TD Méthodologie de la Programmation
Feuille d'exercices n°4

`Sebastien.Varrette@imag.fr` (Bureau BR.4.07)

1 Gestion de tableaux dynamiques d'entiers

Le but de cet exercice est la construction d'un programme `tab_management.c` qui contiendra un ensemble de fonctions permettant de manipuler des tableaux de n entiers. Pour cela, on demande de réaliser les fonctions suivantes, définies par leurs prototypes :

- `int * init_tab(int n, int min, int max) ;`
  Cette fontion alloue un tableau de `n` entiers, remplit le tableau avec des nombres aléatoires compris entre `min` et `max` et renvoie un pointeur vers le premier élément du tableau alloué (c'est à dire le tableau lui-même).
  Indications : j'insiste sur l'expression "allouer" le tableau :-) Pour le caractère aléatoire, on utilisera les fonctions fournies en annexe.
- `void free_tab(int tab[]) ;`
  libère l'espace mémoire alloué pour stocker le tableau `tab`.
- `void print_tab(int tab[], int n) ;`
  affiche à l'écran le contenu du tableau `tab` de taille `n`.
- `int getMin(int tab[], int n) ;`
  renvoit l'élement minimum du tableau `tab` de taille `n`.
- `int getIndiceMin(int tab[], int n) ;`
  renvoit l'indice dans le tableau de l'élement minimum (si plusieurs éléments du tableau sont minimaux, on renverra le plus petit indice d'entre eux).
- `int getMax(int tab[], int n) ;`
- `int getIndiceMax(int tab[], int n) ;`
- `void echange(int tab[], int n, int i1, int i2) ;`
  échange les éléments d'indices `i1` et `i2` dans le tableau `tab`.
- `void decalage(int tab[], int n, int i) ;` décale les éléments du tableau de `i` indices (`i` entier relatif).
  Exemple : si tab est le tableau `(2, 6, 3, 9)`, alors après un décalage de `i=-1`, tab contient `(6, 3, 9, 2)`. Après un décalage de `i=2` par rapport à la configuration initiale, tab contient `(3, 9, 2, 6)`
- `int sum_tab(int tab[], int n) ;`
  Calcul de la somme des éléments du tableau `tab`.
- `double moy_tab(int tab[], int n) ;`
  Calcul de la moyenne des éléments du tableau `tab`.
- `void tri_insertion(int tab[], int n) ;`
  Tri les éléments du tableau dans l'ordre croissant par la méthode de tri par insertion. Une métaphore fréquemment utilisée pour le tri par insertion est le tri d'un jeu de cartes : en effet, comme pour un jeu de cartes, on trie d'abord les premiers éléments, puis, pour chaque élément suivant, on regarde où il doit être placé dans la partie du tableau déjà triée.

D'une manière un peu plus formelle, le principe est le suivant : Pour chaque élément $i \in [1, n-1]$, insérer i au bon endroit dans `tab[0..i-1]`[1] qui est un tableau déjà trié.
Exemple : `tab = (14, 6, 3, 9)`.

1. `tab[0..0]=(14)` est déjà trié.
2. `tab[1]=6` et on insère cet élément au bon endroit dans `tab[0..0]=(14)`. on obtient `tab[0..1]=(6,14)`.
3. `tab[2]=3` et on insère cet élément au bon endroit dans `tab[0..1]=(6,14)`. on obtient `tab[0..2]=(3,6,14)`.
4. `tab[3]=9` et on insère cet élément au bon endroit dans `tab[0..2]=(3,6,14)`. on obtient `tab[0..3]=(3,6,9,14)` et on a fini.

Pour insérer un élément `tab[i]` dans le tableau déjà trié `tab[0..i-1]`, une façon de faire consiste à trouver l'indice j dans le tableau `tab[0..i-1]` où `e=tab[i]` devrait être placer. Ensuite, il suffit de déplacer les éléments de `tab[j..i-1]` dans `tab[j+1..i]`. il ne reste plus qu'à placer e en `tab[j]`.

Au final, la fonction `main` devra être la suivante :

```
#define MIN 0
#define MAX 10
int main() {
    int i, *tab = NULL;
    int n = ask_n(); // demande à l'utilisateur un entier n
    init_rand();     // initialise le générateur aléatoire
    tab = init_tab(n, MIN, MAX); //initialise le tableau
    print_tab(tab, n);
    printf("Elément min: tab[%i]=%i\n",getIndiceMin(tab, n),getMin(tab, n));
    printf("Elément max: tab[%i]=%i\n",getIndiceMax(tab, n),getMax(tab, n));
    printf("Somme des éléments du tableau: %i\n",sum_tab(tab, n));
    printf("Moyenne des éléments du tableau: %.2f\n",moy_tab(tab, n));
    for (i=0; i<n; i++) { //test des échanges
        printf("Echange des éléments d'indices 0 et i");
        echange(tab, n, 0, i);
        print_tab(tab, n);
    }
    printf("Test décalage vers la gauche: ");
    decalage(tab, n, -2);
    print_tab(tab, n);
    printf("Test décalage vers la droite: ");
    decalage(tab, n, 2); //on doit retomber sur nos pieds
    print_tab(tab, n);
    printf("Test tri par insertion du tableau: ");
    tri_insertion(tab, n);
    print_tab(tab, n);
    free_tab(tab);
```

[1] `tab[0..i-1]` est la notation utilisée pour le tableau de taille i constitué des éléments d'indices 0 à $i-1$

```
    return 0;
}
```

2 Annexe

```
#include <stdlib.h>   //see also 'man 3 rand'
#include <time.h>
/**
 * initialize random generator
 */
void init_rand(){ srand(time(NULL)); }
/**
 * return a random number between 1 and m
 * @param m    upper bound
 */
unsigned long myRand(unsigned long m){
    return 1+(unsigned long)(((double) m)*rand()/(RAND_MAX+1.0));
}
```