

# 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.

# Déclaration de variable et de constante

- En pseudo-code on prend l'habitude de déclarer les variables et les constantes en haut de l'algorithme car cela nous force à réfléchir d'avance aux emplacements mémoires que nous aurons besoin.
- En programmation, les constantes devrait être déclaré dans le haut de l'algorithme, mais c'est permis de déclarer les variables au fur et à mesure qu'on les utilise.

# 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} 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 timeInSeconds = Console.ReadLine();

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

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

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

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

# Conventions de nommage

1. Utilisez des noms significatifs. *On ne le répètera jamais assez.*
  - i. Soyez spécifique
  - ii. Pas de mot non nécessaire
  - iii. Pas d'abréviation (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 cntt; // abbréviation non universellement reconnue  
int intAge; // le type est dans le nom de la variable  
const int MAJORITY = 18; // Pas spécifique  
bool isTheAgeOver18YearsOld; // Mot non-nécessaire.
```

# 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 faite 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.



# don't

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

```
int result = a+b;
```

```
int a=5;  
a --;
```

```
double a = 1.99;
```

# do

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