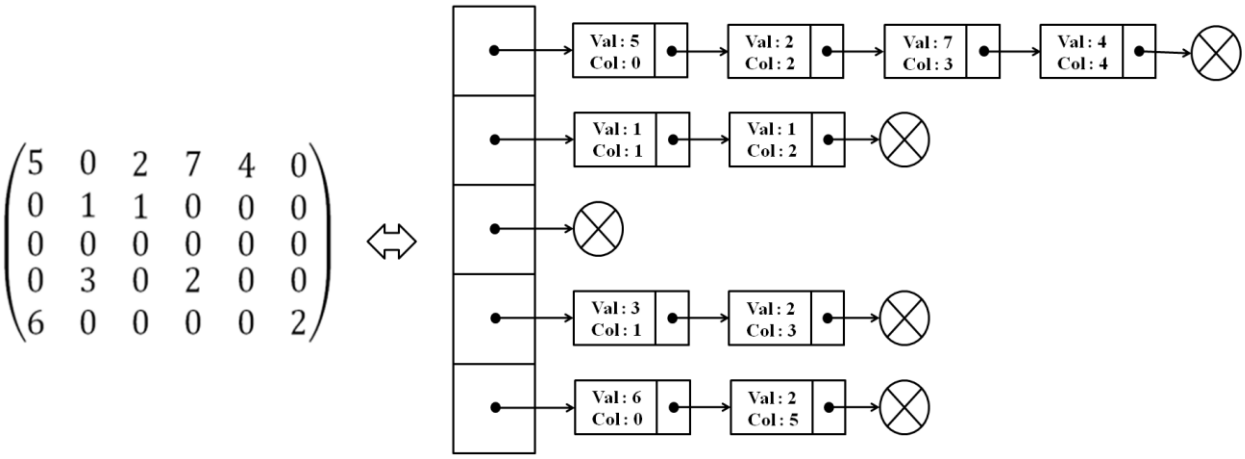
AI01/NF16 - TP3 : Les listes linéaires chaînées

Ce TP a pour objectif de se familiariser avec les listes chaînées et les différentes opérations nécessaires pour les manipuler.

Enoncé:

Une matrice est dite creuse lorsque le nombre d'éléments nuls y figurant est très supérieur à celui des éléments non nuls. Par souci d'économie, on peut représenter une telle matrice en ne tenant compte que des éléments non nuls : une matrice $N \times M$ est représentée par un tableau de N listes chaînées qui contiennent les éléments non nuls d'une ligne de la matrice ordonnés selon le rang de leur colonne. Chaque élément d'une liste contient l'indice de la colonne et la valeur de l'élément. Par exemple, voici une matrice creuse avec la représentation correspondante.



A. Structures de données :

Définir les structures de données suivantes :

- Le type **`element`** associé à une structure qui comporte les champs :
  - Un indice de colonne de type *int*.
  - Une valeur de type *int*.
  - Un pointeur sur l'élément suivant.

- Le type **`liste_ligne`** qui est un pointeur de type **`element`**

- Le type **`matrice_creuse`** associé à une structure qui comporte les champs :
  - Un pointeur de type **`liste_ligne`** qui représente un tableau de N éléments de type **`liste_ligne`** (le tableau sera donc alloué dynamiquement avec la fonction *malloc* en fonction du nombre de lignes de la matrice).
  - Un entier **`Nlignes`** qui représente le nombre de lignes de la matrice.
  - Un entier **`Ncolonnes`** qui représente le nombre de colonnes de la matrice.

B. Fonctions à implémenter :

1. Ecrire une fonction qui permet de remplir une matrice creuse, représentant une matrice de taille $N \times M$, à partir d'une saisie de l'utilisateur au clavier (l'utilisateur devra donc entrer successivement $N \times M$ nombres entiers positifs, négatifs ou nuls).

Le prototype de la fonction est comme suit :

```
void remplirMatrice(matrice_creuse *m, int N, int M);
```

2. Ecrire une fonction qui permet d'afficher une matrice creuse sous forme de tableau à deux dimensions (afficher à l'écran N lignes et M colonnes) :

```
void afficherMatrice(matrice_creuse m);
```

3. Ecrire une fonction qui permet d'afficher toutes les listes chaînées qui représentent une matrice creuse :

```
void afficherMatriceListes(matrice_creuse m);
```

4. Ecrire une fonction qui renvoie la valeur de l'élément de la ligne *i* et la colonne *j* de la matrice. Le prototype de la fonction est comme suit :

```
int rechercherValeur(matrice_creuse m, int i, int j);
```

5. Ecrire une fonction qui affecte une valeur donnée à l'élément de la ligne *i* et la colonne *j* de la matrice.

```
void affecterValeur(matrice_creuse m, int i, int j, int val);
```

NB : **val** *peut avoir pour valeur zéro !*

6. Ecrire une fonction qui réalise, de manière la plus optimisée possible, la somme de deux matrices données de mêmes dimensions, et sauvegarde le résultat directement dans la première matrice (ne pas utiliser les fonctions *rechercherValeur* et *affecterValeur*). Le prototype de la fonction est comme suit :

```
void additionerMatrices(matrice_creuse m1, matrice_creuse m2);
```

7. Ecrire une fonction qui retourne le nombre d'octets gagnés par cette représentation pour une matrice creuse donnée (attention à bien tenir compte de la taille d'un entier et de la taille d'un pointeur) :

```
int nombreOctetsGagnes(matrice_creuse m);
```

C. Interface:

Utiliser les fonctions précédentes pour écrire un programme permettant de manipuler les matrices creuses. Le programme propose un menu qui contient les fonctionnalités suivantes :

1. Remplir une matrice creuse
2. Afficher une matrice creuse sous forme de tableau
3. Afficher une matrice creuse sous forme de listes
4. Donner la valeur d'un élément d'une matrice creuse
5. Affecter une valeur à un élément d'une matrice creuse
6. Additionner deux matrices creuses
7. Calculer le gain en espace en utilisant cette représentation pour une matrice donnée
8. Quitter

***NB** : votre programme devra permettre de gérer plusieurs (deux minimum) matrices creuses*

Consignes générales:

- Sources

À la fin du programme, les blocs de mémoire dynamiquement alloués doivent être proprement libérés.

L'organisation MINIMALE du projet est la suivante :

- Fichier d'en-tête tp3.h, contenant la déclaration des structures/fonctions de base,
- Fichier source tp3.c, contenant la définition de chaque fonction,
- Fichier source main.c, contenant le programme principal.

- Rapport

Votre rapport de quatre pages maximum contiendra :

- La liste des structures et des fonctions supplémentaires que vous avez choisi d'implémenter et les raisons de ces choix.
- Un exposé succinct de la complexité de chacune des fonctions implémentées.

Votre rapport et vos trois fichiers feront l'objet d'une remise de devoir sur **Moodle** dans l'espace qui sera ouvert à cet effet (un seul rendu de devoir par binôme).