|  |  |  |
| --- | --- | --- |
| Description : Afficher l'image d'origine | **Université Cadi Ayyad**  **Faculté des sciences Semlalaia Marrakech**  **Département d'informatique** | Description : Afficher l'image d'origine |

**Structure de données Réalisation en langage C : Listes chainées TD 2**

**Exercice 1 (Liste chainée simple)**

Un polynôme peut être représenté par une liste chainée dans laquelle chaque nœud contient un coefficient, un exposant et un pointeur vers le terme suivant.

1. Définir une fonction de création d’un polynome (**Polynome \* CreePolynomeVide()**).
2. Définir une fonction permettant d’ajouter des nœuds à un polynome (**void AjouterNoeud(Polynome \* po, int coef, int exp**).
3. Etablir un algorithme d’addition de deux polynômes (**polynome1** et **polynome2**).
4. Ecrire le programme C correspondant à vos fonctions et à votre algorithme.

**Exercice 2 (Liste chainée ordonnée)**

On cherche à gérer une liste chainée formée de nombres entiers. Chaque nombre introduit est immédiatement placé dans la liste en respectant l’ordre décroissant.

1. Ecrire un algorithme qui permet de gérer cette liste chainée
   1. Fonction de création de liste **InitListe()**
   2. Fonction **ListeVide()**
   3. Fonction **ListePleine()**
   4. Fonction d’**ajout (Liste \*L, int nombre)**
   5. Fonction de **suppression (Liste \*L, int nombre)**
   6. Fonction d’**affichage (Liste \* L)**
2. Ecrire le programme C correspondant à votre algorithme.

**Exercice 3 (Liste chainée bidirectionnelle)**

Une liste bidirectionnelle est une liste d’éléments qui sont chainés dans les deux sens.

1. Ecrire un algorithme pour insérer, supprimer et chercher des éléments dans une liste bidirectionnelle.
2. Traduire votre algorithme en C.