

Chapitre 3

Entrées/Sorties en C++

Les entrées et sorties sont gérées dans C++ à travers des objets particuliers appelés **streams** ou bien **flots (ou flux)**.

Pour utiliser ces objets, il faut inclure :

```
#include <iostream>
```

1. E/S dans le langage C

Un rappel des entrées et sorties en C:

```
printf ("%d", 5, 5); // affichage de 5 uniquement.
```

```
printf ("%d, %d", 5); // affichage de 5 et un entier quelconque à cause du 2nd %d qui n'a pas été spécifié.
```

`printf` est peu sécuritaire car le programmeur doit spécifier le type.

2. E/S dans le langage C++

<iostream> offre une interface O.O. plus sécuritaire.

Deux opérateurs sont surchargés de manière appropriée pour les flots:

- l'opérateur d'insertion << (écriture)
- l'opérateur d'extraction >> (lecture)

Les flots prédéfinis sont:

- cout associé à la sortie standard (équivalent à stdout dans le langage C),
- cerr associé à la sortie erreur standard (équivalent à stderr dans le langage C),
- cin associé à l'entrée standard (équivalent à stdin dans le langage C),
(Je passe sur le 4^e flot clog).

Le bon format (bonne opération) est choisi en fonction du type des arguments,

© Mohamed N. Lokbani

v3.00

POO avec C++

Chapitre 3 : Entrées/Sorties en C++

Les types écrits ou lus sont:

char short int long float double char*

Soit le programme exemple.cpp suivant:

```
#include <iostream>
```

```
int main() {  
    int i;  
    std::cout << "Entrer un entier: " << std::endl; // endl équivalent de \n en C  
    std::cin >> i;  
    std::cout << "Le carre de " << i << " est: " << (i*i) << std::endl;  
    return 0;  
}
```

Si exemple est l'exécutable de exemple.cpp, vous obtenez comme résultat:

```
prompt>exemple  
Entrer un entier:  
20 (retour chariot obligatoire)  
Le carre de 20 est: 400  
prompt>
```

© Mohamed N. Lokbani

v3.00

POO avec C++

Un autre exemple ... et on s'évite par la même occasion les indirections des ::, qui portent le nom de **opérateur de résolution de portée**, en précisant l'espace de nom d'où seront prises les différentes fonctions d'entrées et sorties. Nous allons aborder plus en détails cet opérateur au chapitre suivant.

```
#include <iostream>

using namespace std;

int main() {
    int n;
    double x;

    // on peut écrire aussi le \n à l'intérieur de la chaîne à afficher
    cout << "Entrer la valeur de n, puis celle de x:\n";
    cin >> n >> x;

    // on incrémente n de 3 unités.
    n+=3;

    // à cause des règles de préséance des opérateurs, on pouvait s'abstenir d'écrire
    // (x*x) entre parenthèses, x*x suffit.
    cout << n << (x*x) << endl;
    return 0;
}
```

Nous obtenons le résultat suivant :

```
prompt>exemple
Entrer la valeur de n, puis celle de x:
10(espace obligatoire)12(retour chariot obligatoire)
13144
prompt>
```

Il n'y a pas eu de séparation entre 13 et 144, car l'espace n'a pas été demandé. Pour l'avoir, il fallait écrire:

```
cout << n << " " << (x*x) << endl;
```

3. Mise en forme de la sortie

Pour mettre en forme les sorties, on utilise des manipulateurs de sortie => un manipulateur de sortie ne produit pas de sortie mais spécifie un paramètre de mise en page qui remplace le paramètre de défaut.

```
#include <iostream.h>
#include <iomanip.h>

int main() {
    int n=123;

    cout << setw(6) << n <<n<<endl;
    //affiche :::123123 où : représente dans cet exemple l'espace blanc

    cout << setw(2) << n <<endl;
    //ignore le format et affiche 123 car n contient 3 caractères et le format tient uniquement sur 2 caractères.

    cout << setfill('*') << setw(6) << n <<endl;
    //affiche ***123 remplace l'espace blanc : par des *

    cout << hex << n << endl;
    //affichage dans la base hexadécimale de n: 7B
    cout << oct << n << endl;
    //affichage dans la base octale de n: 173
    return 0;
}
```