Les flots d'entrées/sorties en C++

Les flots prédéfinis en C++ sont des objets des classes "**ostream**" et "**istream**":
Par exemple :

cout pour l'écran.
cin pour le clavier.

Ces classes disposent de 2 opérateurs (utilisables sur des objets) de direction: « **<<** » et « **>>** »
Ils permettent de diriger la variable (à droite) de l'opérateur vers le flot (à gauche) de l'opérateur
cout et **cin** sont donc des objets prédéfinis et peuvent être utilisés comme tel.

1.] Entrées sorties standards

1.1 Sortie standard « vers l'écran ».

Pour afficher à l'écran, on utilisera l'instruction suivante :

cout << variable ;

Ex : *cout* << « message à afficher » ;

Cette ligne de programme affichera le texte **message à afficher**.

L'écriture suivante est acceptée :

cout << « **\n**voici la valeur du code postal : » << postal << « **\n** »;

Elle provoquera l'affichage suivant :

```
Voici la valeur du code postal : 75018
```

L'écriture de la ligne suivante *cout* << ***manipulateur*** << *variable*
permet la mise en forme de l'affichage.
Dans un premier temps on se contentera des manipulateurs ***hex*** et ***dec***.
cout << *hex* << variable //affichera la variable en hexadécimal.
cout << *dec* << variable //affichera la variable en décimal.

1.2 Entrée standard « à partir du clavier ».

De la même manière, on peut effectuer des lectures à partir du clavier de la façon suivante :

char *ville ;
cin >> ville ;

Cette ligne de programme affectera à la variable « ville », les caractères frappés au clavier jusqu'à l'apparition d'un caractère **séparateur** (espace, retour chariot, fin de ligne, tabulation…).

Ex :

char *adresse ;
cin >> adresse ;

Si on frappe :

```
140 rue de la paix
```

On obtiendra : `adresse = «140»` car l'espace frappé après **140** est un séparateur de chaîne.
Le résultat n'est pas celui attendu.

La solution consiste à utiliser la méthode **getline** (..) de la classe '**istream'**:

*cin.**getline***(char *chaine , int nb , char carac_fin = '\n') ;

Cette instruction range dans la variable pointée par « chaine » **tous** les caractères frappés au clavier jusqu'à la rencontre du « carac_fin » (par défaut carac_fin = '\n' caractère de fin de ligne) **ou** lorsque le nombre de caractères frappés est égal à « nb ».

Ex :

```
char *adresse ;
cin.getline(adresse,10) ;  //ommission du caractere de fin car il possede une valeur par defaut ='\n'
```

Si on frappe

```
140 rue de la paix
```

On obtientadresse => |'1' |'4' |'0' |' ' |'r' |'u' |'e' |' ' |'d' |'e' |'\0' |

```
char *adresse ;
cin.getline(adresse,100) ;
```

Si on frappe

```
140 rue de la paix
```

On obtientadresse => |'1' |'4' |'0' |' ' |'r' |'u' |'e' |' ' |'d' |'e' |' ' |'l' |'a' |' ' |'p' |'a' |'i' |'x' |'\0'|

2.] ENTREES/SORTIES SUR LES FICHIERS

Introduction

- Les fichiers stockées sur les disques sont identifiés dans le système de fichiers : ils possèdent un chemin et un nom connu du système d'exploitation, ex : "/--/cpp/essai.cpp" .

Ouvrir un fichier consiste à le charger en mémoire vive de l'ordinateur à partir d'une adresse fixée par le système. Pour utiliser le fichier ouvert le système d'exploitation fournit au processus un numéro (qui renvoie à une variable pointant sur cette adresse) : c'est **l'identificateur du fichier**.

- Les fichiers manipulés sont à accès séquentiel : la lecture ou l'écriture dans un fichier ouvert utilise un **pointeur sur la position courante** dans le fichier. Ce pointeur est placé automatiquement au début du fichier à l'ouverture et se déplace au fur et à mesure des opérations sur le fichier jusqu'à la fin de fichier constituée d'un caractère spécial : **EOF** (EOF = -1).

- En C++, l'accès à des fichiers en lecture et en écriture se fait à l'aide des flots de fichiers d'entrée (*ifstream*) pour la lecture, de sortie (*ofstream*) pour l'écriture , d'entrée/sortie (*fstream*).

Ces flots n'utilisent qu'un seul tampon pour accéder aux fichiers. Par conséquent, il n'existe qu'une seule position dans le fichier, qui sert à la fois en lecture et en écriture.

Déclaration de variables "fichiers"

```
#include <fstream>
ifstream  entree;                  // fichier en lecture
ofstream  sortie;                  // fichier en écriture
fstream  entree_sortie;            // fichier en lecture/écriture
```

- On peut ouvrir le fichier au moment de la déclaration :

```
ifstream  entree ( "fichier_a _lire" ) ;          // fichier en lecture
ofstream  sortie ( "fichier_a _ecrire" );          // fichier en écriture
fstream  entree_sortie ( "fichier_a_modifier" );// fichier en lecture/écriture
```

- ou bien utiliser la fonction **open( )** selon la syntaxe suivante :

```
entree.open ( "fichier_a _lire" );
sortie.open  ( "fichier_a _ecrire " );
entree_sortie.open ( "fichier_a_modifier" );
```

- **Il faut toujours tester si l'ouverture du fichier a été effectuée sans erreur !!** avant d'envisager des opérations sur le fichier. Le cas d'erreur le plus fréquent est l'ouverture d'un fichier en lecture avec un nom ou un chemin erroné :

```
if ( !entree )  cout << "problème d'ouverture";
```

ou

```
if ( entree . fail ( ) )   cout << "problème d'ouverture";
```

L'état d'un flot est **eof ( ) , fail ( ) , bad ( ) , good ( )**

ex : *if (entree.eof( )) cout << "fin de fichier atteinte"; // teste la fin du fichier*

- **Modes d'ouverture** :

Lors de l'ouverture d'un fichier, on peut spécifier comme deuxième paramètre certains modes particuliers :

ios::in : lecture au début du fichier (**par défaut pour ifstream**)
ios::out : écriture au début du fichier (**par défaut pour ofstream**)
ios::app : ajout de données ou écriture à la fin du fichier
ios::trunc : efface un fichier existant (**par défaut pour ofstream**)
ios::ate : se positionne à la fin du fichier
ios::binary : ouverture de fichiers "binaires" et non "texte"

- **Exemple :**

```
#include <iostream>
#include <ifstream>
using namespace std ;
```

```
int main ()
{
    ifstream entree;
    char nom[10];

    cout << "nom du fichier : ";
    cin >> nom;
    entree.open(nom);
    if (!entree)
            cout << "le fichier n'existe pas!" << endl;
    return 0;
}
```

● **Remarques :**

- On aurait pu ouvrir le fichier par : i*fstream entree(nom);*

- Pour ouvrir un flot à la fois en lecture et en écriture sur un fichier (pour permettre l'une *ou* l'autre de ces opérations), on doit déclarer un flot de type **fstream** et spécifier la réunion des **deux modes d'accès**.

```
fstream   entree_sortie ("nom_fichier", ios::out | ios::in);
```

Lecture/écriture dans les fichiers

● Les **opérations d'entrée (lecture)** se font à l'aide de l'opérateur : **>>**
Les **opérations de sortie (écriture)** se font à l'aide de l'opérateur : **<<**

● Les manipulateurs définis dans **iomanip.h** (voir chapitre Entrées/Sorties standards) peuvent être utilisés pour lire ou écrire dans les fichiers.

```
entree >> i >> x;                 // lecture
sortie << setw(4) << x << endl;   // écriture
entree_sortie >> j;
entree_sortie << x;               // Lecture suivie d'écriture
```

● Les flots possèdent également d'autres fonctions de lecture et écriture parmi lesquelles :

-lecture/écriture d'un caractère : **get** et **put**

ex *:* `char tampon;  fi.get(tampon);  fo.put(tampon);`

- lecture d'une ligne dans le flot : **getline ( )**

Déplacement dans un fichier

- Le déplacement de la position courante dans le fichier ouvert "***seek***" peut s'effectuer en absolu ou en relatif.
- On peut obtenir la valeur de la position courante : "***tell***".

- Pour un fichier en écriture on dispose des fonctions : **seekp** et **tellp**

```
sortie.seekp( 0);          // positionnement absolu en 0, c'est-à-dire au début
sortie.seekp(0, ios::end); // positionnement à la fin
sortie.seekp(0, ios::beg); // positionnement au début
sortie.seekp(3, ios::cur); // avance de 3
```

Remarque : il n'est pas possible de déplacer la position dans un fichier ouvert en mode **ios::app**

- Pour un fichier en lecture on dispose des fonctions : **seekg** et **tellg**

```
entree.seekg(0, ios::end);    // positionnement à la fin
entree.seekg(0, ios::beg);    // positionnement au début
entree.seekg(3, ios::cur);    // avance de 3
entree.seekg(-5, ios::cur);   // recule de 5
```

Remarque : L'usage combiné de seekg et tellg permet de déterminer facilement le nombre d'octets d'un fichier en lecture :

```
entree.seekg(0, ios::end);              // positionnement à la fin
cout << ifs.tellg() << " octets\n";     // nombre d'octets
```

Fermeture des fichiers

```
sortie.close();
entree.close();
```

Exemple : fonction qui recopie un fichier dans un autre

```
int copiefichier ( char *dest , char *source ) // copie le fichier source dans dest,  renvoie 1 si ok, 0 sinon
{
   ifstream fi (source);
   if (!fi) return 0;      //source impossible à lire
   ofstream fo ( dest);
   if (!fo) return 0;
   char tampon;
   while ( fo && fi.get (tampon) )
       fo.put(tampon);
   return fo.good() && fi.eof();                // en fin de fonction, on teste l'état des fichiers
}
```