TD 1 – soutien – itérations

1. Nombre parfait

Problème : dire si un nombre entier plus grand que 1 est parfait ou non (s'il est égal à la somme de ses diviseurs ; on ne compte pas comme diviseur le nombre lui-même. Par exemple, 6 est parfait 6=3+2+1).
Analyse : Lire un entier n >1. On va chercher tous les diviseurs de n et les additionner dans une variable S qu'on comparera avec n à la fin.
Pour obtenir tous les diviseurs de n une seule fois, il suffit de parcourir tous les entiers de 1 à n div 2.

```
Programme Parfait
Var N : entier
Début
lire(N)
S ← 0
Pour i←1 à N div 2 Inc +1 Faire
      Si i mod N = 0
            Alors S←S+i
      FinSi
FinPour
Si S=N
      Alors écrire(N, 'est parfait')
      Sinon écrire(N, 'n''est pas parfait')
FinSi
Fin
```

2. Multiplication Egyptienne

Problème : Multiplier deux entiers positifs par la multiplication égyptienne. Les primitives acceptées sont : addition, multiplication par 2, division par 2.
Analyse sur un exemple : 25*19 = 25*18+25 = 50*9+25 = 50*8+50+25 = 100*4+75 = 200*2+75 = 400*1+75 = 400*0+400+75 = 475

X*Y = X*(Y-1)+X si Y est impair
X*Y = X*2*Y/2 si Y est pair et ≠ 0

Nécessité d'une variable S pour faire les sommes intermédiaires.

```
Programme Multiplication-Egyptienne
Var A, B, X, Y, S : entier
Début
lire(A)
lire(B)
S ← 0
X ← A
Y ← B
TantQue Y≠0 Faire
      Si (Y mod 2 ≠ 0)
            Alors Y←Y-1
                  S←S+X
            Sinon X←X*2
                  Y←Y div 2
      FinSi
FinTantQue
ecrire(S)
Fin
```

Améliorations :
Le nombre d'itération dépend uniquement de Y. Vu que la multiplication est commutative, Y doit être le minimum de A et de B.
Quand Y est impair, son état suivant est pair. Il est donc inutile de tester la parité de Y après un passage par l'état impair.

```
Programme Multiplication-Egyptienne
Var A, B, X, Y, S : entier
Début
lire(A)
lire(B)
S ← 0
Si A<B Alors Y ← A
            X ← B
       Sinon X ← A
            Y ← B
FinSi
TantQue Y≠0 Faire
      Si (Y mod 2 ≠ 0)
            Alors Y←Y-1
                  S←S+X
      FinSi
      X←X*2
      Y←Y div 2
FinTantQue
ecrire(S)
Fin
```

3. Racine carrée

L'algorithme de Newton est basé sur la convergence vers √a de la suite suivante :

$$\begin{cases} u_0 = 1 \\ u_{n+1} = 1/2 * (u_n + a/u_n) \end{cases}$$

```
Programme racine-carrée
Const epsilon = 0,001
Var a, u, usuiv : réel
Début
 Ecrire ('Entrez le réel positif dont vous voulez calculer la racine carrée : ')
 lire(a) {l'utilisateur est docile}
   usuiv ← 1
 Répéter
   u ← usuiv
   usuiv ← (u + a/u) / 2
 Jusqu'à val-abs(u-usuiv)<=epsilon
 Ecrire('√', a, ' = ',u)
Fin
```

4. Suite

$$\begin{cases} u_0 = 1/3 \\ u_{n+1} = 4 * u_n - 1 \end{cases}$$

```
Programme suite
Var i, n : entier
    u : réel
Début
 Ecrire ('Entrez n (entier positif) : ')
 lire(n) {l'utilisateur est docile}
   u ← 1/3
 Pour i←1 à n inc +1 Faire
   u ← 4*u-1
 FinPour
 Ecrire('le ', n,'ième élément de la suite est ', u)
Fin
```