# Programme Langage C

Fonctions et Procédures

Fichiers

Pointeurs

Les Listes

# Les Sous-Programmes

## Problématique :

Ecrire un programme qui permet de remplir un tableau de n cellules d'entiers. Le programme affiche le tableau avant et après décalage cyclique d'un rang vers la droite.

## Solution :

Pour résoudre ce problème, il faudra :

- Remplir le tableau
- Afficher le tableau
- Décaler les valeurs
- Afficher à nouveau le tableau

On remarque que l'affichage du tableau revient deux fois : ce qui est à éviter. Pour pallier à cela, on utilise les sous-programmes.

## Introduction

Un programme est u ensemble d'instruction pour résoudre un problème. Dès fois, plusieurs instructions sont nécessaires pour effectuer une seule action. En général, il est préférable de regrouper ces instructions dans un bloc : module ou sous-programme.

Les sous-programmes vont nous permettre entre autre de ne pas se répéter dans le code, de l'organiser, d'avoir une meilleure lisibilité...

# Fonctions

## Définition

Un sous-programme est une fonction. Une fonction est sous-programme qui doit obligatoirement retourner un et un seul résultat.

## Syntaxes

### Prototype :

Il représente l'entête de la fonction.

```
Type_de_retour nom_Fonction (paramètres);
```

Remarque

* Les noms des variables des paramètres sont facultatifs pour les prototypes
* Il est impossible de regrouper les paramètres par type

## Corps de la fonction :

Il représente l'ensemble des instructions que la fonction doit exécuter.

```
Type_de_retour nom_Fonction (paramètres)
  {
        Instructions
  }
```

Remarque :

* Les noms de variables sont obligatoires lors de l'implémentation

Il existe deux types de variables avec les sous-programme :

- ✓ Les variables globales : elles sont déclarées en dehors de toutes les fonctions et généralement en haut
- ✓ Les variables locales : elles sont déclarées au sein des fonctions. Elles ont une portée

## Procèdure

C'est une fonction qui ne retourne rien.

Remarque :

* Dans une fonction, on doit obligatoirement avoir le mot return.

* Une procédure commence toujours par void.
* Pour appeler une procédure, il faut écrire le nom de la procédure et les paramètres réels
* Pour appeler une fonction, il faut le faire soit :
    - Dans une expression d'affichage
    - Dans une expression d'affectation
    - Dans une expression de calcul
    - Dans une condition

# Gestion des Fichiers

Problématique :

Ecrire un programme qui permet de saisir le login et le mdp d'un utilisateur. Le programme effectue deux scénarios possibles : le login et le mdp existent et un menu est afficher sinon un message d'erreur sera affiché. Un utilisateur peut enregistrer ses informations pour une éventuelle connexion.

Solution :

Pour résoudre ce problème, il faut :

* Enregistrer les données de l'utilisateur
* Vérifier les données disponibles
* Afficher le menu ou le message d'erreur

Pour enregistrer des données de manière permanente, on utilise les fichiers.

## Introduction

Un fichier est un ensemble de données stockées en général sur un support externe. Un fichier structuré contient une suite d'enregistrement homogène, qui regroupe le plus souvent plusieurs composants appartenant à un ensemble (*champs*). Dans un fichier séquentiel, les enregistrements sont mémorisés consécutivement dans l'ordre de leurs entrées et ne peuvent être lus que dans cet ordre. Si on a besoin d'un enregistrement précis, il faut lire tous les enregistrements qui le précédent.

On peut choisir entre deux modes d'accès :

- Fichier binaire : chaque information est stockée selon les règles de codage imposées par son type. Les données ne peuvent être lues ou écrites que par un programme. La taille du fichier est alors optimale et les données sont facilement lues ou écrites en peu d'instructions.
- Fichier texte : chaque information est sous la forme d'une succession de code ASCII. Les données du fichier peuvent être créer ou consulter à l'aide d'un éditeur de texte ou d'un programme. Les fichier texte sont délicats et plus long à lire.

Quel que soit le mode d'accès envisagé (*binaire ou texte*), on suit toujours la même procédure :

- Ouvrir le fichier
- Lire ou Ecrire dans le fichier
- Fermer le fichier

# Ouverture et Fermeture d'un fichier

## Le pointeur de type FILE

Pour des raisons d'efficacité, les accès à un fichier se font par l'intermédiaire d'une mémoire tampon (*Buffer*). Ce dernier est une zone de la mémoire centrale réservé à un ou plusieurs enregistrements du fichier. L'utilisation de la mémoire tampon a l'effet de réduire le nombre d'accès à la périphérie d'une part et le nombre de mouvements de la tête de lecture/écriture d'autre part.
Pour pouvoir travailler avec un fichier, le programme a besoin d'un certain nombre d'informations au sujet de ce fichier :

- ☞ L'adresse de la mémoire tampon
- ☞ La position actuelle de la tête de lecture/écriture
- ☞ Le type d'accès au fichier : Lecture ou Écriture

Toutes ces informations sont regroupées dans une structure nommée FILE et définies dans stdio.h. Les fonctions liées à la gestion des fichiers ont parmi leurs arguments un pointeur de type FILE.

## Ouverture d'un fichier

### Syntaxe


NomFichier est le nom physique du fichier (*chemin éventuellement inclus*)
Mode indique le mode d'accès choisi et le type de tâches possibles.

La valeur retournée par fopen est un flot de données (*le pointeur de type FILE*). Si l'exécution de cette fonction ne se déroule pas normalement, alors la valeur retournée est le pointeur NULL.
Les différents modes d'accès sont les suivants :

| Mode | Signification |
|---|---|
| r | Ouverture d'un fichier texte en Lecture |
| w | Ouverture d'un fichier texte en Ecriture |
| a | Ouverture d'un fichier texte en Écriture à la fin |
| rb | Ouverture d'un fichier binaire en Lecture |
| rb | Ouverture d'un fichier binaire en Ecriture |
| ab | Ouverture d'un fichier binaire en Ecriture à la fin |
| r+ | Ouverture d'un fichier texte en Lecture/Ecriture |
| w+ | Ouverture d'un fichier texte en Lecture/Ecriture |
| a+ | Ouverture d'un fichier texte en Lecture/Ecriture à la fin |
| r+b | Ouverture d'un fichier binaire en Lecture/Ecriture |
| w+b | Ouverture d'un fichier binaire en Lecture/Ecriture |
| a+b | Ouverture d'un fichier binaire en Lecture/Ecriture à la fin |

## Remarque :

* Si le mode d'ouverture contient la lettre r, le fichier doit exister
* Si le mode contient la lettre w, le fichier peut ne pas exister. Dans ce cas, il sera créé. Si le fichier existait, son ancien contenu serait écrasé.
* Si le mode contient la lettre a, le fichier peut ne pas exister, dans ce cas il sera créé. Sinon les nouvelles informations seront ajoutées à la fin de son contenu.

## Fermeture d'un fichier

### Syntaxe

```
fclose (flot)
```

Flot représente la variable de type FILE.

Cette fonction permet de fermer le flot de données qui a été associé à un fichier par la fonction fopen. Elle retourne 0 si le fichier s'est fermé normalement.

# Les Entrées/Sorties Formatées

## La fonction d'écriture (fprintf)

La fonction fprintf permet d'écrire des données dans un fichier

### Syntaxe

fprintf (fichier, "code(s) format", expression(s))

Fichier est le flot de données retourné par la fonction fopen

Ex:

Ecrire un programme qui permet d'écrire dans un fichier texte des produits. Le nombre de produits est saisi par l'user. Produit (code, libelle, prix, quantité).

```
main()
{
        produit p;
        int n, i;
        FILE *fich;
        do
        {
                puts ("entrer le nombre de produit : ");
                scanf ("%d", &n);
        } while(n<=0);
        fich = fopen ("produits.txt", "w");
        if (fich != null)
        {
                For (i=0; i<n; i++)
                {
                        fflush(stdin);
```

```
                puts("libelle : ");
                gets(p.libelle);
                puts("code :");
                scanf("%d", &p.code);
                puts("prix et quantité : ");
                scanf("%d %d", &p.prix, &p.qte);
                fprintf(fich, "%d %s %d %d\n", p.code, p.libelle, p.prix, p.qte);
        }
    }else
    {
        puts("erreur!!!");
    }
    fclose(fich);
}
```

## Lecture d'un fichier (fscanf)

Cette fonction permet de lire les données d'un fichier

fscanf (fichier, "code(s) format", variable(s))

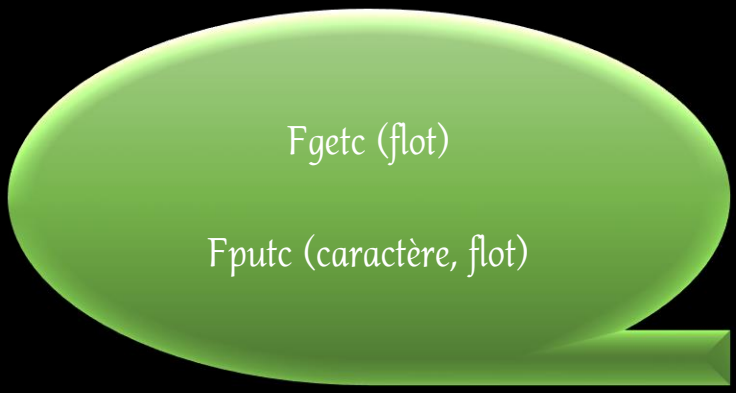
Ex:

Ecrire un programme qui permet de lire un fichier créé par le programme précédent

## Ecriture et Lecture de caractères

Les fonctions fgetc et fputc permettent respectivement de lire et d'écrire un caractère dans un fichier. La première fonction retourne le caractère lu dans le fichier tandis que la seconde écrit le caractère dans le flot de données.

### Syntaxe




Fputc retourne l'entier correspondant au caractère lu.
Pour les deux fonctions, EOF est retourné en cas de fin fichier pour fgetc et en cas d'erreur pour fputc

## Application :

Ecrire un programme qui permet de copier le contenu d'un fichier txt vers un autre. Le programme demande à l'utilisateur le nom des deux fichiers.

## Ecriture et Lecture de chaines (fgets et fputs)

Les fonctions fputs et fgets permettent respectivement d'écrire et de lire dans un fichier. L'objet écrit ou lu est de type chaine de caractères.

### Syntaxe

```
fgets (chaine, nbre_car, flot);
fputs (chaine, flot);
```

## Application :

Ecrire un programme qui permet de créer un fichier txt et un autre qui permet de lire ce fichier.

```
#include <stdio.h>

Main()
```

```
{
        FILE *f ;
        chaine text[100] ;

        f = fopen (“texte.txt”, “w”);
        if (f)
        {
                puts (“Entrer une chaine : “);
                gets (text);
                fputs (text, f);
                fclose(f);
        }
        else
        {
                puts (“erreur!!!”);
        }
}
```

```
#include <stdio.h>

main ()
{
        FILE *f;
        chaine text [100];

        f = fopen (“texte.txt”, “r”);
        if (f)
        {
                fgets (text, 100, f);
                printf ("%s", text
                fclose(f);
```

```
        }
        else
        {
                puts (”erreur!!!”);
        }
}
```

## Les Entrées/Sorties non Formatées

L'écriture non formatée se fait par bloc (enregistrement). Les fonctions fwrite et fread sont respectivement utilisées pour écrire et lire des blocs de données dans un fichier.

### Syntaxes

```
fwrite (&variable, taille, nbre_bloc, flot);
fread (&variable, taille, nbre_bloc, flot);
```

### Application :

Ecrire un programme qui permet de créer un fichier d'articles.

Article (code, libellé, prix, quantité). La saisie s'arrête lorsque code = 0.

### Application 2 :

Ecrire un programme qui permet de lire le fichier créé.

## Ftell – fseek

Ftell renvoie la position de la tête d'écriture dans un fichier

Fseek permet de lire un bloc dans le fichier de par sa position

Fseek (flot, nbre_bloc, position);

Position:

- SEEK_SET : renvoie 0 ; le début du fichier
- SEEK_CUR : renvoie 1 ; la position actuelle du curseur
- SEEK_END : renvoie 2 ; à la fin du fichier