**{EPITECH.}**

09

**Normes, Documentations**

**Tincani Geoffrey**

Sommaire

[Norme 3](#_Toc243207787)

[Norme commune aux différents langages 3](#_Toc243207788)

[Norme C++ 4](#_Toc243207789)

[Documentation 5](#_Toc243207790)

# Norme

## Norme commune aux différents langages

* Toujours préciser **signed** ou **unsigned**, afin d’assurer la compatibilité des différents compilateurs.
* N’utiliser que des boucles while ou mieux, do…while, beaucoup plus rapide en exécution.
* Les nom de méthodes ne doivent pas contenir de \_ et mettre en majuscule les différents mots : MaMethode..TaGeuleDeCon…
* Utiliser le moins possible de double et de float, si un int ou autre peut faire l’affaire. Et éviter notamment les cast float -> int -> float.
* Utiliser le moins de divisions et de multiplications possible, mais privilégier la multiplication à la division (c’est plus rapide de multiplier par 1/k que de diviser par k).
* Utilisation de calcul utilisant fréquemment les mêmes valeurs (genre cos(12, factoriel) ou autre. Dans de cas, mettez les valeurs dans un tableau ou utiliser la méta-programmation (pas de calculs les valeurs seront directement dans la lib).
* Utiliser les décalages de bit plutôt que les multiplications ou divisions si multiple de 2 : exemple => x << 2 au lien de x \*= 2.
* Calculé ce qui peut l’être : genre a = b \* 300\*10 ; a remplace par b \* 3000 ; le compilateur ne pré-calcule pas dans ce cas le 3000 (vu qu’il par de la gauche) alors que a = 300 \* 100 ; serait pré-calculée…

## Norme C++

* Ajouter **this->** pour l’accès aux attributs.
* Précisez (**void**) pour les méthodes sans paramètres.
* Ne pas **exporter** les classes qui n’ont pas besoin de l’être.
* Utiliser au maximum la **pré-incrémentation** (++i), qui est deux fois plus rapide (une opération en moins que i++) quitte à utiliser une ligne de plus.
* Pour les .h, **#ifndef \_\_NOMDELACLASSE\_H\_\_** …
* Utiliser des fonctions **inline** si possible, pré-compilations : que du bonheur.
* Spécifier **const** partout ou ca peut l’être.
* Ne pas **allouer dynamiquement** ce qui n’a pas besoin de l’être (spéciale dédicace …) Le tas est plus rapide que la pile (il semblerait).

# Documentation