

Correction TD3-ASD

Les procédures et les fonctions

Exercice 1

1) Procédure SaisieBornes (Var Bi: entier, Bs : entier)

DÉBUT

 Répéter

 Ecrire (“Donner deux valeurs Bi et Bs : ”)

 Lire (Bi, Bs)

 Jusqu’à (Bi > 0 et Bi < Bs)

FIN

2) Procédure CalculeProduit (Bi: entier, Bs : entier ; Var P : entier)

 Var k : entier

DÉBUT

 P ← 1

Pour k de Bi à Bs faire

 P ← P * k

Fin pour

FIN

3) a) Ce sous-programme doit être une procédure puisque ces trois lignes ne retournent aucune valeur.

b) Procédure Affichage (Bi: entier, Bs: entier, P : entier)

DÉBUT

 Ecrire (“Borne inférieure Bi=”, Bi)

 Ecrire (“Borne supérieure Bs=”, Bs)

 Ecrire (“La somme des entiers comprise entre”, Bi, ”et”, Bs, ”est”, P)

FIN

c) Algorithme ProduitIntervalle2

 Var

 Bi, Bs, P : entier

DÉBUT

SaisieBornes (Bi, Bs)

CalculeProduit (Bi, Bs, Somme)

Affichage(Bi, Bs, Somme)

FIN

4) a) Fonction CalcProd (Bi: entier, Bs : entier) : entier

 Var

 k, P :entier

DÉBUT

 P ← 1

Pour k de Bi à Bs faire

 P ← P * k

Fin pour

 CalcProd ← P

FIN

b) Algorithme SommeIntervalle3

 Var

 Bi, Bs, P : entier

DÉBUT

SaisieBornes (Bi, Bs)

 P ← *CalculeProd (Bi, Bs)*

Affichage (Bi, Bs, P)

FIN

Exercice 2

Fonction Premier (n : entier) : booléen

Var

j : entier

hyp : booléen

DÉBUT

j ← 2

hyp ← vrai

Tant que (j ≤ n div 2 et hyp =vrai) Faire

Si (n mod j = 0) alors

hyp ← faux

Sinon

j ← j + 1

Finsi

FinTantque

Premier ← hyp

FIN

Fonction Test (n : entier) : booléen

Var

Trouve : booléen

DÉBUT

Trouve ← vrai

Tant que (n > 0 et Trouve=vrai) Faire

Si n mod 2 =0 alors

n ← n div 2

Si non

Trouve ← faux

Fin si

FinTantque

Test ← Trouve.

FIN

Algorithme Exercice2

VAR

N : entier

DÉBUT

Ecrire ("Donner un entier : ")

Lire (N)

Si (Premier (N) = vrai et Test (N) =vrai) alors

Ecrire (N , "est premier et une puissance de 2")

Si non

Si (Premier (N) = vrai ou Test (N) =vrai) alors

Ecrire (N , "est premier ou une puissance de 2")

Si non

Ecrire (N , "est ni premier ni une puissance de 2")

Fin si

Fin si

FIN

Exercice 3

Fonction MINIMUM (x : entier, y : entier) : entier

VAR

```

    min : entier
DÉBUT
    Si (x < y) alors
        min ← x
    Sinon
        min ← y
    Fin si
    MINIMUM ← min

```

```

FIN
Fonction MAXIMUM ( x: entier , y : entier ) : entier
VAR

```

```

    max : entier
DÉBUT
    Si (x > y) alors
        max ← x
    Si non
        max ← y
    Fin si
    MAXIMUM ← max

```

```

FIN
Algorithme MINMAX
VAR

```

```

    a, b, c, d : entier
DÉBUT
    Ecrire ("Donner 4 entier : ")
    Lire (a , b , c , d )
    Ecrire ("Le minimum de 4 entiers =",MINIMUM(MINIMUM(a ,b),MINIMUM(c ,d)))
    Ecrire ("Le maximum de 4 entiers=",MAXIMUM(MAXIMUM(a,b),MAXIMUM(c ,d)))

```

```

FIN

```

Exercice 4

```

Procédure SAISIE (var X : entier)

```

```

DÉBUT
    Répéter
        Ecrire ("Donner un entier : ")
        Lire (X)
    Jusqu'à (X>0)

```

```

FIN

```

```

Fonction SOMME ( X : entier ) : entier
VAR

```

```

    S:entier
DÉBUT
    S ← 0
    Tantque ( X <> 0 ) faire
        S ← S + X mod 10
        X ← X div 10
    Fin tantque
    SOMME ← P

```

```

FIN
Fonction PRODUIT ( X : entier ) : entier
VAR

```

```

    P, Y : entier
DÉBUT
    P ← 1
    Tantque ( X <> 0 ) faire
        Y ← X mod 10
        Si ( Y ≠ 0 ) alors
            P ← P * Y
        Fin si
        X ← X div 10
    Fin tantque
    PRODUIT ← P
FIN

```

Algorithme AMISCOUSINS

```

VAR
    A, B : entier
DÉBUT
    SAISIE ( A )
    SAISIE ( B )
    Si ( SOMME ( A ) = SOMME ( B ) ) alors
        Ecrire ( A, "et", B, "sont amis" )
    Sinon
        Si ( PRODUIT ( A ) = PRODUIT ( B ) ) alors
            Ecrire ( A, "et", B, "sont cousins" )
        Sinon
            Ecrire ( A, "et", B, "ne sont pas amis ou cousins " )
        Fin si
    Fin si
FIN

```

FIN

Exercice 5

Fonction INVERSION (N : entier) : entier

VAR NINV : entier

```

DÉBUT
    NINV ← 0
    Tantque ( N > 0 ) Faire
        NINV ← NINV * 10
        NINV ← NINV + N mod 10
        N ← N / 10
    Fin Tant que
    INVERSION ← NINV

```

FIN

Fonction PALINDROME (N : entier) : entier

```

DÉBUT
    Si ( N = INVERSION ( N ) ) alors
        PALINDROME ← 1
    Sinon
        PALINDROME ← 0
    fin si

```

FIN

Algorithme exercice5

VAR

N : entier

DÉBUT

Répéter
 Écrire ("Donner un réel : ")
 Lire (N)
Jusqu'à (N > 0)
Si (N = INVERSION (N)) alors
 écrire (N, " est palindrome");
Sinon
 écrire (N, "n'est pas palindrome");
fin si

FIN

Exercice 6

Fonction SAISIE () : entier

Var

x : entier

DÉBUT

Répéter
 Écrire ("Donner un réel : ")
 Lire (x)
Jusqu'à (x >= 10 ET x <= 99 Et x MOD 10 < > x DIV 10)
 SAISIE ← x

FIN

Procédure LISTE_9 (x : entier)

Var

xs , diff : entier

DÉBUT

Écrire ("la liste vers 9 de " , x , "est la suivante : ")
Répéter
 Écrire (" ", x)
 xs ← (x MOD 10)*10 + x DIV 10
 diff ← abs(x - xs)
 x ← diff
Jusqu'à (diff = 9)

FIN

Algorithme exercice6

VAR

n : entier

DÉBUT

n ← SAISIE ()
 LISTE_9 (n)

FIN

Exercice 7

Fonction SOMME (i : entier) : réel

VAR

S : réel j : entier

DÉBUT

S ← 0
Pour j de 1 à i faire
 S ← S + 1/j

Fin Pour

SOMME ← S

FIN

Fonction PRODUIT (n : entier) : réel

```

VAR
    P : réel i : entier
DÉBUT
    P ← 1
    Pour i de 1 à n faire
        P ← P * SOMME(i)
    Fin Pour
    PRODUIT ← P

```

FIN

Algorithme Exercice7

```

VAR

```

```

    n : entier

```

```

DÉBUT

```

```

    Répéter

```

```

        Ecrire ("Donner un entier : ")

```

```

        Lire ( n)

```

```

    Jusqu'à ( n>0 )

```

```

    Ecrire("Le produit est = ", PRODUIT(n))

```

FIN

Exercice 8

Procédure SAISIE (VAR X : réel)

```

DÉBUT

```

```

    Répéter

```

```

        Écrire ("Donner un réel : ")

```

```

        Lire (X)

```

```

    Jusqu'à (X > 0)

```

FIN

Fonction RACINE (X : réel) : réel

```

VAR

```

```

    V0, V1 : réels

```

```

DÉBUT

```

```

    V1 ← (1 + X)/2

```

```

    Répéter

```

```

        V0 ← V1

```

```

        V1 ← (V0 + X/V0)/2

```

```

    Jusqu'à ( abs(V0 – V1) <= 0,0001)

```

```

    RACINE ← V1

```

FIN

Algorithme exercice8

```

VAR

```

```

    X : réel

```

```

DÉBUT

```

```

    SAISIE (X)

```

```

    Ecrire ("La racine de ", X, "=", RACINE(X))

```

FIN

Exercice 9

Fonction CALCUL (n : entier) : réel

```

VAR

```

```

    U , U0 , U1, i : entier

```

```

DEBUT

```

```

    U0 ← 2

```

```

    U1 ←3

```

Pour i de 3 à n + 1 faire

U ← U1 + U0

U0 ← U1

U1 ← U

Fin Pour

CALCUL ← U

FIN

Algorithme exercice9

VAR

n : entier

DÉBUT

Répéter

Ecrire ("Donner un entier : ")

Lire (n)

Jusqu'à (n > 0)

Ecrire("Le" n , " ème terme de la suite est = ", CALCUL(n))

FIN