Université Moulay Ismail **Année universitaire: 2022/2023**
Faculté des sciences de Meknès **Filière: SMIA/S2**
Département d'Informatique **Module: INFO2**

TD N°3 : Les structures répétitives (corrigés)

Exercice 1

1. Ecrire un algorithme qui demande un nombre de départ, et qui calcule la somme des entiers jusqu'à ce nombre. Par exemple, si l'on entre **5**, le programme doit calculer:

   1+ 2 + 3 + 4 + 5 = 15

```
Algorithme Somme;
Var
    N, i, Som: Entier;
Debut
    Ecrire("Entrez un nombre : ");
    Lire (N);
    Som←0;
    Pour i ← 1 à N faire
     Som ← Som + I;
    FinPour
    Ecrire ("La somme est : ", Som);
Fin
```

---

2. Ecrire un algorithme qui demande un nombre de départ, et qui ensuite affiche les dix nombres suivants. Par exemple, si l'utilisateur entre le nombre 17, le programme affichera les nombres de 18 à 27.

```
Algorithme NombresSuivants;
Var
    N, i: Entier;
Debut
    Ecrire ("Entrez un nombre : ");
    Lire (N);
    i ← 0;
    Ecrire ("Les 10 nombres suivants sont : ");
    TantQue (i < 10) faire
      i ← i + 1;
      Ecrire (N + i);
    FinTantQue
```

Fin

Exercice 2

Ecrire un algorithme qui demande un nombre compris entre 10 et 20, jusqu'à ce que la réponse conenne. En cas de réponse supérieure à 20, on fera apparaitre un message: **Plus petit!**, et inversement, **Plus grand !** si le nombre est inférieur à 10

```
Algorithme NombreSaisi;
      Var
            N :Entier;
      Debut
      N←0;
      Ecrire ("Entrez un nombre entre 10 et 20");
      TantQue (N < 10 OU N > 20) faire
        Lire (N);
        Si (N < 10) Alors
          Ecrire ("Plus grand !");
        SinonSi (N > 20) Alors
          Ecrire ("Plus petit !");
        FinSi
      FinTantQue
      Fin
```

Exercice 3

Ecrire un algorithme qui demande successivement 20 nombres à l'utilisateur, et qui lui dise ensuite quel était le plus grand parmi ces 20 nombres:

Entrez le nombre numéro 1: 12

Entrez le nombre numéro 2: 14

etc.

Entrez le nombre numéro 20: 6

Le plus grand de ces nombres est: 14

Modifiez ensuite l'algorithme pour que le programme affiche de surcroît en quelle position avait été saisie ce nombre:

C'était le nombre numéro 2

```
Algorithme Plusgrand1;
Var
      N, i, PG :Entier;
      Debut
      PG ← 0;
```

```
Pour i ← 1 à 20 faire
  Ecrire ("Entrez un nombre : ");
  Lire (N);
  Si (i = 1 OU N > PG) Alors
    PG ← N;
  FinSi
FinPour
Ecrire ("Le nombre le plus grand était : ", PG);
Fin
```

```
Algorithme Plusgrand2;
        Var
          N, i, PG, IPG :Entier;
          Debut
          PG ← 0;
          Pour i ← 1 à 20 faire
            Ecrire ("Entrez un nombre : ");
            Lire (N);
            Si (i = 1 ou N > PG) Alors
              PG ← N;
              IPG ← I;
            FinSi
          FinPour
          Ecrire ("Le nombre le plus grand était : ", PG);
          Ecrire ("Il a été saisi en position numéro ", IPG);
          Fin
```

Exercice 4

Ecrire un algorithme permettant de lire une suite de nombres réels saisis au clavier et d'afficher la somme et le nombre des éléments lus. Le dernier élément à lire pour stopper la saisie est ***Zéro***.

```
Algorithme  Suite_nombres;
        Var
                N, Som : réels;
                 i: entier;
        Début
        i← 0;
        Répeter
                Ecrire("entrer un  nombre: ");
                Lire(N);
                Som← Som+N;
                i ← i+1;
         Jusqu'à (N=0)
        Ecrire("la somme des nombres est :  ", Som);
        Ecrire("le nombre d'élément lus est :  ", i);
        Fin
```

Exercice 5

Calculer a^b avec a réel et b entier par multiplication successives.

Version 1:

```
Algorithme Puissance;
    Var
        a, P: réel;   // va contenir le résultat de la puissance
        b, i: entier;
    Début
        Ecrire("Entrer la valeur de a=") ;
        Lire(a);
        Ecrire("Entrer la valeur de b=") ;
        Lire(b);
        P ← 1 ;                          // initialisation du résultat du produit
        Pour i ← 1 à Abs(b) faire  // abs() fonction qui retourne la valeur absolue de b
            P ←P * a;              //produit de axa… b fois

        Fin Pour
            Si (b < 0) alors       // si b est négative alors le résultat sera 1/P
                P← 1 / P;
            Fin Si
            Ecrire(a," à la puissance ", b, "=", P) ;
    Fin
```

Version 2:

```
Algorithme Puissance;
    Var
        a, P: réel;   // va contenir le résultat de la puissance
        b, i: entier;
    Début
        Ecrire("Entrer la valeur de a=") ;
        Lire(a);
        Ecrire("Entrer la valeur de b=") ;
        Lire(b);
        Si (a=0) alors
            Si (b<=0) alors
                Écrire (“Erreur”);
            Sinon
                P←0;
                Écrire (a, “Puissance”, b,”=”, P);
            Finsi
```

```
Sinon
P ← 1 ;                              // initialisation du résultat du produit
Pour i ← 1 à Abs(b) faire   //abs() fonction qui retourne la valeur absolue de b
        P ←P * a;            // produit de axa… b fois

Fin Pour
        Si (b < 0) alors       // si b est négative alors le résultat sera 1/P
                P← 1 / P;
        Fin Si
        Ecrire(a," à la puissance ", b, "=", P) ;

Fin
```

Exercice 6

Ecrire un algorithme qui lit un entier positif et vérifie si ce nombre est premier ou non.

Remarque : un nombre premier n'est divisible que par 1 et par lui-même (par exemple: 19).

Exemple :

soit n=19 ?

Diviseurs de n=19 (à part 1 et 19) :
2, 3, 4, 5, 6, 7, 8, 9 : on vérifie si ses nombres sont des diviseurs de 19
C-à-d : soit i allant de 2 à (n div 2) : à chaque étape tester si i est diviseur de n (c-à-d : si (n mod i=0) alors i est diviseur de 19.
Pour savoir si on a trouvé au moins un diviseur, on doit ajouter une variable nb_div qui permet de compter combien de diviseurs, trouvés.
Pour voir si n est premier ou non il suffit de comparer nb_div à 0

```
Algorithme Premier;
    Var
        n, i, nb_div: Entier ;
    Début
        Ecrire("Entrer un entier possitif=") ;
        Lire(n) ;
        nb_div ← 0 ;   // initialisation du nombre de diviseurs
        i←2 ;
        Tant que (i <= ndiv2) Faire
            Si ( n Mod i == 0 ) Alors
                nb_div← nb_div + 1 ;  /* incrémentation du nombre de
                diviseurs */
            FinSi
            i ← i +1 ;
        Fin Tant que

        Si (nb_div > 0) Alors
            Ecrire("C'est un nombre premier") ;
```

Sinon

Ecrire("Ce n'est pas un nombre premier") ;

FinSi

Fin

Exercice7

Ecrire un algorithme qui lit deux entiers positifs A et B puis calcule et affiche leur PGCD en utilisant la méthode suivante:

- Si A = B; PGCD(A, B) = A
- Si A>B; PGCD(A, B) = PGCD(A-B, B)
- Si A<B; PGCD(A, B) = PGCD(A, B-A)

Exemple : PGCD(18, 45) = PGCD(18, 27) = PGCD(18, 9) = PGCD(9, 9) = 9

Corrigé :

On a une boucle de type soit **Tantque** soit **Repeter**.
Test d'arrêt : lorsque a=b
A chaque étape soit on modifie la valeur de A ou bien celle de B.

```
Algorithme PGCD ;
    Var
        a, b: Entier ;
    Début
        Ecrire("Entrer la valeur de a =") ;
        Lire(a) ;
        Ecrire("Entrer la valeur de b =") ;
        Lire(b) ;
        Répéter
            Si ( a > b ) Alors
                a← a - b ;
            FinSi
            Si ( a < b ) Alors
                b← b - a ;
            FinSi
        Jusqu'à ( a=b )
            Ecrire("Le PGCD =", a) ;
        Fin
```

Exercice 8 :

Ecrire un algorithme qui calcule le PPCM (Plus Petit Commun Multiple) de deux entiers positifs A et B en utilisant la méthode suivante :

- Permuter, si nécessaire, les données de façon à ranger dans A le plus grand des deux entiers A et B;

- Chercher le plus petit multiple de A qui est aussi multiple de B.

Exemple :

PPCM(6, 8) = PPCM(8, 6) = est ce que 8 est un multiple de 6 sinon.

On teste si 16, qui est 8+8, est multiple de 6 si oui on s'arrête sinon on continue

Ensuite 24, qui est 16+8, et puisque 24 est multiple de 6 on s'arrête 24.

Corrigé :

1- Lecture de a et b les données
2- Si (a<b) alors permuter a et b (c←**a ; a←b ;b←c ;)**
3- Soit ppcm←a ;
Boucle par exemple tant que ppcm n'est pas multiple de b faire
ppcm←ppcm+a ;
****
ppcm←a ;
Tant que (ppcm mod b !=0)faire
    ppcm←ppcm+a ;
FinTantque

Algorithme PPCM ;
Var
    a, b, ppcm, c: Entier;
Début
    Ecrire("Entrer la valeur de a =") ;
    Lire(a) ;
    Ecrire("Entrer la valeur de b =") ;
    Lire(b) ;
    Si (a < b) **Alors**
        c ← a ;
        a ← b ;
        b ← c ;
    Finsi
    ppcm ← a ;
    Tant que (((ppcm) Mod b) !=0) **Faire**
        ppcm ← ppcm +a ;
    Fin Tant que
    Ecrire ("PPCM =", ppcm) ;
Fin

Exercice 9:

Ecrire un algorithme qui calcule et affiche les 100 premiers termes de la suite de Fibonacci.

La suite de Fibonacci est définie par:

- F0 = 0
- F1 = 1
- Fn = Fn -2 + Fn -1 pour n > 1.

Corrigé :

On choisit 3 variables qui représentent les termes Fn, **Fn-1** et Fn-2 : Fn est représentée par une variable notée Fc (courant)

Fn-1 est représentée par une variable notée Fd (dernier)

Fn-2 est représentée par une variable notée Fad (avant dernier)

Itération	**Fad**	**Fd**	**Fc**
i ←2	**1**	**1**	**2**
i ←3	**1**	**2**	**3**
i ←4	**2**	**3**	**5**
i ←5	**3**	**5**	**8**

Pour chaque étape de la boucle :

- On calcule d'abord par Fc ←Fad+Fd ;
- Ensuite Fad←Fd ;
- Et enfin Fd←Fc ;

Algorithme Fibonacci ;

```
Var
   // Fad pour stocker Fn-2, Fd pour stocker Fn-1 et Fc pour //stocker le Fn
   Fad, Fd, Fc, i: Entier;
Début
   Fad ← 1;
   Ecrire("F0 = ", Fad) ;
   Fd ← 1 ;
   Ecrire("F1 = ", Fd) ;
   Pour i de 2 à 99 Faire
          Fc ← Fad+ Fd ;
          Ecrire("F",i," = ", Fc);
          Fad ← Fd;   /* préparation de  la valeur pour la prochaine itération Fn-2
         prendra Fn-1 */
          Fd ← Fc;         // Fn-1 prendra Fn
   FinPour
Fin
```

Exercice 10

Un nombre parfait est un nombre présentant la particularité d'être égal à la somme de tous ses diviseurs, excepté lui-même.

Le premier nombre parfait est 6 = 3 + 2 + 1.

Ecrire un algorithme qui affiche tous les nombres parfais inférieurs à 1000.

Corrigé :

Soit n un entier de 6 à 1000 est représenté par : boucle Pour n←6 à 1000 faire
Tester si n est parfait?

Parcourir ses diviseurs, les sommer et comparer cette somme à n
pour passer sur tous les entiers compris entre 6 et 1000

```
s←0 ;
Pour i ←1 à (n div 2) faire
        Si (n mod i=0) alors
                s←s+i;
        Finsi
FinPour
si(s=n) alors
        ecrire(n);
Finsi
```

```
Algorithme parfaits ;
    Var n, s, i: Entier;
    Début
    //pas de données
    Pour n ← 6 à 1000 faire
                // parcourir tous les entiers de 6 à 1000 pour chercher les parfaits
        s←0 ;   // pour sommer les diviseurs de n
        Pour i ← 1 à ( n div 2 ) faire
         Si (n mod i == 0) alors
                s← s + i;       // si i est un diviseur de n, on l'ajoute à s
         Finsi
        FinPour i
        Si ( s== n) Alors
                Ecrire(n, " est un nombre parfait") ;
        FinSi
    FinPour n
Fin
```