Programmation séquentielle - Travail pratique sur les doubles pointeurs

1 Préambule

Ce travail pratique est organisé en deux parties traitant des notions de listes et de doubles pointeurs. Les parties à implémenter sont les suivantes

1. Une liste circulaire simplement chaînée avec une application au problème de Josèphe (permutation d'une liste de nombres).
2. Un tri sans déplacer les données grâce à un tableau de pointeurs sur ces données, en pointant sur celles-ci par ordre croissant.

2 Listes simplement chainées

2.1 Buts

— Approfondissement du fonctionnement des pointeurs en C.
— Mise en œuvre des listes circulaires simplement chaînées en C.
— Application au problème de Josèphe.

2.2 Enoncé

Écrire des fichiers `circ_list.h` et `circ_list.c` pour la manipulation de listes circulaires simplement chaînées. Ceci sera ensuite utilisé pour résoudre le problème de Josèphe.

2.3 Cahier des charges pour les listes circulaires

Pour manipuler des listes circulaires, vous devrez implémenter les fonctions suivantes

— Crée une nouvelle liste vide. Renvoie simplement un pointeur `NULL`
```
element* list_create();
```
— Teste si la liste est vide
```
bool list_empty(element* head);
```
— Renvoie le nombre d'éléments de la liste
```
unsigned int list_count(element* head);
```
— Déplace le pointeur head sur l'élément suivant. Retourne la nouvelle tête de liste

```
element* list_move(element* head);
```

— Insère un élément après le pointeur head. Retourne le pointeur sur l'élément inséré
```
element* list_insert_after(element* head, int data);
```

— Insère un élément avant le pointeur head. Retourne le pointeur sur l'élément inséré
```
element* list_insert_before(element* head, int data);
```

— Recherche un élément dans la liste. Retourne un pointeur sur le 1er élément trouvé ou `NULL` si l'élément est absent
```
element* list_search(element* head, int data);
```

— Supprime un élément de la liste sans libérer l'élément pointé. Renvoie un pointeur sur le 1er élément trouvé à supprimer et l'enlève de la liste, ou `NULL` si l'élément est absent
```
element* list_remove(element** head, int data);
```

— Supprime toute la liste en libérant chaque élément
```
void list_free(element** head);
```

Le fichier `circ_list.h` ne contiendra que les prototypes de fonctions donnés ci-dessus. Le fichier `circ_list.c` peut bien sûr contenir d'autres fonctions utilitaires. Les éléments de la liste devront être définis en C de la manière suivante

```
typedef struct element {
    int data;
    struct element* next;
} element;
```

Écrire un programme de test utilisant vos fichiers sur les listes qui imprime notamment la liste.

2.3.1 Bonus

Quand vous avez fini tout ce qui précède vous pouvez implémenter la fonction suivante.

— Appel d'une fonction qui modifie tous les éléments de la liste grâce à la fonction `action()`
```
void list_process(element* head, int (*action)(int));
```

Comme vous pouvez le voir cette syntaxe est un peu étrange. `list_process()` prend en argument une *fonction* `action()` qui prend un `int` en argument et retourne un `int` (c'est donc une fonction qui prend une fonction en argument). Quelques informations sur ce type de fonctions sur ce lien.

2.4 Le problème de Josèphe

Le problème de Josèphe [1] sert à trouver une permutation de nombres en utilisant une liste circulaire (un anneau). Dans cette liste circulaire, tous les nombres

1. Pour plus d'informations sur le problème de Josèphe https://fr.wikipedia.org/wiki/Probl%C3%A8me_de_Jos%C3%A8phe

entiers compris entre `1` et `n` sont initialement introduits dans le sens horaire. Puis, dans une deuxième phase, les `n` nombres seront éliminés de cette liste comme suit: en commençant à partir de l'élément contenant `n`, on supprime successivement tous les `k`-ème éléments de la liste, en effectuant un parcours circulaire dans la liste; on répète ceci jusqu'à ce que tous les éléments aient été supprimés.

Exemple : si `n = 8` et `k = 3`, la suppression des nœuds contenant les huit entiers se fait dans cet ordre : `3, 6, 1, 5, 2, 8, 4, 7`.

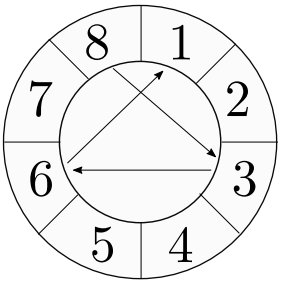

FIGURE 1 – L'ordre de parcours pour `n=8` et `k=3`.

2.5 Cahier des charges pour le problème de Josèphe

Votre programme `josephe.c` doit utiliser la structure de liste circulaire simplement chaînée définie précédemment et être bien modulaire. Il comprendra au moins

- une fonction pour insérer un nombre dans une liste circulaire ; celle-ci sera appelée n fois pour initialiser le problème de Josèphe ;
- une fonction pour supprimer le `k`-ème élément dans la liste circulaire à partir du pointeur d'accès à la liste.

Les éléments extraits doivent être affichés au fur et à mesure de leur suppression dans la liste circulaire. Le lancement du programme avec la valeur de `n` suivie de `k` sur la ligne de commande doit afficher la liste des éléments extraits avec un nombre par ligne.

Par exemple

```
./josephe 8 3
3
6
1
5
2
8
4
7
```

3 Tri d'un tableau par pointeurs

3.1 But

Utilisation de pointeurs pour trier un tableau en `C` sans déplacer le contenu.

3.2 Énoncé

On souhaite trier un tableau d'entiers, nommé `tab` sans déplacer les données. Pour cela, on utilise un deuxième tableau de pointeurs `ptr` qui pointent sur les éléments du tableau d'entiers. Initialement, le pointeur `ptr[i]` pointe sur `tab[i]`. A la fin du tri, le tableau de pointeurs pointe sur les entiers dans l'ordre croissant, c'est-à-dire `ptr[0]` pointe sur le plus petit élément de `tab`, `ptr[1]` pointe sur le deuxième plus petit, etc (voir les fig. 2 et fig. 3).

Situation initiale

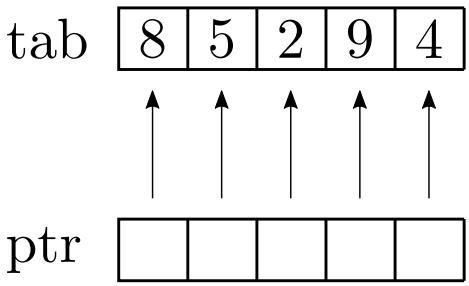

FIGURE 2 – Le tableau `tab` et `ptr` dans leur état initial. Les éléments de `ptr` pointent sur l'élément correspondant à leur indice dans `tab`.

Situation finale

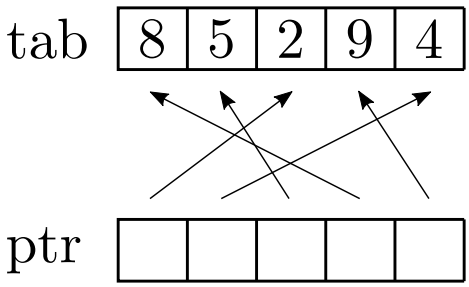

FIGURE 3 – Le tableau `tab` et `ptr` dans leur état final. Les éléments de `ptr` pointent sur les éléments de `tab` dans l'ordre croissant. `tab` n'a pas été modifié.

3.3 Cahier des charges

— Écrire un programme `pointer_sort.c`. Celui-ci contiendra une fonction de tri qui prend en paramètre un tableau d'entiers et retourne un tableau de pointeurs sur ces entiers. Ces pointeurs devront pointer sur les éléments du tableau d'entiers par ordre croissant. Utiliser le tri à bulle ou un tri de votre choix pour réaliser cette fonction.

— Le lancement du programme avec une liste quelconque de nombres entiers sur la
ligne de commande doit afficher la liste triée avec un nombre par ligne.

Par exemple

```
./pointer_sort 18 2 34 21 7 4
2
4
7
18
21
34
```