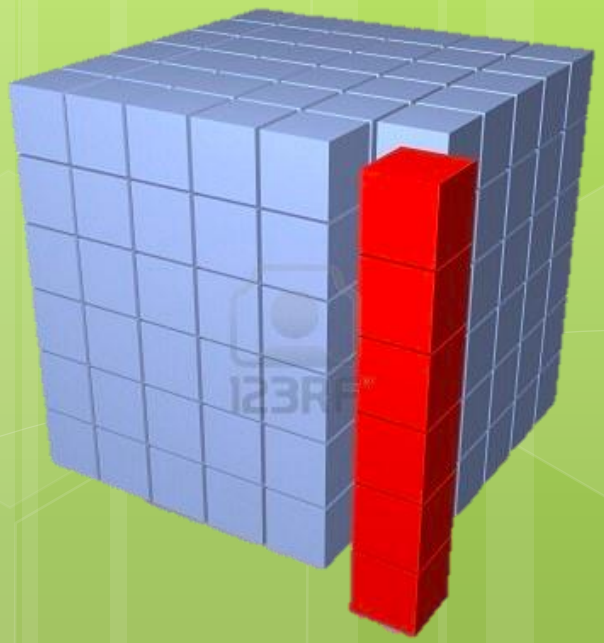


Objets Avancés

Benjamin Hamon
Sophie Huart
Alexandre Fischer
Sébastien Corbeau
Alain Le
Jeremy Moriaux

SOMMAIRE

- Contexte du projet
- Organisation
- Démarche
- Aide à la manipulation
- Notion de vue
- Optimisation

Contexte du projet

- Créer une bibliothèque logicielle C++
- Programmation générique

- Conteneurs multidimensionnels
- Tableaux à N dimensions

- Gestion homogène de tout type de conteneurs

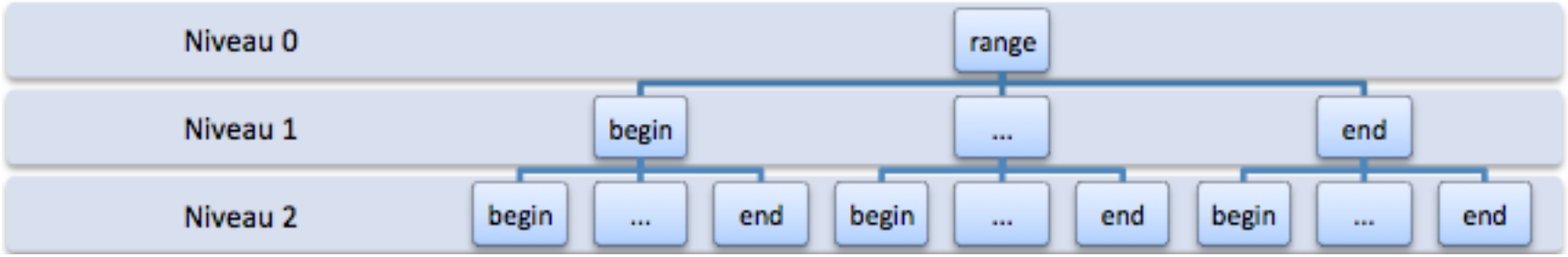
Organisation

- Mise en place d'un dépôt GITHUB
- Répertoires Sources : « src » et « include »
- Répertoire de tests : « tests »
- Les fichiers Makefile

Démarche

- Trois catégories de conteneurs:
  - Les classes de conteneur génériques.
  - Les tableaux natifs du langage.
  - Les pointeurs sur des séries d'éléments contigus en mémoire.

Démarche



- Une dimension = Un « range »
- Type de données et type d'itérators par le biais d'un « traits »
- Paire d'itérators : « begin » et « end »

Démarche - extensibilité

- 3 étapes pour intégrer un nouveau type de conteneur :
  - Spécialisation du traits
  - Définition des fonctions begin (const et no-const)
  - Définition des fonctions end

Aide à la manipulation

- Pour le moment, écriture un peu lourde.
- Type de base difficilement reconnaissable.

- Introduction d'un second « traits ».
- Permet l'inspection des types consécutifs des tableaux.
- Notion de dimension.

Notion de vue

- Permet de supporter les types natifs
  - Tableaux (int[20])
  - Pointeurs (doube***)
- Ajoute une fonctionnalité de sélection de sous-espace
  - Cas d'application : matrices

Optimisation – Les défauts

- Défauts constatés :
  - Récursivité pour l'exploration des dimensions
  - Création de nombreux objets temporaires (indices, vues, …)
  - Perte de performance due à la généricité (random access iterator)

Optimisation – La solution

- Création d'un sous-langage de manipulation des tableaux
  - Mise en place d'un AST
- Gain de lisibilité
- Gain de temps d'exécution
- Réduction de l'occupation mémoire

Merci de votre attention

- Des question ?