Norme de programmation : NoobZik-Dev

Version : v0.01 (En cours de rédaction)

Cette norme est inspiré de l'école 42. Le but d'une norme est de rendre un code source facilement lisible par tout le monde.

Si vous êtes dans mon groupe, vous vous engager à respecter scrupulusement cette norme

Pré-requis

- Connaissance parfaite sur le fonctionnement environnement de développement GIT.
- Connaissance partiel du fonctionnement de Gitkraken.
- Aptitude à être autonome sur la correction des erreurs de compilations débile.
- Coder sur Atom ou Visual Code (**GEDIT STRICTEMENT BANNI**) Avec son header 42.

Note pour atom : Vous devrez récuperer la version modifié du module header 42 à cette adresse (Il est écrit comment appliquer cette modification.).

Quelques plugins pour simplifier la vie (atom)

- Docblocker (Permet d'écrire les commentaire de manière rapide).
- File icon (Permet de colorer les fichiers, dans le but de switcher rapidement).
- Linter (Pour la compilation en live en enrengistrant le fichier).
- Linter GCC Pour le code c.
- Linter Java pour le code java
- Linter Ocaml

Norme de programmation

Généralités

Chaque fichier doit commencer dès la première ligne par le header 42

```
/* ************************************************************************** */
/*                                                                            */
/*                                                        :::      ::::::::   */
/*   BinarySearchTree.h                                 :+:      :+:    :+:   */
/*                                                    +:+ +:+         +:+     */
/*   By: NoobZik <rakib.hernandez@gmail.com>        +#+  +:+       +#+        */
/*                                                +#+#+#+#+#+   +#+           */
/*   Created: 2017/11/28 12:16:51 by NoobZik           #+#    #+#             */
/*   Updated: 2017/12/03 15:21:45 by NoobZik          ###   ########.fr       */
/*                                                                            */
/* ************************************************************************** */
```

Chaque ligne ne devra pas dépasser 80 colonne y compris tabulation + espace !

Une tabulation compte pour 2 espace et non pour 2 tabulation

Il est strictement interdit de mettre des commentaires dans un corps de fonction.

En revanche, chaque fonction doit être commenté de la facon suivante :

```
/**
 * [function name]
 * [description]
 * [parameters description]
 * @param [type] [description]
 * @return       [description]
 */
```

Il est strictement interdit utiliser les mots-clées suivantes :

- for (car c'est tomber de face)

- do - while
- switch case
- goto

L'indenation d'une fonction en générale se fait de la façon suivante :

```
// Une espace entre chaque mot clée
// Correct
int foo (char*) {

}
// Non correct
int foo() {

}

// Egalement non correct
int foo()
{

}
```

Dans le cas ou le prototypes d'une fonction dépasse 80 colonne, il est toléré de retirer un espace.

Dans ce cas, l'espace entre l'accolade ouvrante et la fin de parenthèse est prioritaire. Ensuite vient l'espace entre le type et la parenthèse ouvrante.

A l'interieur d'une fonction, seule les instructions

```
if else for while
```

comportent un espace entre l'instruction et la parenthèse ouvrante.

Accolade optionnel :

Dans le cas ou il y a une seule instruction, on n'est pas obligé de mettre des accolades.

Les variables devront être alignée verticalement.

```
// Correct
int     boujour;
float   boujourr;
double  poo;

// Incorrect
int bonjour;
float bonjourr;
double poo;
```

Lorsqu'une instruction avec des opérateurs dépasse 80 colonne, il faut revenir à la ligne au niveau de la premiere opération, avec le type d'opération devant.

```
// Correc
while (jesuisunevariable < jesuisuneautrevariable && bruh ^ okay
       || voila) {

}
// Incorrect
while (jesuisunevariable < jesuisuneautrevariable && bruh ^ okay ||
        voila) {

}
```

Dès qu'il y a une instruction if - else avec une seule instruction chacune, on doit utiliser la condition tertiaire.

```
// Incorrect
if (voila) {
  return true;
}
else {
  return false;
}

// correct
return (voila) ? true : false;
```

Cas particulier

Il existe un cas particulier auquel cette norme ne s'applique pas. Si l'instruction est *break* ou *continue*. Elle ne s'applique pas car ce sont des mot clées et non une instruction.