Conventions de nommage/codage

Table des matières

1 Général

Tout doit être écrit en anglais. (Éventuellement pas les commentaires).

2 Fichiers

2.1 Fichiers Header (.h)

Chaque fichier .h contient un #define pour empêcher les inclusions multiples.

*#ifndef FILENAME_H_*

*#define FILENAME_H_*

…

*#endif // FILENAME_H_*

3 Nommage

3.1 Règle général de nommage

Les noms de fonctions, de variables de fichiers devraient être descriptifs : éviter les abréviations.

3.2 Noms des fichiers

Les noms des fichiers ne sont composés que de **minuscules**. Les mots peuvent être séprarés par des « _ ». *(my_useful_class.cpp – my_useful_class.h)*

3.3 Noms des variables

3.3.1 Nom des variables communes

Les variables commencent par une minuscule et chaque nouveau mot commence par une majuscule.

string tableName ;

3.3.2 Noms des variables de classes

Même chose que les variables communes ;

class TableInfo {

private :

    string tableName ;

} ;

3.4 Noms des constantes

Tous est en majuscule et les mots sont séparés par un « _ ».

const int DAYS_IN_WEEK = 7 ;

3.5 Noms des fonctions et méthodes

3.5.1 Noms des fonctions/méthodes régulières

Les fonctions commencent par une minuscule et chaque nouveau mot commence par une majuscule.

void addTableEntry() ;

3.5.2 Noms des accessors/mutators (getters/setters)

Les getters/setters commencent par get/set suivi de la même convention que pour les fonctions normales (le 1er mot commençant par une majuscule!).

int getNumEntries() ;

void setNumEntries(int newValue) ;

3.6 Noms des enumérations

Le nom de l'énumérations commence par une majuscule et ses champs se nomment comme les constantes.

```
enum AlternateUrlTableErrors {
        OK = 0,
        OUT_OF_MEMORY = 1,
} ;
```

3.7 Noms des classes

Les classes ont la même convention que les variables normales. Elles sont juste une majuscule au début.

```
class TableInfo {
} ;
```

4 Commentaires

4.1 Entête de fichiers

Chaque fichier doit avoir un commentaire en haut contenant :
- le nom de l'autre (des auteurs)
- la date de création
- une description de son contenu.

En général, un fichier .h décrira les classes qui sont déclarées dans le fichier avec un aperçu de ce qu'elles sont et comment elles sont utilisées.
Un fichier .cpp devrait contenir plus d'informations sur les détails de mise en œuvre ou des discussions sur des algorithmes difficiles.

Essayer de ne pas dupliquer les commentaires entre le .h et le .cpp .
Les commentaires doubles peuvent diverger.

*/**

** Auteur :Thibaud Duchoud*

** Date : 22.04.2015*

***

** Description*

** ----------*

** Bla bla bla.*

**/*

4.2 Commentaires de fonctions

4.2.1 Déclaration de la fonction (.h en général)

Chaque déclaration de fonction devrait avoir des commentaires qui décrivent ce que fait fonction et comment l'utiliser.

Types de choses à mentionner dans les commentaires à la déclaration de la fonction:

- Description de ce que fait la fonction et comment l'utiliser.
- Description des entrées/sorties. (param. entrés et retour).
- Si la fonction alloue de la mémoire que l'appelant doit libérer.

4.2.2 Définition de la fonction (implémentation)

Eventuellement si la fonction fait quelque chose de délicat, il faudrait un commentaire

explicatif. (Ne pas répéter les commentaires du .h).

4.3 Commentaires des classes

Chaque définition de classe doit avoir un commentaire qui décrit cette classe et comment elle doit être utilisée.

5 Autres

5.1 Accolades

Je pense qu'on va utiliser les accolades avec cette convention (accolades « du haut » juste après le nom et non par après un retour à la ligne).

Class A {

} ;

if(a == 1) {

} else if (a == 2) {

} else {

}

5.2 Tabulation ou espace ?

On utilise soit des espaces soit des indentations, pas les deux.

Je pense qu'il est mieux d'utiliser 3 espaces comme indentation.

Configurer l'IDE pour qu'il utilise 3 espaces comme tabulation.

6 Sources

https://google-styleguide.googlecode.com/svn/trunk/cppguide.html

http://geosoft.no/development/cppstyle.html