

# Programmation C

---

## Listes chaînées par pointeurs

---

### Définition de la structure :

Dans les premiers exercices, on manipule des listes contenant une valeur entière. On utilisera donc les types :

```
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. Éléments d'une liste

- Écrire une fonction qui prend en argument une liste chaînée et retourne le nombre d'éléments qu'elle contient.
- Écrire une fonction qui prend en argument une liste chaînée et un entier `x` et retourne le nombre d'éléments inférieur à l'entier `x` qu'elle contient.
- Écrire une fonction de recherche d'un élément dans une liste. La fonction renvoie l'adresse de la cellule qui contient l'élément s'il est présent et `NULL` sinon.
- Écrire une fonction de recherche de l'élément minimum dans une liste. La fonction renvoie l'adresse de la cellule qui contient l'élément minimum ou `NULL`.
- Écrire une fonction qui prend en argument une liste chaînée et retourne le nombre d'éléments différents qu'elle contient.
- Écrire une fonction `estTrie` qui vérifie si une liste est en ordre croissant
- Donner la complexité des fonctions.

### Exercice 2. Concatène

Écrire une fonction qui reçoit deux listes et place les cellules de la deuxième à la fin de celle de la première. Après l'appel la deuxième liste doit être vide. Il n'y a pas de création de nouvelle cellule.

### Exercice 3. Ajout Extraction

- Écrire une fonction `Liste AlloueCellule(int x)` qui alloue si possible une nouvelle cellule contenant la valeur `x` et renvoie son adresse.
- Écrire une fonction `Liste ExtraireTete(Liste *l)` d'extraction de la cellule en tête de liste.
- Écrire une fonction `int InsereTete(Liste *l, int x)` d'insertion d'une cellule en tête de liste.
- Écrire une fonction `int InsereAprès(Liste *l, int x, int y)` qui insère une cellule contenant `y` après celle qui contient `x`, si `x` est présent.  
Si `x` n'est pas dans la liste, la cellule contenant `y` est insérée en fin de liste.
- Écrire une fonction `duplique` qui reçoit une liste et en crée une copie.

### Exercice 4. (Libération d'une liste)

Écrire une fonction `void LibereListe(Liste *l)` qui libère tout l'espace mémoire occupé par une liste chaînée.

**Exercice 5. (Liste de mots)**

- Définir les types `CelluleMot` et `ListeMot` nécessaires pour gérer des listes de mots.
- Que faut-il transformer dans les fonctions précédentes ?
- Écrire une fonction `ListeMot AlloueCelluleMot(char *m)` qui alloue une nouvelle cellule et recopie le paramètre `m` (attention 2 `malloc` et libération si echec!)
- Écrire la fonction d'insertion sans répétition pour ce type.