Université Paul Sabatier
L2 - S4 - Printemps 2005

Programmation impérative et langage C

feuille de **TD n° 6 :** Preuves de programmes

◇ Exercice **1** : Vérifier le programme suivant qui prétend calculer le terme de rang $N > 0$ de la suite de fibonacci définie par la relation de récurrence suivante :

$$F_n = \begin{cases} 1 \text{ si } n = 0 \text{ ou } n = 1 \\ F_{n-2} + F_{n-1} \text{ si } n > 1 \end{cases}$$

```
{
⋮
scanf("%d", &N);
/*PE : N ≥ 1 */
i = 1;
a = 1;
b = 1;
/* INV : (F_i = b) ∧ (F_{i-1} = a) ∧ (0 ≤ i ≤ N) */
while (i < N)
    {
    b = b + a;
    a = b − a;
    i = i + 1;
    }
/* PS : F_N = b */
printf("%d", b);
}
```

◇ Exercice **2** : Soit $N \in \mathbb{N}^*$, et $B(0..N-1)$ un tableau d'entiers croissants (non strictement). Pour un tableau $B(0..N-1)$,

– on appelle sous-tableau tout tableau extrait $B(d, f)$ avec $0 \leq d \leq f \leq N-1$.
– On appelle plateau de B tout sous-tableau dont les cases contiennent toutes la même valeur.

On souhaite écrire un programme qui retourne dans p la longueur du plus long plateau de B.

1. Ecrire la spécification de ce programme.
2. Prouver que le programme suivant est solution :

$\vdots$
$/ * Q * /$
$j = 1;$
$p = 1;$
$/ * INV * /$
while $(j \neq N)$
{
$/ * INV \wedge (j \neq N) * /$
if $(B[j-p] == B[j])\ p = p + 1;$
$j = j + 1;$
}
$/ * INV \wedge (j = N) * /$
$/ * R * /$

où :
- Q est le prédicat d'entrée
- l'invariant est :

$$\begin{aligned} INV = \ & (0 \leq p \leq j \leq N) \\ & \wedge (\exists K : 0 \leq K \leq j - p : B[K] = B[K + p - 1]) \\ & \wedge (\forall K : 0 \leq K \leq j - p - 1 \rightarrow (B[K] \neq B[K + p])) \end{aligned}$$

- R est le prédicat de sortie

$\diamond$ Exercice **3** : Puissance russe : vérifier le programme suivant qui, pour 2 entiers positifs ou nuls A et B, calcule la puissance B^{eme} de A.

$\vdots$
$PE : / * (A \geq 0) \wedge (B \geq 0)\ * /$
$x = A;$
$y = B;$
$z = 1;$
INV1: $/ * (z \times x^y = A^B) \wedge (y \geq 0) * /$
while $(y \neq 0\ )$
{
$/ * (z \times x^y = A^B) \wedge (y \geq 0) * /$
INV2 = INV1$\wedge cond1 = / * (z \times x^y = A^B) \wedge (y > 0) * /$
while $(y\ \%\ 2 = 0)$
}
$/ * (z \times x^y = A^B) \wedge (y > 0) \wedge (y\ \%\ 2 = 0) * /$
$y = y/2;$
$x = x * x;$
$/ * (z \times x^y = A^B) \wedge (y > 0) * /$
}
$/ * (z \times x^y = A^B) \wedge (y > 0) \wedge (y\ \%\ 2 \neq 0) * /$
$y = y - 1;$
$z = z * x;$
$/ * (z \times x^y = A^B) \wedge (y \geq 0) * /$
}
$PS : / * (z = A^B) \wedge (y = 0) * /$

Preuves de programmes : Cas particulier de deux boucles imbriquées.

On veut prouver le programme qui suit :

$/ * PE * /$
init1 :
$/ * INV1 * /$
while $cond1$
    {
    $/ * INV2 * / = / * INV1 \wedge cond1 * /$
    init2 ;
    $/ * INV2 * /$
    while $cond2$
            {
            $/ * INV2 \wedge cond2 * /$
            corps2 ;
            $/ * INV2 * /$
            }
    $/ * INV2 \wedge \neg cond2 * /$
    fin2 ;
    $/ * INV1 * /$
    }
$/ * INV1 \wedge \neg cond1 * /$
fin1 ;
$/ * PS * /$

On note "corps1" la séquence des instructions situées dans la boucle 1.
La preuve nécessite les étapes suivantes :

– Preuve partielle de la boucle 1 :

Étape 1 : $/ * PE * / \rightarrow$ pfp("init1", $INV1$)

Étape 2 : Parce qu'on ne peut exprimer pfp("corps1", $INV1$), on est contraint de décomposer cette étape en 5 sous-étapes :
– Preuve partielle de la boucle 2 :

2.1 $INV1 \wedge cond1 \rightarrow$ pfp("init2",$INV2$)

2.2 $INV2 \wedge cond2 \rightarrow$ pfp("corps2", $INV2$)

2.3 $INV2 \wedge \neg cond2 \rightarrow$ pfp("fin2", $INV1$)

Aprés l'éxécution de corps1 on a donc bien INV1 et l'étape 2 est démontrée sous réserve de la bonne terminaison de la boucle 2 :

– Bonne terminaison de la boucle 2 :
On choisit une variante de la boucle 2 : f2, il faut alors montrer les 2 implications suivantes :

2.4 $INV2 \wedge cond2 \rightarrow (f2 > 0)$

2.5 $INV2 \wedge cond2 \wedge (T = f2) \rightarrow$ pfp("corps2", $(T > f2)$)

Étape 3 : $INV1 \wedge \neg cond1 \rightarrow$ pfp("fin1",PS)

– Bonne terminaison de la boucle 1 :
On choisit une variante de la boucle 1 : f1, il reste enfin à montrer les 2 implications suivantes :
Étape 4 : $INV1 \wedge cond1 \rightarrow (f1 > 0)$
Étape 5 : $INV1 \wedge cond1 \wedge (T = f1) \rightarrow$ pfp("corps1", $(T > f1)$)
mais pfp("corps1", $(T > f1)$) ne peut être évaluée directement, on se contentera de justifier en français pourquoi systématiquement la variante1 décroit strictement à chaque passage dans la boucle 1.

◇ ◇ ◇◇ ◇ ◇◇