Université de Mila
Département de Mathématique et Informatique
1ère Année LMD MI

TP 3 : Les fichiers

1. Définition :

Un fichier informatique est au sens commun, une collection, ou un ensemble de données numériques (les 0,1) réunies sous un même nom, enregistrées sur un support de stockage permanent, appelé mémoire de masse, tel qu'un disque dur, un CD-ROM, une mémoire flash.., et manipulées comme une unité.

Un fichier comporte **un nom de fichier** qui sert à désigner le contenu et y accéder. Ce nom comporte souvent un **suffixe** appelé l´**extension**, qui renseigne sur la nature des informations contenues dans le fichier et donc des logiciels utilisables pour le manipuler.

2. Types de fichiers :

En programmation, on a principalement deux types de fichiers :

- **Le fichier texte :** Ils sont constitués d'une suite de caractères formant un texte (chaîne de caractères). On les utilise pour enregistrer des textes mais également des valeurs numériques en vue de les échanger avec d'autres logiciels. Ils sont lisibles par un simple éditeur de texte.
- **Le fichier binaire :** contenant des données sous forme **d'octets** qui n'ont donc de sens que pour le logiciel qui les utilise, ce type de fichiers est **illisible par un éditeur de texte**, il est constitué d'une collection **d'enregistrements**, chaque enregistrement contenant une collection d'unités logiques d'informations encore appelées **champs**.

3. Manipulation des fichiers binaire :

La plupart des langages de programmation actuels, **en particulier C++**, disposent d'instructions permettant la manipulation de fichiers. Ces instructions peuvent être classées comme suit:

- Ouverture et création d'un fichier,
- Fermeture d'un fichier
- Lecture et écriture d'enregistrements du fichier,
- Positionnement dans le fichier,
- Détection de la fin du fichier.

Remarque :

Pour utiliser un fichier physique **F** dans un programme, il fallait que ce programme comporte une variable fichier **f**. L'association entre **f** et **F** s'effectuera donc au moyen d'un procédé appelé **assignation**, de telle sorte que les modifications apportées à **f** dans le programme affecteront directement **F** sur son support.

En langage C++, cette **étape est intégrée** dans l'ouverture de fichier.

3.1 Ouverture et création d'un fichier

Pour ouvrir un fichier en C++, on utilise la fonction prédéfini « **fopen** »,

La syntaxe est : `FILE* fopen("nom-de-fichier","mode");`

`FILE*` : la valeur de retour de la fonction est un **pointeur** sur le type fichier.

"*nom-de-fichier*" : Le premier argument de fopen est le nom du fichier concerné, fourni sous forme d'une chaîne de caractères (exp : "f.txt").

"*mode*" : Le second argument, ***mode***, est une chaîne de caractères qui spécifie le mode d'accès au fichier, le tableau ci-dessous montre les différents modes :

Le mode	interprétation	Si le fichier existe	Si n'existe pas
"r"	Ouvrir en lecture	Lire à partir du début	Erreur
"w"	Ouvrir en écriture	Ecraser le contenu	Créer fichier
"a"	Ouvrir en lect/ écriture	Ecrire à la fin	Créer fichier
"r+"	Ouvrir en lect/ écriture	Lire à partir du début	Erreur
"w+"	Ouvrir en lect/ écriture	Ecraser le contenu	Créer fichier
"a+"	Ouvrir en lect/ écriture	Ecrire à la fin	Créer fichier

Exemple : ouvrir en lecture un fichier nommé « données.txt » qui se trouve dans la partition C dans le disque dure.

File * f; // déclaration d'un pointeur sur un fichier

Char nom = "C:\ données.txt"; // une chaine de caractères contient le chemin

f = fopen(nom,"r");

Remarque : la fonction fopen dans ce cas renvoi l'adresse de la structure FILE associée au fichier. Elle renvoie **NULL** si elle ne parvient pas à ouvrir le fichier.

Donc : Le test de la valeur renvoyée par fopen est indispensable pour prévenir les erreurs : fichier inexistant, support physique défectueux ou saturé, nombre excessif de fichiers ouverts...

Voici un exemple complet :

```
# include <iostream>
using namespace std;

char nom[6]= "p.txt";
FILE *f;
f = fopen(nom,"r") ; // ouverture du fichier en lecture
if (f == NULL) // teste s'il y a un problème d'ouverture
cout << "erreur d'ouverture du fichier " << nom << endl; // afficher erreur
else  //  la réussite de l'ouverture
{
// ici on fait la lecture de données à partir du fichier
}
```

3.2 La fermeture d'un fichier

Pour fermer un fichier en C++, on utilise la fonction prédéfini « **fclose »**,

La syntaxe est : **fclose** (`FILE*`) ;

Exemple : fclose(f); // fermeture du fichier, f est un pointeur sur un fichier.

Remarque :

Il est indispensable de fermer un fichier avant la fin du programme qui l'utilise pour éviter la perte de données.

3.3 Lecture et écriture à partir d'un fichier

Après l'ouverture du fichier, plusieurs possibilités sont offertes : lire les informations qu'il contient, en modifier quelques-unes, en supprimer certaines, ou alors en ajouter d'autres.

a) Lecture : pour lire des données, on utilise la fonction **fread()** comme suit :

int **fread**(void* **adr_buffer**, int **taille_element**, int **nb_elements**, FILE* **fic**);

- La fonction **fread** renvoi le nombre d'éléments lus.
- **adr_buffer :** l'adresse de la variable qui sert un tampon où ranger les données à lire.
- **taille_element :** la taille en octet d'un élément
- **nb_elements :** un entier qui précise le nombre d'élément qu'on va lire
- **fic** : Le pointeur sur le fichier à lire

Exemple : nb_lu = fread(&b, sizeof(int) , 1 ,f); // lecture d'un élément de type entier du fichier pointer par f, et mettre la donnée lue dans la variable b qui est de type entier.

Remarque importante: cette fonction nous permet de lire un ou plusieurs élément du fichier, donc pour lire tous le fichier il faut répéter l'exécution de cette fonction jusqu'à la fin du fichier, pour détecter la fin du fichier, on a deux façons :

1) la fonction fread() renvoi le nombre d'élément effectivement lus, Si la valeur renvoyée diffère du nombre d'éléments à lire, c'est que l**a fin du fichier a été rencontrée**. On l'utilise comme condition d'une boucle tant que.

2) la deuxième est d'utiliser la fonction prédéfini **feof(),** elle renvoi la valeur NULL si c'est la fin du fichier, et une autre valeur sinon.

b) Ecriture : pour écrire des données, on utilise la fonction **fwrite()** comme suit (elle est similaire à la lecture) :

int **fwrite**(void* **adr_buffer**, int **taille_element**, int **nb_eleme**, FILE* **fic**);

c) Modification : pour modifier un élément d'un fichier, on doit positionner le curseur sur cet élément, en suite, on fait un écrasement de ses valeurs par les nouvelles.

Donc, on fait une lecture du fichier, élément par élément à l'aide de la fonction **fread()** jusqu'à l'élément cible.

La fonction **fseek** permet de positionner le curseur à la position voulu.

Syntaxe : int **fseek (**FILE ***Stream,** long **Offset,** int **Origin);** où :

- **fseek :** renvoi un **0** si l'opération est réussi, **autre valeur** sinon
- **Stream :** c'est le pointeur sur le fichier
- ***Offset*** est le nombre d'octets du déplacement, compté algébriquement à partir de ***Origin***.
- ***Origin*** est une constante qui vaut **SEEK_SET** (« à partir du début du fichier ») ou **SEEK_END** (« à partir de la fin du fichier ») ou, **SEEK_CUR** (« à partir de la position courante »).

Exemple applicatif :

Voici un programme qui contient trois fonctions :

- La première permet d'écrire un certain nombre de points dans un fichier, chaque point contient un nom, et les coordonnées X, et Y.
- La deuxième fonction permet de lire les valeurs de ces points.
- La troisième modifier les coordonné d'un point.

Tout d'abord, on doit définir une structure **Point** qui contient 3 champs.

```
#include <iostream>
//#include <stdio.h>
using namespace std;
// la définition du type Point
struct Point
{
char n[5];
float X,Y;
};

const  int MAX = 10; // le nombre max de points à saisir
 char nom[6]= "p.txt"; // chaine de caractères qui contient le nom du fichier
Point v[MAX];
FILE *f; // le pointeur sur un fichier
//-------------la fonction SaisirPoint-----------
Point SaisirPoint()
{
Point P;
cout << "Saisissez le nom du point ";
cin >> P.n;
cout << "Saisissez les coordonnées du point ";
cin >> P.X >> P.Y ;
return P;
}
//-------------la function ecrire--------------------------
void ecrire (FILE *f)
{

int i,m;

cout << ">>>>> insertion des points dans le fichier:"<< nom << endl;
cout << "combien du points voulez vous saisir:";
cin >> m;

if ((f = fopen(nom,"w")) == NULL) // ouverture du fichier en écriture
cout << "erreur d'ouverture (création) du fichier " << nom << endl;
else
{
for (i=0; i<m; i++) // on remplit le vecteur
v[i] = SaisirPoint(); // appel à la fonction SaisirPoint
fwrite(v,sizeof(Point),m,f); // écriture de v dans le fichier
fclose(f); // fermeture du fichier
}
}
//---------------- La fonction lire -----------------------
void lire (FILE* f)
{
int nb_lu,i;
Point ww;
cout << ">>>>>>>>>>>lecture  des points du fichier : " << nom << endl;
if ((f = fopen(nom,"r")) == NULL) // ouverture du fichier en lecture
cout << "erreur d'ouverture du fichier " << nom << endl;
else
{
// tant que la fin du fichier n'est pas atteinte
```

```
while ( (nb_lu = fread(&ww,sizeof(Point),1,f)==1))
{
// la fonction fread est integrée dans la condition de la boucle
cout << ww.n << " -> " << ww.X <<"  " <<ww.Y<<endl; // affichage du point lu

}
fclose(f); // fermeture du fichier
}
}
//--------------la fonction modifier-------------------------
void modifier (FILE* f)
{
char val[5],;
Point p[0],n_val[0];
bool trouve;
cout << ">>>>>>modification d'un point dans le fichier : " << nom << endl;
if ((f = fopen(nom,"r+")) == NULL) // ouverture du fichier en "mise à jour"
cout << "erreur d'ouverture du fichier " << nom << endl;
else
{
cout << "Saisir la valeur à mettre à jour ?" << endl;
cin >> val;
trouve = false;
//tantque n'est pas la fin du fichier et val n'est pas trouvé
while ((! feof(f)) && (! trouve)) // recherche de val
{
fread(&p,sizeof(Point),1,f); // lis un point
if ( strcmp(val,p[0].n)==0) // comparer avec val
trouve = true;
}
if (trouve) // si val existe, donc le curseur est sur le point suivant
{

fseek(f,-sizeof(Point),SEEK_CUR); // positionner le curseur sur le point précédant

cout << "Saisir la nouvelle valeur ?" << endl;
cin >> n_val[0].n;

cout << "Saisir la nouvelle coordonée X ?" << endl;
cin >> n_val[0].X;
cout << "Saisir la nouvelle coordonée Y ?" << endl;
cin >> n_val[0].Y;

fwrite(&n_val,sizeof(Point),1,f); // inserer le nouveau point
}
else cout << val << " n'est pas trouvée" << endl;
fclose(f); // fermeture du fichier
}

}
//----------------------------------------
int main()
{
 // les appels aux fonctions
ecrire(f);
lire (f);
modifier(f);
lire (f); // pour afficher les modification

system("PAUSE");
}
```

Exercice:

On vous demande de réaliser un **système de gestion des étudiants**. Ce système nous permet de :

- Saisir les données d'un étudiant ; chaque étudiant est identifié par son : **nom**, **prénom**, **date de naissance**, **groupe**, les **notes** de l'algorithmique, l'algèbre, et l'analyse.
- D'afficher les données d'un étudiant
- Afficher les données des étudiants d'un groupe.
- Modifier les données d'un étudiant.