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

Fin
```

Programme en python :

```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)

écrire("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

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 :

```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 :

```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 :

```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")
```