Travaux Pratiques N°7
Les fonctions

I. Déclaration

En C, les sous-programmes s'appellent des fonctions. Une fonction a toujours un type de retour, qui correspond au type du résultat qu'elle peut renvoyer et qui peut être n'importe lequel des types que nous avons précédemment étudié; ce type de retour peut être void si on souhaite que la fonction ne renvoie rien. Elle a aussi un nom. Et, enfin, elle a une liste de zéro, un, ou plusieurs, paramètres.

Syntaxe :

```
TypeDeRetour NomFN (Type1 Parametre1, Type2 Parametre2)
{
    /*variables locales*/

      instructions de la fonction
      return valeur_de_retour ; /*optionnelle si la fonction ne retourne rien (de type void)*/
}
```

II. Passage de paramètres :

Tout paramètre d'une fonction peut être de type ordinaire (int, float,…) ou de type pointeur.

Dans le cas d'un paramètre ordinaire, le passage de ce dernier est appelé passage par valeur. Dans ce cas la fonction fait une copie de la valeur du paramètre. Elle n'utilise que la copie, qui peut être modifiée. Mais à la fin du traitement de la fonction, la valeur de la variable passée en paramètre n'a pas été modifiée, puisque c'est seulement la copie qui a été modifiée.

Dans le cas d'un paramètre de type pointeur, le passage de ce dernier est appelé passage par adresse. Dans ce cas la fonction ne fait pas de copie de la valeur du paramètre, comme c'est fait dans le cas du passage par valeur. Par conséquent, si le paramètre change de valeur, alors à la sortie de la fonction, la variable passée en paramètre contient la dernière modification, donc la dernière valeur.

III. Les prototypes

Le prototype d'une fonction est une description d'une fonction qui est définie plus loin dans le programme. On place donc le prototype en début de programme (avant la fonction principale *main()*). Le prototype d'une fonction reprend exactement l'en-tête de la fonction, mais pas son corps, qui est remplacé par un point-virgule.

TypeDeRetour NomFN (Type1 Parametre1, Type2 Parametre2) ;

Cette description permet au compilateur de « vérifier » la validité de la fonction à chaque fois qu'il la rencontre dans le programme.

IV. Appels des fonctions

Une fonction est appelée comme étant une instruction du programme si elle ne retourne rien, et elle est appelée dans une expression s'une affectation, comparaison ou autre.

Exemples :

```
#include <stdio.h>
float carre (float X) {
        X= X*X;
        return (X) ;
}

void main(){
        float A = 2, B;
        B= carre (A) ;
        /*la valeur de A ne change pas après l'appel de la fonction carre */
        printf("le carre de %f est=%f ",A , B);
}
```

```
#include <stdio.h>
void permutation (int *, int *) ; /*prototype de la fonction permute */
void main(){
  int a = 2, b = 5;
  printf("debut programme principal :\n a = %d \t b = %d\n",a,b);
  permutation(&a, &b);
  printf("fin programme principal :\n a = %d \t b = %d\n",a,b);

}
void permutation (int *P1,  int *P2){
  int temp;
  temp = *P1;
  *P1 = *P2;
  *P2 = temp;
}
```

V. Travail demandé

Exercice 1 :

1. Ecrire une fonction de prototype int puissance (int x, int y) qui calcule x^y.
2. Ecrire la fonction main() permettant de saisir au clavier deux entiers A et B, calculer et afficher A^B en utilisant la fonction puissance.

Exercice 2 :

Réaliser en C un algorithme d'une fonction qui recherche le premier nombre entier naturel dont le carré se termine par n fois le même chiffre.
Exemple : pour n = 2, le résultat est 10 car 100 se termine par 2 fois le même chiffre.

Exercice 3 :

1. Ecrire une fonction qui permet de vérifier l'appartenance d'un entier x dans un tableau T de n entiers.
2. Tester cette fonction en écrivant un programme principal (main).

Exercice 4 :
Écrire une fonction C qui permet d'afficher les sous séquences strictement croissantes depuis un tableau de N entiers.
Exemple pour T = 5|7|9|2|3|1|20|25
La fonction affiche :
5|7|9
2|3
1|20|25

Exercice 5 : Récursivité simple
1. Ecrire une fonction non récursive qui calcule et retourne la factorielle de l'entier passé en argument.
2. Ecrire une fonction récursive qui calcule et retourne la factorielle de l'entier passé en argument.
3. Tester ces deux fonctions en écrivant un programme principal (main).

Exercice 6: Récursivité simple
Soit la suite numérique U_n suivante : Si n = 0 alors $U_0 = 4$ sinon si n > 0 alors $U_n = 2*U_{n-1} + 9$
Écrire une fonction C qui calcul le terme U_n.

Exercice 7: Récursivité croisée
Écrire deux fonctions C qui permettent de calculer les $n^{èmes}$ (n passé en argument) termes des suites entières U_n et V_n définies ci-dessous.

$$\begin{cases} U_0=1 \\ U_n=V_{n-1}+1 \end{cases} \qquad \begin{cases} V_0=0 \\ V_n=2*U_{n-1} \end{cases}$$