Première version où j’ai mis les principales choses à aborder pour la charte en repompant sur celui de Laurent.

Comme c’est encore un brouillon, je propose un système de validation :

* Rouge : A méditer
* Bleu : A valider
* Noir : validé
  1. Règles générales
     1. Généralités

Nombre de ligne max par fichier ?

Indentation ? 2 espaces.

Taille max d’une ligne ? Où aller à la ligne ?

Accolades ?

Langue du code ? Anglais (nom de variables, de fonctions, de classes…)

Langage des commentaires ? Français ?

Langage de prog ? C ou C++ ? (grande question)

On verra plus tard

* + 1. Lignes vides

Combien de ligne vide entre chaque fonction ?

Séparer les déclarations de variables ?

Les différentes parties du corps d’une fonction ?

* + 1. Espaces

Différents cas :

« If », « While », « For » ?

Opérateurs binaires ?

Cast ?

* + 1. Commentaires

On uniformise // ou / \* \*/ ? ou on fait différent selon le cas ?

* + 1. Déclaration et Affectations

On sépare les affectations au début du code ? (voir les lignes vides)

On déclare séparément les variables de boucles avant de les utiliser ?

(donc pas de for (int i = 0; ….) ? )

On évite « truc ? machin : chose » ?

* + 1. Parenthèses

On bourrine les parenthèses pour bien séparer les conditions ?

Exemple : (charte de Laurent)

if (a == b && c == d) // ÉVITER

if ((a == b) && (c == d)) // OK

* 1. Règles spécifiques
     1. Fichiers

Classes ? Chaque classe dans un fichier .cpp du même nom et un .h .

Les fichiers seront contenus dans src contenant :

* Un dossier contenant les .cpp (nom spécial ?)
* Un dossier contenant les .h (nom spécial ?)
* Un fichier makefile
  + 1. En-tête
* Les noms (projet, fichier)
* La date de création,
* Auteur ?
* Une description de l’objectif du fichier,
* Historique ?

Mise en forme ? A voir, ou sinon, on reprend le design de la prof.

* + 1. Autres

Séparation des :

* #Include
* #define
* Structures
* Classes
* Corps des fonctions

Proposition dans la charte :

/\*=================================

NOM DE LA ZONE

=================================\*/

//contenu de la zone

/\*===============================\*/

(A méditer si on veut personnaliser à fond le projet)

* + 1. Variables

Déclaration et initialisation directe ?

Séparation des variables si y en a vraiment trop ? (genre telles variables sont pour la première partie de la fonction, celles là pour la 2e partie….)

Pas d’initialisation multiples genre truc = machin = 0

Pas d’initialisation chelou genre truc = (machin = chose – tartanpion) + bidule

Eviter les nombres qui se balladent dans le code => utiliser des #define

Variables locales ? Doit avoir un nom explicite, minuscule puis majuscules pour les mots suivants, sans espace ni underscore. Si pointeur, le nom finira par \_p

Variables globales ? lol ?

Nom de variables interdites comme for, while, go…

Constantes dans les #define ? tout en majuscule et underscore ?

* + 1. Fonctions et procédure

Toujours mettre la signature/proto dans le .h

Corps :

{

1ère partie : Déclaration des variables

2ème partie : utilisation d’assert, vérifier les données si elles sont bonnes

3ème partie : On traite tout ce bordel

Return machin si y a un type retour

}

Pas de return barbare dans la 3ème partie.

* 1. Règles de nommage

**Fichier** : tout en minuscule, mots séparés par un : \_   
ex : liste\_simplement\_chainée

**Classe** : Premiere lettre en majuscule et chaque mot debute par une maj sans \_   
ex : MaPremiereClasse[?](http://code.google.com/p/ig3projetc/w/edit/MaPremiereClasse)

**Fonctions** : premiere lettre en minuscules, chaque autre mot commence par une maj sans \_

* Action : verbe calculeFermTrans
* Vérification : is\_lol

Toujours un type retour (ou void)

Toujours quelque chose dans les paramètres (ou void)

**Variables** : alors pour les

* variables membres des classes quelque chose comme ca : m\_variableMembre
* variables non membres (parametres de fonction ou variable de fonction): variableDeFonction
* pointeur : variable\_p
* Compteur, hors compteur de boucle : variable \_cpt ?

**Paramètres :** On indique si c’est input \_i, output \_o, ou \_io ?

**Structures :** Si on en fait, je propose qu'elles finissent par un s pour qu'on identifie ca directement

**Abréviations autorisées :** min, max ?

Y a des variables ou on met le type avant genre p\_pointeur, c\_classe, cpt\_compteur… au lieu de pointeur\_p, classe\_c, compteur\_cpt.