Variables structurées – Pointeurs – Chaîne de caractères

Exercice 1

Rappel :

```
// Une structure permet d'agréger des types de données différents ou identiques afin d'en faciliter
// la manipulation.
// Par exemple une structure Point va associer les composantes X et Y d'un couple de coordonnées.
// En pratique on définit un nouveau type structuré :
//      typedef struct point { int X; int Y; } Point ;
// et on declare les variables de ce type dont on a besoin :
//      Point P1, P2;
// l'accès aux différents champs de P1 ou de P2 se fait à l'aide d'un '.' :
//      P1.X = 10 ; P1.Y = 20;
// On peut assigner directement une variable structurée à une autre du même type :
//      P2 = P1 ;
// Chaque champ possède son propre type définit dans la structure :
//      printf("\n P2 = ( %d , %d \n", P2.X, P2.Y );
// Et Oh merveille ! on peut créer un tableau de structures du même type :
//      Point rectangle[2] = {{0,0},{5,3}};
//      printf("rectangle p1 = ( %d, %d)", rectangle[0].X, rectangle[0].Y);
//      printf("rectangle p2 = ( %d, %d)", rectangle[1].X, rectangle[1].Y);
//
// La taille d'une variable structurée s'obtient également avec sizeof()
```

- Créer un nouveau projet nommé TP4
- Créer un fichier d'entête *tp4.h* dans le dossier **Header Files**
- Le fichier d'entête contient la déclaration de la structure *heure* et du type associé HEURE: **typedef struct** heure { int heure ; int minute ;} HEURE;
- Dans le fichier main.c, faire un #include de "tp4.h"
- Dans le main(), déclarer trois variables de type HEURE : *HeureDebut*, *HeureFin*, *Duree*. HeureDebut vaut 12h30, Duree vaut 00 :45.
- Additionner ces deux valeurs pour déterminer la valeur de *HeureFin*. Et afficher les trois valeurs au format hh:mm

Remarque : Les minutes sont des valeurs inférieures à 60. C'est le programme qui calcule le report des minutes en heures quand il y en a au-moins 60 (utiliser l'opérateur %, qui calcule le reste de la division entière encore appelé *modulo*).

Exercice 2

Rappel :

```
// Une variable de type pointeur contient une adresse de donnée et non la donnée elle-même.
// Un pointeur se déclare à l'aide du caractère * :
//      char * MyPtr; // est la déclaration d'un pointeur de caractère
// Bien sûr, il est possible d'affecter à un pointeur l'adresse d'une variable classique,
// on dit que MyPtr pointe sur 'Z'
//      char UneLettre = 'Z'
//      MyPtr = &UneLettre;
// L'utilisation de la valeur pointée se fait en déréférençant le pointeur également avec * :
//      *MyPtr = 'A'
// ou encore :
//      printf("%c", *MyPrt );
// Dans cet exemple, la variable UneLettre contient désormais la valeur 'A'
// Un pointeur non initialisé avec une adresse valide NE DOIS JAMAIS ETRE UTILISE !
// Typiquement pour dire qu'un pointeur n'est pas valide, on lui donne la valeur NULL
//      char *NewPtr = NULL;
//
// Sauf pour tester si il est NULL, on n'effectue pas d'opération de comparaison sur les pointeurs.
// Par exemple, on n'écrit pas if (Ptr1 <= Ptr2) cela peut s'avérer très hasardeux.
//
// Il est possible de faire des opérations sur les pointeurs afin de pointer ailleurs :
//      Ptr1++
// Si Ptr1 pointe sur une structure dont la taille est de 10 octets, le pointeur est incrémenté de 10
// et non de 1.
// Le nom d'un tableau peut être utilisé comme un pointeur, par exemple les deux syntaxes suivantes
// sont équivalentes :
//      tab[12]= 623;
//      *(tab + 12) = 623;
```

- Définir les valeurs suivantes :
  ```
  #define SEPARATEUR '/'
  #define TAILLETAB1 20
  ```

- Déclarer un tableau de 20 entiers :
  ```
  int MyTab1[TAILLETAB1];
  ```

- A l'aide d'une boucle **for**, remplir MyTab1[] avec les nombres de 1 à 20 et afficher le contenu du tableau en séparant chaque valeur par le caractère '/'.

- En utilisant une variable MyPtr1 de **type pointeur d'entier** que l'on initialise sur le dernier élément du tableau, afficher le tableau de la fin au début en décrémentant le pointeur MyPtr jusqu'à arriver au premier élément .

Exercice 3

Rappel :

```
// Les chaînes de caractères n'existent pas en tant que type en langage C.
// Les chaînes de caractères standards, où une lettre peut être représentée par un octet, sont donc
// construites sous la forme d'une suite de codes en mémoire, chaque code correspondant à une lettre
// précise et le dernier code valant zéro pour baliser la fin de chaîne.
// par exemple, la chaîne de caractères constante "BONJOUR" existe en mémoire sous la forme
// 66 79 78 74 79 85 82 0
// Ce qui conduit à l'idée de ranger ces chaînes de caractères dans des tableaux de type char
// en n'oubliant pas le zéro à la fin.
// La taille du tableau va déterminer la longueur maximum d'une chaines. Pour ranger un mot de
// 15 caractères dans un tableau de char, il faut un tableau de 16 cases dont une pour le zéro de fin
// chaîne. Si dans le même tableau on souhaite mettre plusieurs chaines, il faut prévoir la place pour
// les zéros.
// une première approche simple va consister à utiliser un tableau de char par chaîne de caractères.
//
// Si le tableau est destiné à recevoir une chaîne de caractères constante, on peut l'initialiser
// comme suit :
//      char mot1[] = "BONJOUR";
// mot1[] ne devrait plus être modifié par la suite et dans tous les cas, il ne pourra contenir au
// plus 7 caractères.
// la syntaxe suivante est également possible :
//      char mot1[] =  {'B','O','N','J','O','U','R','\0'};
// où \0 représente l'indispensable fin de chaîne.
//
// Si une chaîne de caractères doit être modifiée, c'est un tableau de char dimensionné qui doit être
// déclaré afin de réserver une place de 30 caractères en mémoire :
//      char nom[30];
// Il est interdit d'écrire : nom = "Skywalker", c'est mal.
// Nous avons vu que le nom d'un tableau est en fait un pointeur sur le premier élément du tableau,
// et le langage C offre des fonctions spécialisées dans la manipulation des chaînes qui travaillent
// sur des pointeurs de chaînes de caractères. Exemple, pour faire une copie :
//      strcpy(nom, "Skywalker");
// nom est un pointeur et "Skywalker" est compris comme un pointeur sur une chaîne constante.
// Pour afficher cette chaîne, on peut utiliser la fonction printf avec un filtre %s:
//      printf("%s",nom);
// La fonction strlen renvoie le nombre de caractères de la chaîne sans compter le zéro.
// Toutes les fonctions de manipulation de chaînes s'attendent à trouver un caractère de fin de chaîne,
// dans le cas contraire, il y a un écrasement de la mémoire ou plantage.
//      int taille;
//      taille = strlen(nom);
//      printf("longueur du nom = %d", taille);
// On peut manipuler les chaines de caractères, caractère par caractère :
//      int i; char car;
//      i=0; do { car=*(mot1+i); *(mot2+i) = car; i++; } while (car != 0);
// ce code recopie mot1 dans mot2.
```

Ecrire un programme qui demande à l'utilisateur un nom, un prénom ainsi que le sexe (H ou F) et qui affiche le nom complet sous la forme :

Madame Annie Fratellini ou Monsieur Paul Verlaine.

Les chaines peuvent contenir jusqu'à 20 caractères, ou plutôt 19, si on considère **la valeur 0 qui termine une chaîne de caractères**. La réponse se fait à l'aide d'un *scanf_s()* et le choix de l'intitulé (Madame ou Monsieur) se fait en fonction de la valeur de la réponse (H ou F) à l'aide de la fonction _getch().

Exercice 4

Ecrire un programme qui vérifie qu'un mot entré au clavier est un palindrome. C'est-à-dire qu'il peut se lire dans les deux sens.

Par exemple : radar

Le test du palindrome ne tient pas compte de la casse (majuscule – minuscule). Le test s'effectue également à partir d'un mot débarrassé de ses caractères non-alphabétiques (non compris entre 'A' et 'Z' ou 'a' et 'z'), ainsi que de tous les caractères accentués. Penser à utiliser les fonctions de manipulation de chaînes de caractères fournies en standard et commençant pas « str .. » (cf string.h).

Exercice 5

Ecrire un programme qui permet d'entrer une phrase au clavier et compte le nombre de mot et la longueur moyenne des mots. Utiliser la fonction gets_s() à la place de scanf() pour lire les caractères tapés au clavier. La phrase est stockée dans un buffer (tableau de char) de 1024 caractères.

Exercice 6

Ecrire un programme qui recherche et affiche toutes les positions d'une valeur voulue dans un tableau de 100 entiers compris entre 0 et 20. Le tableau est initialisé avec des nombres aléatoires. Le programme indique s'il n'a pas trouvé la valeur.

L'affichage se fait comme suit :

> Entrer la valeur recherchée : 12
> La valeur 12 a été trouvée en 0 puis en 5, puis en 37.

Contrainte : La recherche se fait à l'aide d'une variable nommée Curseur de **type pointeur d'entier** et qui doit :

- être correctement déclarée
- être initialisée avec l'adresse du premier élément du tableau
- parcourir toutes les positions du tableau afin de comparer l'élément pointé avec la valeur recherchée.