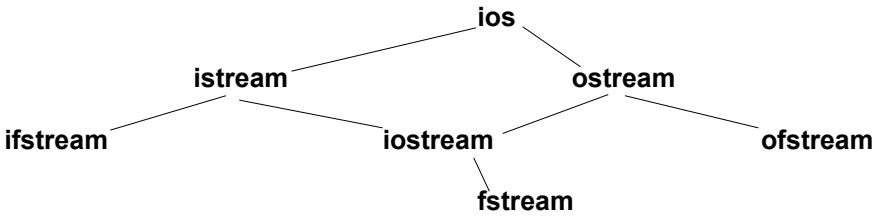
Fiche : LES FLOTS D'ENTREE ET DE SORTIE

1 – Entrées - sorties

Un flot est un "canal" recevant ou fournissant de l'information. Ce canal est associé à un périphérique ou à un fichier. Un flot d'entrée est un objet de type **istream** tandis qu'un flot de sortie est un objet de type **ostream.**
Le flot **cout** est un flot de sortie prédéfini, connecté à la sortie standard **stdout.**
De même, le flot **cin** est un flot d'entrée prédéfini, connecté à l'entrée standard **stdin**
et le flot **cerr** correspond à la sortie standard d'erreur.



Une portion de la hiérarchie de la classe de flux d'E/S.

Les "espaces", tabulations horizontale \t et verticale \v, fin de ligne \n, retour chariot \r et changement de page \f, servent de "délimiteurs" (comme dans *scanf*), y compris pour les chaînes de caractères.

2 – Gestion de fichiers par des flux

L'utilisation des classes *ofstream et ifstream* demande l'inclusion du fichier **fstream.h.**

1. ifstream : classe permettant de gérer des entrées à partir d'un fichier.

La classe *ifstream,* permet de créer un flot d'entrée associé à un fichier:

```
ifstream Fich ( char * nom_fichier , mode_ouverture );
```

```
ifstream Fich2 ("toto.txt"); // utilisation du constructeur avec le
                             //mode d'ouverture par défaut (ios::in)
```

On peut aussi relier un fichier et un mode plus tard en utilisant la fonction membre **`open()`** :

```
ifstream flot;
flot.open("exemple.txt", ios::app);
```

Modes d'ouverture du Fichier :

BIT de mode	ACTION
ios::in	Ouverture en lecture (par défaut pour la classe ifstream)
ios::out	Ouverture en écriture (par défaut pour la classe ofstream)
ios::app	Ouverture en ajout de données (écriture en fin de fichier)
ios::ate	Se place en fin de fichier après ouverture
ios::trunc	Si le fichier existe, son contenu est perdu
ios::nocreate	Le fichier doit exister
ios::noreplace	Le fichier ne doit pas exister (sauf si ios::ate ou ios::app est activé)

Principales méthodes associées

get (char & c) : extrait un caractère en entrée et le range dans c,
int get() : extrait un caractère et en renvoie la valeur (EOF si fin de fichier),
read (void *adr, int taille) : lit *taille* caractères et les range à *adr.*
getline(char *buffer, int taille) : permet de saisir une ligne de texte.
void flush() : vide le flux.
seekg (déplacement, *origine*) permet d'agir sur le pointeur de fichier pour objet de classe ifstream.

origine :	**ios::beg**	seek from beginning of file
	ios::cur	seek from current location
	ios::end	seek from end of file

2. ofstream : classe permettant de gérer des sorties à partir d'un fichier

La classe *ofstream* permet de créer un flot de sortie associé à un fichier :
ofstream flot (char * nom_fichier , mode-ouverture);

```
ofstream Fich ("tata.doc"); // utilisation du constructeur avec le
                            //mode d'ouverture par défaut
(ios::out)
```

On peut aussi relier un fichier et un mode plus tard en utilisant la fonction membre open() :

```
ofstream flot;
flot.open("exemple.txt", ios::app);
```

Principales méthodes associées

void flush() : vide le tampon sans retour chariot.

put(char c) : insère un caractère dans le flux.

write(char* buffer, int n) : insère n caractère de buffer dans le flux.

seekp (déplacement, ***origine***) permet d'agir sur le pointeur de fichier pour objet de classe ofstream.

origine :	**ios::beg**	seek from beginning of file
ios::cur	seek from current location	
ios::end	seek from end of file	

3. Autres méthodes utiles permettant de tester l'état d'un flux

int good() : retourne une valeur différente de zéro si la dernière opération d'entrée/sortie s'est effectuée avec succès et une valeur nulle en cas d'échec.

int fail() : fait l'inverse de la méthode précédente

int eof() : retourne une valeur différente de zéro si la fin de fichier est atteinte et une valeur nulle dans le cas contraire.

int bad() : retourne une valeur différente de zéro si vous avez tenté une opération interdite et une valeur nulle dans le cas contraire.

int rdstate() : retourne la valeur de la variable d'état du flux. Retourne une valeur nulle si tout va bien.

void clear() : remet à zéro l'indicateur d'erreur du flux. C'est une opération obligatoirement à faire après qu'une erreur se soit produite sur un flux.

Surdéfinition de **!** : permet de dire si la dernière opération s'est bien passée. Notamment l'ouverture d'un fichier.

Exemple :

```
if( !flot )
{
 cerr << "erreur ouverture" << endl ;
 exit(1);
}
```

Dans tous les cas, la méthode ***close*** permet de fermer le fichier.

```
flot.close()
```

3 – Formatage des données → Pour soigner la présentation

A chaque flot, est associé un "statut de formatage" constitué d'un mot d'état et de 3 valeurs numériques (gabarit, précision et caractère de remplissage). Voici les principaux bits du mot d'état :

NOM DE CHAMP	NOM DU BIT	SIGNIFICATION
ios::basefield	ios::dec ios::oct ios::hex	conversion décimale conversion octale conversion hexadécimale
	ios::showbase ios::showpoint	affichage indicateur de base(sortie) affichage point décimal (en sortie)
ios::floatfield	ios::scientific ios::fixed	notation "scientifique" notation "point fixe"

Action sur le statut de formatage :
On peut utiliser, soit des "manipulateurs" qui peuvent être "simples" ou "paramétriques", soit des fonctions membres. Les manipulateurs non paramétriques : ils s'emploient sous la forme :

```
flot << manipulateur; // voir l'exemple en fin de document
```

MANIPULATEUR	UTILISATION	ACTION
dec	Entrée/sortie	active le bit de conversion décimale
hex	"	active le bit de conversion hexadécimale
oct	"	active le bit de conversion octale
endl	sortie	insère un saut de ligne et vide le tampon
ends	sortie	insère un caractère de fin de chaîne (\0)

Les manipulateurs paramétriques : ils s'utilisent sous la forme :
istream & manipulateur (argument) ou ostream & manipulateur (argument).

MANIPULATEUR	UTILISATION	ROLE
setbase (int)	Entrées/sortie	définit la base de conversion
setprecision (int)	"	définit la précision des flottants
setw (int)	"	définit le gabarit (tombe à 0 après

Exemples

```
#include <stdio.h>
#include <cstring>                  //strcmp()
#include <iostream>
#include <fstream>
using namespace std;

int main( void )
{
     ofstream SortieFich("Sortie.txt");
     ifstream EntreeFich("Sortie.txt");
     char ch[80];
     cout << "bonjour"<< endl;
     cout << "texte =" ;
//Ecriture
     do
     {
       cin >> ch;
       SortieFich << ch << endl;
     } while(strcmp(ch, ".") !=0);
     SortieFich.close();

//lecture
     char c;
     while (EntreeFich.get(c))
           cout << c;
//autre solution
/*   while (!EntreeFich.eof())
     {
           EntreeFich >> ch;       //attention aux espaces correspondants
                                   // à des fins de lignes
           cout << ch << endl;
     }                             */
```

```
      EntreeFich.close();
      puts("Pressez touche");
      cin >>c;
}
```

```
// Lecture et affichage du contenu d'un fichier séquentiel.
#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
#include <stdlib.h>

void sortieLigne( int, const char *, double );

int main()
{
   int compte;
   char nom[ 30 ];
   double solde;
// Le constructeur de ifstream ouvre le fichier.  ios::in est optionnel
   ifstream entreeFichierClient( "clients.dat" );

   if ( !entreeFichierClient )  // opérateur ! est surchargé
   {
      cerr << "Le fichier n'a pas pu être ouvert.\n";
      exit( 1 );
   }
   cout << setiosflags( ios::left ) << setw( 10 ) << "Compte"
        << setw( 13 ) << "Nom" << "Solde\n";
   while ( entreeFichierClient >> compte >> nom >> solde )
      sortieLigne( compte, nom, solde );

   return 0;  // Le destructeur de ifstream ferme le fichier.
}

void sortieLigne( int cpt, const char *nom, double sld )
{
   cout << setiosflags( ios::left ) << setw( 10 ) << cpt
        << setw( 13 ) << nom << setw( 7 ) << setprecision( 2 )
        << resetiosflags( ios::left )
        << setiosflags( ios::fixed | ios::showpoint )
          << sld << '\n';
}
```

Contenu du fichier "clients.dat"

100 Dupond 24.98
200 Dupont 345.67
300 Goulet 0
400 Dupuis -42.16
500 Breton 224.62