CCI - Principes des langages de programmation
TD 6 : Pointeurs et tableaux, structures et listes chaînées

François Yvon[*], 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).*

Exercice 3 *Ecrire un programme qui donne le min et le max d'un tableau d'entiers à l'aide de pointeurs.*

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

Exercice 4 *Écrire un programme qui à l'aide d'une boucle affiche tous les caractères de 0 à 255 ainsi que leur numéro.*

Exercice 5 *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.*

2 Les structures

Les entiers et les flottants ne sont pas les seuls types de données disponibles, dans la suite nous verrons d'ailleurs d'autres types de base. Mais ces types de base ne sont pas toujours suffisants, on peut avoir besoin de types plus évolués que l'on nomme en général types structurés, ces types peuvent être définis par l'utilisateur en fonction de ses besoins.

2.1 Définitions

Nous allons maintenant voir les structures (il existe d'autre types structurés comme les tableaux et les unions.) Une structure est le regroupement au sein d'une même valeur de plusieurs autres valeurs auxquelles on affecte un nom. Voici un exemple :

```
{
    champ1 = 1 ;
    champ2 = "une chaine" ;
    champ3 = 1.5 ;
}
```

Pour pouvoir déclarer une structure il faut tout d'abord définir la forme structure à l'aide du mot clef `struct`, par exemple :

```
struct exemple {
    int champ1;
    char * champ2;
    float champ3;
};
```

ATTENTION ! ne pas oublier le **;** après l'accolade fermant la structure.

La définition des structures se fait en général au même niveau que les variables globales pour que celle-ci soit accessible dans tout le programme. On pourra après définir des variables pouvant contenir une structure de cette forme toujours grâce au mot clef structure :

```
/* la variable s peut contenir une structure de forme exemple */
struct exemple s ;
```

Ecrire la définition de la forme d'une structure permettant de représenter un point sur le plan avec les champs x et y de type float. Vous donnerez comme nom à cette forme de structure point.

2.2 Utilisation

Pour accéder aux champs d'une structure on utilise le point **.** :

```
int x ; struct exemple s ; x = s.champ1 ;
```

On peut utiliser le champ d'une structure comme une autre variable :

```
struct exemple s ; s.champ1 = 1 ;
```

Ecrire un programme qui utilise la structure de point définie précédemment, déclarer une variable origine correspondant au point (0, 0). Votre programme devra afficher la valeur des champs de la variable origine.

Exercice 6 *: Structure et vecteurs*
Modifier le programme précédant permettant à partir des coordonnées de l'espace (x,y,z) de 2 points rentrées par l'utilisateur d'afficher en sortie :

- *les coordonnées des deux points rentrées*
- *la distance séparant les deux points*
- *les coordonnées du vecteur*

2.3 Structures et structures récursives

Exercice 7 *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.*

2.4 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 à la cellule
2      struct cellule *suivant //pointeur vers la cellule suivante de
3      la liste
4  };
```

Exercice 8 *Créez une liste chaînée de 10 éléments, où l'entier de chaque cellule est demandé à l'utilisateur. Affichez ensuite cette chaîne.*

3 Pour bien approfondir...

Exercice 9 *Fonctions de string.h*

- *Créer une fonction pour concaténer deux chaînes.*
- *Créer une fonction pour inverser une chaine.*
- *Créer une fonction pour chercher une sous-chaine donnée par l'utilisateur.*

Exercice 10 *: Structure et vecteurs (suite)*
Modifier le programme sur les vecteurs de façon à pouvoir rentrer quatre points A, B, C, D (toujours de l'espace). Calculer les produits scalaire et vectoriel des deux vecteurs ainsi formés : u=vecteur(AB), v=vecteur (CD)

Exercice 11 *Fonctions sur les listes chaînées*

- *Créez deux listes chaînées de 10 éléments chacune.*
- *Créer une fonction permettant de concaténez deux listes chainées.*
- *Créer une fonction permettant d'effacer un élément d'une liste chainée en donnant sa position. Testez cette fonction en effaçant le 3ème, le dernier et le premier de la liste.*