Programmation : TP 11

Objectifs : [C] Pointeurs et structures.

[C] Listes modifiables

Une liste est une structure de données utilisée pour manipuler une collection d'objets de même type. Contrairement à un tableau, une liste peut contenir un nombre d'éléments variable dynamiquement, et sa taille n'a donc pas besoin d'être connue a priori. L'accès aux éléments d'une liste s'effectue de manière séquentielle, en commençant par le premier élément de la liste puis en parcourant les suivants un à un : on ne peut pas obtenir directement un élément arbitraire de la liste comme on le fait grâce à l'index pour un tableau.

FIG. 1 – Structure de liste

1. La figure **??** illustre une implémentation de listes d'entiers : chaque « cellule » de la liste contient un élément et un pointeur vers l'élément suivant ; la fin de la liste est codée par le pointeur `NULL`. Définir le type `liste` capable de représenter une collection d'entiers (en utilisant un alias de type `typedef`), ainsi que la **constante** `vide` qui représente la liste vide. Écrire les fonctions suivantes :
   – `liste cons(int, liste)` : ajoute un élément en tête d'une liste ;
   – `int est_vide(liste)` : teste si une liste est vide ;
2. Écrire la fonction `void print(liste)` qui affiche les éléments d'une liste.
3. Écrire la fonction `int longueur(liste)` qui calcule le nombre d'éléments d'une liste.
4. Écrire la fonction `void concatenation(liste* l1, liste l2)` qui « ajoute » la liste `l2` à la fin de la liste `l1` **en modifiant** cette dernière. Il faut nécessairement passer **l'adresse** de la liste `l1` en paramètre de `concatenation` pour pouvoir la modifier.
5. Écrire la fonction `void insertion(int e, liste* l)` qui insère un élément à sa place dans une liste triée par ordre croissant. La figure **??** illustre l'insertion de l'entier `10` dans la liste de la figure **??**. Comme pour `concatenation`, `insertion` manipule **l'adresse** de la liste `l` pour pouvoir la modifier.

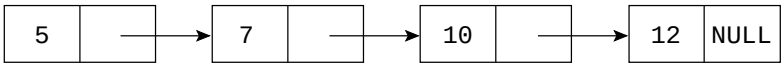

FIG. 2 – Insertion d'un élément dans une liste

6. Écrire une fonction de tri par insertion d'un tableau d'entiers : `int* tri(int* t, int n)`, `n` étant la taille de `t`. Pour réaliser ce tri, on insèrera un à un les éléments du tableau à leur place dans une liste triée (initialement vide), puis on transformera cette liste en un nouveau tableau renvoyé par la fonction.