Coding Standard

AP4A/B

17 septembre 2020

1 Introduction

Un coding standard a pour objectif de vous fournir un ensemble de règles de bon usage qui vont vous permettre d'avoir un code propre, qui se maintient facilement et compréhensible. Chaque entreprise peut vous en fournir un et son application est impérative.

2 Création des fichiers

- Toutes les déclarations de classes doivent être présentes dans un fichier MaClasse.hpp
- Les implémentations des classes se feront dans un fichier MaClasse.cpp
- Un bloc de commentaire présentant l'auteur, la date de création, la licence (privé ou libre) et le nom de la classe doit être présent en en-tête de chaque fichier.

3 Header et déclaration de classe

Tous les fichiers hpp doivent posséder les guards. Un guards est un principe de programmation c qui permet de limiter le nombre de définition. Un non respect de ceci provoquera une erreur de définition lors de l'inclusion multiple de ce fichier. Exemple :

```
#ifndef NOMFICHIER_H
#define NOMFICHIER_H

// Declaration classe, fonctions...

#endif // NOMFICHIER_H
```

Les commentaires présentant le fichier, la classe, les méthodes et les attributs doivent être écrites en doxygen (générateur automatique de documentation). Exemple :

```
/**
 * @author nom_auteur
 * @file MaClass.hpp
 * @date 15/09/2020
 * @brief Description de la classe, son objectif...
 */
```

Les noms de classes sont écrits en Pascal case (nommé aussi upper camel case).

```
class MaClasse
```

Les attributs de classe sont écris en camel case et commencent avec "m_". Par exemple, si la classe MaClass possède un attribut count, alors le nom de l'attribut sera m_count.

```
class MaClass
{
public:
        float m_maVariable;
}
```

les keywords de visibilité devront être au même niveau que les accolades.
Les méthodes/fonctions sont écrites en camel case.

```
void maFonction();
```

Les paramètres de fonctions s'écrivent sous le format suivant :

```
void maFonction(Type param_p);
```

Concernant les énumérations, le caractère 'E' doit être placé en début de nom. Les valeurs d'enum sont écrites en snake case ; commençant par un "e_". Exemple :

```
enum EMonEnum :
{
        e_ma_valeur_1,
        e_ma_valeur_2,
}
```

Ne jamais utiliser "use namespace std" dans un fichier header ! De manière générale, éviter d'utiliser un "use namespace" dans un fichier header.

4 Règle au niveau de la syntaxe du code

Les déclarations de pointeur doivent commencer par 'p' avant le nom de la variable. Exemple :

```
float* pPointeurFloat = nullptr;
```

Le caractère désignant le type référence ou pointeur (respectivement '*' et '&') doit être collé au type de base. Exemple :

```
float variableFlot = 5.0f;
float* pPointeurFloat = nullptr;
float& referenceSurMaVariable = fVariableFloat;
```

Les accolades sont toujours mis à la ligne, décalées par une tabulation par niveau de visibilité. Exemple :

```
if (condition)
{
        if (condition2)
        {
                // Faire quelque chose
        }
}
```

De plus, pensez à régler votre tabulation dans votre IDE pour qu'elle correspondent à 2 espaces.

Les switchs doivent être écrites de cette manière :

```
switch (condition)
{
        case 1 :
        {
                // Faire quelque chose
        }
        break;

        //...

        default :
        {
                // Faire quelque chose
        }
        break;
}
```

Ne pas hésiter à commenter, même des lignes qui vous semble clair. Ce qui est clair le jour de l'écriture du code ne le sera peut être plus dans 1 ou 2 mois.

5 Annexe

Quelques exemples de classes implémentant le coding standard :

Listing 1 – Header standard d'une classe C++

```
/**
 * @author auteur
 * @file MaClass.hpp
 * @date 15/09/2020
 * @brief Description de la classe, son objectif
 */

//
// Define guards
#ifndef MACLASS_H
#define MACLASS_H

/**
 * @enum EMonEnum
 * @brief Description de l'enum
 */
enum EMonEnum :
{
        e_ma_valeur_1,
        e_ma_valeur_2,
}

/**
 * @class MaClass
 * @brief Description de la classe
 */
class MaClass
{
public:
        // Definition de la forme canonique

        // ...

        /**
         * @brief Description de la methode
         * @return retour de la methode
         * @param parametre - parametre demande
         */
         int getCount() const;

         void DoSomething(const Type& param);
```

```
 private:
         int m_count; ///< descriptif de l'attribut
 }

 #endif // MACLASS_H
```

Listing 2 – Implementation d'une classe C++

```
/**
 * @author auteur
 * @file MaClass.cpp
 * @date 15/09/2020
 * @brief Description de la classe, son objectif...
 */

#include "MaClass.hpp"

int MaClass::getCount()
{
        return m_count;
}

void MaClass::DoSomething(const Type& param)
{
        if (condition)
        {
                // Do something
        }
}
```