Module 8 : Les sous-programmes

Situations

Problème de lessive

Najla, Douja et Zohra ont fait leurs lessives aujourd'hui. Or, **Najla** fait sa lessive tous les 3 jours, **Douja** tous les 4 jours et **Zohra** tous les 6 jours.

Figure 1, Lessive

Questions

1. Combien passera-t-il de temps avant que les trois femmes ne refassent leurs lessives le même jour ?
2. En supposant que :
   - **Najla** fait la lessive tous les `lf1` jours. Avec `lf1 > 0`
   - **Douja** fait la lessive tous les `lf2` jours. Avec `lf2 > 0`
   - **Zohra** fait la lessive tous les `lf3` jours. Avec `lf3 > 0`

   Déterminer quand les trois femmes referons leurs lessives le même jour ?
3. Ecrire l'algorithme d'un programme pour résoudre ce problème.

Solutions

1. On pourra déterminer graphiquement le temps requis pour voir les trois femmes faire leurs lessives le même jour. Et ce en utilisant l'échelle temporelle suivante :

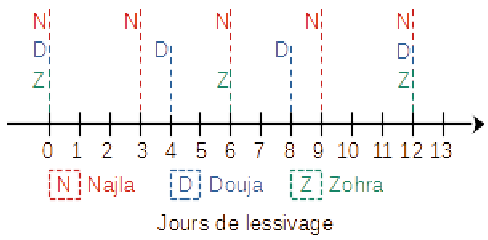

Figure 2, Jours de lessive

   On en déduit qu'il faudra attendre **12 jours**.
2. On remarque que le temps requis pour voir les trois femmes faire leurs lessives dans une même journée peut être calculé en utilisant la formule suivante :

   ppcm(3, 4, 6) = ppcm(3, ppcm(4, 6)) = ppcm(ppcm(3, 4), 6)= 12

   Plus généralement le **temps requis pour voir les femmes faire leurs lessives la même journée** est :

   ltf = ppcm(lf1, lf2, lf3) = ppcm(ppcm(lf1, lf2), lf3) = ppcm(lf1, ppcm(lf2, lf3))

   PPCM = Plus Petit Commun Multiple, c'est le plus petit nombre qui est multiple des trois nombres
3. L'algorithme de cette situation peut être écrit de deux façons :

1ère façon : méthode classique

```
Algorithme Lessive
Début
  // Saisie du jour de lessive de chaque femme
  Répéter
    Ecrire("Lessive femme 1 ? ") ; Lire(lf1)
```

Objet	Type/Nature
lf1, lf2, lf3, ltf, p1, p2	entier

```
  Jusqu'à (lf1 > 0)
  Répéter
    Ecrire("Lessive femme 2 ? ") ; Lire(lf2)
  Jusqu'à (lf2 > 0)
  Répéter
    Ecrire("Lessive femme 3 ? ") ; Lire(lf3)
  Jusqu'à (lf3 > 0)
  // Après combien de jours la 2ème et la 3ème femme font leurs lessives
  p1 ← lf2
  TantQue p1 mod lf3 ≠ 0 Faire
    p1 ← p1 + lf2
  Fin TantQue
  // Après combien de jours les 3 femmes font leurs lessives
  p2 ← lf1
  TantQue p2 mod p1 ≠ 0 Faire
    p2 ← p2 + lf1
  Fin TantQue
  ltf ← p2
  // Affichage du résultat
  Ecrire("Toutes les femmes feront leurs lessives après", ltf, "jours")
Fin
```

2ème façon : méthode modulaire

Programme Principal

```
Algorithme Lessive2
Début
  // Saisie du jour de lessive de chaque femme
  saisie("Lessive femme 1 ? ", lf1)
  saisie("Lessive femme 2 ? ", lf2)
  saisie("Lessive femme 3 ? ", lf3)
  // Après combien de jours les 3 femmes font leurs lessives
  ltf ← ppcm(lf1, ppcm(lf2, lf3))
  // Affichage du résultat
  Ecrire("Toutes les femmes feront leurs lessives après", ltf, "jours")
Fin
```

TDOG

Objet	Type/Nature
lf1, lf2, lf3, ltf	entier

Procédure saisie

```
procédure saisie(msg: chaîne, @nj: entier)
Début
  Répéter
    Ecrire(msg)
    Lire(nj)
  Jusqu'à (nj > 0)
Fin
```

TDOL

Objet	Type/Nature
-	-

Fonction ppcm

```
fonction ppcm(a, b: entier):entier
Début
  p ← a
  TantQue (p mod b ≠ 0) Faire
    p ← p + a
  Fin TantQue
  retourner p
Fin
```

TDOL

Objet	Type/Nature
p	entier

Fraction irréductible

En mathématiques, une **fraction est irréductible** s'il n'existe pas de fraction égale ayant des termes plus petits. Autrement dit, une fraction irréductible **ne peut pas être simplifiée**.

Théorème

Soient **a** un entier et **b** un entier naturel non nul. Alors $\frac{a}{b}$ est irréductible si et seulement si **a** et **b** sont premiers entre eux.

Exemple

La fraction $\frac{12}{20}$ n'est pas irréductible car 12 et 20 sont des multiples de 4 : $\frac{12}{20} = \frac{3 \times 4}{5 \times 4} = \frac{3}{5}$ (simplification par 4). On peut aussi écrire $\frac{12}{20} = \frac{12 : 4}{20 : 4} = \frac{3}{5}$.
La fraction $\frac{3}{5}$ est irréductible car 1 est le seul entier positif qui divise à la fois 3 et 5.

Méthode de simplification

Pour réduire directement une fraction, il suffit de **diviser le numérateur et le dénominateur par leur plus grand commun diviseur**. D'après le lemme de Gauss, cette forme réduite est unique.

Exemple

Pour réduire la fraction $\frac{42}{390}$, on calcule $\mathrm{PGCD}(42, 390) = 6$ puis on simplifie par 6 : $\frac{42}{390} = \frac{6 \times 7}{6 \times 65} = \frac{7}{65}$.

Problème

On souhaite écrire l'algorithme d'un **programme modulaire** qui calcule la somme de deux fractions :

$$\frac{p1}{q1} + \frac{p2}{q2} = \frac{p1 \times q2 + p2 \times q1}{q1 \times q2} = \frac{p3}{q3} = \frac{\frac{p3}{pgcd(p3,q3)}}{\frac{q4}{pgcd(p3,q3)}} = \frac{ps}{qs}$$

*Figure 3, Somme de deux fractions avec : p1, p2, ps ∈ ℤ et q1, q2, qs ∈ ℤ**

Solutions

Solution non modulaire

```
Algorithme Somme_Fraction
Début
  // Saisie de p1 et q1
  Ecrire("Fraction : p1 / q1")
  Ecrire("p1 ? ") ; Lire(p1)
  Répéter
    Ecrire("q1 ? ") ; Lire(q1)
  Jusqu'à q1 ≠ 0
  // Saisie de p2 et q2
  Ecrire("Fraction : p2 / q2")
  Ecrire("p2 ? ") ; Lire(p2)
  Répéter
    Ecrire("q2 ? ") ; Lire(q2)
  Jusqu'à q2 ≠ 0
  // Calcul de la somme des deux fractions
  ps ← p1 * q2 + p2 * q1
  qs ← q1 * q2
  // Calcul du PGCD(ps, qs)
  a ← ps
  b ← qs
  TantQue b ≠ 0 Faire
    r ← a mod b
    a ← b
    b ← r
  Fin TantQue
  // Simplification
```

TDO

Objet	Type/Nature
p1, q1, p2, q2, ps, qs, a, b, r	entier

```
  ps ← ps div a
  qs ← qs div a
  // Affichage
  Ecrire(p1, "/", q1, "+", p2, "/", q2, "=", ps, "/", qs)
Fin
```

Solution modulaire

Programme Principal

```
Algorithme Somme_Fraction1
Début
  // Saisie de p1 et q1 / p2 et q2
  saisie_frac("Fraction : p1 / q1", p1, q1)
  saisie_frac("Fraction : p2 / q2", p2, q2)
  // Calcul de la somme des deux fractions
  ps, qs ← somme_frac(p1, q1, p2, q2)
  // Simplification
  ps, qs ← simp_frac(ps, qs)
  // Affichage
  Ecrire(p1, "/", q1, "+", p2, "/", q2, "=", ps, "/", qs)
Fin
```

TDOG

Objet	Type/Nature
p1, q1, p2, q2, ps, qs	entier

Procédure saisie_frac

```
Procédure saisie_frac(msg: chaîne, @p, @q: entier)
Début
  Ecrire(msg)
  Ecrire("Numérateur ? ") ; Lire(p)
  Répéter
    Ecrire("Dénominateur ? ") ; Lire(q)
  Jusqu'à q ≠ 0
Fin
```

TDOL

Objet	Type/Nature
-	-

Fonction somme_frac

```
Fonction somme_frac(a1, b1, a2, b2: entier):(entier, entier)
Début
  a ← a1 * b2 + a2 * b1
  b ← b1 * b2
  retourner a, b
Fin
```

TDOL

Objet	Type/Nature
a, b	entier

Fonction simp_frac

```
Fonction simp_frac(a, b: entier):(entier, entier)
Début
  p ← pgcd(a, b)
  retourner a div p, b div p
Fin
```

TDOL

Objet	Type/Nature
p	entier

Fonction pgcd

```
Fonction pgcd(a, b: entier):entier
Début
  TantQue b ≠ 0 Faire
    r ← a mod b
    a ← b
    b ← r
  Fin TantQue
  retourner a
Fin
```

TDOL

Objet	Type/Nature
r	entier