TD 7bis de langage C

Les pointeurs de fonctions

Il est possible de définir des variables ayant le type "pointeur de fonction". La valeur d'une telle variable est l'adresse en mémoire du début du code exécutable d'une fonction, et sa déclaration s'écrit selon l'une des deux syntaxes suivantes :

*type_fonction (*nom_variable)(paramètres_de_la_fonction);*
*type_fonction (*nom_variable)(types_paramètres_de_la_fonction);*

Pour lire de telles déclarations sans risque de confusion, on a intérêt à procéder en plusieurs étapes.

Exemple :

*int (*p_f)(float r);* ou *int (*p_f)(float);*

Pour comprendre cette déclaration, on pourra procéder de la manière suivante :

- ***p_f*** : le nom de la variable déclarée est ***p_f***
- ****p_f*** : la variable ***p_f*** est un pointeur.
- ***(*p_f)()*** : l'objet pointé est une fonction
- ***int (*p_f)(float r)***, ou ***int (*p_f)(float)*** : l'objet pointé est une fonction retournant un entier, ayant un seul paramètre de type flottant.

Attention :
Il ne faut surtout pas oublier les parenthèses autour de ****p_f***, sans quoi le compilateur comprendrait que ***p_f*** est une fonction retournant un pointeur sur un entier, ayant un seul paramètre de type flottant !
L'intérêt essentiel de telles variables est de pouvoir passer des fonctions en paramètres à une fonction. Dans un cas de ce genre, le paramètre formel doit être de type "pointeur de fonction".

Exercice 1 :

Ecrire un programme qui permet de calculer la valeur approchée de la dérivée d'une fonction trigonométrique (cos, sin ou tan), en un point choisi par l'utilisateur.
On utilise pour cela l'identité suivante, permettant le calcul approché de la dérivée d'une fonction f au point x :

f'(x) == (f(x+EPSILON)-f(x-EPSILON))/(2*EPSILON)

pour EPSILON suffisamment petit.

Exercice 2 :

On désire écrire un programme qui trie et affiche un ensemble de vecteurs du plan, par ordre croissant, selon la valeur de leur norme euclidienne. Pour cela, on utilise la fonction ***qsort***, déclarée dans le fichier *stdlib.h*, qui réalise un tri rapide ("quick sort"), et qui a l'en-tête suivant :
*void qsort(void *tab,size_t nb_elem,size_t taille, int (*compare)(const void *,const void *))*

Remarques :

- La fonction ***qsort*** est "doublement générique" : d'une part, elle peut trier des éléments de type quelconque ; d'autre part, elle peut trier des éléments (de même type) selon n'importe quel critère de comparaison.
- Le mot-clé ***const*** signifie que les deux objets pointés par les deux paramètres de type ***void ** ne peuvent pas être modifiés à l'intérieur de la fonction pointée par compare. Cela vise à éviter tout "effet de bord".

La fonction ***qsort*** réalise le tri (croissant) du tableau d'adresse ***tab***, comportant ***nb_elem*** éléments de ***taille*** octets. Pour cela, ***qsort*** va effectuer différentes comparaisons entre les éléments du tableau, en appelant une fonction de comparaison. Cette fonction de comparaison devra retourner :

- Une valeur négative si l'élément pointé par son premier paramètre est inférieur à l'élément pointé par son second paramètre (au sens d'un certain ordre).
- Une valeur nulle si les deux éléments sont égaux. Dans ce cas, les deux éléments apparaissent, après le tri, dans un ordre imprévisible a priori.
- Une valeur positive si le premier élément est supérieur au second.

Après avoir défini le type vecteur par :

```
typedef struct
{
        double x;
        double y;
} vecteur;
```

réaliser les tâches suivantes :

- Ecrire la fonction ***norme***, qui retourne la valeur de la norme euclidienne d'un vecteur dont l'adresse est passée en paramètre, d'en-tête :
  *double norme(const vecteur *v)*
- Ecrire la fonction ***compare_norme***, d'en-tête :
  *int compare_norme(const void *v1,const void *v2)*
  qui compare les normes euclidiennes des deux vecteurs pointés par ***v1*** et ***v2***.
- Ecrire un programme permettant de lire les coordonnées tapées au clavier d'au plus 100 vecteurs, puis de les afficher à l'écran, une fois triés par normes croissantes.