TP2 - Listes simplement chaînées

AL31/CNAM STMN A1, J.-M. ROBERT

Remarque préliminaire :

Chacune des fonctions implantées sera testée depuis une fonction principale (`int main()`). Le travail effectué sera déposé sur le Moodle sous la forme d'une fichier source. Les réponses aux questions posées seront insérées en commentaire dans le code source.

1 Fonctions et procédures utilitaires

1.1 Préliminaires

Dans un fichier source C :

1. Définir la structure de données d'une cellule (maillon) d'une liste simplement chaînée.
2. En utilisant `typedef`, définir un type liste (qui sera un pointeur sur une cellule de tête).
3. Implanter une procédure d'affichage d'une cellule (maillon) de la liste chaînée (prototype `void affiche_cell(cell_t n)`) ;
4. Implanter une procédure d'affichage d'une liste chaînée complète, soient toutes les cellules de la liste (attention à la condition d'arrêt et à la dernière cellule affichée). (prototype `void affiche_list(list_t l)`) ;

1.2 Utilitaires de création/destruction

Dans les listes qui seront créées dans la suite, on se base sur les hypothèses suivantes :

— Les listes créées seront des listes chaînées d'entiers positifs.

— Les listes pointeront vers une cellule de fin de liste, qu'on pourra nommer `end`. Cette cellule aura pour clé la valeur -1, pour la distinguer des autres cellules de la liste. Le pointeur vers l'élément suivant sera initialisé à `NULL`.

1. Implanter une fonction de insertion en tête de la liste chaînée d'un maillon de valeur `elt` (prototype `list_t insert_in_head(int elt, list_t l)`) ; cette fonction retournera la liste (le pointeur vers la nouvelle tête de liste).
2. Implanter une fonction de insertion en queue de la liste chaînée d'un maillon de valeur `elt` (prototype `list_t insert_in_tail(int elt, list_t l)`) ; cette fonction retournera la liste (le pointeur vers la tête de liste).
3. Implanter une fonction de suppression en tête de la liste chaînée (prototype `list_t delete_in_head(list_t l)`) ; cette fonction retournera la liste (le pointeur vers la nouvelle tête de liste).
4. Implanter une fonction de suppression en queue de la liste chaînée (prototype `list_t delete_in_tail(list_t l)`) ; cette fonction retournera la liste (le pointeur vers la tête de liste).

   Pour les fonctions de d'insertion, il faut utiliser la fonction `malloc()` pour allouer la mémoire des nouvelles cellules. Ne pas oublier d'utiliser `free()` pour les fonctions de suppression !

   Vous pourrez tester ces fonctions à partir de la fonction principale `main()`, après avoir au préalable créé le nœud `cell_t *end` et une variable de type `list_t`.

5. Implanter une fonction de création d'une liste chaînée à n éléments
   (prototype `list_t create_list(int n)`) ; cette fonction retournera la liste (le pointeur vers la tête de liste).

   (les clés seront aléatoires comprises entre `0` et `MAX_VAL-1`, macro définie en en-tête du fichier source et on utilisera la fonction `rand()` qui renvoie un entier entre 0 et 2^{31} vue au TP précédent)
6. Implanter une procédure de destruction d'une liste chaînée
   (prototype `void free_list(list_t l)`).

   Ces deux dernières feront usage des premières fonctions créées.

2 Fonctions et procédures de recherche et de tri

2.1 Minimum - Maximum

Implanter les fonctions de recherche dans une liste chaînée de l'élément minimum et l'élément maximum. Les prototypes sont les suivants :

— `cell_t * min_of_list(list_t l)` ;
— `cell_t * max_of_list(list_t l)` ;

Ces deux fonctions renvoient un pointeur vers le maillon de l'élément correspondant dans la liste.

2.2 tri

1. Implanter les fonctions d'échange d'un élément. Les prototypes sont les suivants :
   — avec la tête de liste chaînée `list_t swap_with_head(cell_t * c, list_t l)` ;
   — avec la queue de liste chaînée `list_t swap_with_tail(cell_t * c, list_t l)` ;
   Ces deux fonctions renvoient la liste mise à jour.
2. À l'aide des fonctions précédentes, implanter le tri séléction d'une liste chaînée
   (prototype `list_t selection_sort(list_t l)`)

   Remarque 1. Cette dernière fonction ne réalise aucune allocation nouvelle, puisque l'opération ne fera que changer l'ordre des cellules de la liste.

   À l'aide des fonctions précédentes, implanter le tri rapide d'une liste chaînée
   (prototype `list_t quick_sort(list_t l)`)

Pouvez-vous comparer les performances ?

3 La bataille !

Le but est d'implanter ce célèbre jeu de cartes où la machine jouera contre elle-même.

1. Proposer une strucuture de données adaptées pour représenter :
   — un jeu de 52 cartes ;
   — la main de chaque joueur après une distribution aléatoire ;
2. implanter le jeu de la carte selon les règles bien connues.
   Vous ferez l'implantation de façon à pourvoir visualiser l'enchaînement des coups.

La partie termine-t-elle toujours ?