Chapitre I
Fichiers

Sommaire

Cours Elearning :
https://elearning.univ-bejaia.dz/course/view.php?id=2749

Page facebook :
https://www.facebook.com/InitiationAlgoProgrammation/

La chaîne Youtube :
https://www.youtube.com/@algo-prog

La playlist sur le langage C :
https://youtube.com/playlist?list=PLwHHAvorm5F-tL9EXDEH0miOKmAj7iUTU

Adapté par : Redouane OUZEGGANE
rouzeggane@gmail.com - redouane.ouzeggane@univ-bejaia.dz

Chapitre I : Fichier

I.1. Introduction

Dans les programmes que nous avons réalisés durant le premier semestre, les données sont stockées en mémoire vive (RAM), et comme vous le savez, la RAM est une mémoire volatile et possède une capacité limitée. Alors que, les applications nécessitent une sauvegarde permanente sur un support de stockage durable (mémoire secondaire : disque dure, flash-disk, disquette, …).

Ainsi, nous utilisons les fichiers pour sauvegarder, d'une façon permanente, les données d'un programme.

I.2. Objectifs

- Comprendre les concepts de base relatifs aux fichiers
- Manipuler et utiliser les fichiers (dans les algorithme et concrètement dans les programmes C : particulièrement dans votre mini-projet).

I.3. Définition

Un fichier est une structure de données logique formée de cellules contiguës permettant l'implantation d'une suite de données physique en mémoire secondaire (Disque dur, CD-ROM, disquette, …). Chaque cellule correspond généralement à un enregistrement.

Exemples de fichier :

- Liste des étudiants
- Liste des matières
- Liste des produits stockés dans un magasin
- *etc.*

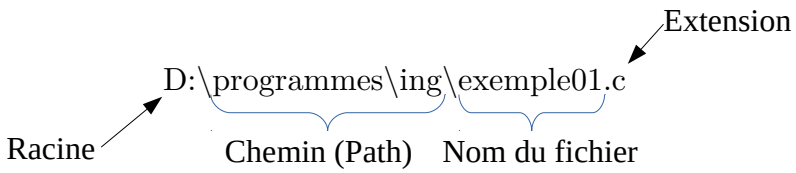
I.4. Éléments liés à un fichier

a) Nom externe (nom physique)

Nom externe est le nom avec lequel le fichier est identifié sur la mémoire secondaire, ce nom est composé de :

- identifiant du support (Racine)
- le chemin vers le fichier (path) : il y a le chemin absolue et le chemin relatif.
- nom du fichier proprement dit
- extension du fichier

Exemples :

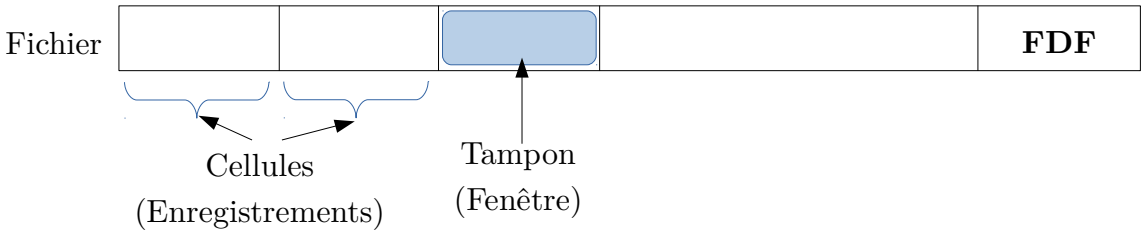


b) Nom interne (nom logique)

Nom interne est le nom avec lequel le fichier est identifié dans un programme (algorithme). C'est le nom logique qui doit être associé avec le nom externe (le nom physique). Plus précisément, un nom interne (logique) est un identificateur d'une variable pour accéder au données physiques du fichier.

c) Tampon (Buffer)

On appel tampon (buffer) d'un fichier, une zone de la mémoire central (RAM) pouvant contenir une cellule (éventuellement un enregistrement) de ce fichier. C'est à travers ce tampon qu'on rends les cases du visible : pour cela, on appelle le tampon ***fenêtre*** à travers laquelle on *voit* le fichier.



d) FDF (EOF)

Chaque fichier se termine par une cellule spéciale dite FDF (pour Fin de Fichier), on anglais c'est : EOF (**End Of File**). Ça permet de savoir la fin de fichier lors du parcours dess données du fichiers (voir les algorithmes et programmes) dans la suite du cours).

e) Indice (pointeur)

Pour chaque fichier, un indice (pointeur) caché, qui est géré automatiquement par le type fichier, est utilisé pour lire ou modifier la cellule actuellement pointée par cet indice.

f) Mode d'accès

Il y a deux types de fichiers, en terme de mode d'accès, à savoir :

- Fichiers à accès séquentiels : pour accéder à la $n^{ième}$ enregistrement, on doit parcours tous les (n-1) cellules précédentes. (Souvent, ce sont les fichiers textes)

- Fichiers à accès direct : Dans ce cas, on peut accéder directement à une cellule (enregistrement) en utilisant son rang (l'indice).

Remarque :

- Un fichier est vu comme un tableau à une dimension (vecteur) illimité (pratiquement illimité).

- Les caractéristiques des fichiers sont étroitement liées aux langages de programmation, et puisque le langage étudié pour vous est le langage C, nous allons se focalisé sur les type de fichiers offerts par ce langage.

Dans les sections qui suivent, nous allons voir, techniquement, comment manipuler les fichiers en algorithmique et la programmation (en langage C). Par la suite, nous allons traiter quelques exemples ou exercices d'application.

I.5. Manipulation des fichiers

Dans ce qui suit, nous allons voir comment :

- Déclarer un fichier,
- Créer un nouveau fichier
- Lire le contenu d'un fichier
- Modifier le contenu d'un fichier

On aura deux type de fichier à manipuler, à savoir :

- Les fichiers textes : mode d'accès séquentiel
- Les fichiers binaires (structurés) : mode d'accès direct

I.5.1. Fichiers Textes

I.5.1.a – Déclaration

Un fichier texte est une fichier de caractères (c'est à dire que le buffer est un caractère ou chaîne de caractères). On déclare un fichier texte comme suit :

Algorithmique	Langage C

Algorithme Exemple;
 	
Variables
f : fichier de caractères;
 	
Début
<instruction(s)>;
Fin.
	#include <stdio.h>
	#include <stdlib.h>
	
	int main()
	{
	FILE* f; //Déclaration d'un fichier f (pointeur)
	
	<Instruction(s)>;
	}

Dans la partie déclaration, on définit le nom interne du fichier (le nom logique).

I.5.1.b – Ouverture

Pour lire ou écrire dans un fichier, il faut tout d'abord l'ouvrir. En algorithmique, il y a trois modes d'ouverture :

- *Ouverture en écriture (**E**)* : Si le fichier n'existe pas, il sera créé. Sinon, il sera écrasé (Perte de données).
- *Ouverture en lecture (**L**)* : Si le fichier existe, on ouvre le fichier, sinon, erreur.
- *Ouverture en ajout* (**A**) (lecture / écriture) : si le fichier n'existe pas, il sera crée, sinon, il sera ouvert et son contenu ne sera pas écrasé.

L'ouverture d'un fichier se fait en utilisant la fonction Ouvrir qui possède deux paramètres :

- Le nom externe (physique) du fichier
- Le mode d'ouverture (E, L ou A)

Cette fonction retourne une variable qui permet d'accéder au fichier : le nom interne du fichier. La syntaxe d'ouverture d'un fichier est comme suit :

Algorithmique	Langage C
<nom_fichier> = Ouvrir ("nom_externe", <mode>);	<nom_fichier> = fopen("nom_externe", <mode>) ;

Tel-que :

<nom_fihcier> : nom interne (logique) du fichier (variable de type fichier) ;

"nom_externe" : Chaine de caractère indiquant le nom externe du fichier ;

<mode> : mode d'ouverture ("L", "E" ou "A")

Par exemple :

Algorithmique	Langage C

Algorithme Exemple;
 	
Variables
f : fichier de caractères;
 	
Début
f ← Ouvrir("fichier01.txt", 'E');
 	
<suite_instruction(s)>;
Fin.
	#include <stdio.h>
	
	int main()
	{
	FILE* f; //Déclaration d'un fichier f (pointeur)
	f = fopen("fichier01.txt", "w");
	
	<Suite_Instruction(s)>;
	return 0;
	}

Dans l'exemple ci-dessus, le fichier sera ouvert en mode écriture :

- Si le fichier n'existe pas, il sera créé
- Sinon, il sera écrasé

I.5.1.c – Fermeture

Après avoir manipuler un fichier, il faut toujours le fermer. La syntaxe de fermeture d'un fichier est comme suit :

Algorithmique	Langage C
fermer(<nom_fichier>);	fclose(<nom_fichier>);

Tel-que :

<nom_fihcier> : nom interne (logique) du fichier (variable de type fichier) ;

Par exemple :

Algorithmique	Langage C

Algorithme Exemple;
 	
Variables
f : fichier de caractères;
 	
Début
f ← Ouvrir("fichier01.txt", 'E');
 	
//Manipuler le fichier (écriture et/ou lecture)
 	
Fermer(f);
Fin.
	#include <stdio.h>
	
	int main()
	{
	FILE* f; //Déclaration d'un fichier f (pointeur)
	f = fopen("fichier01.txt", "w");
	
	//Manipuler le fichier (lecture et/ou écriture)
	
	fclose(f);
	return 0;
	}

I.5.1.d – Manipulation de fichier : lecture et écriture

Après avoir ouvert un fichier, on peut écrire dedans (du contenu textuelle : chaîne de caractères ou des valeurs numériques (entiers ou réels). On peut aussi lire le contenu d'un fichier texte avec un format déterminé : caractère, chaîne de caractères ou des valeurs numériques (entiers ou réels).

Nous commençons par la syntaxe d'écriture dans les fichiers textes :

Algorithmique	Langage C
écrire(<nom_fichier>, <p1>, <p2>, ...); Lire(<nom_fichier>, <p1>, <p2>, ...);	fprintf(<nom_fichier>, <ccf>, <p1>, <p2>, ...); fscanf(<nom_fichier>, <ccf>, &<p1>, &<p2>, ...);

Tel-que :

<nom_fihcier> : nom interne (logique) du fichier (variable de type fichier) ;

<p1> : première variable tampon (chaîne de caractères)

<p2> : deuxième variable tampon (chaîne de caractères)

<ccf> : (uniquement pour le langage C) chaîne de caractère formatée (voir les exemples dans la section suivante).

I.5.1.e – Exemples sur les fichiers textes

Exemple 1 : Création d'un fichier (sans données textuelles : vide)

Dans cet exemple, nous allons voir comment créer un fichier texte vide (sans écrire des données textuelles).

Algorithmique	#	Langage C

Algorithme Exemple01_Creer_Fichier;
 	
Variables
f : fichier de caractères;
 	
Début
f ← Ouvrir("exemple_fichier_01.txt", 'E');
 	
Fermer(f);
Fin.
	01
	02
	03
	04
	05
	06
	07
	08
	09
	10
		#include <stdio.h>
		
		int main()
		{
		FILE* f; //Déclaration d'un fichier f (pointeur)
		f = fopen("exemple_fichier_01.txt", "w");
		
		fclose(f);
		return 0;
		}

L'exemple ci-dessus est accessible via le lien : https://www.dropbox.com/s/y1cpjc91lba2j28/exemple01.c?dl=0		

Exemple 2 : Création d'un fichier (avec des données textuelles)

Il faut ajouter une variable tampon (soit caractère ou bien (mieux) chaîne de caractère). On veut écrire du texte avec le clavier et le saisir dans le fichier texte, on répète l'opération jusqu'à ce que l'utilisateur saisie la chaîne de caractère "end" :

Algorithmique	#	Langage C

Algorithme Exemple02_Remplir_Fichier;
 	
Variables
f : fichier de caractères;
x : chaîne_caractères;
Début
f ← Ouvrir("exemple_fichier_01.txt", 'E');
 	
Lire(x); //Introduire x par clavier ...
Tant-que (x <> 'end') faire
écrire(f, x); //écrire la valeur de x dans f
Lire(x); //Introduire x par clavier ...
Fin-Tantque;
 	
Fermer(f);
Fin.
	01
	02
	03
	04
	05
	06
	07
	08
	09
	10
	11
	12
	13
	14
	15
	16
	17
	18
	19
		#include <stdio.h>
		#include <string.h>
		
		int main()
		{
		FILE* f; //Déclaration d'un fichier f (pointeur)
		char x[255];
		f = fopen("exemple_fichier_01.txt", "w");
		
		scanf("%s", x);
		while (strcmp(x, "end") != 0)
		{
		fprintf(f, "%s\n", x);
		scanf("%s", x);
		}
		
		fclose(f);
		return 0;
		}

Le programme est disponible sur les liens suivants : version 01 : https://www.dropbox.com/s/z293jgn2egpgu64/exemple02_v01.c?dl=0 version 02 : https://www.dropbox.com/s/w8oquzd8ccsjemd/exemple02_v02.c?dl=0		

Remarque :

Avec l'ouverture le mode « Écriture », le fichier sera créé ou écrasé, il faut faire attention pour ne pas perdre des données.

Exemple 3 : Lire un fichier texte

Dans cet exemple, nous allons voir comment lire le contenu du fichier créé précédemment (l'exemple 2) et l'afficher sur la sortie standard (écran).

Algorithmique	#	Langage C

Algorithme Exemple03_Lire_Fichier;
 	
Variables
f : fichier de caractères;
x : caractere;
Début
f ← Ouvrir("exemple_fichier_01.txt", 'L');
	01
	02
	03
	04
	05
	06
	07
		#include <stdio.h>
		#include <string.h>
		
		int main()
		{
		FILE* f; //Déclaration d'un fichier f (pointeur)
		char x;

```
   Tant-que ( Non(FDF(f)) ) faire
        Lire(f, x); //lire un caractère de fihcier f
        Écrire(x); //Afficher x sur l'écran
   Fin-Tantque;

   Fermer(f);
Fin.
```

```
08    f = fopen("exemple_fichier_01.txt", "r");
09
10   while (!feof(f) )
11   {
12        fscanf(f, "%c", &x);  //Lire un caractère
13        printf("%c", x);
14   }
15
16    fclose(f);
17    return 0;
18  }
```

Le programme est disponible sur les liens suivants :
https://www.dropbox.com/s/p2k5739vwhu7jdu/exemple03.c?dl=0

Exemple 4 : Modifier un fichier texte

Modifier le fichier précédent en ajoutant du texte à la fin du fichier ....

Algorithmique

```
Algorithme Exemple04_Modifier_fichier;

  Variables
     f : fichier de caractères;
     x : chaîne_caractères;
Début
   f ← Ouvrir("exemple_fichier_01.txt", 'A');

   Lire(x); //Introduire x par clavier ...
   Tant-que (x <> 'end') faire
        écrire(f, x); //écrire la valeur de x dans f
        Lire(x); //Introduire x par clavier ...
   Fin-Tantque;

   Fermer(f);
Fin.
```

Langage C

```
01  #include <stdio.h>
02  #include <string.h>
03
04  int main()
05  {
06     FILE*  f; //Déclaration d'un fichier f (pointeur)
07     char x[255];
08     f = fopen("exemple_fichier_01.txt", "a");
09
10     scanf("%s", x);
11    while ( strcmp(x, "end") != 0)
12    {
13         fprintf(f, "%s\n", x);
14         scanf("%s", x);
15    }
16
17     fclose(f);
18     return 0;
19  }
```

Le programme est disponible sur les liens suivants :
Version 01: https://www.dropbox.com/s/iwxihtbxd88vdp5/exemple04_v01.c?dl=0
Version 02: https://www.dropbox.com/s/982so9w6zzll0zh/exemple04_v02.c?dl=0

Exemple 5 : Dupliquer un fichier

Dans cet exemple, nous allons comment dupliquer un fichier dans un nouveau fichier :

Algorithmique	#	Langage C

Algorithme Exemple05_Copier;
 	
Variables
f1, f2 : fichier de caractères;
x : caractere;
Début
f1 ← Ouvrir("exemple_fichier_01.txt", 'L');
f2 ← Ouvrir("exemple_fichier_02.txt", 'E');
 	
Tant-que (Non(FDF(f))) faire
Lire(f1, x); //lire un caractère de fihcier f1
Écrire(f2, x); //écrire x dans f2
Fin-Tantque;
 	
Fermer(f1);
Fermer(f2);
Fin.
	01
	02
	03
	04
	05
	06
	07
	08
	09
	10
	11
	12
	13
	14
	15
	16
	17
	18
	19
		#include <stdio.h>
		#include <string.h>
		
		int main()
		{
		FILE *f1, *f2;
		char x;
		f1 = fopen("exemple_fichier_01.txt", "r");
		f2 = fopen("exemple_fichier_02.txt", "w");
		while (!feof(f))
		{
		fscanf(f1, "%c", &x); //Lire un caractère
		fprintf(f2, "%c", x); //écrire un caractère
		}
		
		fclose(f1);
		fclose(f2);
		return 0;
		}

Le programme est disponible sur les liens suivants : https://www.dropbox.com/s/68rn9oqq403vrsm/exemple05.c?dl=0		

Remarques :

- Pour supprimer un fichier en langage C, il suffit d'utiliser la fonction *remove* :

```
int remove( const char * fileName );
```

- Pour renommer un fichier en langage C, il suffit d'utiliser la fonction *rename* :

```
int rename( const char * oldName, const char * newName );
```

I.5.2. Fichiers Binaire (Structurés)

Un fichier binaire est souvent utilisé avec les structures (type enregistrement).

I.5.2.a – Déclaration

Un fichier binaire est déclaré comme suit :

Algorithmique	Langage C

Algorithme Exemple;
Type
<Type_Enregistrement> = Enregistrement
<champ_1>:<type_champ_1>;
<champ_2>:<type_champ_2>;
....
<champ_n>:<type_champ_n>;
Fin;
 	
Variables
f : fichier de <Type_Enregistrement>;
x : <Type_Enregistrement>;//Tampon ...
 	
Début
<instruction(s)>;
Fin.
	#include <stdio.h>
	#include <stdlib.h>
	
	typedef struct {
	<type_champ_1> <champ_1>;
	<type_champ_2> <champ_2>;
	....
	<type_champ_n> <champ_n>;
	} <Type_Enregistrement>;
	
	int main()
	{
	FILE* f; //Déclaration d'un fichier f (pointeur)
	<Type_Enregistrement> x; //Buffer ...
	
	<Instruction(s)>;
	}

En langage C, pour la variable fichier (le nom logique), il n y a pas une différence entre les fichiers textes et les fichiers binaires. La différence sera explicite au niveau de l'ouverture de fichier.

I.5.2.b – Ouverture

Comme les fichiers textes, les fichier binaire possède trois modes d'ouverture : E, L et A. Et l'ouverture d'un fichier se fait en utilisant la fonction Ouvrir qui possède deux paramètres : le nom physique et le mode d'ouverture. Cette fonction retourne le nom interne du fichier. La syntaxe d'ouverture d'un fichier est comme suit :

Algorithmique	Langage C
<nom_fichier> = Ouvrir ("nom_externe", <mode>);	<nom_fichier> = fopen("nom_externe", <mode>) ;

Tel-que :

<nom_fihcier> : nom interne (logique) du fichier (variable de type fichier) ;

"nom_externe" : Chaine de caractère indiquant le nom externe du fichier ;

<mode> : mode d'ouverture ("L", "E" ou "A") En C : "rb", "wb" ou "ab" respectivement.

Par exemple :

Algorithmique	Langage C

Algorithme Exemple;
Type //Déclarer un type <Type_Enregistrement>
Variables
f : fichier de <Type_Enregsitrement>;
 	
Début
f ← Ouvrir("fichier01.bin", 'E');
 	
<suite_instruction(s)>;
Fin.
	#include <stdio.h>
	typedef//Déclarer un type <Type_Enregistrement>
	int main()
	{
	FILE* f; //Déclaration d'un fichier f (pointeur)
	f = fopen("fichier01.bin", "wb");
	
	<Suite_Instruction(s)>;
	return 0;
	}

Dans l'exemple ci-dessus, le fichier sera ouvert en mode écriture :

- Si le fichier n'existe pas, il sera créé

- Sinon, il sera écrasé

I.5.2.c – Fermeture

Identique aux fichiers textes :

Algorithmique	Langage C
fermer(<nom_fichier>);	fclose(<nom_fichier>);

Tel-que <nom_fihcier> est nom interne (logique) du fichier.

Par exemple :

Algorithmique	Langage C

Algorithme Exemple;
Type //Déclarer un type <Type_Enregistrement>
Variables
f : fichier de <Type_Enregsitrement>;
 	
Début
f ← Ouvrir("fichier01.bin", 'E');
//Manipuler le fichier (écriture et/ou lecture)
Fermer(f);
Fin.
	#include <stdio.h>
	typedef//Déclarer un type <Type_Enregistrement>
	int main()
	{
	FILE* f; //Déclaration d'un fichier f (pointeur)
	f = fopen("fichier01.txt", "wb");
	//Manipuler le fichier (lecture et/ou écriture)
	fclose(f);
	return 0;
	}

I.5.2.d – Manipulation de fichier : lecture et écriture

Après avoir ouvert un fichier, on peut écrire dedans des enregistrements (structurés). On peut aussi lire le contenu d'un fichier enregistrement par enregsitrement.

Nous commençons par la syntaxe d'écriture dans les fichiers textes :

Algorithmique	Langage C
écrire(<nom_fichier>, <p1>, <p2>, ...); Lire(<nom_fichier>, <p1>, <p2>, ...);	fwrite(<adr_tampon>, <tb>, <nb>, <nom_fichier>); fread(<adr_tampon>, <tb>, <nb>, <nom_fichier>);

Tel-que :

<nom_fihcier> : nom interne (logique) du fichier (variable de type fichier) ;

<p1>, <p2>, ..., : variables tampons (buffer) de même type enregistrement

Pour le C :

<adr_tampon> : adresse du buffer de type enregistrement (il peux être un tableau)

<tb> : taille du bloc : taille mémoire d'une seule variable d'enregistrement.

<nb> : nombre de blocs à lire (en générale 1)

I.5.2.e – Exemples sur les fichiers binaires (Structurés)

Exemple 1 : Création d'un fichier vide

Dans cet exemple, nous allons voir comment créer un fichier binaire vide de type produit. Un produit est défini par une référence, désignation, couleur, prix unitaire et quantité en stock.

Algorithmique	#	Langage C
Algorithme Exemple01_Creer_Fichier;	01	#include <stdio.h>
Type	02	typedef struct {
Tproduit = Enregistrement	03	char reference[20];
reference : Chaîne_Caracters[20];	04	char designation[100];
designation : Chaîne_Caracters[100];	05	char couleur[15];
couleur : Chaîne_Caracters[15];	06	float prix_unitaire;
prix_unitaire : réel;	07	int qte_stock;
qte_stock : entier;	08	} Tproduit;
Fin;	09	
	10	int main()
Variables	11	{

```
    f : fichier de TProduit;

Début
   f ← Ouvrir("produits.dat", 'E');

   Fermer(f);
Fin.
```

```
   FILE*  f; //Déclaration d'un fichier f (pointeur)
   f = fopen("produits.dat", "wb");

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

L'exemple ci-dessus est accessible via le lien :
https://www.dropbox.com/s/j0z28nz7hwx3yh4/exemple06.c?dl=0

Exemple 2 : Création d'un fichier et le remplir

Il faut ajouter une variable tampon de type enregistrement, l'utilisateur arrête la boucle d'ajout d'enregistrement par un choix (entier, 0:non et autre:oui) :

Algorithmique

```
Algorithme Exemple02_Creer_Fichier;
  Type
    TProduit = Enregistrement
        reference : Chaîne_Caracters[20];
        designation : Chaîne_Caracters[100];
        couleur : Chaîne_Caracters[15];
        prix_unitaire : réel;
        qte_stock : entier;
    Fin;

  Variables
     f : fichier de TProduit;
     x : TProduit;   choix, n : entier;
Début
   f ← Ouvrir('produits.dat', 'E');
    n ← 0;
   répéter
      n ← n+1;
      écrire('Enregistrement N°', n, ' : ');
      écrire('   Référence : ');
      Lire(x.reference);
      écrire('   Désignation : ');
      Lire(x.designation);
      écrire('   Couleur : ');
      Lire(x.couleur);
      écrire('   Prix unitaire : ');
      Lire(x.prix_unitaire);
```

Langage C

```
#include <stdio.h>
typedef struct {
   char reference[20];
   char designation[100];
   char couleur[15];
   float prix_unitaire;
   int qte_stock;
} TProduit;

int main()
{
   FILE*  f; //Déclaration d'un fichier f (pointeur)
   TProduit x; int choix, n;

   f = fopen("produits.dat", "wb");

   //Pour le programme, voir le lien ci-dessous
...
   fclose(f);

   return 0;
}
```

```
      écrire('  Qte Stock : ');
      Lire(x.qte_stock);

      écrire(f, x);

     écrire(' Voullez vous continuer ? (0:non) : ') ;
     Lire(choix);
  Jusqu'à (choix = 0);

  écrire('Vous avez créés ', n, ' enregistrements') ;

  Fermer(f);
Fin.
```

Le programme est disponible sur les liens suivants :
https://www.dropbox.com/s/1ef2y3473r1m47m/exemple07.c?dl=0

Remarque :

Avec l'ouverture le mode « Écriture », le fichier sera créé ou écrasé, il faut faire attention pour ne pas perdre les données.

Exemple 3 : Lire un fichier binaire

Dans cet exemple, nous allons voir comment lire le contenu du fichier créé précédemment (l'exemple 2) et l'afficher sur la sortie standard (écran).

Algorithmique	#	Langage C
Algorithme Exemple03_Lire_Fichier;	01	#include <stdio.h>
Type	02	typedef struct {
TProduit = Enregistrement	03	char reference[20];
reference : Chaîne_Caracters[20];	04	char designation[100];
designation : Chaîne_Caracters[100];	05	char couleur[15];
couleur : Chaîne_Caracters[15];	06	float prix_unitaire;
prix_unitaire : réel;	07	int qte_stock;
qte_stock : entier;	08	} TProduit;
Fin;	09	
	10	int main()
Variables	11	{
f : fichier de TProduit;	12	FILE* f; //Déclaration d'un fichier f (pointeur)
x : TProduit; choix, n : entier;	13	TProduit x; int choix, n;
Début	14	
f ← Ouvrir('produits.dat', 'L');	15	f = fopen("produits.dat", "rb");

```
 n ← 0;
 Tant-que (Non(FDF(f))) faire
    n ← n+1;
    Lire(f, x);

    écrire('Enregistrement N°', n, ' : ');
    écrire('   Référence : ', x.reference);
    écrire('   Désignation : ', x.reference);
    écrire('   Couleur : ', x.reference);
    écrire('   Prix unitaire : ', x.reference);
    écrire('   Qte Stock : ', x.reference);
 Fin-Tant-que;

 écrire('Vous avez affiché', n, ' objects') ;

  Fermer(f);
Fin.
```

```
16
17    n = 0;
18    while (!feof(f))
19    {
20       n++;
21       fread(&x, sizeof(TProduit), 1, f);
22
23       printf("Enregistrement N° %d : \n", n);
24       printf("    Reference : %s\n", x.reference);
25       printf("    Designation : %s\n", x.designation);
26       printf("    Couleur : %s\n", x.couleur);
27       printf("    Prix unitaire : %.2f\n", x.prix_unitaire);
28       printf("    Qte Stock : %d\n", x.qte_stock);
29       printf("\n");
30    }
31
32       printf("Vous avez affiché %d enregistrements
.→ (Produits).", n);
33
34    return 0;
35 }
```

Le programme est disponible sur les liens suivants :
https://www.dropbox.com/s/zomz6vzkjv9jp8p/exemple08.c?dl=0
fichier produits.dat : https://www.dropbox.com/s/e47o6ue5vh82y3r/produits.dat?dl=0

Exemple 4 : Modifier un fichier binaire

Modifier le fichier précédent en ajoutant en ajoutant un (des) enregistrements.

Algorithmique	#	Langage C

Algorithme Exemple04_Modifier_Fichier;
  Type
  TProduit = Enregistrement
  reference : Chaîne_Caracters[20];
  designation : Chaîne_Caracters[100];
  couleur : Chaîne_Caracters[15];
  prix_unitaire : réel;
  qte_stock : entier;
  Fin;
 	
  Variables
  f : fichier de TProduit;
  x : TProduit; choix, n : entier;
Début
	01
	02
	03
	04
	05
	06
	07
	08
	09
	10
	11
	12
	13
	14
		#include <stdio.h>
		typedef struct {
		  char reference[20];
		  char designation[100];
		  char couleur[15];
		  float prix_unitaire;
		  int qte_stock;
		} TProduit;
		
		int main()
		{
		  FILE* f; //Déclaration d'un fichier f (pointeur)
		  TProduit x; int choix, n;

```
   f ← Ouvrir('produits.dat', 'A');
    n ← 0;
   répéter
      n ← n+1;
      écrire('Enregistrement N°', n, ' : ');
      écrire('   Référence : ');
      Lire(x.reference);
      écrire('   Désignation : ');
      Lire(x.designation);
      écrire('   Couleur : ');
      Lire(x.couleur);
      écrire('   Prix unitaire : ');
      Lire(x.prix_unitaire);
      écrire('   Qte Stock : ');
      Lire(x.qte_stock);

      écrire(f, x);

     écrire(' Voullez vous continuer ? (0:non) : ') ;
     Lire(choix);
   Jusqu'à (choix = 0);

            écrire('Vous avez ajouté ', n, '
enregistrements') ;

   Fermer(f);
Fin.
```

```
   f = fopen("produits.dat", "ab");

   //Pour le programme, voir le lien ci-dessous
...
   fclose(f);

   return 0;
}
```

Le programme est disponible sur les liens suivants :
https://www.dropbox.com/s/k93paf89805z7tl/exemple09.c?dl=0
fichier produits.dat : https://www.dropbox.com/s/e47o6ue5vh82y3r/produits.dat?dl=0

Exemple 5 : Dupliquer un fichier

Dans cet exemple, nous allons comment dupliquer un fichier dans un nouveau fichier :

Algorithmique	#	Langage C
Algorithme Exemple05_Dupliquer_fichier; Type TProduit = Enregistrement reference : Chaîne_Caracters[20]; designation : Chaîne_Caracters[100]; couleur : Chaîne_Caracters[15];	01 02 03 04 05 06	#include <stdio.h> typedef struct { char reference[20]; char designation[100]; char couleur[15]; float prix_unitaire;

```
        prix_unitaire : réel;
        qte_stock : entier;
    Fin;

  Variables
     f1, f2 : fichier de TProduit;
     x : TProduit;   n : entier;
Début
   f1 ← Ouvrir('produits.dat', 'L');
   f2 ← Ouvrir('produits2.dat', 'E');
    n ← 0;
   Tant-que (Non(FDF(f))) faire
      n ← n+1;
      Lire(f1, x);
      écrire(f2, x);
   Fin-Tant-que;

   écrire('Vous avez duppliqué', n, ' objects') ;

   Fermer(f1);
   Fermer(f2);
Fin.
```

```
   int qte_stock;
} TProduit;

int main()
{
    FILE*  f1, f2;
    TProduit x; int n;

  f1 = fopen("produits.dat", "rb");
  f2 = fopen("produits.dat", "wb");

   n = 0;
   while (!feof(f1))
   {
      n++;
      fread(&x, sizeof(TProduit), 1, f1);
      fwrite(&x, sizeof(TProduit), 1, f2);
   }

   printf("Vous avez duppliqué %d objets.", n);
   fclose(f1) ; fclose(f2);
    return 0;
}
```

Le programme est disponible sur les liens suivants :
https://www.dropbox.com/s/wyhb5rquolvg4io/exemple10.c?dl=0
fichier produits.dat : https://www.dropbox.com/s/e47o6ue5vh82y3r/produits.dat?dl=0

Remarques :

- Avec les fichiers binaires, on peut positionner le pointeur du fichier (l'indice caché) sur une cellule (enregistrement) en donnant son indice (0 ≤ indice < Nombre d'enregistrements du fichier). Par exemple, pour mettre le pointeur sur l'enregistrement d'indice 2 dans le fichier, on écrit :

Algorithmique	Langage C
positionner(f, 2);	fseek(f, 2*sizeof(TProduit), SEEK_SET);

- On peut réaliser un programme qui permet de compter le nombre d'enregistrement dans un fichier, voir le programme suit (le deuxième paramètre de fseek est en octets) : https://www.dropbox.com/s/hewynwen2myfrcx/exemple11.c?dl=0

Autres Exercices

1) Soit produits.dat fichier de produits créé précédemment. Écrire un algorithme / programme C qui permet de donner n : le nombre d'enregistrement dans le fichier.

2) Soit produits.dat fichier de produits créé précédemment. Écrire un algorithme / programme C qui permet de supprimer l'enregistrement en donnant son indice (0 jusqu'à n-1)

3) Soit produits.dat fichier de produits créé précédemment. Écrire un algorithme / programme C qui permet d'insérer un enregistrement en donnant l'indice d'insertion (0 jusqu'à n).