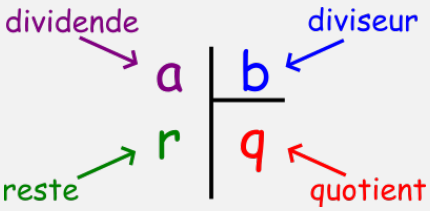
LIFAPI – TD 3 : Encore des algos…

Objectifs : Approfondir les notions vues dans le TD précédent (boucles, conditions, structures de données, entrées / sorties, …)

La division euclidienne (ou division entière) et le modulo



Le reste est également appelé **modulo** => a modulo b = r
Ici, a, b, q et r sont des entiers.

Remarque : en affectant dans un entier le résultat d'un calcul réel, on récupère la partie entière du résultat.

1. Écrire un algorithme qui teste si la somme des valeurs de 2 dés est paire ou impaire et affiche le résultat à l'écran.

```
Début
  de1, de2, somme : entiers
   somme ← de1 + de2
   Si somme modulo 2 = 0 Alors
      Afficher ("La somme est paire.")
   Sinon
      Afficher ("La somme est impaire.")
   Fin Si
Fin
```

2. Écrire un algorithme qui calcule la somme des chiffres qui composent un nombre choisi par l'utilisateur.
   Exemple : valeur saisie : 1234 ➔ résultat : 10 (= 1 + 2 + 3 + 4)

```
Début
  Variables : nbre, sdc, i : entier
  Afficher ('Donnez une valeur')
  Saisir(valeur)
  sdc ← 0
  Tant que (valeur > 0) faire
       sdc ← sdc + (valeur  modulo 10)
       valeur ← valeur / 10
  Fin tant que
  Afficher ('La somme des chiffres qui composent ', nbre,' est : ', sdc)
Fin
```

3. Écrire un algorithme qui calcule les racines réelles (si elles existent) d'un polynôme du second degré décrit par 3 coefficients réels a, b et c (a non nul). Les solutions seront affichées à l'écran.

```
Début
    a,b,c,sol1, sol2, delta : réel
    Afficher('Entrez les 3 coefficients du polynôme')
    Saisir(a,b,c)
    delta ← b*b -4*a*c // attention à l'écriture de l'expression !
    Si (delta < 0)   Alors afficher ('pas de racines réelles')
                     Sinon  Si (delta = 0)    Alors   sol1← -b /(2*a)
                                                      Afficher ('une racine double :', sol1)
                                                Sinon   sol1 ← (-b + sqrt(delta)) / (2*a)
                                                        sol2 ← (-b -  sqrt(delta)) / (2*a)
                                                        Afficher(sol1,sol2)
                                  Fin Si
    Fin Si
    Fin
```

4. Écrire l'algorithme d'un programme permettant de vérifier si un entier est premier ou non. Rappel : un nombre premier est un nombre qui n'est divisible que par 1 et par lui-même.

```
Première version avec un pour mais sans s'arrêter dès qu'on a trouvé un diviseur
Début
  N : entier
  afficher ("Donnez un entier")
  saisir (N)
  EstPremier ← Vrai
  Pour i de 2 à N - 1 Faire // rmq on peut interrompre la boucle à N/2 voir √n
     Si N modulo i = 0 Alors
        EstPremier ← Faux
     Fin Si
  Fin Pour
  Si EstPremier Alors
     Afficher "Ce nombre est premier."
  Sinon
     Afficher "Ce nombre n'est pas premier."
  Fin Si
Fin
```

2ème version avec un tant qui permet d'optimiser l'algo

```
Début
      N : entier
      afficher ("Donnez un entier")
      saisir (N)
      EstPremier ← Vrai
      i ← 2
      Tant que i < N ET EstPremier = Vrai Faire
         Si N modulo i = 0 Alors
            EstPremier ← Faux
         Fin Si
         i ← i + 1
      Fin Tant que

      Si EstPremier Alors
         Afficher "Ce nombre est premier."
      Sinon
         Afficher "Ce nombre n'est pas premier."
      Fin Si
   Fin
```

5. Écrire un algorithme qui permet de générer une valeur aléatoire comprise entre 1 et 6.
   Outil : pour choisir un nombre aléatoire, on utilisera en algorithmique : aleatoire() qui retourne un entier compris entre 0 et MAX exclu, MAX étant une valeur très grande que l'on ne choisit pas.
   Généraliser la formule précédente pour obtenir une valeur aléatoire dans [a ; b].

```
Valeur entre 1 et 6
Début
  Val : entier
  Val ← aleatoire() modulo 6 +1
  Afficher (Val)
Fin
```

Le modulo donne le reste de la division entière par 6 donc une valeur comprise entre 0 et 5. Si on lui ajoute 1 on a bien une valeur comprise entre 1 et 6.

Formule générale

```
Début
  Val : entier
  Val ← aleatoire() modulo (b-a+1) + a
  Afficher (Val)
Fin
```

6. Écrire un algorithme permettant de trouver une valeur choisie aléatoirement par le programme. Le joueur disposera au maximum de 6 tentatives pour trouver cette valeur et le programme lui indiquera à chaque essai si sa valeur est trop grande ou trop petite.
   Outil : le programme choisira une valeur dans l'intervalle [10 ; 60] de la même manière qu'à la question précédente.

```
Variables : a_trouver, valeur, nb_essais : entiers
Début
  a_trouver ← aleatoire() modulo 51 + 10
  nb_essais ← 0
  Faire
    Afficher('Donnez une valeur')
    Saisir(valeur)
    Si (valeur > a_trouver)    Alors Afficher('trop grand')
                               Sinon   Si (valeur < a_trouver)
                                           Alors Afficher('trop petit')
                                       Fin si
    Fin si
    nb_essais ← nb_essais + 1
  Tant que ((valeur <> a_trouver) et (nb_essais < 6))
  Si (valeur = a_trouver)    Alors Afficher('gagné en ',nb_essais)
                             Sinon Afficher ('perdu trop d essais')
  Fin si
Fin
```