1 Sans Ordinateurs !

1.1 Explications

Il est possible en C de déclarer en une seule instruction plusieurs variables de même type. Ces variables seront rangées dans la mémoire les unes à la suite des autres : c'est ce qu'on appelle un tableau. Exemple avec un tableau d'entiers :

```c
int tab[3]; /* reserve sur la pile trois entiers */
tab[0] = 1; /* le premier s'appelle tab[0] */
tab[1] = 2; /* le deuxieme s'appelle tab[1] */
tab[2] = 3; /* le troisieme s'appelle tab[2] */
tab[3] = 4; /* ATTENTION : pas de controle de debordement !*/

/* il est possible de declarer et definir en une seule instruction :*/
int tab[]={1,2,3};
/* il devient optionnel de mettre la taille dans les [] car le
compilateur peut compter les elements presents entre les accolades */
```

Il est possible de passer les tableaux en paramètre lors de l'appel de fonction. Dans ce cas attention, il faut deux variables : une variable qui identifie le début du tableau, et une variable qui donne la taille du tableau (puisque pas de controle de débordement). Attention, le type de la variable `tab` n'est pas un type "normal", c'est un *pointeur*. C'est en fait une valeur entière contenant l'adresse mémoire du tableau. Si la valeur pointée est de type `type`, le type du pointeur est `type*`. Exemple avec un tableau de caractères (les deux fonctions sont strictement équivalentes) :

```c
void affiche_tableau(char tab[], int taille)
/* remarquez qu'il n'y a rien entre les [] */
{
        int i;
        for(i=0;i<taille;i++)
                printf("%d <-> %c\n", tab[i], tab[i]);
}
void affiche_tableau_2(char* tab, int taille)
{
        int i;
        for(i=0;i<taille;i++)
                printf("%d <-> %c\n", tab[i], tab[i]);
}
```

Le pointeur ne donnant comme information que le début du tableau, il est toujours nécessaire d'utiliser un deuxième paramètre pour la taille du tableau.

Exception a cette règle : si par convention le tableau contient un marqueur de fin. Par exemple, si on manipule des tableaux d'entiers positifs, il est possible d'utiliser une valeur négative pour marquer la fin du tableau :

```c
/* Parametre : tableau contenant comme marqueur de fin la valeur -1 */
void affiche_tableau(int tab[])
{
        int i;
        for(i=0;tab[i]!=-1;i++)
                printf("%d ", tab[i]);
        printf("\n");
}
```

Cela ne marche évidemment que si la convention est respectée : si on écrit du code qui appelle la fonction avec un tableau ne contenant pas -1, la fonction risque de rentrer dans une boucle infinie et de créer un débordement mémoire.

Par convention, en C, une **chaine de caractères** est un tableau de caractères contenant comme marqueur de fin le caractère $'\backslash 0'$. C'est ce qui permet par exemple à printf de les afficher (format %s) sans que l'on passe la taille en paramètre.

1.2 Questions

1. Code 1 (rappel)

```c
#include <stdio.h>

void f(int a)
{
        a=10;
}

int main()
{
        int a=20;
        f(a);
        printf("%d\n",a);
        return 0;
}
```

Qu'affiche ce programme ? Que faut il modifier si l'on veut que l'appel de la fonction ait un effet sur l'affichage du main ligne 12 ? Dessinez la pile.

2. Code 2

```c
#include <stdio.h>
void affiche_tableau(int tab[], int taille)
{
        int i;
        for(i=0;i<taille;i++)
                printf("%d␣",tab[i]);
        printf("\n");
}
void f(int a[])
{
        a[0]=11;
        a[1]=12;
        a[2]=13;
}
int main()
{
        int a[]={1,2,3};
        f(a);
        affiche_tableau(a,3);
        return 0;
}
```

Qu'affiche ce programme ? Dessinez la pile.

3. Code 3

```c
#include <stdio.h>
void f(int a[])
{
        a[0]=10;
}
int main()
{
        int a=1;
        f(&a);
        printf("%d\n",a);
        return 0;
}
```

Qu'affiche ce programme ? Dessinez la pile.

2 1 étudiant par poste (si possible)

Commencez par recopier et tester les codes précédents pour vérifier vos intuitions. Pour la suite, vous écrirez une fonction main() unique que vous ferrez évoluer pour tester les différentes fonctions demandées.

1. ecrire une fonction qui affiche un tableau de caractères passé en paramètre à l'envers
2. ecrire une fonction qui affiche une chaine de caractères passée en paramètre à l'envers.
3. combien de fois votre fonction précédante a-t-elle parcouru la chaine ? si necessaire réécrivez la fonction pour que la chaine ne soit parcourue qu'une seule fois (pensez à la récursivité).
4. ecrire une fonction qui prend en paramètre un tableau de réels et retourne la moyenne des éléments présents dedans.
5. ecrire une fonction qui prend en paramètre un tableau d'entiers et multiplie tous ses éléments par deux.
6. ecrire une fonction qui prend en paramètre une chaine de caractères et change ses éléments en majuscules.
7. testez la fonction précédante avec le main suivant :

```c
int main()
{
        char chaine1[] = {'a','b','c','\0'};
        char* chaine2 = "coucou";
        min_to_maj(chaine1);
        printf("chaine 1 : %s\n",chaine1);
        min_to_maj(chaine2);
        printf("chaine 2 : %s\n",chaine2);
        return 0;
}
```

3 Bonus

Attention : sizeof ne sert **pas** et ne doit **pas** servir à calculer la taille d'un tableau. Exemple à tester pour vous convaincre :

```c
#include <stdio.h>
void f(int tab[])
{
        printf("taille (dans la fonction) : %lu\n",sizeof(tab));
}
int main()
{
        int tab[]={1,2,3};
        printf("taille (dans le main) : %lu\n",sizeof(tab));
        f(tab);
        return 0;
}
```