Informatique

9

Algorithmique

Résumé

<table>
<tr><td colspan="2">Terminaison</td><td>Prouver l'existence d'un variant de boucle évoluant de manière strictement monotone dans un espace fini
Attention : dans le cas d'un «<code>for i in L:</code>», modifier la taille de L dans la boucle peut conduire à une non terminaison</td></tr>
<tr><td rowspan="4">Correction</td><td>for</td><td>Notons $\mathcal{P}(k)$ la propriété après l'itération pour $i = k$
Corriger une boucle « for » consiste à réaliser 3 étapes :
- Initialisation : $\mathcal{P}$ vraie à la première itération : $\mathcal{P}(0)$
- Transmission : Si $\mathcal{P}$ vraie à l'itération i, $\mathcal{P}$ vraie à l'itération $i+1$: $\mathcal{P}(i) \Rightarrow \mathcal{P}(i+1)$
- Sortie : A la dernière itération n, $\mathcal{P}$ doit permettre de démontrer que le résultat obtenu est le résultat attendu : $\mathcal{P}(n) \Rightarrow Résultat$
Remarque : la première itération correspond à $i = 0$</td></tr>
<tr><td>while</td><td>Notons $\mathcal{P}(i)$ la propriété après la i eme itération.
Corriger une boucle « while » consiste à réaliser 3 étapes :
- Initialisation : $\mathcal{P}$ vraie avant la première itération : $\mathcal{P}(0)$
- Transmission : Si $\mathcal{P}$ vraie avant l'itération i, $\mathcal{P}$ vraie après: $\mathcal{P}(i) \Rightarrow \mathcal{P}(i+1)$
- Sortie : Après la dernière itération n (lorsque la condition devient fausse pour la première fois), $\mathcal{P}(n)$ doit permettre de démontrer que le résultat obtenu est le résultat attendu : $\mathcal{P}(n) \Rightarrow Résultat$
Remarque : $i = 0$ correspond à l'état initial</td></tr>
<tr><td>Attention</td><td>Rédaction en 5 étapes : Algorithme – Propriété – Initialisation – Transmission – Sortie</td></tr>
<tr><td>Outils de vérification</td><td><code>assert booleen, "message si False"</code>
<code>L = []</code>
Typage des fonctions (indicatif): <code>def f(x:float)->float:</code>
<code>try:
    x = L.pop()
except:
    print("L est vide")</code></td></tr>
<tr><td>Complexité en temps</td><td>$O(f(n))$</td><td>$f(x) = O(g(x))$ signifie $\exists N \in \mathbb{R}^+ \ \& \ A \in \mathbb{R} \ / \forall x > N, \left|\frac{f(x)}{g(x)}\right| < A$
Le nombre d'opérations d'un algo s'écrit sous la forme $af(n) + b$
Si une complexité est indépendante de n, on dit $O(1)$
Si $n = 100$ et la complexité est en $O(n)$, NE PAS écrire $O(100)$!!!</td></tr>
</table>

$f(n)$	$C(n)$	Preuve	$f(n)$	$C(n)$	Preuve
10	$O(1)$	$\frac{10}{1} = 10$	n	$O(n^3)$	$\frac{n}{n^3} \underset{+\infty}{\sim} \frac{1}{n^2} \underset{+\infty}{\to} 0$ A éviter toutefois
$2n$	$O(n)$	$\frac{2n}{n} = 2$	$2^n + 3^n$	$O(3^n)$	$\frac{2^n + 3^n}{3^n} = \left(\frac{2}{3}\right)^n + 1 \underset{+\infty}{\sim} 1$
$2n^2$	$O(n^2)$	$\frac{2n^2}{n^2} = 2$	2^{n+1}	$O(2^n)$	$\frac{2^{n+1}}{2^n} = 2$
$2n + 3n^2$	$O(n^2)$	$\frac{2n + 3n^2}{n^2} \underset{+\infty}{\sim} \frac{3n^2}{n^2} = 3$	$\log n$	$O(\ln n)$	$\frac{\log n}{\ln n} = \frac{\frac{\ln n}{\ln 2}}{\ln n} = \frac{1}{\ln 2}$