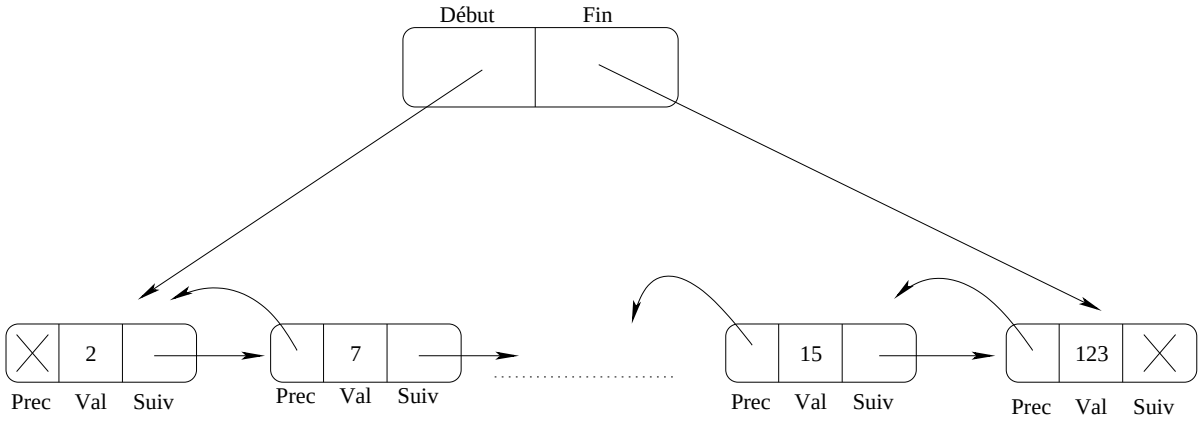
Liste doublement chaînée en dynamique

1 Contexte

Il s'agit d'implanter la structure de données LDCD (Liste_Doublement_Chaînée_Dynamique) d'entiers décrite par le schéma suivant :



Les valeurs sont rangées par ordre croissant

Cette structure de données permet de stocker des entiers par ordre croissant et de les afficher soit par ordre croissant, soit par ordre décroissant.

Vous devrez utiliser le mécanisme de la compilation séparée.

2 Questions à résoudre

1. Proposer en langage C un type de données dynamique pour une LDCD d'entiers.
2. Écrire la fonction INIT_LDCD qui retourne la LDCD vide.
3. Écrire la fonction AFFICHER_CROISSANT_LDCD qui affiche par ordre croissant tous les entiers stockés dans la LDCD donnée en paramètre.
4. Écrire la fonction AFFICHER_DECROISSANT_LDCD qui affiche par ordre décroissant tous les entiers stockés dans la LDCD donnée en paramètre.
5. Écrire la fonction AJOUTER_A_LDCD qui renvoie la LDCD construite en ajoutant l'entier donné en paramètre dans la LDCD donnée en paramètre. L'entier doit être ajouté à la "bonne place", c'est-à-dire en respectant l'ordre.
6. Écrire la fonction SUPPRIMER_A_LDCD qui renvoie la LDCD construite en supprimant l'entier donné en paramètre de la LDCD donnée en paramètre. Si l'entier n'appartient pas à la LDCD, celle-ci sera renvoyée sans modification
7. Écrire la fonction MAP qui prend en paramètre :
   – une fonction f qui prend un entier en paramètre et renvoie un autre entier,
   – et une LDCD,
   et qui renvoie le résultat de l'application de f à chaque élément de la LDCD. Ce résultat est donc forcément sous la forme d'une liste. Vous considérerez que la fonction f est accessible par un pointeur de fonction.

3 Le TAD

Sorte LDCD

Utilise BOOL, INT, FUNC : INT → INT

Constructeurs
init : → LDCD
ajout : LDCD × INT → LDCD

Projecteurs
affCroissant : LDCD →
affDecroissant : LDCD →
taille : LDCD → INT
appartient : LDCD × INT → BOOL
index : LDCD × INT → INT
valeur : LDCD × INT → INT
nbOccurrences : LDCD × INT → INT
supprimer : LDCD × INT → LDCD
map : LDCD × FUNC → LDCD

Préconditions
valeur(l,i) ssi $1 \leq$ i $\leq$ taille(l)
map(l,f) ssi f est monotone (conserve l'ordre)

Axiomes
taille(init()) = 0
taille(ajout(l,x)) = taille(l)+1

appartient(init(),x) = false
appartient(ajout(l,x),y) = (x=y) $\vee$ appartient(l,y)

(index(l,x) = i) $\wedge$
($\neg$ appartient(l,x) $\rightarrow$ (i=0)) $\wedge$
(appartient(l,x) $\rightarrow$
((1 $\leq$ i $\leq$ taille(l)) $\wedge$
($\forall j = 1 \ldots i-1$(valeur(l,j) < valeur(l,i))) $\wedge$
($\forall j = i+1 \ldots$ taille(l)(valeur(l,j) $\geq$ valeur(l,i)))

(valeur(l,i) = x) $\wedge$ (appartient(l,x)) $\wedge$ ($\forall j = 1 \ldots i-1$(valeur(l,j) $\leq$ x)) $\wedge$ ($\forall j = i+1 \ldots$ taille(l)(valeur(l,j) $\geq$ x))

nbOccurrences(init(),x) = 0
(nbOccurrences(ajout(l,x),y) = n) $\wedge$ (x=y $\rightarrow$ n=nbOccurrences(l,y)+1) $\wedge$ (x$\neq$y $\rightarrow$ n=nbOccurrences(l,y))

(supprimer(l,x) = l') $\wedge$
(appartient(l,x) $\rightarrow$
((taille(l')=taille(l)-1) $\wedge$ (nbOccurrences(l',x)=nbOccurrences(l,x)-1)))
$\wedge$ ($\neg$ appartient(l,x) $\rightarrow$ l=l')

(map(l,f) = l') $\wedge$ (taille(l) = taille(l')) $\wedge$ ($\forall i = 1 \ldots$ taille(l)(valeur(l',i) = f(valeur(l,i))))

Remarque importante : pas d'axiome pour les projecteurs d'affichage puisqu'on n'a pas de "sortie" à caractériser.