LIFAP1 – TD 2 : Algorithmes plus évolués

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

1. Écrire un algorithme permettant de calculer la somme des n premières puissances de 2.
   Exemple : valeur saisie : 6 ➔ résultat : 63 (= 1 + 2 + 4 + 8 + 16 + 32).
   Outil : 2^i s'écrira en algorithmique : puissance (2, i)

```
Début
  Variables : n, somme, i : entier
  Afficher('donnez la valeur de n')
  Saisir(n)
  somme←0
  Pour i allant de 0 à n-1 par pas de 1 faire
    somme← somme + puissance(2,i)
  Fin Pour
  Afficher (somme)
Fin
```

2. Écrire un algorithme permettant de calculer la somme des n premiers nombres impairs.
   Exemple : valeur saisie : 6 ➔ résultat : 36 (= 1 + 3 + 5 + 7 + 9 + 11)
   Quel lien pouvez-vous établir entre la valeur obtenue et le nombre n ?

```
Début
  Variables : n, somme, i : entier
  Afficher('donnez la valeur de n')
  Saisir(n)
  somme←0
  Pour i allant de 1 à 2*n par pas de 2 faire
    somme← somme + i
  Fin Pour
  Afficher (somme)
Fin
```

Exemple pour n=5
Somme = 1+3+5+7+9 = 25 = 5^2
Donc la somme des n premiers nombres impairs est égale au carré de n.

3. Écrire un algorithme permettant de lire 20 nombres entiers au clavier. Si le nombre x saisi est pair, on affiche la valeur (x / 2) sinon on affiche (3*x + 1). Attention, on ne mémorisera pas les 20 valeurs saisies.

```
Puisque l'on connaît le nombre de passages dans la boucle, on utilise la boucle pour :
Début
  Variables : nbre, i : entier
  Pour i allant de 1 à 20 par pas de 1 faire
        Afficher ('Entrez un nombre')
        Saisir(nbre)
        Si (nbre modulo 2) = 0      Alors    nbre ← nbre / 2
                                    Sinon    nbre ← 3*nbre + 1
        FinSi
        Afficher(nbre)
  FinPour
Fin
```

4. Écrire un algorithme qui calcule la moyenne de n valeurs saisies par l'utilisateur, n étant choisi préalablement par l'utilisateur.

```
Début
  Variables  valeur, nbre, i, somme: entier

  Afficher ('Donnez le nombre de valeurs')
  Saisir(nbre)
  somme ← 0
  Pour i allant de 1 à nbre par pas de 1 faire
        Afficher ('Entrez une valeur')
        Saisir(valeur)
        somme ← somme + valeur
  FinPour
  Afficher ('La moyenne est : ' , somme / nbre)
Fin
```

5. Écrire un algorithme qui teste si un entier choisi par l'utilisateur est multiple de 5 ou multiple de 7.

```
Début
Variables  valeur : entier

Afficher ('Donnez une valeur')
Saisir(valeur)
Si ((valeur modulo 5) = 0) ou ((valeur modulo 7) = 0)
      Alors Afficher (valeur, 'est multiple de 5 ou de 7')
      Sinon Afficher (valeur, 'n'est ni multiple de 5, ni multiple de 7')
FinSi
Fin
```

6. 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 > 10) faire
     sdc ← sdc + (valeur  modulo 10)
     valeur ← valeur / 10
  Fin tant que
  Afficher ('La somme des chiffres qui composent ', nbre,' est : ', sdc)
Fin
```

7. 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. Les solutions seront affichées à l'écran.

```
Variables   a,b,c : réels
sol1, sol2, delta : réel
Début
Afficher('Entrez les 3 coefficients du polynôme')
Saisir(a)
Saisir(b)
Saisir(c)
delta ← b*b -4*a*c
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

        Remarque : si a=b=0 alors on n'a pas un polynôme !
```

8. 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 : pour choisir un nombre aléatoire, on utilisera en algorithmique : aleatoire()

```
Variables : a_trouver, valeur, nb_essais : entiers
Début
  a_trouver ← aleatoire()
  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
```

Pour s'entraîner

1. Écrire l'algorithme d'un programme permettant de vérifier si un entier est premier ou non.
   Rappel : un nombre est premier s'il n'est divisible que par 1 et par lui-même.
   Exemple : premier(15) renverra 'faux'
   premier(13) renverra 'vrai'

```
variables locales     i : entier, est_premier : booléen
Début
    est_premier ← vrai
    Afficher ("Donnez un entier positif")
    Saisir(n)
    Pour i allant de 2 à racine(n) par pas de 1 faire
      Si (n modulo i ) = 0   Alors est_premier ← faux
      Fin si
    Fin pour
    Si est_premier Alors Afficher (n, " est un nombre premier")
                   Sinon Afficher (n, " n'est pas un nombre premier")
    fin si
Fin
```

Autre solution plus efficace (sortie de la boucle dès que l'on tombe sur un diviseur) :

```
variables locales     i : entier, est_premier : booléen
Début
     est_premier ← vrai
     Afficher ("Donnez un entier positif")
     Saisir(n)
     i ← 2
     Tant que ((est_premier) et (i <= racine(n))) faire
       Si (n modulo i ) = 0    Alors est_premier ← faux
       Fin si
       i ← i +1
     Fin pour
     Si est_premier Alors Afficher (n, " est un nombre premier")
                    Sinon Afficher (n, " n'est pas un nombre premier")
     fin si
Fin
```