De la spécification à l'algorithme (partie 2)

Sébastien Jean

IUT de Valence
Département Informatique

v1.0, 1er novembre 2025

IUT Valence
Université Grenoble Alpes

Pseudo-code : forme générale d'un algorithme (rappels)

- Syntaxe :

```
FONCTION nom (paramètres) : type du résultat
    (déclarations des variables et constantes)
    (instructions)
FIN FONCTION
```

- N.B. : l'indentation facilite la lecture

IUT Valence
Université Grenoble Alpes

Pseudo-code : déclarations des variables et constantes (rappels)

- Syntaxe :

```
VARIABLE somme : entier

CONSTANTE MAX : entier (42)
```

IUT Valence
Université Grenoble Alpes

Pseudo-code : affectation d'une variable, appel de fonction (rappels)

- Syntaxe :

```
somme ← 42

taille ← LIRE_ENTIER()

AFFICHER("le message")
```

Pseudo-code : opérateurs arithmétiques (rappels)

- Syntaxe :

```
somme ← - somme          // Opposé

somme ← somme + 1        // Addition
somme ← somme - 1        // Soustraction
somme ← somme * 3        // Multiplication
x ← somme / 2            // Division (résultat réel)

somme ← somme div 2      // Division entière
somme ← somme mod 2      // Reste de la division entière (modulo)
```

IUT Valence Université Grenoble Alpes

Pseudo-code : opérateurs logiques (rappels)

- Syntaxe :

```
est_pair ← VRAI
est_pair ← FAUX

est_pair ← NON est_pair

est_pair ← est_pair OU VRAI
est_pair ← est_pair ET VRAI
est_pair ← est_pair OUEX VRAI // OU Exclusif
```

- N.B. : l'expression utilisant l'opérateur logique est de type booléen

IUT Valence
Université Grenoble Alpes

Pseudo-code : opérateurs relationnels (rappels)

- Syntaxe :

```
condition ← somme ≤ 2
condition ← somme < 2
condition ← somme = 2
condition ← somme ≠ 2
condition ← somme ≥ 2
condition ← somme > 2
```

- N.B. : l'expression utilisant l'opérateur relationnel est de type booléen

IUT Valence
Université Grenoble Alpes

Pseudo-code : retour de résultat (rappels)

- Syntaxe :

```
RETOURNER 1

RETOURNER "Abitbol"

RETOURNER x

RETOURNER a mod 2
```

- N.B. : plus **aucune instruction ne peut être exécutée** après un `RETOURNER` (sortie de la fonction en exportant le résultat)

IUT Valence
Université Grenoble Alpes

Pseudo-code : structure de contrôle SI (rappels)

- Syntaxe :

```
SI est_pair = VRAI ALORS
  AFFICHER("pair")

SINON
  AFFICHER("impair")

FIN SI
```

- Exprime l'alternative
- N.B. : `SINON` est optionnel

IUT Valence
Université Grenoble Alpes

Pseudo-code : structure de contrôle SELON (rappels)

- Syntaxe :

```
SELON numero
 0, 1 :
    AFFICHER("cas 1")
    APPEL1(numero)

 8, 9 :
   AFFICHER("cas 2")
   APPEL2(numero)

 AUTREMENT :
    AFFICHER("Interdit")

FIN SELON
```

- Exprime l'alternative (à plus de 2 cas)
- N.B. : `AUTREMENT` est optionnel

IUT Valence
Université Grenoble Alpes

Pseudo-code : structure de contrôle POUR

- Syntaxe :

```
POUR compteur DE 1 A 7 AVEC UN PAS DE 2

  AFFICHER_ENTIER(compteur)

FIN POUR
```

- **Itération** pour un nombre donné de tours
- Nécessite l'utilisation d'une variable
  - La valeur de départ (`DE ...`) est affectée à la variable avant d'entrer la première fois dans la boucle
  - Les instructions du corps de la boucle s'exécutent si la valeur de la variable est inférieure ou égale à la valeur de fin (`A ...`)
  - A la fin de chaque tour de boucle la valeur du **pas** est ajoutée à celle de la variable (`AVEC UN PAS DE ...`)

IUT Valence Université Grenoble Alpes

Pseudo-code : structure de contrôle POUR

- Quel est l'affichage produit ?
- Que vaut `compteur` à la fin ?

IUT Valence
Université Grenoble Alpes

Pseudo-code : structure de contrôle POUR

Corps de la fonction de calcul de la factorielle

Spécification du calcul de la factorielle

- **Donnée d'entrée** : n, entier
- **Donnée de sortie** : r, entier
- **Pré-condition** : $n \geq 0$
- **Post-condition** : $r = n! = 1 \times 2 \times \ldots \times (n-1) \times n$

Signature de la fonction factorielle

- **factorielle** (n : entier) : entier

Corps de la fonction

Corps de la fonction de calcul de la factorielle

```
FONCTION factorielle (n : entier) : entier

  VARIABLE resultat : entier
  VARIABLE i : entier

  resultat ← 1

  POUR i DE 1 A n PAR PAS DE 1

    resultat ← resultat * i

  FIN POUR

  RETOURNER resultat

FIN FONCTION
```

IUT Valence
Université Grenoble Alpes

Pseudo-code : structure de contrôle TANT QUE

- Syntaxe :

```
compteur ← 0

TANT QUE compteur mod 2 = 0

  AFFICHER(compteur)
  compteur ← compteur + 1

FIN TANT QUE
```

- **Itération** conditionnelle
- La condition de poursuite est évaluée avant chaque tour de boucle

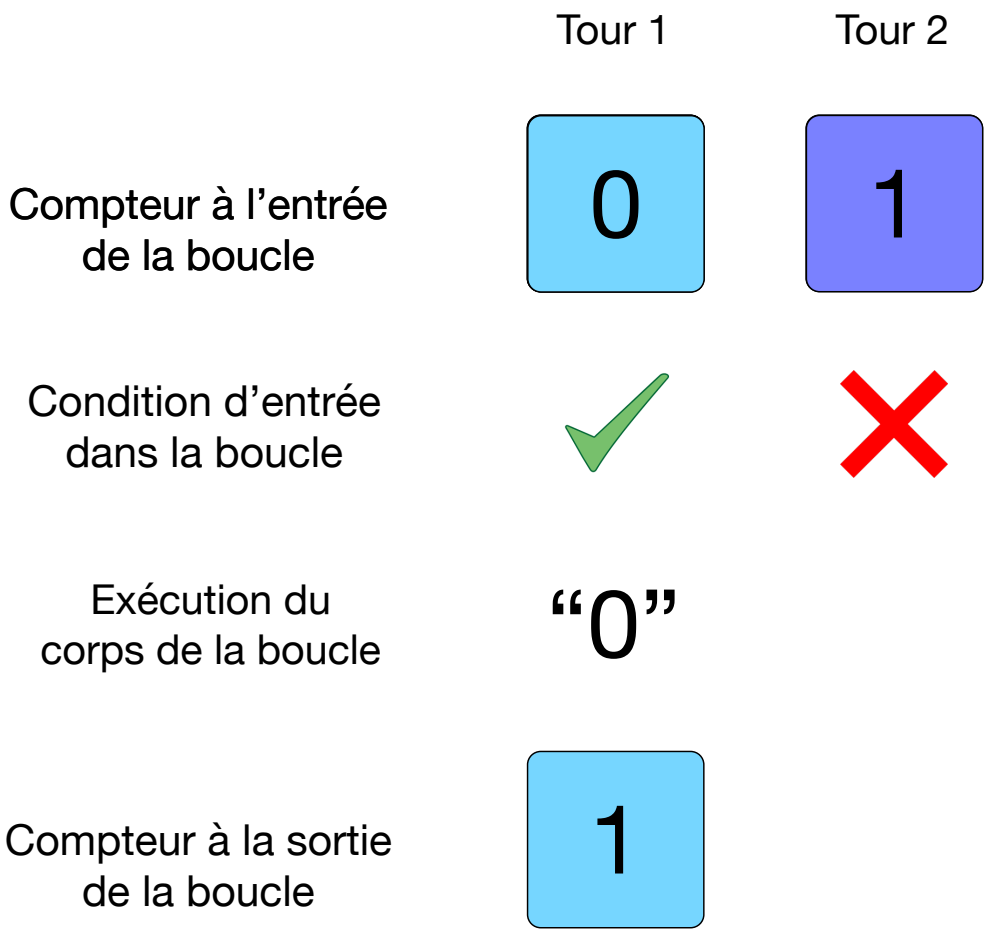
IUT Valence Université Grenoble Alpes

Pseudo-code : structure de contrôle `TANT QUE`

- Quel est l'affichage produit ?
- Que vaut `compteur` à la fin ?

IUT Valence
Université Grenoble Alpes

Pseudo-code : structure de contrôle `TANT QUE`



Corps de la fonction de calcul de la factorielle

- L'algorithme peut-t'il se réécrire avec un `TANT QUE` ?

IUT Valence
Université Grenoble Alpes

Corps de la fonction de calcul de la factorielle

```
FONCTION factorielle (n : entier) : entier

    VARIABLE resultat : entier
    VARIABLE i : entier

    resultat ← 1
    i ← 1

    TANT QUE i ≤ n

        resultat ← resultat * i
        i ← i + 1

    FIN TANT QUE

    RETOURNER resultat

FIN FONCTION
```

IUT Valence
Université Grenoble Alpes

Pseudo-code : structure de contrôle REPETER ... JUSQU'A

- Syntaxe :

```
compteur ← 0

REPETER

  AFFICHER(compteur)
  compteur ← compteur + 1

JUSQU'A compteur mod 2 = 0
```

- **Itération** conditionnelle
- La condition de sortie est évaluée après chaque tour de boucle

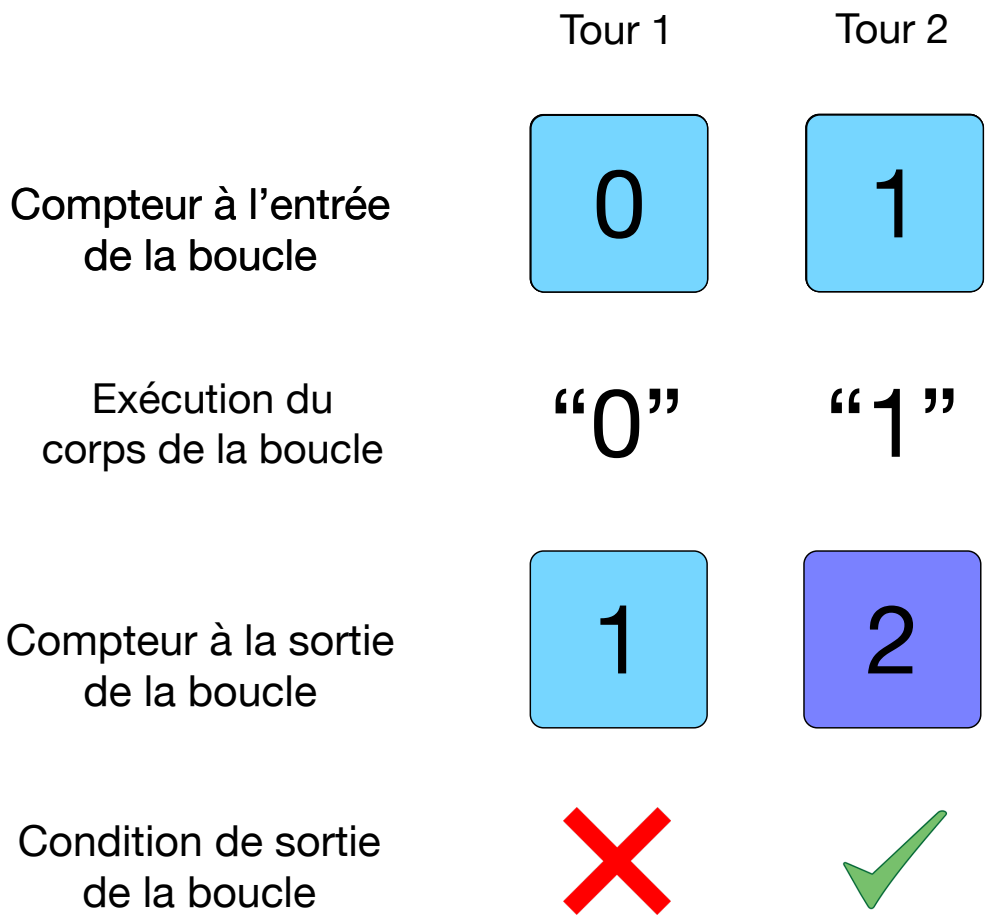
IUT Valence
Université Grenoble Alpes

Pseudo-code : structure de contrôle `REPETER ... JUSQU'A`

- Quel est l'affichage produit ?
- Que vaut `compteur` à la fin ?

IUT Valence
Université Grenoble Alpes

Pseudo-code : structure de contrôle `REPETER ... JUSQU'A`



IUT Valence Université Grenoble Alpes

Fin !

IUT Valence
Université Grenoble Alpes