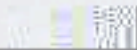


LE LANGAGE C++ MASTER 1 LA STL

Jean-Baptiste.Yunes@univ-paris-diderot.fr
U.F.R. d'Informatique
Université Paris Diderot - Paris 7



nov. 2012

- La STL : une bibliothèque de modèles de classes et fonctions
 - conteneurs
 - algorithmes et objets fonction
- itérateurs et allocateurs
- chaînes
- flux



LES CONTENEURS

- : tableaux dynamiques 1D
 - : listes doublement chaînées
 - : files à double accès
 - : files simples
 - : piles
 - : tableaux associatifs
 - : ensembles
- 
- : ensemble de booléens ou de bits

- : c'est un du ;

- il contient des types

- (type utilisé à l'instanciation du)

- : qui se comporte comme un pointeur

- + beaucoup d'autres...

- : c'est un du
- il contient des fonctions membres permettant d'obtenir des itérateurs
 - : pointe sur le 1er élément
 - : pointe sur le suivant du dernier
 - : pointe sur le dernier
 - : pointe sur le précédent du premier

-  : c'est un du

- il contient des fonctions et opérateurs d'accès aux éléments

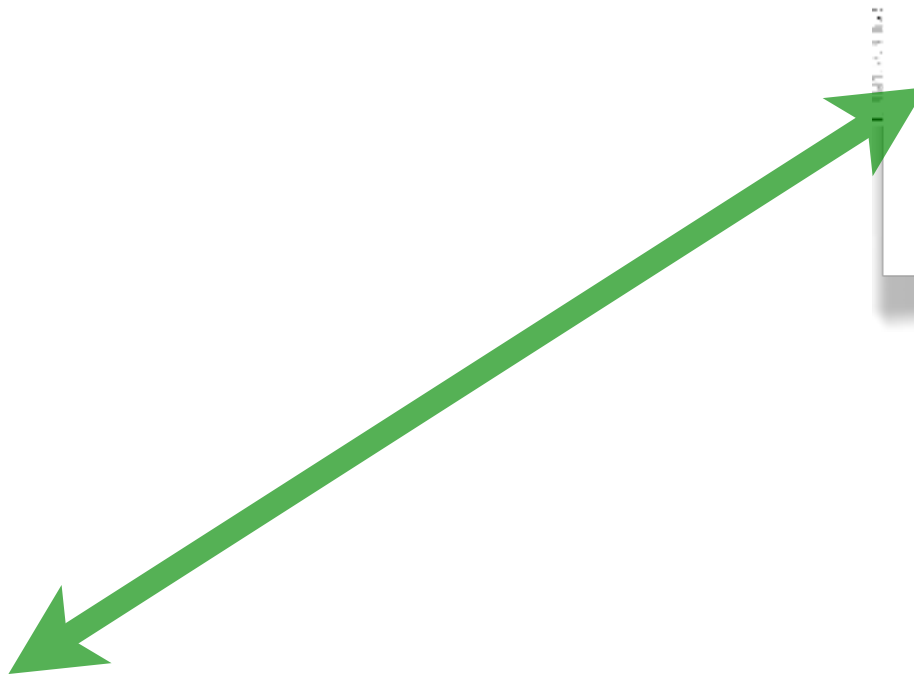
- : accès non contrôlé

- : accès contrôlé ()

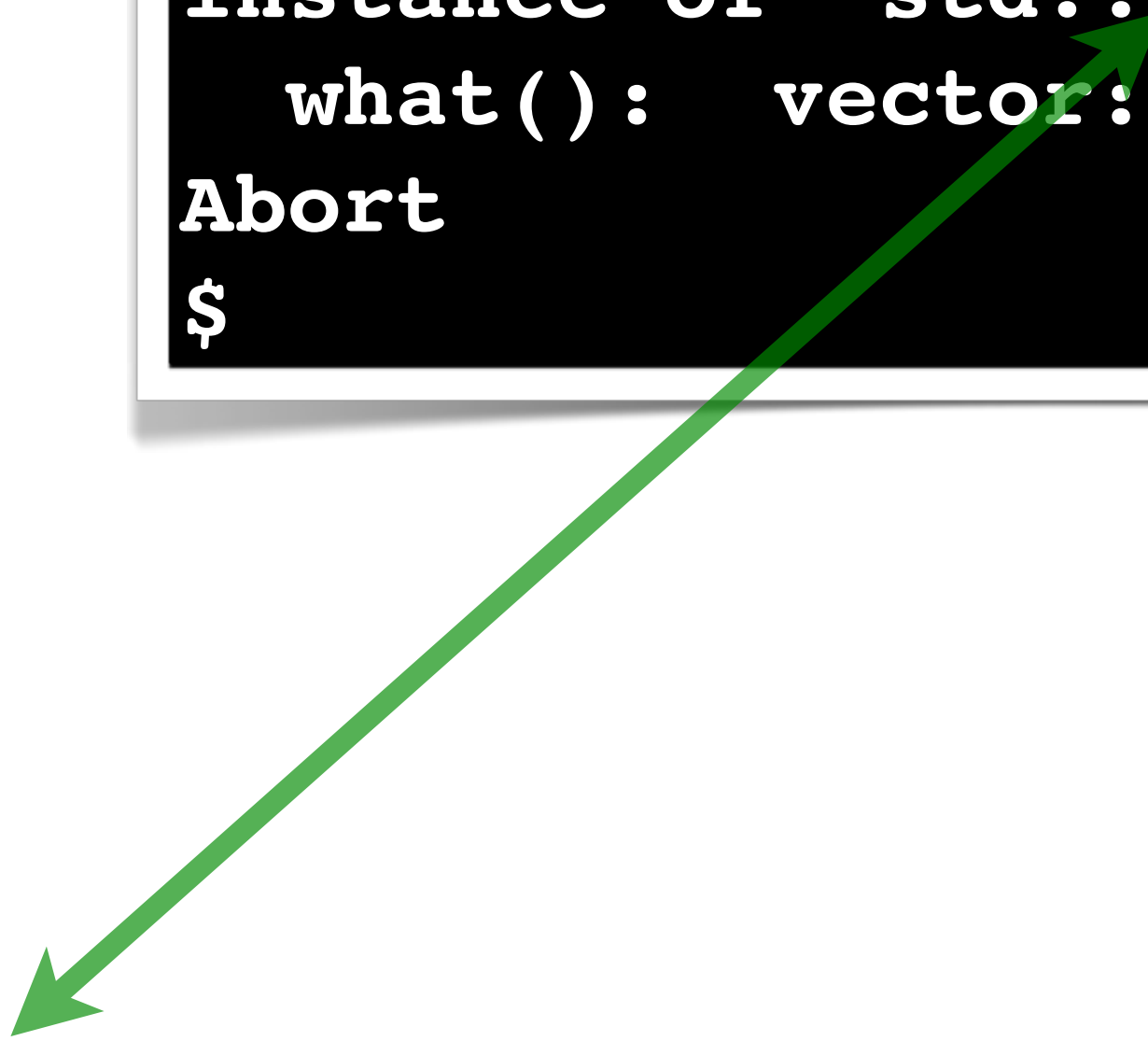
- : premier élément

- : dernier élément

```
$ ./test  
un,deux  
Segmentation fault  
$
```




```
$ ./test  
un,deux  
terminate called after throwing an  
instance of 'std::out_of_range'  
    what():  vector::_M_range_check  
Abort  
$
```



• `vector<T>` : c'est un conteneur du type `T`

• il contient des constructeurs


`vector<T>(n, val)`

• crée un vecteur de taille *n*, les éléments tous initialisés à *val*

`vector<T>(i1, i2)`

• crée un vecteur avec les éléments *i1* *i2*

• crée un vecteur par copie

-  : c'est un du
- il contient des opérateurs/fonctions d'affectation

- `pop()` : c'est un `void` du `Stack` et qui peut-être vu comme une pile
 - il contient des fonctions de manipulation de pile
- `peek()` : attention `peek()` ne renvoie pas la valeur!!! Il faut consulter la valeur avant de dépiler (`pop()`).

```
$ ./test  
quatre  
trois  
deux  
un  
$
```

on peut utiliser les
itérateurs de flux (in a
while crocodile)

see you later
alligator

- `list` : c'est un `list` du `list` et qui peut-être vu comme une liste
 - il contient des fonctions de manipulation de liste
 - `insert` *position* *element* ajoute l'*élément* avant la *position*
 - `insert_n` *n* *element* ajoute *n* copies de l'*élément*
 - `delete` *position* supprime l'élément à la *position* indiquée
 - `delete_range` *p1* *p2* efface le séquence *p1* *p2*
 - `clear` efface tout

- : c'est un du
- il contient des attributs de taille
 - : nombre d'éléments
 - : est-il vide ?
 - : + grande taille possible
- redimensionne avec initialisation des nouveaux
- : capacité actuelle avant redimensionnement automatique
- n : réserve de la place pour n éléments sans les initialiser

- : c'est un du
- il contient des fonctionnalités annexes
 - : pour échanger deux vecteurs
 - : ordre lexicographique

- À quoi sert, par exemple, ???
 - permet d'écrire des templates indépendants des types inclus dans d'autres types!



Voilà un bon usage de ce mot-clé

LES SÉQUENCES

- les séquences sont des structures ordonnées, on y trouve :
 - : plutôt un tableau
 - : optimisé pour l'insertion/suppression mais pas d'indexage
 - : optimisée pour les parcours dans les deux sens
- et les dérivées

LES CONTENEURS ASSOCIATIFS



- est une simple séquence de paires (clé,valeur)
- les clés doivent être comparables (opérateur `<`)
- sinon c'est
- les types inclus sont :

- est une simple séquence de paires
(clé,valeur)

Un type défini par
le template!

```
$ ./test  
trouve  
$
```

- est une réduite aux seules clés

ALGORITHMES ET OBJETS FONCTION

LES OBJETS FONCTION

LES FONCTEURS



- En surchargeant l'opérateur fonctionnel on a déjà vu comment obtenir des objets qui pouvaient être considérés comme des fonctions
 - ce sont des foncteurs ou objets fonction
- La STL en fait une utilisation intensive en liaison avec les algorithmes

- La STL définit quelques grandes classes de foncteurs :

- les fonctions unaires :

- les fonctions binaires

```
$ ./test  
5  
$
```

- La STL définit quelques foncteurs dont
 - les prédicats qui sont simplement des foncteurs renvoyant un booléen

- is_less / is_less_equal

- is_greater / is_greater_equal / is_equal

- is_not_less / is_not_greater

```
$ ./test  
1  
0  
$
```

- La STL définit quelques foncteurs dont

- les fonction arithmétiques

- \cdot /

- \cdot /

- On trouve aussi les liaisons/adaptateurs/inverseurs

LES ALGORITHMES



- Les algorithmes qui ne modifient pas les séquences

```
$ ./test  
012  
$
```

- Bien que ces algorithmes soient classés dans ceux qui ne modifient pas la séquence, il est à noter que cela n'interdit pas de modifier les éléments de la séquence...
- Mais on devrait utiliser

```
$ ./test
```

```
012
```

```
123
```

```
$
```

```
$ ./test  
c'est nul  
$
```

- Les algorithmes qui modifient les séquences

- /
- /
- / / /
- /
- /
- / remove_if() /
- /
- /
- /

• Les tris

•

•

•

/

•

•

/

•

•

•

/

•

```
$ ./test  
c'est nul  
abracadabra  
bool  
abracadabra  
bool  
c'est nul  
$
```

version avec comparateur
implicite : operateur <

```
$ ./test  
c'est nul  
abracadabra  
bool  
c'est nul  
bool  
abracadabra  
$
```

version avec fonction
(ou binary-function)

- Les opérations ensemblistes

- Les comparaisons

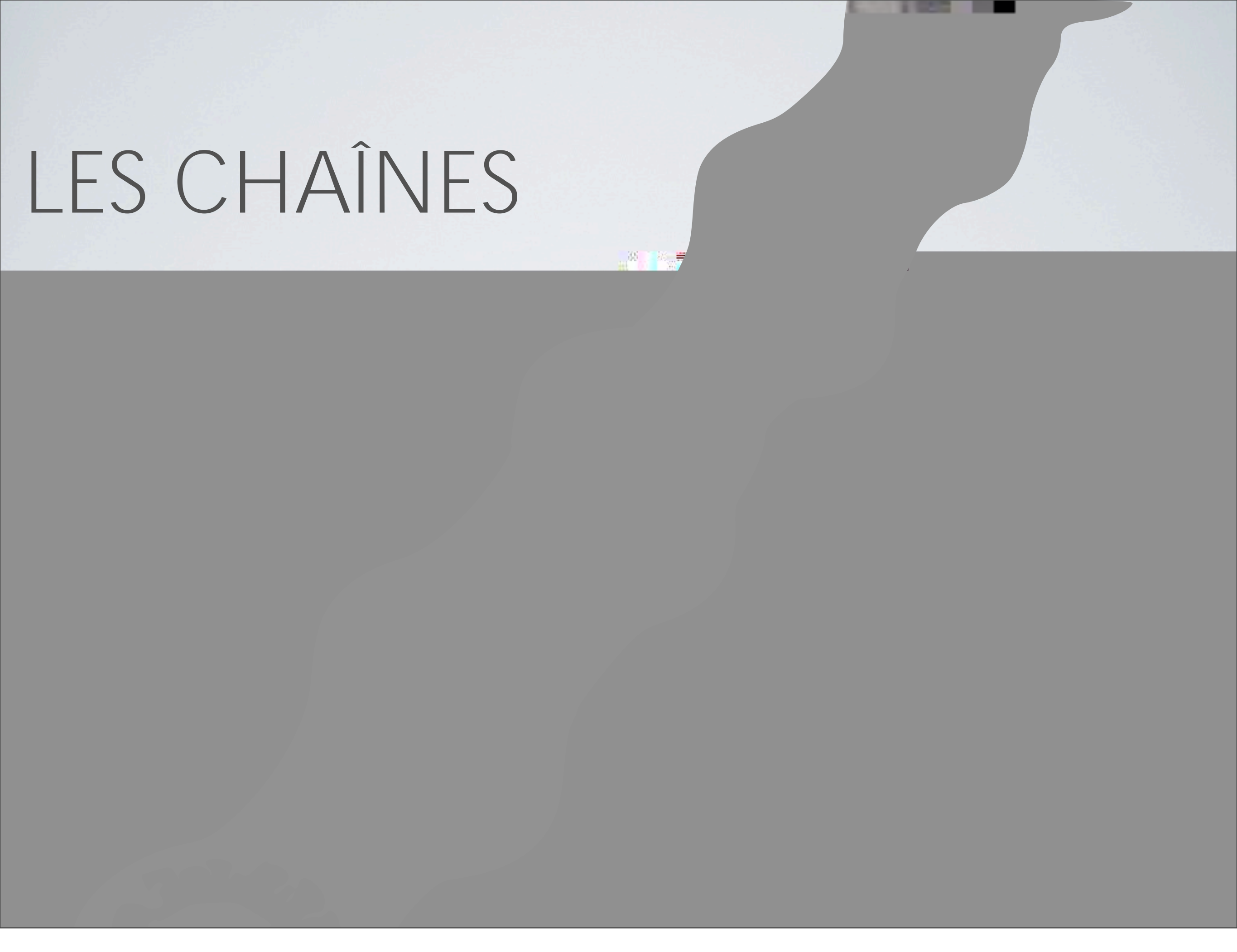
- Les permutations

- Les tas

ITÉRATEURS ET ALLOCATEURS



- Nous n'en dirons que le strict que minimum
- Les itérateurs permettent de se déplacer dans une séquence
 - de manière similaire à l'arithmétique des pointeurs
- Les allocateurs correspondent aux classes qui permettent de réaliser des allocations
 - il s'agit d'abstractions permettant de détacher les constructions de la STL des contingences matérielles



LES CHÂÎNES

- `<string>` est une simple séquence de caractères...
 - elle est toutefois différente d'une séquence ordinaire
 - car certaines opérations désirées sont particulières
 - Noter que les méthodes disponibles sont bien trop nombreuses pour être exposées ici
 - mais aucune n'est particulièrement surprenante, donc
- consultez la documentation!

LES FLUX

- Les deux classes de bases sont
 - , et sa sous-classe
- Toutes deux ne sont pas directement instanciables mais contiennent essentiellement les attributs et actions de base de la gestion des flux :
 - état du flux
 - propriétés de formatage du flux

- Les fonctions membres les plus importantes sont :

- `isok` : tout est ok
- `isfatal` : le flux est dans un état irrécupérable
- `iserror` : une erreur est apparue, consulter `errno` pour connaître sa gravité
- `isend` : la fin du flux a été atteinte
- `reset` : on remet l'état à une valeur par défaut
- `sync_with_stdio(bool b=true) // statique`

- Sous la classe on trouve deux classes :
 - la classe de base de tous les flux de lecture
 - la classe de base de tous les flux d'écriture

• pour

on trouve • pour

on trouve

- Sous la classe on trouve

- Sous la classe on trouve

- Sous la classe on trouve

- Ces différentes classes se distinguent essentiellement par la source (pour les `InputStream`s) ou par la destination (pour les `OutputStream`s) de l'opération :

- `FileInputStream` / `FileOutputStream` : fichier

- `ByteArrayInputStream` / `ByteArrayOutputStream` : mémoire

- donc par leur constructeur...

- Quelques exemples :



- Quelques exemples :



- boolalpha / noboolalpha
- dec / hex / oct
- endl / ends
- fixed / scientific
- flush
- internal
- left / right
- setbase
- setfill
- setiosflags / resetiosflags
- setprecision
- setw
- showbase / noshowbase
- showpoint / noshowpoint
- showpos / noshowpos
- skipws / noskipws
- unitbuf / nouunitbuf
- uppercase / nouppercase
- ws

```
$ ./test
truec_____1a85
$
```

LES ITÉRATEURS DE FLUX



- Il existe deux itérateurs de flux :
- ils servent à itérer une opération d'entrée ou de sortie sur une séquence...
- il s'agit d'afficher chaque élément et d'utiliser un séparateur


```
$ ./test
```

```
0:1:2:
```

```
$
```