

## SERIE TD 1 ALGO

### Exercice 1 :

Ecrire un programme qui saisit deux entiers a et b, calcule et affiche le quotient entier, le reste de la division et le ratio (quotient réel).

### Résolution :

#### PROGRAMME\_APP1

```
Var a, b : entier
DEBUT
|   Repeter
|       ecrire (« Veuillez entrer deux nombre entier »)
|       lire a, b
|   Jusqu'à (b != 0)
|   afficher (« → La division entre », a, « et », b, « a pour |
|       quotient entier : », a div b, « comme reste de
|       division : », a mod b, « et comme ratio : », a/b)
FIN
```

### Exercice 2 :

Ecrire un programme qui demande à l'utilisateur de donner le rayon d'un cercle et lui retourne sa surface et son périmètre.

$PI = 4 * \text{arc tangente de } 1$ . la fonction arc tangente est atan ex : atan(2).

// Aire =  $\pi * r^2$  | Perimetre =  $2 * \pi * r$

### Résolution :

#### PROGRAMME\_APP2

```
Var ray : reel
Const PI = 4 * atan(1)
DEBUT
|   Repeter
|       ecrire ("Veuillez donner le rayon du cercle : ")
|       lire ray
|   Jusqu'à (ray > 0)
```

```
|  afficher (« La surface de votre cercle est : », PI*sqr(ray))
|  afficher (« Le Périmètre de votre cercle est : », 2*PI*ray)
FIN
```

Exercice 3 :

Version 1 :

Faire un programme qui saisit 3 résistances : R1, R2 et R3.

Calculer et afficher la résistance en série :  $R1 + R2 + R3$

Calculer et afficher la résistance en parallèle :  $(R1 * R2 * R3) / (R1*R2 + R2*R3 + R1*R3)$

Version 2 :

Demander a l'utilisateur d'indiquer son choix.

S'il entre la valeur 1, calculer et afficher la fréquence en série.

S'il entre la valeur 2, calculer et afficher la fréquence en parallèle.

Résolution :

#### PROGRAMME APP3-V1

```
Var r1, r2, r3, rs, rp: reel
DEBUT
|  ecrire (« Veuillez saisir trois Résistances : »)
|  lire r1, r2, r3
|  rs <- r1 + r2 + r3
|  rp <- (r1 * r2 * r3) / (r1*r2 + r2*r3 + r1*r3)
|  afficher (« La Resistance en serie est égale à : », rs)
|  afficher (« La resis. En parallèle donne : », rp)
FIN
```

#### PROGRAMME APP3-V2

```
Var r1, r2, r3, rs, rp: reel
Var choice : entier
DEBUT
|  ecrire (« Veuillez saisir trois Résistances : »)
```

```

|  lire r1, r2, r3
|      rs <- r1 + r2 + r3
|      rp <- (r1 * r2 * r3) / (r1*r2 + r2*r3 + r1*r3)
|
|  Repeter
|      ecrire (« Veuillez faire un choix entre 1 et 2 »)
|      lire choice
|  Jusqu'à (choice = 1 OR choice = 2)
|  SELONQUE
|      choice = 1 : afficher (« La Resistance en serie est égale
à : », rs)
|      choice = 2 : afficher (« La resis. En parallèle donne : », rp)
|  SINON
|      afficher (« CHOIX INDISPONIBLE »)
|  FINSELONQUE
FIN

```

#### Exercice 4

Ecrire un programme qui saisit un réel x et un entier n et affiche x à la puissance n.

Version 1 : utiliser la fonction pow du fichier d'en-tête <math.h> ex : pow(x,n)

Version 2 : en utilisant une boucle

Résolution :

#### PROGRAMME APP4-V1

```

Var x : reel
Var n, i : entier
DEBUT
|  ecrire (« veuillez saisir un réel x et un entier n »)
|  lire x, n
|  afficher(x, « Puissance », n, « donne : », pow(x,n))

```

FIN

PROGRAMME APP4-V2

Var x : reel

Var n, i, p: entier

DEBUT

| Repeter

|        ecrire(« veuillez saisir un réel x et un entier n »)

|        lire x, n

|        Jusqu'à (n&gt;0)

|        p = 0

|        Pour i allant de 1 à n faire

|            p \*= x

|        FinPour

|        afficher(x, « Puissance », n, « donne : », p)

FIN

## Exercice 5 :

Ecrire un programme qui saisit 5 variables de type entier au clavier et qui affiche leur somme. Utiliser une boucle (for ou while ou do..while).

## Résolution :

PROGRAMME APP5

Var x, som : entier

DEBUT

|        som &lt;- 0

|        pour i allant de 1 à 5 faire

|            ecrire (« Veuillez donner un nombre entier »)

|            lire x

|            som += x

```

|   FinPour
|   afficher(« La somme des nombres saisi est de : », som)
FIN

```

#### Exercice 6 :

Faire un programme qui saisit les coordonnées de 2 points A (x1, y1) et b(x2, y2) et qui affiche la distance entre les 2 points.

Formule : distante = racine carrée de  $((x1 - x2)^2 + (y1 - y2)^2)$

Racine carrée : sqrt. Ex : sqrt(7) ; <math.h>

#### PROGRAMME APP6

```

Var x1, x2, y1, y2, dist : réel
DEBUT
|   ecrire(« Veuillez saisir les coord x1 et y1 du Point A »)
|   lire x1, y1
|   ecrire(« Veuillez saisir les coord x2 et y2 du Point B »)
|   lire x2, y2
|       dist <- sqrt(sqr((x1-x2)) + sqr((y1-y2)))
|   afficher(« La distance entre le point A et le point B est : »,
dist)
FIN

```

Exercice 7 : Décomposition d'un montant en euros Écrire un algorithme permettant de décomposer un montant entré au clavier en billets de 20, 10, 5 euros et pièces de 2, 1 euros, de façon à minimiser le nombre de billets et de pièces.

#### PROGRAMME APP7

```

Var nb20,nb10,nb5,np2 : entier
Var montant, reste : entier
DEBUT
|   Repeter
|       ecrire("Veuillez entrez un montant : ")
|       lire  montant

```

```

| Jusqu'à (montant > 0)
| nb20 <- montant div 20
| reste <- montant mod 20
| afficher(nb20, « Billet(s) de 20 euros »)
| nb10 <- reste div 10
| reste <- reste mod 10
| afficher(nb10, « Billet(s) de 10 euros »)
| nb5 <- reste div 5
| reste <- reste mod 5
| afficher(nb5, « Billet(s) de 5 euros »)
| np2 <- reste div 2
| reste <- reste mod 2
| afficher(np2, « Pièces(s) de 2 euros »)
| afficher(reste, « Pièce(s) de 1 euros »)
FIN

```

**Exercice 8 :** Ecrire un algorithme permettant de résoudre une équation du second degré.

$$Ax^2 + bx + c = 0$$

#### PROGRAMME APP8

```

Var a, b, c, deltaRoot : réel
DEBUT
|  ecrire(« Veuillez donner les variables a, b et c de votre
polynômes [ax2 + bx + c = 0] » )
|  deltaRoot = sqrt((b * b) - (4 * a * c))
|  Si (deltaRoot > 0 ) alors
|      afficher("\n⇒ √Δ(Delta) = ", deltaRoot, " "qui est supérieur à
Zero »)
|      afficher (« L'Equation admet deux solutions x1 et x2 : »)
|      afficher(« X1 =», (-b + deltaRoot) / 2 * a, « X2 = », (-b
- deltaRoot) / 2 * a)

```

```

|   Sinon
|       Si(deltaRoot = 0) alors
|           afficher("\n➡  $\sqrt{\Delta}(\Delta) =$ ", deltaRoot," "qui est égale
à Zero »)
|           afficher (« L'Equation admet une unique solution X0 : »)
|
|           afficher(« X0 =», -b / (2 * a))
|       Sinon
|           afficher("\n➡  $\sqrt{\Delta}(\Delta) =$ ", deltaRoot," "qui |
est inferieur à Zero »)
|           afficher (« L'Equation n'a pas de solution réelle »)
|       FinSi
|   FinSi
FIN

```

Exercice 9 : Ecrire un algorithme qui donne la durée de vol en heure minute connaissant l'heure de départ et l'heure d'arrivée.

#### PROGRAMME APP9

```

Var hd, md, ha, ma, h, m : entier
DEBUT
|   ecrire(« Veuillez donner l'heure et la minute de départ : »)
|   lire(hd,md)
|   ecrire(« Veuillez donner l'heure et la minute de arrivée : »)
|   lire(ha,ma)
|   m <- [ha*60+ma] - [hd*60+md]
|   h <- m/60
|   m <- m mod 60
|   ecrire(« La Durée de Vol est : », h, « H : », m, « MN »)
FIN

```

#### PROGRAMME APP9V2

```
Var hd, md, ha, ma, h, m : entier
DEBUT
  Ecrire (« entrer horaire de départ et d'arrivée »)
  Lire (hd, md, ha, ma)
  Si ha>hd alors
    Si ma>md alors
      h <= ha-hd et m <= ma-md
      Ecrire (h, m)
    Sinon
      h <= ha-hd-1 et m <= ma+60-md
      Ecrire (h, m)
    FinSi
  Sinon
    Si ma>md alors
      h <= ha-hd+24 et m <= ma-md
      Ecrire (h, m)
    Sinon
      h <= ha-hd+24-1 et m <= ma+60-md
      Ecrire (h, m)
    FinSi
  FinSi
FIN
```

Exercice 10 : Ecrire un algorithme qui lit trois valeurs entières (A, B et C) et qui permet de les trier par échanges successifs Et enfin les afficher dans l'ordre 4.

#### PROGRAMME APP10

```
Var a, b, c : entier
Fonction switch (donnée a, b : entier) : entier
```



```
Var tmp : entier
DEBUT
    tmp <- a
    a <- b
    b <- tmp
FIN
DEBUT
|   Ecrire (« Entrer les valeurs entières : »)
|   lire a, b, c
|   Si (a > b) alors
|       switch(a,b)
|       Si (b > c) alors
|           switch(b,c)
|           Si (a > b) alors
|               switch(a,b)
|           FinSi
|       FinSi
|   Sinon
|       Si (b > c) alors
|           switch (B, C)
|           Si (a > b) alors
|               switch (a, b)
|           FINSI
|       FINSI
|   FINSI
|   afficher (« a,b,c dans l'ordre donne : » a, b, c)
FIN
```

**Exercice 11:** Ecrire un algorithme calculateur permettant la saisie du premier entier (a) de l'opération (+ ou - ou \* ou / : sont des caractères) et du deuxième entier (b) et qui affiche le résultat.

PROGRAMME APP11

```
Var a, b : entier
Var oper : caractere
DEBUT
|   ecrire (« Veuillez entrer le premier entier : »)
|   lire a
|   ecrire (« Veuillez saisir l'opération à faire »)
|   lire oper
|   ecrire (« Saisissez le deuxième entier : »)
|   lire b
|   Suivant (oper) faire
|       oper = « + » :
|           ecrire (« La somme de », a, « et de », b, « est égale à »,
a+b)
|       oper = « * » :
|           ecrire (« Le produit de », a, « et de », b, « est
égale à », a*b)
|       oper = « / » :
|           Si (b = 0) alors
|               ecrire (« DIVISION IMPOSSIBLE »)
|           Sinon
|               Ecrire (« La division de », a, « par », « est
égale à », a/b)
|           FinSi
|       oper = « - » :
|           ecrire (« La soustraction de », a, « et de », b, « égale à
», a-b)
|   SINON
```

```
|         ecrire (« OPERATEUR INVALIDE »)
|
|   FinSuivant
|
FIN
```

Exercice 12 : Un nombre est parfait s'il est égal à la somme de ses diviseurs stricts (différents de lui-même). Ainsi par exemple, l'entier 6 est parfait car  $6 = 1 + 2 + 3$ . Écrire un algorithme permettant de déterminer si un entier naturel est un nombre parfait.

#### PROGRAMME\_APP12

```
Var cpt, nbSaisi, i : entier
DEBUT
|   ecrire (« Donner un nombre entier : »)
|   lire nbSaisi
|   cpt = 0
|   Pour i allant de 1 à nbSaisi / 2 faire
|       Si (nbSaisi mod i = 0) alors
|           cpt += nbSaisi / i
|       FinSi
|   FinPour
|   Si (cpt = nbSaisi) alors
|       afficher (« Le nombre saisi est un nombre PARFAIT ! »)
|   Sinon
|       afficher (« Le Nombre saisi n'est pas PARFAIT ! »)
|   FinSi
FIN
```

Exercice 13 : Faire un programme qui saisit une date (jour, mois et année) et qui indique si la date est valide

#### PROGRAMME\_APP13

```
Type date = STRUCTURE
DEBUT
|   Jour : 1 .. 31
```

```
    Mois : 1 .. 12
    Annee : entier
FIN
Var ldate : date

DEBUT
|   ecrire (« Donner le jour, le mois et l'annee de la         date à
verifier : »)
|   lire ldate.jour, ldate.mois, ldate.annee
|   Si ((ldate.jour > 1 ET ldate.jour < 31) ET (ldate.mois > 1 ET
ldate.mois < 12)) alors
|       Si (ldate.mois = 2 ET ldate.jour > 29) alors
|           afficher(« DATE INVALIDE ! »)
|       Sinon
|           afficher(« LA DATE SAISIE EST VALIDE ! »)
|       FinSi
|   Sinon
|       afficher(« LA DATE SAISIE N'EST PAS VALIDE ! »)
|   Fin
FIN
```