Programmation Procédurale en Langage C – TD1

Syntaxe, déclarations de variables, premiers tableaux

Exercice 1 : Noms de variables

Lesquels des identificateurs suivants sont acceptés en C ?

fonction-1	limite_inf.	a
_MOYENNE_du_MOIS_	lim_supérieure	_
3e_jour	__A_	3

Exercice 2 : Explication de texte

Soit le code source en C ci-dessous. Essayer de distinguer et de classifier autant que possible les éléments qui composent ce programme (commentaires, variables, déclarations, instructions, mots clés, etc...)

```c
/* Ce programme calcule la somme de 4 nombres entiers saisis au clavier */
#include <stdio.h>

int main () {
  int nombre, somme, compteur ;

  somme = compteur = 0 ;      /* Initialisation des variables */

  while (compteur < 4) {      /* Lecture des donnees */
    printf ("Entrez un nombre entier : ") ;    /* Lire la valeur du nombre suivant */
    scanf ("%d", &nombre) ;

    somme += nombre ;         /* Ajouter le nombre au resultat */
    compteur++ ;              /* Incrementer le compteur */
  }

  printf ("La somme est : %d \n", somme) ;  /* Affichage du resultat */

  return 0 ;
}
```

Exercice 3 : Types de base

Quel(s) type(s) numérique(s) pouvez-vous utiliser pour les groupes de nombres suivants ? Cochez avec un ∘ les choix possibles et avec un × le choix le plus économique.

Sur une machine qui donne `sizeof(char)` → 1 octet
`sizeof(short)`→ 2 octets
`sizeof(int)` → 4 octets

		signed			unsigned					long
	Nombres	char	short	int	char	short	int	float	double	double
(a)	`1 12 4 0 -125`									
(b)	`1 12 -4 0 250`									
(c)	`1 12 4 0 250`									
(d)	`1 12 -4 0.5 125`									
(e)	`-220 32000 0`									
(f)	`-3000005.000000001`									
(g)	`410 50000 2`									
(h)	`410 50000 -2`									
(i)	`3.14159265 1015`									
(j)	$2*10^{7}$ `10000001`									
(k)	$2*10^{-7}$ `10000001`									
(l)	$-1.05*10^{50}$ `0.0001`									
(m)	`305.122212 0 -12`									

Exercice 4 : Parcours de tableaux

1) Écrire une **fonction** qui affiche les valeurs d'un tableau `a` de `n` entiers, et dont le prototype sera le suivant :

```
void tableau_afficher (int a[], int n) ;
```

2) Écrire une **fonction** qui calcule la somme des éléments d'un tableau `a` de `n` entiers. Pour ce faire, utiliser une variable d'aide entière `int sum` initialisée à `0`. La fonction retournera la somme calculée et son prototype sera :

```
int tableau_somme (int a[], int n) ;
```

3) Écrire une **fonction** qui multiplie chaque élément d'un tableau `a` de `n` entiers par un scalaire. Le prototype de la fonction sera le suivant :

```
void tableau_scalaire (int k, int a[], int n) ;
```

Remarque : 4.1 – 5 lignes / 4.2 – 6 lignes / 4.3 – 4 lignes

Exercice 5 : Parcours de tableaux (*optionnel*)

Cet exercice est pour ceux qui ont de l'expérience en C, et qui ont donc déjà fini les précédents...

1) Écrire une **fonction** qui réalise l'addition des éléments de 2 tableaux `a` et `b`, de taille identique `n`, pour mettre le résultat dans un troisième tableau `dest`, également de taille `n`. Le prototype de la fonction sera le suivant :

```
void tableau_addition (int dest[], int a[], int b[], int n) ;
```

2) Écrire une **fonction** qui vérifie si 2 tableaux `a` et `b`, de taille identique `n`, sont égaux (*c.à.d* que leurs éléments sont identiques). Pour ce faire, utiliser une variable d'aide booléenne `char egaux` initialisée à `1`. La fonction retournera vrai si les tableaux sont égaux, faux le cas échéant. Le prototype de la fonction sera le suivant :

```
char tableau_egal (int a[], int b[], int n) ;
```

3) Écrire une **fonction** qui inversion un tableau `a` de `n` entiers. Par exemple, [2, 1, 4, 3] deviendra [3, 4, 1, 2]. Pour ce faire utiliser une variable d'aide entière `int tmp`. Le prototype de la fonction sera le suivant :

```
void tableau_renverser (int a[], int n) ;
```

Remarque : 5.1 – 4 lignes / 5.2 – 8 lignes / 5.3 – 6 lignes