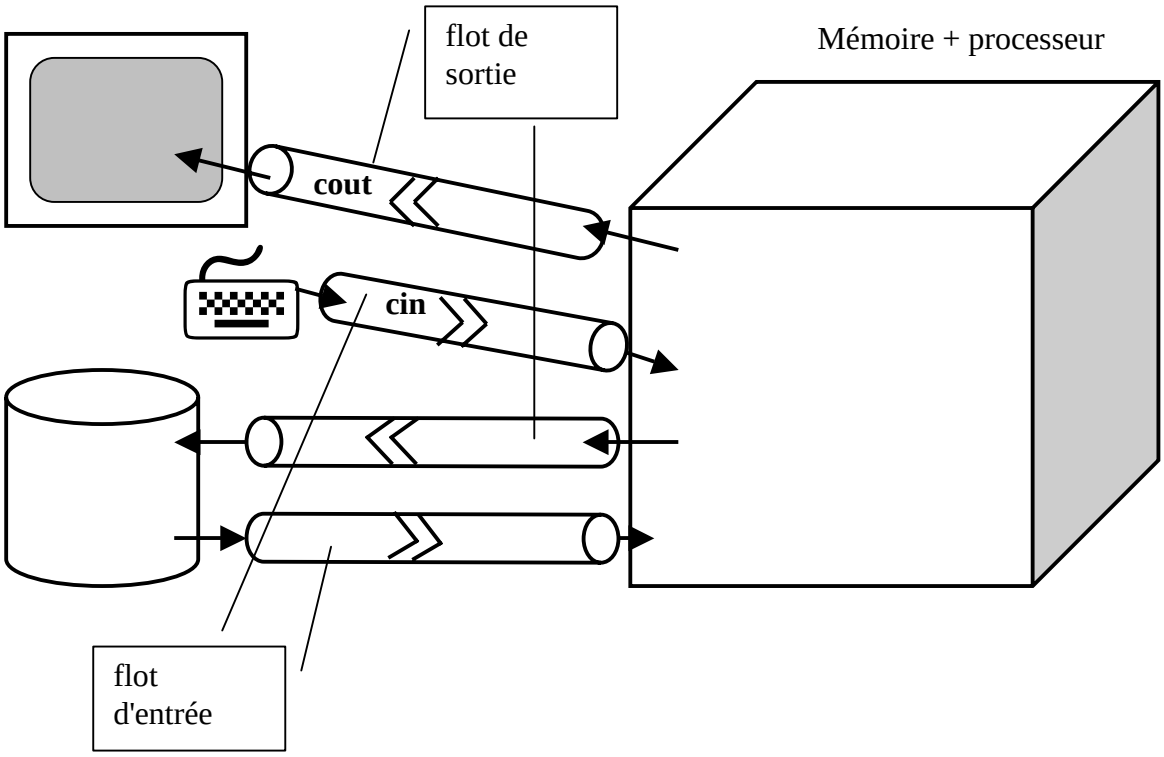
Les fichiers en C++

Généralité sur les flots

Les entrées et sorties en C++ se font toujours par l'intermédiaires de **flots** (ou flux), qu'on peut considérer comme des canaux
- recevant de l'information dans le cas d'un flot de sortie
- fournissant l'information dans le cas d'un flot d'entrée

Le flot de sortie standard est **cout**. Ce flot est connecté par défaut à l'écran.
Le flot d'entrée standard est **cin**. Ce flot est connecté par défaut au clavier.

Les opérateurs de manipulation des flots, qui permettent le transfert de l'information sont
- << qui permet l'écriture sur un flot de sortie
- >> qui permet la lecture sur un flot d'entrée



Ouverture et fermeture d'un fichier

Pour utiliser les fichiers en C++, il est nécessaire d'inclure le fichier d'en-tête **<fstream.h>.**
Ce fichier inclut lui-même <iostream.h>: donc lorsqu'on inclut fstream.h, il ne faut pas inclure iostream.h

- Ouvrir en lecture

ifstream *nom_fichier_logique* ("*chemin_fichier_physique*");

entre parenthèses et entre guillement, on indique le chemin physique du fichier, c'est-à-dire :
- le nom du fichier
- suivi de son extension
- et, si le fichier n'est pas contenu dans le même répertoire que l'exécutable, son chemin absolu complet (ex:" c:\travail\essai.txt")

Remarque : si le fichier est dans le même répertoire que le programme, seul le nom de ce fichier est utile.

ex: ifstream test ("essai.txt");

➢ Ouvrir en écriture

création d'un nouveau fichier ou écriture par dessus le fichier s'il existe déjà

ofstream *nom_fichier_logique* ("*chemin_fichier_physique*");

ex: ofstream test ("essai.txt")

➢ Ouvrir en mode ajout

ofstream <nom_fichier_logique> ("chemin_fichier_physique", **ios::in**|**ios::app**);

ex: ofstream test ("essai.txt", ios::in, ios::app);

➢ Ouvrir en mise à jour

La mise à jour (modifications ou suppression d'une donnée) d'un fichier en C++ est très délicate à effectuer directement. Le plus simple est de charger tout le fichier en mémoire (mode lecture), dans un tableau d'enregistrements par exemple, de faire les modifications en mémoire, puis de réécrire toutes les données modifiées à la place de l'ancien fichier (mode écriture).

➢ Fermeture d'un fichier

*nom_fichier_logique***.close( );**

ex: test.close( );

Manipulation d'un fichier

➢ Test de la fin d'un fichier

*nom_fichier_logique***.eof( )**

➢ lecture à partir d'un fichier

nom_fichier_logique >> *variable1* [>> *variable2*>> …];

ex: test >>nom>> prenom;
Signification : on lit à partir du fichier appelé test, les variables nom et prenom

☝ Rappelons que l'espace et le saut de ligne sont des séparateurs de lecture.

➢ écriture sur un fichier

nom_fichier_logique << *expression1* [<<*expression2*<< …];

☝ Il faut écrire soit même les différents séparateurs
(espaces ou sauts de ligne, ou tout autre caractère de séparation)

test<< "bidule"<<" "<<"machin"<<"\n"<<"truc"<<" "<<"chouette";
après cette instruction, le fichier nommé "essai.txt" va se présenter ainsi:

Attention!
En C++, on ne peut pas manipuler les structures globalement, même pas lors des opérations d'entrée-sortie dans des fichiers (contrairement à l'algorithmique). Il est donc nécessaire de lire et d'écrire séparément chaque champ élémentaire de donnée.

- **Exemple complet**

Voilà 2 programmes:
- l'un qui permet de saisir et mémoriser dans un fichier nommé répertoire, le nom et le numéro de téléphone de 10 personnes
- l'autre qui permet d'afficher à l'écran le contenu du fichier répertoire, en sautant une ligne entre chaque personnes

```
# include <fstream.h>    //ne pas inclure iostream !!
main ( )
{
// déclaration de variables permettant la communication entre le fichier et la mémoire centrale
string nom;
string tel;

ofstream fichrep ("repertoire.txt");   // ouverture du répertoire en écriture

for(int i=0; i<10; i++)
{
    cout << "\npersonne "<< i+1 << ":\n"
    cout << "nom? ";
    cin >> nom;
    fichrep <<nom << " ";
    cout << "\ntelephone?";
    cin >> tel;
    fichrep << tel << "\n";
}
fichrep.close();
}
```

```
# include <fstream.h>
# include <conio.h>
# include <cstring.h>
main( )
{
string nom;
string tel;

ifstream fichrep ("repertoire.txt");          // ouverture du fichier en lecture

fichrep >> nom >> tel;
while (! fichrep.eof( ) )// tant qu'on est pas arrivé à la fin du fichier…
{
      cout << nom << " \t"<< tel << "\n";          // …on affiche
      fichrep >> nom >> tel;              // et on lit les informations suivantes
}
fichrep.close( );
getch( );
}
```

TP 14 : les fichiers

Cet exercice va permettre la création et la consultation d'un ou plusieurs fichiers sur les joueurs du tournoi des 6 nations de rugby.

Il s'agira de créer deux petits programmes :

- l'un permettant de saisir et d'enregistrer dans un fichier les informations concernant joueurs des équipes de rugby du tournoi des 6 nations (France, Angleterre, Pays de Galles, Irlande, Ecosse, Italie).

- l'autre permettant de consulter le contenu du fichier crée par le programme précédent.

Premier programme : saisie

On veut mémoriser le nom, le prénom, la taille et le poids de chacun des 15 joueurs de l'équipe (on ne tient pas compte des remplaçants). Chaque joueur a un numéro de poste (de 1 à 15) qui détermine son rôle dans l'équipe (le numéro 1 est pilier gauche, le numéro deux est talonneur, …).

Le premier programme devra saisir les informations concernant tous les joueurs de la façon suivante:

France
joueur 1: *Christian Califano 104*
…
joueur 15: *Thomas Castaignede 78*

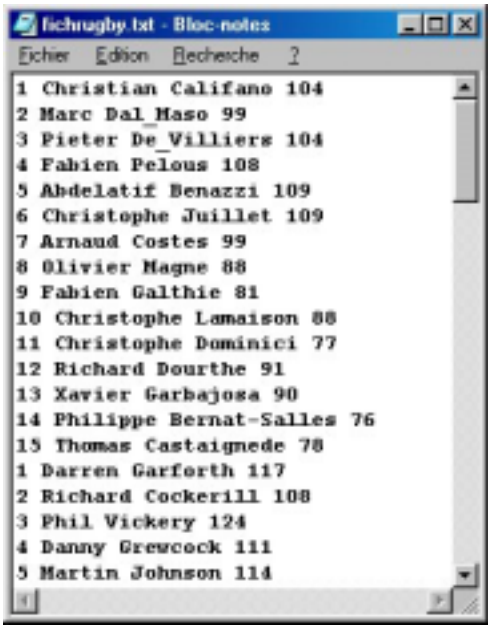
Angleterre
joueur 1: …

Vous avez le choix entre deux solutions pour mémoriser ces informations

- Première solution : un seul fichier pour les 6 équipes

Cette solution est artificielle mais a l'avantage de la simplicité. Les équipes sont enregistrées les unes après les autres (dans l'ordre : France, Angleterre, Pays de Galles, Irlande, Ecosse, Italie)

Le fichier devra se présenter de la manière suivante :



- Deuxième version : un fichier par équipe

Cette solution est plus réaliste. Il faut cependant gérer les noms de fichier de façon à ce que le traitement de saisie d'une équipe puisse être inclus dans une boucle. Le plus simple est d'utiliser un tableau de 6 éléments, contenant le chemin des 6 fichiers (France.txt, Angleterre.txt, …)

Deuxième programme : consultation

Une fois les informations enregistrées grâce au premier programme, un deuxième programme doit permettre la consultation du ou des fichiers en lecture. Ce programme propose à l'utilisateur les choix suivants :

1- afficher les 15 joueurs d'une équipe
2- afficher les 6 joueurs d'un poste donné
0- quitter

Remarque:

Il pourrait vous paraître étonnant d'utiliser des programmes pour créer et consulter un fichier texte simple. Dans ce cas précis, il est vrai qu'il serait bien plus simple d'utiliser directement le Bloc Notes. Mais dans la plupart des applications d'entreprise, les données ne sont pas saisies directement (dans un fichier texte ou dans une base de donnée) mais par l'intermédiaire d'un programme. De même, ces données ne sont pas visualisables directement, mais aussi par l'intermédiaire d'un programme. C'est pourquoi cet exercice vous entraîne sur un exemple simple, au codage de programmes de saisie et de consultation de fichiers textes.