TP PROGRAMMATION C: LES FONCTIONS

Exercice 1

1. Écrire une fonction, nommée **Affiche1**, se contentant d'afficher le message **"bonjour"** (elle ne possèdera aucun argument ni valeur de retour),
2. Écrire une fonction, nommée **Affiche2**, qui affiche **"bonjour"** un nombre de fois égal à la valeur reçue en argument (**int**) et qui ne renvoie aucune valeur,
3. Écrire une fonction, nommée **Affiche3**, qui fait la même chose que **Affiche2**, mais qui, de plus, renvoie la valeur (int) 0 (**return 0** ;).
4. Écrire un petit programme appelant successivement chacune de ces trois fonctions, après les avoir convenablement déclarées sous forme d'un prototype.

Exercice 2

1. Ecrire un programme se servant d'une fonction **MOYENNE** de type **float** pour afficher la moyenne arithmétique de quatre nombres réels entrés au clavier.

Prototype : **float MOYENNE(float , float) ;**

2. Ecrire une fonction **MIN** et une fonction **MAX** qui déterminent le minimum et le maximum de deux nombres réels.

Prototypes: **float MIN(float , float) ;**
float MAX(float , float) ;

3. Ecrire un programme se servant des fonctions MIN et MAX pour déterminer le minimum et le maximum de 3 réels entrés au clavier.

Exercice 3 Ecrire un programme se servant d'une fonction **F** pour afficher la table de valeurs de la fonction définie par**:**

$\boldsymbol{F(x) = \sin(x) + \ln(x) - \sqrt{x}}$ où **x** est un entier compris **entre 1 et 10**.

Exercice 4 : La formule de conversion des températures exprimées en degré Fahrenheit en degré Celsius est :

$$\boldsymbol{F = \frac{9}{5}C + 32}$$

Ecrire une fonction permettant de calculer et afficher une liste d'équivalence pour des températures comprises **entre 0°F et 300°F** avec un incrément de 10°F (sous la forme d'un tableau à 2 colonnes : la première colonne donnera les degrés Celsius, la seconde les degrés Fahrenheit).

Exercice 5

Écrire une fonction permettant de compter et d'afficher le nombre de diviseurs d'un entier n positif donné. Écrire 2 appels permettant d'afficher le nombre de diviseurs de 720 et 984.

Exercice 6

En mathématiques, on définit la fonction factorielle de la manière suivante :

$$\boldsymbol{n!} = \begin{cases} \boldsymbol{1} & \boldsymbol{si\ n = 0} \\ \boldsymbol{n * (n-1) * (n-2) * \ldots * 1} & \boldsymbol{si\ x > 0} \end{cases}$$

1.Ecrire les fonctions suivantes :

- **FACT** de type double qui reçoit la valeur n (type int) comme paramètre et qui fournit la factorielle de n comme résultat.
- **COMBIN**: qui utilise la fonction FACT pour calculer C_n^p à partir de n et p:

$$\boldsymbol{C_n^p = \frac{n!}{p! * (n-p)!}}$$

2. Ecrire un petit programme qui teste la fonction **COMBIN**

Exercice 7

1. Ecrire une fonction **int puissance(int x, int n)** qui retourne la valeur de x^n.
2. Ecrire un programme qui demande deux entiers n et m et qui calcul $\sum_{i=1}^{n} \boldsymbol{i^m}$
   Pour éviter de devoir faire une double boucle, il suffira donc de calculer $\sum_{i=1}^{n} \boldsymbol{puissance(i,m)}$

TP PROGRAMMATION C: LES TABLEAUX

Exercice 1

1. Ecrire un algorithme qui déclare et remplisse un tableau de 7 valeurs numériques en les mettant toutes à zéros. Afficher ensuite les éléments du tableau.

2. Ecrire un algorithme qui déclare et remplisse un tableau contenant les six voyelles de l'alphabet latin. Afficher ensuite les éléments du tableau.

Exercice 2 : Ecrire un programme qui lit la dimension N d'un tableau T du type **int** (dimension maximale : 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau.Calculer et afficher la somme des éléments du tableau.

Exercice 3 : Ecrire un programme constituant un tableau, à partir de deux tableaux de même longueur préalablement saisis. Le nouveau tableau sera la somme des éléments des deux tableaux de départ.

Tableau 1:

2	1	7	4	2	5	7

Tableau 2:

1	2	1	5	6	4	0

Tableau à constituer:

3	3	8	9	8	9	7

Exercice 4 : Ecrire un programme qui calcule le produit scalaire de deux vecteurs d'entiers X et Y (de même dimension). Si X(x1, x2, ..., xn−1, xn) et Y (y1, y2, ..., yn−1, yn) Alors le produit scalaire: $\sum_{i=1}^{n} x_i \times y_i$

Exercice 5 : Soit le tableau T suivant :

10	7	9	7	10	6	7	4	8	8

Pour chaque élément de T on ne garde que sa première occurrence et on remplace les autres par 0.

10	7	9	0	0	6	0	4	8	0

Pour regrouper les éléments restant au début du tableau T.

10	7	9	6	4	8	0	0	0	0

Ecrire un programme qui fait le traitement ci-dessus pour un tableau T de N (N<100) entiers positifs non nuls et détermine et affiche le nombre d'éléments différents de T.

Exercice 6: Ecrire un programme qui construit et affiche une matrice carrée unitaire U de dimension N. Une matrice unitaire est une matrice, telle que: $Uij = \begin{cases} 1 \; si \; i = j \\ 0 \; si \; i \neq j \end{cases}$

Exercice 7 : Ecrire une fonction **Lecture** et une fonction **Affichage** qui effectuent respectivement la lecture et l'affichage d'un tableau de **N** entiers (N **constante** définie dans le programme). **Prototypes :**

void Lecture(int tab[], int N) ; void affichage(int tab[], int N);

Ecrire une fonction **somme** prenant en argument un tableau T de **N entiers** et qui calcule la somme des éléments de ce tableau.

Prototype : int somme(int tab[], int N) ;

3.Tester toutes les fonctions à l'aide de la fonction principale main.

Exercice 8 : Algorithmique de bases sur les tableaux

1. Ecrire une fonction qui retourne l'indice de l'élément le plus petit contenu dans un tableau et une fonction qui retourne l'indice de l'élément le plus grand contenu dans un tableau.
2. Ecrire une fonction qui calcule la somme des valeurs contenues dans un tableau.
3. Ecrire une fonction qui calcule la moyenne des valeurs contenues dans un tableau.
4. Ecrire une fonction qui effectue le produit scalaire de deux vecteurs de même taille représentés par des tableaux à une dimension.
5. Ecrire une fonction qui indique si un élément donné appartient à un tableau ou pas

Année Universitaire : 2019-2020

Filière : GTE-S2

TP PROGRAMMATION C: Les types structures et les pointeurs

Exercice 1

1.Ecrire un programme qui permet :

- Déclarer un entier ***i*** et un pointeur vers un entier, ***p***
- Initialiser la valeur de ***i***
- Modifier l'entier pointé par ***p*** (en utilisant ***p*** et non pas ***i***)
- Afficher la valeur de ***i***.

2.Écrire, de deux façons différentes, un programme qui lit 10 nombres entiers dans un tableau avant d'en rechercher le plus petit :

- En utilisant uniquement le « formalisme tableau »,
- En utilisant le « formalisme pointeur », chaque fois que cela est possible

Exercice 2

Ecrire un programme qui effectue les traitements suivants. Après chaque traitement et pour chaque pointeur ou valeur, afficher le contenu du pointeur, son adresse, la valeur pointée et l'adresse de la valeur pointée. Utiliser le format **%p** pour l'affichage des pointeurs.

- Déclarer un pointeur **p** sans l'initialiser et un pointeur **q** initialisé à **NULL** ;
- Le pointeur **p** pointe sur une valeur entière **v** initialisée à **10** ;
- A l'aide du pointeur **p**, modifier cette valeur entière **v** "à partir du clavier"
- Utiliser seulement le pointeur **p** pour initialiser une valeur entière **w** à la valeur de **v** ;

Exercice 3

Ecrire une fonction **swap(a,b)** qui échange les valeurs de deux variables entières a et de b passée en paramètre.

Exercice 4

Ecrire une fonction **incremente** qui incrémente une variable entière passée en paramètre. Son prototype est : **void incremente(int * var);**

Exercice 5

1. Pour représenter un nombre complexe ($z = a + ib, ou\ a\ et\ b \in \mathbb{R}\ et\ i^2 = -1$), créer une structure **complexe** qui contient les champs « partie réelle a » et la « partie imaginaire b ».

2. Ecrire un programme qui :

   a. Saisit deux complexes c1 et c2
   b. Calcule la somme s= c1 + c2
   c. Affiche c1, c2 et s

Exercice 6

Soit la structure de type **point** définit comme suit :

```
struct point {
        float x ;
        float y ;
    }
```

Écrire un programme qui :

1. Lit au clavier les coordonnées dans un tableau de points. Le nombre de point **NP** du tableau sera fixé par une instruction **#define**.

2. Affiche à l'écran l'ensemble des informations précédentes