Programmation

Convention d'écriture de code

Erwan Guillou

UFR Informatique
Université Claude Bernard Lyon1

7 novembre 2008

Plan du cours

Conventions d'écriture

- Chaque programmeur a sa propre façon de coder.
- Chaque éditeur (de texte) a sa propre façon de présenter le code source.
- Un gros projet implique :
  - de nombreux programmeurs
  - un très grande quantité de ligne de code

La problématique

Les contraintes du développement de projet
- beaucoup de développeurs
- le code est conséquent
- Il est souvent nécessaire de revenir en arrière

Exemple : Studio de dev de jeux
- Plusieurs projets + outils communs
- 80 développeurs et 50 Go de sources (et bibliothèques)

Donc...
- il faut coder proprement

Conventions d'écriture

Objectif : normaliser la rédaction du code pour
- Augmenter la lisibilité et la compréhension du code source.
- Obtenir du code prédictible et facilement modifiable.
- Que tous les développeurs d'un même projet puissent comprendre et apréhender le code des autres.

Attention
- Ce ne sont que des recommandations...
- Il faut s'en fixer lorsque l'on travaille à plusieurs sur un même projet

Les fichiers

Extensions
- Les fichiers d'entête : .h
- Les fichiers source C : .c, .c++, .C, .cc ou .cpp

Classes/Structures
- déclarée dans un fichier d'entête définie dans un fichier source.
- Le nom des fichiers devrait correspondre au nom de la classe.
- **Exception :** les classes génériques («template»)

Implementations
- Les implémentations se trouvent dans les fichiers source.
- **Exception :** les fonctions inline

Les fichiers

Contenu
- Le contenu d'un fichier ne doit pas dépasser 80 colonnes.
- Les caractères spéciaux comme «TAB» et le saut de page («page break») doivent être évités.
- La césure des lignes trop longues doit être effectuée d'une manière lisible, logique et évidente

Exemple

```
somme = a + b + c +
        d + e;
for ( int noTable = 0 ; noTable < nTables ;
      noTable += sautTable )
...
```


Les fichiers

Divers
- Les fichiers d'entête doivent contenir une garde d'inclusion multiple.
- Les énoncés doivent pouvoir se faire indépendamment du système dexploitation.
- Les énoncés d'inclusion doivent se trouver seulement au début d'un fichier.

Exemple

```
#ifndef __MODULE__
#define __MODULE__
...
#endif
```

Règles de nommage

Classes / types

- en minuscules, le premier caractère en majuscule
- le début de chaque nouveau mot en majuscule.

Variables / fonctions / méthodes

- en minuscules, le premier caractère en minuscule
- le début de chaque nouveau mot en majuscule.

Exemple

```
Types : Ligne, SystemeAudio, PointDeControle
Variables : ligne, application, compteur, compteurDeLigne
Fonctions : obtenirInstance(), lireNom()
```


Règles de nommage

Constantes

- en majuscules
- caractère souligné entre chaque mot.

Pour augmenter la lisibilité

- remplacer les constantes par des méthodes qui retournent une valeur.

Exemple

```
MAX_VITESSE, COULEUR_ROUGE, PI
int obtenirVitesseMaximale() {
    return 25;
}
```


Règles de nommage

Autres déclarations

- Namespaces
  - en minuscules
- Les types génériques
  - une lettre majuscule
- abréviations
  - en minuscules

Exemple

```
model::analyzer, io::iomanager, math::geometry
template<typename T>, template<typename C, class D>
exportHtmlSource(); // NON: exportHTMLSource();
openDvdPlayer(); // NON: openDVDPlayer();
```


Règles de nommage

Conventions

- obtenir»/«modifier» , «lire»/«ecrire»
  - là où un attribut est accessible directement.
- calculer», méthodes de calcul.
- trouver», méthodes de recherche.
- initialiser», méthode d'initialisation.

Exemple

```
line.getLength(); // NON: line.getLineLength();
ligne.lireLongueur(); // NON: ligne.lireLongueurLigne();
```


Règles de nommage

Conventions

«

- obtenir»/«modifier» , «lire»/«ecrire»
  - là où un attribut est accessible directement.

«

- calculer», méthodes de calcul.

«

- trouver», méthodes de recherche.

«

- initialiser», méthode d'initialisation.

Exemple

```
employe.obtenirNom();
matrice.calculerInverse();
imprimante.initialiserStyle();
```


Règles de nommage

Divers

- Les pluriels, pour une collection d'objets.
- Le préfixe «n», variables représentant un nombre d'objets.
- Le préfixe no, variables représentant un numéro d'entité.
- Les variables d'itération, un caractère minuscule

Exemple

```
vector<Point> points; int valeurs[];
nPoints, nLignes
EmployeeNo noEmploye
i, j, k etc.
```


Les déclarations

Les types

- Les types locaux à un seul fichier doivent être déclarés à l'intérieur de ce fichier.
- Classes
  - Les parties d'une classe doivent être ordonnées de la manière suivante : public, protected et private
  - Chaque section doit être identifiée explicitement.
  - Les sections non applicables ne doivent pas être mentionnées.
- La conversion des types : se fait de manière explicite, on ne doit jamais dépendre de la conversion implicite.

Les déclarations

Les variables

- Les variables devraient être initialisées lorsqu'elles sont déclarées (ou au plus tôt)
- L'utilisation des variables globales devrait être minimisée.
- Les attributs de classes ne devraient jamais être déclarées publics.
- **Exception :** structure de données, sans comportement

Les pointeurs et références C++

- le symbole doit être au plus près du type plutôt que du nom.

Les structures de contrôle

Les boucles

- Seuls les énoncés de boucle doivent être inclus dans la construction for().
- Les variables de boucle doivent être initialisées juste avant la boucle.
- L'utilisation de boucles do-while peut être évitée.
- L'utilisation de break et continue dans les boucles doit être évitée.
- La forme while (true) doit être utilisée pour les boucles infinies.

Les structures de contrôle

Les conditionnelles

- Les expressions conditionnelles complexes doivent être évitées.
- Le cas le plus fréquent d'une construction if doit être mis dans la partie if-then et l'exception dans la partie else.
- Le code conditionnel doit être mis sur une ligne distincte.
- Les énoncés qui exécutent du traitement ne doivent pas se trouver à l'intérieur de conditions.

Pour aller plus loin

Références

- C++ Programming Style Guidelines
  http ://geosoft.no/development/cppstyle.html
- Code Complete, Steve McConnel - Microsoft Press
- Programming in C++, Rules and Recommendations
  M Henricson, e. Nyquist, Ellemtel (Swedish telecom)
  http ://www.doc.ic.ac.uk/lab/cplus/c++.rules/
- C++ Coding Standard
  Todd Hoff
  http ://www.possibility.com/Cpp/CppCodingStandard.htm
- C / C++ / Java Coding Standards from NASA
  http ://v2ma09.gsfc.nasa.gov/coding_standards.html