Enseignants : A. Bonhomme, L. Bougé le 27/11/00
Email : {Alice.Bonhomme,Luc.Bouge}@ens-lyon.fr

TD Programmation 8 : Allocation dynamique

1 Des listes dynamiques

Nous allons reprendre ce qui a été fait pendant le TD6 sur la gestion mémoire pour manipuler des listes. L'objectif est de définir progressivement une allocation aléatoire de la mémoire. Pour ceci, vous vous baserez sur un module de gestion de la mémoire que vous avez déjà fait pendant le TD6.

1.1 Nouvelle définition du type `ilist`

Reprendre un des modules de gestion de la mémoire du TD 6 que vous avez implémenté. Dans ce module, remplacer la définition du type `ilist` par la définition suivante, la définition d'une cellule et de la mémoire reste la même :

```
#define EMPTY NULL

typedef int * ilist;

typedef struct {
    int item;
    ilist next;
/* + autres champs suivant le module de gestion de la mémoire choisi */
} cell;

static cell memory[NB_CELL];
```

La seule différence par rapport au module du TD6 est la manière d'accéder à la cellule mémoire. Pour récupérer l'élément contenu dans la première cellule de la liste `l` (c'est-à-dire `car l`), on utilise `memory[l].item` dans le module du TD6. Maintenant, on utilisera `l->item` ou `(*l).item` (les deux formes sont équivalentes).

- Réécrire la fonction `ilist get_new_list_cell()` avec cette nouvelle définition du type `ilist`.

- Reprendre les fonctions `cons`, `car` et `cdr`. Les autres fonctions (que vous avez du écrire uniquement à l'aide de `cons`, `car`, `cdr` et `empty`), ne changent pas.
- Tester vos fonctions sur le programme de test du TD6.

1.2 Allocation dynamique de la mémoire

On va maintenant allouer dynamiquement la mémoire c'est-à-dire sans passer par un tableau statique.

- Écrire la fonction `ilist get_new_list_cell()` qui alloue dynamiquement une cellule.
- Écrire la fonction `void free_list_cell(ilist l)` qui libère l'espace mémoire utilisé par la cellule de la liste.
- Écrire la fonction `void free_list(ilist l)` qui libère l'espace mémoire utilisé par toute la liste (c'est-à-dire par toutes les cellules de la liste `l`).
- Reprendre le programme de test avec cette nouvelle gestion de la mémoire.
- Écrire une fonction `rplacd(ilist l1, ilist l2)` qui remplace le `cdr` de `l1` par `l2`. La tester sur un exemple.
- Générer deux listes `l1` et `l2`, puis une liste `l3`. Exécuter `rplacd (l1, l3)`, puis `rplacd(l2,l3)`. Libérer la liste `l2`. Que se passe-t'il pour `l1`. Imprimer cette liste.

2 Allocation d'un tableau à deux dimensions

L'objectif de cet exercice est de travailler sur les différentes déclarations possibles d'un tableau à deux dimensions, et ce que cela implique sur la gestion de la mémoire.

Un tableau à deux dimensions (`N` lignes et `M` colonnes) peut être déclaré des trois manières suivantes :

```
#define N 4
#define M 2

/* allocation statique */
int static_tab[N][M];

/* allocation semi-statique */
int * semi_static_tab[N];

/* allocation dynamique */
int ** dynamic_tab;
```

Quelle que soit la déclaration du tableau, les éléments pourront être accèdés par `tab[i][j]`. Toutefois, l'emplacement des éléments du tableau ne seront pas situés au même endroit dans la mémoire suivant les définitions.

2.1 Allocation statique

- Écrire une fonction `init_static_tab` qui initialise tous les éléments du tableau à 0.
- Écrire une fonction `print_static_tab` qui imprime pour chaque case `tab[i][j]` du tableau les informations suivantes : adresse de la case, valeur de `i`, valeur de `j`, valeur de l'élément de la case. (*aide :* utiliser le format `% p` pour afficher une adresse en hexadécimal). Tester cette fonction sur le tableau initialisé.
- On définit une variable `int line[M]`. On souhaite copier les élément de `line` dans une ligne du tableau.
  - Écrire une fonction `copy_line_in_static_tab (int * linesrc, int i)` qui copie la ligne `line` dans la ième ligne du tableau, à l'aide d'une boucle `for`.
  - Initialiser `line` avec des valeurs autres que 0, et tester cette fonction en imprimant ensuite le tableau avec `print_static_tab`.
  - Écrire une fonction `copy_line_bis_in_static_tab( int * line, int i)` qui copie la ligne `line` à l'aide de la fonction `memcpy` (pour la syntaxe, `man memcpy`). Cette fonction permet de copier tout un espace mémoire source vers un espace mémoire cible.
  - Comme précédemment, tester cette fonction.

2.2 Allocation semi-statique

- Écrire une fonction `alloc_semi_static_tab` qui alloue la partie dynamique du tableau.
- Écrire une fonction `free_semi_static_tab` qui libère ce tableau. (**Attention :** *dans vos programme, il faut toujours libérer les données que vous allouez!!)*
- Écrire une fonction `init_semi_static_tab` qui initialise le tableau à 0, puis la fonction `print_semi_static_tab`. Pourquoi ne peut-on pas utiliser la même fonction que pour `static_tab` en passant le tableau par paramètre ? (si vous ne voyez pas, passer à la question suivante, elle vous aidera à comprendre).
- Initialiser le tableau, puis imprimer le. Que remarquez-vous ? Comparez par rapport aux résultats obtenus pour `static_tab`.
- Écrire une fonction `replace_line_in_semi_static_tab (int * line, int i)` qui remplace **physiquement** la ligne `i` du tableau par celle passer en paramètre.
- Tester cette fonction avec une ligne non nulle. Imprimer le tableau ainsi modifié. Que remarquez vous ?
- Modifier la ligne `line` et imprimer le tableau. Que remarquez-vous ? est-ce normal ?

2.3 Allocation dynamique

- Écrire une fonction d'allocation du tableau `alloc_dynamic_tab`.
- Écrire une fonction de libération du tableau `free_dynamic_tab`.

- Reprendre les fonctions écrites pour `semi_static_tab` en passant le tableau par paramètre et pour que ces fonctions soient utilisables avec `dynamic_tab`. Pourquoi ceci est-il possible ? Quelle est alors la différence entre ces deux déclarations ?
- Tester ces nouvelles fonctions avec les deux tableaux.

2.4 Pour aller plus loin

On souhaite maintenant allouer dynamiquement un tableau à deux dimensions, mais avec un espace mémoire contigü, ce qui n'est le cas dans aucune des déclaration vues précédemment.

Comment déclarer le tableau ? Comment l'allouer ? Comment accéder aux données ?

Proposer une solution et implémenter les fonctions vues précédemment.