|  |
| --- |
| **DATALOG** |
| Big Data – Système Embarqué |
| **Charte de nommage** |

|  |
| --- |
| HASSAN AUNIM  27/06/2016 |

Table des matières

[1. But de ce document 3](#_Toc453924794)

[2. Introduction 3](#_Toc453924795)

[3. Conventions de nommage 3](#_Toc453924796)

[3.1. Types class / struct / union / enum 3](#_Toc453924797)

[3.2. Variables 4](#_Toc453924798)

[3.3. Constantes et contenu des enums 4](#_Toc453924799)

[3.4. Fonctions 4](#_Toc453924800)

[3.5. Méthodes 4](#_Toc453924801)

[3.6. Fichiers 5](#_Toc453924802)

[4. Convention de codage 5](#_Toc453924803)

[4.1. Blocs 5](#_Toc453924804)

[4.2. Conditions 5](#_Toc453924805)

[4.3. Longueur d’une ligne 6](#_Toc453924806)

[4.4. Commentaires 6](#_Toc453924807)

[4.5. Ordre de déclaration des membres dans une classe 7](#_Toc453924808)

# 1. But de ce document

Ce document décrit les conventions de codage C++ appliquées durant ce projet. Ce document est destiné aux développeurs contribuant au code source C++ du projet.

# 2. Introduction

Le but de ce document est de définir le style de programmation en C++ pour le projet Hermes. Les règles et recommandations présentées ici ne sont pas définitives, mais doivent être vues comme un document dynamique. Des suggestions d'améliorations sont les bienvenues.

Les programmes développés selon ce document devraient être :

* Correct
* Facile à maintenir

De façon à atteindre ce but, les programmes devraient :

* Avoir un style de codage défini
* Être facile à lire et à comprendre
* Être portable à d'autres architectures
* Ne pas comporter d'erreurs
* Être maintenable par différents programmeurs

Les questions de modélisation ne concernent pas ce document.

# 3. Conventions de nommage

## 3.1. Types class / struct / union / enum

Les noms de classe / struct / union et enum commencent par une majuscule. Si le nom est composé de plusieurs mots, chaque mot commence par une majuscule, et est collé. (Upper Camel Case)

Exemple : RuleManager, InputPluginHandle

## 3.2. Variables

Le nom commence par une minuscule. Si le nom est composé de plusieurs mots, chaque mot commence par une majuscule à l’exception de la première lettre, et est collé. (Lower Camel Case)

Exemple : ruleManager, inputPluginHandle

## 3.3. Constantes et contenu des enums

Les noms doivent être en majuscule.

Exemple : GRAVITY, RESET

## 3.4. Fonctions

Le nom commence par une minuscule. Si le nom est composé de plusieurs mots, chaque mot commence par une majuscule à l’exception de la première lettre, et est collé. (Lower Camel Case)

Exemple : ruleManager, inputPluginHandle

## 3.5. Méthodes

Dans une classe les accesseurs commencent en général par get (une exception peut être faite pour les booléens ou l'on peut remplacer par is (ex : isAlive()) A l'inverse les méthodes servant à fixer une valeur sont préfixés de set. (Lower Camel Case)

Exemple : getSomething(), setSomething()

Dans le cas d’un constructeur ou d’un destructeur, il y a une majuscule en début de mot, à l’identique du nom de la classe.

Exemple : Manager(), Accelerometer()

## 3.6. Fichiers

Le nom du fichier doit correspondre au nom de la classe principale qu'il contient, s'il en contient une. La déclaration des classes est faite dans le .h portant le nom de la classe. La définition dans le .cpp du même nom.

Exemple :

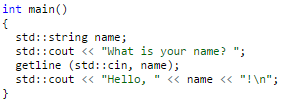
* Buffer.h : déclaration de la classe
* Buffer.cpp : définition des méthodes de la classe Buffer

# 4. Convention de codage

## 4.1. Blocs

Les accolades ne suivent pas un mot du langage, et le code à l’intérieur est indenté. (Allman)

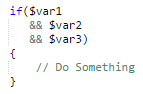
Exemple :



## 4.2. Conditions

Les conditions ne doivent pas dépasser la largeur de l’écran. Si c’est le cas elle doivent être coupées en morceaux.

Exemple :



**Note**: Si le contenu d’une condition ou d’une boucle peut être écrite sur une seule ligne alors les accolades ne sont pas nécessaires.

## 4.3. Longueur d’une ligne

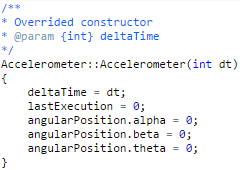
Les lignes de code ne doivent pas dépasser la largeur de l’écran.

## 4.4. Commentaires

La déclaration de chaque fonction ou méthode est nécessairement précédé d’un commentaire explicite sous la forme d’un commentaire long (/\*\* \*\*/). Il peut y avoir les balises @param afin de définir les différents arguments passés à la méthode.

Les commentaires doivent être exclusivement en anglais.

Exemple :



## 4.5. Ordre de déclaration des membres dans une classe

Les membres publics sont présentés en premiers, ce sont eux qui intéressent un utilisateur de la classe. Viennent ensuite les membres protected puis les membres privés. Pour un même niveau d'accessibilité les méthodes sont présentées après les variables membres.

Exemple :

