Programme informatique = Problème

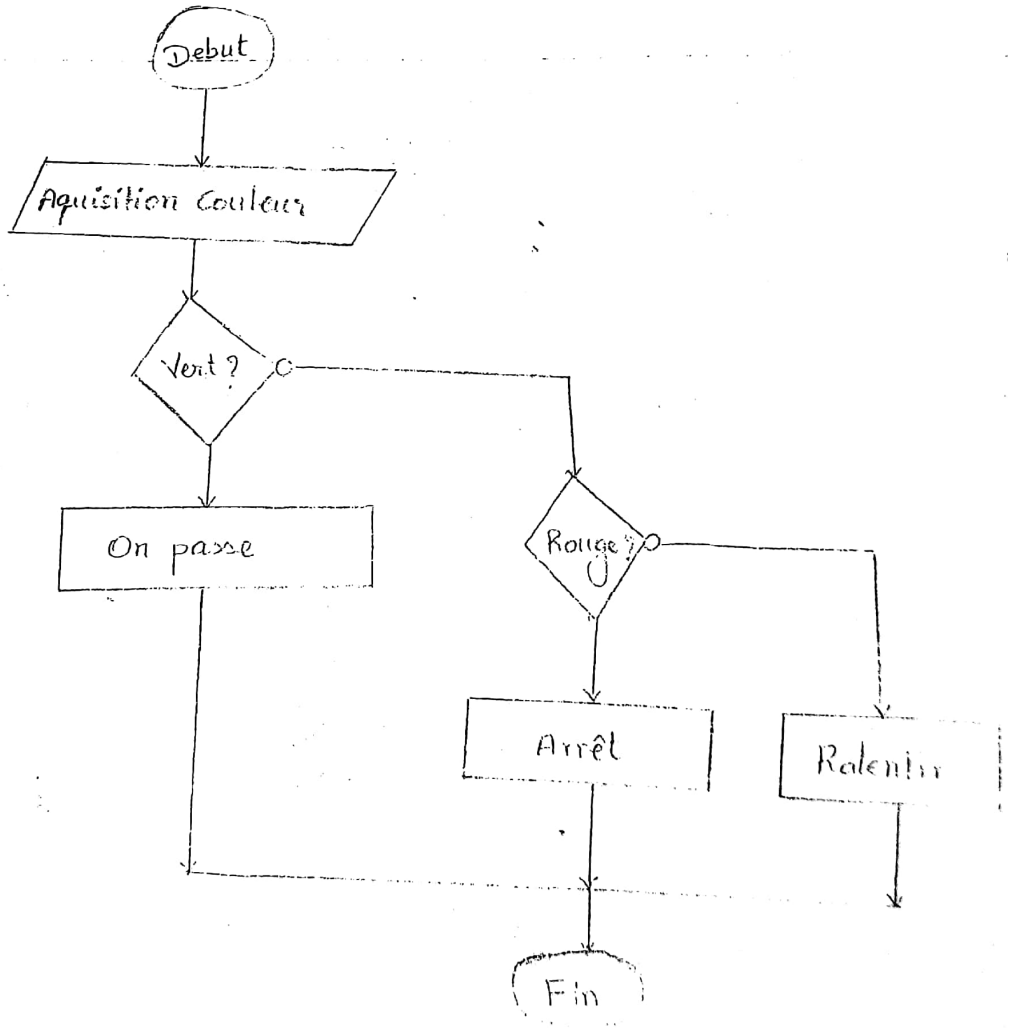
Analyse, Recherche de la solution : algorithme :
50% Evaluation

Codification dans un langage compréhensible par la machine (voir historique).

Sous forme de graphe

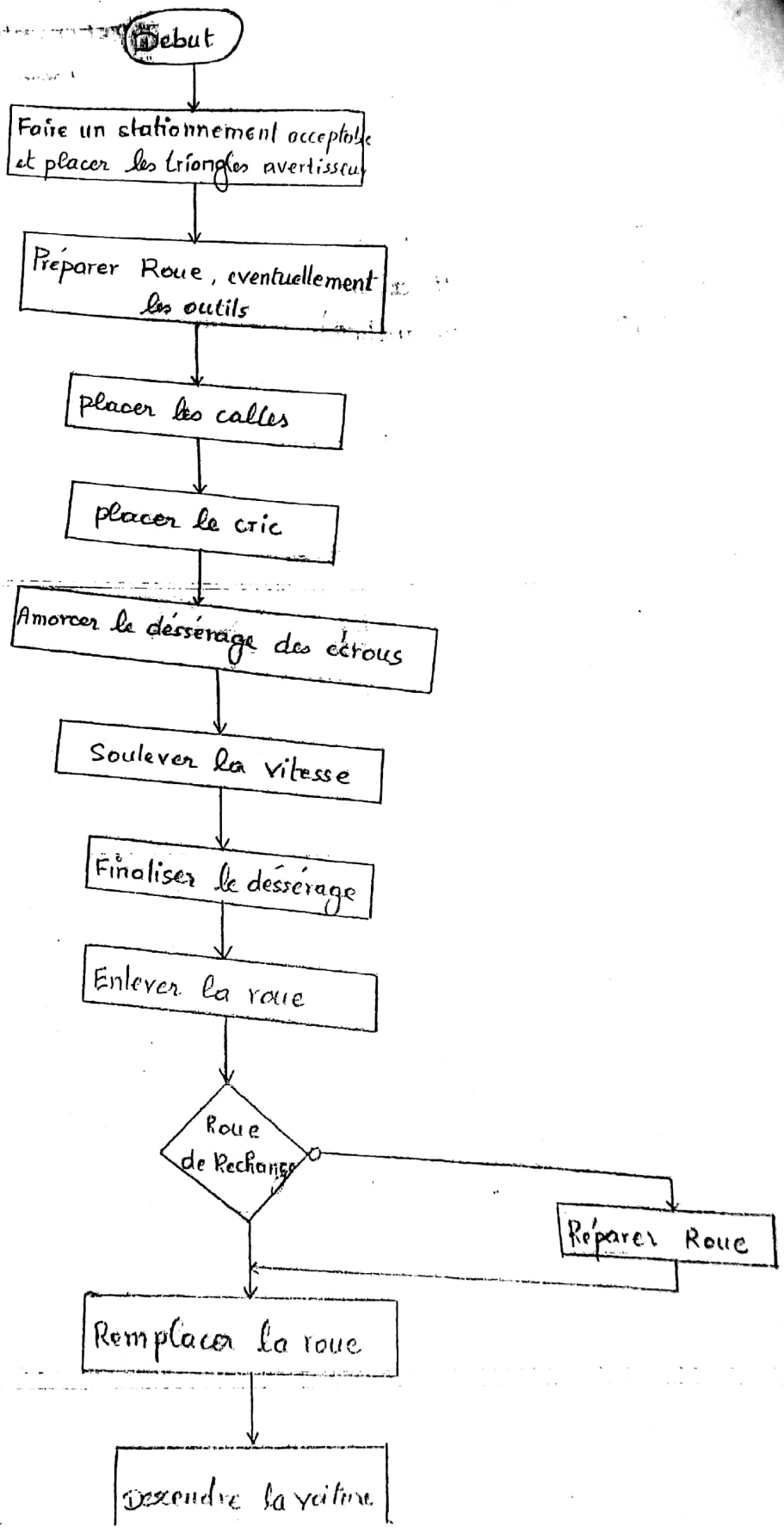
Sous forme littéraire (instruction)

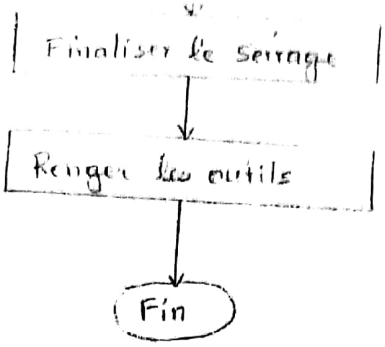
Feu tricolore :



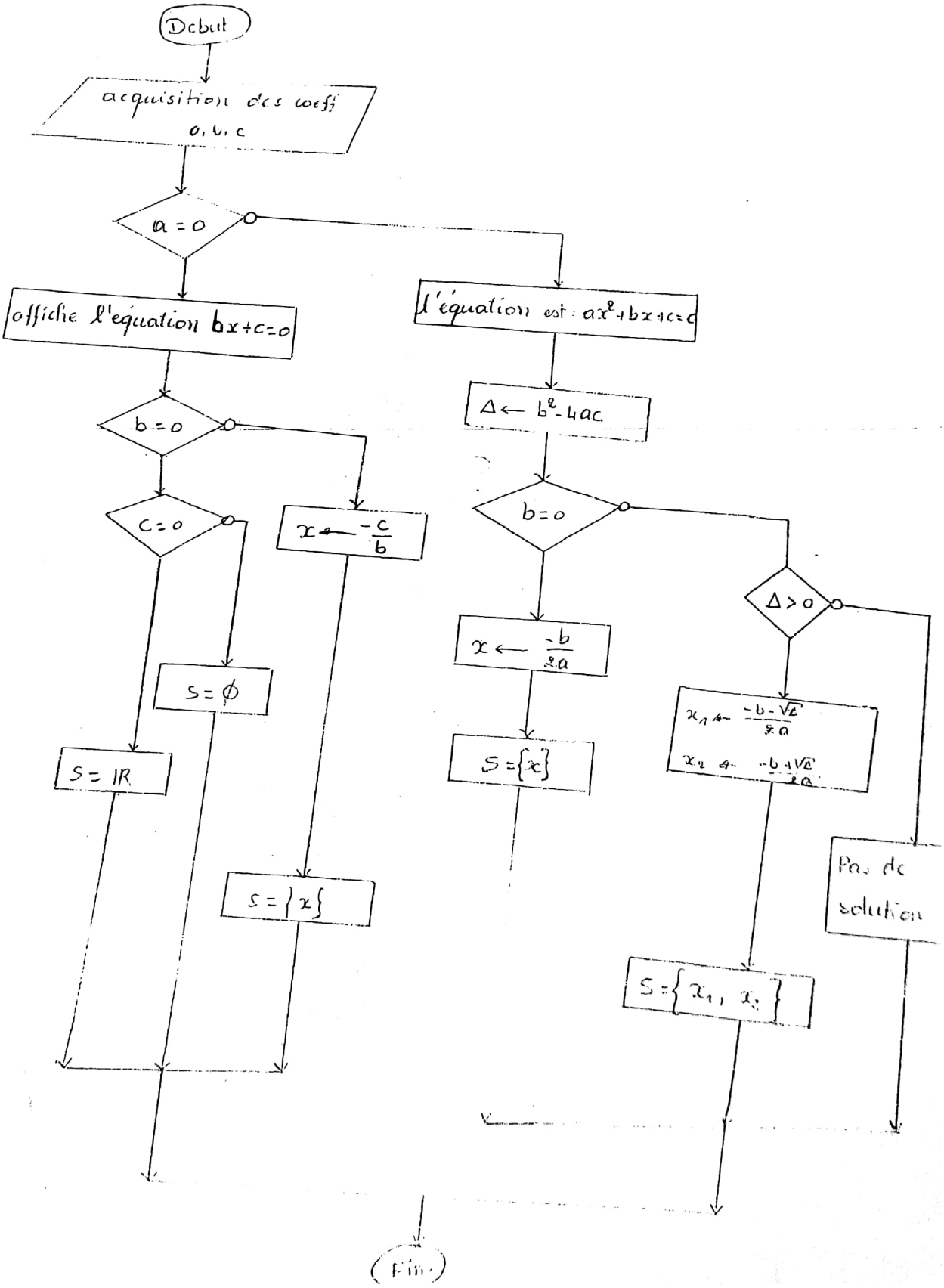
Remplacement d'une roue crevée :

Pour se faire on a besoin :
- Roue
- Démonte Roue
- cric
- Avertisseurs triangle
- calles et support de cric.





Equation du second dégré : $ax^2+bx+c=0$



// Programme :

```
Program equation_du_second_degré;
  var a, b, c, Δ, x1, x2 : réel;

Debut
    Afficherln('Ce programme résout l'equation ax²+bx+c=0');
    Afficher('Donnez les differents cœfficients a, b, c :');
    Saisirln(a, b, c);
    Afficher('Voici votre équation :', a, 'x²+', b, 'x+', c, '=0');
    Si a=0
      alors Debut
              Afficherln('votre equation est :', b, 'x+', c, '=0');
              Si b=0
                alors si c=0
                        Afficherln('S=IR')
                        sinon afficherln('S=ϕ');
                      Fsi
                sinon Debut
                        x ⟵ -c/b;
                        Afficherln('S={', x, '}');
                      Fin;
              Fsi;
            Fin
      Sinon Debut
              Afficherln('L'equation est :', a, 'x²+', b, 'x+', c, '=0');
              Δ ⟵ sqr(b) - 4*a*c;
              Si Δ=0
                alors Debut
                        x ⟵ -b/2*a
                        Afficherln('S={', x, '}');
                      Fin
                sinon si Δ>0 Debut
                        x1 ⟵ (-b - sqrt(Δ))/2*a;
                        x2 ⟵ (-b + sqrt(Δ))/2*a;
                        Afficherln('S={', x1, ';', x2, '}');
                      Fsi
                      Fin;
              Fsi;
            Fin;
    Sinon Afficherln('Pas de solution dans IR');
Fin.
```

Pos (chaine1, chaine2, nombre); permet de rechercher chaine2 dans chaine1 à partir de la position d'un nombre de chaine. Si chaine est trouvée, alors ~~sschaine~~ Pos renvoie la position du 1er caractère de chaine2 dans chaine1; sinon ~~sschaine~~ Pos renvoie 0.

Exple: Pos('ouagadougou', 'ouaga', 2); = 0

Concat (chaine1, chaine2), fait la concatenation de chaine1 à chaine2.

Exemple: Concat ('salut', 'les amis') = 'salut les amis'

sschaine: (chaine, P_1, nombre); extrait un nombre de caractère à partir de la position P_1 de chaine.

Exemple: sschaine ('informatique', 3, 6) = 'format'.

Str (nbre; s); | nbre = nombre
s = chaine de caractère.

str convertit nombre en chaine de caractère et le stocke dans s.

str (370, s) => s '370'.

Ord (caractère); restitue le code Asci du caractère.

Exemple: Ord ('B') => 66.

Chr (nombre); renvoie le caractère associé au nombre.

Exemple: Chr (66) => 'B'.

Instructions élémentaires:

* Affectation:
indicateur ← valeur;
(variable)

VI

Exemple: VAR a : entier ;

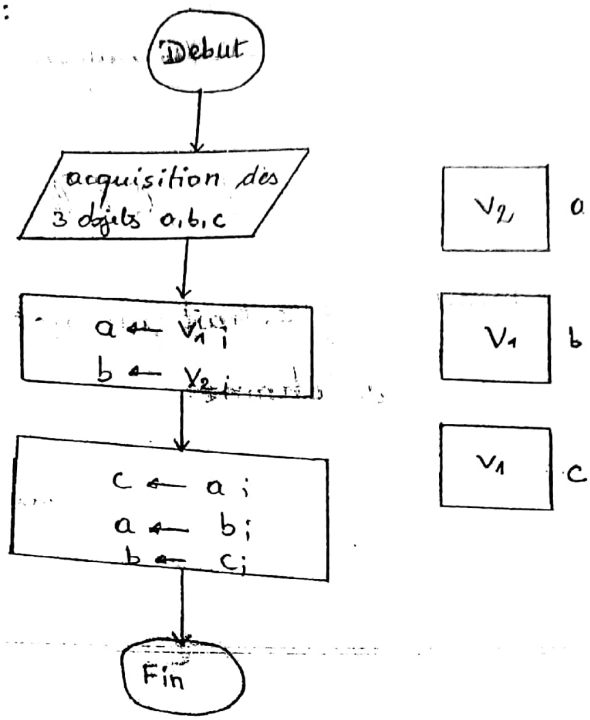
Debut

a ← 10

Fin.

Exemple: Elaborer un algo qui permute 2 réels.

1. logigramme :



2. Programme :

Program permute_2_valeurs ;

Var a,b,c : reel;

Debut

a ← 6.57 ;

b ← 100 ;

{Permutation}

c ← a ;

a ← b ;

b ← c ;

Fin.

Saisir (id1 , id2 . .);

saisir (id1) ;
saisir (id2) ;
⋮
saisir (idn) ;

Saisirln (id1 ; id2 , ...) ;

id : objet déclaré variable obligatoirement.

saisirln, après la saisie ramène le curseur à la ligne (suivante). ~~manuelle~~.

NB: La saisie manuelle consiste à introduire la valeur par le clavier. Cette opération bloque l'exécution du programme tant qu'elle n'est pas validée par la touche entrée.

Opération d'affichage :

Pour publier des commentaires et/ou les resultats, nous utilisons l'instruction afficher ou afficherln.

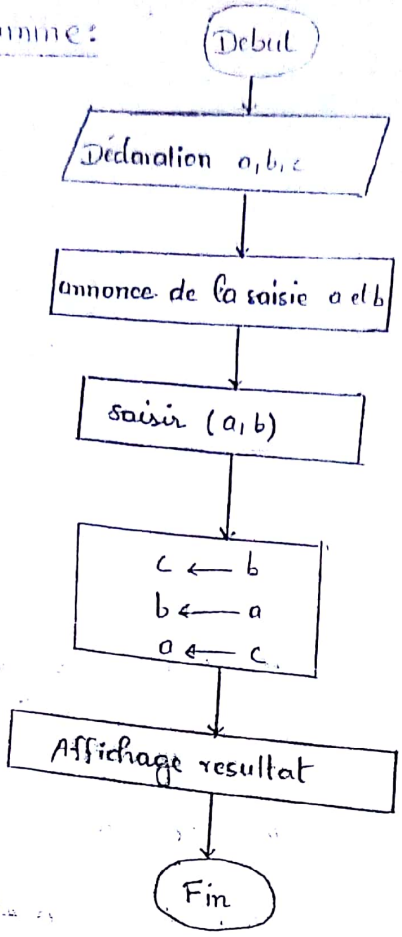
Syntaxe : afficher (el1 ; el2 , el3 . .) ;

afficherln (element1 ; élément2 . . .) ;

Avec afficherln, après affichage des éléments, le curseur revient à la ligne suivante.

→ Reprendre le programme précédent en permettant de choisir les valeurs et d'afficher les résultats.

L'algigramme:



Program Permuteversion2 ;

Var a, b, c : entier ;

Debut

Afficherln ('Ce programme permute 2 entiers');

Afficher ('Veuillez introduire 2 entiers:');

Saisirln (a, b);

Afficherln ('Votre premier entier est', a, 'et votre second est', b);

c ← b ;

b ← a ;

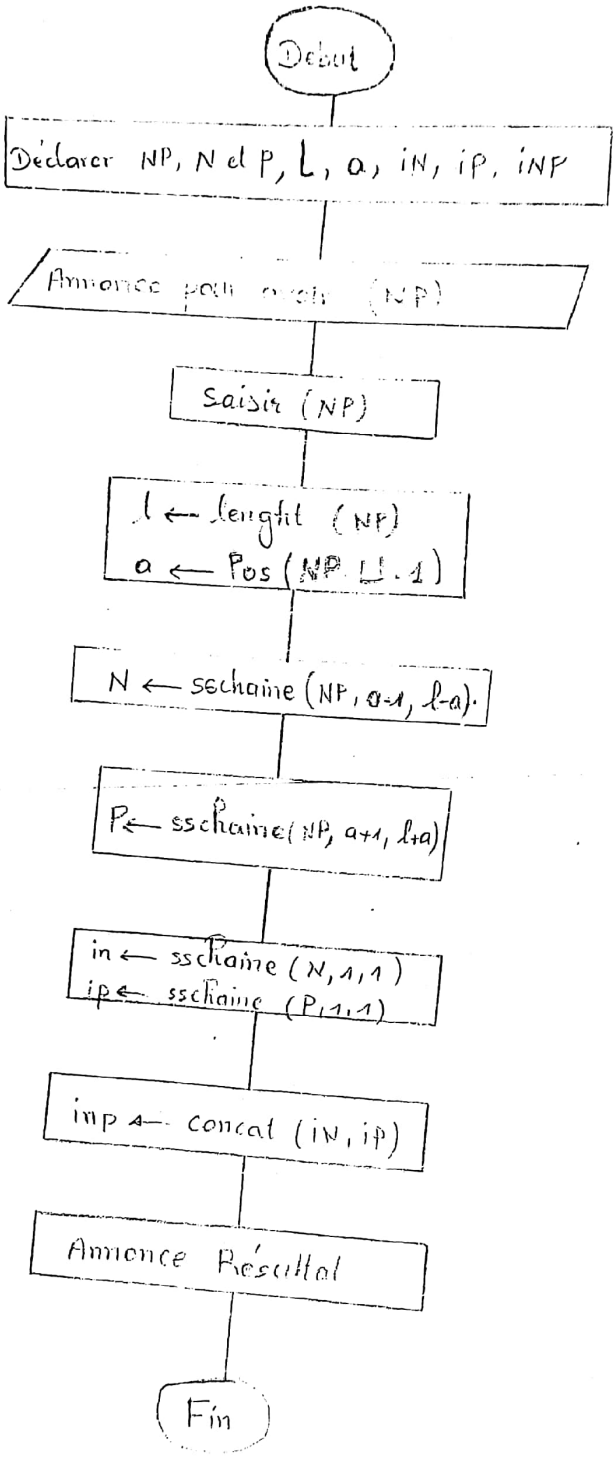
a ← c;

Afficherln ('Après permutation votre premier entier devient', a, 'et votre second devient', b);

Fin.

Réaliser un programme qui permet de saisir le nom et le prénom d'une personne et d'afficher les initiales.

Logigramme :




```
Program Nom_Prenom ;
  Var a, L : entiers ;
      NP, N, P, iN, iP, iNP : chaine de caractère ;
```

Debut

afficheln ('Ce programme permet d'afficher le nom et le pr

afficher ('Veuillez saisir votre nom et prenom séparé d'espace');

saisirln (NP);

afficher ('Vous vous appelez:', NP);

L ← length (NP);

a ← Pos (NP, ␣, 1);

N ← sschaine (NP, 1, a-1);

P ← sschaine (NP, a+1, L-a);

iN ← sschaine (N, 1, 1);

iP ← sschaine (P, 1, 1);

iNP ← Concat (iN, iP);

afficheln ('Votre nom est:', N, 'et comporte', a-1, 'caracteres.');

afficherln ('Votre prenom est:', P, 'et comporte', l-a, 'caracteres.');

afficherln ('L'initiale de votre nom est:', iN);

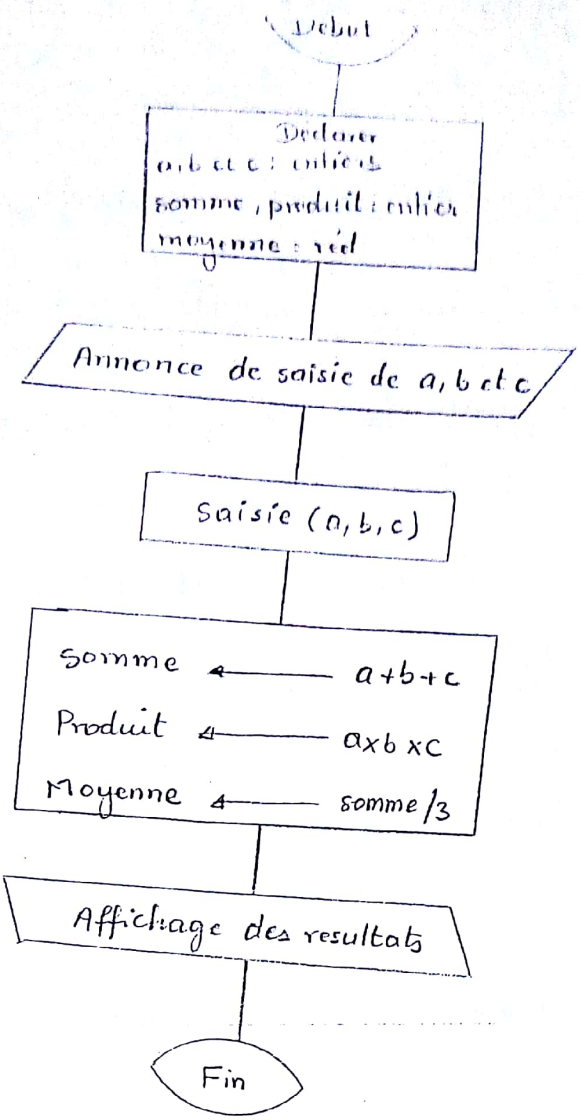
afficheln ('L'initiale de votre prénom est:', iP);

afficheln ('Vos initiales sont:', iNP);

Fin.

③ Elaborer un programme qui permet de calculer et d'afficher la somme, le produit et la moyenne de 3 entiers.

* Logigramme :




```
Program Calcul;
  Var a, b, c : entiers ;
      somme, produit : entiers ;
      moyenne : réel;

Debut
  afficherln ('Ce programme fait le calcul sur 3 entiers.');
  afficher ('Introduiser vos 3 entiers :');
  saisirln (a, b, c);
  Somme ← a + b + c ;
  Produit ← a * b * c ;
```

```
moyenne ← somme /3 ;
afficherln ( a, '+', b, '+', c, '=', somme );
afficherln ( a, '*', b, '*', c, '=', produit);
afficherln ('(', a, '+', b, '+', c, ')/3', '=', moyenne );
afficherln ('La moyenne de', a, ',', b, 'et', c, 'est :' moyenne);
Fin.
```

Structuce conditionnelle:

Syntaxe:

```
Si condition
   alors action ;
fsi ;
```

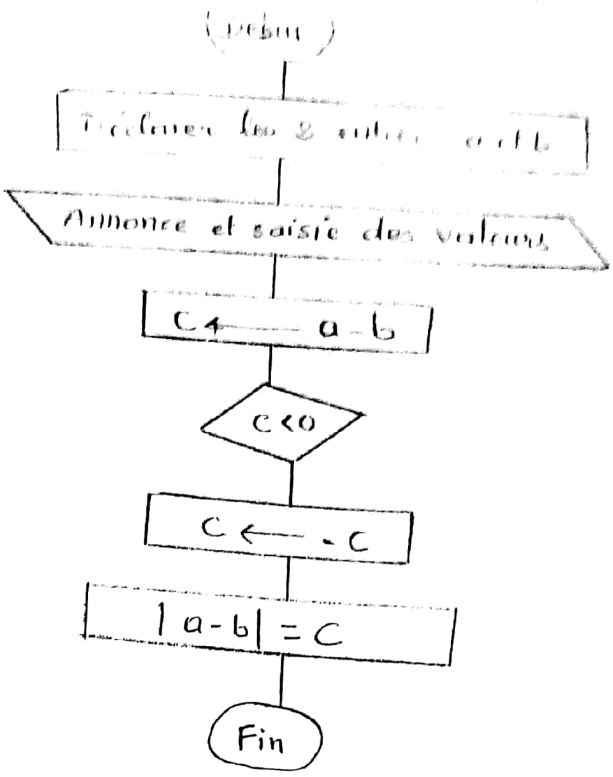
- Si la condition est vraie, alors on éxecute l'action.
- si l'action a plusieurs instructions, alors on les limite avec début et fin.

Exemple:

```
Si a<0
   alors debut
            · afficher ('c''est un nombre negatif');
            a ← a+1 ;
            Fin;
Fsi.
```

④ Elaborer un programme qui permet de calculer la |a-b|.

* logigramme:



* Program Valeur_abs;

Var a, b, c : entiers;

Debut

Afficherln ('Ce programme calcule |a-b|');

Afficher ('Veuillez introduire a et b :');

Saisirln (a, b);

c ⟵ a-b

Si c<0

alors c ⟵ -c ;

Fsi;

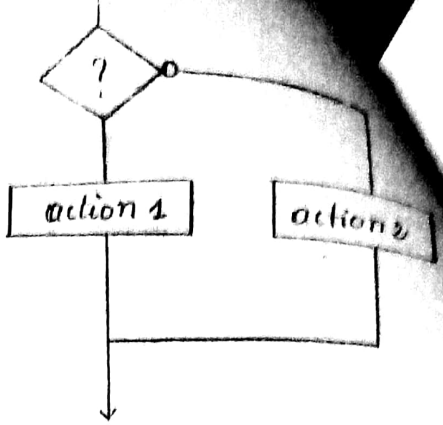
Afficher ('|', a, '-', b, '|=', c);

Fin.

Structure alternative :

Syntaxe :

```
Si condition
    alors action 1;
    sinon action 2;
Fsi;
```



Exemple: Si a<0

alors afficher ('Il est négatif');
sinon afficher ('Il est positif');
fsi;
Fin.

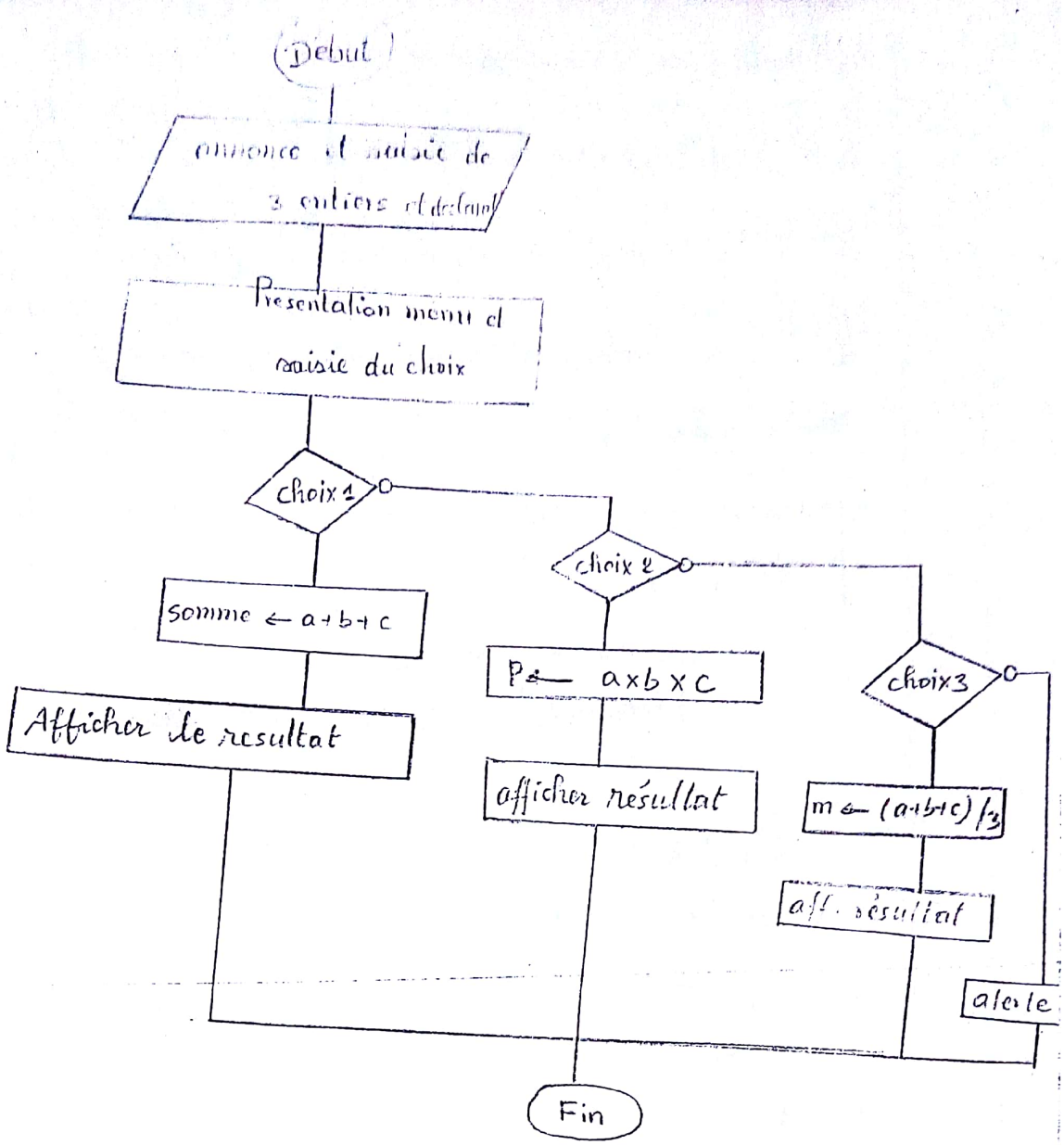
Structure du choix:

syntaxe:

objet déclaré obligatoirement variable de type scalaire (- entier ou - caractè

```
Suivant (Variable) de
        V1: action 1;
        V2: action 2;
        ⋮    ⋮
        Vn: actionn;
Sinon   action 0;
fsuivant;
```

⑤ A partir d'un menu affiché à l'écran, effectuer la somme ou le produit ou la moyenne de 3 nombres (entiers). Nous appelons menu l'association d'un numéro séquentiel aux différents cho proposés par un programme.




```
Program Calcul_v3 ;
    Var a, b, c : entier ;
        Choix : entier ;
        Somme : entier ;
        P     : entier ;
        M : reel ;
Debut
    Afficherln ('Ce programme fait le calcul sur 3 entiers.');
    Afficher ('Veuillez saisir vos entiers :');
    Saisirln (a, b, c);
```

```
Afficherln ('1: pour addition');
Afficherln ('2: pour multiplication');
Afficherln ('3: pour la moyenne');
Afficher (' Faites votre choix entre 1, 2, et 3:');
Saisir (choix);
   Suivant (choix) de
      1: Debut
           Somme ← a+b+c ;
           afficherln(a,'+', b, '+', c '=', somme);
         Fin;
      2: Début
           P← a*b*c ;
           afficherln (a, 'x', b ,'x', c ,'=', P);
         Fin;
      3: Debut
           M ← (a+b+c) /3 ;
           afficherln ('/', a, '+', b, '+', c, ')/3 =', M);
         Fin
   Sinon afficherln (' Attention votre choix est incorrecte !!');
   Fsuivant.
Fin.
```

Les boucles :

structure Tant que :

Syntaxe :

Tant que condition faire

Debut action;

FTQ;

Programme impair;

Var a: entier;

Debut

Pour a ← 1 Jusqu'à 100 faire

Debut

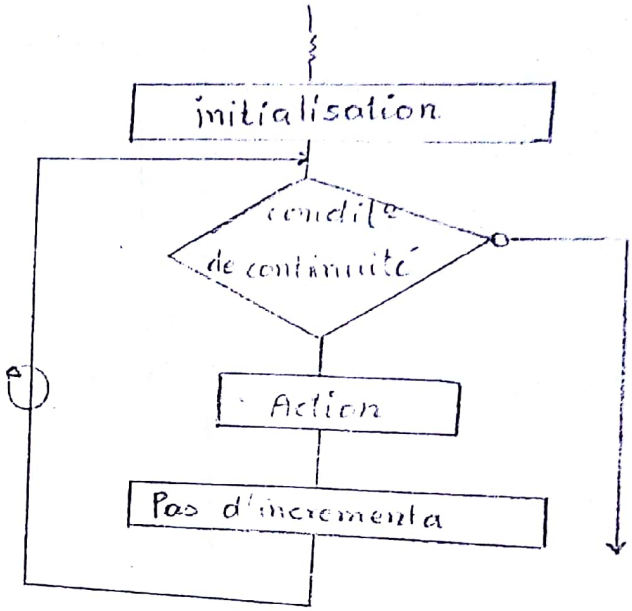
Afficher(a);

a ← a+2;

Fin;

Fpour;

Fin.



Structure Repeter:

Syntaxe:

Repeter

action;

action;

JQA condition

a ← 5;

Repeter

a ← a+3;

JQA a >= 28;

Structure Pour:

Syntaxe:

Pour variable ← valeur initiale JQA finale, faire

Debut

action;

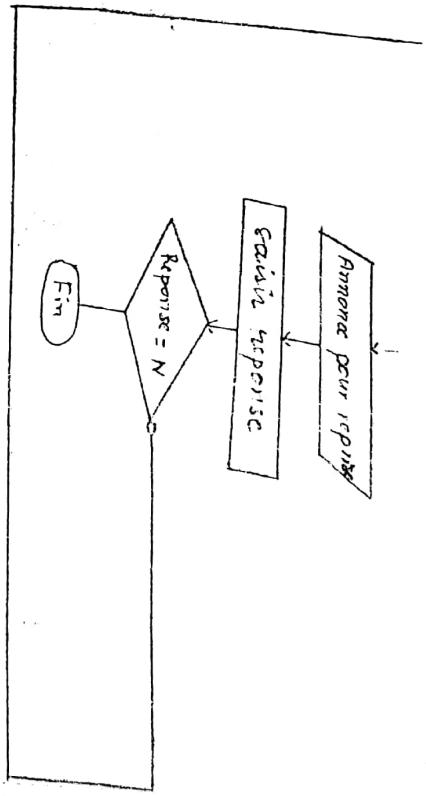
Pas d'incrementation

Fin;

Fpour;

```
Program impaire_V2;
  Var a, i : entier;
Debut
  Afficherln ('Voici des nombres impaires <= 9
  a ⟵ 1;
  Pour i ⟵ 1 JQA 50 faire
    Debut
      Affiche (a);
      a ⟵ a+2;
    Fin;
  Fpour;
Fin.
```

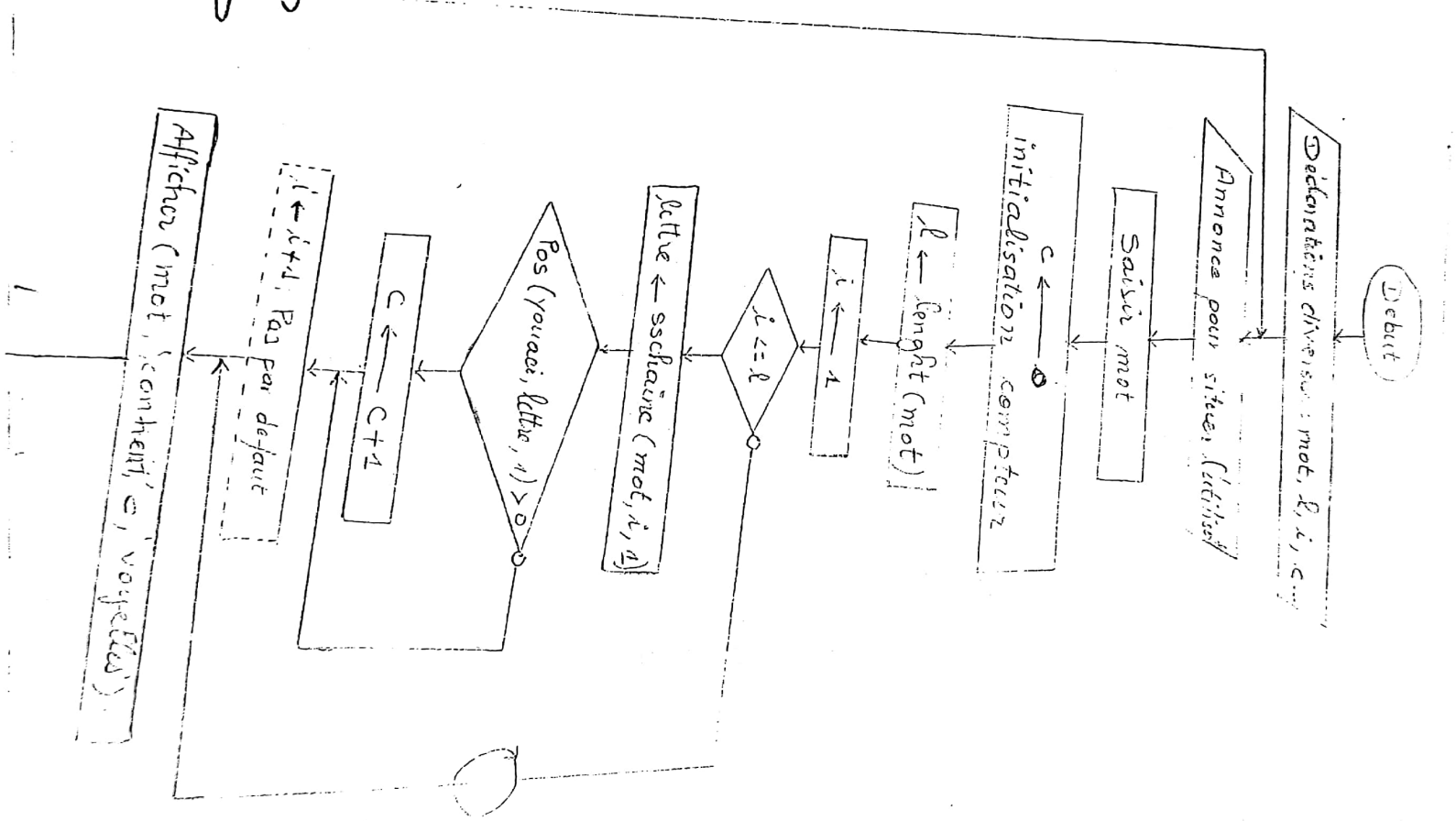
```
Program impaire_V3;
  Var a : entier;
Debut
  Afficherln ('Nombres impaires <= 99');
  a ⟵ 1;
  Repeter
    afficherln (a);
    a ⟵ a+1;
  JQA a <= 99;
Fin.
```



Fin.

(7) Ecrire l'algorithme qui compte le nombre de voyelles d'un mot saisie au clavier.

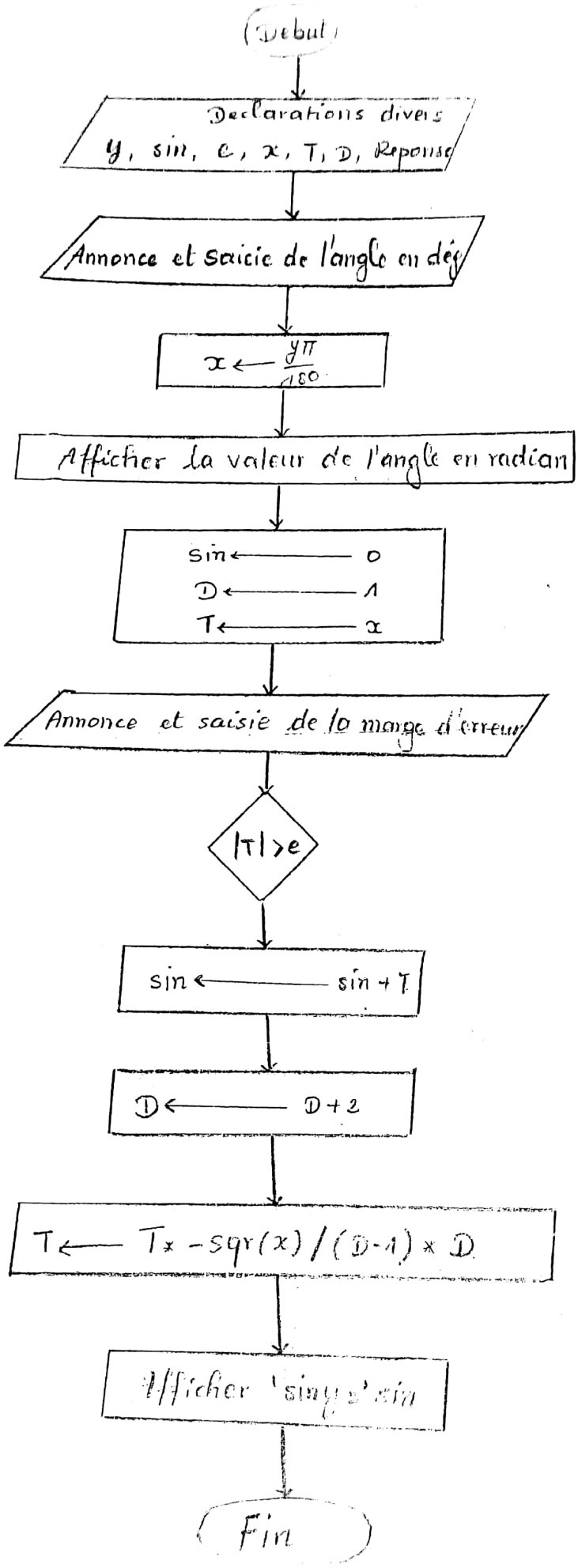
Le programme :



```
Program compteVoyelles ;
  Var mot, lettre : chaine de caractere ;
      i, c, l : entier ;
      Reponse : chaine de caractere ;
Debut
  Repeter
    Afficherln ('Ce programme donne le nombre de voyelles d''un mot : ');
    Afficher ('Donnez votre mot : ');
    Saisirln (mot);
    c ← 0 ;
    l ← lenght (mot);
    Pour i ← 1 JQA l faire
      Debut
        lettre ← sschaine (mot, i, 1) ;
        Si Pos ('youaei', lettre, 1) > 0
          alos c ← c+1 ;
        Fsi ;
      Fin ;
    Fpour ;
    Afficherln (mot, 'contient', c, 'voyelles.');
  Afficher ('Fin du programme. Voulez-vous le reprendre ? o/n ');
  Saisir (Reponse);
  JQA (Reponse = 'n') ou (Reponse = 'N') ;
Fin.
```

(8) Elaborer un programme qui permet de calculer sinx.

On posera $\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \dots + (-1)^r \frac{x^{2r+1}}{(2r+1)!} + \dots$



X Programme calcul_sinx ;

Var y, sin, e, x, T : réel ;

D : entier ;

Reponse : caractère ;

```
Debut
    Repeter
        Afficherln('Ce programme permet de calculer sinx.');
        Afficher('Veuillez introduire y en dégré :');
        Saisirln(y);
        x ← y*PI/180 ;
        Afficher('La valeur de votre angle en radian est', x);
        sin ← 0 ;
        D ← 1 ;
        T ← x ;
        Afficher('Donner votre marge d'erreur :');
        Saisirln(e);
        Tant que (valeur) abs(T)>e faire   (enlever)
            Debut
                sin ← sin + T ;
                D ← D+2 ;
                T ← T* -sqr(x)/(D-1)*D ;
            Fin ;
        FTQ ;
    Afficherln('sin(', y, ')=', sin);
    Afficher('Vous-êtes à la fin de ce programme, voulez-vous le reprendre?');
    Afficherln('oui ou non');
    Saisir(Reponse)
    JQA('reponse=','non');
Fin.
```

Les tableaux

Vecteurs — matrices

Identif — Identif

Syntaxe:

Nom Vect: tableau [indice min ..... indice max] de type de données;

	1	2	3	4	5	6	7
V	'b'	'w'	'c'	'a'	'o'	'y'	'i'

Déclaration:

Var V: tableau [1,7] de caractère;

Designation d'un élément:

V[4] = o

Affectation d'un élément:

V[5] ← 'o';

Saisie d'un élement:

saisir (V[7]);

Affichage d'un élement:

Afficher (V[2]); → w.

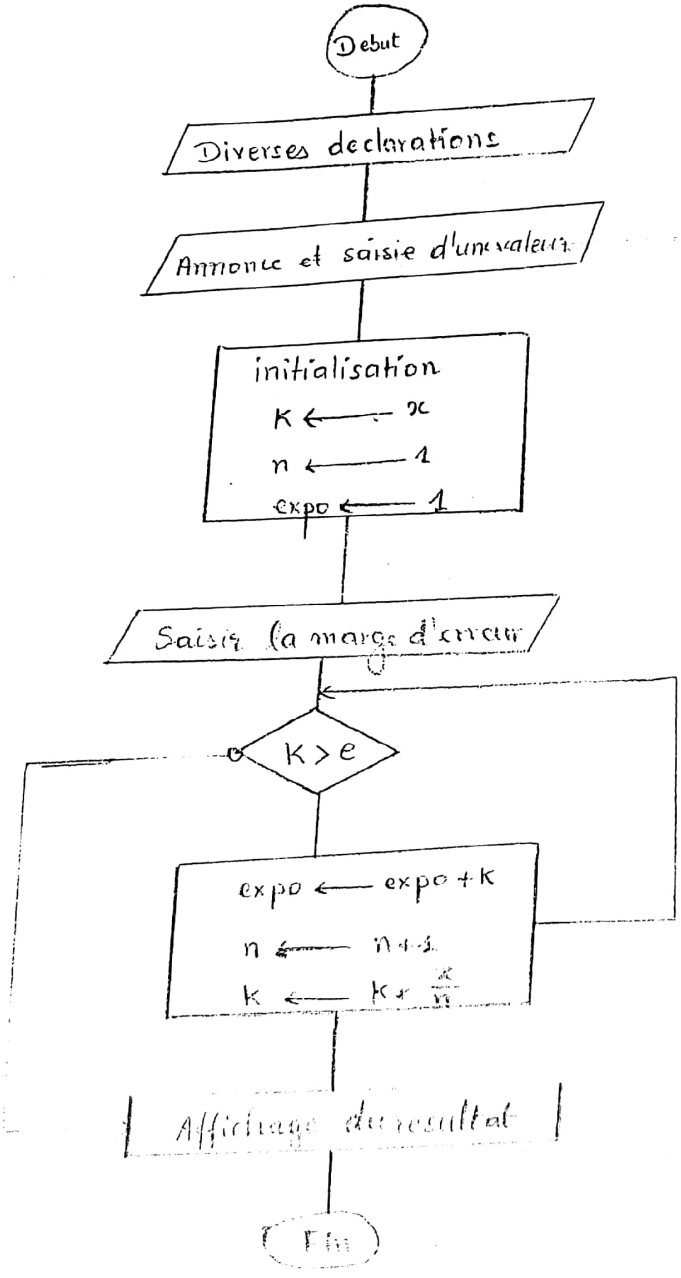
```
Program SaisieIG-2018;
    Var IG2018 : Tableau [1, 50] de chaine de caractère;
        i : entier;
Debut
    Afficherln ('saisie des membres IG2018');
    Pour i ← 1 JQA 50 faire
```

```
        Debut
            Afficher ('IG2018(', i, ') = ');
            Saisir (IG2018 [i]);
        Fin;
    Fpour;
Fin.
```

```
Program saisieig2018_V2;
  Const max = 100;
  Var ig2018 : Tableau [1.. max] de chaine de caractère;
      Numero, I, C : entier;
Debut
  Pour i ← 1 JQA max faire
      ig2018 [i] ← 'Absent';
  Fpour;
  Afficherln ('saisie classe ');
  Afficher ('Donnez le nombre d''élèves : ');
  Saisirln (T);
  C ← 0;
  Repeter
      Afficher ('Donner votre numero : ');
      Saisirln (numero);
      Si numero <>0
      Alors Debut
                Afficher ('Introduire votre nom : ');
                Saisirln (ig2018 [numero]);
                Si ig2018 [numero] <> 'Fin'
                Alors
                    C ← C + 1;
                Fsi;
            Fin;
  Fsi;
JQA (numero = 0) ou (C = T) ou (ig2018 [numero] = 'Fin');
```

```
    Si ig2018[numero] = 'Fin'
        Alors ig2018[numero] ← 'Absent';
    Fsi;
    Afficherln('Voici l'état de la classe:');
    Pour i ← 1 JUSQU'A T faire
        Afficherln(i, 'ig2018:', ig2018[i]);
    Fpour;
Fin.
```

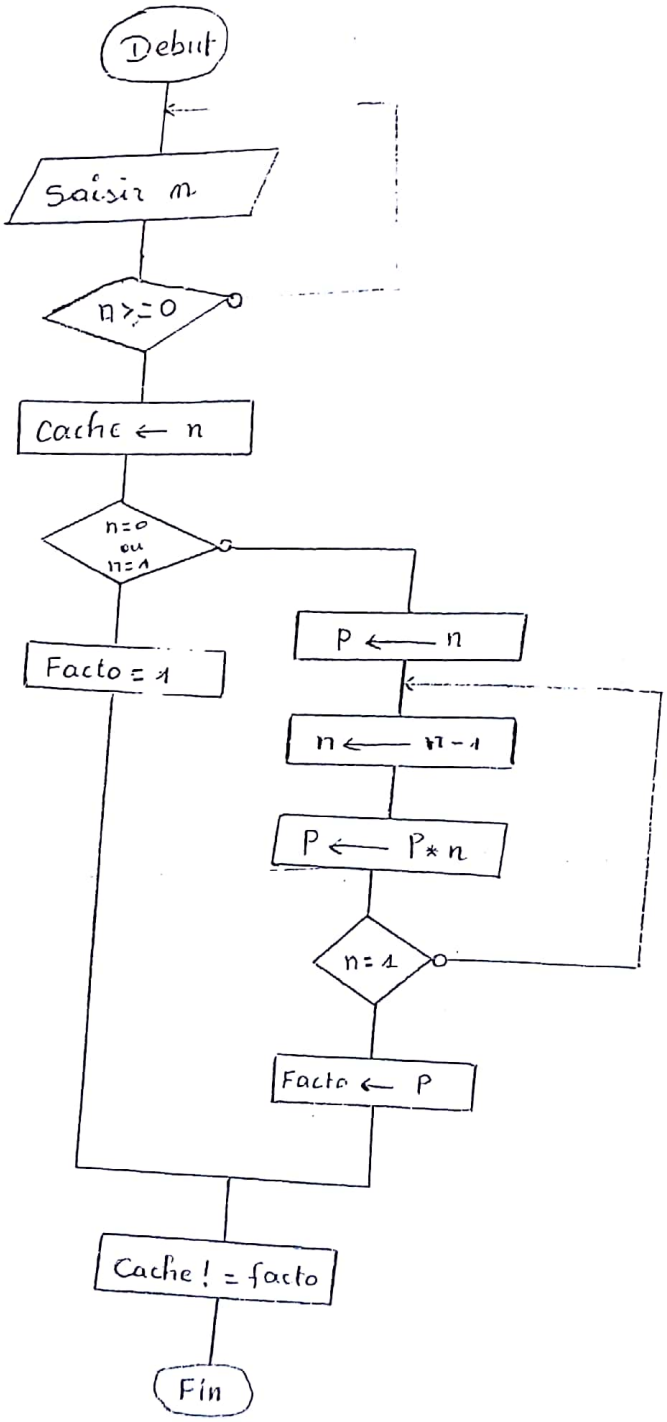
(9) Etablir un programme qui permet de calculer l'expo d'un nbre



```
Programme exponentielle ;
Var   n : entier;
      e, K, expo, x : reel;
      Reponse : string
Debut
  Repeter
    Afficherln ('Ce programme permet de calculer l''expo x.');
    Afficher ('Veuillez introduire x :');
    Saisirln (x);
    K ← x;
    n ← 1;
    expo ← 1;
    Afficher ('Donnez votre erreur admissible :');
    Saisirln(e);
    Tant que K>e faire
      Debut
        expo ← expo + K;
        n ← n+1;
        K ← K * x/n ;
      Fin;
    FTQ;
  Afficherln ('expo(', x, ')=', expo);
  Afficher('Vous êtes à la fin de ce programme, voulez-vous le');
  Afficher ('reprendre oui ou non ?');
  Saisir (reponse);
  JQA (reponse ='non');
Fin.
```

Réaliser un programme qui permet de calculer factorielle n.
n doit être positif.

*Logigramme :



X Program factorielle;

Var n, facto : entier;

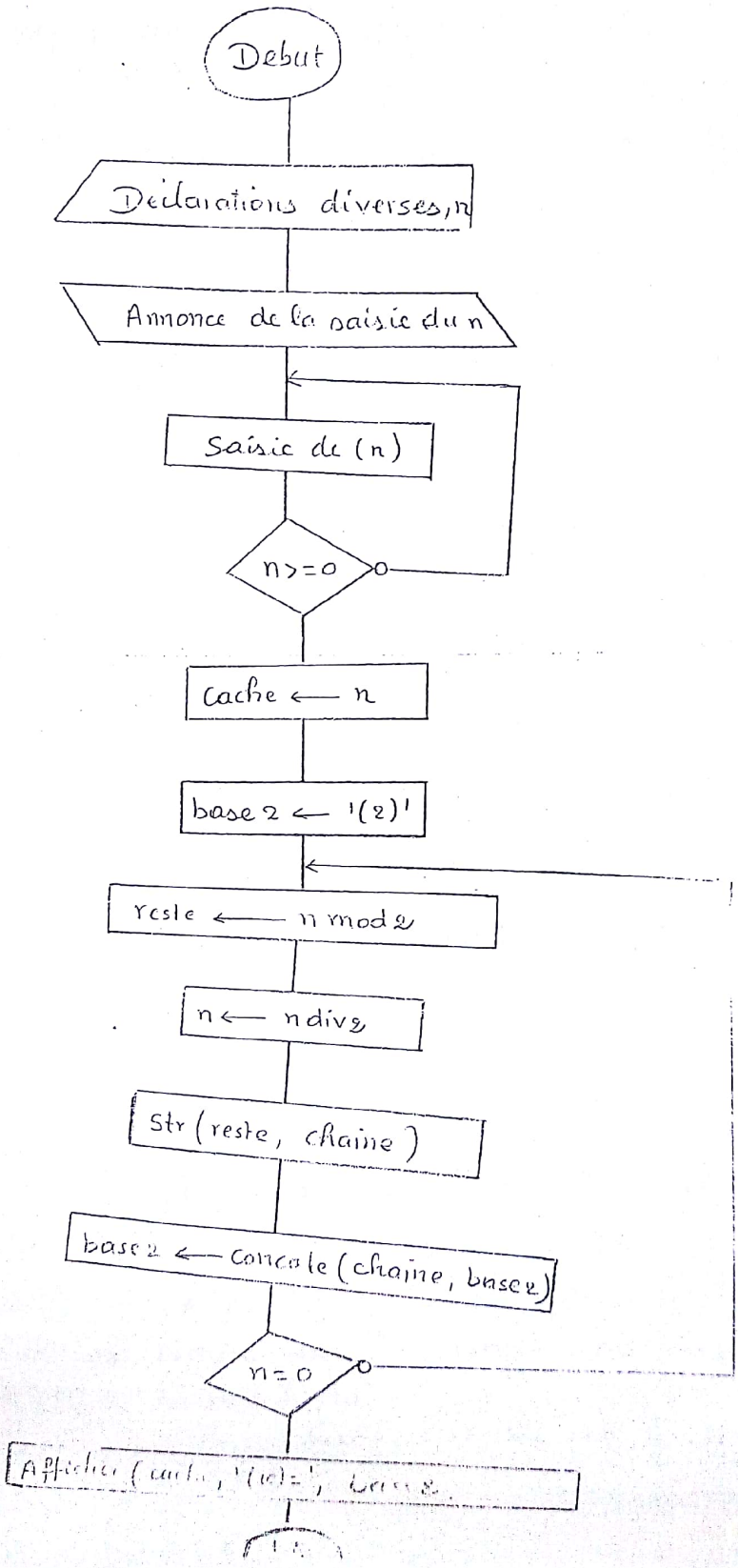
P, cache : entier;

Reponse : caractere;

```
Repeter
    AfficherLn('Ce programme calcule n!');
    Repeter
        Afficher('Donnez un entier positif :');
        SaisirLn(n);
        Si n<0 alors
            Debut
                Afficher('Vous avez saisi', n, 'qui est négatif !!');
            Fin;
        Fsi;
    JQA n>=0;
    Si (n=0) ou (n=1)
        Alors facto ← 1
    Sinon Debut
            cache ← n;
            Repeter
                P ← n;
                n ← n-1;
                P ← P* n;
            JQA n=1;
            Facto ← P;
          Fin;
    Fsi;
    Afficher(cache, '!=', Facto);
    Afficher('Vous êtes à la fin de ce programme, voulez-vous le');
    Afficher('reprendre oui ou non ?');
    Saisir(reponse);
    JQA (reponse='non');
Fin.
```

Ecrire un algo qui permet de convertir un nombre de base x en base 2 par division successive par 2, jusqu'à quotient = 0. Le nombre à saisir doit être obligatoirement positif.

* Logigramme :



X Programme Conversion_base10_en_base2;

```
Var  n, cache, reste : entier;
     base2 : caractère; (string)
     chaine: chaine de caractère; (string)
Debut
    Afficherln('Ce programme permet de convertir un nombre de base10 en base2');
    Repeter
        Afficherln('Veuillez introduire un nombre entier:');
        Saisirln(n);
        Si n<0
            alors afficherln('Attention!! Le nombre doit être positif!');
        Fsi;
    JQA n≥0;
    cache ⟵ n;
    base2 ⟵ '(2)';
    Repeter
        Reste ⟵ n mod(2);
        n ⟵ n div(2);
        str(Reste, chaine);
        base2 ⟵ Concat(chaine, base2);
    JQA n=0;
    Afficherln(cache, '(10)', '=', base2);
Fin.
```

the following programme doesn't belong here; do

(12) Elaborer un programme qui permet de saisir un vecteur et faire le tri de ce vecteur.

```
Algorithme trivecteur ;
const max = 50 ;
var i, j, l : entier ;
    T : Tableau [1..max] d'entier ;
    cas : entier ;
Debut
    Afficher ln ('Ce programme fait le tri croissant d'un vecteur !');
    Repeter
        Afficher ('Donner la taille de votre vecteur <=', max, ':');
        Saisir ln (l);
    JQA ((l >= 2) et (l <= max));
    Afficher ln ('Veuillez remplir le vecteur');
    Pour i ← 1 JQA l faire
        Debut
            Afficher ('T[', i, ']');
            Saisir ln (T[i]);
        Fin ;
    Fpour ;
        Effacer l'ecran ;
    Afficher ln ('Voici le vecteur saisi.');
    Pour i ← 1 JQA l faire
        Afficher (T[i], ' ');
    Fpour ;
    Afficher ln ;
    Pour i ← 1 JQA l-1 faire
        Pour j ← i+1 JQA l faire
            Si T[i] > T[j]
            alors Debut
```

```
                    cag ← T[i];
                    T[i] ← T[j];
                    T[j] ← cag;
                Fin;
            Fsi;
        Fpour j;
    Fpour i;
    Afficherln ('Voici le vecteur trié');
    Pour i ← 1 JSQA l faire
        Afficher (T[i], ' ');
    Fpour; Readln;
Fin.
```

(13) Elaborer un programme qui permet de faire la somme de deux matrices.

```
Program somme_2_matrices;
  const T=10;
  Var A, B, C : Tableau [1..T, 1..T] de entier;
      i, j, l, k : entier;

Debut
    Effacer l'ecran;
    Afficherln ('Ce programme permet de faire la somme de 2 matrices');
    Afficher ('Donnez l'ordre des matrices <', T, 'x', T, ':');
    Saisirln (l, K);
    Afficherln ('Veuillez remplir la prémière matrice.');
    Pour i ← 1 JSQA l faire
        Pour j ← 1 JSQA K faire
```

```
                Afficher ('A[', i, ',', j, '] = ');
                Saisirln (A[i,j]);
            Fin;
        Fpour j;
    Fpour i;

Afficherln ('Maintenant remplissez la 2e matrice.');
Pour i ← 1 JQA l faire
    Pour j ← 1 JQA K faire
        Debut
            Afficher ('B[', i, ',', j, '] = ');
            Saisirln (B[i,j]);
        Fin;
    Fpour j;
Fpour i;

Afficherln ('Maintenant nous allons faire la somme de A+B.');

Pour i ← 1 JQA l faire
    Pour j ← 1 JQA K faire
        C[i,j] ← A[i,j] + B[i,j];
    Fpour j;
Fpour i;

Pour i ← 1 JQA l faire
    Debut
        Afficherln;
        Pour j ← 1 JQA K faire
            Afficher (C[i,j], ' ');
        Fpour j;
    Fin;
Fpour i;
Fin.        saisirln;
```

(14) Elaborer un programme qui permet de faire le produit de deux matrices.

```
Program Produit_de_deux_matrices;
  Const max = 10;
  Var A,B,C : tableau [1..max, 1..max] de reel;
      i, j, K, l, m, n : entier;
      Reponse : caractère;
Debut
  Repeter
    Effacer ecran;
    Afficherln ('Ce programme fait le produit de deux matrices.');
    Afficher ('Donnez la taille du nombre de colonne de la');
    Afficherln ('matrice A et le nombre de lignes de B <=', max, ':');
    Saisirln (n);
    Afficher ('Donner le nombre de lignes de la matrice A <=', max, ':');
    Saisirln (l);
    Afficher ('Donner le nombre de colonne de B <=', max, ':');
    Saisirln (m);
    Afficherln ('Veuillez remplir la matrice A de taille', l, 'x', m);
    Pour i ← 1 à l faire
      Pour K ← 1 à n faire
        Debut
          Afficher ('A[', i, ';', K, ']=');
          Saisirln (A[i,K]);
        Fin;
    Fpour i Fpour K;
```

```
Afficher('Maintenant remplissez B de taille ', m, ' ; ', m);
    Pour K ← 1 JQA m faire
        Pour j ← 1 JQA m faire
            Debut
                Afficher('B[', K, ';', j, ']=');
                Saisir(B[K,j]);
            Fin;
        Fpour j;
    Fpour K;

Pour i ← 1 JQA l faire
    Pour j ← 1 JQA m faire
        Debut
            C[i,j] ← 0;
            Pour K ← 1 JQA n faire
                C[i,j] ← C[i,j] + A[i,K] * B[K,j];
            Fpour K;
        Fin;
    Fpour j;
Fpour i;

Afficher('Voici votre matrice produit C.');
    Pour i ← 1 JQA l faire
        Debut
            Afficher;
            Pour j ← 1 JQA m faire
                Afficher(C[i,j], ' ');
            Fpour j;
        Fin;
    Fpour i;
```

```
        Afficher ('Vous êtes à la fin de ce programme voulez vous le
        AfficherLn (' le repreter ou non ?:');
        SaisirLn(reponse);
    JQA(reponse = 'non');
Fin.
```

(15) Elaborer un programme qui permet de saisir un vecteur d'afficher ce vecteur, puis de rechercher le plus grand et le plus petit de ce vecteur.

```
Program Leplus_grond_leplus_petil ;
  Const max = 100;
  Var taille, i, petit, grond : entiers;
      V : Tableau [1.. max] de entier;
      Reponse : chaine de caractere ;
Debut
  Repeter
    Afficher ('Ce programme de saisir un vecteur puis d''afficher le plus')
    AfficherLn (' petit et le plus grond de ce vecteur.');
    Afficher ('Donner la taille de votre vecteur <=', max, ':');
    SaisirLn (taille);
    AfficherLn ('Veuillez à present remplir votre vecteur.');
      Pour i ← 1 JQA taille faire
        Debut
          Afficher ('V[', i, ']=');
          Saisir (V[i]);
        Fin;
      Fpour;
```

```
Petit ⟵ V[1];
Grand ⟵ V[1];
Pour i ⟵ 2 JQA taille faire
    Si Petit > V[i]
        alors Petit ⟵ V[i]
    Sinon Si Grand < V[i]
            alors Grand ⟵ V[i];
          Fsi;
    Fsi;
Fpour;
Pour i ⟵ 1 JQA taille faire
    AfficherLn(V[i], '␣');
Fpour;
Afficher('le plus petit est', petit, 'et le plus grand est', Grand);
Afficher('Vous êtes à la fin de ce programme voulez-vous');
AfficherLn('le reprendre oui ou non ? : ');
SaisirLn(reponse);
JQA (reponse = 'non') ou (reponse = 'NON');
Fin.
```

(16) Avec la structure menu, élaborez un programme.

```
Programme  Facto_base_tri_croissant ;
const  max = 20 ;
Var  n, cache, P, facto : entier ;
     Tampon, i, j, cam : entier ;
chaine, base3, Reponse : chaine de caractere ;
     V: Tableau [1..max] de entier ;
taille, choix, cachette, reste : entier ;

Debut
  Repeter
    Afficherln ('Bonjour et bienvenue sur cette page consacrée aux programmes.');
    Afficherln ('Ce programme permet de faire soit :');
    Afficherln ('1: Le calcul de la factorielle d''un nombre;');
    Afficheln ('2: La conversion de la base 10 en base 3;');
    Afficherln ('3: Le tricroissant d''un vecteur.');
    Afficher ('Faites votre choix entre 1, 2 et 3 :');
    Saisirln (choix);
    Suivant choix de
    1: Debut
         Afficherln ('Vous avez choisi de faire le calcul de la factorielle.');
         Repeter
           Afficher('Veuillez introduire votre entier positif :');
           Saisirln (n);
           Si n<0
             alors Afficher ('Attention, n doit être positif !!!');
           Fsi;
         JQA n>=0;
         Si n=0 ou n=1
           alors facto ⟵ 1
         Sinon Debut
                 cachette ⟵ n;
```

```
                P ← n;
                Debut
                    n ← n-1;
                    P ← P*n;
                Fin;
                JQA n=1;
            Facto ← P;
        Fin;
    Fsi;
    Afficherln (cachette, '!=', facto);
Fin;
2: Debut
    Afficherln('Maintenant vous avez choisi de faire la conversion.');
    Repeter
        Afficher('Veuiller introduire votre nombre entier positif:');
        Saisirln (n);
        si n<0
            alors afficher ('chiarlot n doit être positif!');
        Fsi;
    JQA n>=0;
    cache ← n;
    base3 ← '(3)';
    Repeter
        reste ← n mod(3);
        Tampon ← n div(3);
        Str (chaine, reste);
        base3 ← concat (chaine, base3);
    JQA n=0;
    Afficherln (cache '(10)=' base3);
Fin;
3: Debut
    Afficherln ('Enfin vous voulez faire le traitement d'un volet');
```

```
saisir(taille);
Afficher('Remplissez votre vecteur maintenant');
Pour i ← 1 JQA taille faire
    Debut
        Afficher('V[ i ]=');
        Saisir (V[i]);
    Fin;
Fpour;
Afficher('Voici votre vecteur');
Pour i ← 1 JQA taille faire
    Afficher (V[i], ' ');
Fpour;

Pour i ← 1 JQA taille - 1 faire
    Pour j ← i+1 JQA taille + 1 faire
        Si V[i] > V[j]
            alors Debut
                    con ← V[i];
                    V[i] ← V[j];
                    V[j] ← con;
                  Fin;
        Fsi;
    Fpour j;
Fpour i;
Afficher('Voici votre vecteur trié');
Pour i ← 1 JQA taille faire
    Afficher (V[i], ' ');
Fpour i;
```

```
        Readln;
        Fin
    Sinon Afficherln ('Votre choix est incorrect.');
    Fsuivant;
    Afficher ('Vous êtes à la fin de ce programme voulez-vous le ');
    Afficherln ('le reprendre ? oui ou non ');
    Saisir (Reponse);
JQA (reponse = 'non') ou (reponse = 'NON');
Fin.
```

(17)

```
Program matrice_sinus_vecteur;
  const max = 25;
  var l, c, i, j, D : entier;
  choix, PP, PG, taille : entier;
      x, sin, y, T : reel;
      V : Tableau [1.. max] de entier;
      A, B, C : Tableau [1.. max, 1.. max] de entier;
      Reponse : chaine de caractere;

Debut
  Repeter
    Afficherln (' Bonjour ');
    Afficherln ('Ce programme permet de faire soit :');
    Afficherln ('1: la somme de deux matrices');
```

```
Afficher (' 2: la saisie d'un vecteur et d'afficher ');
Afficherln (' 3: la saisie d'un vecteur et d'afficher le plus petit ');
Afficher ('Faites votre choix entre 1, 2 ou 3 : ');
Saisirln (choix);
Suivant choix de
1 : Debut
    Afficherln ('Vous avez choisi de faire la somme de deux matrice.');
    Repeter
        Afficher ('Introduisez la taille de vos matrices <=', max, 'x', max, ':')
        Saisirln (l, c);
        si (l < 2) ou (c < 2) et (c > max) ou (l > max)
            alors Afficher ('Données erronées.');
        Fsi;
    JQA (l >= 2) et (c >= 2) et (l <= max) et (c <= max);
    Afficher ('Veuillez remplir vos matrices.');
        Pour i ← 1 JQA l faire
            Pour j ← 1 JQA c faire
                Debut
                    Afficher ('A[', i, j, ']=');
                    Saisir (A[i,j]);
                    Afficher ('B[', i, j, ']=');
                    Saisir (B[i,j]);
                Fin;
            Fpour j;
        Fpour i;
    Pour i ← 1 JQA l faire
        Pour j ← 1 JQA c faire
            C[i,j] ← A[i,j] + B[i,j];
    Fpour i;
```