

Complément de C++ ISIMA2

Laurent B. Garcia
David R.C. Hill

Introduction à la librairie standard
du C++

Evolution de la programmation

Le but de la manipulation est d'écrire un programme qui affichera "HELLO

BASIC au Lycée

```
10 PRINT "HELLO WORLD"
```

```
20 END
```

En PREPA, DEUG, BTS ou DUT

Mon bon Blaise 😊

```
program HELLO(input, output)
    begin
        writeln('HELLO WORLD')
    end.
```

En 1ère année d'Ecole d'Ingénieur...

LISP, Scheme et Cie...

```
(defun HELLO  
  (print  
    (cons 'HELLO (list 'WORLD))  
  )  
).
```

ZZ1 Étudiant expérimenté...

C la vie...

```
#include <stdio.h>

int main(int argc, char ** argv)
{
    char *message[] = {"HELLO ", "WORLD"};
    int i;
    for(i = 0; i < 2; ++i)
        printf("%s", message[i]);
    printf("\n");
}
```

C++ en deuxième année

Étudiant très expérimenté

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

class string
{
    private:
        int size;
        char *ptr;

    public:
        string() : size(0), ptr(new char('\0')) {}
        string(const string &s) : size(s.size)
        {
            ptr = new char[size + 1];
            strcpy(ptr, s.ptr);
        }
        ~string()
        {
            delete [] ptr;
        }

        friend ostream &operator <<(ostream &, const string &);
        string &operator=(const char *);
};
```

La suite...

```
ostream &operator<<(ostream &stream, const string &s)
{
    return(stream << s.ptr);
}

string & string::operator=(const char *chrs)
{
    if (this != &chrs)
    {
        delete [] ptr;
        size = strlen(chrs);
        ptr = new char[size + 1];
        strcpy(ptr, chrs);
    }
    return(*this);
}
```

// et enfin...

```
int main(int, char **)
{
    string str;

    str = "HELLO WORLD";
    cout << str << endl;

    return(0);
}
```

Genèse de la STL

Demande de la communauté C++

- Pas de bibliothèque de classes conteneur

 - Chacun développe sa bibliothèque dans son coin

 - Réutilisabilité nulle

 - Apprentissage nécessaire à chaque fois

 - Fiabilité douteuse

- Pas de classe chaîne de caractères

 - Mêmes conséquences

L'existant

- Bibliothèque ADA

- Non orientée objet

Résultat : Librairie Standard du C++ ex STL

Contenu :

Classes

- string

- conteneurs

 - de base (vector, deque, list)

 - spécialisés (stack, queue, priority_queue)

 - associatifs (set, map)

- iostream revu et corrigé

- utilitaires

fonctions

- algorithmes qui travaillent sur les conteneurs

- génériques grâce à la notion d'itérateur

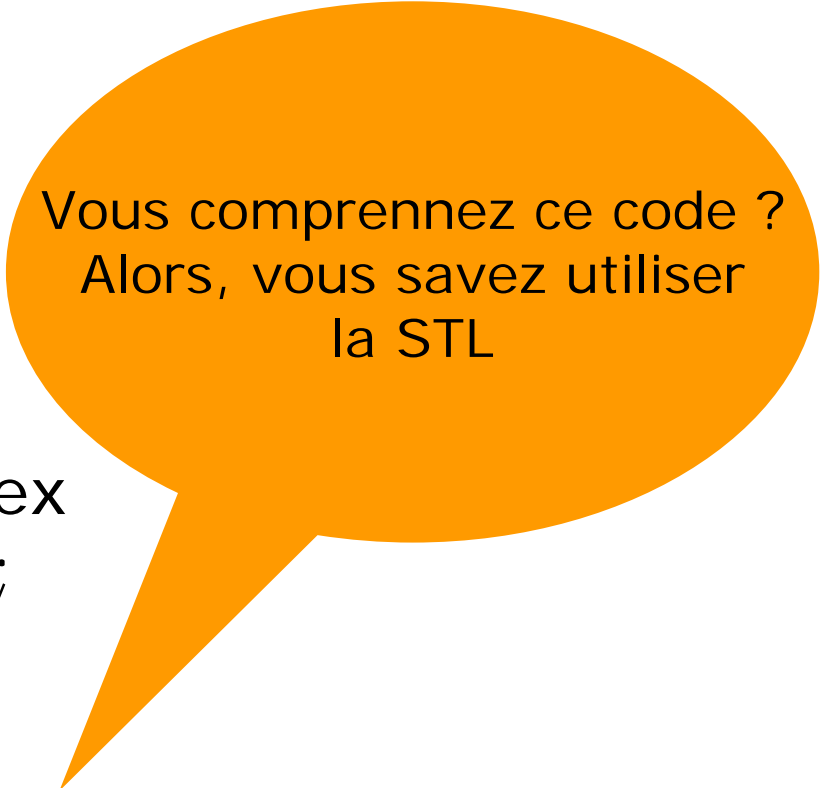
Notion d'itérateur

Définition :

Balise localisant un emplacement dans une collection

Vous en utilisez déjà : pointeurs dans tableau

```
int tableau[10];  
int niveau=10;  
int *courant=tableau;  
int *fin=tableau+niveau;  
while (courant != fin)  
{  
    // action sur *courant, ex  
    cout << *courant << endl;  
    ++courant;  
}
```



Vous comprenez ce code ?
Alors, vous savez utiliser
la STL

Notion d'itérateur BIS

Tout itérateur est isomorphe à un pointeur dans un tableau

Même code avec STL :

```
#include <vector>
using namespace std;
typedef vector<int> VecInt;

VecInt v(10);
VecInt::iterator courant=v.begin();
VecInt::iterator fin=v.end();

while (courant != fin)
{
    cout << *courant << endl;
    ++ courant;
}
```

Dissection du code précédent

```
#include <vector>
using namespace std;
```

Notion de namespace

Elimine les collisions de noms de classes

Ex 2 classes Matrice dans 2 bibliothèques

Avec namespace : préfixe de nommage

Ex : stats::Matrice ou pde::Matrice

Suppression du préfixe :

Classe/fonction isolée: using stats::Matrice;

Tout l'espace : using namespace stats;

Particularité : pas de .h dans les headers
de la librairie standard du C++

Dissection du code (le retour)

```
typedef vector<int> VecInt;
```

Généricité : tous les types sont template

Les itérateurs sont en sous classe des
conteneurs

Souvent :

```
typedef VectIntIt VecInt::iterator;
```

Les classes fondamentales

La classe string

Encore une classe template ... sur le type char de base !

Supporte toutes les opérations de base avec les opérateurs classiques

Conversion vers et depuis char *

Les conteneurs fondamentaux

Trois classes de base

vector

Modélise un vecteur à croissance dynamique

Operations en bout de vecteur et acces direct en $O(1)$ amorti

deque

Liste spécialisée dans les opérations aux 2 bouts

Acces direct en $O(\log(n))$ aux 2 bouts en $O(1)$

list

Liste doublement chaîne circulaire classique

Toute insertion / deletion en $O(1)$

Accès directe en $O(n)$

Les spécialisées !

Basées sur une collection de base mais
avec opérations spécifiques

Collections :

stack

objet pile (pop, push et top)

queue

objet file (pop, push, front et back)

priority_queue

file à priorité, implémentée sous la forme d'un
tas minimax

pop, push, top

Les conteneurs associatifs

Utilisent une clef (paire, valeur)

Deux types et 2 catégories :

set (clef et valeur confondues)

map (clef et valeur distinctes)

set et map : une seule valeur par clef

multiset et multimap : plusieurs valeurs autorisées par clef

Méthodes les plus courantes

Méthode

Action

`iterator begin()`

Premier élément du vecteur

`iterator end()`

Après le dernier élément du vecteur

`int size()`

Nb d'éléments présents dans le vecteur

`int capacity()`

Capacité d'accueil actuelle du vecteur

`void push_back(const T& elem)`

Ajoute un élément au bout du vecteur

`void pop_back()`

Retire l'élément au bout du vecteur

`void push_front(const T& elem)`

Ajoute un élément au début du vecteur

`void pop_front()`

Retire l'élément de début du vecteur

`const T& front()`

Ref sur l'élément en tête de vecteur

`const T& back()`

Ref sur l'élément en fin de vecteur

`empty()`

True si vecteur vide

`T& operator[](int idx)`

Ref sur l'élément d'index idx

`const T& operator[] (int i) const`

Idem mais en version constante

Opérations avec itérateurs

Méthode

Action

`void erase (i terator i t)`

Supprime l'élément spécifié

`void erase (i terator debut,
i terator fi n)`

Supprime les éléments [debut, fin[

`void insert(i terator pl ace,
const T& el em)`

Insère `elem` à l'emplacement `place`

`void insert(i terator pl ace,
i terator debut,
i terator fi n)`

Insère à la position `place`, les éléments de [debut, fin[

Les algorithmes

Ensemble d'opérations communes

Copie d'éléments entre conteneurs

Ecrasement

Insertion

Transfert d'éléments entre conteneurs

Recherche d'éléments

Fonctions plutôt que méthodes

STL initialement non orientée objet

Utilise abondamment les itérateurs

Algorithmes courants

Copie d'éléments

```
copy (debut, fin, destination);
```

Attention ! remplace les éléments !

Pour ajouter des éléments :

```
copy(debut, fin, inserter(destination));
```

Recherche d'éléments

```
place = find (debut, fin, element);
```

Affichage d'une collection

```
ostream_iterator<Type> oi (cout, " ");
```

```
copy(debut, fin, oi);
```

Aller plus loin avec les bibliothèques C++

Boost C++ Libraries - Mozilla Firefox

Fichier Edition Affichage Aller à Marque-pages Outils ?

http://www.boost.org/ OK

Google Traduction ScienceDirect ResearchIndex Conf. Journaux SYSTRAN ISIMA News Tout-Savoir.Net - Aide ...

...one of the most highly regarded and expertly designed C++ library projects in the world."
— Herb Sutter and Andrei Alexandrescu, C++ Coding Standards



Welcome to Boost.org!

Boost provides free peer-reviewed portable C++ source libraries.

We emphasize libraries that work well with the C++ Standard Library. Boost libraries are intended to be widely useful, and usable across a broad spectrum of applications. The [Boost license](#) encourages both commercial and non-commercial use.

We aim to establish "existing practice" and provide reference implementations so that Boost libraries are suitable for eventual standardization. Ten Boost libraries are already included in the C++ Standards Committee's Library Technical Report (TR1) as a step toward becoming part of a future C++ Standard. More Boost libraries are proposed for the upcoming TR2.

Getting started: Boost works on almost any modern operating system, including UNIX and Windows variants. Follow the [Getting Started Guide](#) to download and install Boost. Popular Linux and Unix distributions such as [Fedora](#), [Debian](#), and [NetBSD](#) include pre-built Boost packages. Boost may also already be available on your organization's internal web server.

Background: The [Background Information page](#) has introductory material to help those educating their organization about Boost.

Participation

Although Boost was begun by members of the C++ Standards Committee Library Working Group, participation has expanded to include thousands of programmers from the C++ community at large.

If you are interested in participating in Boost, please join our main [developers mailing list](#). Discussions are highly technical, and list members are encouraged to participate in formal reviews of proposed libraries. There is also a [users mailing list](#), and several [project specific lists](#).

Libraries

- Documentation
- License
- Download
- Getting Started

Regression Tests

- General Info
- Release
- Development

Search Boost

Boost 

Google™ Powered

Groups

- Boost (Developers)
- Boost Users
- Announcements
- Boost General Interest
- Project-Specific
- Discussion Policy

Contribute

- Formal Reviews
- Review Schedule
- Requirements
- Guidelines
- Submissions

Support

- FAQ
- Request Support
- Report Bugs
- Suggest Features
- Commercial Support
- Mailing Lists
- Version History

Other Resources

- Sandbox Files
- Files (Members)
- Main CVS
- Sandbox CVS
- Tools
- People
- Who's Using Boost?
- Moderators
- More Boost
- More C++

Terminé

Démarrer Inbox - Mo... 4 Microso... Résultats d... Poste de tr... Microsoft P... Boost C+... FR « 15:29