CCI - Principes des langages de programmation
TD 6 : Structures avancées, pointeurs et tri

Nicolas Bredeche[*], Thomas Tang[†]

12 octobre 2006

1 Pour bien assimiler...

1.1 Tableaux et Pointeurs

Exercice 1 *Soit le programme suivant :*

```
0     int main(){
1       int i = 1 ;
2       int * pt ;
3       pt = malloc (sizeof(int));
4       if (pt != NULL){
5           *pt = i ;
6           printf("i = %i\t*pt = %i\n",i,*pt) ;
7           *pt = 2 ;
8           printf("i = %i\t*pt = %i\n",i,*pt) ;
9           pt = &i ;
10          *pt = 3 ;
11          printf("i = %i\t*pt = %i\n",i,*pt) ;
12          exit(0) ;
13      }
14      else{
15          printf("Probl\‘{e}me d’allocation de la m\’{e}moire\n");
16          exit(-1);
17      }
18    }
19
```

La valeur affichée de i change-t-elle lors du deuxième affichage ? Et lors du troisième affichage ? Pourquoi ?

En langage C, un tableau est en fait un pointeur sur le premier élément du tableau, l'accès au *i*-ième élément se fait en ajoutant *i* à l'adresse de ce pointeur. Notez que les chaînes de caractères sont en fait des tableaux (une chaîne définit avec `char *` est en fait un pointeur sur le premier caractère de la chaine.

On peut se déplacer dans un tableau à l'aide de l'addition sur les pointeurs. Concrètement, si `tab` est un tableau d'entiers, l'expression `*(tab + i)` est équivalente à `tab[i]`

Exercice 2 *À l'aide de l'addition sur les pointeurs, utilisez un pointeur pour remplir et afficher un tableau des N premiers entiers, avec N demandé à l'utilisateur. Puis remplir et afficher un tableau de N caractères avec la lettre 'i' (resp. 'p') pour toutes les positions impaires (resp. paires).*

[*]bredeche@lri.fr
[†]ttang@club-internet.fr

Exercice 3 *Écrire un programme qui à l'aide d'une boucle affiche tous les caractères de 0 à 255 ainsi que leur numéro.*

Exercice 4 *Remplir un tableau avec l'alphabet en minuscules (utilisez l'affichage de l'exercice précédent pour connaître les caractères correspondants). Passer ensuite tous les caractères en majuscules.*

1.2 Tris

Exercice 5 *Dans cette exercice on se propose de trier un tableau à l'aide de deux pointeurs. Le but est de classer les entiers contenu dans le tableau par ordre croissant. On va trier le tableau par la fin, et pour ce faire, on a besoin d'un pointeur sur le dernier élément du tableau (*`dernier = tab + (taille - 1)`*). On commence par chercher le plus grand élément du tableau et on l'échange avec le dernier élément, puis on recommence en décalant le dernier élément (*`dernier = dernier - 1`*) jusqu'à ce que le tableau soit trié.*

Cet exercice sera fait en utilisant d'une part la notation tab[i], et d'autre part avec des pointeurs.

1.3 Structures et structures récursives

Exercice 6 *Vous définirez deux structures* `personne` *et* `adresse` *liées entre elles (une personne habite à une adresse.) Votre structure personne contiendra également les liens de filiations (père et mère.) Dans votre programme vous créerez 5 personnes dont 3 habitent à la même adresse et 2 ont les mêmes parents. Utilisez des pointeurs dans vos structures pour qu'une même adresse n'existe pas plusieurs fois en mémoire. Prévoir une fonction qui affiche une personne et l'appeler pour toutes les personnes définies.*

2 Pour bien approfondir...

Exercice 7 *Algorithme de classement des valeurs d'un tableau dans l'ordre croissant par recherche de minimum (tri par extraction)*

1. *Etat initial du tableau, recherche sur l'ensemble du tableau de la plus petite valeur et permutation avec le premier élément.*
2. *Le traitement précédent est de nouveau appliqué mais sur le tableau réduit du premier élément déjà placé.*
3. *Le traitement est identique avec à chaque permutation le tableau réduit d'un élément. le traitment s'arrête lorsque le nombre d'éléments restant à classer est égal à 1.*

Commencez à trier à la main la liste suivant en respectant les règles de l'algorithme : (5,4,7,10,1)
Ecrivez une fonction ***void tri_extraction (int *tab)*** *qui réalise ce tri.*

Exercice 8 *Algorithme de classement des valeurs d'un tableau par permutation et retour en arrière (tri par bulles ou par échange)*

1. *Comparer deux éléments voisins*
2. *Si les éléments sont classés, passage à l'élément suivant (sauf au bas du tableau)*
3. *Si les éléments ne sont pas classés, permutez les deux éléments, et revenez vers l'élément précédent (sauf en haut du tableau où l'on passe à l'élément suivant)*
4. *Le traitement s'arrête quand la comparaison des deux éléments du bas de tableau n'entraîne pas de permutation.*

Commencez à trier à la main la liste suivant en respectant les règles de l'algorithme : (5,4,7,10,1)
Ecrivez une fonction ***void tri_bulles (int *tab)*** *qui réalise ce tri.*

Exercice 9 *Listes chaînées*

Une liste chainée est formée de cellules liées les unes aux autres par des pointeurs. On a une cellule entête et une cellule queue. Chaque cellule de la liste est de type :

```
0  struct cellule{
1      int id;                 //information correspondant \‘{a} la cellule
2      struct cellule *suivant //pointeur vers la cellule suivante de
3      la liste
4  };
```

Créez une liste chaînée de 10 éléments, où l'entier de chaque cellule est demandé à l'utilisateur. Affichez ensuite cette chaîne.