15 Preuves de programmes

◇ Faire la preuve d'un programme consiste à démontrer deux points :
- sa *terminaison* : le fait que ce programme s'arrête après un nombre fini d'étapes. Avec ce que l'on a vu jusqu'à présent, le seul point qui peut empêcher la terminaison d'un programme est la présence de boucles *while* mal construites ;
- sa *correction* : le fait que les résultats produits par le programme sont bien ceux attendus.

Pour simplifier, on s'intéresse dans la suite à faire la preuve de *fonctions* PYTHON.

◇ Résultat clé pour la *terminaison* d'une boucle *while* : il n'existe pas de suite infinie, à valeurs entières et positives qui soit strictement décroissante. Conséquence : si on arrive à mettre en évidence une quantité φ, à valeurs entières et positives, telles que φ décroit strictement à chaque tour de la boucle, alors la boucle termine nécessairement.

◇ Résultat clé pour la *correction* d'une boucle *while* : si on arrive à mettre en évidente une propriété P telle que
- P est vraie au point ① ;
- *si* P est vraie au point ②, *alors* P est vraie au point ③,

alors P sera vraie au point ④. Une telle propriété P est appelée un *invariant de boucle*. Noter l'analogie avec le principe de récurrence en mathématiques.

```
①
tant que (condition)
    ②
    Instructions...
    ③
fin tant que
④
```

Exemple Python. Somme des termes d'une liste L (si la liste est vide, alors la somme est nulle). On note $n = \text{len}(L)$. Le principe du programme :
- On a une variable s qui vaut initialement 0 ;
- On parcourt tous les termes de la liste et chaque terme est ajouté à s.

Ainsi, une fois que toute la liste a été parcourue, s contient la somme de tous les termes de la liste.

```
def somme(L):
    s = 0
    i = 0
    # Point 1
    while i<len(L):
        # Point 2
        s = s+L[i]
        i = i+1
        # Point 3
    # Point 4
    return s
```

◇ Pour montrer que cette fonction termine, on remarque que la quantité $n-i$ est entière, positive (car dans la boucle, on a toujours $n-i \geqslant 0$) et strictement décroissante à chaque tour de la boucle (puisque i augmente à chaque fois de 1). Comme il n'existe pas de suite infinie qui soit à valeurs entières, positives et strictement décroissante, la boucle *while* ne peut être effectuée qu'un nombre fini de fois, donc la fonction termine.

◇ Pour montrer la correction de cette fonction, on démontre que la propriété :

$$P : s = L[0] + \cdots + L[i-1]$$

est un invariant de boucle. Tout d'abord, P est vraie au point 1 (indiqué par un commentaire). Supposons P vraie au point 2, c'est à dire $s = L[0] + \cdots + L[i-1]$. Notons s' la nouvelle valeur de s au point 3 et i' la nouvelle valeur de i au point 3, on a $i' = i+1$ et :

$$s' = s + L[i] = L[0] + \cdots + L[i-1] + L[i] = L[0] + \cdots + L[i'-1]$$

Par conséquent P est vraie au point 3. La propriété P est bien un invariant de boucle, elle est donc vraie au point 4 (juste à la sortie de la boucle). Au point 4, on a $i = n = \text{len}(L)$, donc $s = L[0] + \cdots + L[n-1]$. Par conséquent la fonction est correcte.

Exemple Python. Déterminer le maximum des éléments d'une liste de longueur $n \geqslant 1$. Le principe du programme :

- On a une variable m qui vaut initialement $L[0]$;
- On parcourt tout les termes $L[1], \ldots, L[n-1]$ et à chaque fois que l'on rencontre un terme strictement supérieur à m, on donne à m la valeur de ce terme.

Ainsi, une fois que toute la liste a été parcourue, m contient le plus grand terme de la liste.

```
def maximum(L):
    m = L[0]
    i = 1
    # Point 1
    while i<len(L):
        # Point 2
        if L[i]>m:
            m = L[i]
        i = i+1
        # Point 3
    # Point 4
    return m
```

◇ Comme pour la fonction précédente, la quantité $n-i$ est entière, positive et strictement décroissante à chaque tour de la boucle, donc la fonction termine.

◇ Pour montrer la correction de cette fonction, on démontre que la propriété :

$$P : m = \max(L[0], \ldots, L[i-1])$$

est un invariant de boucle. En effet, cette propriété est vraie au point 1 (indiqué par un commentaire). Supposons P vraie au point 2, c'est à dire $m = \max(L[0], \cdots, L[i-1])$. Notons m' la nouvelle valeur de m au point 3 et i' la nouvelle valeur de i au point 3 ($i' = i+1$), on distingue deux cas. Si $L[i] > m$, alors $m' = L[i]$ et :

$$\max(L[0], \ldots, L[i'-1]) = \max(\underbrace{L[0], \ldots, L[i-1]}_{\max : m}, \underbrace{L[i]}_{>m}) = L[i] = m'$$

Si $L[i] \leqslant m$, alors $m' = m$ et :

$$\max(L[0], \ldots, L[i'-1]) = \max(\underbrace{L[0], \ldots, L[i-1]}_{\max : m}, \underbrace{L[i]}_{\leqslant m}) = m = m'$$

Dans tous les cas, P est vraie au point 3. La propriété P est bien un invariant de boucle, elle est donc vraie au point 4. Au point 4, on a $i = n = \text{len}(L)$, donc $m = \max(L[0], \cdots, L[n-1])$. Par conséquent la fonction est correcte.