Bonnes pratiques de programmation en C

Sources utilisées

- https://emmanuel-delahaye.developpez.com/tutoriels/c/bonnes-pratiques-codage-c/
- https://users.ece.cmu.edu/~eno/coding/CCodingStandard.html#names
- https://stackoverflow.com/

- Eviter l'utilisation de variables globales afin d'éviter de potentiels comportements imprévisibles du programme si une fonction modifie la variable globale par exemple :

```
int g = 10;

void uneFonction(){
    g = 5;
}

void uneAutre(){
    printf("%d", g); //Si uneFonction() est appelée avant uneAutre() on aura 5 au
lieu de 10
}
```

- Respecter les conventions de nommage du projet/entreprise ou bien celles recommendées, mais ne pas changer de convention en cours de développement :
    - Variables -> camelCase
    - Fonctions -> camelCase
    - Structures -> PascalCase

- Commentaires au standard ISO-1990 ainsi que commentaire blocs :

```
/* Ceci est un commentaire normalisé */

/*
** Ceci est un commentaire bloc
*/
```

- Nommer les variables de façon claires et compréhensives pour éviter d'avoir à commenter le code pour les expliquer :

```
int v = 100; // 100 quoi, pommes, bananes ?

int vitesse = 100; // C'est mieux mais m/s ou km/h ?
```

```c
int vitesseKmh = 100; // OK
```

- Définir les constantes à l'aide de macros :

```c
#define PI 3.14
```

- Ne pas inclure les fichier .c, le faire par le biais du compilateur :

main.c

```c
#include "add.c" // NON
#include "add.h" // OUI

int main(){

    printf("%d", add(1, 2));

    return 0;
}
```

add.c

```c
int add(int a, int b){
    return a + b;
}
```

plutot faire :

```
$ gcc main.c add.c -o <nom_executable>
```

- Déclarer au prealable (*forward declaration*) les fonctions définies :

foo.c

```c
void salut(); // Déclaration préalable de salut()

int main(){

    salut();
```

```
    return 0;
}

void salut(){
    printf("Salut !");
}
```

- Eviter l'utilisation du mot clé **goto** qui rend le flux d'exécution du programme compliqué à suivre.
- Initialiser toutes les variables :

```
int a; // NON

int a = 0; // OUI
```

- Convention de nommage des header guards ou *gardes fous* pour protéger le programme des inclusions multiple de memes en-tetes :

```
#ifndef NOM_HEADER_GUARD_H
#define NOM_HEADER_GUARD_H

void foo();

#endif
```

- Commenter les fonctions avec de la docstring (selon l'IDE ou l'éditeur de code), certaines extensions comme Doxygen peuvent permettre de générer de la documentation :
    - https://doxygen.nl/