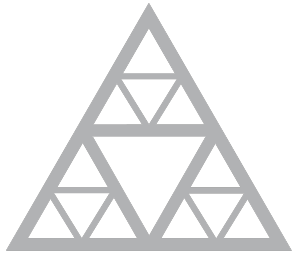
Introduction à la programmation en C++

Vecteurs - Templates

Nicolas Audebert

Vendredi 8 décembre 2017



École des Ponts
ParisTech

Avant toute chose

Rendus de TP et des exercices

Les rendus se font sur **Educnet**, même en cas de retard. **Pas par mail**.

1. Le code rendu **doit compiler**.
2. Le code rendu doit **être propre** (indentation, noms de variables clairs).
3. Le code rendu doit **être commenté** (réponses aux questions, fonctionnement du code).
4. Rassembler le code dans une seule archive (`.zip`, `.rar`, `.tar.gz`, etc.).

Un exercice ou un TP rendu en retard ou ne respectant pas une des consignes ci-dessus sera pénalisé.

Plan de la séance

STL : vector

Standard Template Library

- Bibliothèque de fonctions disponible de base en C++
- Elle contient de nombreux modules :
  - Structures de données : chaînes de caractères, tableaux, piles...
  - Algorithmes classiques : tri, $n^{\text{ième}}$ éléments...
  - Lecture/écriture (console, fichiers, réseau...)

Quelques exemples

Nous avons déjà utilisé la STL, notamment les modules `iostream` et `string`.

Les vecteurs de la STL

La STL propose une implémentation de vecteurs dans la classe `vector`.

- Utilise les tableaux dynamiques
- Abstraction de la gestion de la mémoire
- Expose une interface type tableau

Utilisation :

```
#include <vector>

using namespace std;
```

Classe template

La classe `vector` est une classe template, elle peut s'adapter à tous les types de données. À la compilation, la classe est spécialisée pour le type de données en question.

```
// Création d'un vecteur d'éléments de type T
vector<T> tab;

// Exemples :
vector<int> t_int;
vector<double> t_double;
vector<Matrix> t_mat;
vector<float*> t_point;
```

Manipulation des vecteurs de la STL

Les vecteurs de la STL s'utilisent de la même façon que les tableaux statiques ou dynamiques.

```
// Création d'un vecteur d'entiers de 100 éléments
vector<int> t(100);

// L'accès aux cases se fait par l'opérateur []
for(int i=0; i<100; i++){
    t[i] = i*i;
}
cout << t[5] << endl;
```

Les vecteurs disposent d'une méthode `size` permettant de récupérer leur taille :

```
cout << t.size() << endl;
```

Manipulation des vecteurs de la STL

- Création et remplissage :

```
// Création d'un vecteur de 100 réels valant tous 5.6
vector<double> t2(1000, 5.6);
```

- Redimensionner un vecteur :

```
// Création d'un vecteur de 100 éléments de type T
vector<T> t(100);
// Redimensionnement
t.resize(1000);
```

Les éléments déjà existants sont conservés, sauf si la taille finale du vecteur est inférieure à la taille initial, auquel cas les éléments surnuméraires disparaissent.

Manipulation des vecteurs de la STL

- Accéder au premier élément :

```
cout << *t.begin() << endl;
```

Il s'agit d'un itérateur, qui fonctionne de manière similaire à un pointeur.

- Accéder à la fin du vecteur :

```
t.end(); // Attention pointe juste derrière la dernière case
```

Ce n'est pas le dernier élément!

Manipulation

- Concaténer deux vecteurs :

```
vector<int> t1 (10,2);
vector<int> t2 (30, 100);
t2.insert(t2.end(), t1.begin(), t1.end());
```

On ajoute le vecteur `t1` à la fin du vecteur `t2`.

- Trier un vecteur :

```
#include <algorithm>

...

std::sort(t.begin(), t.end());
```

TP

Serpent

Un serpent qui se déplace et s'allonge tout les x pas de temps.

Tron

Un serpent deux joueurs qui s'allonge à tout les pas de temps.