8

Preuve d'un algorithme

8.1 Compétences exigées

— Objectifs —

Les capacités évaluées dans cette partie de la formation sont :

- justifier qu'une itération (ou boucle) produit l'effet attendu au moyen d'un invariant,
- démontrer qu'une boucle se termine effectivement.

8.2 Position du problème

Les itérations permettent d'écrire des programmes qui exécutent plusieurs fois les mêmes instructions. Ce sont les seules instructions qui nous permettent d'écrire un programme dont le temps d'exécution est arbitrairement long, ou même infini. Par exemple, l'algorithme suivant ne s'arrête jamais :

```
1: TANT_QUE 1 = 1 FAIRE
2:     DEBUT_TANT_QUE
3:     ...
4:     FIN_TANT_QUE
```

Algorithme 28 : boucle infinie 1

Un autre exemple :

```
1: INITIALISATION
2: n ← 0
3: TANT_QUE n ⩾ 0 FAIRE
4:     DEBUT_TANT_QUE
5:     n ← n + 1
6:     FIN_TANT_QUE
```

Algorithme 29 : boucle infinie 2

Avec les boucles, il y a deux questions qui surgissent :

- La **terminaison** : la boucle se termine-t-elle ? Sous quelles conditions ?
- La **validité** : que calcule la boucle en question ?

Pour les boucles `POUR...DE...A`, le problème de terminaison ne se pose pas. Il n'en est pas de même pour les boucles `TANT QUE`.

8.3 Terminaison

8.3.1 Méthode

Pour prouver qu'un algorithme faisant intervenir une boucle conditionnelle se termine, on cherche à identifier une expression à valeurs entières positives strictement décroissante et minorée : mathématiquement, cela se traduit par :

$$\exists m \in \mathbb{R} \text{ tq } \forall n \in \mathbb{N}, m \leqslant u_n$$

Cette expression est appelée **variant de boucle**.

8.3.2 Exemple

On peut par exemple considérer l'algorithme suivant qui effectue la division euclidienne d'un entier a par un autre entier b :

```
1:  VARIABLES
2:  a, b : entiers
3:  SORTIES
4:  AFFICHER le quotient q et le reste r
5:  DEBUT_ALGORITHME
6:      INITIALISATION
7:      q ← 0
8:      r ← a
9:      TANT_QUE r ⩾ b FAIRE
10:         DEBUT_TANT_QUE
11:         r ← r − b
12:         q ← q + 1
13:         FIN_TANT_QUE
14:     AFFICHER quotient q et reste r
15: FIN_ALGORITHME
```

Algorithme 30 : division euclidienne

Cet algorithme permet de calculer le quotient q et le reste r de la division de a par b.

— Principe —

À chaque itération de la boucle, r est minoré par b et décroît strictement puisque b est strictement positif. Ceci assure la terminaison de la boucle.

En sortie de boucle, on sait que $r < b$ mais comme r est positif, r est en réalité compris dans l'intervalle $[0, b-1]$

8.4 Les invariants de boucle

Il est indispensable qu'un algorithme se termine mais on souhaite également qu'il effectue la tâche voulue !

C'est ce qu'on vérifie au moyen d'un **invariant de boucle**.

— Remarque —

Lorsqu'il est indiqué dans la structure de l'algorithme, l'invariant de boucle est en principe noté entre accolades, en tant que commentaire.

8.4.1 Définition

— Invariant de boucle —

On appelle invariant de boucle une proposition vérifiant les conditions suivantes :

- La proposition est vraie avant l'entrée dans la boucle.
- Si cette proposition est vraie au début d'une itération, elle reste vraie à la fin de l'itération, donc au début de l'itération suivante.

Dans ce cas, cette proposition sera vraie à la sortie de la boucle.

Il s'agit en fait d'un raisonnement par récurrence.

8.4.2 Exemple

Reprenons l'exemple de la division euclidienne de a par b.

- Initiation : avant la boucle, $q = 0$ et $r = a$. On a donc $a = b \times q + r$.
- Hérédité : supposons que $a = b \times q + r$ au début d'une itération. Notons q' et r' les valeurs de q et r à la fin de l'itération.
  On a $q' = q + 1$ et $r' = r - b$ de sorte que $b \times q' + r' = b \times (q + 1) + (r - b) = b \times q + r = a$. En fin d'itération, on a alors $a = b \times q + r$ avec $r \in [0, b - 1]$ (condition de sortie), ce qui assure que q et r sont bien le quotient et le reste de la division euclidienne de a par b.

$\{a = b \times q + r\}$ constitue donc un invariant de boucle.

On peut le montrer en remplissant le tableau suivant, en prenant par exemple $a = 17$ et $b = 4$:

Itération	q	r	$b \times q + r$
Avant la boucle			
première itération			
deuxième itération			
troisième itération			
quatrième itération			

8.4.3 Implémentation en *Python*

⇒ **Activité 8. 42**

1. Proposez une fonction permettant de calculer le résultat de la division euclidienne de a par b.
2. Introduisez dans la fonction un test permettant de savoir si l'invariant de boucle est vérifié à chaque itération.

Le résultat :

```
Division euclidienne de 17 par 4 :
Invariant vérifié
Invariant vérifié
Invariant vérifié
Invariant vérifié
(4, 1)
```

8.5 Applications

8.5.1 Algorithme d'Euclide

8.5.1.1 Algorithme

Soit l'algorithme suivant qui calcule le plus grand commun diviseur (pgcd) n de deux entiers a et b :

```
1:  VARIABLES
2:  a, b : entiers
3:  n, p : entiers
4:  SORTIES
5:  AFFICHER le pgcd n de a et b
6:  DEBUT_ALGORITHME
7:      INITIALISATION
8:      n ← a
9:      p ← b
10:     TANT_QUE p ≠ 0 FAIRE
11:         DEBUT_TANT_QUE
12:         n, p ← p, n%p
13:         # n%p est le reste de n/p
14:         FIN_TANT_QUE
15:     AFFICHER le pgcd n
16: FIN_ALGORITHME
```

Algorithme 31 : algorithme d'Euclide

8.5.1.2 Terminaison

⇒ **Activité 8.43**

Vérifiez la terminaison de cet algorithme.

8.5.1.3 Invariant de boucle

⇒ **Activité 8.44**

Vérifiez que $\{pgcd(a,b) = pgcd(n,p)\}$ constitue un invariant de boucle.

8.5.1.4 Implémentation en *Python*

⇒ **Activité 8.45**

1. Proposez une fonction permettant de calculer le pgcd de a et b de façon itérative.
2. Proposez une fonction permettant de calculer le pgcd de a et b de façon récursive.
3. Introduisez dans la deuxième fonction un test permettant de savoir si l'invariant de boucle est vérifié à chaque itération. Vous pourrez utiliser la première fonction pour vérifier cet invariant dans la deuxième fonction.

Le résultat :

```
pgcd de 56 et 42 :
14
Invariant vérifié
Invariant vérifié
14
```

8.5.2 Puissance

⇒ **Activité 8.46**

Soit n un entier strictement positif et a un réel. Considérons l'algorithme suivant qui calcule $s = a^n$:

```
1:  VARIABLES
2:  a, r, s : flottants
3:  n, N : entiers
4:  SORTIES
5:  AFFICHER a^n
6:  DEBUT_ALGORITHME
7:      INITIALISATION
8:      s ← a
9:      N ← n
10:     r ← 1
11:     TANT_QUE N > 0 FAIRE
12:         DEBUT_TANT_QUE
13:         SI N%2 = 0 # (N pair) ALORS
14:             DEBUT_SI
15:             s ← s × s
16:             N ← N/2
17:             FIN_SI
18:         SINON
19:             DEBUT_SINON
20:             r ← r × s
21:             N ← N − 1
22:             FIN_SINON
23:         FIN_TANT_QUE
24:     AFFICHER s
25: FIN_ALGORITHME
```

Algorithme 32 : puissance

Montrez que l'algorithme se termine et vérifiez que l'invariant de boucle est $\{s^N \times r = a^n\}$.

8.5.3 Factorielle

⇒ **Activité 8.47**

Considérons l'algorithme suivant qui calcule la factorielle res d'un entier n :

```
1:  VARIABLES
2:  n, i, res : entiers
3:  SORTIES
4:  AFFICHER res
5:  DEBUT_ALGORITHME
6:  |  INITIALISATION
7:  |  i ← 0
8:  |  res ← 1
9:  |  TANT_QUE i < n FAIRE
10: |  |  DEBUT_TANT_QUE
11: |  |  i ← i + 1
12: |  |  res ← res × i
13: |  |  FIN_TANT_QUE
14: |  AFFICHER res
15: FIN_ALGORITHME
```

Algorithme 33 : factorielle

Montrez que l'algorithme se termine et vérifiez que $\{res = i!\}$ est un invariant de boucle.