Travaux pratiques n°4 (2h): Les tableaux

Objectifs

- Déclarer et utiliser des tableaux en langage C
- Approfondir l'utilisation d'itératives

Partie 1: Compléments pratiques au langage C

Le préprocesseur du C:

Nous avons vu en cours que la directive *define* permet de déclarer des constantes:

```
#define MAX 10
#define AGECADET  12
#define AGEMINIME  10
#define AGEPUPILLE  8
#define AGEPOUSSIN  6
```

Ces directives ne sont pas des instructions, les constantes ainsi définies seront remplacé par leur valeur à la compilation par le pré-compilateur.

Ainsi considérons l'extrait de programme suivant:

```
if (age >= AGECADET)
        printf("Catégorie cadet") ;
else
        if (age >= AGEMINIME)
                printf ("Catégorie minime") ;
        else
                if (age >= AGEPUPILLE)
                        printf("Catégorie pupille") ;
                else
                        if (age >= AGEPOUSSIN)
                                printf("Catégorie poussin") ;
```

à la compilation le pré-compilateur remplacera les constantes par leur valeur avant la traduction ce qui donnera après pré-compilation et avant traduction:

```
if (age >= 12)
        printf("Catégorie cadet") ;
else
        if (age >= 10)
```

```
                printf ("Catégorie minime") ;
        else
                if (age >= 8)
                        printf("Catégorie pupille") ;
                else
                        if (age >= 6)
                                printf("Catégorie poussin") ;
```

La directive permet ainsi de paramétrer son programme à moindre coût (pas de place réservée pour les constantes).

Pour les tableaux elle est utile pour définir les tailles:

```
#define MAX 10
...
int tab[MAX];
```

Pas de test de limite sur les tableaux en C:

Comme on vous l'a indiqué en cours, à l'exécution, il n'y a aucun test de dépassement de tableau. Pour vous en convaincre, tester le programme suivant:

```
#include <stdio.h>
int main(void)
{
 int tab1[10] = {1,2,3,4,5,6,7,8,9,10};
 int tab2[10] = {20,21,22,23,24,25,26,17,28,29};
 int i;
 for (i=0; i < 20; i++)
  printf("\n element %d de tab2 %d", i,tab2[i]);
 return (0);
}
```

Question:

Qu'en déduisez vous?

Rangement d'un tableau à N dimensions en mémoire :

La vision multidimensionnelle d'un tableau n'est dû qu'à un découpage logique. Physiquement les éléments d'un tableau à plusieurs dimensions sont rangés linéairement en mémoire (les uns derrière les autres). Pour vous en convaincre testez le petit programme suivant:

```
#include <stdio.h>
int main(void)
```

```
{
 int tab[4][3] = {{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
 int i;
 for (i=0; i < 12; i++)
      printf("\n element %d de tab1 %d", i, tab[0][i]);
 return (0);
}
```

Question:

Qu'en déduisez vous?

Partie 2: Les tableaux en langage C

Exercice 1

Écrire un programme qui propose à l'utilisateur de faire les actions proposées dans le menu suivant:

0- Initialiser le tableau de notes
1- Afficher le tableau de notes
2- Calculer la moyenne des notes du tableau
3- Donner la meilleure note du tableau
4- Rechercher une note dans le tableau
5- Rechercher le nombre d'occurrences d'une note dans le tableau
6- Sortir

Exercice 2: Palindrome ...

Soit un tableau *tabCar* de 30 caractères initialisé (partiellement ou totalement). Vérifier le tableau contient un palindrome, c'est à dire que les caractères du tableau se lisent à l'identique de droite à gauche ou de gauche à droite.
Par exemple:

radar, rotor sont des palindromes

Modifier le programme précédent afin de détecter des palindromes de phrase.
Par exemple:

et la marine va venir a malte
esope reste ici et se repose

sont des phrases palindromes.

Exercice 3: Recherche d'une séquence.

Soit deux tableaux, *tab* de 20 entiers initialisé complètement à la déclaration et *tabSeq* de 5 entiers initialisé complètement par l'utilisateur. Écrire un programme, qui recherche si la séquence d'entiers de *tabSeq* est incluse dans le tableau *tab* (attention aux cas limites).
Par exemple:
si tab={23,45,67,89,8,90,91,4,6,89,56,13,24,53,76,43,90,890,8,0}
et tabSeq={89,56,13,24,53}

La réponse sera « oui »

Exercice 4: la diagonale

Soit un tableau à deux dimensions de 5 par 5 entiers complètement initialisé, écrire un programme qui teste si tous les nombres des deux diagonales sont identiques:
Par exemple pour:

6 2 3 5 6
4 6 2 6 1
1 3 6 7 9
1 6 3 6 8
6 0 1 4 6

Le programme affichera « OUI »

Exercice 5: Échange de triangles

Écrire un programme qui échange le triangle inférieur avec le triangle supérieur dans un tableau à deux dimensions. C'est donc le tableau obtenu en faisant une symétrie par rapport à la diagonale principale.

Exemple:

```
10 11 45 78          10 23 56 47
23 44 12 56   ===>   11 44 90 78
56 90 67 89          45 12 67 55
47 78 55 34          78 56 89 34
```

Exercice 6: Carré magique.

Un carré (tableau d'entiers de N lignes et N colonnes initialisées) est dit magique lorsque la somme d'une ligne, d'une colonne ou d'une diagonale quelconque est toujours égale au même nombre.

Concevoir un programme en C qui vérifie si un carré (donné par l'utilisateur) est magique.
Par exemple:

15	8	1	24	17
16	14	7	5	23
22	20	13	6	4
3	21	19	12	10
9	2	25	18	11

est un carré magique.