TP 5 : Tableaux, structures et pointeurs

Exercice 1 : Structure nombre rationnel et tableaux

1. Créer une structure `NbrRationnel` à deux champs : `Num` et `Den` contenant respectivement le numérateur et le dénominateur d'un nombre (on renommera la structure simplement `NbrRationnel` à l'aide d'un `typedef`). On écrira ensuite le corps des fonctions suivantes :

```
/* Affiche le rationnel r sous la forme: "r.Num/r.Den" */
void AfficherRationnel (NbrRationnel r);

/* Renvoie -1 si r1 et plus petit que r2, 0 si r1 est égal à r2 et 1 si \
r1 est plus grand que r2  */
int ComparerRationnel (NbrRationnel r1, NbrRationnel r2);

/* Calcule et affiche la somme de deux rationnels */
void SommeRationnel (NbrRationnel r1, NbrRationnel r2);

/* Affiche le plus grand element contenu dans le tableau tab */
void PlusGrandRationnel(NbrRationnel tab[], int taille_tab);
```

   On testera ces fonctions en déclarant un tableau de 5 rationnels et en écrivant dans le main un bloc d'instructions permettant d'afficher l'ensemble des éléments du tableau et précisant lequel de ces éléments est le plus petit. Exemple :

```
12/7
8/13
14/2
1/9 (le plus petit)
2/3
1
```

Exercice 2 : Affichage de l'occupation mémoire en cours d'exécution

Dans cet exercice l'objectif est de représenter en mémoire les données déclarées dans un programme, ainsi que leurs différentes valeurs, à un moment donné de l'exécution. Pour cela, vous représenterez l'occupation des données en mémoire en utilisant le format d'affichage suivant :
`identificateur : adresse valeur`
par exemple

```
a :   0xff8c7960  10
b :   0xff8c7964   5
```

etc.

Soit le programme suivant. Notez que l'objectif est simplement de vous familiariser avec la syntaxe et la sémantique des instructions manipulant des pointeurs.

```c
#include <stdlib.h>
#include <stdio.h>

int main(void)
   {
    int a = 10;
    int b = 5;
    int tab[3] = {1,2,3};
    int *p_int;

    /* representer l'occupation memoire */

    tab[0] = a;
    *(tab + 1) = b;
    p_int = tab + 2;

    /* representer l'occupation memoire */

    *p_int = *(p_int - 1);
    p_int = p_int - 1;
    *p_int = *(p_int - 1);
    p_int = p_int - 1;
    *p_int = *(p_int + 2);
```

```
    /* representer l'occupation memoire */

    printf("%d\t%d\t%d\t%d\t%d\n",a,b,tab[0],tab[1],tab[2]);

    return EXIT_SUCCESS;
```

Répondre au problème précédent nécessite de dupliquer 3 fois le code responsable de l'affichage du tableau. Proposer une version encapsulant ce code dans une fonction `occupation_memoire`. La fonction sera alors appelée 3 fois.

Exercice 3 : Nombre d'occurrences d'un entier dans un tableau d'entiers

Écrire un programme qui, étant donné un tableau d'entiers déjà initialisé, demande à l'utilisateur quel entier chercher et affiche ensuite le nombre d'occurrences de cet entier dans le tableau.

1. Écrire le programme en utilisant l'opérateur `[]`
2. Écrire le programme en utilisant explicitement les pointeurs pour accéder aux éléments du tableau, c'est-à-dire sans utiliser une variable d'indice.

Exercice 4 : Gestion par pointeurs d'un tableau de personnes (pointeurs et structures, pointeurs et fonctions)

La structure `etudiant` contient un prénom et un nom (`LG` caractères chacun) et `NBNOTES` notes.

1. Créer la structure `etudiant` à l'aide d'un `typedef`
2. On déclare un tableau `tabetu` de `MAX` étudiants. Écrire une procédure de saisie `creation` et une procédure d'affichage `affiche`, en utilisant un pointeur qui se déplace sur ce tableau.
3. Le programme principal appelle ces procédures, puis demande si des modifications systématiques de notes doivent être faites (numéro de l'épreuve à modifier, et différence à appliquer sur la note de cette épreuve), en appelant une fonction `modifie_notes`. Réaliser ces modifications toujours à l'aide d'un pointeur, et appeler la procédure d'affichage après chaque modification.