Développement logiciel (S4)

Initiation au langage C
3- Les tableaux et les fichiers

Objectifs : Être capable de développer des applications basiques de type console en C. Être capable d'utiliser les tableaux et les fichiers.

Avertissement

Tout au long de ce TP, vous répondrez au questions posées sur votre compte-rendu. Ce compte-rendu devra inclure les programmes réalisés. Ces programmes devront être commentés !

Qu'est-ce qu'un tableau ?

Un tableau est un ensemble ordonné d'éléments de même nature ; les éléments sont manipulés individuellement ; ils sont désignés par un nombre [0, N − 1].

Un tableau est déclaré par son type, puis son nom. On déclare le nombre d'éléments du tableau entre crochets [ ].

Exemple : `int tablo[6]` est un tableau de 6 entiers.

tablo	42	8	15	23	16	4
	tablo[0]	tablo[1]	tablo[2]	tablo[3]	tablo[4]	tablo[5]

Il est possible d'initialiser un tableau lors de sa déclaration, par exemple :

```
int tablo[6] = {42, 8, 15, 23, 16, 4}
```

permet d'initialiser le tableau conformément au schéma de la figure précédente.

Si le nombre de données est inférieure à N, alors les autres valeurs sont initialisées à 0.

```
int tablo[6] = {42, 8, 15}
```

est équivalent à :

```
int tablo[6] = {42, 8, 15, 0, 0, 0}
```

Pour accéder à un élément du tableau, il suffit de faire suivre le nom du tableau de la valeur entre crochet. Exemple : après l'instruction :

```
x = tablo[2];
```

On obtient x = 15.

Généralement pour accéder un par un aux éléments d'un tableau, on utilise une boucle for.

Il est possible de travailler avec des tableaux à deux dimensions. Les mathématiciens appellent cela une matrice.

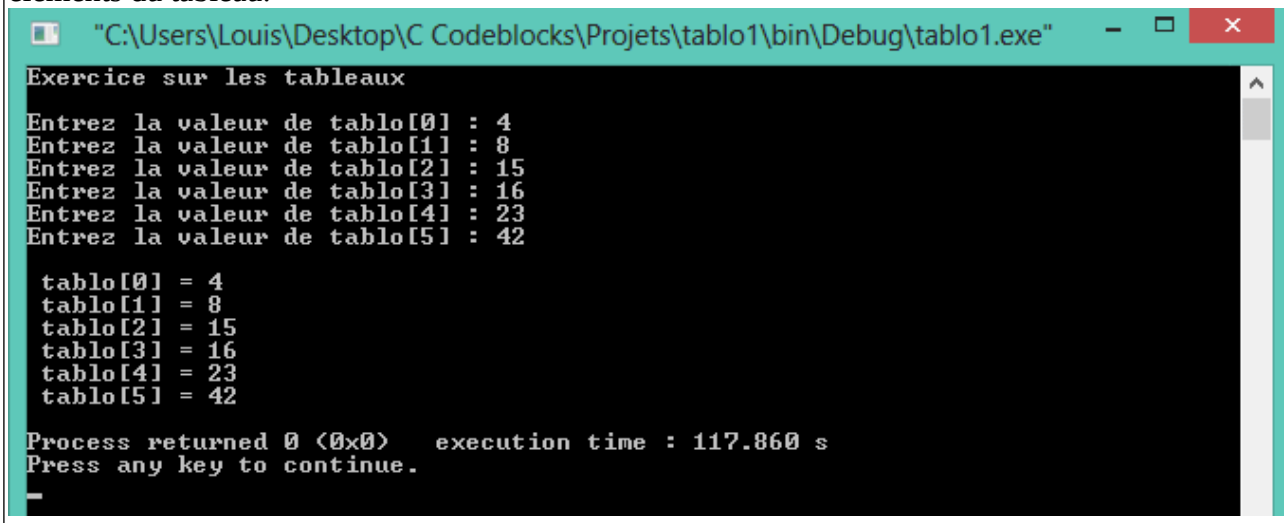
Exemple : `int tablo2d[3][6]` correspond à un tableau de 18 éléments organisés en 3 lignes et 6 colonnes.

tablo2d						
	42	8	15	16	23	4
	81	12	31	46	65	66
	19	23	87	15	24	63

Ici, tablo2d[1][2]= 23

Manipulation n°1

Q.1) Créer un projet appelé tablo1 qui permet de saisir les 6 entiers du tableau tablo et ensuite les affiche à l'écran, conformément à la copie d'écran jointe.
Pour cela vous utiliserez 2 boucles for successives. Une première boucle où vous demanderez à l'utilisateur de saisir un à un les éléments du tableau puis une boucle for où vous afficherez tous les éléments du tableau.



Opérations sur les tableaux

Comment détecter le plus grand élément d'un tableau ? Le plus petit ?

Travaux dirigés

On considère le tableau suivant :

4	15	23	8	42	16

Comment détecter le plus grand élément du tableau ? Quel opération faites vous ? Comment pourrait faire un ordinateur ?

Pour répondre à ces questions, nous allons parcourir le tableau élément par élément.

Étape 1 : examen du premier élément

4

Quel est le plus grand élément ?

Étape 2 : examen du 2eme élément

4	**15**

Quel est le plus grand élément ? Quel test effectuez vous ?

Étape 3 : examen du 3eme élément

4	15	**23**

Quel est le plus grand élément ? Quel test effectuez vous ?

Etape 4 : examen du 4eme élément

4	15	23	**8**

Quel est le plus grand élément ? Quel test effectuez vous ?

Etape 5 : examen du 5eme élément

4	15	23	8	**42**

Quel est le plus grand élément ? Quel test effectuez vous ?

Etape 6 :examen du 6eme élément

4	15	23	8	42	**16**

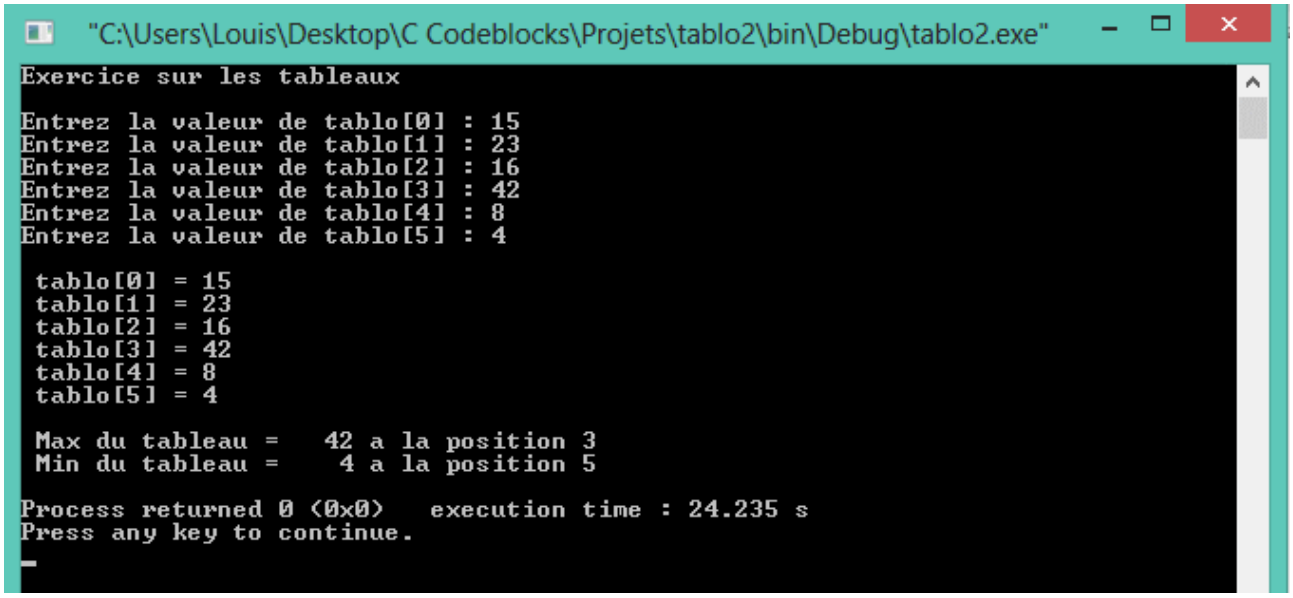
Quel est le plus grand élément ? Quel test effectuez vous ?

Le test est-il fini ?

Écrire un algorithme qui permet de détecter le plus grand élément d'un tableau.
Modifier votre algorithme pour détecter le plus petit élément d'un tableau.

Manipulation n°2

Q.1) Créer un projet appelé tablo2, qui sur la base des algorithmes précédents, détermine la plus grande et la plus petite valeur dans le tableau d'entiers tablo. Afficher ensuite la valeur et la position du maximum et du minimum. Si le tableau contient plusieurs maxima ou minima, le programme retiendra la position du premier maximum ou minimum rencontré.



Les fichiers

Pour conserver les résultats de l'exécution d'un programme ou pour gérer de grandes suites de données, on utilise les fichiers.
Nous allons voir ici comment écrire et lire simplement des données dans un fichier en utilisant les fonctions fprintf et fscanf qui sont équivalentes aux fonctions printf et scanf que vous avez déjà utilisées. D'autres fonctions de lecture et d'écriture existent et seront juste citées ici.

Écriture d'un fichier

Le programme suivant permet d'écrire la donnée 8 dans un fichier test.txt crée lors de l'exécution du programme.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    FILE* fichier = NULL; // Cette instruction constitue la déclaration du fichier
    int donnee = 8; // C'est la donnée à écrire dans le fichier
    fichier = fopen("test.txt", "w"); // le fichier test.txt est ouvert en écriture (w)
    /* Test de l'ouverture pour vérifier les éventuels problèmes */
        if (fichier != NULL) // Si pas de pb d ouverture fichier
        {
            fprintf(fichier, "%d \n", donnee); // on ecrit la donnée
        }
        else // Si pb d'ouverture fichier
        {
            printf("Probleme acces fichier \n"); //
        }
    fclose(fichier); // fermeture du fichier
return 0;
}
```

Manipulation n°3

Q.1) Le fichier correspondant au programme donné est fourni. Créer un projet nommé fich_ecr_1 avec ce fichier. Compiler et exécuter le programme. Rechercher l'endroit ou le fichier test.txt a été créé. Ouvrir le fichier avec le bloc-note. Conclure.

Q.2) Créer un nouveau projet nommé fich_ecr_2 et modifier le programme précédent pour écrire maintenant une suite de 6 entiers en utilisant un tableau.

Lecture d'un fichier

Le programme suivant permet de lire la donnée présente dans le fichier test.txt et de l'afficher à l'écran.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    FILE* fichier = NULL; // declaration fichier
    int donnee = 0; // donnee à lire initialisée par défautà 0
    fichier = fopen("test.txt", "r"); // le fichier test.txt est ouvert en lecture (r)
    /* Test de l'ouverture pour vérifier d'éventuels problèmes */
    if (fichier != NULL) // si pas de pb d'ouverture fichier
    {
       fscanf(fichier, "%d", &donnee); // on lit la donnée
       fclose(fichier); // fermeture du fichier
    }
    else // si pb d'ouverture fichier
    {
        printf("Probleme acces fichier\n");
    }
    // affichage de la donnée lue
      printf("La donnee lue est : %d \n", donnee);

    return 0;
}
```

Manipulation n°4

Q.1) Le fichier correspondant au programme donné est fourni. Créer un projet nommé fich_lect_1 avec ce fichier. Compiler et exécuter le programme. Que ce passe t-il ? Placer au bon endroit le fichier test.txt écrit avec le programme d'écriture (manipulation précédente). Vérifier que le programme fonctionne correctement. Modifier le fichier avec le bloc-note. Vérifier le bon fonctionnement du programme..

Q.2) Créer un nouveau projet nommé fich_lect_2 et modifier le programme précédent pour lire maintenant une suite de 6 entiers en utilisant un tableau.

Autres instructions de lecture/écriture

Les autres instructions pour lire et écrire dans un fichier sont les suivantes :

fputc, fputs, fgetc, fgets.

L'instruction fputc permet d'écrire un caractère à la fois dans un fichier.

L'instruction fputs permet d'écrire une chaîne de caractères dans un fichier.

L'instruction fgetc permet de lire un caractère dans un fichier.

L'instruction fgets permet de lire une chaîne de caractères dans un fichier.

L'instruction fgetc permet en particulier de tester si on est arrivé en fin de fichier. En effet, si la fonction n'arrive pas à lire de données, elle renvoie un caractère appelé EOF : End Of File : Fin de fichier.

A titre d'exemple, le programme suivant lit tous les caractères d'un fichier un à un et les écrit à chaque fois à l'écran, jusqu'au dernier caractère.

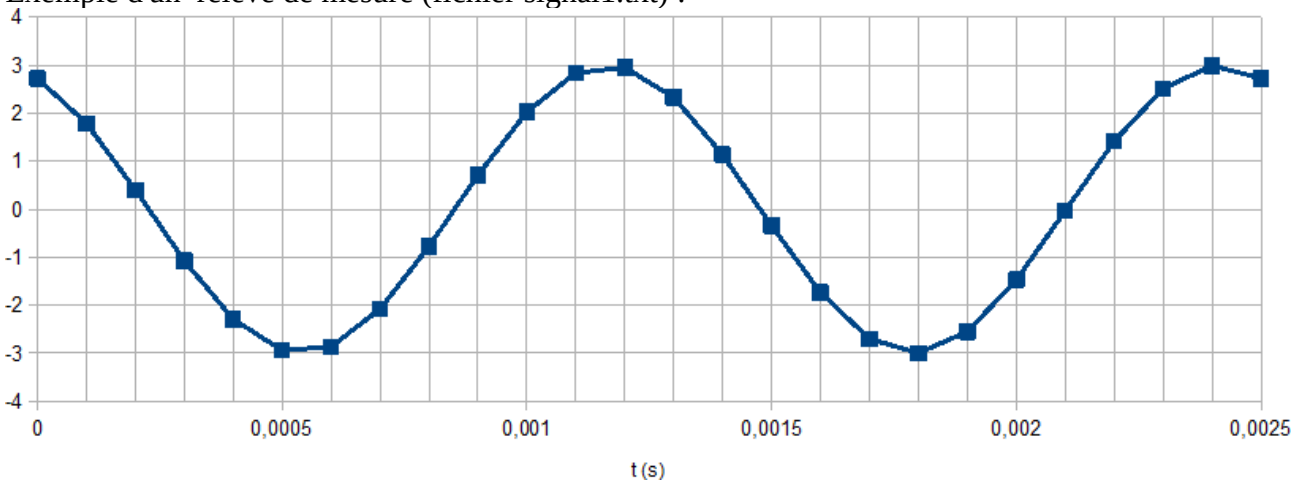
```
int main()
{
    FILE* fichier = NULL;
    int caractereActuel = 0;

    fichier = fopen("test.txt", "r");

    if (fichier != NULL)
    {
        // Boucle de lecture des caractères un à un
        do
        {
            caractereActuel = fgetc(fichier); // On lit le caractère
            printf("%c", caractereActuel); // On l'affiche
        } while (caractereActuel != EOF); // On continue tant que fgetc n'a pas retourné
EOF (fin de fichier)

        fclose(fichier);
    }

    return 0;
}
```

Une autre solution est d'utiliser la fonction feof(file), c'est ce qui est fait dans l'exemple donné en mini-projet.

Mini projet

Un oscilloscope relève des signaux évoluant au cours du temps. Les résultats des mesures sont stockés par l'appareil de mesure dans un fichier .txt.

Exemple d'un relevé de mesure (fichier signal1.txt) :



Chaque point est mesuré à intervalle de temps régulier. Cet intervalle régulier est la période d'échantillonnage T_E. Ici T_E = 0,0001 s soit 0,1ms ou 100 µs.
La fréquence d'échantillonnage correspond au nombre de points mesurés par seconde. La relation entre période d'échantillonnage T_E et fréquence d'échantillonnage f_E est :

$$f_E = \frac{1}{T_E}$$

Structure du fichier :

fréquence d'échantillonnage
Valeur de premier point (en V)
Valeur de deuxième point (en V)
…....
…....
Valeur du dernier point (en V)

3 fichiers d'exemples différents sont donnés (signal1.txt, signal2.txt, signal3.txt).

Nombre de points de mesure :

On ne connaît pas le nombre de points de mesure (qui est variable) avant de lire le fichier. Celui ci devra être déterminé lors de la lecture du fichier. On supposera que celui ci est toujours inférieur à 150. On lira donc toutes les données jusqu'à la fin du fichier et on implémentera un compteur pour déterminer le nombre d'éléments lus dans le fichier. Pour dimensionner le tableau on pourra se limiter à 150 points au maximum.

Cahier des charges

Le programme doit indiquer :

- le nombre de points du fichier,
- la durée du signal,
- la valeur maximale,
- la valeur minimale,
- l'amplitude crête à crête (=valeur maximale – minimale)
- la valeur moyenne du signal,

et si possible (pour les étudiants qui avancent plus vite)

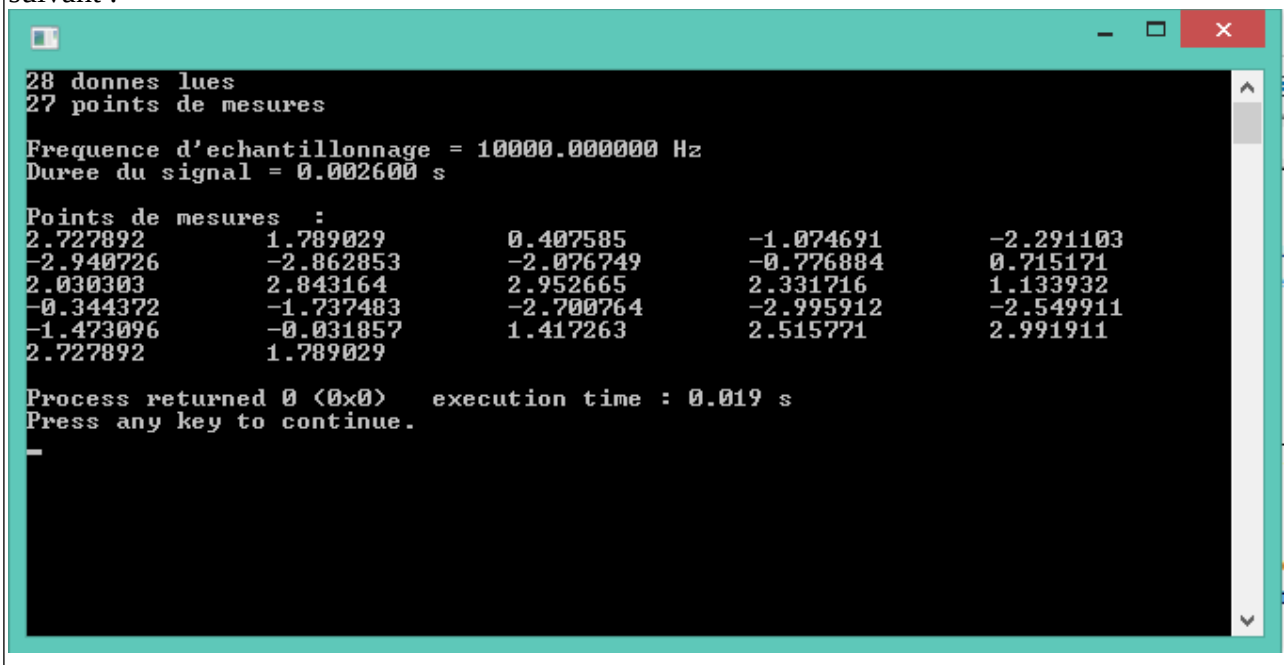
- la valeur efficace du signal (en bonus),
- la fréquence du signal (en bonus)

Pour réaliser le programme complet, vous procéderez par étapes.
!! pour utiliser les fonctions mathématiques, rajouter : #include <math.h> en tête de programme !!

Manipulation n°4

Un programme est donné en annexe. Il permet de lire tous les points de mesure du fichier « signal1.txt » et de les stocker dans un tableau.

Q.1) S'approprier le programme fourni et le compléter afin d'afficher la fréquence d'échantillonnage, les points de mesures et la durée du signal conformément à la copie d'écran suivant :



Q.2) Compléter le programme pour déterminer la plus grande valeur du tableau, puis la plus petite et afficher les résultats «valeur maximale = …, valeur minimale =..., amplitude crête à crête »

Q.3) Valider le fonctionnement de votre programme sur les signaux 2 et 3.

Q.4) Compléter votre programme afin que celui-ci calcule et affiche la valeur moyenne du signal.

Q.5) Bonus (pour ceux qui ont fini en avance!)
Compléter votre programme pour afficher la valeur efficace du signal (rechercher la définition sur internet si besoin).

Q.6) Super bonus
Compléter votre programme en effectuant une détermination de la fréquence du signal. Plusieurs méthodes sont possibles : en repérant la distance entre deux maxima (ou minima) successifs ou en repérant les passages à zéro du signal...

Annexe : Programme permettant de lire tous les éléments du fichier et les ranger dans un tableau.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    FILE* fichier = NULL;
    int i = 0 ;
    int Nmes ; // Nombre de mesures
    int Nmax ; // Nombres d'élements dans le fichier (hors EOF)
    float tablo[151] = {0};
    fichier = fopen("signal1.txt", "r"); // ecrire ici le nom du fichier traité
    if (fichier != NULL)  // si le fichier existe
    {
    while(!feof(fichier)) // tant que l'on a pas atteint la fin du fichier
    {
        fscanf(fichier,"%f", &tablo[i]); // lire chaque élément du fichier
        i ++ ;
    }
        fclose(fichier);
    }
    else
    {
        printf("Probleme acces fichier\n");
    }
        Nmax = i-1 ; // on incrémente i apres affectation donc i-1
        printf( "%d donnes lues \n", Nmax);
        Nmes = Nmax - 1 ; // le premier point est fech
        printf( "%d points de mesures \n", Nmes);

        return 0;
}
```