[../icons/ujf.gif] [../icons/ujf.gif]

Les Fichiers

Licence de Physique/M2 ITDD

Table des matières

1 Les fichiers

Il peut être utile d'écrire et/ou de lire des informations dans un fichier. Ce fichier peut être de **type texte ou binaire** (*formaté*). L'avantage du premier est sa lisibilité (un simple éditeur permet de visualiser son contenu) ; le second est plus économique en place occupée sur le disque dur ; L'inconvénient est que la lecture est plus délicate : il faut connaître la structure des données qu'il contient.

1.1 Fichiers Texte

1.1.1 Écriture

```
#include<iostream>
#include<fstream> // utilisation des fichiers
using namespace std;
void EcrireNb()
{
   ofstream Sortie("nombre.txt");// ouverture en ecriture
   Sortie<<2.5<<" "<<3.8<<endl;
   Sortie<<5.2<<" "<<5.3<<endl;
   Sortie<<1.5<<" "<<2.901<<endl;
   Sortie.close(); // fermeture du fichier
}
void Ecrire()
{
   ofstream Sortie("bidon.txt");// ouverture en ecriture
   Sortie<<"Voici une ligne"<<endl;
   Sortie<<"et voici une autre ligne"<<endl;
   Sortie.close(); // fermeture du fichier
}
```

Remarques :

- L'include `<fstream>` est nécessaire pour l'utilisation des fichier (f=*file*, *stream*=flot).
- *Sortie* est un objet de la classe **ofstream** (=*out file stream*) ; cela crée un nouveau fichier (ici, *nombre.txt* ou *bidon.txt*) et l'ouvre en écriture. Si le fichier existe déjà, il est écrasé.
- Noter que la syntaxe est la même que celle utilisée pour le **cout**.
- Observer le contenu des fichiers *nombre.txt* et *bidon.txt*.

1.1.2 Lecture

```
#include<iostream>
#include<fstream> // utilisation des fichiers
using namespace std;
void LireNb()
{
   ifstream Entree("nombre.txt");// ouverture en lecture.
   double x,y;
   while (Entree>>x>>y)
      cout<<x<<" + "<<y<<" = "<<x+y<<endl;
   Entree.close(); // fermeture du fichier
}
void LireMot()
{
   ifstream Entree("bidon.txt");// ouverture en lecture.
   char mot[80];
   cout<<"--------- Mot par Mot -------"<<endl;
   while (Entree>>mot)
      cout<<mot<<endl;
   Entree.close(); // fermeture du fichier
}
void LireLigne()
{
   ifstream Entree("bidon.txt");// ouverture en lecture.
   char ligne[80];
   cout<<"--------- Ligne par Ligne -------"<<endl;
   while (Entree.getline(ligne,80))
      cout<<ligne<<endl;
   Entree.close(); // fermeture du fichier
}
```

Remarques :

- *Entree* est un objet de la classe **ifstream** (=i*nput file stream*) : le fichier (*nombre.txt* ou *bidon.txt*) est ouvert en lecture.
- La syntaxe de la lecture est la même que celle du cin dans les 2 premières fonctions ; ainsi, les **espaces servent de délimiteur** pour les diverses entrées.
- Remarquer la syntaxe du `while(Entree>>x>>y)` : l'on affecte les variables x et y avec ce vient du flot *Entree* et si tout ce passe bien `Entree>>x>>y` renvoie une valeur non nulle (le **while** continue). Par contre en cas d'erreur (ici la fin du fichier) cette commande renvoie 0 permettant de sortir du **while**.
- La fonction *LireLigne()* utilise la méthode **ifstream : :getline()** ; cela permet de lire une ligne entière (jusqu'au caractère de fin de ligne) en gardant les espaces. Cette méthode prend comme premier argument une chaîne de caractères et comme second, le nombre de caractères lus au maximum (donc plus petit que la longueur de la chaîne).

1.1.3 Lecture/Écriture

Enfin il est possible ouvrir un fichier en lecture et écriture simultanément en utilisant la classe **fstream** :

```
void LireEcrire()
{
   fstream EntreeSortie("bidon.txt",ios::in|ios::out);
   char ligne[80];
   EntreeSortie.getline(ligne,80);
```

```
    EntreeSortie<<"une nouvelle ?"<<endl;
    EntreeSortie.close(); // fermeture du fichier
}
```

Remarques :

- Dans l'ouverture du fichier (classe **fstream**) nous avons rajouté un argument : `ios::in|ios::out` ; **ios** est une classe et **in** et **out** sont 2 attributs de celle-ci signifiant respectivement ouverture en lecture et écriture (remarquer qu'un **ifstream** se déclare aussi comme *ifstream Entree("bidon.txt",ios::in)* et de même pour les **ofstream** (avec **ios : :out**).
- La fonction *LireEcrire()* lit la première ligne et en écrit une nouvelle juste après. Observer ce qu'est devenue l'ancienne deuxième ligne.

1.2 Fichiers binaires

Voici 2 fonctions pour illustrer l'écriture et la lecture en binaire.

```
void EcrireBin()
{
   ofstream Sortie("bidon.bin");
   int i=3;
   Sortie.write((char*)&i,sizeof(int));
   double x[3]={0.54,2.8,15.5};
   Sortie.write((char*)x,sizeof(x));
   Sortie.close();
}
void LireBin()
{
   ifstream Entree("bidon.bin");
   int j;
   Entree.read((char*)&j,sizeof(j));
   cout<<j<<endl;
   double y[3];
   Entree.read((char*)y,sizeof(y));
   cout<<y[0]<<" "<<y[1]<<" "<<y[2]<<endl
   cout<<"----- Acces Direct -----"<<endl;
   double z;
   Entree.seekg (sizeof(int)+sizeof(double),ios::beg) ;
   Entree.read((char*)&z,sizeof(z));
   cout<<z<<endl;
   Entree.close();
}
```

Remarques :

- pour l'écriture, on utilise la méthode **ofstream : :write()** ; cette méthode admet comme premier argument un **char*** et comme second la taille de ce qu'on veut écrire. Dans l'exemple donné, *&i* est un **int*** ; on le convertit en **char*** (on parle de cast). De même pour le tableau x (qui est un **double***).
- pour la lecture, on commence par lire l'entier (toujours à l'aide d'un **cast**) puis les 3 doubles (**lecture séquentielle**). Ensuite on donne un exemple de **lecture à accès direct** : il s'agit d'aller lire un élément à un endroit donné du fichier sans avoir besoin de le lire depuis le début. Pour cela, on utilise la méthode **ifstream : :seekg()** : on saute le nombre d'octets voulus (ici 1 entier et 1 double) à partir du début (**ios : :beg**) [1] et on lit un **double** comme précédemment.

1. il existe aussi ios : :end et ios : :cur pour "depuis la fin" ou "depuis la position courante".