Bonnes pratiques

- Planifier son algorithme en pseudo-code
- Conventions de nommage
- Standard de programmation
- Commentaires et documentation du code

Planifier son algorithme en pseudo-code

- Un programme informatique ne peut pas faire d'opération complexe, mais il peut faire plusieurs opérations simple dans un temps record. Ce qui donne l'impression qu'il effectue des opérations complexe.
    - Programmer un algorithme informatique, c'est l'art de découper quelque chose de complexe en une série d'instruction simple.
- Ne commencez pas à coder sans réfléchir, analysez et planifiez les étapes du programme.
    - Déterminer les constantes
    - Déterminer les entrées et sorties
    - Quelles sont les étapes pour convertir les entrées en sorties.
- Un algorithme bien planifié doit être compréhensible au premier coup d'oeil, même sans l'exécuter.

don't

```
int t = Console.ReadLine();
Console.WriteLine($"{t/86400} days, {(t-(t/86400)*86400)/3600} hours, {(t-(t/86400)*86400-((t-(t/86400)*86400)/3600)*3600)/ 60} minutes, {t-(t/86400)*86400-((t-(t/86400)*86400)/3600)*3600-((t-(t/86400)*86400-((t-(t/86400)*86400)/3600)*3600)/60)*60} seconds");
```

do

```
const int SECONDS_IN_MINUTE = 60;
const int MINUTES_IN_HOUR = 60;
const int HOURS_IN_DAY = 24;

const int SECONDS_IN_HOUR = SECONDS_IN_MINUTE * MINUTES_IN_HOUR;
const int SECONDS_IN_DAY = SECONDS_IN_HOUR * HOURS_IN_DAY;

Console.WriteLine("Écrivez un nombre de seconde à convertir en notation JJ, HH, MM, SS");
int timeInSecond = Console.ReadLine();

int days = timeInSecond / SECONDS_IN_DAY;
timeInSecond = timeInSecond - days * SECONDS_IN_DAY;

int hours = timeInSecond / SECONDS_IN_HOUR;
timeInSecond = timeInSecond - hours * SECONDS_IN_HOUR;

int minutes = timeInSecond / SECONDS_IN_MINUTE;
timeInSecond = timeInSecond - minutes * SECONDS_IN_MINUTE;

Console.WriteLine($"{days} days, {hours} hours, {minutes} minutes, {timeInSecond} seconds");
```

Conventions de nommage

1. Utilisez des noms significatifs. *On ne le répètera jamais assez.*
    i. Soyez spécific
    ii. Pas de mot non nécéssaire
    iii. Pas d'abbréviation (excepté si elle est universellement reconnue. ex: km)
    iv. Pas de mot inventé
    v. Ne pas écrire le type de la variable dans le nom de celle-ci
2. Soyez constant à travers votre code.
3. Vous pouvez nommez vos variables en français ou en anglais, en autant que vous respectez le point #2.

do

```
string title;
string content;
int age;
const int AGE_OF_MAJORITY = 18;
bool isMajor;
```

don't

```
string titre; // Changement de langue
string ctnt; // abbréviation non universellement reconnue
int intAge; // le type est dans le nom de la variable
const int MAJORITY = 18; // Pas spécific
bool isTheAgeOver18YearsOld; // Mot non-nécéssaire.
```

Standard de programmation

- Utiliser la syntaxe lowerCamelCase pour nommer vos variables.
- Utiliser la syntaxe UPPER_SNAKE_CASE pour nommer vos constantes.
- Ajouter un espace de chaque côté des opérateurs arithmétique.
  - Une exception est faites pour les opérateurs `++` et `--` qui sont collés sur la variable puisqu'il n'y a pas de valeur après l'opérateur.
- Limiter le nombre de caractère par ligne à 120. Nous devons pouvoir voir tout le code, sur un écran conventionnel, sans barre de navigation horizontale.
- Pas d'espace ou de saut de ligne inutile.

do

```
int priceBeforeTaxe = 10;
const int MIN_VALUE = 100;
int result = a + b;
int a = 5;
a++;
```

don't

```
 int price-before-taxe = 10;
    const int minValue = 100 ;


  int result = a+b;

 int a=5;
a --;

  double a = 1.99;
```