

# TD : preuves et terminaisons

## 1 Terminaison

**Exercice 1.** 1. On considère la fonction

```
1 void bouclefor0(int n){
2   for (int i = n; i >= 0; i--)
3     printf("i=%d\n",i);
4 }
```

Etudier la terminaison.

2. On considère la fonction

```
1 void bouclefor1(int n){
2   for (int i = 0; i < n; i++)
3     printf("i=%d\n",i);
4 }
```

Etudier la terminaison.

3. On considère la fonction

```
1 void bouclefor2(int n){
2   for (int i = 0; i < n; i++){
3     i=i-2;
4     printf("i=%d\n",i);
5   }
6 }
```

Etudier la terminaison.

4. On considère la fonction

```
1 int g(int x);
2
3 void bouclefor3(int n){
4   for (int i = 0; i < n; i++)
5     printf("i=%d\n",g(i));
6 }
```

Etudier la terminaison.

**Exercice 2.** On considère la fonction

```

1 void f(int p){// précondition : p supposé positif
2   int c=0;
3   while (p > 0){
4     if (c == 0){
5       p = p - 2;
6       c = 1;
7     }
8     else{
9       p = p + 1;
10      c = 0;
11     }
12   }
13 }
```

On numérote les passages dans la boucle à partir de 1. Le passage 0 désigne ce qui se passe avant la boucle. Les variables du programme sont indicées selon le tour de boucle :  $p_0$  désigne la valeur de `p` juste avant la boucle ;  $p_i$  la valeur de `p` à la fin du passage  $i$ .

On cherche un variant  $v_i$  sous forme de combinaison linéaire entière de  $p_i$  et  $c_i$ . On veut donc trouver  $a, b$  dans  $\mathbb{N}$  telle que la suite

$$(ap_i + bc_i)_{i \in \mathbb{N}} \text{ soit strictement décroissante.}$$

1. Expliquer pourquoi aucune des variables de ce programme n'est un variant.
2. Ecrire une fonction

```

1 int cl(int a, int b, int p0, int res[100])
2
```

qui prend en paramètre un nombre  $p_0$  et remplit le tableau `res` (assez gros) avec les CL  $ap_i + bc_i$  jusqu'à trouver  $p = 0$ . Elle renvoie le nombre de passages dans la boucle.

3. Ecrire une fonction

```

1 bool check (int n, int tab[n])
```

qui prend en paramètre un nombre  $n$  et un tableau et indique si ce tableau est rangé par ordre décroissant strictement.

4. Ecrire une fonction

```

1 bool find (int ab[2])
```

qui lance `cl` jusqu'à trouver un couple  $a, b$  telle que la suite  $(ap_i + bc_i)_i$  soit strictement décroissante.

5. A partir de ce résultat établir la preuve de terminaison de `f`.

**Exercice 3.** Ecrire une fonction qui prend en paramètre un flottant  $u_0$  puis détermine le rang du dernier terme strictement positif de la suite récurrente définie par  $u_{n+1} = \frac{1}{2}u_n - 3n$ .

Montrer que le programme termine.

La remarque selon laquelle on ne peut pas trouver de suite infinie d'entiers positifs strictement décroissante est valable pour toute puissance de  $\mathbb{N}$  munie de l'ordre lexicographique.

Par exemple, il n'existe pas de suite  $((x_n, y_n, z_n))_{n \in \mathbb{N}}$  strictement décroissante à termes dans  $\mathbb{N}^3$ . On peut donc étendre la notion de variant aux puissances de  $\mathbb{N}$  et s'en servir pour établir la terminaison d'algorithmes.

**Exercice 4.** Un enfant très ordonné entreprend de ranger ses petits trains. Chaque train est formé d'une suite d'éléments, indistinctement wagons ou locomotives. L'enfant suit la procédure suivante :

- Prendre un train non rangé.
  - S'il contient un seul élément le ranger dans la caisse.
  - sinon, le séparer en deux trains plus petits, qui sont remis dans les trains non rangés.
- Recommencer tant qu'il reste des trains à ranger.

L'enfant ne suit aucune stratégie pour choisir le prochain train à traiter, ni pour sélectionner l'endroit où couper un train en deux.

Ce processus va-t-il s'arrêter ?

**Exercice 5.** On considère le programme suivant

---

```
1 fonction Aleatoire(int x, int y, int m):
2   Entree : x,y entiers , m > 0
3   Sortie : ce que sont devenus x,y
4   debut
5   tant_que (x ≠ 0 ET y ≠ 0)
6     si y > 0:
7       y ← y - 1;
8     sinon si x > 0:
9       x ← x - 1;
10      y=randint(-m,m);
11     sinon :
12       x ← -x - 1;
13   renvoyer x,y
14 fin
```

---

L'appel `randint(-m,m)` renvoie un nombre aléatoire entre  $-m$  et  $m$ .

Etudier la terminaison du programme.

## 2 Correction

**Exercice 6.** Montrer que le programme suivant d'affichage du reste et du quotient de la division euclidienne est correct :

```

1 void diviser(int n, int d){
2   assert (n>=0 && d>0);
3   int q =0, r=n;
4   while (r >= d){
5     q = q + 1;
6     r = r - d;
7   }
8   printf ("le quotient de %d par %d est %d; le reste vaut %d\n",n,d,q,r);
9 }
10

```

**Exercice 7.** Montrer que le programme de calcul du produit de 2 nombres est correct

```

1 void mult(int a, int b){
2   assert (b>0);
3   y=b, r = 0;
4   while (y > 0){
5     r = r+a;
6     y = y-1;
7   }
8   return r;}
9

```

**Exercice 8.** On donne le code suivant, dit d'« exponentiation rapide » :

```

1 int expoRap(int a,int n){
2   //précondition n entier >= 0
3   assert (n>=0);
4   int p=1,b=a,m=n;
5   while (m>0){
6     if (m % 2 == 1)
7       p=p*b;
8     b=b*b;
9     m = m/2;
10  }
11  return p;
12 }

```

Il est basé sur la propriété suivante :

$$a^b = \begin{cases} a^k \times a^k & \text{si } b = 2k \\ a \times a^k \times a^k & \text{si } b = 2k + 1 \end{cases}$$

Montrer sa terminaison et sa correction.

**Exercice 9.** On considère la fonction de McCarthy :

```

1 || let rec f n =
2 ||   if n > 100
3 ||   then n-10
4 ||   else f (f (n+11));;

```

1. Montrer que  $f$  termine pour tout  $x \geq 100$
2. En raisonnant par récurrence descendante, montrer que  $f(n)$  termine pour tout  $n \leq 100$  et déterminer sa valeur.

**Exercice 10.** On cherche ici le pgcd de deux entiers positifs. Cette recherche se fait sur le constat suivant : le PGCD de deux nombres n'est pas changé si on remplace le plus grand d'entre eux par leur différence. Autrement dit, si  $a \geq b$ , alors  $\text{pgcd}(a, b) = \text{pgcd}(b, a - b)$ . Par exemple, le PGCD de 252 et 105 vaut 21, mais c'est aussi le PGCD de  $252 - 105 = 147$  et 105.

Par convention  $\text{pgcd}(n, 0) = n$  pour tout entier naturel  $n$ .

1. Montrer que si  $a > b$ ,  $\text{pgcd}(a, b) = \text{pgcd}(b, r)$  où  $r$  est le reste de la division euclidienne de  $a$  par  $b$ .
2. L'algorithme d'Euclide sur deux nombres entiers positifs  $a$  et  $b$  avec  $a > b \geq 0$  procède comme suit :
  - Si  $b = 0$ , l'algorithme termine et rend la valeur  $a$  ;
  - Sinon, l'algorithme calcule le reste  $r$  de la division euclidienne de  $a$  par  $b$ , puis recommence en remplaçant  $a$  par  $b$  et  $b$  par le reste de la division de  $b$  par  $a$ .

Écrire une fonction `int euclide(int a, int b)` qui calcule le pgcd de deux nombres entiers positifs  $a, b$ .

3. Etablir que votre programme termine pour tout couple d'entiers positifs.
4. Etablir la correction de l'algorithme.