Listes simplement chaînées

Sujet 01 : Etude de la politique de gestion LIFO (Last In - First Out)

L'objectif de ce TD est d'étudier deux solutions d'implémentation du mode de gestion LIFO. La première utilise une structure de donnée statique : un **tableau**. Le seconde s'appuie sur une structure de données dynamique : une **liste simplement chaînée**.

Le principe LIFO est analogue à celui d'une pile d'assiettes :

- dès qu'un élément doit être ajouté, il l'est au sommet de la pile
- dès qu'on souhaite ôter un élément, c'est toujours celui du sommet de la pile qui l'est

On supposera ici que les "assiettes" manipulées (i.e. les informations stockées) sont des valeurs réelles. Ce pourrait bien entendu être des données composites telles ques des étudiants,...

1 Solution à base de tableau

Dans la suite de l'énoncé, `Pile` représente un tableau de valeurs réelles géré en mode LIFO, `Sommet` représente la valeur de l'indice qui repère en permanence le rang de l'élément qui se trouve au sommet de la pile, et `Taille` représente le nombre maximum de valeurs que peut contenir le tableau.

Travail demandé

1. Ecrire la fonction `PileInit(Pile,Taille,Sommet)` qui initialise l'indice `Sommet` (i.e. pas de données dans la pile). Si les fonctions `Empiler`, `Depiler` et `EditerPile` que vous allez devoir coder sont correctes, peu importe ce qui se trouve dans le tableau `Pile` au-delà de l'indice `Sommet`. Vous ne devriez jamais accéder à ces données "non significatives".

2. Ecrire la fonction `PileVide(Pile,Sommet)` qui retourne la valeur `1` si la pile est actuellement vide et la valeur `0` dans le cas contraire.

3. Ecrire la fonction `PilePleine(Pile,Taille,Sommet)` qui retourne la valeur `1` si la pile est actuellement pleine et la valeur `0` dans le cas contraire.

4. Ecrire la fonction `Empiler(Pile,Taille,Sommet,Val)` qui empile, si cela est possible, la valeur `Val` dans `Pile` et met bien entendu à jour `Sommet`.

5. Ecrire la fonction `Depiler(Pile,Sommet,Val)` qui dépile, si cela est possible, la valeur réelle qui se trouve actuellement au sommet de la pile, retourne sa valeur dans `Val` et met à jour `Sommet`.

6. Ecrire la fonction `EditerPile(Pile,Sommet)` qui affiche un par un les éléments de la pile en commençant par le sommet.

7. Ecrire un programme C qui permet à l'utilisateur de simuler toutes ces opérations sur une pile de valeurs réelles en proposant soit d'empiler une valeur qu'il précisera, soit d'extraire une valeur qu'on lui affichera. Un état de la pile sera affiché après chaque opération.

Idée : Ecrire une fonction `menu()` qui affiche ledit menu, effectue la saisie du choix de l'utilisateur et retourne un entier (1,2,...) représentant ce choix. Votre programme principal peut alors effectuer un `switch` sur cette valeur pour exécuter l'action désirée. Là encore vous pouvez écrire une fonction `actionEmpiler` avec les paramètres adéquats qui se chargera de lire la valeur à empiler puis de l'empiler effectivement. Idem pour les autres actions. Cela vous allègera considérablement votre programme principal.

2 Solution à base de liste simplement chaînée

Travail demandé

1. Définir les éléments (structures, pointeurs,...) nécessaires à la manipulation d'une liste simplement chaînée de valeurs réelles. Dans la suite, `Pile` représente l'adresse de l'élément au sommet de la pile, donc en tête des élements de la liste simplement chaînée.
   Rappel : N'hésitez pas à faire de petits croquis pour bien comprendre ce qui se passe (ou devrait se passer...)

2. Ecrire la fonction `PileInit(Pile)` qui initialise `Pile` à une valeur nulle (pointeur `NULL`).

3. Ecrire la fonction `PileVide(Pile)` qui retourne la valeur `1` si la pile est actuellement vide et la valeur `0` dans le cas contraire.

4. Ecrire la fonction `Empiler(Pile,Val)` qui empile la valeur réelle `Val` dans la pile et met à jour `Pile` (cette fonction devra utiliser le mécanisme d'allocation dynamique).

5. Ecrire la fonction `Depiler(Pile,Val)` qui dépile, si cela est possible, la valeur réelle qui se trouve actuellement au sommet de la pile, retourne sa valeur dans `Val` et met à jour `Pile`.

6. Ecrire la fonction `EditerPile(Pile)` qui affiche un par un les éléments de la pile en commençant par le sommet.

7. Ecrire un programme C qui permet à l'utilisateur de simuler toutes ces opérations sur une pile de valeurs réelles en proposant soit d'empiler une valeur qu'il précisera, soit d'extraire une valeur qu'on lui affichera. Un état de la pile sera affiché après chaque opération. Même conseil que dans la première section : fonction `menu()`, etc...

3 Debriefing

Dresser un bilan comparatif (et objectif) des deux méthodes étudiées. Quels sont leurs avantages et inconvénients respectifs ?