2e année – Spécialité Informatique

Année 2024 – 2025

Programmation orientée objet en C++

Notes de cours

Sébastien Fourey

Version du 3 septembre 2024

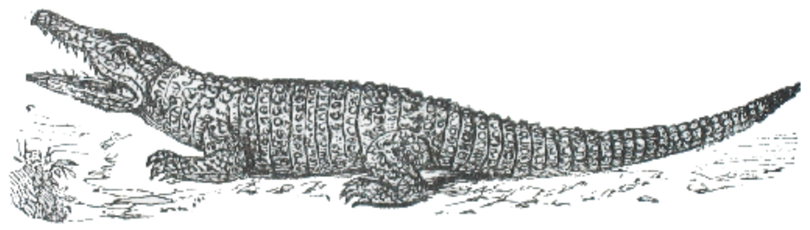

Fig. 145. — Crocodile. Il y en a qui atteignent 8 mètres de long.

Chapitre 4

La bibliothèque standard

4.1 Flux d'entrée/sortie

4.1.1 Hiérarchie des flux d'E/S en C++

En C++ les entrées/sorties sont implémentées par une hiérarchie relativement complexe de classes dont deux au moins sont connues du débutant : les classes `istream` et `ostream` dont les deux instances respectives `cin` et `cout` sont déclarées dans le fichier d'en-tête `<iostream>`.

La hiérarchie de ces classes est donnée par la figure 4.1 dans laquelle on trouve :

- Les classes les plus générales regroupant les propriétés communes à n'importe quel flux (`ios_base` et `ios`, cf. sections 4.1.2 et 4.1.3) ;
- Les classes spécialisées des flux d'entrée (`istream`) et des flux de sortie (`ostream`) ;
- La classe spécialisée des flux d'entrée ou sortie (`iostream`) ;
- Les classes spécialisées des flux de fichiers (`ifstream`, `ofstream` et `fstream`) ;
- Les classes spécialisées des flux de chaînes (`istringstream`, `ostringstream` et `stringstream`).

Le lecteur aura compris que la dernière catégorie de la liste précédente permet de « mimer » avec une syntaxe C++ les manipulations possibles avec les fonctions `sprintf` et `sscanf` de la bibliothèque C.

En fait, la bibliothèque standard C++ ne définit pas directement les classes d'entrée/sorties. Ces dernières ne sont effet que des instanciations de modèles de classes standards. Le paramètre des modèles de flux est le type de *caractère* utilisé. Les modèles standards qui correspondent aux classes d'entrées sorties sont représentés dans la figure 4.2.

Exercice 4.1 *Quelle relation peut on faire entre les classes* `ostream` *et* `istream` *que vous connaissez déjà, associées à la surcharge possible des opérateurs de flux* `<<` *et* `>>` *; et les méthodes* `toString()` *du langage Java ?*

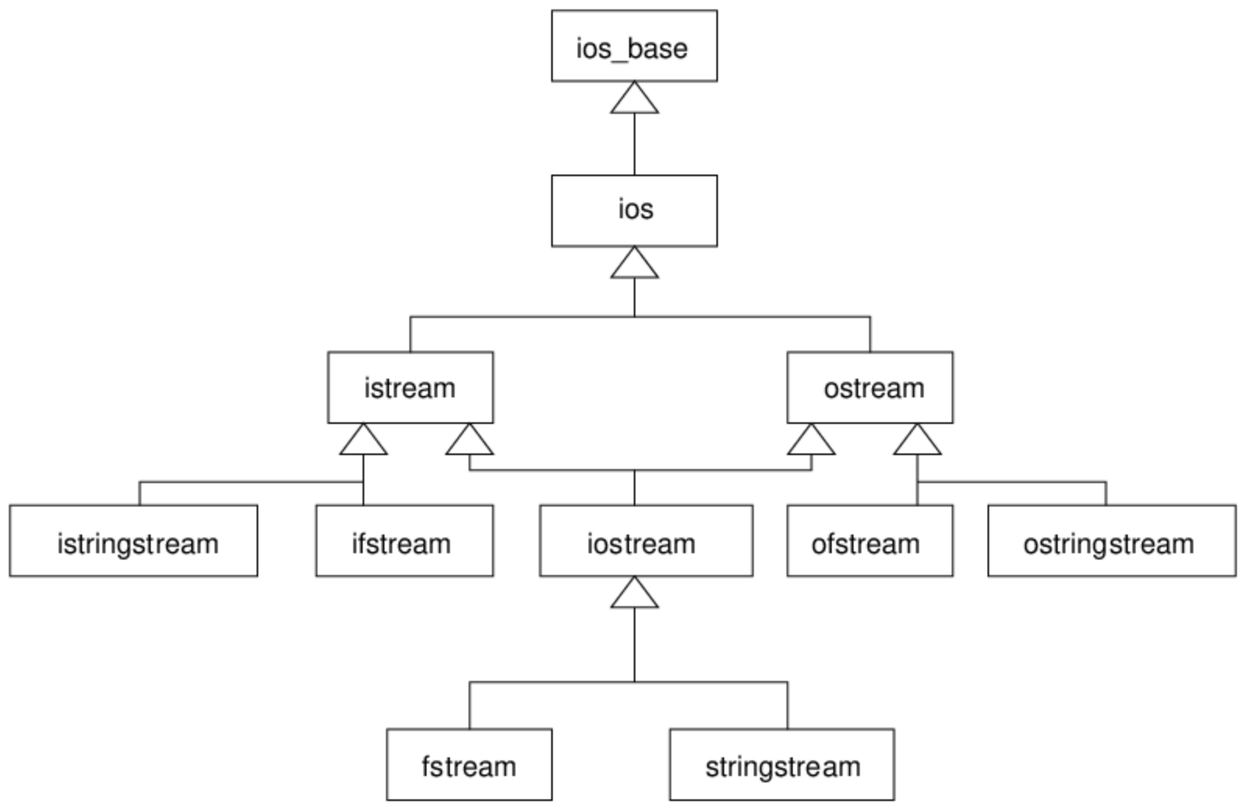

FIGURE 4.1 – Les classes de flux d'E/S en C++.

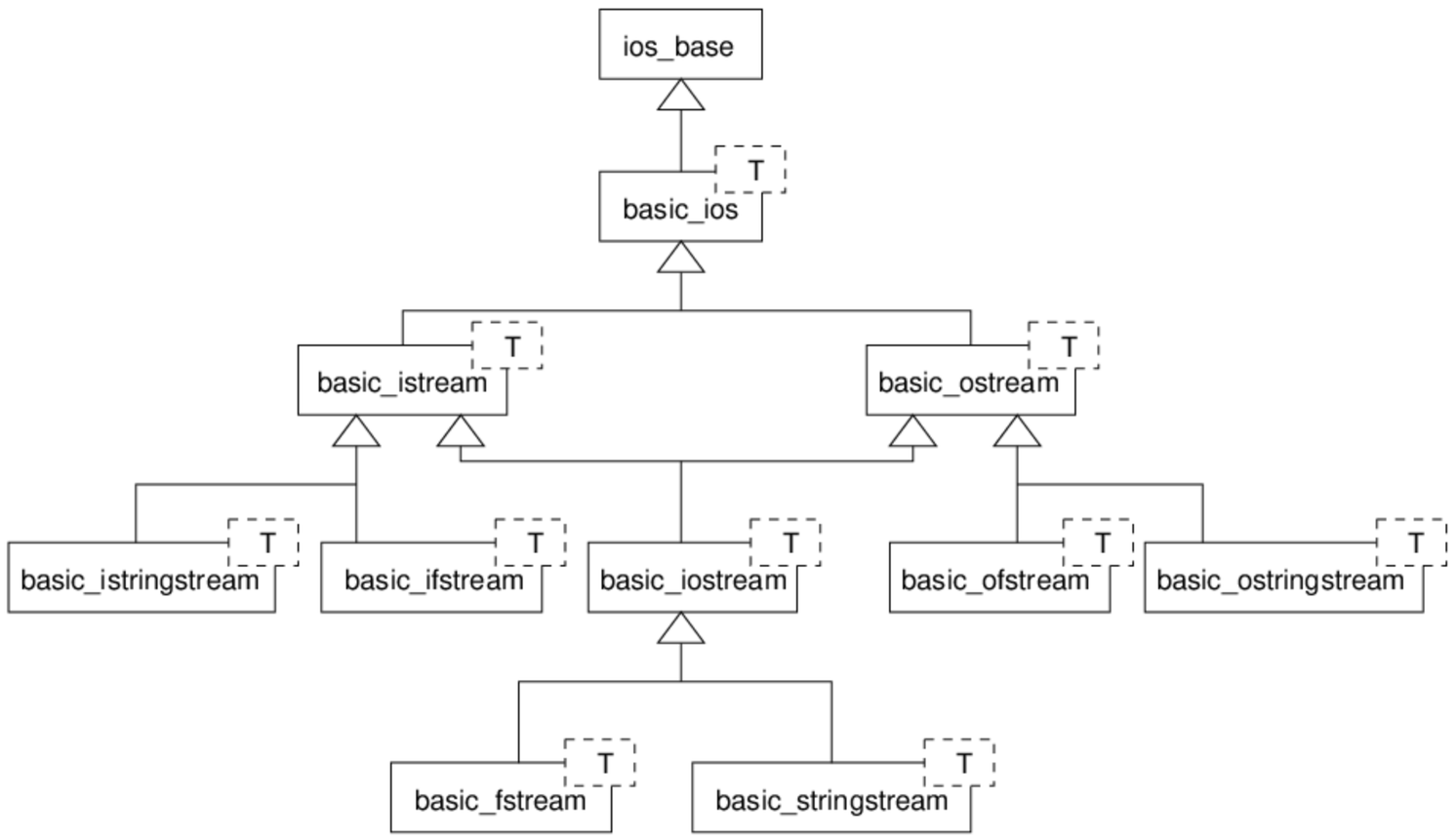

FIGURE 4.2 – Les modèles de classes de flux en C++.

4.1.2 La classe `ios_base`

Cette classe mère de toutes les classes de flux est déclarée dans l'en-tête standard `<ios>`. Elle définit un grand nombre de constantes (cf. tableau 4.1) et quelques méthodes (cf. tableau 4.2) qui sont utilisées dans toutes les classes dérivées.

4.1.3 La classe `ios`

Elle fournit les services de gestion de tampons aux classes dérivées (grâce à un objet de la classe `streambuf`) ainsi qu'un ensemble de méthodes permettant de connaître l'état d'un flux. Les méthodes usuelles de cette classe sont données dans le tableau 4.3.

Format (`ios::fmtflags`)

boolalpha	left	uppercase
hex	right	initbuf
oct	internal	skipws
dec	showbase	adjustfield
fixed	showpoint	basefield
scientific	showpos	floatfield

Modes d'ouvertures (`ios::openmode`)

in	out	binary
trunc	app	ate

Positionnement (`ios::seekdir`)

beg	cur	end

État du flux (`ios::iostate`)

eofbit goodbit badbit failbit

TABLE 4.1 – Constantes définies par la classe `ios_base`. (Leurs types respectifs.)

Formats

```
  fmtflags  flags() const;
  fmtflags  flags(fmtflags);
  fmtflags  setf(fmtflags);
  fmtflags  setf(fmtflags f, fmtflags mask);
      void  unsetf(fmtflags);
streamsize  precision() const;
streamsize  precision(streamsize);
streamsize  width() const;
streamsize  width(streamsize);
```

TABLE 4.2 – Méthodes usuelles définies par la classe `ios_base`.

État

```
            operator void*() const;
   bool     operator!() const;
iostate     rdstate() const;
   void     clear(iostate status = goodbit);
   void     setstate(iostate status);
   bool     eof();
   bool     fail();
   bool     good();
   bool     bad();
```

Remplissage

```
char_type   fill();
char_type   fill(char_type);
```

Synchronisation

```
 ostream*   tie(ostream *);
```

TABLE 4.3 – Méthodes usuelles définies par la classe `ios`

4.1.4 La classe `ostream`

Elle est déclarée dans l'en-tête standard `<ostream>` et définit comme son nom l'indique l'ensemble des méthodes propres aux flux de sortie. Les méthodes essentielles sont rappelées dans le tableau 4.4.

Constructeur

`ostream(streambuf*);`

Sortie et position

`ostream&  operator<<(T );`
`ostream&  put(char_type);`
`ostream&  write(const char_type *, streamsize);`
`ostream&  flush();`
`pos_type  tellp();`
`ostream&  seekp(pos_type p);`
`ostream&  seekp(off_type p, ios_base::seekdir d);`

Gestion des manipulateurs

`ostream&  operator<<(ostream& (*pf)(ostream &));`
`ostream&  operator<<(ios& (*pf)(ios &));`
`ostream&  operator<<(ios_base& (*pf)(ios_base &));`

TABLE 4.4 – Les principales méthodes de la classe `ostream`.

Des manipulateurs (cf. tableau 4.5) sont déclarés dans l'en-tête `<iomanip>`. Grâce à une surcharge de l'opérateur `<<`, ils permettent d'agir sur les flux de sortie « à la volée », c.-à-d. avec la même syntaxe que celle qui est utilisée pour envoyer des données dans un flux.

Exercice 4.2 *Combien de fonctions ou méthodes sont appelées dans l'instruction ci-dessous ?*

```
std::cout << "Hello world" << std::endl;
```

Manipulateurs simples

`endl`	`hex`	`showbase`
`ends`	`oct`	`noshowbase`
`flush`	`dec`	`showpoint`
`boolalpha`	`fixed`	`noshowpoint`
`noboolalpha`	`scientific`	`showpos`
`left`	`uppercase`	`noshowpos`
`right`	`nouppercase`	`unitbuf`
`internal`		`nounitbuf`

Manipulateurs avec paramètre

`setw(int)`	`setbase(int)`	`setfill(char_type)`
`setprecision(int)`		
`resetiosflags(ios_base::formatflags)`		
`setiosflags(ios_base::formatflags)`		

TABLE 4.5 – Les manipulateurs utilisables avec les `ostream`.

4.1.5 La classe `istream`

C'est l'analogue de la classe `ostream` pour les flux d'entrée. Elle est déclarée dans l'en-tête standard `<istream>`. Ses principales méthodes sont données dans le tableau 4.6.

Constructeur

`istream(streambuf *);`

Entrée et position

`istream&`	`operator>>(T &);` T : type de base
`int_type`	`get();`
`istream&`	`get(char_type &);`
`int_type`	`peek();`
`istream&`	`put_back(char_type);`
`istream&`	`unget();`
`istream&`	`read(char_type *, streamsize);`
`streamsize`	`readsome(char_type *, streamsize);`
`istream&`	`get(char_type *, streamsize);`
`istream&`	`get(char_type *, streamsize, char_type delm);`
`istream&`	`getline(char_type *, streamsize);`
`istream&`	`getline(char_type *, streamsize, char_type delm);`
`istream&`	`ignore(streamsize n, int_type d=traits::eof);`
`int`	`sync();`
`pos_type`	`tellg();`
`streamsize`	`gcount() const;`
`istream&`	`seekg(pos_type p);`
`istream&`	`seekg(off_type p, ios_base::seekdir d);`

TABLE 4.6 – Les principales méthodes de la classe `istream`.

Les manipulateurs suivants peuvent agir sur un `istream` :

```
hex oct dec boolalpha noboolalpha skipws noskipws ws
```

La gestion de ces manipulateurs passe par la définition des méthodes suivantes dans la classe `istream`.

```
istream&  operator>>( istream& (*pf)(istream &) );
istream&  operator>>( ios& (*pf)(ios &) );
istream&  operator>>( ios_base& (*pf)(ios_base &) );
```

4.1.6 Fichiers

Le fichier d'en-tête `<fstream>` fournit les déclarations des classes `fstream`, `ifstream` et `ofstream`. Quelques méthodes propres aux fichiers (ouverture à partir d'un nom, fermeture, état) sont données dans le tableau 4.7.

Attention aux flux de fichiers passés comme arguments de fonctions. [. . .]

fstream

```
fstream();
fstream(const char*, ios_base::openmode m=ios_base::out|ios_base::in);
void  open(const char *p, ios_base::openmode m=ios_base::in|ios_base::out);
bool  is_open() const;
void  close();
```

ofstream

```
      ofstream();
      ofstream(const char *p, ios_base::openmode m=ios_base::out);
void  open(const char *p, ios_base::openmode m=ios_base::out);
bool  is_open() const;
void  close();
```

ifstream

```
      ifstream();
      ifstream(const char *p, ios_base::openmode m=ios_base::in);
void  open(const char *p, ios_base::openmode m=ios_base::in);
bool  is_open() const;
void  close();
```

TABLE 4.7 – Les principales méthodes des classes `*fstream`.

4.1.7 Les flux chaînes de caractères

Ils sont définis dans l'en-tête standard `<sstream>`. Il s'utilisent bien entendu comme des flux génériques, et les fonctions utiles qui permettent les conversions avec des chaînes standard sont données dans le tableau 4.8.

stringstream

```
          stringstream(ios_base::openmode m=out|in);
explicit  stringstream(string &s, openmode m=out|in);
  string  str() const;
    void  str(const string &) const;
```

ostringstream

```
          ostringstream();
explicit  ostringstream(string &);
```

istringstream

```
          istringstream();
explicit  istringstream(string &);
```

TABLE 4.8 – Les principales méthodes des classes `*stringstream`.

4.2 La classe `string` des chaînes de caractères

Elle est définie dans le fichier d'en-tête `<string>` et possède les méthodes listées dans le tableau 4.9.

Quelques méthodes

```
              string();
              string(const char *);
              string(const char *, size_type);
const char&   at(size_type);
      char&   at(size_type);
const char&   operator[](size_type);
      char&   operator[](size_type);
    string&   operator=(const char *);
      bool    empty() const;
    size_t    length() const;
    size_t    size() const;
const char*   c_str() const;
  size_type   copy(char*, size_type, size_pos p=0) const;
       int    compare(char *) const;
       int    compare(string &) const;
  string&     operator+=(const string &);
  string&     operator+=(const char *);
  string&     insert(size_type, const string &);
  string&     insert(size_type, const char *);
size_type     find(const string &, size_type i=0);
size_type     find(const char *, size_type i=0);
   string     substr(size_type i, size_type n);
```

Fonctions et operateurs

```
      bool    operator==(const string &, const string &);
      bool    operator==(const string &, const char &);
      bool    operator==(const char *, const string &);
    string    operator+(const string &, const string &);
```

TABLE 4.9 – Les principales méthodes de la classe **string** et quelques fonctions associées.