Chapitre 3

Entrée/Sorties en C++

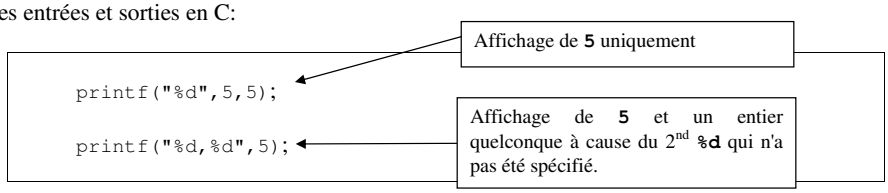
- Les entrées et sorties sont gérées dans C++ à travers des objets particuliers appelés **streams** ou bien **flots (ou flux)**.

- Pour utiliser ces objets, il faut inclure :

```
#include <iostream>
```

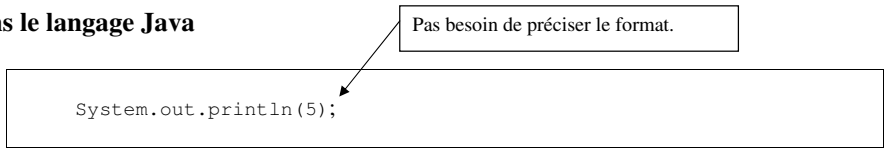
1. E/S dans le langage C

Un rappel des entrées et sorties en C:



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

2. E/S dans le langage Java



- C'est une bonne chose de ne pas être obligé de préciser le format de sortie.
- Java se charge d'afficher dans le format qui conviendrait le mieux.
- Un inconvénient à cette approche! Et si l'on voulait avoir une sortie formatée?
- C'est possible en Java mais assez complexe à faire.
- Dans la version 5 de Java, l'utilisation du formatage a été grandement facilitée.
- En effet, le formatage à la manière C a été introduit dans la version 5 de java.

3. 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)

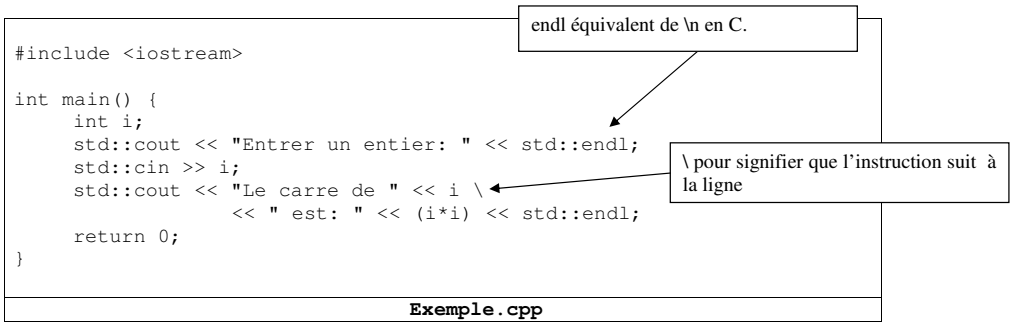
- Ces deux opérateurs sont « surchargés » car on leur fait faire autre chose.
- En réalité, ces opérateurs sont utilisés dans des opérations de manipulation de bits.

- 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),

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

- Les types écrits ou lus sont:

```
char short int long float double char*
```



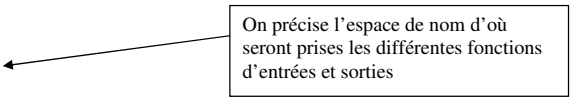

```
#include <iostream>

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

Exemple.cpp

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>
```


- On veut éviter l'utilisation des indirections « :: », qui portent le nom de **opérateur de résolution de portée**.




```
#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 suffirait.
     cout << n << (x*x) << endl;
     return 0;
}
```

Exemple.cpp

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;
```

- On pouvait utiliser aussi « " » à la place de « ' ».

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


4. 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.
- Ce paramètre remplace le paramètre par défaut.