TD 8 : Types abstraits de données

Compétences

- Algorithmique sur les listes chainées et piles

La liste chaînée

Introduction

Une liste chaînée est une structure de donnée permettant de stocker de l'information. Contrairement au tableau la liste est de taille variable. Son nom vient du fait qu'elle est similaire à une chaîne composée de maillon : chaque maillon est relié au maillon suivant, de sorte qu'en ne tenant que le premier maillon, on peut accéder à tous les maillons de la chaîne. Par ailleurs, on peut facilement agrandir ou rétrécir la chaîne en lui ajoutant ou retirant des maillons.

Pour les listes chaînées la séquence est mise en oeuvre par le pointeur porté par chaque élément qui indique l'emplacement de l'élément suivant. Le dernier élément de la liste ne pointe sur rien

Spécifications

Soit la classe `liste` permettant d'initialiser une liste. Son interface est la suivante :

— Opérations
  - **isEmpty**()->**bool** : Vrai si la liste est vide
  - **addFirst(element)->list** : Ajoute un élément en tête de liste

— Propriétés
  - **head** : Pointeur sur le premier maillon si non null
  - **next** : Passer au maillon suivant
  - **value** : Obtenir la valeur de l'information contenue dans le maillon.

Pour s'exercer

1. Écrire une fonction `length(L:liste)->int` qui prend en paramètre une liste et qui renvoie le nombre d'éléments de cette liste.
2. Écrire une fonction `Occurrences(L:liste, element)->int` qui renvoie le nombre de fois que **element** est trouvé dans la liste.
3. Écrire une fonction `isSorted(L:liste)->bool` qui vérifie si une liste est triée par valeurs croissantes.
4. Écrire une fonction `deleteFirstOccurrence(L:liste, element)->liste` qui supprime la première occurrence d'un élément donnée d'une liste. Si l'élément n'existe pas, la liste reste inchangée.
5. Écrire une fonction `reverse(L:liste)->liste` qui inverse l'ordre des éléments d'une liste.

D'autres exemples seront étudiés en TD machine.

La pile

Introduction

Une pile sert à stocker de l'information. Son nom vient de la manière particulière dont elle permet d'accéder à cette information. Prenons l'analogie avec une pile d'assiettes. On ne peut poser ou **empiler** une nouvelle assiette que sur le dessus ou **sommet** de la pile, et on ne peut retirer ou **dépiler** uniquement l'assiette qui se trouve au sommet de la pile. Une pile peut bien sûr être vide, ce qui doit pouvoir être vérifié, car cela n'aurait pas de sens de dépiler l'information d'une pile vide.

Spécifications

Soit la classe `Stack` permettant d'initialiser une pile. Son interface est la suivante :

- **isEmpty**() : Vrai si la pile est vide
- **push(element)** : Empile un élément
- **top**() : Affiche le sommet de la pile
- **pop**() : Dépile un élément

Quelques incontournables

1. Écrire une fonction `copy` qui prend en paramètre une pile et renvoie une autre pile contenant les mêmes éléments que la pile de départ et dans le même ordre.
2. Écrire une fonction `reverse` qui prend une pile en paramètre et inverse l'ordre des éléments de cette pile.
3. Écrire une fonction `separate` qui prend en paramètre une pile `p` ne contenant que des nombres entiers et renvoyant une pile des nombres impairs et une pile des nombres pairs, la pile `p` ne doit pas être vide à la fin.
4. Écrire une fonction `bottomTop` qui échange le sommet et le fond de la pile.

Applications

Trier les éléments

Écrire une fonction `sort` qui prend en paramètre une pile A d'un ensemble de nombres entiers et retourne une pile B contenant ces nombres triés avec le minimum au sommet de la pile. Pour effectuer le tri on s'autorise une unique pile temporaire que l'on appelle C.

Texte bien parenthésé

Considérons une chaîne de caractères contenant des parenthèses rondes () et des crochets que l'on pourrait appelés des parenthèses carrées []. À chaque parenthèse ouvrante doit correspondre une parenthèse fermante du même type, et réciproquement. Par ailleurs, si une parenthèse ouvrante est ouverte à l'intérieur d'un autre couple de parenthèses, sa parenthèse fermante doit elle aussi de trouver à l'intérieur du même couple.

1. Écrire une fonction `parentheses(text:str)` qui prend en paramètre une chaîne de caractères uniquement de paranthèses et qui renvoie True si la chaîne est bien parenthésée et False sinon. Attention cette fonction ne traite que les parenthèses rondes.
2. Tester votre fonction sur les chaines "(()())", "())((" et "(()"
3. Modifier la fonction `parentheses` pour une chaîne de caractères pouvant contenir les caractères de "a" à "z".
4. Proposer des chaînes de caractères pour tester votre nouvelle fonction.
5. Effectuer une dernière modification pour tenir compte des deux type de parenthèses rondes et carrées.
6. Tester votre fonction sur les chaînes "z(y[x()w]v)u" et "z(y[x)w)"