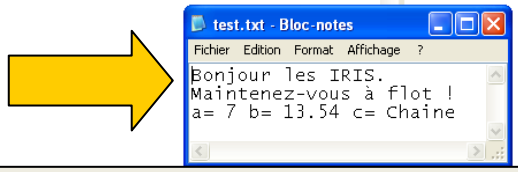
MANIPULATION DES FICHIERS AVEC LES FLOTS D'ENTREES/SORTIES

1. Ecriture de fichiers ASCII

```
#include <conio.h>
#include <iostream>
#include <fstream>          // fichier d'entête pour gérer les flots d'entrées/sorties

using namespace std;

int main()
{
 ofstream fichier;
 int a=7 ;
 double b=13.54;
 string strTest="Chaine";

  fichier.open("test.txt");
  if ( ! fichier.is_open())      // si le fichier n'est pas ouvert
 {  cout <<"Impossible d'ouvrir le fichier " << endl;          return 1;  }

  // Ecriture dans le fichier
 fichier << "Bonjour les IRIS. \nMaintenez-vous à flot !" << endl;
 fichier << "a= "<<a <<" b= "<<b<< " c= "<< strTest.c_str()  <<endl ;

 fichier.close();  // Fermeture du fichier

  _getch( );   return (0);
}
```



Info utile : Pour se repositionner au début du fichier lors d'une écriture, écrire les instructions suivantes :

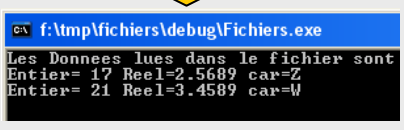
`fichier.seekp(0L);` // positionnement du pointeur d'écriture au début du fichier (***p*** comme ***p***ut dans seek***p***)

2. Lecture de fichiers ASCII

```
#include <conio.h>
#include <iostream>
#include <fstream>          // fichier d'entête pour gérer les flots d'entrées/sorties
using namespace std;

 int main()
{
 ifstream fichier; // Déclaration d'un flot en entrée
 int entier;
 double reel;
 char car;

 fichier.open("FichierALire.txt");
 if (! fichier.is_open())
 {
          cout <<"Impossible d'ouvrir le fichier " << endl  ;  _getch();
          return (1);   // Erreur a l'ouverture, on quitte...
 }

 cout <<"Les Donnees lues dans le fichier sont "<<endl;
 while ( ! fichier.eof())       // Tant que la fin de fichier n'est pas atteinte
 {
         if (fichier >> entier >> reel >> car)  // Si on peut lire 3 valeurs
                 cout <<"Entier= "<<entier <<" Reel="<<reel <<" car="<<car <<endl;
 }

 // Fermeture du fichier
 fichier.close();

  _getch();   return (0);
}
```

FichierALire.txt

```
17      2.5689   Z
21      3.4589   W
```



Info utile : Pour se repositionner au début du fichier lors d'une lecture, écrire les instructions suivantes :

`fichier.clear();` // réinitialise les bits d'erreur du fichier
`fichier.seekg(0L);` // positionnement du pointeur de lecture au début du fichier (***g*** comme ***g***et dans seek***g***)

3. Lecture, ecriture dans un fichier binaire

Dans le cas des fichiers binaires (non lisibles dans un éditeur de texte), **il ne faut pas** utiliser l' opérateur >> pour lire dans le fichier. A la place, on utilise la méthode ***read.*** En ce qui concerne l'écriture, on peut utiliser << ou la méthode **write** comme le montre l'exemple ci-dessous.

```
#include <conio.h>
#include <iostream>
#include <fstream>          // fichier d'entête pour gérer les flots d'entrées/sorties

using namespace std;


int main()
{
 ifstream fichierIn;
 ofstream fichierOut;
 char octet;

 fichierIn.open("test.bin", ios::in | ios::binary);
 if (! fichierIn.is_open())
 {
     cout <<"Erreur d'ouverture " << endl ;
     return (1);   // Erreur a l'ouverture, on quitte...
 }

 // Suppression du fichier de sortie s'il existe déjà avec ios::trunc
 fichierOut.open("test2.bin", ios::out | ios::binary | ios::trunc);

 while ( ! fichierIn.eof())     // Tant que la fin de fichier n'est pas atteinte
 {
  if (fichierIn.read(&octet , 1))   // si on lit correctement un octet du fichier d'entrée. Ne pas oublier le & devant la variable
          fichierOut.write(&octet , 1);   //...inscrire l'octet lu dans le fichier de sortie. Ne pas oublier le & devant la variable

 }

 // Fermeture des fichiers
 fichierIn.close();
 fichierOut.close();

 return (0);
}
```

Si on veut lire ou écrire plusieurs octets en une seule fois, il faut créer un tableau pour stocker les octets lus et remplacer 1 par un autre nombre.

Info utile : Pour connaître la position des pointeurs de fichiers

```
fichier.tellg();        // donne la position du pointeur de lecture
fichier.tellp);         // donne la position du pointeur d'écriture
```

4. Modificateurs de mode

- *ios::in* : si le fichier n'existe pas, un nouveau fichier vide est créé. Si le fichier existe, il est ouvert et devient accessible en lecture.
- *ios::out* : si le fichier n'existe pas, un nouveau fichier vide est créé, prêt à être écrit. Si le fichier existe, il est ouvert, son contenu précédent est effacé et le fichier est prêt à être écrit.
- *ios::trunc* : si le fichier n'existe pas, un nouveau fichier vide est créé. Si le fichier existe, il est ouvert, son contenu précédent est effacé.
- *ios::app* : si le fichier n'existe pas, un nouveau fichier vide est créé, prêt à être écrit . Si le fichier existe, il est ouvert en écriture et toutes les données écrites le seront à la fin du fichier existant sans effacement possible des données existantes.
- *ios::binary* : ouverture d'un fichier binaire pour lecture ou écriture.
- *ios::ate (pour* at end*) :* si le fichier n'existe pas, un nouveau fichier vide est créé, prêt à être écrit . Si le fichier existe, il est ouvert en écriture et le pointeur de fichier est positionné à la fin du fichier existant, mais il est possible de le repositionner à n'importe quel endroit pour réécrire n'importe où.

En utilisant fstream, on ouvre en lecture et en écriture un fichier.

Le fonctionnement est le même que pour ifstream ou ofstream comme le montre l'exemple ci-dessous.

```
// Programme permettant de travailler en lecture ou en écriture simultanément sur le même fichier texte
#include <conio.h>
#include <iostream>
#include <sstream>          // nécessaire pour la fonction getline()
#include <fstream>          // fichier d'entête pour gérer les flots d'entrées/sorties

using namespace std;

int main()
{
        string chaine;
        fstream flux;

        flux.open("test1.txt",ios::in | ios::out |ios::trunc);// Ouverture en lecture, écriture et effacement du fichier s'il existe
        if (! flux)
        {
                cout << "Erreur d'ouverture de fichier " << endl;
                _getch();
                return 1 ;
        }
        flux << "Salut la compagnie\nBonjour les IRIS "<< endl;

        flux.clear( );                          // Remise à zéro des bits d'erreurs
        flux.seekg(0);                          // on positionne le pointeur de lecture au debut

        // On relit tout
        cout <<"Voici ce que contient le fichier " << endl ;
        while ( ! flux.eof() )
        {
                if (getline(flux , chaine))     // si on a pu lire une chaine de caractères
                        cout << chaine << endl; // on l'affiche
                else break;                     // on sort de la boucle si erreur
        }

        flux.clear( );                          // Remise à zéro des bits d'erreurs
        flux.seekp(0);                          // On positionne le pointeur d'écriture au début
        flux << "coucou "<< flush;              // On écrit coucou au début du fichier

        flux.seekg(0);                          // on positionne le pointeur de lecture au début

        // On relit tout
        cout <<"\n\nVoici maintenant ce que contient le fichier" << endl ;
        while ( ! flux.eof() )
        {
                if (getline(flux , chaine))     // si on a pu lire une chaine de caractères
                        cout << chaine << endl; // on l'affiche
                else break;                     // on sort de la boucle si erreur
        }

        flux.close();
        _getch( );   return (0);
}
```