Université Moulay Ismail **Année universitaire : 2023/2024**
Faculté des sciences de Meknès **Filière : MIP/Digitalisation**
Département d'Informatique **Module : INFO2**

TD N°1 : Les fonctions et les procédures
(Corrigé)

Exercice 1 :

a. Écrivez une fonction (***Max( )***) qui demande deux nombres à l'utilisateur, calcule et affiche le plus grand des deux. Tester la fonction

```
Fonction Max(a : entier , b : entier) : Entier
    Début
        Si (a > b) alors
            écrire("le plus grand nombre est : ", a)
            retourner a
        Sinon
            écrire("le plus grand nombre est : ", b)
            retourner b
        FinSi
    Fin
```

Test de la fonction :

Algorithme Maximum

```
    Variables x,y : entier

    Fonction Max(a : entier , b : entier) : Entier
        Début
            Si (a ≥ b) alors
                écrire("le plus grand nombre est : ", a)
                retourner a
            Sinon
                écrire("le plus grand nombre est : ", b)
                retourner b
            FinSi
        Fin

    Début
        Ecrire ("Entrer le premier nombre : ")
        Lire(x)
        Ecrire ("Entrer le deuxième nombre :")
        Lire(y)
        écrire("le plus grand nombre  est :",  Max(x,y))

    Fin
```

Programme en python :

```
def max(a:int, b:int) :
      """La fonction max() existe déjà en python! Mais le but ici c'est
      d'appliquer le cours d'algorithmique, notamment les structures
      conditionnelles et les fonctions """

    if a<b:
        print("le plus grand nombre est : ", b)
        max=b
    else:
        print("le plus grand nombre est :", a)
        max=a
    return max
x=int(input("Entrer le premier nombre : "))
y=int(input("Entrer le deuxième nombre : "))
print("Le plus grand nombre  est :", Max(x,y))
```

b. Écrivez une fonction (***Maxim****(, , ,)*) qui demande trois nombres à l'utilisateur, calcule et affiche le plus grand en utilisant la fonction ***Max***.

Fonction Maxim(a : entier , b : entier, c : entier) : Entier

Variables

Max2, Max : entier

Début

Max2 ← Max (a,b)

Max ← Max (Max2,c)

Ecrire ("Le plus grand nombre est : ", Max)

retourner Max

Fin

Programme en python :

Solution 1 :

```
def max(x:int, y:int, z:int) :
    if x<y<z:
        #print("z est le maximum: ", z)
        max=z
    elif x<z<y:
        #print("y est le maximum: ", y)
        max=y
    else:
        #print("x est le maximum: ", x)
        max=x
    return max

print("Le maximum est: ", max(5,10,48))
```

Solution 2 :

```
def max(a:int, b:int) :
    """La fonction max() existe déjà en python! Mais
    le but ici c'est d'appliquer le cours
    d'algorithmique, notamment les structures
    conditionnelles et les fonctions """

    if a<b:
        print("le plus grand nombre est : ", b)
        max=b
    else:
        print("le plus grand nombre est :", a)
        max=a
    return max
def Maxim(x,y,z):
    max2=Max(x,y)
    max3=Max(max2,z)
    return max3


x=int(input("Entrer un nombre entier x:"))
y=int(input("Entrer un nombre entier y:"))
z=int(input("Entrer un nombre entier z:"))
print("Le maximum est: ", Maxim(x,y,z))
```

Exercice 2 :

Ecrire une fonction qui permet de calculer la **multiplication de deux nombres A et B** entiers en utilisant l'addition.

Fonction Multiplier(a:entier,b:entier) : entier

Variables produit, i : entiers
Debut
Produit←0
Pour i ← 1 A a Faire
Produit←produit +b
Finpour
Retourner produit
Fin

Algorithme ….
Fonction Multiplier

Programme en python :

```
def multiplier(a:int,b:int) :
    produit=0
    for i in range (1, a+1) :
        produit=produit + b
```

```
    return produit

a = int(input('Entrer le 1er nombre: '))
b = int(input('Entrer le 2eme nombre: '))

print("La multiplication de ", a," et ",b,"est = ",multiplier(a,b) )
```

Exercice 3 :

Ecrire une **fonction distance** ayant comme paramètres 4 doubles xa, ya, xb et yb qui représentent les coordonnées de deux points A et B et qui renvoie la distance AB. Tester cette fonction dans un programme principal.

```
Fonction distance (xa:Réel, xb:Réel, ya:Réel, yb:Réel): Réel
Variables
        AB: Réel
Début
        AB <--sqrt(pow((xa-xb),2) + pow((ya-yb),2))
        retourner AB
Fin

Algorithme test
variables
        xa, xb ,ya ,yb, distanceAB : Réels
fonction distance()
Début
        Ecrire ("Entrer les coordonnées du point A:")
        Lire(xa,ya)
        Ecrire ("Entrer les coordonnées du point B:")
        Lire(xb,yb)
        Ecrire("La distance AB est : ", distance(xa,xb,ya,yb))
Fin
```

Programme en python :

```
from math import sqrt, pow
def distance(xa:float, xb:float, ya:float, yb:float) :
    AB=sqrt(pow((xa-xb), 2)+pow((ya-yb),2))
    return AB


print('Entrer les coordonnées du point A:')
xa = int(input('xa:'))
ya = int(input('ya:'))
print('Entrer les coordonnées du point B:')
xb = int(input('xb:'))
```

```
yb = int(input('yb:'))
print("La distance AB est : ", distance(xa,xb,ya,yb))
```

Exercice 4 :

Ecrire une procédure qui affiche tous les **nombres pairs** compris entre deux valeurs entières positives lue x et y.

Procédure Nombres_Pairs() :

```
Variables
    x, y, z : entier
Début
   Lire (x, y)
  Si (x > y) Alors
     z ← x
     x ← y
     y ← z
  FinSi
   TantQue (x ≤ y)
        Si ( x MOD 2 = 0) Alors
              Ecrire (x)
        FinSi
         x ← x + 1
  FinTantQue
Fin
```

Programme en python :

```
def nombres_pairs(x: int, y: int):
    if (x > y) :
        z = x
        x = y
        y = z
    while (x <= y)  :
        if ( x % 2 == 0) :
           print(x)
        x = x + 1

x = int(input('Entrer le 1er nombre: '))
y = int(input('Entrer le 2eme nombre: '))
print("les nombres pairs sont:", nombres_pairs(x,y))
```

Exercice 5 :

Ecrire une fonction qui calcule le **PGCD** de deux entiers strictement positifs.

Solution 1 :

Fonction PGCD (a : entier, b : entier) : entier

```
Début
        TantQue (a * b ≠ 0)
              Si (a > b) Alors
                    a ← a - b
```

```
                    Sinon
                            b ← b - a
                    FinSi
            FinTantQue
            Si (a = 0)
                Retourner b
            Sinon
                Retournea
            FinSi
        Fin
```

Solution 2 :

```
Fonction PGCD (a : entier, b : entier) : entier
        Début
            Si (a > b) Alors
                c ← a
                a ← b
                b ← c
            FinSi
            Pgcd ←1
            Pour i allant de 2 à (a/2) faire
                Si (a %i=0) ET (b%i=0) Alors
                  Pgcd ←i
                FinSi
            FinPour
            Si (b%a = 0)
                Retourner a
            Sinon
                Retourner pgcd
            FinSi
        Fin
```

Programme en python :

```
def PGCD(a: int, b: int):
    while (a * b != 0):
        if (a > b) :
            a = a - b
        else:
            b = b - a
    if (a == 0) :
       return b
    else:
        return a

a = int(input('Entrer le 1er nombre: '))
b = int(input('Entrer le 2eme nombre: '))
print("Le PGCD est:", PGCD(a,b))
```

Exercice 6 :

Ecrire une procédure qui permet de saisir un nombre entier positif et **d'afficher son image miroir.** Exemple le nombre est 3524, on doit afficher 4253.

Procédure Miroir (x : entier)

```
Variables
   a : entier
Début
  TantQue (x <> 0)
     a ← x % 10
     Ecrire(a)
     x ← x / 10
  FinTantQue
Fin
```

Programme en python :

```
def miroir(x: int):
    b=""
    while(x!=0):
        a=x%10
        b=b+str(a)
        x=x//10
    print(b)

x = int(input('Entrer le nombre: '))
miroir(x)
```

Exercice 7 :

a. Écrire en langage algorithmique une fonction booléenne qui retourne vrai si un entier n passé en paramètre est un **nombre parfait**, faux sinon.

```
Fonction parfait (n: entier) : booléen
    Variables
       res : booléen
       i, som : entier
    Début
      Som← 1
      Pour i allant de 2 à n div 2 faire
         Si (n % i = 0) Alors
             Som←som + i
         FinSi
      FinPour
      Si (som=n) alors
         Retourner(vrai)
      Sinon
        Retourner(faux)
    Fin
```

Programme en python :

```
def parfait(x: int):
    Som=1
    for i in range (2, (x//2)+1):
        if (x % i == 0) :
            Som=Som + i
    if(Som==x) :
        return "TRUE"
    else:
        return "FALSE"

x = int(input('Entrer un nombre: '))
print ("Le nombre ", x, "est parfait! : ", parfait(x))
```

b. Écrire en algorithmique le programme principal permettant d'afficher la liste des nombres parfaits compris entre 1 et 10000. On utilisera le résultat renvoyé par la fonction précédente.

```
Algorithme Afficher_nombres_parfaits
    Variables
        i : entier
        Fonction parfait (n : entier) : booléen
            Variables
                res : booléen
                i, som : entier
            Début
                Som← 1
                Pour i allant de 2 à n div 2 faire
                  Si (n % i = 0) Alors
                     Som←som + i
                  FinSi
                FinPour
                Si (som=n) alors
                  Retourner(vrai)
                Sinon
                  Retourner(faux)
                FinSi
          Fin

    Début
        Pour i = 1 à 10000
          Si (parfait(i) = vrai) Alors
                    Ecrire (i, "est un nombre parfait")
          FinSi
        FinPour
    Fin
```

Programme en python :

```
def parfait(x: int):
    Som=1
```

```
    for i in range (2, (x//2)+1):
        if (x % i == 0) :
            Som=Som + i
    if(Som==x) :
        return "TRUE"
    else:
        return "FALSE"

for i in range(1,10000) :
    if (parfait(i)=="TRUE"):
        print(i, "est un nombre parfait")
```

Exercice 8 :

On se propose de calculer l'expression : $e^{x} = \sum_{i=0}^{n} \frac{x^{i}}{i!}$

Pour cela, on a besoin des fonctions puissances et factorielle.

a. Ecrire une fonction **Puissance** (X, k) qui calcule X^K

Fonction puissance (x : réel, k : entier) : réel
Variables
P : réel
I : entier
Début
P←1
Pour i = 1 à k
P=P*x
FinPour
Retourner (P)
Fin

Programme en python :

```
def puissance(x:float, k:int):
"""La fonction pow() est une fonction intégrée et elle existe
déjà en python! Mais le but ici c'est de définir notre propre
fonction puissance() en utilisant les boucles et les fonctions
"""
    P=1
    for i in range(0, k):
        P=P*x
    return (P)

x = float(input('Entrer le nombre: '))
y = int(input('Entrer l exposant: '))
print("La puissance est :", puissance(x,y))
```

b. Ecrire une fonction **Fact** (k) qui calcule K ! (la factorielle de 4, notée 4!, vaut1x2x3x4)

Fonction factorielle (k : entier) : entier
Variables
F, i : entier
Début
F←1

```
        Pour i =2 à k
            F=F*i
        FinPour
        Retourner (F)
    FinFonction
```

Programme en python :

```python
def factorielle(k:int):
    f=1
    for i in range(2, k+1):
        f=f*i
    return (f)

x = int(input('Entrer le nombre: '))
print("Le factoriel est :", factorielle(x))
```

c. Ecrire le programme principal qui calcul e^x (on commencera par saisir X et n).

```
Algorithme Exercice8
    Variables
        x, e : réel
        n : entier
    Début
        Répéter
            Ecrire (''introduisez un nombre n>=0 :'')
            Lire (n)
        Jusqu'à (n>=0)
        Ecrie('' donnez une valeur x :'')
        lire(x)
        exponentielle (x,n)
        e←0
            Pour i = 0 à n
                e←e+ puissance(x,i)/factorielle(i)
            FinPour
        Ecrire(''L'expression e^x='',e)
Fin
```

Programme en python :

```python
def puissance(x:float, k:int):
    P=1
    for i in range(0, k):
        P=P*x
    return (P)

def factorielle(k:int):
    f=1
    for i in range(2, k+1):
        f=f*i
    return (f)
```

```
n=int(input("introduisez un nombre n>=0 :"))
x = int(input('Entrer le nombre: '))
e=0
for i in range(1, n+1):
    e=e+puissance(x,i)/factorielle(i)

print("L’expression e^x=", e)
```

Exercice 8 :

a. Ecrire une fonction qui saisit un tableau d'entier de N éléments.

```
Fonction OccurrenceTab (n : entier, tableau T : réel, val : réel): entier
    Variables
            i, N : entiers ;
            T : tableau [1..100] d'entiers ;
    Début
            Ecrire ("Entrez le nombre de valeurs des tableaux :") ;
            Lire (N) ;
            Ecrire ("Saisissez les éléments du premier tableau :") ;
            Pour i ← 1 à N faire
                    Ecrire ("Entrez le nombre n° ", i) ;
                    Lire (T1[i]);
            FinPour
```

b. Ecrire une fonction qui saisit un tableau d'entier de N éléments.

```
Fonction OccurrenceTab (n : entier, tableau T : réel, val : réel): entier
  Variables i, occur : entier
    Début
            occur ← 0
            Pour i allant de 1 à n faire
                 si (T[i] = val) alors
                       occur ←  occur +1
                  Finsi
             FinPour
            Retourner (occur)
    Fin
```

c. Ecrire l'algorithme principale où on fait l'appel de la fonction **OccurrenceTab**.

```
Algorithme Recherche-Occurrence
    Variable tableau A[10] : réel
          p,Occ :entier
          x :réel
    Début
            p ← 10
            SaisieTab(p, A)          //Fonction déjà étudiée en cours
```

```
        Ecrire("saisir la valeur de x")
        Lire(x)
        Occ←OccurrenceTab(10,A ,x)
        Ecrire("Le nombre d'occurrence de l'élément",x,"est :,Occ)
    Fin
```

Exercice 7

```
Procedure Second_deg (a, b, c : entier)
Variable
delta, x1, x2 : réel
Debut
Si ( a = 0)  Alors
  Si (b = 0) Alors
     Si (c = 0) Alors
        Ecrire (''R est la solution'')
     Sinon
       Ecrire (''Impossible'')
     FinSi
  Sinon
   x1 ←    - c / b
   Ecrire (x1) ;
  FinSi
Sinon
  delta ←b*b – 4*a*c
  Si (delta < 0) Alors
   Ecrire (''Pas de solution dans R'')
  Sinon
   Si (delta = 0) Alors
    x1 ←    -b / 2*a
    Ecrire (''Solution double '', x1)
  Sinon
```

```
  x1 ← - b + √delta / 2*a
  x2 ← - b - √delta / 2*a
  Ecrire (x1, x2)
  FinSi
 FinSi
FinSi
FinProcédure
```

Exercice 8

a-

```
Fonction multiple (A : entier, B : entier) : entier
Variables
    Res, Y : entier
Début
   Res ←0
  Si (B < 0) Alors
     Y ← -B
  Sinon
     Y ← B
  FinSi
  TantQue (Y > 0)
     Res ← Res + A
     Y ←Y – 1
  FinTantQue
  Si (B < 0) Alors
    Res ← - Res
  FinSi
  Retourne (Res)
```