Benjamin Hamon

Sophie Huart

Alexandre Fischer

Sébastien Corbeau

Alain Le

Jeremy Moriaux

Objet avancé

Programmation générique

# Contexte du projet

L’objectif du projet est de créer une bibliothèque logicielle C++ en utilisant les principes de programmation générique et des techniques d’implantations associées vues en cours (traits, méta-programmation, tag dispatching, EDL, etc…).

Notre sujet porte sur les conteneurs multidimensionnels. L’objectif est de trouver des concepts pour gérer et implanter des conteneurs de dimension N arbitraire. Pour cela, il faut fournir une implantation d’un tableau à N dimensions et d’en inférer les extensions nécessaires pour gérer de manière homogène d’autres conteneurs en mode N dimensions.

# Organisation

Un dépôt GITHUB est mis à notre disposition : https://github.com/bhamon/genericprogramming

Le dépôt est divisé en plusieurs parties :

* Les répertoires « include » et « src » : ils contiennent les sources de notre librairie. Ces sources sont écrites dans l’espace de noms « pps::range ».
* Le répertoire « tests » : il contient les différents tests unitaires de la librairie. Chaque fichier cpp est compilé séparément. Le fichier « tests.h » contient un ensemble de fonctions utilitaires pour réaliser les tests.
* Les fichiers Makefile : ils permettent de compiler les tests ainsi que la librairie.

# Démarche

## Premiers pas

Un tableau multidimensionnel au sens large du terme peut prendre diverses formes en C++, comme le montre l’extrait de code ci-dessous :

std::vector<int> t1;

std::list<std::deque<double\*\*>> t2;

char t3\*[20];

On constate d’entrée de jeux que l’on peut séparer les conteneurs en trois catégories :

* Les classes de conteneur génériques.
* Les tableaux natifs du langage.
* Les pointeurs sur des séries d’éléments contigus en mémoire.

Dans tous les cas, on peut dégager un modèle en s’inspirant de la « standard library ». Ce modèle sera spécialisé pour prendre en compte les différentes implémentations concrètes.

Ainsi, nous avons fait le choix de considérer une dimension d’un tableau comme un « range », c’est-à-dire une paire d’itérators permettant d’accéder à la dimension inférieure, jusqu’à atteindre les données.

La spécification des types de données et des types d’iterators s’effectue par le biais d’un traits dont l’implémentation est localisée dans le fichier « traits.h ». Ce traits a été spécialisé pour les tableaux natifs, les pointeurs sur des éléments contigus en mémoire et les conteneurs de la « standard template library ».

La récupération de la paire d’iterators d’une dimension s’effectue par le biais des fonctions « begin » et « end », localisées dans les fichiers « begin.h » et « end.h ». De la même manière que pour le traits, la fonction « begin » a été spécialisée pour les tableaux natifs, les pointeurs et les conteneurs de la STL. Néanmoins, la fonction « end » n’a été spécialisée que pour les conteneurs de la STL, la récupération du nombre d’éléments présents dans un tableau natif et dans un espace mémoire pointé ne pouvant être effectué.

Grâce à ce panel, il est possible de gérer n’importe quel tableau multidimensionnel formé de tableaux natifs, de pointeurs et de conteneurs de la STL, comme le montre le fichier de test « test\_usage.cpp ».