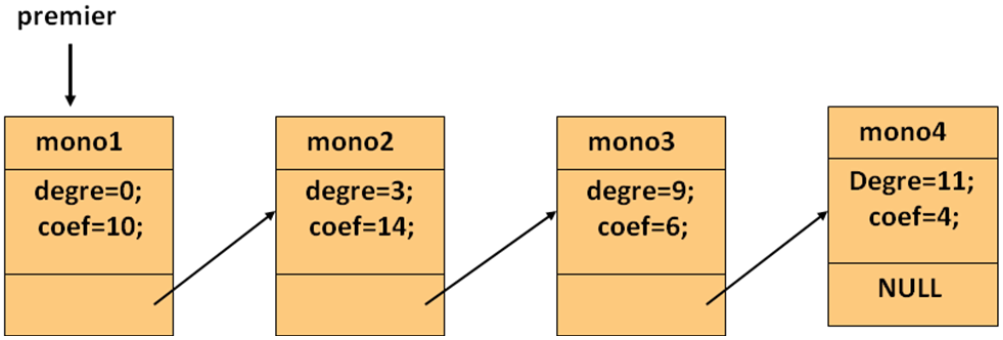
Module : Structures de données
Série- N°3 : Listes chaînées (suites)

Exercice 1 :

Pour cet exercice, on utilisera la liste chainée développée dans le cours.

Soit L un polynôme à une seule variable, qu'on peut le représenter par une liste chaînée dont les nœuds sont des monômes, (comme ci-dessous: $P(X)=10+ 14*X^3 + 6*X^9 + 4*X^{11}$).



Soient les déclarations suivantes :

struct donnees_monome{int degre; float coef;};

struct noeud_monome {struct donnees_monome D; struct noeud_monome *suivant;};

typedef struct donnees_monome DM;

typedef struct noeud_monome* mon;

1) Ecrire une fonction qui permet d'évaluer un polynôme en un point x, par exemple P(0)=10;
2) Ecrire une fonction qui permet de calculer fog(x)=f(g(x)), f et g sont deux polynômes, x est un réel donné.
3) Ecrire une fonction qui permet de multiplier un polynôme par un scalaire λ (multiplication de la liste par λ)
4) Ecrire une fonction qui permet de copier un polynôme A. Elle retourne une copie de A.
5) Ecrire une fonction qui permet de simplifier un polynôme A en remplaçant tous les monômes ayant des degrés égaux par un seul monôme qui est leur somme.
6) Ecrire une fonction qui permet de trier un polynôme A par ordre croissant des degrés de monômes (on suppose que A est simplifiée).
7) Ecrire une fonction qui permet de renvoyer une liste DeriveDuPoly qui est le polynôme dérivé du polynôme A.

8) Ecrire une fonction qui permet de renvoyer une liste DeriveDuPolyOrdreN qui est le polynôme dérivé d'ordre n du polynôme A.

9) Ecrire une fonction qui permet d'additionner deux polynôme A et B.

10) Ecrire une fonction qui permet de multiplier un polynôme A par un autre polynôme B.

Exercice 2 :

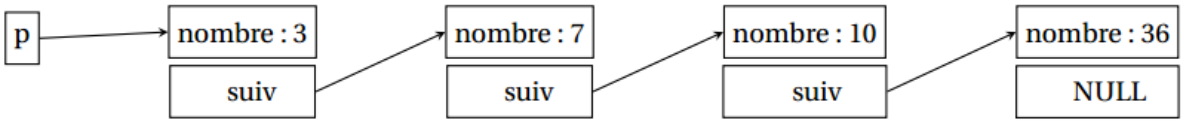
Dans cet exercice, on se propose de représenter des ensembles finis d'entiers strictement positifs triés par ordre croissant par des listes chaînées définies en langage C comme suit

typedef struct ens
{ int nombre ; //un nombre entier strictement positif élément de l'ensemble
struct ens *suiv ; // l'adresse de l'élément suivant
} ensembleListe ;

Appellation:On appellera "Ensemble Liste d'adresse p" une liste chaînée d'éléments de type ensembleListe (définie plus haut) et possédant les propriétés suivantes :

- Le premier élément a l'adresse p.
- Le dernier élément a dans son champ suiv la valeur NULL
- Pour tout élément d'adresse el de la liste chaînée, tel que (el->suiv !=NULL), on a (0< el->nombre < ((el->suiv) ->nombre) (liste triée par ordre croissant de nombres) Exemple

L'ensemble {3,7,10,36} sera représenté par l'Ensemble Liste d'adresse p comme suit :



Soit la déclaration globale suivante : ensembleListe *p ;

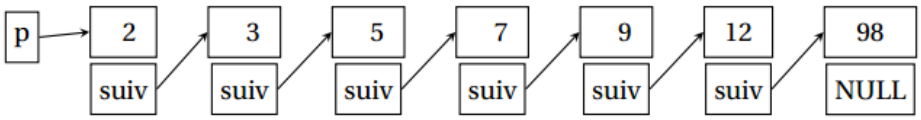
On suppose avoir définie et inséré des éléments dans l' "Ensemble Liste d'adresse p" (p est déclaré plus haut).

Travail à faire:

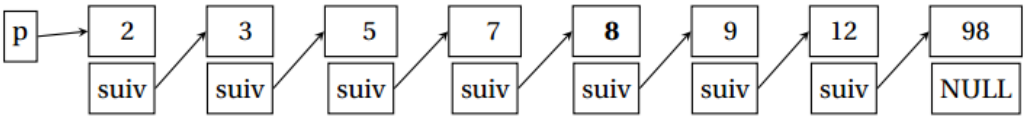
Écrire une fonction d'entête : void inserer(int val) qui permet d'insérer à sa place l'élément de type ensembleListe dans l' "Ensemble Liste d'adresse p" pour que la liste reste toujours triée par ordre croissant. Cet élément a dans son champ nombre, la valeur val (paramètre de la fonction), en plus, on suppose que : (val>p->nombre)(voir remarque et exemple) Remarque

- Si le paramètre val est la valeur du champ nombre d'un élément qui existe déjà dans la liste, aucun élément ne sera inséré.

Exemple : Soit l' "Ensemble Liste d'adresse p" suivant :



Après l'appel de la fonction **inserer(8), l' "Ensemble Liste d'adresse p"** devient :



Exercice 3 :

Représentation des grands nombres par des listes chainées

Dans cette exercice, on utilisera des listes chainées pour représenter des nombres entiers positifs ou nuls.

Pour optimiser la représentation d'un nombre, on se propose de le décomposer en plusieurs parties. Chaque partie peut contenir 4 chiffres.

Soit N un nombre entier positif ou nul de NC chiffres. $N=C_{NC-1}C_{NC-2}...C_i...C_1C_0$, ($C_0$ chiffre des unités, C_1 chiffre des dizaines,...), Alors N sera décomposé ainsi :

N = $\boxed{C_{NC-1}C_{NC-2}C_{NC-3}C_{NC-4}}$ $\boxed{C_{NC-5}C_{NC-6}C_{NC-7}C_{NC-8}}$ … $\boxed{...C_1C0}$

1ère partie 2ème partie ... dernière partie

Exemple : le nombre N= **8002590300407896420003** peut être décomposé ainsi :

N= 8002 5903 0040 7896 4200 03

1ère partie 2ème partie 3ème partie 4ème partie 5ème partie 6ème partie

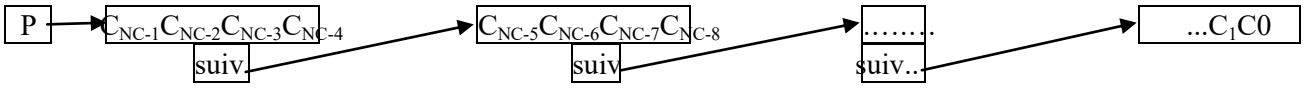
Pour représenter un nombre en le découpant ainsi, on va utiliser une liste chainée de type **ListeNombres** définie en langage **C** somme suit :

typedef struct Liste { short

int partie;

struct Liste * suiv; //l'adresse de l'élément suivant **}ListeNombres;**

Chaque partie du nombre est un élément de la liste chainée contenant la valeur de la partie et l'adresse de la partie suivante. Le nombre N=CNC-1CNC-2...Ci...C1C0 sera représenté par une liste de type **ListeNombres** comme suit :



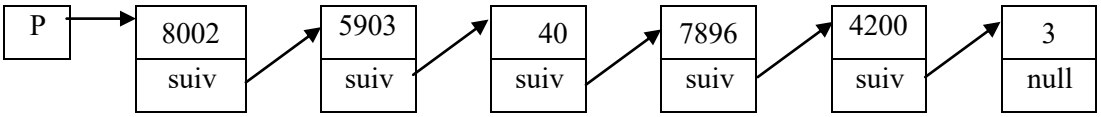
P (adresse du premier élément de la liste).

La valeur du champ partie étant déclarée de type **short int** , les **0**(**zéros**) non significatifs (à gauche) n'apparaitrons pas dans la valeur de la partie (voir exemple).

Exemple : Le nombre **8002590300407896420003** peut être décomposé ainsi :

N=	8002	5903	40	7896	4200	3
	1ère partie	2ème partie	3ème partie	4ème partie	5ème partie	6ème partie

et sera représenté par une liste de type **ListeNombres comme suit:**



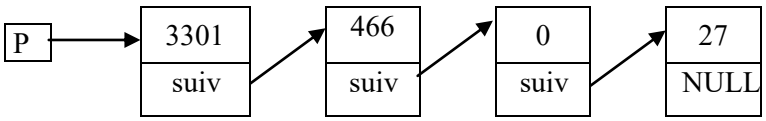
❖ **Création d'une liste chainée des parties d'un nombre**

Soit un nombre entier positif ou nul défini par une **ChaineChiffres S** (ne contenant que des caractères chiffres: '0','1', '2',...,'9'). On se propose de le représenter par une liste chainée de type **ListeNombres**. Pour ce faire :

- Ecrire une fonction de prototype : **ListeNombre * regrouperChiffres(char * S)**; qui permet de créer une nouvelle liste chainée de type **ListeNombres** pour représenter le nombre défini par la **ChaineChiffres S** (en paramètre) . Cette fonction retourne l'adresse du premier élément de cette liste crée.

Exemple : soit le nombre **330104660000027** représenté par une **ChaineChiffres S="330104660000027"**

L'appel de la fonction **regrouperChiffres(S),** va retourner l'adresse du premier élément p de la liste crée ainsi :



Plan

EX1 : Représentation d'un polynôme à une seule variable, par une liste chaînée dont les nœuds sont des monômes

EX2 : Représentation des ensembles finis d'entiers strictement positifs triés par ordre croissant par des listes chaînées

EX3 : Représentation des grands nombres par des listes chainées