

Devoir Surveillé <input checked="" type="checkbox"/>	Examen <input type="checkbox"/>	Session principale <input checked="" type="checkbox"/>
		Session de contrôle <input type="checkbox"/>
Matière : Algorithmique et structures de données II		Semestre : 2
Enseignant(s) : Majdi Jribi et Rabaa Youssef		Date: 09 Mars 2022
Filière(s) : MPI		Durée: 1h30
Barème : 05-15		Documents : autorisés <input type="checkbox"/>
Nombre de pages : 02		non autorisés <input checked="" type="checkbox"/>

### Exercice 1 (05 pts)

Un nombre romain peut se composer des chiffres suivants(représentés par des lettres majuscules):

Chiffre romain	M	D	C	L	X	V	I
Valeur	1000	500	100	50	10	5	1

Un nombre romain est interprété grâce aux règles suivantes:

- La valeur du nombre est égale à celle du chiffre romain si celui-ci est composé d'un seul chiffre.
- Si le nombre contient au moins deux chiffres, on considère les deux chiffres les plus à gauche :
  - Si le 1<sup>er</sup> chiffre (le plus à gauche) est supérieur ou égal au 2<sup>ème</sup>, alors la valeur du nombre romain correspond à la **somme** de la valeur du 1<sup>er</sup> chiffre et de la valeur du nombre composé de tous les chiffres restants du nombre.
  - Si le 1<sup>er</sup> chiffre est inférieur au 2<sup>ème</sup>, alors la valeur du nombre correspond à la **différence** entre la valeur du nombre composé des chiffres restants et la valeur du 1<sup>er</sup> chiffre (le plus à gauche).

Exemples:

- IIII = 4 ; IV = 4
- VIII = 8 ; IX = 9
- MIM = 1999 ; MCMXCIX = 1999

1. Ecrire en langage C une fonction itérative  

```
int evaluer_chiffre_roman (Char c)
```

 qui renvoie la valeur numérique associée au chiffre romain passé en paramètre sous la forme d'un simple caractère.
2. Ecrire en langage C une fonction récursive  

```
int evaluer_nombre_roman(char * ch)
```

 qui reçoit en paramètre un nombre romain représenté par une chaîne de caractères, et qui renvoie le résultat de l'évaluation de ce nombre.

Remarque: on suppose que la syntaxe du nombre romain donné est correcte.

### Exercice 2 (15 pts)

Nous souhaitons modéliser les matrices à l'aide des listes chaînées. Pour cela nous définissons les structures de données suivantes :

```
typedef struct Noeud {
    float valeur;
    struct Noeud *next_col;
    struct Noeud *next_lig;
} Noeud;
```

```
typedef struct {
    Noeud *first;
    int nb_lignes;
    int nb_colonnes;
} Matrice;
```

La structure **Noeud** contient trois champs :

- Un champs de données **valeur** de type réel.
- Un champ pointeur sur la colonne suivante **next\_col** qui est un pointeur sur le nœud de la colonne suivante au nœud actuel.
- Un champ pointeur sur la ligne suivante **next\_lig** qui est un pointeur sur le nœud de la ligne suivante au nœud actuel.

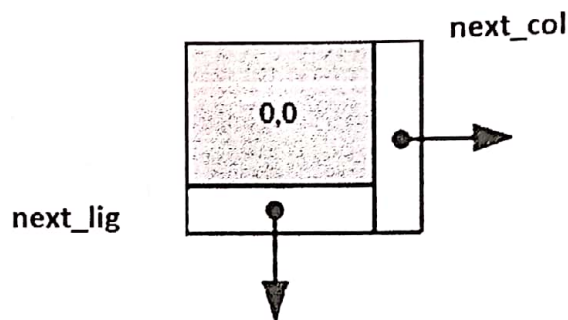


Figure 1 : Illsutration de la structure Noeud

Une matrice (qui correspond au type **Matrice**) est représentée par une structure de données contenant trois champs :

- Un champ **first** qui est pointeur sur Noeud et qui correspond à l'adresse de la matrice. C'est l'adresse de l'élément le plus à gauche et en haut de la matrice (comme mentionné par la figure 2).
- Un champ **nb\_lignes** de type entier qui indique le nombre de lignes de la matrice.
- Un champ **nb\_colonnes** de type entier qui indique le nombre de colonnes de la matrice.

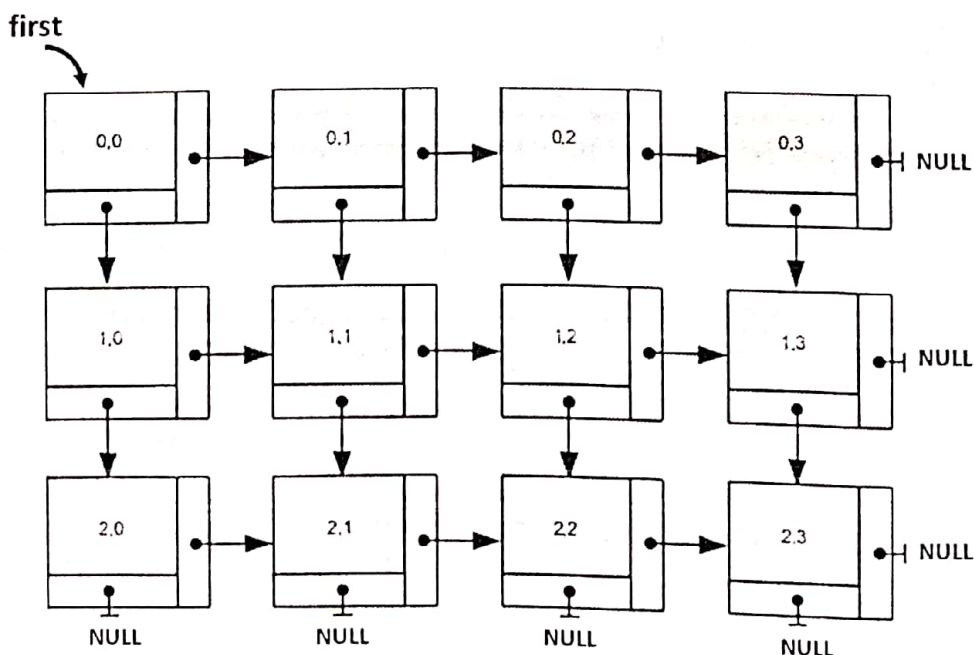


Figure 2 : Illustration d'une matrice avec nb\_lignes=3 et nb\_colonnes=4

1- Ecrire en langage C la fonction

**Matrice Creation (Matrice M, int nb\_lignes, int nb\_colonnes)**

qui permet de créer une Matrice **M** de **nb\_lignes** lignes et de **nb\_colonnes** colonnes. Les champs de données seront saisies par l'utilisateur.

2- Ecrire en langage C la fonction

**Matrice Insérer\_Lig (Matrice M, int nb\_lignes, int nb\_colonnes, Matrice Ligne, int pos)**

qui permet d'insérer dans la Matrice **M** de **nb\_lignes** lignes et de **nb\_colonnes** colonnes une matrice **Ligne** (contenant 1 ligne et **nb\_colonnes** colonnes) dans une ligne d'indice **pos** dans la matrice **M**)

3- Ecrire en langage C la fonction

**Matrice Insérer\_Col (Matrice M, int nb\_lignes, int nb\_colonnes, Matrice Colonne, int pos)**

qui permet d'insérer dans la Matrice **M** de **nb\_lignes** lignes et de **nb\_colonnes** colonnes une matrice **Colonne** (contenant **nb\_lignes** lignes et 1 colonne) dans une colonne d'indice **pos** dans la matrice **M**)

4- Ecrire en langage C la fonction

**Matrice Supp\_Lig (Matrice M, int nb\_lignes, int nb\_colonnes, int pos)**

qui permet de supprimer de la Matrice **M** de **nb\_lignes** lignes et de **nb\_colonnes** colonnes la ligne (contenant 1 ligne et **nb\_colonnes** colonnes) d'indice **pos** dans la Matrice **M**.

5- Ecrire en langage C la fonction

**Matrice Supp\_Col (Matrice M, int nb\_lignes, int nb\_colonnes, int pos)**

qui permet de supprimer de la Matrice **M** de **nb\_lignes** lignes et de **nb\_colonnes** colonnes la colonne (contenant **nb\_lignes** lignes et 1 colonne) d'indice **pos** dans la Matrice **M**.