**Afin que votre code soit facilement compréhensible pour un programmeur extérieur, vous devrez respecter la norme de code suivante :**

* Le code ainsi que les commentaires doivent être écrits en anglais
* Les noms de fonctions/variables doivent décrire leur usage *(à quoi elles servent/ce qu’elles font)*
* Les noms de Classe, de Structure, d’Enum et de Namespace doivent commencer par une majuscule et respecter le PascalCase.
* Le type “enum class” doit être privilégié au type “enum” simple.
* Les variables membres de Classe/Structure doivent commencer par ”m\_” (m\_myVar)
* Les variables globales doivent commencer par ”g\_” (g\_myGlobal) et doivent être utilisées avec parcimonie. (max: 3)
* Les paramètres doivent commencer par ”p\_” (p\_myParam)
* Les variables statiques doivent commencer par ”s\_” (s\_myStatic)
* L’unité de mesure d’une variable doit être explicitement représentée (struct ou suffixe \_unit)
* Les fonctions devraient contenir 35 lignes au maximum mais la limite peut être augmentée si la situation le requiert
* Vos fonctions ne doivent pas prendre plus de 5 paramètres
* Les noms de variables et de fonction doivent commencer par une majuscule et respecter le PascalCase
* Une ligne ne doit pas dépasser 132 caractères *(penser à retourner à la ligne suivante)*

|  |
| --- |
| *// Wrong*  *if (playerPosition->x >= \_points[pointFollowed].x && playerPosition->x <= \_points[pointFollowed].x + && playerPosition->y >= \_points[pointFollowed].y && playerPosition->y <= \_points[pointFollowed].y && \_pointFollowed == \_points->size())*    *// Ok*  *if (playerPosition->x >= \_points[pointFollowed].x &&*  *playerPosition->x <= \_points[pointFollowed].x &&*  *playerPosition->y >= \_points[pointFollowed].y &&*  *playerPosition->y <= \_points[pointFollowed].y &&*  *pointFollowed == \_points->size())* |

* Chaque accolade doit être placée sur une ligne à part entière

|  |
| --- |
| int addition\_int(int a, int b) { *// Wrong*  return a + b; } *// Wrong*   int addition\_int(int a, int b)  { *// Ok*  return a + b;   } *// Ok* |

* Le type et le nom d’une fonction doivent être séparées par une tabulation

|  |
| --- |
| *// Wrong*  int fct1();  Vector\* fct2();  int fct1() {...}  Vector\* fct2() {...}  *// Ok*  int fct1();  Vector\* fct2(); |

* Les déclarations de variables doivent le plus souvent possible être regroupées

et alignées via des tabulations

|  |
| --- |
| int counter; *// Wrong*  char[128] array; *// Wrong*   int counter; *// Ok*  char[128] array; *// Ok* |

* Chaque instruction doit séparer, par un espace, les variables et les opérateurs utilisés

|  |
| --- |
| result=a+b; *// Wrong*   result = a + b; *// Ok* |

* Chaque mot-clé utilisant une condition doit avoir un espace entre lui et sa condition

|  |
| --- |
| if(condition) *// Wrong*  while(condition) *// Wrong*   if (condition) *// Ok*  while (condition) *// Ok* |

* Les valeurs ASCII doivent être représentées sous leur forme de caractère

|  |
| --- |
| char my\_char = 65; *// Wrong*  char my\_char = ‘A’; *// Ok* |

* Les alias préprocesseurs et variables constexpr doivent être écrits en majuscule

|  |
| --- |
| *// Wrong*  #define min\_pos\_x 0  #define max\_pos\_x 800  *// Ok*  #define MIN\_POS\_X 0  #define MAX\_POS\_X 800 |

* Les valeurs constantes doivent être représentées par des variables constantes ou des alias préprocesseurs

|  |
| --- |
| if (\_pos.x >= 0 && \_pos.x <= 800) *// Wrong*  *// Ok*  #define MIN\_POS\_X 0  #define MAX\_POS\_X 800  if (\_pos.x >= MIN\_POS\_X && \_pos.x <= MAX\_POS\_X)  *// Ok*  const min\_pos\_x = 0;  const max\_pos\_y = 800;  if (\_pos.x >= min\_pos\_x && \_pos.x <= max\_pos\_y) |

* Les constantes d’un énumérateur doivent être écrites en majuscule

|  |
| --- |
| *// Wrong*  enum Type  {  alpha,  beta  }  *// Ok*  enum class Type  {  ALPHA,  BETA  } |

* Une ligne vide doit séparer les includes et les fonctions

|  |
| --- |
| #include <stdlib.h>  **int** **main**() {  **return** 0; } |

* Une ligne vide doit séparer les directives préprocesseurs du reste

|  |
| --- |
| #ifndef \_\_MY\_PUT\_CHAR\_\_ #define \_\_MY\_PUT\_CHAR\_\_  **void** **my\_put\_char**(**char** c);  #endif |

* Les structures ne devraient pas avoir de membres privés
* Les structures ne devraient jamais hériter d’autres structures ou de classes
* Les headers des fonctions et classes doivent être commentés en suivant le format Doxygen