DAUPHINE UNIVERSITÉ PARIS | TUNIS

Le langage C++
Master Actuariat

Séance 5 : Les flots

Les entrées sorties en C++

✓ Les entrées et les sorties sont considérées comme des **flots**. Un flot est associé à un **fichier** ou à un **périphérique**.

★ Les flots d'entrée sont de type **istream**, comme par exemple **cin**.
- Le clavier est un istream qui se nomme **std : :cin**.

★ Les flots de sortie sont de type **ostream**, comme par exemple **cout**.
- L'écran est un ostream qui se nomme **std : :cout**.
- La sortie d'erreur est un ostream qui se nomme **std ::cerr**.

✓ La classe **iostream** permet de combiner *flots de sortie et flots d'entrée*.

Les entrées sorties en C++

Possibilité de modifier la façon dont les éléments sont lus ou écrits dans le flot :

dec	lecture/écriture d'un entier en décimal
oct	lecture/écriture d'un entier en octal
hex	lecture/écriture d'un entier en hexadécimal
endl	insère un saut de ligne et vide les tampons
setw(int n)	affichage de *n* caractères
setprecision(int n)	affichage de la valeur avec *n* chiffres avec éventuellement un arrondi de la valeur
setfill(char)	définit le caractère de remplissage
flush	vide les tampons après écriture

```
#include <iostream.h>
#include <iomanip.h>  // attention a bien inclure cette librairie
int main() {
  int i=1234;
  float p=12.3456;
  cout << "|" << setw(8) << setfill('*')
       << hex << i << "|" << endl << "|"
       << setw(6) << setprecision(4)
       << p << "|" << endl;
}
```

```
|*****4d2|
|*12.35|
```


Les fichiers en C++

C++ fournit les classes suivantes pour gérer des E/S sur les fichiers :

- **ofstream** : classe stream pour écrire sur les fichiers
- **ifstream** : classe stream pour lire à partir de fichiers
- **fstream** : classe stream à la fois lire et écrire à partir de / vers des fichiers.

Ces classes sont dérivées directement ou indirectement des classes **istream** et **ostream**.
On a déjà utilisé des objets dont les types étaient de ces classes : cin (un objet de la classe istream) et cout (un objet de classe ostream). On pourra donc utiliser les flux de fichiers de la même façon.

Les fichiers en C++

✓ En C++, on peut utiliser les fichiers de C (type FILE *, utilise la notion de pointeur), ou les fichiers de C++ (types ofstream, ifstream, utilise la notion d'objet).

✓ Les **fichiers** sont des flots particuliers.

✓ Pour **accéder** aux fichiers il faut inclure l'entête
#include <fstream>

✓ On distingue :
- **Les fichiers textes :** contiennent des informations sous la forme de caractères (en général en utilisant le code ASCII). Le source d'un programme C++ est un exemple de fichier texte.
- **Les fichiers binaires** : contiennent directement la représentation mémoire des informations. Un fichier binaire peut aussi être vu comme une séquence d'octets. Un exemple de fichier binaire est le résultat de la compilation d'un programme C++ (programme exécutable).

Les fichiers texte en C++

Écrire dans un fichier :

✓ Les **écritures** sur un fichier se font avec un objet **ofstream**.
✓ Pour **ouvrir un fichier** on appelle le constructeur
ofstream (const char * Filename, mode)

✓ Les différents **modes d'ouverture** sont :
- **ios : :app** (append) ajout à la fin du fichier.
- **ios : :ate** (at end) met l'index à la fin du fichier.
- **ios : :binary** (binary) Pour ouvrir un fichier binaire.
- **ios : :in** (input) permet la lecture
- **ios : :out** (output) permet l'écriture.
- **ios : :trunc** (truncate) vide le fichier à l'ouverture.

Remarque :
✓ Un **ofstream** a par défaut un mode d'ouverture **ios_base::out | ios_base::trunc** ce qui signifie ouverture en écriture et effacement du contenu du fichier.

Les fichiers en C++

Il est conseillé avant toute opération sur le flux de tester les indicateurs d'état en appelant une de ces méthodes :

- **bad() :** Renvoie **true** si une opération de lecture ou d'écriture échoue. Par exemple dans le cas que nous essayons d'écrire dans un fichier qui n'est pas ouvert en écriture.
- **fail() :** Retourne la valeur **true** dans les mêmes cas que bad(), mais aussi dans le cas où une erreur de format se produit, comme quand un caractère est extrait et on essaye en fait de lire un nombre entier.
- **eof() :** Renvoie **true** si un fichier ouvert en lecture a atteint la fin.
- **good() :** C'est le drapeau d'état le plus générique. Il retourne **false** lorsque les méthodes précédentes retourneraient **true**.

Écriture simple dans un fichier texte

```
ofstream monFlux("C:/Testcpp/scores.txt"); // équivalent à ofstream
//monFlux("C:/Testcpp/scores.txt" , ios::out);
if(monFlux.good())
{
//Tout est prêt pour l'écriture.
}
else
{
cerr << "ERREUR: Impossible d'écrire dans le fichier." <<endl;
exit(1);   // arrêt du programme avec un code d'erreur différent de 0
}
```

Écriture dans les fichiers textes

✓ L'écriture dans un fichier s'effectue comme l'affichage à l'écran avec l'opérateur **<<**

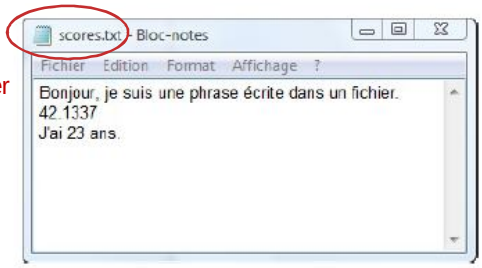
✓ Toute variable ou constante des types de base (bool, char, int, float, double, string,...) peut être écrite.

Syntaxe

NomFlux << Variable;

Exemple :

Nom physique du fichier

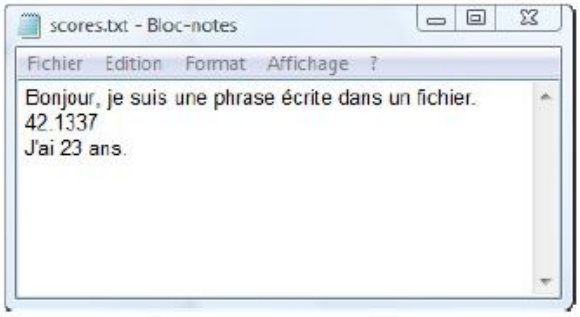



```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main()
{
ofstream monFlux("C:/Testcpp/scores.txt");
if(monFlux.good())  // ou encore if(monFlux)
                    // on teste si tout est OK
{
monFlux << "Bonjour, je suis une phrase écrite dans un fichier." << endl;
monFlux << 42.1337 << endl;
int age(23);
monFlux << "J'ai " << age << " ans." << endl;
}
else
{
cerr << "ERREUR: Impossible d'ouvrir le fichier." << endl;
}
monFlux.close();        // INDISPENSABLE DANS TOUS LES LANGAGES
                        // (mais C++ le fera automatiquement si on l'oublie)
return 0;
}
```

Si le fichier n'existait pas, le programme le créerait automatiquement ! Par contre, il faut que le dossier existe. Le dossier C:/Testcpp/doit exister.

Impression des données dans le fichier scores.txt

A l'exécution ...



Que se passe-t-il si le fichier existe déjà ?

Il sera **supprimé et remplacé** par ce que vous écrivez, ce qui est problématique !!! si l'on souhaite ajouter des informations à la fin d'un fichier pré-existant.

```
ofstream monFlux("C:/Testcpp/scores.txt",ios::app);
```


Les fichiers texte en C++ : Lire un fichier

Ouvrir un fichier en lecture ...

Pour ouvrir un flot en entrée (lecture) il faut définir un objet **ifstream** puis l'ouvrir

```
ifstream monFlux("C:/Testcpp/scores.txt"); //Ouverture d'un fichier en lecture
// équivalent à ifstream monFlux("C:/Testcpp/scores.txt" , ios::in);
if(monFlux.good())
{
//Tout est prêt pour la lecture.
}
else
{
cerr << "ERREUR: Impossible d'ouvrir le fichier en lecture." <<endl;
exit(1);  // arrêt du programme avec un code d'erreur différent de 0
}
```

Remarque :

Un **ifstream** a par défaut un mode d'ouverture **ios_base::in** ce qui signifie lecture seule.

Lecture dans les fichiers textes

✓ La **lecture dans un fichier** texte s'effectue comme la lecture au clavier avec l'opérateur **>>**.

✓ **Toute variable** ou constante des types de base (bool, char, int, float, double, string,...) **peut être lue**.

Syntaxe :

NomFlux >> Variable;

✓ La lecture s'arrête au premier espace ou au premier caractère qui ne peut pas faire partie de la représentation ASCII du type lu (ce sera alors le prochain caractère à lire).

Lecture dans les fichiers textes

La fonction qui permet de **lire un fichier texte ligne par ligne** : **getline()**.

Syntaxe : **bool getline(ifstream &, string & line)**

getline renvoie un **bool** indiquant si l'on peut continuer à lire :
True : il reste encore des données
False : c'est la fin du fichier

Exemple : Afficher le contenu du fichier **scores.txt**

Solution 1

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main()
{
string ligne; //Une variable pour stocker les lignes lues
ifstream monFlux("C:/test_cpp/scores.txt");
getline(monFlux, ligne);
while (!monFlux.eof())    //tq ce n'est pas la fin du fichier
{
cout << ligne << endl;
getline(monFlux, ligne)
}
system("PAUSE");
return 0;
}
```


Solution 2

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main()
{ string ligne; //Une variable pour stocker une ligne entière
ifstream monFlux("C:/test_cpp/scores.txt");
getline(monFlux, ligne);   // ligne reçoit la 1ère ligne écrite dans le fichier
double nbre;
monFlux>>nbre;  // nbre reçoit le nombre flottant
while (!monFlux.eof())    //tq ce n'est pas la fin du fichier
{
cout << ligne << endl;
cout<<nbre<<endl;
monFlux.ignore() // ignore permet d'ignorer les cc restants sur une lige et ainsi on peut
                 //réutiliser getline
getline(monFlux, ligne)  // permet de stocker la 3ème phrase dans ligne
cout << ligne << endl;
}
system("PAUSE");
return 0;
}
```

Les fichiers en C++ : Open() et close()

En C++ Les fichiers ouverts sont automatiquement refermés lorsque l'on sort du bloc où le flux est déclaré. Il arrive par contre qu'on ait besoin de fermer le fichier avant sa fermeture automatique

Pour fermer un fichier on utilise la méthode close( ).

Nom_flux.close()

De la même manière, il est possible de retarder l'ouverture d'un fichier après la déclaration du flux en utilisant la fonction open().

Nom_flux.open()

```
void f()
{
ofstream flux; //Un flux sans fichier associé
flux.open("C:/data.txt"); //On ouvre le fichier C:/data.txt
//Utilisation du fichier
flux.close(); //On referme le fichier. On ne peut plus écrire dans le fichier à
                //partir d'ici
}
```

Les fichiers en C++ : Exercices

1- Ecrire un programme C++ qui permet de saisir et de mémoriser dans un fichier nommé **répertoire**, le nom et le numéro de téléphone de 5 personnes.

2- Ecrire un programme C++ qui permet **d'afficher** à l'écran le contenu du fichier **répertoire**, en sautant une ligne entre chaque personne.

```
#include <iostream>
#include<iomanip>
#include <fstream>
#include <string>
using namespace std;
int main()
{ string nom;  int tel;
ofstream repfile("C:/test_cpp/repertoiretxt");
// ou ofstrem repfile;
// repfile.open()= ("C:/test_cpp/repertoire.txt");
if(repfile)
{ for (int i=0;i<5;i++)
{ cout<<"Personne N° "<<i+1<<endl;
cout<<"Nom: ";
cin>>nom;
repfile<<nom<<" ";
cout<<"Telephone : ";
cin>>tel;
repfile<<tel<<endl;
} }
else
cout << "ERREUR: Impossible d'ouvrir le fichier." << endl;
repfile.close();
system("PAUSE");
return 0;
}
```

Exécution

Personne N° 1
Nom : XX
Telephone : 23253

Personne N° 2
Nom :YY
Telephone : 235648

…..

Personne N°5
Nom : LL
Telephone : 23654

repertoire.txt

XX 23253
YY 235648

…..

LL 23654

```
#include <iostream>
#include<iomanip>
#include <fstream>
#include <string>
using namespace std;
int main()
{
ifstream repfile("C:/test_cpp/tel.txt");
string nom;
int tel;
//  ou string line;
if(repfile)
{
repfile>>nom>>tel;        //  getline(repfile,line);

while(!repfile.eof()) //Tant qu'on n'est pas à la fin, on ramène les lignes
{
cout << nom <<" "<<tel<<endl;  //cout<<line;
repfile>>nom>>tel;    //getline(repfile,line);
}
}
else
cerr<<"ERREUR"<<endl;
system(« pause »);
repfile.close();
return 0;
}
```

Les fichiers binaires en C++

La manipulation d'un fichier binaire s'effectue presque comme celle d'un fichier texte. La différence réside au niveau de la désignation du fichier et l'écriture (resp. la lecture) des données.

- << ➔ **f.write**(...)
- >> ➔ **f.read**(...)

Pour créer un fichier binaire il faut ajouter la clause **ios : :binary**

```
ofstream monFlux("C:/Testcpp/scores.txt",ios::binary);
```

Pour écrire une variable ***v*** (quel que soit son type)

```
monFlux.write((char *)&v,sizeof(v));
```

Pour lire une variable ***v*** à partir d'un fichier (quel que soit son type)

```
monFlux.read((char *)&v,sizeof(v));
```


Les fichiers : Curseur dans le fichier

Ces fonctions sont particulièrement utiles dans un fichier binaire où les données ont en général toutes la même taille.

❶Connaître sa position

Ces fonctions renvoient **la position de la tête** d'écriture (resp.lecture) (exprimée en **nombre d'octets** depuis le début du fichier, le premier octet ayant le numéro 0)

Pour ifstream	Pour ofstream
tellg()	tellp()

Syntaxe :

```
Nom_flux.tellp();
```

```
ofstream fichier("C:/data.txt");
int position = fichier.tellp();          //On récupère la position
cout << "Nous nous situons au " << position << "eme caractere du fichier." <<endl;
```

Les fichiers : Curseur dans le fichier

❷Se déplacer

Pour ifstream	Pour ofstream
seekg()	seekp()

Syntaxe : **Nom_flux.seekp**(nombred'octets, position);

Les trois **positions** possibles sont :
- le début du fichier : **ios::beg** ;
- la fin du fichier : **ios::end** ;
- la position actuelle : **ios::cur**.

```
ifstream fichier("C:/data.txt");
fichier.seekg(10,ios::beg);      //se placer au 10ème octect (cc) après le début du fichier
fichier.seekg(20,ios::cur);      //aller 20 caractères plus loin que l'endroit où se situe le curseur
```


Les fichiers : Connaître la taille

Pour connaître **la taille d'un fichier**, on **se déplace à la fin** et on demande au flux de nous dire où il se trouve.

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
ifstream fichier("C:/meilleursScores.txt");      //On ouvre le fichier
fichier.seekg(0, ios::end);                      //On se déplace à la fin du Fichier
int taille;
taille = fichier.tellg();
//On récupère la position qui correspond donc a la taille du fichier !
cout << "Taille du fichier : " << taille << " octets." << endl;
return 0;
}
```

Les fichiers binaires en C++ : Exercices

1- Ecrire un programme C++ qui permet de saisir et de mémoriser dans un fichier binaire nommé **rep_binary**, le nom et le numéro de téléphone de 5 personnes.

2- Ecrire un programme C++ qui permet **d'afficher** à l'écran le contenu du fichier binaire **rep_binary**, en sautant une ligne entre chaque personne.

```
#include <iostream>
#include<iomanip>
#include <fstream>
#include <string>
using namespace std;
int main()
{ // création du fichier binaire qui contient 10 nom et 10 num de tel
string nom;
int tel;
char *repbinary="C:/test_cpp/rep_binary.txt";
ofstream repfile(repbinary,ios::binary);   // ou encore ofstream repfile("C:/test_cpp/rep_binary.txt ");
if(repfile)
{
for (int i=0;i<5;i++)
{
cout<<"Personne N° "<<i+1<<endl;
cout<<"Nom:";
cin>>nom;
repfile.write((char*)&nom,sizeof(nom));
cout<<"Telephone :";
cin>>tel;
repfile.write((char*)&tel,sizeof(tel));
}
}
else
cout << "ERREUR: Impossible d'ouvrir le fichier." << endl;
system("PAUSE");
return 0;
}
```

```
#include <iostream>
#include<iomanip>
#include <fstream>
#include <string>
using namespace std;
int main()
{
// Programme d'afficher le contenu du fichier tel_binary.txt
string nom;
int tel;
char *repbinary="C:/test_cpp/tel_binary.txt";
ifstream repfile(repbinary,ios::in);
if(repfile)
{
repfile.read((char*)&nom,sizeof(nom));
repfile.read((char*)&tel,sizeof(tel));

while(!repfile.eof())
{cout << nom <<" "<<tel<<endl;
repfile.read((char*)&nom,sizeof(nom));
repfile.read((char*)&tel,sizeof(tel));
}
}
else
cerr<<"ERREUR"<<endl;
repfile.close();
system("pause");
return 0;
}
```