Algorithmique et Programmation 1
IMAC 1ere année

TP 10

Listes chaînées

Dans cette séance de travaux dirigés, on travaillera sur les listes chaînées.

Pendant le TP, vous écrirez vos codes dans des fichiers source. Ayez le réflexe d'enregistrer régulièrement vos sources. À la fin de la séance, mettez-les dans une archive (commande `tar -zcvf nom_archive fichiers_source`) et rendez-le suivant les instructions données sur e-learning. N'oubliez pas de commenter votre code.

Exercice 1. (Fonctions de base : ajout d'une cellule, affichage, longueur)

1. Définir les types strucutés :

```
1  typedef struct cellule{
2    int valeur;
3    struct cellule *suivante;
4  }Cellule, *Liste;
```

2. Définir une fonction `insereTete(Liste *lst, int n)` qui initialise une nouvelle cellule contenant la valeur `n` et l'insère au début de la liste passée en argument. Pensez à gérer l'erreur éventuelle crée par l'appel à `malloc`.
3. Définir une fonction `afficheListe(Liste lst)` qui affiche les valeurs successives de la liste passée en argument.
4. Définir une fonction `longueurListe(Liste lst)` qui retourne la longueur de la liste passée en argument.

Exercice 2. (Quelques fonctions élémentaires)

1. Définir une fonction `int nbInferieurs(Liste lst, int)` calculant le nombre de valeurs de la liste passée en argument inférieures à `n`.
2. Définir une fonction `recherche(Liste lst, int n)` qui recherche la valeur `n` dans la liste passée en argument. Si `n` est présente dans la liste, la fonction retourne l'adresse de la cellule. Sinon, la fonction retourne `NULL`.
3. Définir une fonction `minimum(Liste lst)` qui retourne l'adresse de la cellule de valeur minimale présente dans la liste passée en argument. Vous pouvez pour cela utiliser la constante `INT_MAX` de `limits.h`.
4. Définir une fonction `estTriee(Liste lst)` testant si les valeurs de la liste passée en argument sont dans l'ordre croissant.

5. Définir une fonction `concatene` qui reçoit deux listes et place les cellules de la deuxième liste à la fin de celles de la première liste. Après l'appel à `concatene`, la deuxième liste doit être vidée.

Exercice 3. (D'autres fonctions)

1. Définir une fonction `extraitListe` qui extrait la cellule de tête de la liste passée en argument.
2. Définir une fonction `insereApres` qui prend en arguments une liste ainsi que deux entiers `n` et `m` et ajoute une cellule contenant la valeur `m` après la cellule contenant la valeur `n`. Si `n` n'est pas présente dans la liste liste passée en argument, alors la nouvelle cellule est ajoutée à la fin de la liste.

Exercice 4. (Un peu de récursivité)

Si vous ne l'avez pas déjà fait, essayez de définir les fonctions précédentes récursivement.