis prendre successivement les éléments de la liste et supprimer tous leurs multiples stricts.*

- *Déduire des questions précédentes une fonction* `Crible` *prenant en argument un entier* `n` *et qui retourne la liste de tous les entiers premiers inférieurs ou égaux à* `n`*.*

▸ **Exercice 4.** *Écrire une fonction* `int KiemeElement(Liste p, int k, int *val)` *qui calcule la valeur du* `k`*ième élément de la liste* `p`*. Par exemple, le* 3*ième élément de la liste* $(5, 7, 9, 10, 2)$ *est* 9*. La fonction retournera* 0 *si la liste a moins de* k *éléments et* 1 *sinon.*

▸ **Exercice 5.** *Écrire une fonction qui teste si une liste est triée (pour l'ordre croissant).*

▸ **Exercice 6.** *Écrire une fonction qui teste si deux listes sont égales (mêmes valeurs aux mêmes positions).*

▸ **Exercice 7.** *Écrire des fonctions pour*

- *dupliquer une liste (même nombre de cellules, avec mêmes valeurs aux mêmes positions) ;*
- *libérer tout l'espace mémoire occupé par une liste.*