5. Entrées et sorties

- La commande C++ '**cout**' (*character out*) affiche ce qui suit à l'écran, soit des chaînes de caractères (entre guillemets doubles à l'anglaise) de façon littérale, soit des valeurs de variables.
- Nécessite la bibliothèque iostream (`#include<iostream>`).
- `endl` (*end line*) fait aller à la ligne suivante.
- Tous les éléments sont séparés par des <<.

Exemple : `cout << "La valeur de i est : " << i << endl;`

Avec la bibiothèque **iomanip** (`#include<iomanip>`) on peut fixer le nombre de chiffres, `setprecision`, et la taille, `setw` (*set width*), dans `cout` :

```
cout << M_PI << endl;     // Affiche 3.14159
cout << setprecision(12);
cout << M_PI << endl;     // Affiche 3.14159265359
cout << 1 << setw(4) << 2 << " " << 3 << endl; // Affiche 1   2 3
```

La commande C++ '**cin**' (*character in*, nécessite `#include<iostream>`) permet de lire des caractères à partir du clavier. Exemple :

```
int a, b; cout << "Valeurs de a et b? "; cin >> a >> b;
```

- Le programme attend jusqu'à ce que l'utilisateur tape les deux valeurs, séparées par espace ou entrée et terminées par entrée.
- Remarque bien l'utilisation des >> pour `cin` (au lieu des << pour `cout`) et le `cout` pour expliquer à l'utilisateur ce qu'il doit faire.

Lire et écrire dans des fichiers

Pour lire ou écrire dans un fichier (méthode C++) on a besoin de la bibliothèque **fstream** (`#include<fstream>`). Puis il faut d'abord déclarer une variable du type `fstream` (*file stream*) et y affecter un fichier :

```
fstream fich;
fich.open("resultats.res", ios::in);
```

Cela ouvre le fichier "resultats.res" (qui doit exister) pour lire. En remplaçant `ios::in` par `ios::out` on ouvre le fichier pour écrire. Dans ce cas le fichier est créé (et un fichier préexistant du même nom sera écrasé). [On peut rajouter à la fin d'un fichier existant avec `ios::out|ios::app` (*input/output stream append*).]
Puis pour lire ou écrire dans le fichier la syntaxe est identique à celle de `cin` et `cout`, en remplaçant `cin` et `cout` par le nom de la variable fstream (ici `fich`).
Exemples :

- Si le fichier est ouvert pour lire (`ios::in`) et contient deux valeurs entières, soit sur une ligne (séparées par une ou plusieurs espaces), soit sur deux lignes : `int a, b; fich >> a >> b;`
- Si le fichier est ouvert pour écrire (`ios::out`) et on veut y écrire deux valeurs entières sur une ligne :
  `int a = 5, b = 2; fich << a << " " << b << endl;`

Enfin on ferme le fichier avec la commande `fich.close();`
(Après on peut éventuellement réutiliser `fich` pour ouvrir un autre fichier.)

Exemple d'un programme qui lit des coordonnées (trois doubles par ligne) dans un fichier dont le nombre de lignes est inconnu à l'avance :

```
#include<iostream>
#include<fstream>
using namespace std;
int main() {
  double x, y, z;
  int i = 0;
  fstream fich;
  fich.open("coordonnees.dat", ios::in);
  while(fich >> x >> y >> z) {
    cout << x << " " << y << " " << z << endl; // affichage des valeurs lues
    i++;     // comptage du nombre de lignes
  }
  cout << "Le fichier coordonnees.dat a " << i << " lignes" << endl;
  fich.close();
  return 0;
}
```

L'expression `fich >> x >> y >> z` non seulement lit les trois coordonnées d'une ligne du fichier, mais donne comme résultat "vrai" si trois valeurs ont en effet été lues et "faux" sinon. On peut donc l'utiliser comme condition de la `while`. Enfin elle déplace aussi l'indicateur de la ligne courante dans le fichier afin que la prochaine instruction de lecture dans le fichier lira la ligne suivante. L'ouverture d'un fichier place toujours cet indicateur à la première ligne.

Graphiques (en faisant appel au Python)

Les fonctions graphiques du C (et du C++) étant plus compliquées, on va faire appel au Python depuis le C pour tracer des courbes. Dans la bibliothèque des fonctions spécifiques au magistère on a créé une fonction `make_plot_py` pour faire cela. Elle s'utilise comme suivant :

```
#include<bibli_fonctions.h>  /* Bibliotheque du magistere qui contient
                                egalement les autres includes et
                                le using namespace std */

int main() {
  int i;                  // On cree d'abord un fichier avec donnees
  fstream fich;
  fich.open("donnees.dat", ios::out);
  for (i = 0; i < 100; i++)
    fich << i/10. << " " << sin(i/10.) << endl;
  fich.close();
  ostringstream pyth;  // A partir d'ici on fait appel a Python
  pyth                 // Dans pyth on ecrit toutes les commandes Python
    << "A = loadtxt('donnees.dat')\n"
    << "x = A[:,0]\n"  // N'oublie pas le \n apres chaque ligne
    << "y = A[:,1]\n"
    << "plot(x,y)\n"
   ;
  make_plot_py(pyth); // Et on envoie ces commandes vers Python qui les execute
  return 0;           // Voir la notice pour d'autres commandes Python
}
```