**ALGORITHMIE TD1 SIMPLON PRAIRIE**

**TD Programmation**

**Marie Perle RAJAONA**

**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).

Programme division

var a, b, quotient, reste : entier

var ratio : reel

Debut

Ecrire (“saisir deux entier a et b avec b différent de 0”)

Lire (a, b )

quotient <- a div b

reste <- a mod b

ratio <- a/b

afficher (“le quotient est : ”, quotient)

afficher (“le reste est : “, reste)

afficher (“la ratio est : “, ratio)

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.

Programme Cercle

const PI = 4 \* atan(1)

var rayon, perimetre, surface : reel

debut

ecrire (“saisir le rayon d’un cercle”)

lire (rayon)

surface <- PI\*rayon\*rayon

perimetre <- 2\*PI\*rayon

afficher (“la surface du cercle est : “, surface)

afficher (“le perimetre du cercle est : “, perimetre)

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.

Programme resistance1

var r1, r2, r3, rS, rP : reel

debut

ecrire (“saisir trois valeurs r1, r2 et r3 “)

lire (r1, r2, r3 )

rS <- (r1+ r2+ r3)

rP <- (r1 \* r2 \* r3) / (r1\* r2 + r2\*r3 + r1\*r3)

afficher (“la resistance en série est : “, rS)

afficher (“la resistance en parallele est : “, rP)

fin

Programme resistance2

var T, f , rs, rp : entier

choice : reel

Début

Ecrire (“saisir 1 ou 2” )

Lire (choice)

rs ← r1+r2+r3

rp ← (r1 \* r2 \* r3) / (r1\*r2 + r2\*r3 + r1\*r3)

Selon que

choice = 1 Ecrire (“la resistance en série est :” , rs )

choice = 2 Ecrire (“la resistance en série est :” , rp )

Sinon ecrire (“ le choix est invalide “ )

FinSelon

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 un boucle

Programme puissance

var x : reel

n : entier

debut

Si n = 0 alors

pow ( x, n ) ← 1

sinon

calculer <-pow (x, n )

fin

Programme puissanceBoucle

var x , puissance : reel

n, i : entier

Debut

Ecrire (“Saisir un reel x et un entier n “ )

lire (x, n )

puissance ← 1

Pour i allant de 1 à n Faire

puissance ← puissance \* x

FinPour

Afficher (“ x à la puissance n est : “ , puissance )

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).

Programme sommeEntierBoucle

Var somme : entier

n = ( 5 )

Debut

Pour

n ← allant de 1 à 5

Faire somme ← (n + 1 )

Afficher ( n )

FinPour

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 coordonneesAB

Var x1, y1, x2, y2, d : entier

Debut

Ecrire (“Donner une valeur à x1” )

Lire (x1)

Ecrire (“Donner une valeur à y1” )

Lire (y1)

Ecrire (“Donner une valeur à x2” )

Lire (x2)

Ecrire (“Donner une valeur à y2” )

Lire (y2)

d ←sqrt ( sqr (x1 – x2) + sqr (y1 – y2) )

Afficher (“la distance entre A et B est : “ d )

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 Monnaie

var montant20, montant10, montant5, montant2, montant1 : entier

Debut

Ecrire (“Saisir le montant de la monnaie à rendre “ )

Lire ( montant ) 500

montant20 ← montant Div 20

montant10 ← ( montant mod 20 ) div 10

montant5 ← ( montant10 mod 10 ) div 5

montant2 ← ( montant5 mod 5 ) div 2

montant1 ← ( montant2 mod 2 ) div 1

Afficher (“Le nombre de billet de 20 est de : montant20“ )

Afficher (“Le nombre de billet de 10 est de : montant10“ )

Afficher (“Le nombre de billet de 5 est de : montant5“ )

Afficher (“Le nombre de pièce de 2 est de : montant2“ )

Afficher (“Le nombre de pièce de 1 est de : montant1“ )

Fin

**Exercice 8 :**

Ecrire un algorithme permettant de résoudre une équation du second degré.

Ax2 + bx + c = 0

Programme equationSecondDegre

var a, b, c, D : reel

Debut

Ecrire(“Donner la valeur de a, b et c ”)

Lire (a, b, c )

Si a = 0 Alors

Si b = 0 alors Ecrire (“Pas de solution ”)

Sinon Ecrire ( “une solution : “ -c/b )

FinSi

Sinon

D ← (b \* b) - (4 \* a \* c )

Si a < 0 Alors

Ecrire (“Pas de solution “)

Sinon

Si a = 0 Ecrire (“La solution est : “ (-b/ a \* a )

Sinon Ecrire (“Deux solutions existent : “ (-b-(sqrt(D)))/(2 \* 2 ) “et” ( (-b+(sqrt(D)))/(2 \* 2 )

FinSi

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.

a. On considère que le départ et l'arrivé ont lieu le même jour

b. On suppose que la durée de vol est inférieure à 24 heures mais peut avoir lieu le lendemain.

Programme dureeDeVol

Var hd, md, ha, ma : entier

hdu, mdu : entier /\*heure duree minute duree \*/

Debut

Ecrire (“donner les horaires de départ h1 et mn1 “ )

Lire (hd, md )

Ecrire (“donner les horaires d’arrivée h2 et mn2 “ )

Lire (ha, ma )

si ma > md alors

hdu ← (ha - hd )

mdu ← (ma - md )

sinon hdu ←(( ha - 1) - hd )

mdu ←( ma + 60 -md ) Ecrire (“la duree du trajet est de :” hdu ; mdu )

FinSi

Fin

Programme dureeDeVol

Var hd, md, ha, ma : entier

hdu, mdu : entier /\*heure duree minute duree \*/

Debut

Ecrire (“donner les horaires de départ h1 et mn1 “ )

Lire (hd, md )

Ecrire (“donner les horaires d’arrivée h2 et mn2 “ )

Lire (ha, ma )

Si ha > hd

SI ma > md

mdu ← (ma -md )

hdu ← ( ha - hd )

Ecrire (“la duree du trajet est de :” hdu ; mdu )

Sinon

mdu ← ma + 60 -md

hdu ← (( ha -1 ) -hd )

Finsinon

ha < hd

SI ma > md

mdu ← (ma -md )

hdu ← ( ha - hd + 24 )

Ecrire (“la duree du trajet est de :” hdu ; mdu )

Sinon

mdu ← ma + 60 -md

hdu ← (( ha -1 ) -hd )+24 Ecrire (“la duree du trajet est de :” hdu ; mdu )

Finsinon

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 TriSuccessif

var a, b, c : entier

Debut

Ecrire (" entrer les valeurs a , b et c ")

Lire(a,b,c)

Si (a > b) alors

Echange (a,b)

Si b > c alors

Echange (b,c)

Si a > b alors

Echange (a,b)

Finsi

Finsi

Sinon

Si b > c alors

Echange (b,c)

Si a >b alors

Echange (a,b)

Finsi

Finsi

Finsi

Ecrire ("les valeurs a , b et c sont (

Fin

**Exercice 11 :**

Ecrire un algorithme calculatrice 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 operationCalculatrice

var a, b : entier

op : caractere

Debut

Ecrire (" saisir le premier entier ")

Lire (a)

Ecrire (" saisir l’opération ")

Lire (op)

Ecrire (" saisir le deuxième entier")

Lire (b)

Selonque :

op = ‘+’ : ecrire ("la somme de ",a, "et de ",b, "est égale",a+b)

op = ‘\*’ : ecrire ("le produit de ",a, "et de ",b, "est égale",a\*b)

op = ‘/’ : si (b= 0) alors ecrire (" division impossible ")

sinon ecrire ("la division de ",a, "par ",b, "est égale", a/b)

finsi

op = ‘-‘ : ecrire ("la soustraction de ",a, "et de ",b, "est égale", a-b)

finselonque

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 nombreParfait

var N, s, d, r : entier

Debut

Ecrire (“Saisir un nombre entier “ )

Lire (N)

s ← 0

Pour d ← 1 à (N div 2 ) Faire

r ← N % d

si r = 0 Alors

s ← s + d

FinSi

Afficher ( N “est un nombre parfait “ )

Fin

**Exercice 13 :**

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

Programme Calendrier

var a, m, j, c : entier

Debut

Ecrire (“ Saisir un jour, un mois et une année après 1582“ )

Lire ( j, m, a )

Pour a ← 1582 à n

Si m<1 ET m>12

Si (j = 30) ET ( (m = 04) OU (m = 6) OU (m = 9) OU (m = 11))

Si (j = 29) ET (m= 2)

Si (j = 28) ET ((m= 2) et ((a mod 4 = 0 ET a mod 100 !=0) OU ( a mod 400 =0))

alors Ecrire (“la date est valide “)

FinPour

Fin