RETOUR SUR LES FONCTIONS

1. Rappels.

1.1. Fonctions

Une fonction est un bloc d'instruction que l'on peut appeler en lui transmettant des paramètres. Les paramètres sont des variables locales à la fonction, c'est à dire qui n'existent que dans le bloc d'instructions de la fonction, elles sont initialisées avec les valeurs transmises à l'appel.

Les fonctions ont un nom et un type. Le type peut être un des types simples, **void** ou un type de structure. Dans le cas où le type n'est pas **void**, la fonction comporte une instruction **return**.

L'instruction **return** est suivie d'une expression du même type que celui de la fonction.

```
type nom_de_fonction ( liste de variables paramètres )
{
    ...
    return expression ; // du type de la fonction
}
```

L'appel d'une fonction se fait avec son nom et en précisant la valeur de chacun de ses paramètres.

Toute fonction (sauf main) **doit** avoir un prototype. Les prototypes sont **obligatoirement** groupés en début de programme après les directives de compilation et les déclarations de type. Ils sont constitués de la ligne d'en tête de chacune des fonctions du programme suivit d'un point virgule.

1.2. Déclaration de types

Il est possible de déclarer des types à l'aide de l'instruction **typedef**. C'est une instruction, il y a donc un point virgule en fin de ligne. Pour un type de variable dimensionné la syntaxe est :

```
typedef type_de_variable_simple Nom_du_nouveau_type[ nombre d'éléments ] ;
```

En général, on déclare des types globaux en début de programme **après** les directives de compilation et **avant** les prototypes.

2. Exercice 1

- ★ Ecrivez une fonction qui calcule la valeur de la fonction $F(t) = 230\sqrt{2}\sin(\omega t)$ en **un** point t.
 - ◆ Déterminez les paramètres d'entrée de la fonction.
 - ◆ Donnez le type de variable retournée.
 - ◆ Choisissez un nom de fonction, et écrivez-là ainsi que son prototype.
 - ◆ À l'aide de la fonction écrite, calculez les valeurs prises par un signal de fréquence 50 Hz durant **une** période sur 100 points. (Calculs dans la fonction main())
- ★ Modifiez la fonction pour qu'elle retourne tous les points dans une variable dimensionnée. Déterminez les paramètres de cette fonction ainsi que son type.
- ★ Créez un type de variable dimensionnée que vous appellerez **datas**. Modifiez la fonction précédente pour qu'elle utilise en paramètre une variable de type **datas**.

3. Exercice 2

- ◆ Créez un type de variable appelé **vect** pouvant contenir **dim** éléments réels. **dim** est une constante définie en directive de compilation.
- ◆ Ecrivez une fonction appelée **saisie**, qui saisisse les coordonnées d'un vecteur de dimension **dim**. **dim** est-il un paramètre de la fonction ?
- ◆ Ecrivez une fonction appelée **produit** qui calcule le produit scalaire de deux vecteurs. Déterminez quelles sont les paramètres nécessaires à cette fonction.
- ★ À l'aide de ces deux fonctions, saisissez deux vecteurs X et Y, puis calculez le produit scalaire de ces deux vecteurs et affichez le résultat.

- ◆ Créez un type de variable appelé **mat**, permettant de stocker les éléments d'une matrice composée d'une collection de **dim vect**.
- ◆ Ecrivez une fonction appelée **extrait**, qui extrait d'une matrice **M** la colonne **k** et la stocke dans un vecteur **V**. Déterminez les paramètres de la fonction.
- ◆ Ecrivez une fonction affiche qui affiche sous forme matricielle une matrice à l'écran. Déterminez les paramètres nécessaires à cette fonction.
- ★ À l'aide des fonctions **produit** et **extrait**, calculez le produit matriciel de deux matrices A et B dans la fonction main. La saisie des deux matrices se fera à l'aide de la fonction **saisie** dans la fonction **main**.
- ★ Testez votre programme sur des matrices de rang 3 et des vecteurs de dimension 3.

4. Rappels produit scalaire / Produit matriciel

$${}^tV_1 = 3\vec{e}_x + 2\vec{e}_y - \vec{e}_z : V_1 = \begin{bmatrix} 3 & 2 & -1 \end{bmatrix}$$

$$\vec{V}_2 = -2\vec{e}_x + 0\vec{e}_y - 6\vec{e}_z : V_2 = \begin{bmatrix} -2 \\ 0 \\ -6 \end{bmatrix}$$

$$V_1 \cdot V_2 = \sum V_1^k V_2^k \text{ où } V^k \text{ est la coordonnée } k \text{ du vecteur } V$$

tV_1 est un vecteur à gauche ou vecteur ligne, V_2 est un vecteur à droite ou vecteur colonne. On note les vecteurs ligne tV

Une matrice est un collection de vecteurs : $M = [[V_1] \dots [V_n]]$

Ainsi la matrice $M = \begin{bmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 9 & -4 & 3 \end{bmatrix}$ est en fait le regroupement des trois vecteurs : $V_1 = \begin{bmatrix} 1 \\ 5 \\ 9 \end{bmatrix}$,

$$V_2 = \begin{bmatrix} 2 \\ 6 \\ -4 \end{bmatrix}, V_3 = \begin{bmatrix} 3 \\ 7 \\ 3 \end{bmatrix}, \text{ soit } M = \left[\begin{bmatrix} 1 \\ 5 \\ 9 \end{bmatrix} \begin{bmatrix} 2 \\ 6 \\ -4 \end{bmatrix} \begin{bmatrix} 3 \\ 7 \\ 3 \end{bmatrix} \right]$$

Mais la matrice M peut aussi être vue comme composée de vecteurs Ligne :

$$M = \left[\begin{bmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 9 & -4 & 3 \end{bmatrix} \right] = \begin{bmatrix} {}^tU_1 \\ {}^tU_2 \\ {}^tU_3 \end{bmatrix}$$

Le produit matriciel est donc le produit des vecteurs ligne de la matrice à gauche par les vecteurs colonnes de la matrice à droite :

$$M \cdot M = \begin{bmatrix} {}^tU_1 \\ {}^tU_2 \\ {}^tU_3 \end{bmatrix} \cdot [[V_1] [V_2] [V_3]]$$

Il s'agit donc d'un ensemble de produits scalaires et l'élément ligne i, colonne j de la matrice résultat est en fait le produit scalaire du vecteur ${}^tU_i \cdot V_j$

Exemple de résultat :

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 9 & -4 & 3 \end{pmatrix}; B = \begin{pmatrix} 2 & 6 & 7 \\ -5 & 2 & 7 \\ 1 & 0 & 1 \end{pmatrix}; A * B = \begin{pmatrix} -5 & 10 & 24 \\ -13 & 42 & 84 \\ 41 & 46 & 38 \end{pmatrix}$$