# ALGO : Construction d'algorithmes

- Objet : Méthode d'élaboration d'un algorithme
- Niveau requis :
  débutant, avisé
- Commentaires : *Pour les problèmes complexes, on divise le problème et on cherche un algorithme simple (procédure) pour chaque élément du programme ou algorithme globale*

## Introduction : la méthode de présentation d'une procédure

Une procédure est un morceau de programme en langage formel utilisable dans un gros programme.

**Elle contient donc un petit algorithme**.

Comment présenter une procédure (son petit algorithme) pour qu'elle soit réutilisable par n'importe qui et dans n'importe quel langage de programmation ?

La méthode suivante répond à ces deux exigences. Et décrit une méthode pour élaborer ces petits algorithmes que l'on appelle procédure.

La **méthode d'approche d'un problème d'algorithme doit se décomposer en six étapes** ; le résultat obtenu, de l'analyse du problème, c'est la procédure. La dernière étape de cette méthode donne l'algorithme de la procédure.

Imaginons un client, un jury, n'importe qui cherche une solution générale, codable en n'importe quel langage de programmation, car ce qu'on a comme programme pour une application informatique, n'est pas super. Par ce qu'on est un super concepteur, on a une idée de comment faire quelque chose, ou comment faire mieux quelque chose. Mais on est pas tout seul. Il faut être compris par les autres, et aussi permettre à son idée d'être choisie. Plus elle sera claire, parlante pour ceux des autres branches, électroniciens, clients, etc, etc, mieux cela voudra.

Pour ce faire, il faut savoir exposé ce qu'est le problème à résoudre AVANT de donner la solution qui permet de le résoudre.

Prenons pour exemple, une procédure qui compte les voyelles d'un mot comporte trois étapes.

Il s'agit distinguer l'exposé de la procédure de la procédure elle-même.

### Méthode de définition du problème et de sa solution

Les six étapes pour présenter une procédure :

1. poser le problème (la moulinette);
2. définition de la nature (type) des données ;
3. le jeu d'essai ;
4. définition de l'interface (ce que fait dans cette procédure en termes algorithmiques) ;
5. définition en termes algorithmique du programme de test ;

6. la procédure

- Les cinq première étape de la méthode d'approche définit ce qu'il faut faire (le quoi ?) ;
- La sixième étape définit comment on le fait (le comment).

**De ces six étapes il n'en restera que trois :**

1. Le jeu d'essai (explication schématique du traitement du problème);
2. le programme de validation de la procédure (programme du jeu d'essai qui utilise la procédure[1)] ;
3. le programme de la procédure (que l'on va coder quand il est au point)

# Rédiger une procédure qui compte les voyelles d'un mot.

On cherchera à exprimer la résolution en meta-langage d'une procédure qui permet de retourner le nombre de voyelles contenu dans une phrase.

On partira du principe que cette phrase est arrêtée par un point.

### 1) Un schéma de ce qu'il faut faire avec cette procédure

```
Entrée       |                            | Sortie
phrase ----> | procédure compte voyelle |----> nombre de voyelles
```

### 2) Définition des données : le structure de données

On définit les types.

- Phrase :

On crée un type phrase pour le type “chaîne de caractère”

```
chaine = tableau[N] de caractères
```

`chaine` : nom de ce nouveau type
`tableau` : mot clé `[N]` : taille du tableau ou indice maximal `de caractères` : nature ou type

### 3) Jeu d'essai

Il permet de penser aux cas limites pour définir plus en détail ce qu'il faut faire avec cette procédure. En tant que débutant, on pose que la condition d'arrêt est donnée. C'est-à-dire on n'envisagera pas le cas où l'utilisateur oublie le point à sa phrase.

```
  | phrase    |  nombre de voyelles  |
1 |"oiseau."  |   5                  |
2 | "."       |   0                  |
3 |"kwskwz."  |   0                  |
4 |"oie."     |   3                  |
```

1) L'utilisateur rentre un mot.
2) Et si l'utilisateur ne rentre que la condition d'arrêt, est-ce que doit contenir la procédure pour que le programme fonctionne ?
3) l'utilisateur est polonais ; pas de voyelle à son mot
4) Il choisit un mot contenant que des voyelles

### 4) Définition de l'interface

Quand le problème se compliquera, il pourra y avoir plusieurs interfaces.
Il s'agit là de réécrire ce qu'on a définit précédemment en termes algorithmiques.

Voilà comment le formalisme exige que soit déterminé l'interface de la procédure :

```
procédure CompteVoyelle(entrée phrase : chaine , sortie nbrVoyelle : entier)

// CompteVoyelle : Cette procédure permet de compter les voyelles d'une phrase.
// phrase : c'est la phrase analysée
// nbrVoyelle : nombre de voyelles (a e i o u) d'une phrase
```

**Explications**

Il s'agit de définir les paramètres de la procédure de manière formelle (imaginaire) ; on donnera les paramètres d'entrée et de sortie ; ces paramètres prendront d'autres noms dans **le programme de test** de la procédure.

`procédure CompteVoyelle(entrée phrase : chaine , sortie nbrVoyelle : entier`

`CompteVoyelle` : nom de la procédure ; la syntaxe du nom sera repris **à l'identique** dans l'algorithme final pour **l'appel** (utilisation) de la procédure.

`phrase` : nom de la variable d'entrée, dans cette “notice”, **dans cette “notice“**
`chaine` : nom du type créé (un tableau nommé chaine) : ré-utilisé (c'est une fonction) par le programme de test de la procédure.

`nbrVoyelle` : nom de la variable de sortie, **dans cette “notice”**

`entier` : type de la variable chaine (elle est d'un type existant).

### 5) Le programme (en langage algorithmique) d'essai

Là encore on définit le QUOI[2], et pas encore le COMMENT[3].

```
CONSTANTES

N = 43 // taille maximum d'une phrase
STOP = '.' // Caractère terminateur

TYPE
chaine = tableau[N] de caractère

VARIABLES
Texte : chaine // texte à analyser
nbV : entier // nombre de voyelles du texte

PROCÉDURE
CompteVoyelle
/*On colle là ce qu'on a rédigé précédemment, "la notice" de la procédure*/

  début
       écrire ("Donnez votre texte.")
       lire (Texte)
       comptevoyelle(texte , nbV)
  fin
```

On voit que ce programme de test utilise la notice de la procédure **CompteVoyelle** est utilise le type `chaine` avec des variables nommées différemment. (Chacun est maître chez soi !).

C'est le programme que le concepteur lit pour lui-même est vérifié que sa procédure fonctionne.

### 6) Le code de la procédure

C'est le programme de la procédure.
On rédige cette fois en termes algorithmiques ce que définissait la notice de la procédure à l'étape n°4, en langage imaginaire. C'est ce programme que le codeur en C ; C# ; C++ ; Pascal ; Ada ... lira et traduira dans le langage de programmation qui est le sien.

```
INTERFACE
procédure CompteVoyelle(entrée phrase : chaine , sortie nbrVoyelle : entier)

// CompteVoyelle : Cette procédure permet de compter les voyelles d'une
phrase.
```

```
// phrase : c'est la phrase analysée
// nbrVoyelle : nombre de voyelles (a e i o u) d'une phrase

VARIABLES

i : entier

  début

       i := 1 // indice de parcours
       nbVoyelle := 0 // nombre de voyelle cherché dans la phrase

       TQ  phrase[i] <> STOP  faire

            // Traiter un caractère

              SI ((phrase[1] = 'a') OU (phrase[1] = 'e') OU (phrase[1] =
'i') OU (phrase[1] = 'o')
                  OU (phrase[1] = 'u') OU (phrase[1] = 'y'))

                ALORS
                nbVoyelle := nbVoyelle + 1

               FINSI

          i := i + 1

       FinTQ
```

1)
une procédure comme une fonction est un sous-programme. C'est un sous-programme qui ne renvoie pas de valeur mais qui peut éventuellement modifier la valeur de ses paramètres
2)
ce qu'il faut faire
3)
comment cette procédure fonctionne