université de TOURS

Développement objet C++

Dominique H. Li

dominique.li@univ-tours.fr

Licence Informatique - Blois

Université de Tours

université de TOURS

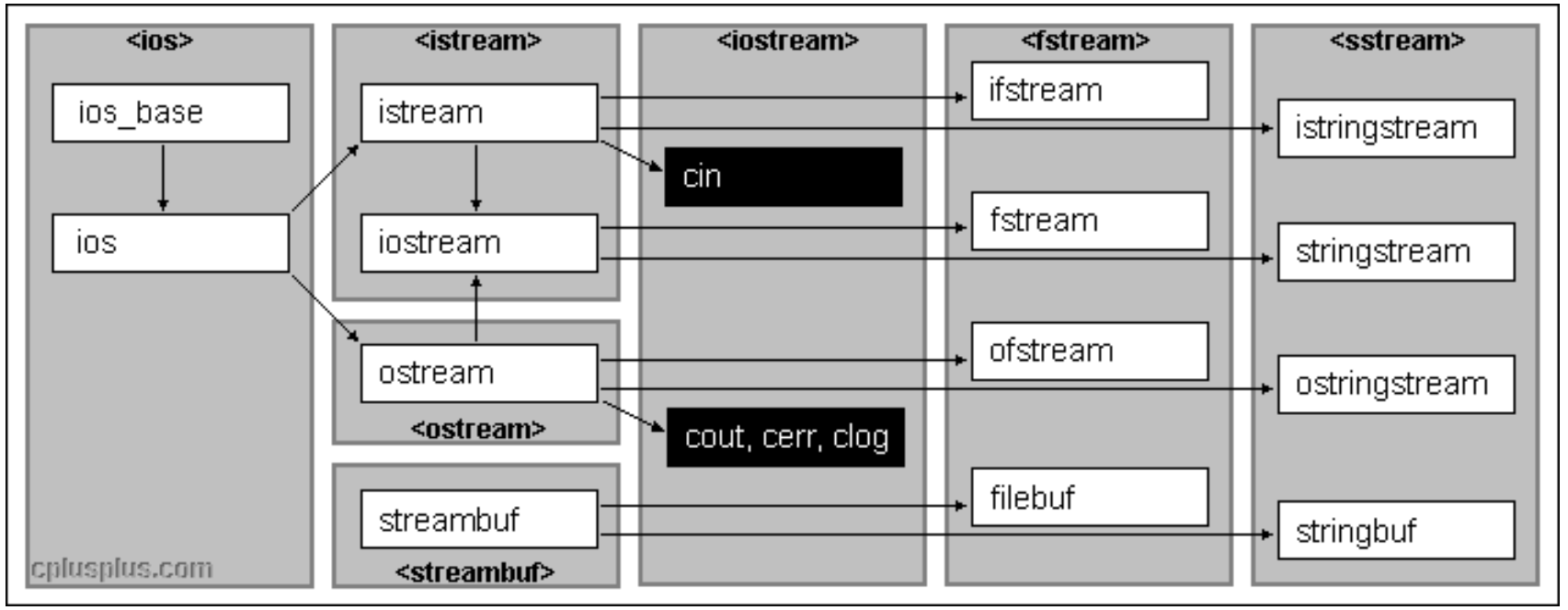
```cpp
#include <iostream>

int main() {
    std::cout << "Hello, World !" << std::endl;
    return 0;
}
```

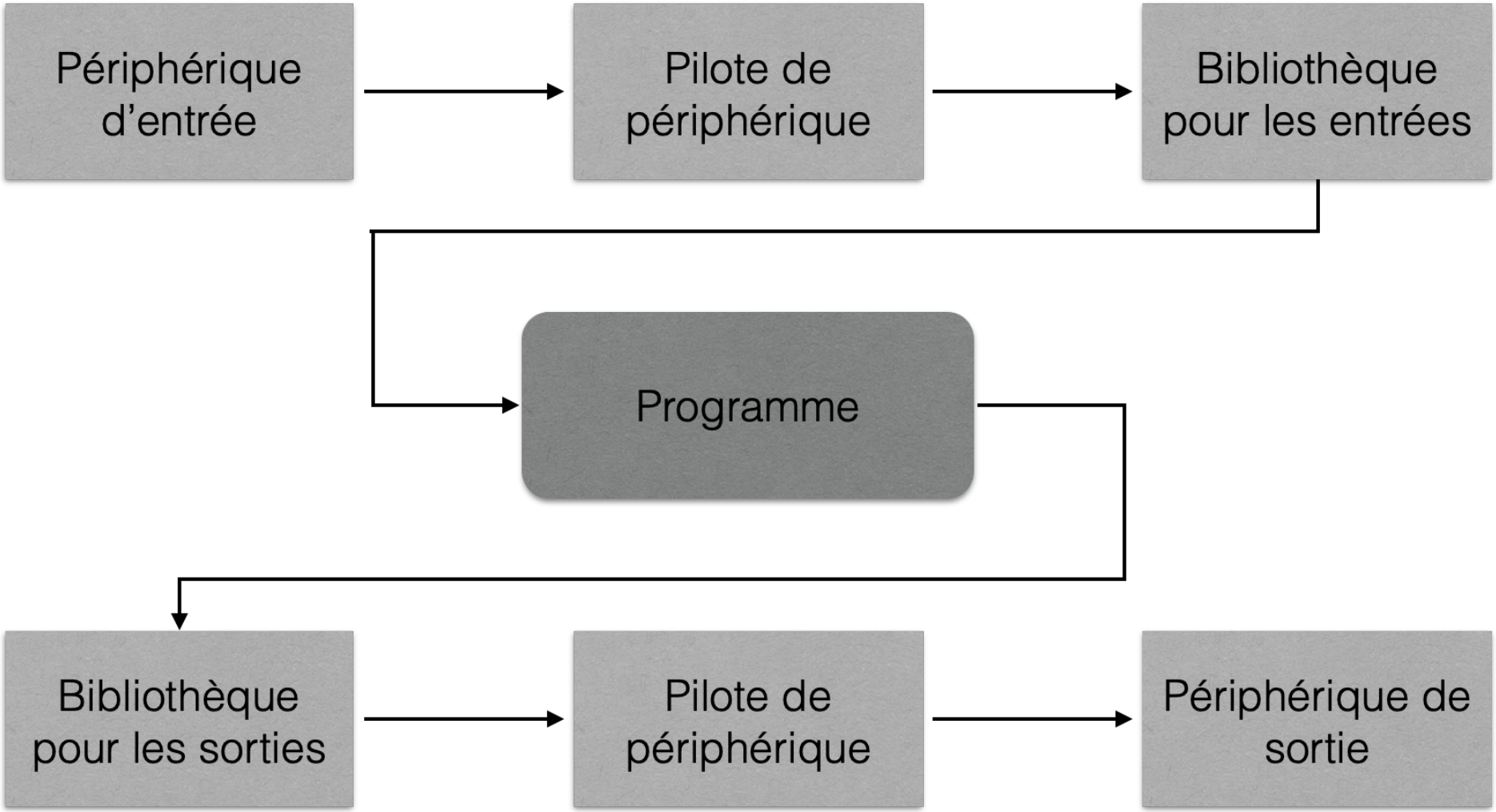
Les flux

Développement objet C++

La bibliothèque d'entrées/sorties



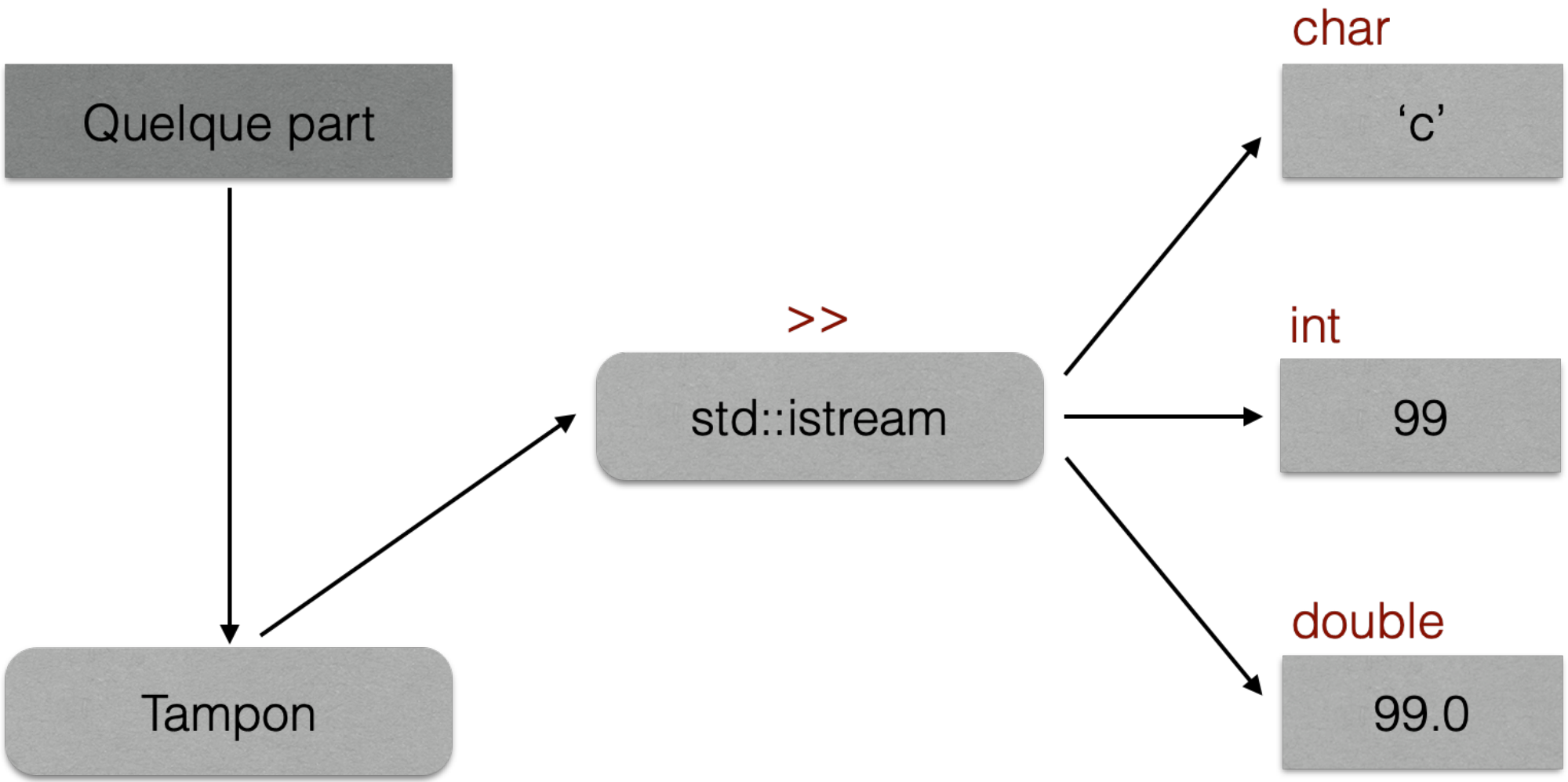
Flux d'entrée et de sortie



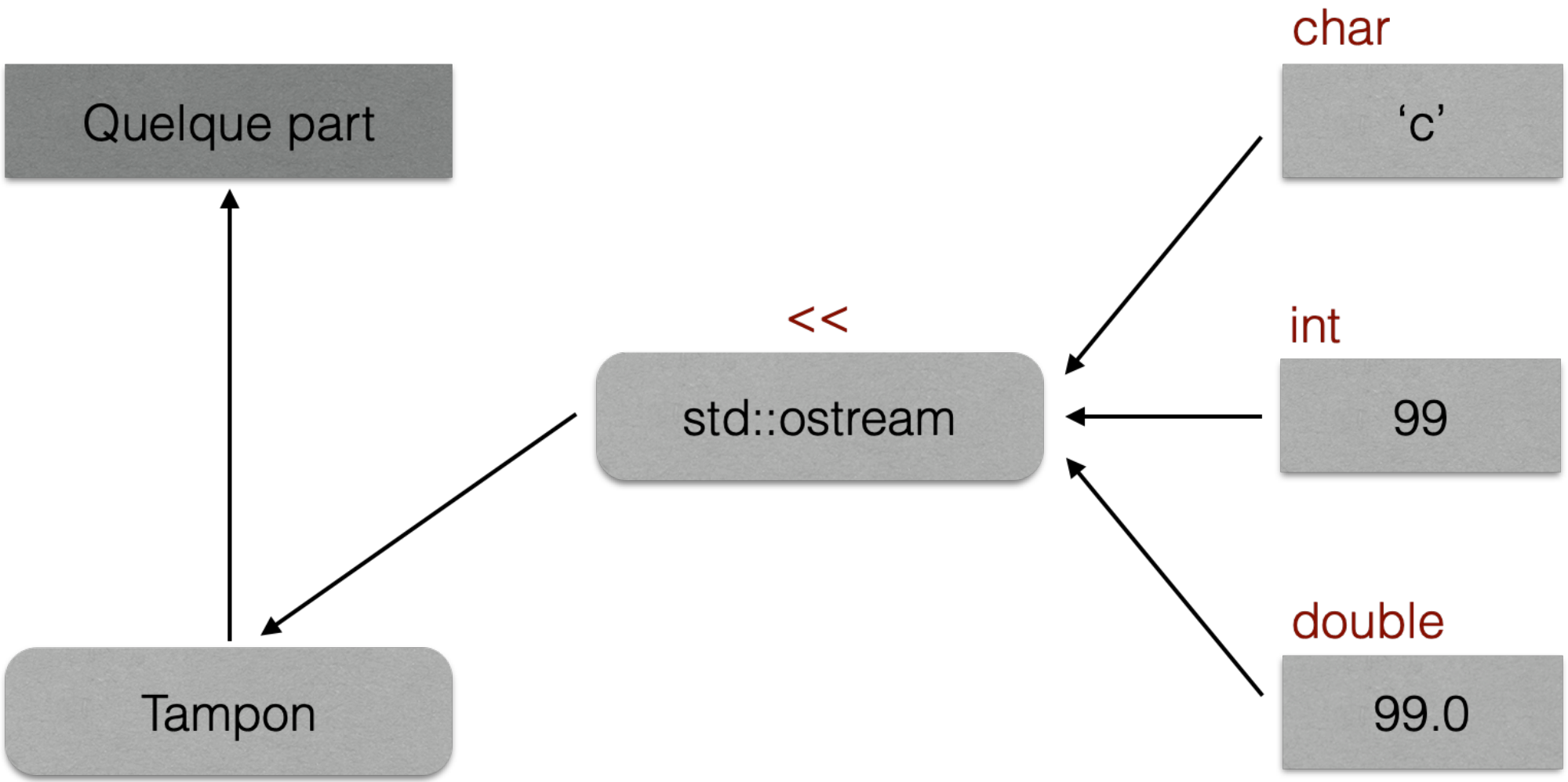
Les modèles des flux d'entrées/sorties

- La bibliothèque standard de C++ dispose de deux types **std::istream** et **std::ostream** pour gérer les flux d'entrée et de sortie
  - **std::istream** permet d'obtenir des suites de caractères de quelque part puis de les transformer en valeurs de différents types
  - **std::ostream** permet de transformer des valeurs de différents types en suites de caractères puis de les envoyer à quelque part
- Les opérateurs **>>** (entrée) et **<<** (sortie) permettent de lire et de écrire dans les flux

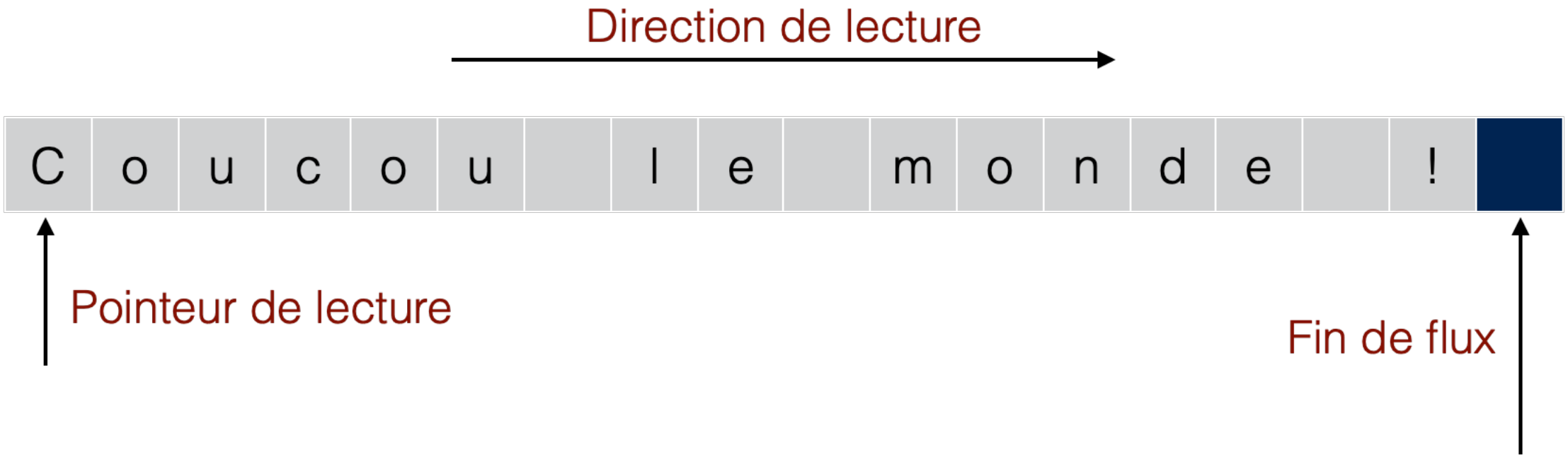
Représentation du flux std::istream



Représentation du flux std::ostream



Les flux d'entrée



Les manipulations des flux d'entrée (1)

- Lire une valeur typée

  istream& operator>>(…);

- Lire un seul ou plusieurs caractères

  int istream::get();
  istream &istream::get(char &c);
  istream &istream::get(char *s, streamsize n);

- Lire une ligne

  istream &istream::getline(char *s, streamsize n);

Les manipulations des flux d'entrée (2)

- Ignorer un certain nombre de caractères

  istream &istream::ignore(streamsize n, int delim);

- Regarder (pré-lire) le caractère suivant

  int istream::peek();

- Lire un bloc de données (caractères)

  istream &istream::read(char *s, streamsize n);

Les manipulations des flux d'entrée (3)

- Positionner le pointeur de lecture

  istream &istream::seekg(streampos pos);

- Récupérer la position du pointeur de lecture

  streampos istream::tellg();

- Reculer le pointeur de lecture

  istream &istream::unget();

Exemple : utilisation du flux d'entrée (1)

```
espace = 0;

while (cin.good()) {
        char c = cin.get();
        if (c == ' ') {
                ++espace;
        }
}
```

Exemple : utilisation du flux d'entrée (2)

```
char s1[256];
std::cout << "Avec espaces : ";
std::cin.sync();
std::cin.getline(s1, 256);
std::cout << "Avec espaces : " << s1 << std::endl;

std::string s2;
std::cout << "Avec espaces : ";
std::cin.sync();
std::getline(std::cin, s2);
std::cout << "Avec espaces : " << s2 << std::endl;
```

Exemple : utilisation du flux d'entrée (3)

```
char c;
int n;
std::string mot;

std::cout << "Entrer un mot ou un nombre : ";
c = std::cin.peek();

if (c >= '0' && c <= '9') {
        std::cin >> n;
        std::cout << "Le nombre est " << n << std::endl;
} else {
        std::cin >> mot;
        std::cout << "Le mot est " << mot << std::endl;
}
```

Exemple : utilisation du flux d'entrée (4)

```
in.seekg(0, std::ios::end);                    // Soit in un flux de std::istream
int taille = in.tellg();
char *tmp= new char[taille];
std::cin.seekg(0, std::ios::beg);
std::cin.read(tmp, taille);
std::cout << tmp << std::endl;
delete[] tmp;
```

Les manipulations des flux de sortie (1)

- Écrire un seul caractère

  ostream &ostream::put(char c);

- Écrire un bloc de données (caractères)

  ostream &ostream::write(const char *s, streamsize n);

- Récupérer la position du pointeur d'écriture

  streampos ostream::tellp();

- Positionner le pointeur d'écriture

  ostream &ostream::seekp(streampos pos);

Les manipulateurs du flux de sortie (2)

- Formats d'écriture d'entiers
  - **dec** — écriture d'un entier en décimal (base 10)
  - **oct** — écriture d'un entier en octal (base 8)
  - **hex** — écriture d'un entier en hexadécimal (base 16)
- Formats d'écriture de réels
  - **setprecision(int n)** — afficher un réel avec au plus **n** chiffres
  - **fixed** — fixer le nombre de chiffres après le point

Les manipulateurs du flux de sortie (3)

- Formats d'écriture de caractères
  - **setw(int n)** — afficher au moins **n** caractères
  - **setfill(char c)** — remplir par le caractère **c** en cas de nécessité

Exemple : utilisation du flux de sortie

```
using namespace std;

int n = 30;
cout << dec << n << endl << oct << n << endl << hex << n << endl;

double pi = 3.14159;
cout << setprecision(5) << pi << endl;                    // 3.1416
cout << setprecision(9) << pi << endl;                    // 3.14159
cout << fixed;
cout << setprecision(5) << pi << endl;                    // 3.14159
cout << setprecision(9) << pi << endl;                    // 3.141590000

cout << setw(10) << 88 << endl;
cout << setfill('0') << setw(8) << 88 << endl;
cout << setfill('A') << setw(5) << 88 << endl;
```

Les fichiers

- Au niveau le plus élémentaire, un *fichier* n'est rien d'autre qu'une simple suite d'octets numérotés à partir de **0**
- Un fichier possède un format, autrement dit un ensemble de règles qui déterminent la signification des octets
  - Dans un fichier texte, les quatre premiers octets seront les quatre premiers caractères
  - Dans un fichier représentant des entiers sous forme binaire, ces même quatre premiers octets seront interprétés comme le premier entier

Les flux de fichier

- La bibliothèque standard de C++ dispose deux flux de fichier
  - Le type **std::ifstream** permet de lire un flux d'octets sur disque et compose des objets à partir de ceux-ci
  - Le type **std::ofstream** permet de convertir les objets de la mémoire en flux d'octets et les écrit sur disque
- Les types **std::ifstream** et **std::ofstream** héritent des types **std::istream** et **std::ostream**

Les manipulations de fichiers

- Pour lire un fichier, il faut
  - connaître son nom puis l'ouvrir
  - y lire les caractères
  - le fermer (généralement fait implicitement)
- Pour écrire un fichier, il faut
  - le nommer puis l'ouvrir ou créer un nouveau fichier de ce nom
  - y écrire des objets
  - le fermer (généralement fait implicitement)

Exemple : lire dans un fichier

```
#include <fstream>
#include <string>
using namespace std;

int main() {
        string s;
        cin >> s;
        ifstream in(s.c_str());                          // Ouvrir un fichier
        string ville = "";
        double temperature = 0.0;
        while (in >> ville >> temperature) {             // Lire 2 entités
                cout << ville << " : " << temperature << endl;
        }
        return 0;
}
```

Exemple : écrire dans un fichier

```
#include <fstream>
#include <string>
using namespace std;

int main() {
        string s;
        cin >> s;
        ofstream out(s.c_str());                    // Ouvrir un fichier
        string ville = "";
        double temperature = 0.0;
        while (cin >> ville >> temperature) {
                out << ville << temperature << endl;        // Écrire 2 entités

        }
        return 0;
}
```

Gestion des erreurs d'entrées/sorties

- États des flux
  - **good()** — les opérateurs ont réussi
  - **eof()** — on a atteint la fin de l'entrée
  - **fail()** — quelque chose d'inattendu s'est produit
- Code exemple

```
while (in >> tampon) {

}
if (in.fail())
std::cerr << "une erreur E/S s'est produite" << std::endl;
```

Les manipulations de fichiers textes

```
std::ifstream ifs;
std::ofstream ofs;

ifs.open("tab.txt");
ofs.open("esp.txt");

char c;
while (ifs.good()) {
        ifs.get(c);
        if (c == '\t')
                ofs.write("    ", 4); // Remplacer une TAB par 4 espaces
        else
                ofs.put(c);
}
```

Conversion de chaînes de caractères

- Les conversions entre les chaînes de caractères et les nombres sont faciles en C++
- Le flux **std::stringstream** (déclaré dans l'en-tête **<sstream>**) permet les conversions entre les chaînes et les nombres

Exemple : chaîne de caractères vers nombre

```
std::string s("3.14159");
double pi;
std::stringstream ss(s);
ss >> pi;                          // pi == 3.14159
```

Exemple : nombre vers chaîne de caractères

```
double pi = 3.14159;

std::stringstream ss;

ss << pi;

std::string s = ss.str();          // s == "3.14159"
```