

Programmation en C/C++

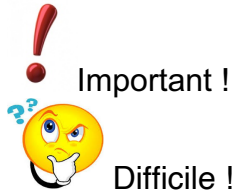
Série 10

Lecture / écriture dans des fichiers textes ou binaires

@ G.CANESI Le Cham

Document pour un usage personnel uniquement

Les fichiers binaires ne sont pas au programme de l'examen



Objectifs

Savoir lire et écrire des données dans un fichier texte ou dans un fichier binaire enregistré sur le disque dur ou une clé USB. Savoir supprimer ou renommer des fichiers sur un disque.

Introduction

Jusqu'à présent nous avons créé des programmes qui demandaient de saisir des données au clavier, stockaient celles-ci dans des variables puis affichaient des résultats à l'écran. En fin de programme toutes ces données étaient perdues.

Il peut être intéressant d'utiliser des données qui sont présentes dans un fichier texte déjà existant au lieu de les saisir au clavier (**opération de lecture** dans un fichier) et d'enregistrer les résultats obtenus dans un fichier texte (**opération d'écriture** dans un fichier) stocké sur un disque.

Un fichier (*file* en anglais) est un ensemble de données stockées sur un support physique (disque dur, clé USB...). Les fichiers textes sont des fichiers séquentiels où les données sont enregistrées consécutivement les unes derrière les autres. Attention ! Leurs données ne pourront être lues que séquentiellement, c'est-à-dire dans ce même ordre : il faudra, par exemple, parcourir les 5 premiers caractères pour atteindre le sixième ! Il faudra donc connaître la structure d'un fichier texte pour en extraire ses données.



Nous aurons besoin de la librairie *stdio.h* ; pensez donc à inclure en tête de votre programme C, la directive *#include <stdio.h>*. *stdio.h* permet, entre autres, de gérer la fonction *fopen()* d'ouverture des fichiers.



Nous aurons besoin d'un pointeur vers le fichier texte (ce pointeur est une variable ayant en mémoire l'adresse de début du fichier texte).
Le fichier texte sera ouvert soit en mode écriture soit en mode lecture.



Attention ! Nous utilisons des fichiers enregistrés sur le disque dur ou une clé USB, dans cette série. Nous aurons donc besoin de connaître les chemins d'accès à ces fichiers (dossier exact où est placé le fichier texte d'extension *.txt*). Dans tous les exemples et corrections de cette série les chemins d'accès correspondent à une situation réelle, mais ils devront être modifiés et adaptés pour fonctionner sur votre propre ordinateur.

Sous Linux (salle machine)

Gestion de fichiers



Cliquer sur le caméléon tout en haut à droite de l'écran, puis sur *Editeurs* puis sur *Fichiers*. Un *clic droit* sur le fichier suivi d'un clic sur *Propriétés* vous permettra de connaître le chemin d'accès à votre fichier.

Exemple de chemin sur les machines de l'établissement :

/home/licencep/Bureau/mesSources/Fichier1.txt

Chemin d'accès par défaut

Si vous ne précisez pas le chemin d'accès, le fichier texte est créé là où se trouve votre projet *Codeblocks*. Pour connaître le dossier où est stocké votre projet : mettez-vous sur votre programme C sous *Codeblocks* puis faire *File* puis *Save File As* et vous verrez le chemin : par exemple */home/licencep/Bureau/mesSources/*

Attention ! Sous Linux, si l'on souhaite écrire dans un fichier sans préciser le chemin d'accès, si le fichier n'existe pas, Linux le crée dans le dossier où est placé le programme C que vous exécutez.

1- Fichiers texte

Écriture dans un fichier texte

Étape 1 : création d'un fichier de "texte brut"

Nous allons dans un premier temps créer un fichier texte (texte brut) nommé "*Fichier1.txt*" et le placer dans le dossier où se trouve le programme C exécutable.

Ouvrir un éditeur quelconque de texte type *Wordpad*. Taper 2 nombres entiers l'un en dessous de l'autre et enregistrer le fichier (au format texte) dans le bon dossier sous le nom *Fichier1.txt*.

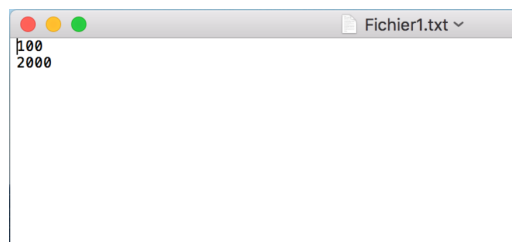
Éditeur de texte sous Linux :



Cliquer sur le caméléon tout en haut à droite de l'écran, puis sur *Éditeurs* puis sur *Kate* qui est un éditeur de texte basique. Tapez vos données et faites *Fichier* puis *Enregistrer sous* ; choisissez le dossier de sauvegarde et précisez bien le nom et l'extension ".txt" de votre fichier (exemple : *Fichier1.txt*)



Attention ! Le fichier doit posséder le format "texte brut" (extension .txt). Le nom du fichier est *Fichier1*, son extension est .txt (fichier de format texte brut). Ne pas enregistrer au format .docx ou .doc de Word.



Le fichier *Fichier1.txt*



Le fichier texte devra être ouvert en mode écriture :
a (on écrit à la fin du fichier ; tout le contenu éventuellement existant est gardé),
w (**attention, danger !** Le contenu existant dans le fichier est entièrement effacé et on écrit dedans à nouveau).

Exemple : `fic = fopen("Fichier1.txt", "w");` notion abordée plus loin.

Ici, on ouvre *Fichier1.txt* en mode écriture.

fic est un pointeur qui stocke l'adresse de début du fichier texte.

Étape 2 : Écriture dans un fichier avec *fputc*



Instruction d'écriture *fputc*

L'instruction *fputc* permet d'écrire un caractère dans le fichier texte (*Fputc* : **file put char**).

Syntaxe : `fputc('x', fic);` x est le caractère à insérer ; *fic* est un pointeur sur le fichier texte (voir plus loin).

Exemple de programme avec *fputc*

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

```

int main() {

FILE* fic = NULL;
fic = fopen("Fichier1.txt", "a");

if (fic != NULL) {
    fputc('1', fic);
    fclose (fic);
}
else
    printf("Ouverture impossible");

return 0;
}

```

Étape 3 : Exécution du programme et vérification

Le chiffre 1 a bien été écrit à la fin de *Fichier1.txt*

Le mode "a" sur la ligne *fic = fopen("Fichier1.txt", "a");* signifie écrire à la fin du fichier. Si le "a" avait été remplacé par un "w", tout aurait été écrasé (effacé) dans le fichier et il serait resté uniquement le 1.

Fichier1.txt doit être placé ici dans le même dossier que le programme exécutable.

Explications sur le programme

// on crée un pointeur nommé fic ; ce pointeur aura en mémoire l'adresse d'un fichier (FILE) ; l'adresse est vide pour le moment.

FILE fic = NULL;*

// on associe le pointeur fic au fichier nommé Fichier1.txt ; on ouvre le fichier en mode "écriture à la fin du fichier"

fic = fopen("Fichier1.txt", "a");

// si le pointeur a bien une adresse en mémoire (l'adresse du fichier texte !)...

if (fic != NULL) {

// on écrit 1 dans le fichier (à la fin)

fputc('1', fic);

// on ferme le fichier et on libère le pointeur fic

fclose (fic);

}

else

printf("Ouverture impossible");

return 0;

}



Instruction d'écriture *fputs*

fputs permet d'écrire une chaîne de caractère dans un fichier (*Fputs* : **file put string**).

Syntaxe : *fputs*("xxx", *fic*); *xxx* est la chaîne à insérer ; *fic* est un pointeur sur le fichier texte.

Exemple

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

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

```
int main(){
```

```
FILE* fic = NULL;
```

```
fic = fopen("Fichier1.txt", "w");
```

```
if (fic != NULL) {
    fputs("Première écriture dans un fichier", fic);
    fclose(fic);
}
```

```
else
    printf("Ouverture impossible");
```

```
return 0;
}
```

Exécution



Remarque

Ne pas utiliser de caractères accentués qui peuvent poser problème.

Les données ont été effacées et remplacée par "Premiere ecriture dans un fichier" (on est en mode "w" : ouverture d'un fichier texte en mode écriture avec effacement de ses données, s'il y en a).



Instruction d'écriture *fprintf*

fprint permet d'écrire du texte, des valeurs comme la fonction *printf* (**File print**)

Syntaxe : *fprintf*(*fic*, "x vaut : %d", *Var1*);

fic est un pointeur sur le fichier texte, dans cet exemple, on écrira : x vaut : suivi du nombre entier contenu dans la variable *Var1*.

Exemple

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

```
#include<stdlib.h>

int main() {

int Annee;
FILE* fic = NULL;
fic = fopen("Fichier1.txt","w");

if (fic != NULL) {
    printf("Quelle est l'annee en cours ?");
    scanf ("%d", &Annee);

    // Ecriture dans le fichier
    fprintf(fic, "Annee actuelle : %d", Annee);

    fclose (fic);
}
else
    printf("Ouverture impossible");

return 0;
}
```

Exécution



Remarque

Ne pas utiliser de caractères accentués qui peuvent poser problème.

Lecture dans un fichier texte

Ces opérations vont permettre de récupérer des données présentes dans un fichier texte ; on dira que l'on lit les données du fichier, par opposition aux écritures vues précédemment.

Nous utiliserons trois instructions différentes :

fgetc : lit un caractère,

fgets : lit une chaîne de caractères,

fscanf : joue le même rôle que *scanf* à partir de données présentes dans des fichiers plutôt que saisies au clavier.



Le fichier texte devra être ouvert en mode lecture *r* (*read*)

Étape préalable : Création d'un fichier de "texte brut"

Nous allons dans un premier temps créer un fichier texte (texte brut) nommé "Fichier1.txt" et le placer sur le bureau de l'ordinateur.

Ouvrir un éditeur quelconque de texte type *Wordpad*. Taper le texte suivant : "Apprentissage du langage C" puis enregistrer le fichier sur le bureau sous le nom *Fichier1.txt*.



Le fichier texte devra être ouvert en mode lecture (*r* pour *read* ; le fichier doit exister !).

Exemple : `fic = fopen("Fichier1.txt", "r");`

Exemple 1 : utilisation de `fgetc()`

Syntaxe de `fgetc()` : `VAR = fgetc(fic);`

Dans cet exemple, on lit un caractère dans le fichier pointé par `fic` et on l'affecte à la variable `VAR`. `VAR` aura le type `int` (code ASCII).

```
#include<stdio.h>
#include<stdlib.h>
int main() {
    int caract; FILE* fic = NULL;

    fic = fopen("Fichier1.txt", "r");

    if (fic != NULL) {
        do{
            caract = fgetc(fic);
            printf("%c", caract);
        } while (caract != EOF);

        printf("\n");
        fclose (fic);
    }
    else
        printf("Ouverture impossible");

    return 0;
}
```

Exemple 2 : utilisation de `fgets()`



Remarque

`fgets()` retourne une chaîne de caractères.

Syntaxe de `fgets()` : `fgets(VAR, 100, fic);`

Dans cet exemple, on lit une chaîne de 100 caractères dans le fichier pointé par `fic` et on l'affecte à la variable `VAR`.

Exemple d'implémentation

```

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

int main() {

char Chaîne[100];
FILE* fic = NULL;

fic = fopen("Fichier1.txt","r");

if (fic != NULL) {
    fgets(Chaîne, 100, fic);
    printf("%s\n",Chaîne);
    fclose (fic);
}
else
    printf("Ouverture impossible");

return 0;
}

```

Exemple 3 : utilisation de **fscanf()**

Syntaxe de *fscanf()* : *fscanf(fic, "%d", &Var1);*

Dans cet exemple, on lit un entier (%d) présent dans le fichier texte pointé par *fic* et on l'affecte à la variable *Var1*.

On travaillera à partir du fichier texte *Fichier1.txt* dans lequel sont enregistrés 2 nombres réels. On cherche à remonter ces 2 nombres dans deux variables que l'on affichera ensuite.

```

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

int main(){

double V1,V2;
FILE* fic = NULL;

fic = fopen("Fichier1.txt","r");

if (fic != NULL) {
    fscanf(fic, "%lg %lg", &V1,&V2);
    fclose (fic);
}
else
    printf("Ouverture impossible");

printf("%lg\t%lg ", V1,V2);

```



```
return 0;
}
```

Remarque

Il faut déclarer les 2 variables réelles puis utiliser le *fscanf* avec deux fois *%lg*.



Différents modes d'ouverture des fichiers

r (mode lecture ; le fichier doit donc exister !)

a (mode écriture ; on écrit à la fin du fichier ; le fichier peut ne pas exister),

w (mode écriture ; on efface le contenu existant dans le fichier et on écrit dedans à nouveau ; le fichier peut ne pas exister).

r+ (mode lecture / écriture) en effaçant l'existant



Position du curseur dans un fichier texte

Il existe quelques fonctions bien pratiques pour se situer et déplacer dans un fichier texte.

ftell La fonction *ftell* renvoie un entier long (comptabilisé en nombre d'octets) correspondant à la position actuelle du curseur dans le fichier texte par rapport au début du fichier.

Prototype : *long ftell (FILE* file);*

fseek La fonction *fseek* permet de déplacer le curseur à partir d'une position connue

Prototype : *int fseek (FILE* file, long décalage, int positionDebut);*

seek_set Indique le début du fichier

seek_end Indique la fin du fichier

seek_cur Indique la position courante dans le fichier

Exemples

fseek(file, 10, SEEK_CUR) déplace le curseur de 10 caractères vers la droite à partir de sa position courante,

fseek(file, -10, SEEK_CUR) déplace le curseur de 10 caractères vers la gauche à partir de sa position courante,

fseek(file, 0, SEEK_END) déplace le curseur à la fin du fichier,

fseek(file, 0, SEEK_SET) déplace le curseur au début du fichier,

fseek(file, 10, SEEK_SET) déplace le curseur 10 caractères après le début du fichier.

Attention ! Si on place le curseur à un endroit donné dans le texte et que l'on écrit dans le fichier, le texte "écrasera" l'ancien texte présent à cet endroit.

Notion de fin de chaîne et de fin de ligne avec \0 ou \n

La fin d'une chaîne est spécifiée par le caractère de contrôle \0.

La fin de ligne est spécifiée par le caractère de contrôle \n.

Une tabulation est spécifiée par le caractère de contrôle \t.

Ces caractères de contrôle posent des problèmes lorsqu'on lit tout un texte contenu dans un fichier (des caractères classiques ainsi que des caractères de contrôle seront "lus" et il faudra donc faire la différence entre eux !).

```
while (car != '\n')  
{...}
```

Notion de fin de fichier avec EOF et feof

La fin d'un fichier est marquée par le symbole **EOF** (*End Of File*).

La fonction **feof** permet de rechercher la fin d'un fichier : elle signifie (*Find End Of File*).

feof a pour format : *feof(fic)* où *fic* est un pointeur sur un fichier.

feof renvoie vrai si la fin du fichier est atteinte.

On pourra donc parcourir un fichier tant qu'on n'a pas atteint la fin de celui-ci avec une boucle du type : *while (!feof(fic))...* (Tant que l'on n'est pas en fin de fichier...) Pratique pour le lire entièrement...

Notion de chemin d'accès

Si l'on ne précise pas son chemin d'accès, le fichier texte devra être placé sur le même disque et dans le même dossier que le programme exécutable.

On peut utiliser en lecture ou en écriture un fichier texte présent à un autre endroit et/ou sur un autre disque (disque dur, clé USB...).

Il faudra alors spécifier le chemin d'accès à ce disque afin d'accéder en lecture ou écriture au fichier texte.

Exemples de chemins d'accès

- **Sous Windows**

C:\sourcesC\Test1.txt\

- **Sous iOS**

Macintosh HD\Utilisateurs\Alpha\Bureau\Fichier1.txt\

- **Sous Linux**

/home/licencep/Bureau/mesSources/Fichier1.txt

Renommer et supprimer des fichiers sur un disque

rename et *remove* sont deux fonctions qui permettent respectivement de renommer et supprimer des fichiers présents sur un disque.

Prototypes

```
int rename(const char* ancienNom, const char* nouveauNom);  
int remove(const char* fichierASupprimer);
```

2- Fichiers binaires

Il est possible de travailler avec des fichiers binaires. En fait, tous les fichiers sont binaires et constitués de 0 et de 1. Mais là nous parlerons de fichiers binaires par opposition aux fichiers textes. Dans un fichier texte on écrit les données sous forme textuelle (compréhensible par l'humain) et le fichier est enregistré (collection de 0 et de 1). Dans un fichier binaire on écrira les données directement en binaire puis le fichier sera enregistré (collection de 0 et de 1).

Avantage des fichiers binaires

Une écriture de données en binaire prendra beaucoup moins de place qu'une écriture avec des codes ASCII (1 octet par caractère). Du coup, la manipulation de fichiers binaires est plus rapide que celle des fichiers textes.

On peut écrire des blocs entiers de données dans un fichier binaire contrairement aux fichiers textes on peut également remonter des blocs entiers de données à partir d'un fichier binaire.

Inconvénients des fichiers binaires

Contrairement aux fichiers textes, les fichiers binaires ne peuvent être lus que par des éditeurs spécifiques. Ils restent incompréhensibles puisqu'écrits en binaire à moins de connaître l'ordonnancement des données.

Écrire dans un fichier binaire

Pour écrire dans un fichier binaire on utilisera le mode "wb" (les données sont écrasées) ou le mode "ab" (les données sont écrites à la fin du fichier) avec le *fopen()* :

Exemple : *Fic = fopen("Essai.bin", "wb");*

Pour écrire dans le fichier binaire ouvert on utilisera *fwrite()*.

fwrite (adresse où sont les données à écrire, nombre d'octets à écrire , nombre de fois où l'on répète l'opération, adresse où l'on se situe dans le fichier)

Exemple : *fwrite(&a, sizeof(int), 1, Fic);*

Libération de la mémoire

On terminera par un *fclose()* qui libère l'espace mémoire

Exemple : *fclose(Fic);*

Lire un fichier binaire

Pour lire un fichier binaire on utilisera le mode "rb" avec le *fopen()* :

Exemple : *Fic = fopen("Essai.bin", "rb");*

Pour lire un fichier binaire on utilisera *fread()*.

fread (adresse où l'on va stocker les données lues , nombre d'octets à lire , nombre de fois où l'on répète l'opération, adresse où l'on se situe dans le fichier)

Exemple : *fread(&a, sizeof(int),1,Fic);*

Exercice de cours 1 avec un entier et un réel

Écrire un programme qui appelle une fonction *EcritBin* () qui va créer un fichier binaire *Essai.bin* et y stocker les nombres 10 et 3.14. Le programme appellera ensuite une fonction *LitBin*() qui va lire le fichier binaire.

Vous pouvez télécharger un éditeur de fichier binaire pour visualiser ce dernier. Repérez-vous l'entier et le réel ?

Corrigé

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

```
void EcritBin ()
```

```
{
```

```
FILE * Fic;
```

```
int a = 10;
```

```
double b = 3.14;
```

```
printf("Écriture du fichier binaire\n");
```

```
Fic = fopen("Essai.bin","wb");
```

```
fwrite(&a, sizeof(int),1,Fic);
```

```
fwrite(&b, sizeof(double),1,Fic);
```

```
fclose(Fic);
```

```
}
```

```
void LitBin()
```

```
{
```

```
int a;
```

```
double b;
```

```
FILE* Fic;
```

```
printf("Lecture du fichier binaire\n");
```

```
Fic = fopen("Essai.bin","rb");
```

```
fread(&a, sizeof(int),1,Fic);
```

```
fread(&b, sizeof(double),1,Fic);
```

```
printf("a : %d \t b : %lf \n", a, b);
```

```
fclose(Fic);
```

```
}
```

```
int main()
```

```
{
```

```
EcritBin () ;
```

```
LitBin();
```

```
return 0;
```


```
}
```

Exercice de cours 2 avec un tableau de double

Écrire un programme qui affecte 20 valeurs réelles (de $0 \cdot 0.2$ à $19 \cdot 0.2$) dans un tableau de *double* T1. Puis recopie d'un seul coup, T1 dans un fichier binaire nommé *TOTO.bin*.

Le programme, dans un second temps, recopie les données du fichier binaire dans un tableau de *double* T2 puis affiche les valeurs réelles contenues dans T2.

Corrigé



```
0.000000
0.200000
0.400000
0.600000
0.800000
1.000000
1.200000
1.400000
1.600000
1.800000
2.000000
2.200000
2.400000
2.600000
2.800000
3.000000
3.200000
3.400000
```

Affichage à l'écran

```
# define MAX 20
# include <stdio.h>
```

```
int main()
```

```
{
```

```
int i;
```

```
double T1[MAX], T2[MAX];
```

```
FILE *fic;
```

```
for(i=0; i<MAX; i++)
```

```
    T1[i] = i*0.2;
```

```
// Ecriture dans le fichier binaire
```

```
fic = fopen("Toto.bin", "wb");
```

```
if (fic != NULL)
```

```
{
```

```
    fwrite(T1, sizeof(double), MAX, fic);
```

```
    fclose(fic);
```

```
}
```

```
else
```

```
    printf("Erreur\n ");
```

```
// Lecture du fichier binaire
```

```
fic = fopen("Toto.bin", "rb");
if (fic != NULL)
{
    fread(T2, sizeof(double), MAX, fic);
    fclose(fic);

    // Affichage de T2
    for(i=0 ; i<MAX ; i++)
        printf("%f\n ", T2[i]);
}

else
    printf("Erreur de lecture\n ");

return 0;
}
```

EXERCICES

Exercice 1

Énoncé

Écrire un caractère saisi au clavier dans un fichier texte nommé Test1.txt.

Corrigé

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

```
int main() {
    char lettre;
    FILE* fic = NULL;
```

```
    printf("Saisir une lettre");
    scanf("%c",&lettre);
```

```
    fic = fopen("Test1.txt","a");
```

```
    if (fic != NULL) {
        fputc(lettre, fic);
        fclose (fic);
        printf("Lettre ecrite");
    }
```

```
    else
        printf("Ouverture impossible");
```

```
    return 0;
}
```

Exercice 2

Énoncé

Écrire un mot saisi au clavier dans un fichier texte nommé Test2.txt.

Ce programme ne prend pas en charge une phrase de plusieurs mots séparés par des espaces.

Corrigé

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

```
int main(){
    char Mot [100] ;
    FILE* fic = NULL;
```

```
    printf("Saisir une phrase complete");
    scanf("%s",&Mot);
```

```

fic = fopen("Test2.txt", "a");

if (fic != NULL) {
    fputs(Mot, fic);
    fclose (fic);
    printf("Phrase enregistrée");
}
else
    printf("Ouverture impossible");

return 0;
}

```

Exercice 3

Énoncé

Écrire à la suite, l'un en dessous de l'autre, les chiffres 1 et 2 dans un fichier texte nommé *Test2.txt*.

Corrigé

```

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

int main(){

FILE* fic = NULL;
fic = fopen("Test2.txt", "a");

if (fic != NULL) {
    fputc('1', fic);
    fputs("\n", fic);
    fputc('2', fic);

    fclose (fic);
}
else
    printf("Ouverture impossible");

return 0;
}

```

Exercice 4

Énoncé

Écrire à la volée 5 entiers saisis au clavier, à la suite, sur une même ligne, dans un fichier texte nommé *Test2.txt*. Les 5 nombres seront séparés par une espace.

Corrigé

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

```
int main()
{
    int Nombre,i ;
    FILE* fic = NULL;
    fic = fopen("Test2.txt","a");

    if (fic != NULL)
    {

        for (i=1 ; i<= 5 ; i++)
        {
            printf("Saisir un entier");
            scanf("%d",&Nombre);
            fprintf(fic, "%d ", Nombre); // attention espace après le %d
        }

        fclose (fic);
        printf("Nombre enregistre");
    }

    else
        printf("Ouverture impossible");

    return 0;

}
```

Exercice 5

Énoncé

Saisir au clavier 5 entiers qui seront stockés dans un tableau à l'aide d'une boucle. Puis, les écrire les uns en dessous des autres dans un fichier texte.

Corrigé

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

```
int main()
{
    int T[5],i ;
    FILE* fic = NULL;

    for (i=0 ; i< 5 ; i++)
    {
        printf("Saisir un entier");
```

```

        scanf("%d",&T[i]);
    }

    fic = fopen("Test2.txt","a");
    if (fic != NULL)
    {
        for (i=0 ; i< 5 ; i++)
        {
            fprintf(fic, "%d\n", T[i]);
        }

        fclose (fic);
        printf("Nombres enregistrés");
    }

    else
        printf("Ouverture impossible");

    return 0;
}

```

Exercice 6



Énoncé

Saisir et enregistrer 6 nombres entiers dans un tableau. Puis, écrire les nombres stockés dans ce tableau dans un fichier texte, les uns en dessous des autres. Enfin, le programme appellera une fonction qui retourne la moyenne de ces nombres. Le programme écrira la valeur retournée (réel !) sur la ligne en dessous des 6 nombres.

Corrigé

```

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

float MOY (int T [], int max)
{
    float somme = 0.0;
    int i ;

    for (i=0 ; i< max ; i++)
    {
        somme+=T[i];
    }
    return (somme/max);
}

```

```

int main()
{
    int T[6],i ;

```

```
FILE* fic = NULL;
```

```
for (i=0 ; i< 6 ; i++)
```

```
{  
    printf("Saisir un entier");  
    scanf("%d",&T[i]);  
}
```

```
fic = fopen("Test2.txt", "a");
```

```
if (fic != NULL)
```

```
{  
    for (i=0 ; i< 6 ; i++)  
    {  
        fprintf(fic, "%d\n", T[i]);  
    }
```

```
    fprintf(fic, "Moyenne : %.2f\n", MOY(T,6));  
    fclose (fic);
```

```
    printf("Nombres enregistrés");  
}
```

```
else
```

```
    printf("Ouverture impossible");
```

```
return 0;
```

```
}
```

Exercice 7

Énoncé

Écrire le programme qui demande le nombre de réels devant être saisis au clavier. Le programme demande ensuite leur saisie et les écrit, à la volée (saisie puis écriture juste après pour chaque nombre), dans un fichier texte, les uns à la suite des autres (séparés par une espace). On n'utilisera pas la structure de tableau.

Corrigé

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

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

```
int main()
```

```
{
```

```
int NBelements;
```

```
int i ;
```

```
float Reel;
```

```
FILE* fic = NULL;
```

```
printf("Combien de reels voulez-vous saisir ?");
```

```

scanf("%d",&NBelements);

fic = fopen("Test2.txt","a");
if (fic != NULL)
{
    for (i=0 ; i< NBelements ; i++)
    {
        printf("Saisir un reel");
        scanf("%f",&Reel);

        fprintf(fic, "%.2f ", Reel);
    }

    fclose (fic);
    printf("Nombres enregistres");
}

else
    printf("Ouverture impossible");

return 0;
}

```

Exercice 8

Énoncé

On travaillera à partir d'un fichier texte Fichier1.txt dans lequel sont enregistrés 3 nombres réels les uns à la suite des autres et séparés par une espace. Écrire le programme qui affiche ces nombres à la volée.

Corrigé

```

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

int main()
{

int i ;
float Reel;
FILE* fic = NULL;

fic = fopen("Test2.txt","r+");
if (fic != NULL)
{
    for (i=0 ; i<3 ; i++)
    {
        printf("Reel numero %d : ",(i+1));
        fscanf(fic, "%f ", &Reel);
        printf ("%.2f \n", Reel);
    }
}

```

```

        fclose (fic);
    }
else
    printf("Ouverture impossible");

return 0;
}

```

Remarque

Ce programme fonctionne que les 3 nombres soient séparés par un ou plusieurs espace(s) ou les uns en dessous des autres. Le curseur se décale au réel suivant à chaque tour de boucle.

Exercice 9

Énoncé

On travaillera à partir d'un fichier texte Fichier1.txt dans lequel seront enregistrés 3 nombres entiers séparés pas des espaces sur la première ligne et 2 autres nombres entiers sur la seconde.

Écrire le programme qui remonte ces 5 nombres dans un tableau. Le tableau est ensuite affiché.

Corrigé

```

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

int main(){

    int i, T[5];
    FILE* fic = NULL;

    fic = fopen("Fichier1.txt", "r");

    if (fic != NULL) {
        for (i=0; i<5; i++)
            fscanf(fic, "%d", &T[i]);

        fclose (fic);
    }
    else
        printf("Ouverture impossible");

    for (i=0; i<5; i++)
        printf("%d\t ", T[i]);

    return 0;
}

```

Remarque

Peu importe les espaces ou les retours à la ligne entre les nombres : ils seront remontés. Le programme "passe" d'un entier à l'autre.

Exercice 10

Énoncé

Écrire un programme qui lit et affiche tout le contenu d'un fichier texte que vous aurez créé préalablement et enregistré sur le disque dur.

Corrigé

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    FILE *fic;
    fic = fopen("Test3.txt", "r");
    int c;
    if (!fic)
    {
        printf("Impossible d'ouvrir le fichier texte ; sortie du programme\n");
        exit(-1); // on sort du programme
    }

    //Tant que l'on n'est pas en fin de fichier...
    while (!feof(fic)) {
        c=fgetc(fic);
        //si le caractère est différent de celui de fin de fichier on l'affiche
        if (c!=EOF)
            printf("%c",c);
    }

    printf("\n");
    fclose(fic);
    return 0;
}
```

Remarque

On est obligé de tester si la variable c contient le caractère de "fin de fichier (EOF)" avant de l'afficher ou pas.

Exercice 11

Énoncé

Écrire un programme qui demande de saisir une phrase au clavier puis l'écrit dans un fichier texte. Ne pouvant pas utiliser *scanf* avec %s car elle ne gère pas les espaces, déclarez plutôt un tableau (*Tab*) de caractères et utilisez la fonction *gets* (*Tab*).

Corrigé

```

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

int main(){
    FILE *fic;
    char Tab[150];
    fic = fopen("File1.txt","w");

    if(fic !=NULL){
        printf("saisissez une phrase : ");
        gets(Tab);
        printf("Phrase saisie : %s\t ",Tab);
        fputs(Tab,fic);
        fclose(fic);
    }
    else {
        printf("Ouverture impossible");
    }

    return 0;
}

```



Exercice 11 bis

Énoncé

Écrire un programme qui demande de saisir une phrase au clavier puis l'écrit dans un fichier texte. Vous n'utiliserez pas, cette fois-ci, de tableau de caractères mais la fonction `getchar()`.

Corrigé

```

#include<stdio.h>
#include<stdlib.h>
int main() {
    char Mot ;
    FILE* fic = NULL;
    fic = fopen("Test2.txt","w");

    if (fic != NULL) {
        printf("Saisir une phrase complete");

        //tant que le caractère est différent de retour à ligne et différent de fin de fichier on saisit...
        while ( (Mot=getchar()) != '\n' && Mot != EOF)
        {
            fputc(Mot, fic);
        }
    }
}

```

```

else
    printf("Ouverture impossible");

fclose (fic);
printf("Phrase enregistree");

return 0;
}

```

Exercice 12

Énoncé

Écrire une phrase dans un fichier texte nommé *Test2.txt*. Écrire ensuite le programme qui affiche le nombre de caractères présents dans cette phrase.

Corrigé

```

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

```

```

int main()
{
    int NBcar = 0;
    FILE *fic;
    fic = fopen("Test2.txt", "r");

    if (!fic)
    {
        printf("Impossible d'ouvrir le fichier texte ; sortie du programme\n");
        exit(-1);
    }

    while (!feof(fic))
    {
        fgetc(fic);
        NBcar++;
    }

    printf("nombre de caracteres dans le fichier : %d", (NBcar-1));

    fclose(fic);
    return 0;
}

```

Remarque

On enlève 1 pour tenir compte du caractère de fin de fichier.

Exercice 13

Énoncé

Écrire un programme qui affiche le 5^{ème} caractère présent dans un fichier texte qui comporte une phrase.

Corrigé

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

```
int main()
{
    char LeChar;
    int NBcar = 0;
    FILE *fic;
    fic = fopen("Test2.txt", "r");

    if (!fic)
    {
        printf("Impossible d'ouvrir le fichier texte ; sortie du programme\n");
        exit(-1);
    }

    while (!feof(fic))
    {
        LeChar = fgetc(fic);
        NBcar++;
        if (NBcar == 5)
        {
            printf("5ieme element : %c", LeChar);
            exit (0);
        }
    }

    fclose(fic);
    return 0;
}
```

Remarque

On a mis un "exit" ; on pourra améliorer le programme pour demander d'afficher un caractère situé à un rang donné.



Exercice 14

Énoncé

Écrire une phrase dans un fichier texte ; enregistrez-le. Écrire ensuite le programme qui appelle une fonction qui affiche cette phrase et renvoie le nombre de mots dans cette phrase.

Penser à calculer le nombre d'espaces...

Corrigé

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

```
int NbMot(FILE *fic);
```

```
int main() {
    FILE *fic;
    fic = fopen("Test3.txt", "r");
    if (fic != NULL){
        printf("le nombre de mot est : %d\n ", NbMot(fic));
        fclose(fic);
    }
    else {
        printf("ouverture impossible\n");
    }
    return 0;
}
```

```
int NbMot(FILE *fic){
    int NbEspaces =0;
    int caract;

    while (!feof(fic)){
        caract = fgetc(fic);
        if (caract!=EOF)
            printf("%c",caract);
        if (caract == 32 )
            NbEspaces ++;
    }
    printf("\n ");

    return ++NbEspaces;
}
```

Exercice 15

Énoncé

Écrire quelques nombres réels dans un fichier texte les uns en dessous des autres.
Écrire ensuite le programme qui demande quel nombre (rang) on souhaite afficher et qui affiche le nombre en question.

Exemple : si on saisit 3 : le troisième nombre sera affiché à l'écran.

Corrigé

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

int main()
{
    int i , Entier;
    // Entier stockera le rang du réel à afficher
    float Reel;
    FILE* fic = NULL;

    printf("Quel réel afficher ?");
    scanf("%d",&Entier);

    fic = fopen("Test2.txt","r+");
    if (fic != NULL)
    {
        for (i=1 ; i<=Entier;i++)
            fscanf(fic, "%f ", &Reel);

        printf("Reel numero %d : %.2f ",Entier, Reel);

        fclose (fic);
    }

    else
        printf("Ouverture impossible");

    return 0;
}
```

Exercice 16



Énoncé

Soit le fichier texte suivant :

Candidat 1 : 12.0
Candidat 2 : 14.5
Candidat 3 : 7.5
Candidat 4 : 16.0

Écrire le programme qui calcule la moyenne des candidats. Attention ! Le nombre de candidats peut être variable : si on change le nombre de lignes dans le fichier texte le programme doit marcher.

```

12.00
14.50
7.50
16.00
la moyenne des 4 candidats est de : 12.50

```

Capture d'écran du résultat

Corrigé

```

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

```

```

int main(){
FILE *fic = NULL;
float note;
float somme= 0;
int cpt =0;
fic = fopen("fichiers.txt","r");

if (fic != NULL) {
    while (!feof(fic)) {
        if (fgetc(fic)==':') {
            fscanf(fic,"%f",&note);
            printf("%.2f\n",note);
            somme+=note;
            cpt++;
        }
    }
    printf("la moyenne des %d candidats est de : %.2f \n",cpt,somme/cpt);
    fclose(fic);
}

else {
    printf("ouverture du fichier impossible");
}

return 0;
}

```

Exercice 17

Énoncé

Écrire une phrase dans un fichier texte. Écrire le programme qui affiche le 3^{ème} mot.

Corrigé

```

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

```

```

int main() {
FILE *fic;
int cpt =0;
char mot[100];

```

```

fic = fopen("Test3.txt", "r");

if (fic != NULL) {
    while (!feof(fic)) {
        // On compte les espaces (code ASCII 32)
        if (fgetc(fic) == 32) {
            cpt++;
            // Le 3ème mot est situé après le 2ème espace !
            if (cpt == 2) {
                fscanf(fic, "%s", mot);
                printf("Le mot numero %d est : %s\n", (cpt+1), mot);
            }
        }
    }
}

fclose (fic);
}

else {
    printf("Ouverture du fichier impossible");
    exit(-1);
}
return 0;
}

```

Remarque

`fgetc(fic) == 32` peut être remplacé par `fgetc(fic) == ' '` (Attention ! Il y a une espace entre les 2 apostrophes).



Exercice 18

Énoncé

Écrire le programme qui demande de saisir un mot au clavier puis l'insère entre le troisième et quatrième mot d'un fichier texte contenant une phrase.

Corrigé

```

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

```

```

int main(){
    FILE *fic;
    char chaine[30];
    char tampon[200];
    int cpt = 0;
    char caract;
    long curseur;

```

```

    fic = fopen("Fichier.txt", "r+");

```

```
printf("Saisir un mot : ");
gets(chaine);
```

```
if (fic != NULL){
    while (!feof(fic)){
        caract = fgetc(fic);
        printf("%c",caract);
        if (caract == 32){
            cpt++;
            if (cpt == 3){
                curseur = ftell(fic);
                fgets(tampon,200,fic);
            }
        }
        fseek(fic,curseur,SEEK_SET);
        fprintf(fic,"%s",chaine);
        fprintf(fic," %s",tampon);
        fclose(fic);
    }
}
else {
    printf("ouverture du fichier impossible");
}
return 0;
}
```



Exercise 19

Énoncé

Écrire le programme qui calcule combien de phrases sont présentes dans un fichier texte. Pour rappel, une phrase se termine par un point, un point d'exclamation ou un point d'interrogation.

Corrigé

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

```
int main(){
    FILE *fic;
    int NbPhrase =0;
    int caract;
    fic = fopen("Test2.txt", "r");
```

```
if (fic != NULL){
    while (!feof(fic)){
        caract =fgetc(fic);
        if (caract == '.' || caract == '?' || caract == '!') {
            NbPhrase++;
        }
    }
}
```

```

    }
}
printf("Il y a %d phrase(s) dans ce fichier texte\n",NbPhrase);
fclose(fic);
}

else{
    printf("Ouverture du fichier impossible");
}
return 0;
}

```

Remarque

On ne gère pas ici une phrase se terminant par trois petits points.



Exercice 20

Énoncé

Écrire le programme qui demande de saisir un mot au clavier puis l'envoie à une fonction qui affiche "Mot présent" si le mot est présent dans un fichier texte contenant une phrase.

Corrigé

```

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

```

```

void EstPresent(FILE *fic, char *mot);

```

```

int main() {
    char mot[30];
    FILE *fic;
    fic = fopen("Fichier.txt", "r");

```

```

    printf("saisir un mot au clavier : ");
    scanf("%s",&mot);

```

```

    EstPresent(fic,mot);
    return 0;
}

```

```

void EstPresent(FILE *fic, char *mot){
    char buffer[20];
    int caract;
    int i,j ;

```

```

    if (fic !=NULL){
        while (!feof(fic)){

```

```

        // reinitialisation du buffer
        for(j=0;j<20;j++){
            buffer[j]= '\0';
        }
        i =0;

        // decoupage des mot du fichier texte
        do {
            caract = fgetc(fic);
            if (caract != ' ' && caract != '.' && caract!='\n' )
            {
                buffer[i]= caract;
                i++;
            }
        } while (caract != ' ' && caract != '.' && caract!= EOF );

        // On verifie si les chaines sont égales
        if (strcmp(buffer,mot)== 0){
            printf("Mot present ...\n");
        }

    }
    fclose(fic);
}

else{
    printf("Ouverture du fichier impossible");
}

return;
}

```



Exercice 21

Énoncé

Écrire le programme qui demande de saisir un mot au clavier puis l'envoie à une fonction qui retourne le nombre d'occurrences de ce mot dans un fichier texte contenant une phrase.

Corrigé

```

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

```

```

// prototype
int NbOccurence(FILE *fic, char *mot);

```

```

int main() {
    char mot[30];

```



```

FILE *fic;
fic = fopen("Test2.txt","r");
printf("saisir un mot au clavier : ");
scanf("%s",mot);
printf("Il y a %d occurrences de ce mot\n", NbOccurence(fic,mot) );
return 0;
}

```

```

int NbOccurence(FILE *fic, char *mot){
char buffer[20];
int caract;
int i,j ;
int NbrOccurence =0;

if (fic !=NULL){
    while (!feof(fic)){
        // reinitialisation du buffer
        for (j=0 ; j<20 ; j++){
            buffer[j]= '\0';
        }
        i =0;

        // decoupage des mot du fichier texte
        do{
            caract = fgetc(fic);
            if (caract != ' ' && caract != '.' && caract!='\n' ){
                buffer[i]= caract;
                i++;
            }
        } while (caract != ' ' && caract != '.' && caract!= EOF );

        // on verifie si les chaines sont égales
        if (strcmp(buffer,mot)== 0){
            NbrOccurence++;
        }
    }
    fclose (fic);
}
else{
    printf("Ouverture du fichier impossible\n");
}
return NbrOccurence;
}

```

Exercice 22

Énoncé

Écrire le programme qui appelle la fonction *Ecrit_Mots* qui écrit une phrase saisie au clavier dans un fichier binaire et qui appelle ensuite la fonction *Mot_3* qui affichera le 3^{ème} mot présent dans le fichier binaire.

Corrigé

#

Exercice 23

Énoncé

Écrire le programme qui appelle la fonction *Ecrit_Mots* qui écrit une phrase d'au moins 10 mots saisie au clavier dans un fichier binaire et qui appelle ensuite la fonction *Nb_Mot* qui affichera le nombre de mots présents dans le fichier binaire.

Corrigé

#

Exercice 24

Énoncé

Écrire le programme qui appelle la fonction *Ecrit_200* qui écrit 200 entiers aléatoires dans un fichier binaire et qui appelle ensuite la fonction *Plus_Grand* qui affichera le nombre le plus grand présent dans le fichier binaire.

Corrigé

#