Université Sultan Moulay Slimane
Ecole National des Sciences Appliquées
Khouribga

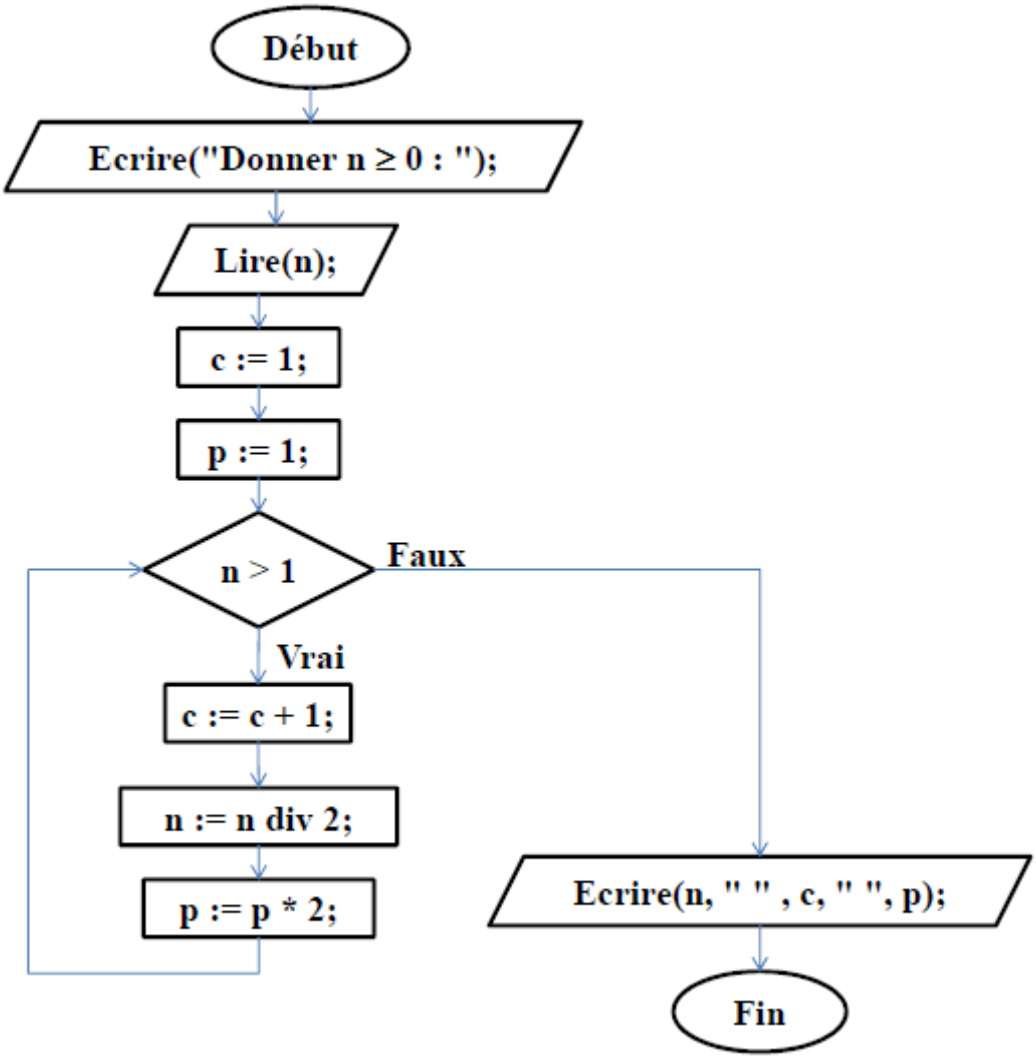
Module : Architecture des ordinateurs & Algorithmique

1ère API / S1 / Année 2018/2019

Solution du Contrôle Continu N° 2 (Janvier 2019)

Exercice 1

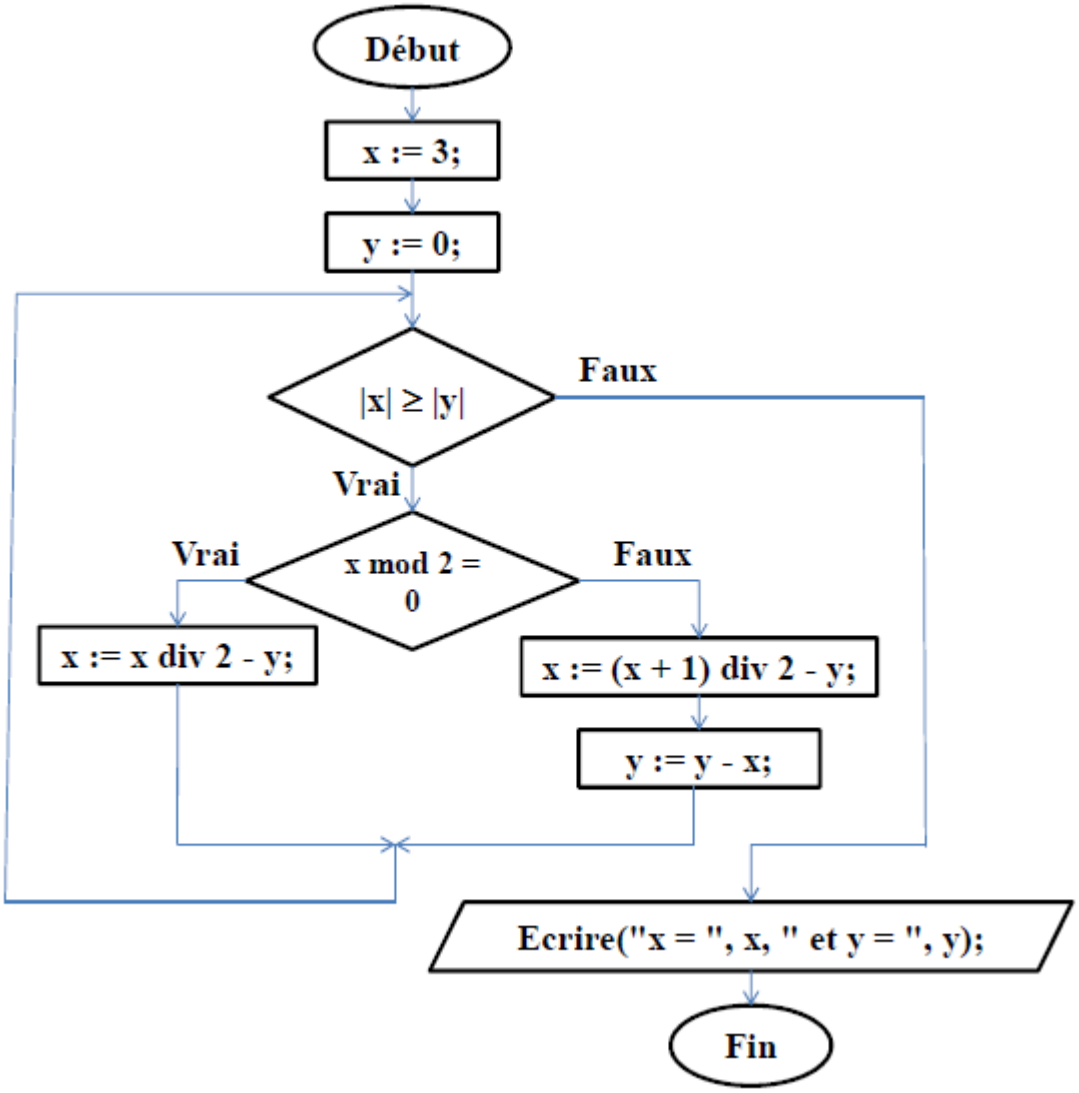
1. L'organigramme correspondant à l'algorithme 1 :



2. L'algorithme 1 affichera les valeurs suivantes :

- 1 2 2, lorsque n = 2
- 1 4 8, lorsque n = 10
- 1 6 32, lorsque n = 32

3. L'organigramme correspondant à l'algorithme 2 :



4. Les valeurs respectives des deux variables x et y dans l'algorithme 2 :

Instant de l'algorithme	Valeur de x	Valeur de y
Juste avant la boucle « TQ »	3	0
Après la 1ère itération « TQ »	2	-2
Après la 2ème itération « TQ »	3	-2
Après la 3ème itération « TQ »	4	-6

L'algorithme 2 affichera :

x = 4 et y = -6

Exercice 2

```
Algorithme Signe_D_Un_Trinôme ;
    Var
        a, b, c, x, delta, x1, x2, t, dx : Réel ;
        signe_a, signe_fx : Entier ;
Début
    { Saisie des coefficients du trinôme f(x) = ax² + bx + c }
```

```
Ecrire("Saisir les valeurs des coefficients du trinôme a, b et c : ");
Lire(a, b, c);

{ Saisie d'une valeur pour un nombre réel x }
Ecrire("Saisir la valeur de x : ");
Lire(x);

{ Affichage du discriminent Δ = b² - 4ac de l'équation f(x) = ax² + bx + c = 0 }
delta := b * b - 4 * a * c;
Ecrire("Le discriminent de f(x) = 0 est : ", delta);

{ Mémorisation du signe de a dans la variable signe_a }
Si (a < 0) Alors
   signe_a := 1;
Sinon
   signe_a := -1;
FinSi

{ Détermination du signe de f(x) dans la variable signe_fx }
Si (delta < 0) Alors
   signe_fx := signe_a  { Si Δ < 0, le signe de f(x) est celui de a }
Sinon
   Si (delta = 0) Alors
      x1 := -b / (2 * a);
      Si (x = x1) Alors { Si Δ = 0, f(x) = 0 pour x = x1 }
         signe_fx := 0;
      Sinon { Si Δ = 0, le signe de f(x) est celui de a, pour x ≠ x1 }
         signe_fx := signe_a;
      FinSi
   Sinon
      x1 := (-b - Racine(delta)) / (2 * a);
      x2 := (-b + Racine(delta)) / (2 * a);
      Si (x1 > x2) Alors  { Pour avoir x1 < x2 }
         t := x1;
         x1 := x2;
         x2 := t;
      FinSi
      Si (x < x1 OU x > x2) Alors
         { Si Δ > 0, le signe de f(x) est celui de a, pour x à l'extérieur des
         racinces }
         signe_fx := signe_a;
      Sinon
         Si (x = x1 OU x = x2) Alors
            { Si Δ > 0, f(x) = 0, pour x = x1, ou x = x2 }
            signe_fx := 0;
```

```
            Sinon
               { Si Δ > 0, le signe de f(x) est le signe contraire de a, pour x à
               l'intérieur des racines }
               signe_fx := - signe_a;
            FinSi
         FinSi
      FinSi
   FinSi
   Selon (signe_fx) Faire
      cas -1 : Ecrire ("f(x) est négatif");
      cas 0  : Ecrire ("f(x) est nul");
      cas 1  : Ecrire ("f(x) est positif") ;
   FinSelon

   { Calcul de la dérivée de f(x) au point x et affichage de son signe}
   dx := 2 * a * x + b ;
   Si (dx > 0) Alors
      Ecrire("La dérivée de f au point x est positive");
   Sinon
      Si (dx = 0) Alors
         Ecrire("La dérivée de f au point x est nulle");
      Sinon
         Ecrire("La dérivée de f au point x est négative");
      FinSi
   FinSi
Fin
```

Exercice 3

```
Algorithme Description_Tremblement_Terre ;
   Var
      Magnitude : Réel ;
      peMagnitude : Entier ;
      description : Chaîne ;
Début
   Ecrire("Saisir la valeur du magnitude : ");
   Lire(Magnitude);
   Si (Magnitude < 0.0) Alors
      Ecrire("Magnitude erroné !!!") ;
   Sinon
      peMagnitude := Partie_Entière(Magnitude) ;
      Selon(peMagnitude) Faire
         cas 0 : description := "Micro";
         cas 1 : description := "Micro";
         cas 2 : description := "Très mineur";
```

```
            cas 3 : description := "Mineur";
            cas 4 : description := "Léger";
            cas 5 : description := "Modéré";
            cas 6 : description := "Forte";
            cas 7 : description := "Important";
            cas 8 : description := "Grand";
            cas 9 : description := "Grand";
            Au Défaut : description := "Météorique";
        FinSelon
    FinSi
    Ecrire("Tremblement de terre ", description);
Fin
```

Exercice 4

```
Algorithme Nombres_Chiffres_Tous_Pairs ;
    Var
        u, d : Entier ; { u : chiffre des unités, d : chiffre des dizaines }
Début
    Pour d := 0 A 8 Pas = 2 Faire
        Pour u := 0 A 8 Pas = 2 Faire
            Ecrire(d * 10 + u) ;
        FinPour
    FinPour
Fin
```

Exercice 5

```
Algorithme Des_Traitements_Sur_Un_Tableau ;
    Const
        N : Entier = 10 ;
    Var
        T, SP : Tableau de N Entier ;
        S, P, M, i, j, imin, dernier, infM, egM, supM, temp, z, nbFois : Entier ;
        ppeT : Entier ;
Début
    { Saisie des éléments du tableau T }
    Pour i := 1 A N Faire
        Ecrire("Donner T[", i , "] : ") ;
        Lire(T[i]) ;
    FinPour

    { Calcul et affichage de la somme S, du produit P et de la moyenne M des
    éléments de T }
```

```
S := 0 ;
P := 1;
Pour i := 1 A N Faire
   S := S + T[i] ;
   P := P * T[i] ;
FinPour
M := S div N ;
Ecrire("La somme des éléments de T est : " , S) ;
Ecrire("Le produit des éléments de T est : " , P) ;
Ecrire("La moyenne des éléments de T est : " , M) ;

{ Calcul et affichage du nombre des éléments de T qui sont inférieurs à M, égaux à M et supérieurs à M }
infM := 0 ;
supM := 0 ;
Pour i := 1 A N Faire
   Si(T[i] < M) Alors
      infM := infM + 1 ;
   Sinon
      Si(T[i] > M) Alors
         supM := supM + 1 ;
      FinSi
   FinSi
FinPour
Ecrire("Nombre des éléments de T inférieurs à ", M, " :  " , infM) ;
Ecrire("Nombre des éléments de T égaux à ", M, " :  " , N - infM - supM) ;
Ecrire("Nombre des éléments de T supérieurs à ", M, " :  " , supM) ;

{ Détermination et affichage de l'indice du plus petit élément de T, puis substitustion de ce plus petit élément de tous les éléments de T}
imin := 1 ;
Pour i := 2 A N Faire
   Si (T[i] < T[imin]) Alors
      imin := i ;
   FinSi
FinPour
Ecrire("L'indice du plus petit élément de T est : ", imin) ;
ppeT := T[imin] ;
Pour i := 1 A N Faire
   T[i] := T[i] - ppeT;
FinPour

{ Recopie de la moitié gauche de T dans sa moitié droite, et vice-versa (en gardant l'ordre des éléments dans les deux moitiés du tableau T}
Pour i := 1 A (N div 2) Faire
```

```
    temp := T[i];
    T[i] := T[i + (N + 1) div 2];
    T[i + (N + 1) div 2] := temp;
FinPour
```

```
{ Décalage des éléments de T d'une case vers la droite : le 1èr élément devient le 2ème élément, le 2ème élément devient le 3ème élément, …, l'avant dernier élément devient le dernier élément, et le dernier élément devient le 1èr élément}
dernier := T[N] ;
Pour i := N - 1 A 1 Pas = - 1 Faire
    T[i + 1] := T[i];
FinPour
T[1] := dernier ;
```

```
{ Saisie d'un nombre entier z et recherche si le tableau T contient au moins deux fois l'élément z}
Ecrire("Donner un entier z : ") ;
Lire(z) ;
nbFois := 0 ;
Pour i := 1 A N Faire
    Si (T[i] = z) Alors
        nbFois := nbFois + 1 ;
    FinSi
FinPour
Si (nbFois >= 2) Alors
    Ecrire("L'élément ", z , " existe au moins deux fois dans T") ;
Sinon
    Ecrire("L'élément ", z , " n'existe pas au moins deux fois dans T") ;
FinSi
```

```
{ Stockage dans un tableau secondaire, les sommes partielles des éléments de T}
SP[1] := T[1] ;
Pour i := 2 A N Faire
    SP[i] := SP[i - 1] + T[i];
FinPour
```