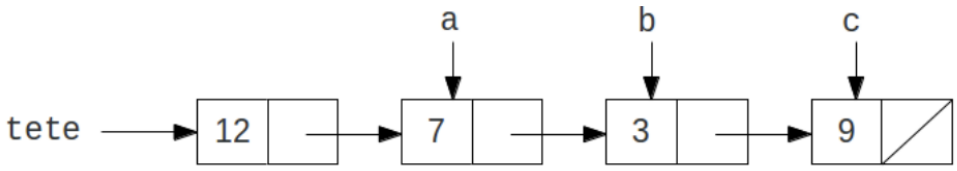
Série 0.2

Rappel:

- Une pile est une structure de données telle que :
  - L'ajout d'un élément se fait au sommet de la pile,
  - La suppression d'un élément se fait également au sommet de la pile.
  - On dit que c'est un algorithme LIFO, ce qui signifie "Last In First Out" : "Le dernier élément qui a été ajouté est le premier à sortir".
- Une file est une structure de données telle que :
  - L'ajout d'un élément se fait a la fin de la file,
  - La suppression d'un élément se fait contrairement à la fin de la file.
  - On dit que c'est un algorithme FIFO, ce qui signifie "First In First Out" : "Le premier élément qui a été ajouté est le premier à sortir".

Exercice 1:

1. Écrire un programme qui crée la liste chainée représentée dans la figure suivante :



2. Écrire une fonction qui cherche si une valeur existe dans la liste.
3. Ajouter au programme précédent les instructions qui permettent de créer et d'insérer l'élément pointé par d, pour que la liste soit toujours strictement croissante.
4. Écrire une fonction qui supprime une valeur dans la liste.
5. Écrire une fonction qui affiche le contenu de la liste.

Exercice 2:

Même structure que l'exercice 1.

1. Écrire une fonction inserer qui permet d'insérer un élément au début d'une liste chaînée d'entiers.
2. Écrire une fonction insererQueue qui permet d'insérer un élément à la fin d'une liste chaînée d'entiers.

Code :
https://github.com/hm43/ExercicesTpPASD.git

Série 0.2

3. Écrire une version récursive de la fonction afficher.
4. Écrire une fonction compter qui retourne le nombre d'éléments d'une liste chainée d'entiers.
5. Écrire une fonction somme qui retourne la somme des éléments d'une liste chaînée d'entiers non vides.
6. Écrire une fonction min qui retourne la valeur du plus petit élément d'une liste chaînée d'entiers non vides.
7. Écrire une fonction existe qui teste si un élément donné existe dans une liste chaînée d'entiers. La fonction doit retourner true si l'élément existe et false sinon.
8. En utilisant les fonctions précédentes, écrire un programme principale qui:
    a. Déclare une liste chainée d'entiers vide li.
    b. Insérer des éléments arbitraires à la liste chaînée li.
    c. Affiche tous les éléments de la liste chaînée li.
    d. Affiche le nombre d'éléments de la liste chaînée li.
    e. Affiche la somme des éléments de la liste chaînée li.
    f. Affiche la valeur du plus petit élément de la liste chaînée li.

Exercice 3

Écrire une fonction link qui crée un lien entre deux listes chaînées d'entiers li1 et li2 non vides. Cette fonction colle le dernier élément de la liste li1 au premier élément de la liste li2.

Exercice 4

1. Écrire une fonction tabToList qui transforme un tableau d'entiers en liste chainée. La fonction doit retourner un pointeur qui indique la tête de la liste.
2. Écrire une fonction listToTab qui transforme une liste chaînée d'entiers en tableau. La fonction doit retourner un tableau qui contient tous les éléments de la liste chaînée.

Exercice 5:

1. Écrire une fonction supprimerListe qui supprime de la mémoire tous les éléments d'une liste chaînée d'entiers.
2. Écrire une version récursive de la fonction supprimerListe.

Code :
https://github.com/hm43/ExercicesTpPASD.git

Série 0.2

3. Écrire une fonction supprimerElement qui supprime la première occurrence d'un élément donnée d'une liste chaînée d'entiers. Si l'élément n'existe pas, la liste reste inchangée.
4. Écrire une version récursive de la fonction supprimerElement.

Exercice 6:

Écrire un programme qui gère une pile à l'aide d'une liste chaînée. Pour cela, vous créerez un type de structure de pile à l'aide d'une liste chaînée dont les éléments sont entiers. Le pointeur d'un élément de la pile pointe vers l'élément suivant.

Le programme se compose de plusieurs fonctions:

1. Une fonction de création créer pile qui retourne une pointeur de type pile, nul.
2. Une fonction vide qui retourne 0 si la pile, passée en paramètre, est non vide, et un nombre différent de 0 dans le cas contraire.
3. Une fonction empiler qui empile l'entier iVal, passé en paramètre, à la pile p, également passée en paramètre.
4. Une fonction depiler qui supprime le premier élément.
5. Une fonction désempiler qui supprime le sommet de la pile p, passée en paramètre. La mémoire occupée par le précédent sommet de la pile est libérée.
6. Une fonction afficher qui affiche le contenu de la pile de manière récursive.
7. Créer un programme de test.

Exercice 7:

1. Même questions de l'exercice 6 en utilisant les Files.