Coding standard

Table des matières

[Organisation 2](#_Toc32919888)

[Convention de nommage 2](#_Toc32919889)

[Fichier 2](#_Toc32919890)

[En-têtes 2](#_Toc32919891)

[Enumérations 2](#_Toc32919892)

[Classes, structures et namespaces 2](#_Toc32919893)

[Fonctions globales 3](#_Toc32919894)

[Fonctions membres 3](#_Toc32919895)

[Variables 3](#_Toc32919896)

# Organisation

Les classes sont organisées et écrites en gardant le lecteur et l’utilisateur de la classe en tête. L’interface de la classe doit être claire, simple et commentée.

# Convention de nommage

## Fichier

* Les fichiers commencent suivent la convention CamelCase upper case.

### En-têtes

* Les fichiers d’entêtes doivent être entourés par :

|  |
| --- |
| #pragma once  #ifndef HD\_INC\_PROJET\_FICHIER\_H  #define HD\_INC\_PROJET\_FICHIER\_H  …………  #endif // HD\_INC\_PROJET\_FICHIER\_H |

* + PROJECT : Nom du projet en CAPITAL
  + FICHIER : Nom du fichier en CAPITAL

## Enumérations

* Les énumérations commencent par la lettre ‘E’ puis suivent la convention CamelCase upper case.
* Les membres de l’énumération suivent la convention snake\_case.

|  |
| --- |
| enum class hdEEnum {  Var\_1 = 'c',  var\_2 = 52,  }; |

## Classes, structures et namespaces

* Les classes, structures et namespace sont définies dans le namespace HUD\_NAMESPACE et suivent la convention CamelCase upper case.
* La déclaration des fonctions et attributs dans les structures et classes doit respecter l’ordre suivant :

1. Les définitions de type publique après le mot clé public:
2. Les fonctions publiques après le mot clé public:
3. Les attributs membres public après le mot clé public:
4. Les fonctions privées après le mot clé private:
5. Les attributs membres privés après le mot clé private:

* Les mots clés public, private, protected sont obligatoires à chaque changement de nature de la déclaration.

|  |  |  |
| --- | --- | --- |
| struct StructOne {  // ...  }; | class ClassOne {  // ...  }; | namespace NamespaceOne {  // ...  }; |

## Fonctions globales

* Les fonctions globales sont définies dans le namespace HUD\_NAMESPACE puis suivent la convention snake\_case
* Les paramètres sont nommés en suivant la convention snake\_case
* Les paramètres des templates utilise le mot clé typename

|  |
| --- |
| template<typename Type>  void function\_name(AttType att\_one, Type att\_two) {  // ...  } |

## Fonctions membres

* Les fonctions membre suivent la convention snake\_case
* Les paramètres sont nommés en suivant la convention snake\_case
* Les paramètres des templates utilise le mot clé typename

|  |
| --- |
| template<typename Type>  void function\_name(AttType att\_one, Type att\_two) {  // ...  } |

## Variables

#### Locales

* Les variables locales suivent la convention snake\_case.

|  |
| --- |
| i32 var\_one; |

#### Globales

* Les variables globales doivent être définies dans le namespace HUD\_NAMESPACE
* Les variables globales doivent commencer par la lettre ‘s’.
* Les variables globales doivent être déclarées dans une unité de compilation (.cpp).
* Les variables globales doivent être précédées du mot clé static.

|  |
| --- |
| // file.cpp  static i32 s\_var\_global; |

#### Membres

* Les variables membres suivent la convention snake\_case

|  |
| --- |
| struct AttType {  i32 member\_attribute;  }; |