

Standard de codage

Définition (Wikipédia)

Les règles de codage sont un ensemble de règles à suivre pour uniformiser les pratiques de développement logiciel, diffuser les bonnes pratiques de développement et éviter les erreurs de développement “classiques” au sein d’un groupe de développeurs. Les règles de codage s’articulent autour de plusieurs thèmes, les plus courants étant :

1. Le nommage et l’organisation des fichiers du code source
2. Le style d’indentation
3. Les conventions de nommage, ou règles de nommage
4. Les commentaires et documentation du code source
5. Recommandations sur la déclaration des variables
6. Recommandations sur l’écriture des instructions, des structures de contrôle et l’usage des parenthèses dans les expressions.

Ressources documentaires

[Wikipedia](#)

[GNU Coding standard](#)

[clang_format](#)

[clang](#)

[LLVM](#)

Indentation du code

En [informatique](#) : [l’indentation](#) consiste en l’ajout de tabulations ou d’espaces dans un fichier, pour une meilleure lecture et compréhension du code

Indenter automatiquement votre code

- Vous pouvez indenter automatiquement sous Qt le code source par les raccourcis suivants

CTRL+A sélectionne le texte

CTRL+I formate automatiquement la sélection dans QT

Lignes orphelines

Les lignes sans codes, ou lignes orphelines doivent être supprimées, sauf si elles servent à délimiter les blocs d'instructions (comme par exemple entre la déclaration des variables et le début des instructions)

Conventions de nommage à respecter

Déclaration des variables

- Les variables doivent avoir des noms représentatifs de leur contenu

****Exemple (je veux déclarer un entier contenant des notes) ****

```
int notes;
```

■

- Pour les tableaux, je fais précéder le nom du tableau de `tab`

****Exemple (Je veux déclarer un tableau contenant des notes (comme 14,34 ou 5,75) ****

```
float tabNote[10];
```

■

- Pour les pointeurs, je fais précéder le nom du pointeur de `ptr`

```
int *ptrNote = nullptr;
```

■

Convention de nommage

Je respecte la notation [lowerCamelCase](#) pour mes variables et fonctions

- Premier mot en minuscule
- Les mots suivants avec une majuscule

Variables et fonctions

****Exemple (Je veux déclarer un entier contenant un nombre de livres) ****

```
int nombreLivre;
```

■

****Exemple (Je veux déclarer une fonction pour sauver une image en niveau de gris) ****

```
void sauverImageNiveauGris();
```

■

Les macro

- Les noms des macro doivent être en majuscules

****Exemple, une macro qui calcule le max de deux nombres ****

```
#define MAX(x,y) ((x)>(y)?(x):(y))
```

■

Les structures

- Le type de la structure doit commencer par une majuscule

****Exemple, j'ai une structure `Pixel` ****

```
typedef struct {  
    unsigned char red;  
    unsigned char blue;  
    unsigned char green;  
} Pixel;
```

■

- Je déclarerai une variable du type Pixel comme indiqué ci-dessous

```
Pixel monPixel;
```

■

Les classes

NOM DES CLASSES

- Chaque classe commencera par 'C' suivi du nom de la classe avec la première lettre en majuscule

****Exemple, une classe contenant des informations sur des avions sera déclarée ainsi ****

```
class CAvion {  
    public :  
        //Méthodes publiques  
    private :  
        //Méthodes et attributs privés  
};
```

■

MÉTHODES DE LA CLASSE

- Les méthodes de la classe respectent la convention lowerCamelCase

ATTRIBUTS DE LA CLASSE

- Pour les attributs, je fais précéder le nom de l'attribut de "_"

****Exemple (un attribut contenant un prénom) ****

```
string _prenom;
```

■

CLASSE CAVION AVEC MÉTHODES ET ATTRIBUTS

```
class CAvion {  
    public :  
        int controlerAssiette();  
    private:  
        float _altitude;
```

```
};
```

Revision #2

Created 28 May 2024 12:35:12 by Nicolas

Updated 1 February 2025 00:10:10 by Nicolas