Travaux Dirigés d'algorithmique n°3

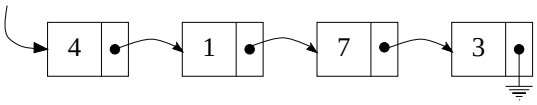
Cours d'Informatique de Deuxième Année

—DEUG Sciences S3-MI—

Listes chaînées

Dans cette séance de travaux dirigés, nous nous intéressons à l'implémentation des listes par des "listes chaînées" (plus précisément, il s'agit de "listes simplement chaînées par pointeurs").

Définition de la structure: Un élément d'une liste chaînée, appelé une *cellule*, contient les informations que l'on souhaite manipuler et également l'adresse de la cellule suivante. Par exemple, la liste d'entiers $(4, 1, 7, 3)$ est représentée par la liste chaînée d'entiers suivante :



Se donner une telle liste chaînée c'est se donner un pointeur sur le premier élément de la liste chaînée.

Les définitions de types suivantes mettent en œuvre la notion de "liste chaînée d'entiers" :

```
typedef struct Cellule
        { int Val;                  /* donnee stockee : un entier.       */
          struct Cellule * PtrSuiv; /* pointeur sur la cellule suivante. */
        } StrCellule;               /* definition d'un nouveau type.     */

typedef StrCellule * PtrCellule;    /* definition d'un nouveau type.     */
```

▸ **Exercice 1.** ***(Création d'une nouvelle cellule)***

- *Écrire une fonction* `AlloueCellule` *créant une nouvelle cellule* `nc` *et retournant un pointeur sur* `nc`.
- *En déduire une fonction de nom* `CreerCellule` *prenant en argument un entier* `val` *et retournant un pointeur sur une nouvelle cellule contenant la valeur* `val`. *Nous veillerons à ce que le champ* `PtrSuiv` *de cette cellule soit initialisé à la valeur* `NULL`.

▸ **Exercice 2.** ***(Insertion d'un élément)***

- *Écrire une fonction qui insère, en tête de liste, un entier dans une liste chaînée d'entiers. La fonction retournera un pointeur sur le premier élément de la liste ainsi modifiée.*
- *En déduire une fonction qui crée (par insertions successives des éléments) une liste chaînée d'entiers non nuls saisis au clavier.*

▸ **Exercice 3.** ***(Affichage d'une liste)***

- *Écrire une fonction de nom* `AffListe` *qui affiche les éléments de la liste chaînée passée en paramètre.*
- *Écrire une fonction de nom* `AffListeInv` *qui affiche, dans l'ordre inverse du chaînage, les éléments de la liste chaînée passée en paramètre.*

  Pour la liste chaînée donnée auparavant, `AffListe` *devra produire la sortie :* `4 1 7 3` *et* `AffListeInv` *la sortie :* `3 7 1 4`.

▸ **Exercice 4.** ***(Suppression dans une liste)***

- *Écrire une fonction* `EffaceUn` *qui supprime la première occurrence d'un entier* `n` *dans une liste chaînée.*
- *Écrire une fonction* `EffaceTout` *qui supprime toutes les occurrences d'un entier* `n` *dans une liste chaînée.*