Djamel Eddine Z E G O U R

Apprendre et enseigner l'algorithmique

Tome 2 : Sujets d'examen corrigés.

Institut National d'Informatique

P r é f a c e

Introduction

Ce livre est le fruit d'une vingtaine d'années d'expérience dans le domaine de l'algorithmique et de la programmation. C'est le cours tel qu'il est assuré à l'Institut National d'Informatique d'Alger (Ex C.E.R.I) pour les étudiants de première année du cycle "ingénieur d'état en informatique".

Il constitue un support de cours pour des étudiants n'ayant aucune connaissance en programmation. Il est aussi destiné à des étudiants ayant déjà une première expérience en programmation et qui veulent connaître davantage sur l'art de la programmation.

Objectif du cours

Le cours couvre quatre grands aspects très liés : algorithmique, donnée, méthodologie et programmation. Dans une première étape, on s'intéresse essentiellement au formalisme algorithmique, ensemble de conventions permettant d'exprimer des solutions. On opère alors sur des objets simples et on développe des algorithmes sur la machine de Turing. Quant à la seconde étape, elle est totalement consacrée à l'étude des structures de données élémentaires à savoir les vecteurs, les listes linéaires et les fichiers . Un langage de programmation pédagogique est utilisé afin de s'initier à la programmation (PASCAL). Une méthode de recherche d'algorithmes, l'analyse descendante, est abordée avec tous les concepts qui s'y rattachent.

Contenu

La première partie renferme le cours d'algorithmique. Les concepts de base sont donnés en montrant la construction de programmes simples depuis leur expression en langage naturel avec ses inconvénients à leur expression entièrement dans un langage algorithmique. Une multitude d'algorithmes sont développés sur la machine de Turing afin de se familiariser avec le langage algorithmique. Une introduction aux fichiers et particulièrement aux structures de fichiers est donnée avec de nombreux programmes. On y trouvera essentiellement, les programmes de création, de maintenance et de tri de fichiers. Une méthode de conception d'algorithmes qu'est l'analyse descendante est exposée et illustrée en PASCAL en présentant tous les concepts qui s'y rattachent tels que la notion de portée, de communication entre modules, paramètres formels et réels, objet locaux et globaux, etc...

La seconde partie fournit un recueil de sujets d'examens. Pour chaque sujet, il est spécifié l'ensemble des cours à connaître.

La partie 3 fournit des corrigés types des sujets présentés dans la partie 2.

La partie 4 présente un ensemble d'exercices de programmation corrigés. Pour chaque programme, nous avons présenté les données et les résultats parfois détaillés dans le but de montrer leur conformité.

La partie 5 présente une série d'annexes très utiles pour un environnement de programmation. L'annexe 1 résume le langage algorithmique utilisé. Les annexes 2 et 3 donnent des compléments d'informations sur quelques notions élémentaires et sur les disques. L'annexe 4

résume les principales fonctions DOS utiles pour toute utilisation de micro-ordinateur. Les annexes 5 et 6 fournissent des moyens, d'une part, pour représenter schématiquement des algorithmes (organigrammes) et, d'autre part, pour les traduire dans des langages de bas niveau (langage d'assemblage par exemple). Dans l'annexe 7, on trouvera une façon de rédiger un dossier de programmation. Enfin, nous avons terminé par l'annexe 8 par un ensemble de conseils pratiques sous forme de proverbes utiles pour tout programmeur.

Remerciements

Nous exprimons nos remerciements les plus chaleureux à notre collègue W.K Hidouci pour ses conseils et surtout son efficacité dans la lecture approfondie de certaines parties de ce manuscrit.

Professeur Djamel Eddine ZEGOUR

Organisation du livre

Tome1

Partie 1
. Cours d'algorithmique

Partie 2 : Annexes
1. Langage algorithmique
2. Notions élémentaires
3. Les Disques
4. Système d'exploitation MS-DOS : aide mémoire
5. Organigrammes
6. Traduction des algorithmes vers les langages de bas niveau
7. Rapport de programmation
8. Proverbes de programmation

Tome2

Partie 1 *(Page 7)*
. Enoncés de sujets d'examens

Partie 2 *(Page 35)*
. Corrigés de sujets d'examens

Partie 3 *(Page 111)*
. Exercices programmés en PASCAL

Partie 4 : Annexes *(Page 143)*
1. Langage algorithmique
2. Rappel PASCAL
3. Proverbes de programmation

S O M M A I R E

I. Enoncés de sujets d'examens

C1. Concepts de base
C2. Concepts de base - Programmation PASCAL - Machine de Turing
C3. Concepts de base - Machine de Turing
C4. Concepts de base - Machine de Turing
C5. Machine de Turing - Programmation PASCAL
C6. Machine de Turing - Vecteurs - Programmation PASCAL
C7. Concepts de base
C8. Machine de Turing - Programmation modulaire - Programmation PASCAL
C9. Vecteurs - Fichiers - Programmation PASCAL
C10. Programmation modulaire - Vecteurs - Programmation PASCAL
C11. Machine de Turing - Vecteurs - Programmation PASCAL
C12. Machine de Turing - Vecteurs - Programmation PASCAL
C13. Listes linéaires chaînées - Programmation PASCAL
C14. Machine de Turing - Programmation PASCAL
C15. Vecteurs - Programmation PASCAL
C16. Listes linéaires chaînées - Vecteurs - Fichiers
C17. Machine de Turing - Programmation PASCAL
C18. Vecteurs - Machine de Turing
C19. Concepts de base - Machine de Turing
C20. Machine de Turing - Vecteurs
C21. Vecteurs
C22. Concepts de base
C23. Machine de Turing - Programmation modulaire - Programmation PASCAL
C24. Programmation modulaire - Vecteurs - Programmation PASCAL
C25. Vecteurs
C26. Machine de Turing - Vecteurs - Programmation PASCAL
C27. Machine de Turing - Programmation PASCAL
C28. Concepts de base - Vecteurs - Programmation PASCAL
C29. Machine de Turing

II. Corrigés de sujets d'examens

Corrigé 1.....
Corrigé 2.....
Corrigé 3.....
Corrigé 4.....
Corrigé 5.....
Corrigé 6.....
Corrigé 7.....
Corrigé 8.....
Corrigé 9.....
Corrigé 10.....
Corrigé 11.....
Corrigé 12.....
Corrigé 13.....
Corrigé 14.....
Corrigé 15.....

Corrigé 16.....
Corrigé 17.....
Corrigé 18.....
Corrigé 19.....
Corrigé 20.....
Corrigé 21.....
Corrigé 22.....
Corrigé 23.....
Corrigé 24.....
Corrigé 25.....
Corrigé 26.....
Corrigé 27.....
Corrigé 28.....
Corrigé 29.....

III. Exercices programmés en PASCAL

PROGRAMME 1. Réalisation d'un dessin
PROGRAMME 2. Etablissement d'une facture
PROGRAMME 3. Course de Ski (I)
PROGRAMME 4. Les mots de la forme ....XY.....
PROGRAMME 5. Les mots de la forme ....X....Y....
PROGRAMME 6. Analyse des constantes "virgule flottante"
PROGRAMME 7. Recherche dichotomique dans un vecteur
PROGRAMME 8. Course de ski (II)
PROGRAMME 9. Parties d'un ensemble
PROGRAMME 10. Inventaire

IV. Annexes

Annexe 1. Langage algorithmique
Annexe 2. Rappel PASCAL
Annexe 3. Proverbes de programmation

Partie 1

Enoncés de sujets d’examens

Exercice 1 : 2N

Ecrire deux algorithmes différents permettant de calculer 2^N pour tout N >=0 donné
a) d'abord en remarquant que 2N = 2 * 2 * . . * 2 (N fois)
b) puis en remarquant que 2N = 2N-1 + 2N-1

Exercice 2 : Extremum

Soit une liste de K nombres (K donné). Ecrire l'algorithme qui permet de rechercher le maximum et le minimum. On suppose que chaque nombre est obtenu par un ordre de lecture.

Exercice 3 : F ?

Que fait l'algorithme suivant ? Justifier votre réponse.

```
ALGORITHME F
VAR
 Res, N, I : ENTIER
DEBUT
 Lire(N)
 I := N
 Res := 1
 TANTQUE I > 1 :
  Res := Res * I
  I :=  I - 1
 FINTANTQUE
 Ecrire( N, res)
FIN
```

Exercice 4 : PGCD

On suppose que MOD(N, A) est une fonction qui détermine le reste de la division de l'entier N par A. ex : Mod(12, 7) = 5
Si Reste est le nom d'un objet de type variable entière l'action " Reste := Mod(N,A)" permet d'affecter à la variable Reste le reste de la division de N par A.
Ceci étant, écrire l'algorithme permettant de déterminer le PGCD de deux nombres A et B donnés.

Exercice 5 : Calcul

Soit l'algorithme suivant :

```
ALGORITHME Calcul
VAR
 E1, E2, E3 : ENTIER
 R : REEL
DEBUT
 E1 := 5 ; E2 := 8
 E3 := E1 * E2
```

```
 E3 :=(E1 * E2) / 3
 R := (E1 * E2) / 3
 E3 := ( E1 + E2) / 2
 R := ( E1 + E2 ) / 2
FIN
```

Quelles sont les différentes valeurs que prennent E3 et R? Faire une trace.

* * * * *

Turing

1. Aspect algorithmique

A - Soit une liste de N entiers. Ecrire les algorithmes suivants :
a) - Calcul de la somme des nombres positifs et celle des nombres négatifs.
b) - Nombre des sous-suites décroissantes.

Exemple : si la liste est 7, 8, -1, 3, 5, -17, 15, -28
a) - somme des nombres positifs = 38
somme des nombres négatifs = -46
b) Les sous-suites décroissantes sont : {7}, {8, -1}, {3}, {5, -17}, {15, -28}

B- On dit que deux nombres x et y sont amis si et seulement si x2 + y2 est un carré parfait.
Ecrire l'algorithme qui recherche tous les nombres entiers amis compris entre 1 et 1000.
On n'utilisera pas l'opération "racine carrée".
Exemple : 3 et 4 sont amis car 32 + 42 = 52.

2. Aspect programmation

a) Traduire B en PASCAL.
b) Ecrire le programme qui réalise le dessin suivant :

```
                    *
      * *
      * 1 *
      * 2 2 *
      * 3 3 3 *
      . .

      * N N N . .     *
      * * * * . .     * * *
```

pour un entier N donné. (1 <= N <= 20)

* * * * *

Exercice 1 : Calcul de la somme 1 + 5 + 9 + . .

Ecrire l'algorithme qui réalise la somme

Sn = 1 + 5 + 9 + . . + (4N + 1)
pour un entier N donné.

Exercice 2 : Calcul de 2N avec l'addition

On dispose d'une machine qui ne sait qu'additionner. Ecrire l'algorithme qui réalise 2^N (N >=0).

Exercice 3 : Comptage du nombre de mots

Soit une suite de caractères terminée par un point('.'). On définit un mot comme étant une suite de caractères ne contenant pas de blanc. La longueur d'un mot est par conséquent le nombre de caractères qu'il contient. Ecrire un algorithme permettant de compter le nombre de mots de longueur L donnée.
NB. On suppose que chaque ordre de lecture délivre le prochain caractère.

* * * * *

Exercice 1 : Calcul de la somme 13 + 33 + 53 .....

Ecrire l'algorithme qui réalise la somme :
$Sn = 1^3 + 3^3 + 5^3 + . . + (2N+1)^3$
pour un entier N donné.

Exercice 2 : Division de deux entiers A et B avec l'opérateur '-'

On dispose d'une machine qui ne sait que soustraire. Ecrire l'algorithme qui fait la division de deux entiers A et B (A >= B >= 0). On imprimera le reste et le quotient.

Exercice 3 : Longueur du mot le plus long

Soit une suite de caractères terminée par un point('.'). On définit un mot comme étant une suite de caractères ne contenant pas de blanc. La longueur d'un mot est par conséquent le nombre de caractères qu'il contient. Ecrire un algorithme qui détermine la longueur du mot le plus long.
NB. On suppose que chaque ordre de lecture délivre le prochain caractère.

* * * * *

C5. Machine de Turing - Programmation PASCAL

1. Aspect algorithmique

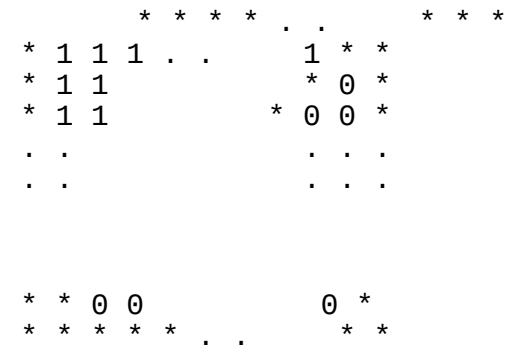
Soit une liste de N entiers naturels (N > 2).
Ecrire les algorithmes suivants :

a) Recherche du minimum et du maximum.
b) Recherche du premier nombre dont le carré est égal à la somme des deux précédents.
c) Recherche du nombre de sous-suites croissantes.

Exemple : si la liste est la suivante : { 7, 8, 1, 3, 5, 17, 15, 28, 3}
a) maximum = 28
minimum = 2
b) c'est le nombre 3 car 32 = 1 + 8
c) les sous-suites sont {7, 8}, {1, 3, 5, 17}, {15, 28}, {3}. Le nombre de sous-suites est alors 4.

2. Aspect programmation

a) Traduire le dernier algorithme en PASCAL.
b) Ecrire le programme PASCAL qui réalise le dessin suivant :



N X N caractères.

* * * * *

Exercice 1 : Moyenne

Soit un vecteur contenant N mesures. Calculer leur moyenne et le nombre de mesures qui diffèrent de la moyenne de plus d'une quantité donnée.
1) Donner l'algorithme
2) Ecrire le programme PASCAL correspondant.

Exercice 2 : Reconnaissance de données arithmétiques

Dans le langage PL1, une donnée arithmétique DECIMAL FIXED (W, D) peut avoir l'une des deux formes suivantes :

chchch.....ch
ou
chchch.....ch.chch.....ch

W est le nombre total de caractères, D est le nombre de chiffres après le point décimal.

Ecrire l'algorithme puis le programme PASCAL permettant de reconnaître N données arithmétiques DECIMAL FIXED (W, D) lues. Les données utilisées sont séparées par des blancs, des virgules ou une combinaison des deux.

Ce programme doit reconnaître ces données, les contrôler et les ranger (sans conversion) dans un tableau en spécifiant leur précision.

Exemple :
Si on a : 123,,,143.5 ,12A3 1538.38 , , 342.56 AABCD 1,2,34 #
on aura :
* * Constante erronée : 12A3
* * Constante erronée : AABCD
Premier élément :123 W=3 D=0
Deuxième élément :143.5 W=5 D=1
Troisième élément :1538.38 W=7 D=2

Remarques :

- Le programme doit signaler les erreurs éventuelles.
- l <= 15 et N <= 10

Exercice 3 : Tri par interclassement

On considère le vecteur initial à N éléments comme N sous-vecteurs triés à un élément. Un premier interclassement des sous-vecteurs deux à deux conduit à des sous-vecteurs triés de deux éléments. On recommence alors jusqu'à obtenir un seul vecteur trié. Si à une étape, on a un nombre impair de sous vecteurs, le dernier est reporté sans changement à l'étape suivante.

Exemple :
12 4 8 11 14 10 11 5 1 13
(4 12) (8 11) (10 14) (5 11) (1 13)
(4 8 11 12) (5 10 11 14) (1 13)
(4 5 8 10 11 11 12 14) (1 13)
(1 4 5 8 10 11 11 12 13 14)

Donner l'algorithme réalisant cette méthode de tri. On utilisera le module d'interclassement sans le développer.

* * * * *

Exercice 1 : Carré parfait

Ecrire un algorithme qui donne comme résultat la valeur :
- VRAI si N est un carré parfait
- FAUX sinon

N étant une donnée.
nombre A est dit carré parfait s'il existe un entier B tel que B2=A.
On n'utilisera pas l'opérateur de puissance.

Exercice 2 : Couples (A, B) tels que A = 3B

Imprimer tous les couples (A, B) d'entiers naturels compris entre 1 et 100 tels que A= 3 B.

Exercice 3 : Calcul de somme

Calculer la somme

S = 1 + 5 + 9 + + 4N+1 pour N >=0

Exercice 4 : Fibo ?

Soit l'algorithme suivant :

```
        ALGORITHME FIBO
VAR  Fb, Fb0, Fb1, K, N : ENTIER
 Trouv : BOOLEEN
DEBUT
 Lire(N)
 Fb0 := 0
 Fb1 := 1
 Trouv := FAUX
 K := 2
 TANTQUE NON Trouv :
  Fb := Fb0 + Fb1
  SI Fb > N :
   K := K + 1
   Trouv := VRAI
  SINON
   K := K + 1
   Fb0 := Fb1
   Fb1 := Fb
   FSI
 FINTANTQUE
 Ecrire(K, Fb)
FIN
```

Questions :

1 Faire une trace complète pour N = 10.
2 Que fait-il ?
3 Peut on l'écrire de façon plus simplifié ?

* * * * *

Programmation PASCAL

Partie 1 : Machine-caractères

Considérons la machine-caractères abstraite définie en cours.

a) Ecrire l'algorithme qui recherche un mot donné dans le ruban de cette machine.

b) Ecrire l'algorithme qui imprime tous les mots commençant par 'T' et se terminant par 'S'.

Partie 2 : Les actions composées

a1) Ecrire l'algorithme qui calcule sans l'utilisation de l'opérateur '**' l'expression suivante :

((AB + CD) / AD) A

pour A,B, C et D donnés et non nuls.

a2) Le traduire en PASCAL.

b1) Ecrire (ou définir en langage algorithmique) les fonctions suivantes :

F(X) = X5 + X2 + 18
G(X) = X7 + X3 + X + 1

et le prédicat A = B

b2) Utiliser ces actions composées pour écrire l'algorithme qui donne les solutions de l'équation F(X) = G(X) pour X dans Z (ensemble des entiers relatifs) et X dans [-1000, +1000].

b3) Traduire cet algorithme en PASCAL.

* * * * *

Exercice 1 : Inventaire

On dispose d'un fichier TEXT PASCAL décrivant la liste des produits en stock dans les différents magasins d'une chaîne de supermarché.

Chaque ligne de ce fichier a le format suivant :

N°produit Prix unitaire Nombre d'unités en stock
<-- 4 -->BB <---- 7 ---->BB <-------- 4---------->

La dernière ne contient que des zéros. B désigne un blanc.

1) Créer à partir de ce fichier un autre fichier STOCK (de type FILE) où chaque enregistrement est une structure à 3 champs : numéro du produit, prix unitaire et nombre d'unités en stock.

2) Pour l'inventaire de fin d'année et à partir du fichier STOCK défini en 1), éditer un état ayant la forme suivante :

MAGASIN 1
RAYON 1 : MONTANT DU STOCK ???
RAYON 2 : MONTANT DU STOCK ???
.....
MONTANT TOTAL DU STOCK ???

MAGASIN 2
RAYON 1 : MONTANT DU STOCK ???
.....

MONTANT DU STOCK POUR L'ENSEMBLE DES MAGASINS ???

où les '?' sont remplacés par les valeurs calculées par le programme.

- Les numéros de magasins et de rayons dans chaque magasin sont supposés tous présents à partir de 1. Le fichier est trié par numéro de magasin croissant et pour un magasin donné par numéro de rayon croissant.

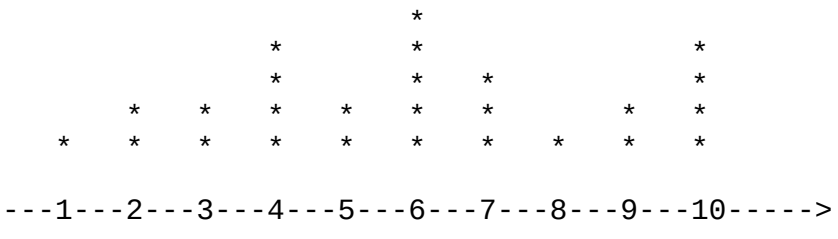
- Pour un produit donné, le numéro de magasin est constitué par les deux premiers chiffres du numéro de produit, et le numéro du rayon par les deux chiffres suivants. Exemple : O4174043 produit vendu au magasin n° 4, rayon n° 17.

Pour les questions 1) et 2) fournir les algorithmes avant de donner les programmes PASCAL correspondants.

Exercice 2 : Histogramme

Une assemblée vote en choisissant une possibilité parmi 10. Donner l'algorithme qui imprime l'histogramme du vote.

Exemple d'histogramme :



Exercice 3 : Tri par insertion

Soit un vecteur T(1..N) dans lequel on a défini une relation d'ordre. On désire trier ce vecteur selon la méthode suivante :

1) On suppose T(1..K) trié avec K<N, puis on insère l'élément T(K+1) dans le sous-vecteur T(1..K+1) à sa bonne position.

2) Faire varier K de 1 à N-1 dans 1)

Ecrire l'algorithme correspondant.

* * * * *

PASCAL

Exercice 1 : Variétés

Considérons une liste de N nombres entiers (N>1). On veut écrire un algorithme qui répond aux questions suivantes :

- déterminer le troisième nombre premier s'il existe,
- déterminer le deuxième carré parfait s'il existe,
- déterminer le nombre des diviseurs du premier nombre pair et du dernier nombre impair.

Pour le faire on définit les modules suivants :

a) PREM(Nbr, Bool) :
En entrée : un nombre quelconque Nbr.
En sortie : Bool <-- VRAI si Nbr est premier,
Bool <-- FAUX si Nbr est non premier.
Rôle : reconnaît si un nombre est premier ou non.

b) CARRE(Nbr, Bool) :
En entrée : un nombre quelconque Nbr.
En sortie : Bool <-- VRAI si Nbr est un carré parfait,
Bool <-- FAUX sinon.
Rôle : reconnaît si un nombre est carré parfait ou non.

c) PAIR (Nbr) :
En entrée : un nombre quelconque Nbr.
en sortie : VRAI si Nbr est un carré parfait, FAUXsinon.
Rôle : reconnaît si un nombre est pair ou non.

NBDIV(Nbr, Ndiviseurs) :
En entrée : un nombre quelconque Nbr.
En sortie : le nombre des diviseurs de Nbr est rangé dans la variable Ndiviseurs.
Rôle : détermine le nombre de diviseurs d'un nombre donné.

QUESTIONS :

a) Ecrire séparément les 4 actions ainsi définies.
b) Utiliser ces actions pour écrire UN SEUL algorithme qui répond aux questions posées.

Remarques :

a) Ne pas utiliser 'opérateur "racine carrée".
b) Chaque nombre de la liste est délivré par un ordre de lecture.
c) Les questions a) et b) sont indépendantes.

Exercice 2 : Norme euclidienne

Etant donnés P vecteurs (X1, X2, . . , XN) de dimension N. (P et N donnés).Quel est le rang du vecteur qui possède la plus grande norme euclidienne ?

Vous devez répondre à cette question :
a) en énonçant d'abord le principe que vous utilisez en langage naturel,
b) puis, en écrivant un algorithme.

Rappel : la norme euclidienne d'un vecteur est donnée par la formule :
Racine carré (somme (X12+X22+ ..... XN2))

Exercice 3 : Id ?

Soit le programme PASCAL suivant :

```
        PROGRAM Id;
VAR
 N, X, I, P, K, W : INTEGER;
 S : REAL;

PROCEDURE Y ( VAR A, B : INTEGER);
VAR L : INTEGER;
BEGIN
 B := 1;
 FOR L:=2 TO A DO B  := B * L
END;
BEGIN
 READ(X); READ(N);
 S := 1;
 FOR I:=1 TO N DO
  BEGIN
   Y(I, P);
   W:= 1;
   FOR K:=1 TO I DO W := W * X;
   S := S + ( W/ P)
  END
END.
```

Questions :

a) Dérouler le programme pour X=3 et N = 4

b) Donner les contenus des variables P, S et W.
c) Que fait le programme ?

* * * * *

Exercice 1 : Le plus apparent

On considère un vecteur contenant N entiers. Ecrire l'algorithme qui recherche l'élément ayant le maximum d'occurrences dans le vecteur, c'est à dire qui apparaît le plus de fois.

Exercice 2 : Course de ski

On désire contrôler par ordinateur l'affichage d'une course de ski. Ecrire l'algorithme et le programme PASCAL qui après chaque arrivée d'un concurrent affiche le classement partiel sous la forme :

```
 <---10---> <-------20---------> <-4> <----14------>
**************************************************
*CLASSEMENT*  NOM DU CONCURRENT *PAYS*    TEMPS     *
*          *                   *    *              *
*          *                   *    * MN  SEC CT   *
**************************************************
*          *                   *    *              *
*    1     *     xxxxx         * yy *  5   10   86 *
*          *                   *    *              *
**************************************************
*          *                   *    *              *
```

Remarques :

1) A chaque arrivée, il faudra insérer le concurrent à la place correspondante au classement fait sur le temps de parcours.

2) Description des données :
Pays, Nom prénom, Temps(en minutes, secondes et centièmes de seconde). Les données sont disposées à raison d'une ligne par concurrent comme suit :

PaysBB Nom et Prénom BBMnScCs
< 4> < 20 > < 6 >

3) L'affichage se fera sur imprimante suivant le format ci-dessus. Chaque tableau sera imprimé au milieu d'une page du listing.

4 Le jeu d'essai comportera 10 concurrents, on n'oubliera pas de traiter les ex-aequos.

Exercice 3 : Edition d'une ligne

Soit S une suite de caractères terminée par ".". Il s'agit de recopier sur listing une suite de caractères en fonction de la suite en entrée. S comprend des caractères représentant une succession de lignes de texte et des caractères-clés exprimant des actions à entreprendre par l'éditeur (votre programme) ; ces derniers ne doivent pas être recopiés sur listing. Il y a 4 caractères-clés qui sont récapitulés dans le tableau ci-dessous :

Caractère	Fonction
/	Annulation du dernier caractère de la ligne en cours de formation (s'il existe)
<	Annulation de toute la ligne
>	Caractère de tabulation sur la ligne en cours; il indique que le prochain caractère doit apparaître dans une colonne multiple de 10.
#	Fin de la ligne en cours qui peut alors être imprimée.

Les lignes à imprimer ne devront pas dépasser 120 caractères. En cas de tentative de création d'une ligne plus longue, on imprimera les 120 premiers caractères, puis un message d'erreur " ligne tronquée".

Exemple :

Soit la suite de caractères en entrée :

X<ABC//234567>ZZZ#1234#ABCD<1234>A//8#<1///AB#>9#1/FIN#.

On aura alors sur listing :

1	2	3	4	5	6	7	8	9	10	11	12
A	2	3	4	5	6	7			Z	Z	Z
1	2	3	4								
1	2	3	4					8			
A	B										
									9		
F	I	N									

Travail à faire :

Faire une analyse du problème en le décomposant en modules, puis écrire l'algorithme répondant au problème posé.
NB. La lecture de S se fait caractère par caractère.

* * * * *

Exercice 1 : vérification syntaxique des déclarations FORTRAN

Soit un ensemble de N lignes où chacune renferme une déclaration FORTRAN. Ecrire un algorithme commenté qui imprime pour chaque déclaration la valeur VRAI si elle est correcte, un message d'erreur si elle est fausse.

Une déclaration FORTRAN a la forme suivante :
Colonne 1 à 6 : des blancs
Colone 7 à 72 : texte de la déclaration

Le texte de la déclaration a la forme suivante :

TYPE Idf1, Idf2, ,Idfn

avec type dans { INTEGER, REAL, LOGICAL}

Idf1, Idf2, ...et Idf3 sont des identificateurs.

a)Entre TYPE et l'identificateur il y a au moins un blanc. Les identificateurs sont séparés par des virgules. Des blancs peuvent exister entre les identificateurs et les virgules.

b) Idfi : suite de caractères alphanumériques dont le premier est alphabétique. La longueur est au maximum égale à 8.

On suppose qu'il existe une relation d'ordre dans l'ensemble des caractères. On a ainsi par définition :

autres caractères <'1'<'2'<... <'9'<'A'<'B'.. < 'Z'

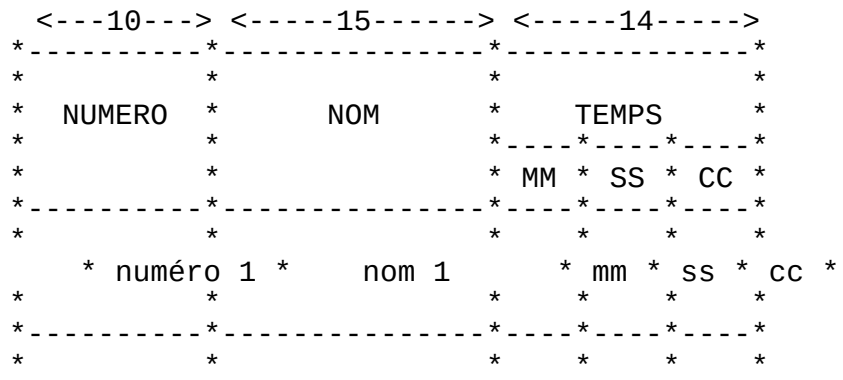
Chaque ligne renfermant une déclaration est lue caractère par caractère. On utilisera la fonction Sauter-ligne qui permet le positionnement en colonne 1 de la ligne suivante.

Exercice 2 : Course de ski

Pendant une course de ski, on a relevé après un essai les temps mis par N concurrents. On désire, après avoir mis les informations sur fichier, obtenir d'abord le tableau d'affichage des temps mis par les N concurrents, puis le tableau d'affichage concernant uniquement les trois premiers.
Les données sur fichier ont la forme suivante :

Première ligne : nombre de concurrents sur 3 positions.
Dans les N lignes suivantes, on a recensé les informations concernant les concurrents. Chaque ligne est organisée comme suit :
Colonne 1 à 4 : Numéro du concurrent
Colonne 12 à 19 : Nom du concurrent
Colonne 30 à 31 : Minutes
Colonne 32 à 33 : Secondes
Colonne 34 à 35 : Centièmes de seconde

Le tableau d'affichage aura la forme suivante :



Etc.

Ecrire l'algorithme et le programme PASCAL correspondant.

* * * * *

Exercice 1 : Parties d'un ensemble donné.

Ecrire l'algorithme qui engendre toutes les parties d'un ensemble V donné. On assimilera V à un vecteur V(1..N)

Exemple :

Donnée : V = { 2, 4, 8 }

Résultat : {}, {2}, {4}, {8}, {2, 8}, {2, 4}, {4, 8}, {2, 4, 8}

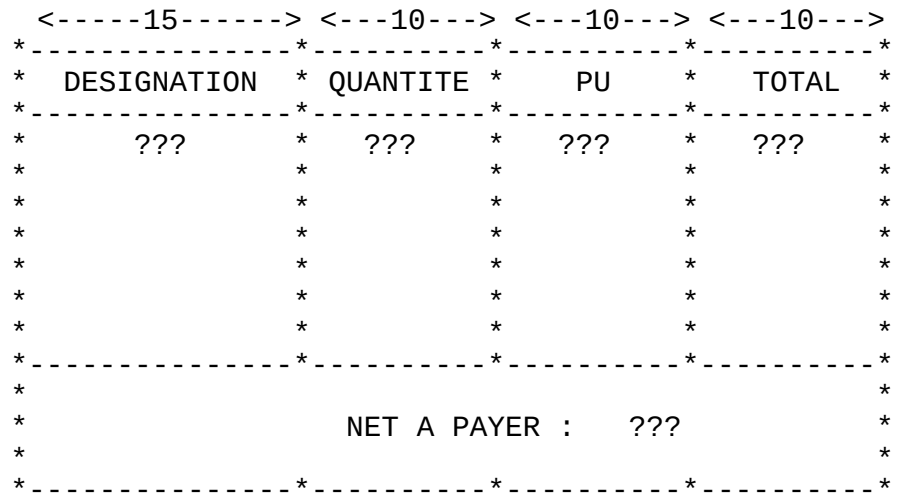
Coder l'algorithme en PASCAL.

* * * * *

Exercice 1: Facture

En vue d'établir une facture, on a relevé sur chaque ligne d'un fichier TEXT PASCAL les informations que l'on organise comme suit :
Ccolonne 1 à 5 : numéro du produit ou désignation
Colonne 10 à 14 : quantité (entier)
Colonne 20 à 28 : prix unitaire
La dernière ligne ne contient que des zéros.

Ecrire l'algorithme puis le programme PASCAL qui édite la facture selon le format suivant :



Exercice 2 : Analyse d'un texte

Soit un texte se terminant par le mot 'FINTEX'. Le texte contient des mots et des commentaires. Un commentaire est une suite de caractères comprise entre /* et */.
Un mot est une suite de caractères ne contenant pas de blancs et ne commençant pas /*. Un mot est dit numérique s'il ne contient que des caractères numériques. Un mot est dit alphabétique s'il ne contient que des caractères alphabétiques.
Ecrire un algorithme commenté qui rend la valeur 0 s'il s'agit d'un commentaire, la valeur 1 si le mot est alphabétique, la valeur 2 s'il est numérique et la valeur 3 dans les autres cas. De plus, dans le cas où le mot est numérique, l'algorithme doit fournir la valeur entière de ce mot et ce en utilisant la fonction PRENDRE(Mot, I) définie ci-dessous.

NB.
- La fonction PRENDRE(Mot, I) donne le caractère de rang I dans le mot Mot.
- On suppose que l'ensemble des caractères est muni d'une relation d'ordre "<" définie comme suit : autres caractères < '0'<'1'<. <'9'<'A'<'B'.. <'Z'
- La lecture se fait caractère par caractère.

Exemple

Texte : DEBUT---A-=-B5--/*AX--*/---V5+---;-A/*X---FIN--385---4B+----/*AB+-/*-X34--*/--25--FINTEX
Résultat
1, 1, 3, 3, 0, 3, 3, 3, 1, 2(385), 3, 0, 2(25)

Exercice 3 : Tri de 4 variables

Ecrire un algorithme qui liste en ordre croissant les contenus de 4 données A, B, C et D.

* * * * *

Exercice 1 : gestion des polynômes

Supposons que nous voulions manipuler des polynômes de la forme :
P(x)= c1xe1 + c2xe2+ .. + cnxen (1)

avec a) e1 > e2 .. > en
b) n<=20
c1, c2, .. cn sont des réels
e1, e2, .. en sont des entiers compris entre 0 et 19.

Un tel polynôme peut être représenté par une structure dont le premier élément indique le nombre de termes non nuls et dont le second élément est un tableau à deux dimensions où chaque ligne représente le couple (ci, ei).

Ecrire les modules suivants :
Point(P, V) : fonction qui rend la valeur P(V)
Somme(P, Q, R) : somme des polynômes P et Q; résultat dans R.
Produit(P, Q, R) : produit des polynômes P et Q; résultat dans R.
Dérivé(P, V) : dérivé du polynôme P; résultat dans R.

Ecrire une procédure PASCAL qui imprime le polynôme P sous la forme (1).

NB. On définira le type polynôme comme suit :

```
TYPE Polynome = STRUCTURE
 Nbr : ENTIER { Nombre d'éléments non nuls }
 Tab : TABLEAU[40, 2] DE REEL { Les monômes }
FIN
```

Exercice 2 : Accès par fonction

On a défini sur les vecteurs deux types d'accès :
- l'accès séquentiel
- l'accès dichotomique

Si l'on veut faire de l'accès séquentiel, on construit le vecteur de telle sorte que l'ordre des éléments est quelconque. Par contre, si on veut faire de l'accès dichotomique l'ordre des éléments est imposé.
On peut envisager un autre type d'accès que l'on définit comme suit :
Soit à construire un vecteur V à partir d'une liste de N valeurs V1,V2,.. Vn. On définit une fonction F qui à toute valeur Vi attribue un entier compris entre 1 et N. L'élément Vi est alors rangé à la place F(Vi). Comme on ne peut dans la pratique trouver une fonction F qui attribue une valeur différente à chaque élément de V, il se peut que l'on ait :
F(Vi) = F(Vj) = F(Vk)= . . = F(Vt)
Dans ce cas
- l'élément Vi sera rangé dans le vecteur à la position F(Vi)
- les autres Vj, Vk, .. , Vt seront rangés dans un vecteur V' de telle sorte qu'ils soient liés entre eux comme le montre la figure ci-dessous :

1	5	0
2	6	1
3	7	0
4	8	0
5	9	0
6	10	3
7		
8	12	2
9	3	0
10		

1	16	4
2	2	5
3	0	0
4	26	0
5	32	0
6		
7		
8		
9		
10		

F(V) = (V + 5) Mod N + 1
N = 10
Valeurs : 6, 12, 10, 9, 8, 7, 16, 2, 3, 0, 5, 26, 32.

L'élément 10 est rangé à la place 6 car F(10) = 6
Dans l'exemple on a F(6) = F(16) = F(26) = 2; le premier élément 6 est rangé à la place 2 et est lié au second élément 16 par l'indice 1; 16 et 26 sont rangés dans V' et sont liés entre eux par l'indice 4.; l'élément 7 est rangé dans V à la position 3; il n'est lié à aucun élément.

Exercice 3 : Tri de couleurs

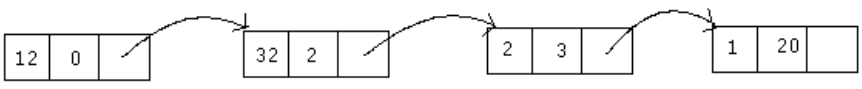
Soit un vecteur de N éléments contenant 3 types de couleurs : vert(V), rouge(R) et blanc(B) . Ecrire un algorithme qui ordonne les couleurs comme suit :

RRR... BBBBB... VVVVV

* * * * *

Exercice 1 : Gestion des polynômes
veut écrire un ensemble de procédures permettant d'effectuer du calcul algébrique sur des polynômes à une variable. On choisit d'exprimer un polynôme sous la forme d'une liste linéaire chaînée. A chaque élément est associé un monôme.

Exemple : le polynôme 12 + 32x2 + 2x3 + x20 (I)
s'exprimera sous la forme :



a) Ecrire un algorithme dont la donnée est une chaîne de caractères constituant un polynôme (que l'on convient de terminer par le caractère '*') et dont le résultat est une liste linéaire chaînée représentant ce polynôme.

b) Ecrire une action composée qui calcule la valeur d'un polynôme en un point donné.

c) Ecrire une action composée qui construit le polynôme somme de deux polynômes P1 et P2 donnés.

d) Ecrire une action composée qui construit le polynôme produit de deux polynômes P1 et P2 donnés.

e) Ecrire une action composée qui construit le polynôme dérivé d'un polynôme donné.

f) Ecrire une action composée qui imprime sous la forme (I) un polynôme P se trouvant en mémoire.

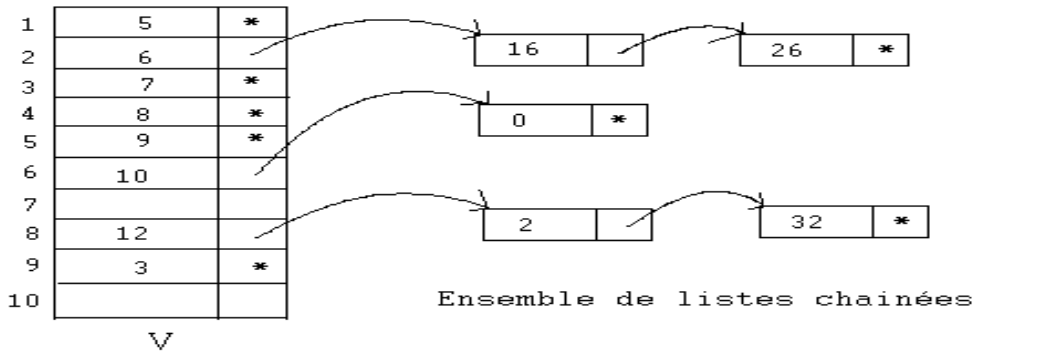
g) Programmer le module a)

NB. Toutes les questions sont indépendantes. On utilisera le module Nombre(Chaine) qui convertit la chaîne de caractères Chaine en un entier.

Exercice 2 : Accès par fonction

On a défini sur les vecteurs deux types d'accès
- l'accès séquentiel
- l'accès dichotomique

On peut définir un autre type d'accès que l'on peut décrire comme suit : Soit à construire un vecteur V à partir d'une liste de N valeurs V1,V2,.. VN. On définit une fonction F qui à toute valeur Vi attribue un entier compris entre 1 et N. L'élément Vi est alors rangé à la place F(Vi). Comme on ne peut dans la pratique trouver une fonction F qui attribue une valeur différente à chaque élément de V, il se peut que l'on ait :
F(Vi) = F(Vj) = F(Vk)= . . = F(Vt)
Dans ce cas
- l'élément Vi sera rangé dans le vecteur à la position F(Vi),
- les autres Vj, Vk, .. , Vt seront rangés dans une liste linéaire chaînée de telle sorte qu'ils soient liés entre eux comme le montre la figure ci-dessous:




```
Exemple avec f(v)= (v +5) Mod N + 1
N = 10
Valeurs = 6,12,10,9,87,16,2,3,0,5,26,32
```

L'élément 10 est rangé à la place 6 car F(10) = 6
Dans l'exemple on a F(6) = F(16) = F(26) = 2; le premier élément est rangé à la place 2; 16 et 26 dans une liste linéaire chaînée; le deuxième champ de l'élément 2 du vecteur contient le pointeur de tête de cette liste. Etc..

Ecrire un algorithme permettant d'insérer un ensemble de valeurs.

* * * * *

Partie A

Soit une suite de caractères se terminant par le caractère ".". Chaque lecture délivre le prochain caractère de la suite.

Ecrire les algorithmes suivants :

1 - Compter le nombre de mots commençant par "M" et se terminant par "S".
2 - Imprimer pour chaque mot rencontré : le mot, sa longueur, le nombre de voyelles et le dernier caractère.

Partie B

Soit une suite de N nombres (N>3). Chaque lecture délivre le prochain nombre.
Ecrire les algorithmes suivants :

1 - Rechercher le premier nombre dont le carré est égal à la somme des 3 précédents.
2 - calculer la somme (-1)i . Xi / i i = 1,2,. ,N
pour N et x donnés.

* * * * *

Exercice 1 : Analyse des constantes "virgule flottante"

Soit sur le ruban de la machine-caractères une suite de constantes réelles représentées en *Virgule flottante*. La suite est terminée par le caractère '$'. Les constantes sont séparées par une combinaison de blancs et de virgules. Ecrire un algorithme qui donne pour chaque constante correcte ses attributs W et D. W étant le nombre total de caractères et D le nombre de caractères de la partie décimale. Une constante en virgule flottante a la forme suivante :

[+/-] [chchch...] [.][chchch....] E [+/-]chch

Les crochets désignent les parties facultatives, le symbole / désigne ou.

Exercice 2 : Classement

Dans une classe de 30 élèves en vue d'établir un classement à partir des notes obtenues en 9 compositions et des coefficients correspondants, calculer pour chaque élève la somme des produits de chaque note par son coefficient et la moyenne correspondante.
NB. Vous devez dire d'abord comment vous organisez les données avant d'écrire l'algorithme correspondant. On utilisera le module de tri sans le développer.

* * * * *

Exercice 1 : Couples parfaits

Imprimer tous les couples (A, B) de nombres entiers compris entiers 1 et 1000 tels que A + B est parfait. (Un nombre est parfait s'il est égal à la somme de ses diviseurs 1 inclus, N exclu)

Exercice 2 : Mots de la forme X ...Y....Z

Sur la machine-caractères déterminer les mots de la forme X...Y...Z de longueur L donnée.

*** * * * ***

Exercice 1 : Tri par insertion

Soit un vecteur T[1..N] à valeurs numériques. Trier ce vecteur selon la méthode suivante:

1) Supposer T[1..K] trié (K<N), insérer l'élément T(K+1) dans le sous-vecteur T[1..K+1] à sa bonne position.

2) Faire varier K de 1 à N-1 dans 1)

Exercice 2 : Représentation des polynômes

Soit le polynôme suivant :
P(x) = a0xn + a1xn-1+ .... an.
(i) - Donner 2 façons de représenter le polynôme. L'une en représentant les termes nuls et l'autre sans les représenter.
(ii) - Ecrire les modules suivants:
Dérivé(P, P') : dérivé du polynôme P, résultat dans P',
Valeur(P, X) : valeur de P pour X donné
dans chacune des 2 représentations.
(iii) - Ecrire l'algorithme qui calcule la valeur P(X) pour X donné en utilisant la factorisation de Horner :
P(x) = (...(a0x + a1)x + ... an-1)x + an,
dans chacune des 2 représentations.

* * * * *

Exercice 1 : Interclassement de deux vecteurs ordonnés

Soient deux vecteurs V1[1..N] et V2[1..M] ordonnés en ordre croissant. Construire un troisième vecteur ordonné V3[1..N+M] ordonné dans le même sens contenant tous les éléments de V1 et V2. Ces derniers ne sont parcourus qu'une seule fois.

Exercice 2 : Permutation de deux sous vecteurs

Soit un vecteur T de N éléments à valeurs quelconques et un entier P tel que 1 <= P <= N. Ecrire un algorithme qui permute les deux sous vecteurs T[1..P] et T[P+1..N] sans utiliser un vecteur intermédiaire. Exemple : Si T = (5, 12, 14, 3, 2, 8) et P = 3 alors après permutation T devient (14, 3, 2, 8, 5, 12).

Exercice 3 : Insertion par fonction

Utiliser la fonction F(X) = X Mod 10 pour ranger les éléments suivants dans le vecteur V[0..9] :
2, 15, 8, 12, 16, 25

L'élément X est rangé dans la case F(X) à moins que celle-ci est peine auquel cas il est rangé dans la première case libre fournie par la séquence cyclique suivante :

F(X)+1, F(X)+2, .. M-1, 0, 1, ..

Concevoir l'algorithme d'insertion de K éléments dans le vecteur V[0..N-1] au moyen de la fonction

F(X) = X Mod N

* * ** *

Exercice 1 : X ?, Y ?

Etablir les traces des algorithmes suivants, puis dire ce qu'ils font en donnant la formule mathématique si c'est possible :

```
ALGORITHME X   ALGORITHME Y
VAR I, S : ENTIER   VAR I, J, N : ENTIER
DEBUT                                   DEBUT
 Lire(N)     Lire(N, i)
 I := 1      Ecrire(1)
 % Fonction x %     J := 1
 S := 1      TANTQUE J <= N :
 TANTQUE I < N :     Ecrire( I*J)
  S := S + I*7     J :=J + 1
  I := I + 2   FINTANTQUE
 FINTANTQUE   FIN
 Ecrire(S)
FIN
```

Exercice 2 : Calcul de la somme 1/2 - 1/3 + 1/4 - .....

Ecrire l'algorithme qui calcule la somme 1/2 - 1/3 + 1/4 - 1/5 + 1/6 - .... avec une précision & donnée.

Exercice 3 : Nombre parfait

Un nombre N est dit *parfait* s'il est égal à la somme de ses *diviseurs*, N exclu (Exemple: 6 = 1 + 2 + 3 est un nombre parfait) Ecrire :
- l'algorithme qui reconnaît si un entier N positif donné est parfait ou non.
- l'algorithme qui détermine tous les nombres parfaits de 1 à 1000.

Exercice 4 : Recherche de couples avec conditions

Ecrire l'algorithme qui imprime tous les couples entiers (A, B) dans l'intervalle [N .. M] tels que la somme A2 + B2 est soit un *multiple de 5* soit un *nombre impair*. N et M donnés.

* * * * *

Programmation PASCAL

Exercice 1 : impression de V en PASCAL

Ecrire la procédure PASCAL *imprimeV (C, X)* qui imprime une grande lettre V avec le caractère C pour Ximpair donné.

Si C = '1', Le V sera écrit comme suit :

```
1  <--  X blancs  -->  1
 1 <--(X-2) blancs->  1
  1   etc...         1
   1               1
    1             1
     1           1
      1         1
       1       1
        1     1
         1   1
          1 1
           1
```

Exercice 2 : Mots de la forme ....XY..... , mots de la forme .....X.....Y.....

Sur la machine-caractères écrire les algorithmes qui déterminent :

a) les mots de la forme .....XY..... c'est à dire n caractères suivis de la chaîne 'XY' suivis de m caractères. n et m sont des entiers quelconques et strictement supérieurs à 0.

b) les mots de la forme ...X....Y... c'est à dire c'est à dire n caractères suivis de la lettre 'X' suivis de m caractères suivis de la lettre 'Y' suivi de p caractères. n ,m et p sont des entiers quelconques et strictement supérieurs à 0.

Exercice 3 : Variétés

On veut écrire un seul algorithme, soit L, qui recherche dans une suite de nombres (N > 2) les éléments suivants :
a) le dernier nombre s'il existe dont le carré est égal à la somme des deux précédents,
b) les 3 plus grands nombres,
c) les éléments qui sont des factorielles de nombres.

1) Déterminer le module nécessaire (sans le développer) pour chaque cas (a, b et c) en définissant clairement ses entrées et sorties.

2) En utilisant ces modules, donner le corps de l'algorithme L.

3) Développer les modules définies dans a).
Exemple : si on prend la suite suivante de nombres :
6, 0, **1**, 32, 23, **45**, 13, 3, 4, 7, **24**, **6**, **120, 51**, 13, **8,** 34, 9

alors L fournit les résultats (en gras) :

a) ===> **8** car 8 X 8 = 51 + 13 (on ne prendra pas 4 bien que 4X4 = 13 + 3 car ce n'est pas le dernier)
b) ===> **45**, **120** et **51**
c) ===> **6**, **1**, **24**, **6** et **120**

* * * * *

PASCAL

Exercice 1 : Visibilité des objets

Soit le programme PASCAL suivant :

```
PROGRAM P;

VAR A, B, C, D : INTEGER;

PROCEDURE P1 (I, J : INTEGER ; VAR A: INTEGER );
BEGIN
 WRITELN('*I = ', I);
 A := (I+ J ) * B
END;

PROCEDURE P2;
BEGIN
 A := B Div C;
 B := A;
 WRITELN('** B = ', B);
END;
```

```
PROCEDURE P3 ( A, B, C : INTEGER);
BEGIN
 C := 2 ;
 C := A + B + C;
 WRITELN('*** C = ', C);
END;

BEGIN
 READ(A, B, C, D);
 P2; WRITELN('Apres p2, A = ', A);
 P3 (A, B, D); WRITELN('Apres p3, D = ', D);
 p1 (A, B, D); WRITELN('Apres p1, D = ', D);
 WRITELN(' Contenu des variables ', A, B, C, D);
END.
```

- Quelle est la portée de tous les objets définis dans le programme.

- Dérouler le programme pour a = 2; b = 3, c = 4 et d = 5.

Exercice 2 : Opérations sur les chaînes de caractères

On peut représenter un mot (chaîne de caractères ne contenant pas de blancs) dans un vecteur MOT(1..N) où chaque élément contient un caractère. N étant la taille maximale des mots.

Par exemple le mot 'abcdeabf' est représenté dans le tableau comme suit :

a	b	c	d	e	a	b	f		
1	2	3	4	5	6	7	8	9	10

Ecrire les modules suivants :

(i) Longueur (MOT) : pour déterminer la longueur du mot MOT.

(ii) Extract(MOT, I, J) : pour extraire le sous-mot du mot MOT commençant à la position I et ayant J comme longueur. O sinon.

(iii) Index(MOT2, MOT1) : pour déterminer la position du mot MOT1 dans le mot MOT2. 0 sinon.

Exemples :

Si Mot = 'abcdeabf' alors :
Longueur(MOT) = 8
Extract(MOT, 5, 2) = 'ef'
Extract(MOT, 2, 5) = 'bcdea'
Si T est le vecteur contenant la chaîne 'ab', Index(MOT, T)=1
Si T est le vecteur contenant la chaîne 'abf', Index(MOT, T)=6
Si T est le vecteur contenant la chaîne 'abx', Index(MOT, T)=0

* * * * *

Exercice 1 : Mises à jour sur vecteur

Soit à ranger un ensemble de données dans un vecteur V(1..M). On suppose qu'initialement, seul le sous-vecteur V(1..N) est rempli (N = M Div 2). Puis, des mises à jour (insertions et suppressions) très fréquentes sont effectuées sur ce vecteur. Une insertion se fait toujours après une valeur donnée. Une suppression est faite logiquement, c'est à dire par le positionnement d'un bit faisant ainsi apparaître un" trou". Quand le nombre de "trous" devient supérieur à 1/3 du nombre de données effectivement présentes, un algorithme de récupération de "trous" doit être lancé. Le traitement s'effectue alors sur le même vecteur.

- Définir la structure de données (c'est à dire la description algorithmique du vecteur V).
- Comment l'initialiser ?
- Ecrire le module de recherche R.
- Utiliser R pour écrire l'algorithme I d'insertion d'une valeur V donnée après une valeur v' donnée du vecteur.
- Utiliser R pour écrire l'algorithme de suppression.
- Quel est le temps moyen de R et de I (Nombre de comparaisons et d'affectations).
- Quel est le gain sur le temps de R après le lancement de l'algorithme de récupération de "trous".

Exercice 2 : Représentation d'un tableau de 3 dimensions en mémoire

En partant de l'hypothèse
- qu'un tableau à 3 dimensions est un vecteur de vecteurs de vecteurs,
- et qu'un vecteur est une suite contiguë d'éléments en mémoire,

écrire un algorithme qui engendre tous les éléments dans leur ordre de rangement en mémoire puis donner la formule donnant la position de l'élément T(i, j, k) dans le tableau T(1..N, 1..M, 1..P). Généraliser la formule pour un tableau à n dimensions.

* * * * *

Exercice 1 : Palindromes

Soit un texte se terminant par un point. Le texte est composé de mots de longueur maximale 20 caractères; ces mots sont séparés par des blancs. Ecrire un algorithme permettant de compter le nombre de mots palindromes (mots égaux à leur image miroir. Exemple : LAVAL, ETE)

Exercice 2 : Epreuve de sélection

Pour recruter de nouveaux employés une entreprise organise une épreuve de sélection comportant un oral et un écrit. L'écrit est en fait un questionnaire à choix multiples. Chaque candidat doit répondre à 5 questions. Pour chaque question, 3 réponses sont proposées et le candidat doit choisir la (ou les) réponse(s) qui lui semble(ent) convenir en la (les) cochant sur la feuille de réponse. Une même question peut admettre plusieurs bonnes réponses.

La notation de épreuve est faite comme suit :

pour chaque question
- avoir coché une bonne réponse donne 5 points,
- avoir coché une mauvaise réponse retire 4 points.

Les candidats ayant obtenu plus de 8 points sont admis à passer l'oral. Le nombre de candidats n'est pas connu mais il ne dépasse pas 100.

Ecrire un algorithme qui imprime :
a) la liste des candidats admis à passer l'oral avec la note obtenue à l'écrit.
b) la liste des candidats refusés.

Remarque : les deux listes ne doivent pas être mélangées.

c) après avoir donné la représentation des données (sur le fichier de données) et la représentation des résultats sur listing, traduire l'algorithme en PASCAL.

* * * * *

Exercice 1 : Losange

Ecrire une procédure en PASCAL qui imprime un losange avec le caractère "o" sachant que la diagonale, soit la ligne i , contient x blancs. Les lignes i-1 et i+1 contiennent chacune x-2 blancs, les lignes i-2 et i+2 contiennent chacune x-4 blancs, etc.. pour x impair donné.

Exercice 2 : Mots de la forme X.....XY.....Y

Sur la machine-caractères, déterminer les mots de la forme X.....XY.....Y avec au plus 3 caractères entre X et X et au moins 3 caractères entre Y et Y.

Exercice 3 : Reconnaissance de constantes arithmétiques

Sur la machine-caractères se trouvent des constantes arithmétiques ayant la forme suivante :

[chchch ch] [.] [chch ch]

c'est à dire n caractères (n>=0) suivis éventuellement du point décimal(".") et de m chiffres (m >= 0). Les crochets désignent une partie facultative.

Ecrire un algorithme qui détermine toutes les constantes en précisant pour chacune d'elle le nombre total de caractères et le nombre de chiffres après le point décimal.

NB. Sur le ruban de la machine-caractères, une constante est un mot.

* * * * *

Exercice 1 : Suite Un = 3 Un-1 + 2 Un-2 + Un-3

Ecrire l'algorithme qui calcule le N-ième terme de la suite :

U0= U1= U2 =1
Un = 3Un-3 + 2Un-2 + Un-1
N étant une donnée.

Exercice 2 : Nombres premiers

Ecrire un algorithme qui liste tous les nombres premiers de 1 à N. N étant donné.

* * * * *

Exercice 1 : M

Sur la machine caractères, écrire l'algorithme qui détermine les mots commençant par 'M'.

Exercice 2 : MME

Sur la machine caractères, écrire l'algorithme qui détermine les mots se terminant par 'MME'.

Exercice 3 : 3E

Sur la machine caractères, écrire l'algorithme qui détermine les mots de longueur L (L donné) contenant trois 'E'.

* * * * *

Partie 2.

Corrigés de sujets d’examens

Exercice 1 : 2^N

Calcul de 2^N : première méthode

```
        ALGORITHME Puissance
 VAR Res, I, N : ENTIER
 DEBUT
  LIRE(N)
  Res := 1
  POUR I=1, N :
   Res := 2 * Res
  FINPOUR
  ECRIRE(Res)
 FIN
```

Calcul de 2^N : deuxième méthode

```
    ALGORITHME Puissance
 VAR Res, I, N : ENTIER
 DEBUT
  LIRE(N)
  Res := 1
  POUR I=1, N :
   Res := Res + Res
  FINPOUR
  ECRIRE(Res)
 FIN
```

Exercice 2 : Extremum

On suppose k>0.

```
 ALGORITHME Minmax
 VAR Min, Max, K, I, Nombre : ENTIER
 DEBUT
  LIRE(K)
  LIRE(Nombre)
  Min, Max := Nombre
  POUR I=2, K :
   LIRE(Nombre)
   SI Nombre > Max : Max := Nombre FSI
   SI Nombre < Min : Min := Nombre FSI
  FINPOUR
  ECRIRE(Min, Max)
 FIN
```

Exercice 3 : F ?

Première itération (i=n), Res prend la valeur n
Deuxième itération (i=n-1), Res prend la valeur n(n-1)
...

(n-1)-ième itération (i=2), Res prend la valeur n(n-1)..2

L'algorithme calcule donc la factorielle de n.

Exercice 4 : PGCD

Calcul du pgcd de A et B.

```
ALGORITHME Pgcd
VAR A, B, X, Y, Rest : ENTIER
DEBUT
 LIRE(A, B)
 SI A > B :
  X := A ; Y := B
 SINON
  X := B ; Y := A
 FSI
 Rest := MOD(X, Y)
 TANTQUE Rest # 0 :
  X := Y
  Y := Rest
  Rest := MOD(X, Y)
 FINTANTQUE
 ECRIRE(Y)
FIN
```

Exercice 5 : Calcul

E3 := 40, 13, 6
R := 13.33, 6.5

* * * * *

1. Aspect algorithmique

A-a) Somme des nombres positifs et celle des nombres négatifs

```
ALGORITHME Somme
VAR
 N, Somp, Somn, I, Nombre
DEBUT
 Somp := 0
 Somn := 0
 LIRE (N)
 POUR I=1, N
```

```
 LIRE(Nombre)
 SI Nombre > 0 :
  Somp := Somp + Nombre
 SINON
  Somn := Somn + Nombre
 FSI
FINPOUR
ECRIRE (Somp, Somn)
FIN
```

A-b) Nombre de sous-suites croissantes

```
  ALGORITHME Nombre
VAR N, Prec, Nbre, Nb, I : ENTIER
DEBUT
 Nb := 1
 LIRE(N)  ; LIRE(Prec)
 POUR I=2, N
  LIRE(Nbre)
  SI Prec < Nbre :
   Nb := Nb + 1
  FSI
  Prec := Nbre
 FINPOUR
 ECRIRE( Nb )
FIN
```

B. Recherche des nombres amis inférieurs à 1000.

```
ALGORITHME Amis
VAR I, J, K, A, N
DEBUT
 POUR I=1, 1000 :
  POUR J= I, 1000 :
   A := I2 + J2
   { Est-ce un carré parfait ? }
   N := A DIV 2
   Carré := FAUX
   K := 1
   TANTQUE K <= N ET NON Carré
    SI K2 = A :
     ECRIRE(I, J)
     Carré := VRAI
    FSI
    K := K + 1
   FINTANTQUE
  FINPOUR
 FINPOUR
FIN
```

2. Aspect Programmation

2-a) Traduction de B)

```
PROGRAM Amis;
VAR  I, J, K, A, N : INTEGER;
 Carre : BOOLEAN;
BEGIN
 FOR I:=1 TO 1000 DO
  FOR  J:= I TO 1000  DO
   BEGIN
    A := SQR(I) + SQR(J);
    { Est-ce un carré parfait ? }
    N := A DIV 2;
    Carré := FALSE;
    K := 1;
    WHILE( K <= N AND NOT Carré ) DO
     BEGIN
      IF SQR(K) = A THEN
      BEGIN
       WRITELN(I, J);
       Carré := TRUE
      END;
      K := K + 1
     END
   END
END.
```

2-b) Impression du dessin

```
PROGRAM Triangle;
VAR N, I, J : INTEGER ;
BEGIN
 READ(N);
 WRITELN('*', 21);
 WRITELN('**', 22);
 FOR I:=1 TO N DO
  BEGIN
   WRITE('*': 21);
   FOR J:=1 TO I DO WRITE(I:1);
   WRITELN('*');
  END
END.
```

* * * * *

Exercice 1 : Calcul de la somme 1 + 5 + 9 + . .

```
        ALGORITHME Somme
VAR S, I, N : ENTIER
DEBUT
 LIRE(N)
 S := 0
```

```
 POUR I=1, 4*N + 1, 4 :
  S := S + I
 FINPOUR
 ECRIRE(S)
FIN
```

Exercice 2 : Calcul de 2N avec l'addition

```
ALGORITHME Puissance
VAR Res, I, N : ENTIER
DEBUT
 LIRE(N) ;  Res := 0
 POUR I=1, N:
  Res := Res + Res
 FINPOUR
 ECRIRE(Res)
FIN
```

Exercice 3 : Comptage du nombre de mots

```
ALGORITHME Nbremotl
VAR  C , Mot : CAR
 K, L, Compt : ENTIER
DEBUT
 LIRE(L) ;  LIRE(C)
 Compt := 0
 TANTQUE C # '.' :
  TANTQUE C = ' ' : LIRE(C) FINTANTQUE
  Mot := '' ;  K := 0
  TANTQUE C # ' ' ET C # '.' :
   K := K + 1 ;    Mot := Mot + C
   LIRE(C)
  FINTANTQUE
  SI K = L : Compt := Compt + 1 FSI
 FINTANTQUE
 ECRIRE(Compt)
FIN
```

* * * * *

Exercice 1 : Calcul de la somme 13 + 33 + 53 + . . .

```
ALGORITHME Somme
VAR S, I, N : ENTIER
DEBUT
 LIRE(N) ;  S := 0
 POUR I=1, 2*N + 1, 2 :
  S := S + I3
 FINPOUR
```

```
 ECRIRE(S)
FIN
```

Exercice 2 : Division de 2 entiers A et B avec l'opérateur '-'

```
ALGORITHME Division
VAR A, B, Q : ENTIER
DEBUT
 LIRE(A, B) ; Q := 0
 TANTQUE A >= B :
  A := A - B
  Q := Q + 1
 FINTANTQUE
 ECRIRE(Q, A)  { A étant Le Reste}
FIN
```

Exercice 3 : Longueur du mot le plus long

```
ALGORITHME Lemotpluslong
VAR  C   : CAR
 K, L : ENTIER
DEBUT
 LIRE(C)
 L := 0
 TANTQUE C # '.' :
  TANTQUE C = ' ' : LIRE(C) FINTANTQUE
  K := 0
  TANTQUE C # ' ' ET C # '.' :
   K := K + 1
   LIRE(C)
  FINTANTQUE
  SI K > L : L := K FSI
 FINTANTQUE
 ECRIRE(L)
FIN
```

* * * * *

1. Aspect algorithmique

a) Recherche du minimum et du maximum

On suppose k>0

```
ALGORITHME Minmax
VAR Min, Max, K, I : ENTIER
DEBUT
 LIRE(K)
 LIRE(Nombre)
```

```
 Min, Max := Nombre
 POUR I=2, K :
  LIRE(Nombre)
  SI Nombre > Max : Max := Nombre FSI
  SI Nombre < Min : Min := Nombre FSI
 FINPOUR
 ECRIRE(Min, Max)
FIN
```

b) Recherche du premier nombre dont le carré est égal à la somme des 2 précédents :

```
ALGORITHME Carré
VAR  A, B, C, I, N  : ENTIER
 Trouv   : BOOLEEN
DEBUT
 LIRE(N)
 LIRE(A, B)
 Trouv := FAUX  ;  I := 2
 TANTQUE I < N ET NON Trouv :
  LIRE(C)
  I := I + 1
  SI C2 = A + B :
   Trouv := VRAI
  SINON
   A := B
   B := C
  FSI
 FINTANTQUE
 SI Trouv : ECRIRE(C) SINON ECRIRE( ' Inexistant' ) FSI
FIN
```

c) Nombre de sous-suites croissantes

```
ALGORITHME Nombre
VAR N, Prec, Nbre, Nb, I : ENTIER
DEBUT
 Nb := 1
 LIRE(N)  ; LIRE(Prec)
 POUR I=2, N
  LIRE(Nbre)
  SI Prec > Nbre :
   Nb := Nb + 1
  FSI
  Prec := Nbre
 FINPOUR
 ECRIRE( Nb)
FIN
```

2. Aspect programmation

a) Traduction de c)

```
PROGRAM Nombre;
VAR N, Prec, Nbre, Nb, I : INTEGER ;
BEGIN
 Nb := 1;
 READ(N)  ; READ(Prec) ;
 FOR I:=2 TO N DO
  BEGIN
   READ(Nbre) ;
   IF Prec > Nbre THEN Nb := Nb + 1 ;
   Prec := Nbre
  END;
 WRITE( Nb )
END.
```

b) Dessin demandé

Se référer au programme P1.

* * * * *

Exercice 1 : Moyenne

Algorithme :

```
ALGORITHME Moyenne
VAR Quantité, Moy  : REEL
 I, Nbre, N   : ENTIER  Mesure   : TABLEAU(1..N) DE REEL
DEBUT
 LIRE(Quantité, N)
 LIRE(Mesure)  { lecture globale du tableau }
 Moy := 0
 POUR I=1, N :
  Moy := Moy + Mesure(I)
 FINPOUR
 Moy := Moy / N
 Nbre := 0
 POUR I=1, N :
  SI Absolu( Moy - Mesure(I) ) > Quantité
   Nbre := Nbre + 1
  FSI
 FINPOUR
 ECRIRE(Moy, Nbre)
FIN
```

Programme PASCAL correspondant :

```
PROGRAM Moyenne ;
VAR Quantité, Moy  : REAL;
 I, Nbre, N   : INTEGER;
```

```
 Mesure   : Array[1..N] OF REAL ;
BEGIN
 READ(Quantité, N);
 { Lecture du tableau des mesures }
 FOR I:=1 TO N DO READ(Mesure[I] );
 Moy := 0;
 FOR I:=1 TO N DO Moy := Moy + Mesure[i]  ;
 Moy := Moy / N ;
 Nbre := 0;
 FOR I:= 1 TO N DO
  IF ABS( Moy - Mesure[i] ) > Quantité
  THEN Nbre := Nbre + 1 ;
 WRITE(Moy, Nbre)
END.
```

Exercice 2 : Reconnaissance de données arithmétiques

Dans l'algorithme qui suit, Mot désigne une variable chaîne de caractères, W et D des variables entières. W désigne le nombre total de caractères de la constante et D le nombre de chiffres après le point décimal.

On utilise le module Chiffre défini comme suit :

Chiffre (C) = vrai si C est un chiffre faux sinon

L'algorithme est :

```
 ALGORITHME Constante_PL1
 TYPE T = STRUCTURE
  Mot  : CAR
  W, D : ENTIER
 FIN
 VAR  C, Mot : CAR
  I, K, N, W, D  : ENTIER
  Tab   : TABLEAU(1..N) DE T
 DEBUT
  LIRE(N)
  LIRE(C)
  I := 0
  TANTQUE C # '#' :
   Mot := ''  ;  W := 0;  D := 0
   { Parcours des blancs }
   TANTQUE  C= ' ' OU C=',' :  LIRE(C)   FINTANTQUE

   { Reconnaissance de la partie entière }
   TANTQUE Chiffre(C)
    W := W + 1;  Mot := Mot + C ; LIRE(C)
   FINTANTQUE

   { Partie décimale si elle existe }
   SI C = '.' :
```

```
 W := W + 1 ; Mot := Mot + C ; LIRE(C)
 TANTQUE Chiffre(C) :
  W := W + 1 ; D := D + 1;
  Mot :=  Mot + C;  LIRE(C)
 FINTANTQUE
FSI

{En cas d'erreur }
SI C #' ' ET C #',' ET C #'#' :
 ECRIRE( ' Constante Erronée ')
 TANTQUE  C# '  ' ET C # ',' ET C #'#'  :
  Mot := Mot + C
  LIRE(C)
 FINTANTQUE
SINON
 SI W # 0 :
  Tab(I).Mot := Mot
  Tab(I).W := W
  Tab(I).D := D
 FSI
FSI
FINTANTQUE
ECRIRE(Tab)
 FIN
```

Programme PASCAL correspondant :

Les fichiers D_Chiffre.pas et R_chiffre.pas contiennent par exemple les données et les résultats de l'énoncé respectivement.

```
PROGRAM Constantes_PL1;
  TYPE
    T = RECORD
      Mot  : STRING[16];
      W, D : INTEGER
    END;

 FUNCTION Chiffre( C : CHAR) : BOOLEAN;
   BEGIN
     IF (C >='0' ) AND ( C <= '9')
     THEN Chiffre := TRUE
     ELSE Chiffre := FALSE ;
   END;

 VAR
   C : CHAR;
   Mot : STRING[16];
   K, I, W, D  : INTEGER ;
   Tab : ARRAY(.1..10.) OF T;
   Fe, Fs : TEXT;
 BEGIN
   ASSIGN(Fe, 'D_Chiffre.Pas');
```

```
 ASSIGN(Fs, 'R-Chiffre.pas');
 RESET(Fe);
 REWRITE(Fs);
 READ(Fe, c);
 I := 0;
 WHILE C <> '#' DO
  BEGIN
Mot := ''  ;  W := 0;  D := 0;
      { Parcours des blancs }
WHILE  (C= ' ') OR (C=',') DO  READ(Fe, c)  ;

      { Reconnaissance de la partie entière }
WHILE Chiffre(C) DO
           BEGIN
              W := W + 1;  Mot := Mot + C ; READ(Fe, c)
    END   ;

      { Partie décimale si elle existe }
       IF C = '.'
      THEN
           BEGIN
         W := W + 1 ; Mot := Mot + C ; READ(Fe, c);
         WHILE Chiffre(C) DO
          BEGIN
  W := W + 1 ; D := D + 1;
  Mot :=  Mot + C;  READ(Fe, c);
           END
       END;

      {En cas d'erreur }
IF (C <>' ') AND (C <>',') AND ( C <> '#' )
      THEN
       BEGIN
         WRITE( Fs, '* * Constante erronée  :  ');
         WHILE  (C <> ' ') AND (C<>',') AND ( C <> '#' )  DO
           BEGIN
              Mot := Mot + C;
              READ(Fe, c)
           END;
         WRITELN(Fs,  Mot);
       END
ELSE
 IF W <> 0
        THEN
         BEGIN
           I := I + 1;
           Tab[I].Mot := Mot;
          Tab[I].W := W;
      Tab[I].D := D
    END;
  END;
 FOR K := 1 TO I DO
```

```
    WRITELN(Fs, TAB[K].Mot,'  W=', Tab[K].W,'  D=', Tab[K].D);
  CLOSE(Fs);
END.
```

Exercice 3 : Tri par interclassement

```
ALGORITHME Tri
VAR Taille, Bi1, Bi2, Bs1, Bs2, N, K : ENTIER
 V, V1 : TABLEAU[1..N] DE ENTIER { Ou d'un autre type }
 Aig : BOOLEEN
DEBUT
 Taille := 1
 Aig := VRAI
 TANTQUE Taille < N :
  Bi1:=1{Borne inférieure premier sous-vecteur}
  TANTQUE Bi1 + Taille <= N :
  { Indices restants}
  Bi2 := Bi1 + Taille
  Bs1 := Bi2 - 1
  SI Bi2 + Taille - 1 > N :
   Bs2 := N
  SINON
   Bs2 := Bi2 + Taille - 1
  FSI
  SI Aig :
   Interclasser(V, Bi1,Bs1,Bi2,Bs2, V1)
  SINON
   Interclasser(V1, Bi1,Bs1,Bi2,Bs2, V)
  FSI
  Bi1 := Bs2 + 1
 FINTANTQUE
 K := Bi1
 TANTQUE K <= N :
  SI Aig :
   V1(K) := V(K)
  SINON
   V(K) := V1(K)
  FSI
  K := K + 1
 FINTANTQUE
 Aig := NON Aig
 Taille := Taille * 2
FINTANTQUE
SI Aig :
 ECRIRE(V)
SINON
 ECRIRE(V1)
FSI
FIN
```

Les paramètres du module *Interclasser(V, Bi1,Bs1,Bi2,Bs2, V1)* sont définis comme suit :

V : le vecteur en entrée
Bi1 : borne inférieure du premier sous vecteur
Bs1 : borne supérieure du premier sous vecteur
Bi1 : borne inférieure du deuxième sous vecteur
Bs1 : borne supérieure du deuxième sous vecteur
V1 : le vecteur en sortie

* * * * *

Exercice 1 : Carré parfait

Un nombre A et dit carré parfait s'il existe un entier B tel que B2 = A. Nous devons donc donner à une variable i successivement les valeurs 0, 1, . A/2 et s'arrêter jusqu'à l'obtention de i2 = A auquel cas A est carré parfait ou i2 > A auquel cas A ne l'est pas.

```
ALGORITHME Carré Parfait
VAR  A, I  : ENTIER
 Carré : BOOLEEN
DEBUT
 I := - 1
 LIRE(A)
 Carré := FAUX
 TANTQUE I < (A DIV 2) ET NON Carré :
  I := I + 1
  SI I2 = A : Carré := VRAI FSI
 FINTANTQUE
 ECRIRE ( Carré )
FIN
```

Exercice 2 : Couples (A, B) tels que A = 3B

Une façon d'obtenir tous les couples est d'utiliser deux boucles imbriquées.

```
ALGORITHME Couples
VAR I, J : ENTIER
DEBUT
 POUR I=1, 100 :
  POUR J=1, 100
   SI I= 3*J :
    ECRIRE(I, J)
    J := J + 1
   FSI
  FINPOUR
 FINPOUR
FIN
```

Cette solution est loin d'être efficace. Si l'on veut avoir un algorithme meilleur, il suffit de remarquer que 1 <= I <= 99 , 1 <= J <= 33 et I = 3*J. Et de ce fait, on utilisera une seule boucle où le critère d'arrêt est soit I < 99 ou J < 33.

```
ALGORITHME Couples
VAR I, J : ENTIER
DEBUT
 POUR J=1, 33 : ECRIRE(3*J, J) FINPOUR
FIN
```

Exercice 3 : Calcul de somme

Très simple, il suffit de remarquer que :

Si = Si-1 + (4*i + 1)

```
ALGORITHME Somme
VAR I, Som, N : ENTIER
DEBUT
 LIRE(N)
 Som := 0
 POUR I=0, N :
  Som :=Som + 4*I + 1
 FINPOUR
 ECRIRE(Som)
FIN
```

Exercice 4 : Fibo ?

a) Trace

```
Fib0  0
Fib1  1
Trouv  FAUX, VRAI
K  2, 3
N  10
NON Trouv VRAI, FAUX
Fib  1
```

Ecriture du couple (3, 1)

b) Quelque soit n, n > 1 F(n) = 3 ou F(n) = 1.
Si n <= 1 il y a une boucle infinie

c) Algorithme simplifié

```
ALGORITHME F31
VAR N : ENTIER
DEBUT
 LIRE(N) { N > 1 }
 SI N > 1 : ECRIRE( 3, 1)
 SINON ECRIRE( 'Pas de solution') FSI
```

```
FIN
```

* * * * *

Partie 1 : Machine caractères

a) Recherche d'un mot donné

```
ALGORITHME Motd
VAR  Motd, C, Mot : CAR
 Trouv   : BOOLEEN
DEBUT
 LIRE(Motd)
 Trouv := FAUX
 LIRE(C)
 TANTQUE C # '.' ET NON Trouv :
  SI C=' ' : LIRE(C)
  SINON
   Mot := ''
   {Formation du mot }
   TANTQUE C # '.' ET C # ' ' :
    Mot := Mot + C
     LIRE(C)
   FINTANTQUE
   SI Mot = Motd : Trouv := VRAI FSI
  FSI
 FINTANTQUE
                ECRIRE(Trouv)
  FIN
```

b) Impression des mots commençant par 'T' et se terminant par 'S'

```
        ALGORITHME Ts
VAR C, Mot, Dernier : CAR
DEBUT
 LIRE(C)
 TANTQUE C # '.' :
  {Parcours des blancs}
  TANTQUE C = ' ' : LIRE(C) FINTANTQUE
  SI C='T' :
   Mot := ''
   TANTQUE C # '.' ET C # ' ' :
    Mot := Mot + C
    Dernier := C
    LIRE(C)
   FINTANTQUE
   SI Dernier = 'S' : ECRIRE(Mot) FSI
  SINON
   {Le premier caractère est différent de  'T'
   aller au prochain blanc}
   TANTQUE C #' ' ET C #'.' :
```

```
   LIRE(C)
  FINTANTQUE
 FSI
FINTANTQUE
  FIN
```

Partie 2 : Actions composées

a1) Calcul de ((AB +CD) / AD)A

Définissant d'abord la fonction puissance

```
FONCTION Puiss (A, B) : ENTIER
VAR A, B,  : ENTIER
DEBUT
 Puiss := 1
 POUR I= 1, B
  Puiss := Puiss * A
 FINPOUR
FIN

ALGORITHME Calcul
VAR  A, B, C, D : ENTIER
DEBUT
 LIRE(A, B, C, D)
 ECRIRE(Puiss(((Puiss(A, B) + Puiss(C, D)) / Puiss(A, D)), A))
FIN
```

a2) Traduction en PASCAL

```
PROGRAM Calcul;
VAR A, B, C, D : INTEGER;

FUNCTION Puiss ( A, B : INTEGER) : INTEGER;
VAR  I, P: INTEGER;
BEGIN
 P := 1;
 FOR I := 1 TO B DO   P := P * A ;
 Puiss := P
END;
BEGIN
 READ(A, B, C, D);
 WRITE(Puiss(((Puiss(A, B) + Puiss(C, D)) / Puiss(A, D)), A))
END.
```

b1) Fonctions F, G et le prédicat Egal

Fonction F(x)

```
FONCTION F( X ) : ENTIER
VAR
 X : ENTIER
```

```
DEBUT
 F := X**5 + X**2 + 18
FIN
```

Fonction G(x)

```
        FONCTION G( X ) : ENTIER
VAR
 X : ENTIER
DEBUT
 G := X**7 + X**3 + X + 1
FIN
```

Prédicat Egal(A, B)

```
        PREDICAT Egal (A, B)
VAR
 A, B : ENTIER
DEBUT
 Egal := ( A = B )
FIN
```

b2) Solutions de l'équation F(x)= G(x) pour x dans l'intervalle [-1000, + 1000]

```
ALGORITHME Solutions
VAR
 I : ENTIER
BEGIN
 POUR I := -1000, 1000 :
  SI Egal (F(I), G(I) ) :
   Ecrire(I)
  FSI
 FINPOUR
FIN
```

b3) Traduction en PASCAL

```
PROGRAM Solutions;
VAR
 I : INTEGER;

 { Définition des fonctions F , G et Egal }

 FUNCTION F( X  : INTEGER) : INTEGER;
 BEGIN
  F := X**5 + X**2 + 18
 END;

 FUNCTION G( X : INTEGER ) : INTEGER;
 BEGIN
  G := X**7 + X**3 + X + 1
 END;
```

```
        FUNCTION  Egal (A, B : INTEGER) : BOOLEAN;
BEGIN
 Egal := ( A = B )
END;

BEGIN
 FOR  I := -1000 TO 1000  DO
  IF Egal (F(I), G(I) ) THEN WRITE( I)
END.
```

* * * * *

Exercice 1 : Inventaire

```
TYPE Typearticle = STRUCTURE
 Numprod : T
 Pu : REAL
 Qte : ENTIER
FIN
TYPE T = STRUCTURE
 Nm : ENTIER  {Numéro de magasin}
 Nr : ENTIER {Numéro de rayon}
 Num: ENTIER
FIN
```

1) Création du fichier Fstock à partir d'un fichier TEXT

```
LIRE(Ftext, Nm, Nr,Num, Pu, Qte)
TANTQUE Nm # 0 :
 Art.Numprod.Nm := Nm
 Art.Numprod.Nr := Nr
 Art.Numprod.Num := Num
 Art.Pu := Pu
 Art.Qte := Qte
 ECRIRE(Fstock, Art)
 LIRE(Ftext, Nm, Nr,Num, Pu, Qte)
FINTANTQUE
```

2) Edition de l'état d'inventaire

```
        LIRE(Fstock, E)
Im := 1 ; Ir := 1
Mray, Mmag, Mtotal := 0 ECRIRE('Magasin N° 1)
TANTQUE NON Dernier(Fstock) :
 SI E.Numprod.Nm = Im
```

```
  SI E.Numprod.Nr = Ir :
   Mray := Mray + E.Qte * E.Pu
   Prendre(Fstock, E)
  SINON

   ECRIRE('Rayon N°', Ir, Mray )
   Mmag := Mmag + Mray
   Mray := 0
   Ir := Ir + 1
  FSI
 SINON
  ECRIRE('Rayon N°', Ir, Mray )
  Mmag := Mmag + Mray
  Mtotal := Mtotal + Mmag
  ECRIRE('Montant Total du magasin',Mmag)
  ECRIRE('Magasin N°', Im)
  Im :=Im + 1
  Ir := 1
  Mmag, Mray := 0
 FSI
FINTANTQUE
ECRIRE('Rayon N°', Ir, Mray )
Mmag := Mmag + Mray
ECRIRE('Montant Total Du Magasin',Mmag)
Mtotal := Mtotal + Mmag
ECRIRE('Montant Total Du Stock',Mtotal)
FIN
```

Programme

Se référer au programme P9.

Exercice 2 : Histogramme

```
ALGORITHME Histogramme
VAR  I, N, V  : ENTIER
 T   : TABLEAU(1..10) DE ENTIER
 Vote  : TABLEAU(M, 10)  DE CAR  { Tableau à 2 dimensions }
BEGIN
 T := -1
 Vote := ' ' {Initialisation globale}
 POUR I = 1, N :
  LIRE(V)
  T(V) := T(V) + 1
  Vote(N -T(V), V) := '*'
 FINPOUR
 ECRIRE(Vote)
FIN
```

Exercice 3 : Tri par insertion

```
ALGORITHME Tri
```

```
VAR  A, I,  K,  N, L : ENTIER
 T : TABLEAU[1..N] DE ENTIER
 Trouv : BOOLEEN
DEBUT
   POUR I = 2, N
 A := T(I)
 K := 1 ; Trouv := FAUX
 TANTQUE K <  I ET NON Trouv :
  SI A <= T(K) :
   Trouv := VRAI
  SINON
   K := K + 1
  FSI
 FINTANTQUE
 SI Trouv :
  POUR L := I, K+1, -1 :
   T(L) := T(L-1)
  FINPOUR
  T(K) := A
 FSI
FINPOUR
  FIN
```

* * * * *

Exercice 1 : Variétés

a) Action Prem

```
   ACTION Prem(Nbr, Bool)
VAR  Nbr, J, N, Q  : ENTIER
 Bool : BOOLEEN
DEBUT
 N := Nbr DIV 2
 J := 2
 Bool := VRAI
 TANTQUE J < N ET Bool :
  Q := Nbr DIV 2
  SI J*Q = Nbr : Bool := FAUX SINON J := J + 1 FSI
 FINTANTQUE
FIN
```

b) Action Carré

```
ACTION Carre(Nbr, Bool)
VAR  Nbr, J, N : ENTIER
 Bool : BOOLEEN
DEBUT
 SI Nbr = 1 : Bool := VRAI
```

```
 SINON
  N := Nbr DIV 2
  J := 2
  Bool := FAUX
  TANTQUE J <= N ET NON Bool :
   SI J*J = Nbr :
    Bool := VRAI
   SINON J := J + 1 FSI
  FINTANTQUE
 FSI
FIN
```

c) Action Pair

```
ACTION Pair(Nbr, Estpair)
VAR  Nbr, Q : ENTIER
 Estpair : BOOLEEN
DEBUT
 Estpair := FAUX
 Q := Nbr DIV 2
 SI Q*2 = Nbr :
  Estpair := VRAI
 FSI
FIN
```

d) Action Nbdiv

```
ACTION  Nbdiv(Nbr, Nbdiv)
VAR  N, Nbr, J, Nbdiv, Q : ENTIER
DEBUT
 SI Nbr = 1 : Nbdiv := 1
 SINON
  N := Nbr DIV 2
  Nbdiv := 2
  J := 2
  TANTQUE J < N :
   Q := Nbr DIV J
   SI Q*J = Nbr :
    Nbdiv := Nbdiv + 1
   FSI
   J := J + 1
  FINTANTQUE
 FSI
FIN
```

Algorithme répondant aux questions posées

```
ALGORITHME Unseul
VAR N, Nbre, I, I1, I2, Dernierimpair, Nb  : ENTIER
 Estpair, Exist, Oui1, Oui2, Oui3   : BOOLEEN
DEBUT
 I1, I2 := 0
```

```
Oui1, Oui2, Oui3 := VRAI
LIRE(N)
 POUR I=1, N :
 LIRE(Nbre)
 SI Oui1 :
  Prem(Nbre, Bool)
  SI Bool :
   I1 := I1 + 1
   SI I1 = 3 :
    ECRIRE(' 3-ième nombre premier:', Nbre)
    Oui1 := FAUX
   FSI
  FSI
 FSI
 SI Oui2 :
  Carré(Nbre, Bool)
  SI Bool :
   I2 := I2 + 1
   SI I2 = 2 :
    ECRIRE('2-ième carré parfait:', Nbre)
    Oui2 := FAUX
   FSI
  FSI
 FSI
 SI Oui3 :
  Pair(Nbre, Estpair)
  SI Estpair :
   Nbdiv(Nbre, Nb)
   ECRIRE(' Le premier nombre pair a ', Nb,
        'diviseurs')
   Oui3 := FAUX
   FSI
  FSI
 FSI
 Pair(Nbre, Estpair)
 SI NON Estpair :
  Dernierimpair := Nombre
  Existe := VRAI
 FSI
FINTANTQUE
SI Existe :
 Nbdiv(Dernierimpair, Nb)
 ECRIRE(' Le dernier nombre impair a ',Nb, 'diviseurs')
SINON ECRIRE (' Pas DE Nombre Impair) FSI
  FIN
```

Exercice 2 : Norme euclidienne

```
ALGORITHME Euclide
VAR  X, S, Normax  : REEL
 N, I, J, P, Rang : ENTIER
DEBUT
```

```
LIRE(P) ; LIRE(N)
Normax := 0
POUR I=1, P :
 { Calcul de la norme euclidienne }
 S := 0
 POUR J=1, N
  LIRE(X)
  S := S + X*X
 FINPOUR
 SI S > Normax :
  Rang := I
  Normax := S
 FSI
FINPOUR
ECRIRE(Rang, Normax)
  FIN
```

Exercice 3 : Id ?

Trace pour X = 3 ; N = 4

Y(1, P) donne P := 1
W := 3
S := 1 + 3/1

Y(2, P) donne P := 2
W := 32 = 9
S := 1 + 3/1 + 9/2

Y(3, P) donne P := 6
W := 33 =27
S := 1 + 3/1 + 9/2 + 27/6

Y(4, P) donne P := 24
W := 34 =81
S := 1 + 3/1 + 9/2 + 27/6 + 81/24

Le programme calcule la somme : S xi / i!

* * * * *

Exercice 1 : Le plus apparent

```
ALGORITHME Maximum
VAR Max, I, J, Compt, N, Res : ENTIER
 T : TABLEAU[1..N] DE ENTIER
DEBUT
 Compt := 0
 Max := 0
```

```
 POUR I:= 1, N :
  POUR J:= I+1, N :
   SI T(I) = T(J): Compt := Compt + 1 FSI
  FINPOUR
  SI Compt > Max :
   Max := Compt
   Res := I
  FSI
 FINPOUR
 ECRIRE( T(Res) )
FIN
```

Exercice 2 : Course de ski

```
        ALGORITHME Ski
TYPE Typ = STRUCTURE
  Temps  : ENTIER
  Nom, Pays : CAR
FIN
VAR  I, J, K, N, Nbr  : ENTIER
  Trouv   : BOOLEEN
  Tab   : TABLEAU[1..10] DE Typ
DEBUT
 LIRE(N)
 Nbr := 0
 POUR I=1, N
  LIRE(Nom, Pays, Mm, Ss, Cc)
  T := (Mm*60 + Ss )*100 + Cc
  J := 1 ; Trouv := FAUX
  TANTQUE J <= Nbr ET NON Trouv
   SI T(J) >= T :
    Trouv := VRAI
   SINON  J := J + 1  FSI
  FINTANTQUE
  SI Trouv :
   POUR K:=Nbr, J, -1 : T(K+1) := T(K) FINPOUR
  FSI
  T(J).Temps := T
  T(J).Nom := Nom
  T(J).Pays := Pays
  Nbr := Nbr + 1
       POUR K = 1, Nbr :
        T:= Tab(.K.).Temps
            Ss := T DIV 100
            Cc := T - 100*Ss
            Mm := Ss DIV 60
            Ss := Ss - 60*Mm
   Nom:= Tab(.K.).Nom
   Pays := Tab(.K.).Pays
            ECRIRE(K, Pays, Nom, Mm, Ss, Cc)
     FINPOUR
```

Programmation :

Se référer au programme P8.

Exercice 3 : Edition d'une ligne

```
ALGORITHME
VAR C : CAR
 Ligne : TABLEAU[1..N] DE CAR
 I, N : ENTIE
DEBUT
 LIRE(C)
 I := 0
 TANTQUE C # '.' :
  SI C = '/' :
   I :=I-1 ; SI  I < 0 : I := 0 FSI
  SINON
   SI C ='<' :
    I := 0
   SINON
    SI C = '>' :
     I := (Quotient(I, 10) + 1) * 10
    SINON
     SI C = '#' :
      Imprimer(Ligne)
     SINON
      I := I + 1
      SI I <= 120 :
       Ligne(I) :=C
      FSI
     FSI
    FSI
   FSI
  FSI
  LIRE(C)
 FINTANTQUE
  FIN
```

* * * * *

Exercice 1 : Vérification syntaxique de déclarations FORTRAN

Définissant d'abord les modules suivants :

Sauter-blancs (K) :

```
ACTION Sauter-Blancs (K)
VAR  K : ENTIER
 C : CAR
DEBUT
 TANTQUE C=' ' ET K <= 72 :
```

```
  LIRE(C)
  K := K + 1
 FINTANTQUE
FIN
```

Erreur(I) :

```
ACTION Erreur(I)
VAR  I : ENTIER
DEBUT
  ECRIRE(Message(I))
 Sauter-ligne
FIN
```

Message (I) :

```
        ACTION Message(I)
VAR  I : ENTIER
DEBUT
 SI I=1  ECRIRE('Zone étiquette erronée')
 SINON
  SI I=2 ECRIRE('Déclaration non identifiée')
  SINON
   SI I=3 ECRIRE('Erreur dans une déclaration')
   SINON ECRIRE('Erreur dans un identificateur')
   FSI
  FSI
 FSI
FIN
```

L'algorithme est le suivant :

```
ALGORITHME Fortran
VAR C, Mot  : CAR
 I, N, L, K  : ENTIER
 Err : BOOLEEN
DEBUT
 LIRE(N)
 {Traiter N déclarations une à une}
 POUR I=1, N :
  {Zone étquette}
  K := 1  ; Err := FAUX
  TANTQUE K < 6 ET NON Err :
   LIRE(C)
   Err :=  (C # ' ')
   K := K + 1
  FINTANTQUE
  SI Err
   Erreur(1)
  SINON
   {Zone type }
   Sauter-Blancs ( K )
```

```
SI K > 72
 Erreur(2)
SINON
 {Former le type}
 M := ''
 TANTQUE C #' ' ET K<= 72 :
  M := M + C
  LIRE(C)
  K := K + 1
 FINTANTQUE
 SI K > 72 :
  Erreur(2)
 SINON
  SI Mot # 'INTEGER' OU Mot #'REAL' OU
            Mot#'Logical' :
   Erreur(2)
   Err := VRAI
  FSI
  {Analyse des identificateurs}
  TANTQUE K  <72 ET NON Err :
   Sauter-Blancs ( K)
   SI K > 72 :
    Erreur(3)
    Err := VRAI
   SINON
   SI C < 'A' : Erreur(4) ;Err:=VRAI
   SINON
    L := 1
    TANTQUE C#',' ET C#' ' ET
                  NON Err ET K <= 72 :
     LIRE(C)
     L := L + 1
     Erreur := NON (C>='1')
     K := K + 1
    FINTANTQUE
    SI K >72 OU L>8 OU Err:
    Erreur(4)
    SINON
     Sauter-Blancs (K)
     SI K<=72 ET C#',' :
      Erreur(3)
     FSI
    FSI
   FSI
  FINTANTQUE
 FSI
FSI
FINTANTQUE
FINPOUR
FIN
```

Exercice 2 : Course de ski

```
ALGORITHME Ski
VAR  Mm, Ss, Cc : ENTIER
 T1, T2, T3, T, I, N  : ENTIER
 Num1, Num2, Num3, Num  : CAR
 Nom1, Nom2, Nom3, Nom  : CAR
DEBUT
 {Initialisation}
 LIRE(N)
 T1, T2, T3 := 360 000
 Num1, Num2, Num3 := ''
 Nom1, Nom2, Nom3 := ''
 POUR I=1, N
  LIRE(Num, Nom, Mm, Ss, Cc)
  ECRIRE(Num, Nom, Mm, Ss, Cc)
  T := (60*Mm + Ss)*100 + Cc
  {C'est Le temps exprimé en Cc }
  SI T < T1 :
   T3 := T2; Num3 := Num2; Nom3 := Nom2
   T2 := T1; Num2 := Num1; Nom2 := Nom1
   T1 := T; Num1 := Num; Nom1 := Nom
  SINON
   SI T < T2 :
    T3 := T2;Num3 := Num2; Nom3 := Nom2
    T2 := T;Num2 := Num1; Nom2 := Nom
   SINON
    SI T < T3:
     T3 := T;Num3 := Num;Nom3 := Nom
    FSI
   FSI
  FSI
 FINPOUR

 {Ecriture des trois premiers}
 Ss := T1 / 100
 Cc := T1 - 100*Ss
 Mm := Ss / 60
 Ss := Ss - 60*Mm
 ECRIRE(Num1, Nom1, Mm, Ss, Cc)

 Ss := T2 / 100
 Cc := T2 - 100*Ss
 Mm := Ss / 60
 Ss := Ss - 60*Mm
 ECRIRE(Num2, Nom2, Mm, Ss, Cc)

 Ss := T3 / 100
 Cc := T3 - 100*Ss
 Mm := Ss / 60
 Ss := Ss - 60*Mm
 ECRIRE(Num3, Nom3, Mm, Ss, Cc)
```

FIN

Programmation :

Se référer au programme P7.

* * * * *

Exercice 1 : Parties d'un ensemble

1. Algorithme

```
ALGORITHME Parties
TYPE T = STRUCTURE
 Tete  : Pointeur(Typemaillon)
 Nb  : ENTIER
FIN
TYPE Typemaillon = STRUCTURE
 Val : ENTIER
 Adr : Pointeur(Typemaillon)
FIN
VAR Liste1, Liste2 : Tableau[1..N] DE T
DEBUT
 LIRE(V)
 LIRE(N)
 ECRIRE({})
 POUR I=I, N
  Allouer(Typemaillon, Q)
  Affval(Q, V(I) )
  Affadr(Q, NIL)
  Liste1(I).Tete := Q
  Liste1(I).Nb := 1
  Imprimer(Q)
 FINPOUR
 N1 := N  ;   N2 := 0
 Aig := VRAI
 Long := 2 ; Compt := 1 + N
 TANTQUE  N # 1 :
  POUR I=1, N :
   POUR J=I+1, N :
    Union(Aig, I, J, L, Nb)
    SI NON Exist(Aig, L) ET Long = Nb:
     SI Aig :
      N2 := N2 + 1
      Liste2(N2).Tete := L
      Liste2(N2).Nb := Nb
     SINON
      N1 := N1 + 1
```

```
      Liste1(N1).Tete := L
      Liste1(N1).Nb := Nb
     FSI
    FSI
   FINPOUR
  FINPOUR
  SI Aig :
   N1 := 0   ;  N := N2
   POUR K:=1, N  : Imprimer(Liste2(K).Tete FINPOUR
  SINON
   N2 := 0   ;  N := N1
   POUR K:=1, N : Imprimer(Liste1(K).Tete FINPOUR
  FSI

  Aig := NON Aig
  Long := Long + 1
  Compt := Compt + 1
 FINTANTQUE
  FIN
```

Module Union :

```
 ACTION Union( Aig, I, J, L, Nb)
 VAR
 DEBUT
  SI Aig :
   L1 := Liste1(I).Tete
   Nb1 :=Liste1(I).Nb
   L2 := Liste1(J).Tete
  SINON
   L1 := Liste2(I).Tete
   Nb1 :=Liste2(I).Nb
   L2 := Liste2(J).Tete
  FSI
  L := L1
  Nb := Nb1
  TANTQUE L2 # NIL :
   SI NON Recherche(Valeur(L2), L1)
    {Ajout  au début de L1}
    Nb := Nb + 1
    Allouer(Typemaillon, Q)
    Affval(Q, Valeur(L2))
    Affadr(Q, L)
    L := Q
   FSI
   L2 := Suivant(L2)
  FINTANTQUE
   FIN
```

Module Exist :

```
 ACTION Exist( Aig, L)
```

```
VAR
DEBUT
 I := 1
 Trouv := FAUX
 SI Aig :
  TANTQUE I <= N2 ET NON Trouv :
   SI Aig :
    SI Egal(Liste2(I).Tete, Liste2(I).Nb, L, Nb) :
     Trouv := VRAI
    SINON I := I + 1 FSI
   FSI
  FINTANTQUE
 SINON
  TANTQUE I <= N1 ET NON Trouv :
   SI Aig :
    SI Egal(Liste1(I).Tete, Liste1(I).Nb, L, Nb) :
     Trouv := VRAI
    SINON I := I + 1 FSI
   FSI
  FINTANTQUE
 FSI
 Exist := Trouv
FIN
```

Module Egal :

```
ACTION Egal (l1, Nb1, l2, Nb2)
   VAR
DEBUT
 SI Nb1 = Nb2 :
  Trouv := FAUX
  TANTQUE L2 # NIL ET NON Trouv :
   SI NON Recherche( Valeur(L2), L1) :
    Trouv := VRAI
   SINON
    L2 := Suivant(L2)
   FSI
  FINTANTQUE
  Egal := NON Trouv
 SINON
  Egal := FAUX
 FSI
   FIN
```

Module Recherche :

```
ACTION Recherche( Val, L)
   VAR
DEBUT
 Trouv := FAUX
 TANTQUE L # NIL ET NON Trouv :
```

```
 SI Valeur(L) = Val :
  Trouv := VRAI
 SINON
  L := Suivant(L)
 FSI
FINTANQTUE
Recherche :=  Trouv
  FIN
```

Programmation

Se référer au programme P5.

Exercice 1 : Facture

Algorithme

```
        ALGORITHME Facture
VAR
 Total, Prix_unitaire : REEL
 Numproduit, Quantite : ENTIER
DEBUT
 Total := 0
 Lire(Numproduit, Quantite, Prix_unitaire)
 TANTQUE Numproduit # 0 :
  Total := Total + Quantité * Prix_unitaire
  Ecrire(Numproduit, Quantite, Prix_unitaire)
  Lire(Numproduit, Quantite, Prix_unitaire)
 FINTANTQUE
 Ecrire(Total)
FIN
```

Programmation
Se référer au programme P6.

Exercice 2 : Analyse d'un texte

Module Numérique

```
 ACTION Numérique ( Mot, l)
   VAR
DEBUT
 I := 1
 C := Prendre(Mot, 1)
 Chiffre := VRAI
 TANTQUE I <= L ET Chiffre
  SI C< '0'  OU C > '9':
   Chiffre := FAUX
  SINON
```

```
  I := I + 1
  SI I <= L
   C := Prendre(Mot, I)
  FSI
 FSI
FINTANTQUE
  FIN
```

Module Alphabétique

```
ACTION Aphabétique ( Mot, l)
VAR
DEBUT
 I := 1
 C := Prendre(Mot, 1)
 Aphabetique := VRAI
 TANTQUE I <= L ET Aphabetique :
  SI C <= Z'  ET C >= A' :
   Aplphabétique := FAUX
  SINON
   I := I + 1
   SI I <= L :
    C := Prendre(Mot, I)
   FSI
  FSI
 FINTANTQUE
  FIN
```

Algorithme :

```
ALGORITHME
VAR
DEBUT
 Continue := VRAI
 TANTQUE Continue :
  SI C = '/' :
   LIRE(C)
   SI C='*':
    FIN := FAUX
    TANTQUE NON FIN
     LIRE(C)
     TANTQUE C # '*' :
      LIRE(C)
     FINTANTQUE
     LIRE(C)
     SI C = '/' :
      ECRIRE(0) ; FIN := VRAI
     FSI
    FINTANTQUE
    LIRE(C)
   SINON
    TANTQUE C # ' ' : LIRE(C) FINTANTQUE
    ECRIRE(3)
```

```
 FSI
SINON
 SI C = ' ' : LIRE(C)
 SINON
  Mot := ''
  L := 0
  TANTQUE C # ' ' :
   Mot := Mot + C
   L :=  L + 1
   LIRE(C)
  FINTANTQUE
  SI Mot='Fintex' : Continue := FAUX
  SINON
   SI Numérique(Mot, L) :
    ECRIRE(2)
   SINON
    SI Alphabetique(Mot) :
     ECRIRE(1)
    SINON  ECRIRE(3) FSI
   FSI
  FSI
 FSI
FINTANTQUE
 FIN
```

Exercice 3 : Tri de 4 variables

```
ALGORITHME  Tri
CONST
 Grande_Valeur = 32 000
VAR  I : INTEGER
 T1, T2, T3, T4, V : ENTIER  { Données entières }
DEBUT
   T1, T2, T3, T4 := Grande_Valeur { Affectation globale }
 POUR I=1, 4 :
 LIRE(V)
 SI V < T1 :
  T4 :  T3 ;  T3 := T2
  T2 := T1 ;  T1 := V
 SINON
  SI V < T2 :
   T4 := T3 ;  T3 := T2
   T2 := V
  SINON
   SI V < T3 :
    T4 := T3
    T3 := V
   SINON
    SI V < T4 : T4 := V  FSI
  FSI
 FSI
```

```
 FINPOUR
 ECRIRE(T1, T2, T3, T4)
  FIN
```

* * * * *

Exercice 1 : Gestion des polynômes

Soit le type polynôme défini comme suit :

```
       TYPE Polynome = STRUCTURE
 Nbre  : ENTIER
 Tab  : TABLEAU(40..2) DE REEL
FIN
```

Module Point

```
 FONCTION Point( P, V) : Réel
 VAR P : Polynome
  V : REEL
  I : ENTIER
 DEBUT
  Point := 0
  POUR I=1, P.Nbre :
   Point := Point + (P.Tab(I, 1) * ( V ** P.Tab(I, 2) ))
  FINPOUR
 FIN
```

Module Dérivée

```
 ACTION  Dérivé( P, R)
 VAR P, R   : Polynome
  I, K, Coef  : ENTIER
 DEBUT
  K := 0
  POUR I=1, P.Nbre :
   Coef := P.Tab(I, 1) *  P.Tab(I, 2)
   SI Coef # 0 :
    K := K + 1
    R.Tab(K, 1) := Coef
    R.Tab(K, 2) := P.Tab(I, 2) - 1
   FSI
  FINPOUR
  R.Nbre := K
 FIN
```

Module Somme

```
 ACTION Somme(P, Q, R)
```

```
VAR  P, Q, R : Polynome
 I, J, K : ENTIER
DEBUT
 K:=0; I, J := 1
 TANTQUE I <= P.Nbre ET J <= Q.Nbre :
  K := K + 1
  SI P.Tab(I, 2) > Q.Tab(J, 2) :
   R.Tab(K, 1) := P.Tab(I, 1)
   R.Tab(K, 2) := P.Tab(I, 2)
   I := I + 1
  SINON
   SI P.Tab(I, 2) < Q.Tab(J, 2) :
    R.Tab(K, 1) := Q.Tab(J, 1)
    R.Tab(K, 2) := Q.Tab(J, 2)
    J := J + 1
   SINON
    Coef := P.Tab(I, 1) + Q.Tab(J, 1)
    SI Coef # 0 :
     R.Tab(K, 1):= Coef
     R.Tab(K, 2) := P.Tab(I, 2)
    SINON  K := K - 1
    FSI
    I := I + 1   ;  J := J + 1
   FSI
  FSI
 FINTANTQUE
 TANTQUE I <= P.Nbre :
  K := K + 1
  R.Tab(K, 1) := P.Tab(I, 1)
  R.Tab( K, 2) := P.Tab(I, 2)
  I := I + 1
 FINTANTQUE
 TANTQUE J <= Q.Nbre :
  K := K + 1
  R.Tab(K, 1) := Q.Tab(J, 1)
  R.Tab(K, 2) := Q.Tab(J, 2)
  J := J + 1
 FINTANTQUE
 R.Nbre := K
FIN
```

Module Produit

On utilise l'action Prod(P, M, O) qui effectue le produit du polynôme P par le monôme M et qui range le résultat dans le polynôme O. On utilise également l'action Affecter(P, O) qui affecte le polynôme P au polynôme O. Elles sont définies comme suit :

Module Prod :

```
ACTION Prod(P, M, O)
VAR  P, M, O  : Polynome
 I  : ENTIER
```

```
DEBUT
 O.Nbre := P.Nbre
 POUR I=1,P.Nbre :
  O.Tab(I, 1) := P.Tab(I, 1) * M.Tab(1, 1)
  O.Tab(I, 2) := P.Tab(I, 2) + M.Tab(1, 2)
 FINPOUR
FIN
```

Module Affecter

```
ACTION Affecter(P, O)
VAR  P, O  : Polynome
  I  : ENTIER DEBUT
 O.Nbre := P.Nbre
 POUR I=1,P.Nbre :
  O.Tab(I, 1) := P.Tab(I, 1)
  O.Tab(I, 2) := P.Tab(I, 2)
 FINPOUR
FIN
```

Pseudo-algorithme du produit :

```
1)  R = P * Premier terme de Q
 POUR I = 2, Q.Nbre :
2)  M := I-ième terme de Q
3)  R := R + P*M
 FINPOUR
```

Module Produit :

```
ACTION Produit(P, Q, R)
VAR  M, P, Q, R, T, U  : Polynome
 I : ENTIER
DEBUT
1) M.Nbre := 1
 M.Tab(1, 1) := Q.Tab(1, 1)
 M.Tab(1, 2) := Q.Tab(1, 2)
 Prod(P, M, R)
 POUR I=2, Q.Nbre :
2)  M.Tab(1, 1) := Q.Tab(I, 1)
  M.Tab(1, 2) := Q.Tab(I, 2)
3)  Prod(P, M, T)
  Somme(R, T, U)
  Affecter(U, R)
 FINPOUR
FIN
```

Procédure PASCAL qui imprime un polynôme

```
      PROCEDURE Imprimer ( P : Polynome )
 VAR
  I : INTEGER
```

```
BEGIN
 FOR I:=1 TO P.nbre DO
  WRITE(P.Tab(.I, 1.), 'x', P.Tab(.I, 2.)  )
END
```

Exercice 2 : Accès par fonction

Dans les trois algorithmes qui suivent V et V' sont définis comme suit :

V : TABLEAU(N, 2) de ENTIER
V' : TABLEAU(M, 2) de ENTIER

Dans l'algorithme d'insertion, on suppose que :
- 999 désigne le vide
t désigne le prochain emplacement libre dans V'

Dans l'algorithme de suppression, on peut procéder comme suit :
suppression logique : on ajoute un bit dans V'
suppression physique: on modifie les liens
Dans les deux cas, les emplacements libérés ne sont pas récupérés. Par conséquent une mise à jour ultérieure est à prévoir.

On optera pour une suppression physique.

On utilisera le module Mettredans défini comme suit :

```
ACTION Mettredans (E, B)
VAR E : ENTIER
 B : BOOLEEN
DEBUT
 SI T <= M :
  SI B : V(K,2) := T
  SINON V'(K,2) := T FSI
  V'(T, 1) := E
  V'(T, 2) := 0
  T := T + 1
 SINON ECRIRE('TABLEAU V' saturé') FSI
FIN
```

Module de recherche

```
ALGORITHME Recherche
VAR  E, I, J  : ENTIER
 Trouv  : BOOLEEN
DEBUT
 LIRE(E) { élément à rechercher }
 I := F(E)
 SI V(I, 1) = E :
   ECRIRE ( 'L'élément exite déjà )
 SINON
  J := V(I, 2)
  Trouv := FAUX
```

```
  TANTQUE J # 0 ET NON Trouv :
   SI V'(J, 1) = E : Trouv := VRAI
   SINON J := V'(J, 2) FSI
  FINTANTQUE
  SI Trouv : ECRIRE ( 'L'élément exite déjà )
  SINON ECRIRE ( 'L'élément n''exite pas' ) FSI
 FSI
FIN
```

Module d'insertion

```
ALGORITHME Inserer
VAR  E, I, J  : ENTIER
 Trouv  : BOOLEEN
DEBUT
 LIRE(E) { élément à insérer }
 I := F(E)
 SI V(I, 1) = - 999 :
  V(I, 1) := E  ; V(I, 2) := 0
 SINON
  SI V(I, 1) = E :
    ECRIRE ( 'L'élément exite déjà' )
  SINON
   J := V(I, 2)
   SI J = 0 : Mettredans(E, VRAI)
   SINON
    Trouv := FAUX
    TANTQUE J # 0 ET NON Trouv :
     SI V'(J, 1) = E : Trouv := VRAI
     SINON J := V'(J, 2) FSI
    FINTANTQUE
    SI Trouv : ECRIRE ( 'L'élément exite déjà' )
    SINON Mettredans(E,FAUX) FSI
   FSI
  FSI
 FSI
FIN
```

Module de suppression

```
ALGORITHME Supprimer
VAR  E, I, J, K  : ENTIER
 Trouv  : BOOLEEN
DEBUT
 LIRE(E) { élément à supprimer }
 I := F(E)
 SI V(I, 1) = - 999 :
  ECRIRE ( 'L'élément n''exite pas' )
 SINON
  SI V(I, 1) = E :
    SI V(I, 2) = 0 : V(I, 1) := - 999
   SINON
```

```
   V(I, 1) := V'(V(I, 2), 1)
   V(I, 2) := V'(V(I, 2),2)
  FSI
 SINON
  J := V(I, 2)
  K := I
  B := VRAI
  Trouv := FAUX
  TANTQUE J # 0 ET NON Trouv :
   SI V'(J, 1) # E :
    K := J ; B := FAUX
    J := V'(J, 2)
   SINON
    Trouv := VRAI
    {Modifier Les Liens}
    SI B : V(K, 2) := V'(J, 2)
    SINON  V'(K, 2):= V'(J, 2) FSI
   FSI
  FINTANTQUE
  SI  NON Trouv :
   ECRIRE ( 'L'élément n''exite pas' )
  FSI
 FSI
FSI
FIN
```

Exercice 3 : Tri de couleurs

```
ALGORITHME Tri_couleur
CONST
 N = 50
VAR
 Limit,I, Pos : ENTIER
 Temp : CAR
 T : TABLEAU[1..N] DE CAR
DEBUT
 LIRE(T)
 Pos := 1;
 POUR I:=1, N
  SI T(.I.) = 'R'
   SI Pos <> I :
    Temp := T(.Pos.);
    T(.Pos.) := T(.I.);
    T(.I.) := Temp;
   FSI
   Pos := Pos + 1
  FSI
 FINPOUR

 Limit := Pos;
 Pos := N;
 POUR   I:=N, Limit , -1
```

```
 SI T(.I.) = 'V'
  SI Pos <> I :
   Temp := T(.Pos.);
   T(.Pos.) := T(.I.);
    T(.I.) := Temp;
  FSI
  Pos := Pos - 1;
 FSI
FINPOUR
ECRIRE(T)
END
```

* * * * *

Exercice 1 : Gestion des polynômes

Soit le type Polynome défini comme suit :

```
        TYPE T = STRUCTURE
 Coef  : ENTIER
 Exp  : ENTIER
FIN
TYPE Typemaillon = STRUCTURE
 Val : T
 Adr : Pointeur(Typemaillon)
FIN
TYPE Polynôme = Pointeur(Typemaillon)
```

Module entrée :

Nous supposons que :
- les polynômes sont bien écrits (pas d'erreurs)
- il n'y a aucun blanc entre la variable X et l'exposant
- les coefficients et les exposants sont des constantes entières.

c'est à dire la syntaxe suivante :

[blancs] [+!-] [blancs] [chch...] [blancs] [[x] [ch.. ch]]

[blancs] désigne 0, 1 ou plusieurs blancs.
Les crochets désignent les parties facultatives.
La partie [ch..ch] après X n'est présente que si ce dernier est présent.

V étant un vecteur de type T.

```
LIRE(C)
P := NIL
TANTQUE C # '*' :
```

```
Coef:= '' ; Exp := ''
Signe := '+'

{ Sauter les blancs}
        TANTQUE C = ' ' : LIRE(C) FINTANTQUE

SI C='+' OU C ='-' : Signe := C ; LIRE(C) FSI

{ Sauter les blancs}
        TANTQUE C = ' ' : LIRE(C) FINTANTQUE

TANTQUE Chiffre(C) :
 Coef := Coef + C
 LIRE(C)
FINTANTQUE

{ Sauter les blancs}
        TANTQUE C = ' ' : LIRE(C) FINTANTQUE

SI C ='X' :
 LIRE(C)
 TANTQUE Chiffre(C) :
  Exp := Exp + C
  LIRE(C)
 FINTANTQUE
FSI
V.Coef := Nombre(Coef)
V.Exp := Nombre(Exp)
Allouer(Typemaillon, Q)
Affval(Q, V) ; Affadr(Q, NIL)
SI P # NIL : Affadr(P, Q)
SINON L := Q   FSI
P := Q
FINTANTQUE
```

Module Point :

```
ACTION Point( P, V, Res) : Réel
VAR P : Polynome
 V : REEL
 I : ENTIER
DEBUT
 Res := 0
 L := P
 TANTQUE L # NIL :
  Res := Res + (Valeur(L).Coef * ( V ** Valeur(L).Exp ))
  L := Suivant(L)
 FINTANTQUE
FIN
```

Module Dérivée :

```
ACTION Dérivé( P, Q)
VAR P, R : Polynome
DEBUT
 R := NIL
 TANTQUE P # NIL  :
  V.Coef := Valeur(P).Coef * ( Valeur(P).Exp - 1 )
  V.Exp :=  Valeur(P).Exp  - 1
  SI Coef # 0 :
   Allouer(Typemaillon, W)
   Affval(W,V)
   SI R#NIL : Affadr(R,W)
   SINON  Q := R  FSI
   R := W
  FSI
 FINTANTQUE
FIN
```

Module Somme

```
ACTION Somme(P, Q, R)
VAR  P, Q, R : Polynome
DEBUT
 L1 := P; L2 := Q ; R := NIL
 TANTQUE L1 # NIL ET L2 # NIL:
  Allouer(Typemaillon, W)
  Affadr(W, NIL)
  SI Valeur(L1).Exp < Valeur(L2).Exp :
   Affval(W, Valeur(L1) )
   L1:= Suivant(L1)
  SINON
   SI Valeur(L1).Exp > Valeur(L2).Exp :
    Affval(W, Valeur(L2) )
    L2:= Suivant(L2)
   SINON
    V.Coef := Valeur(L1).Coef + Valeur(L2).Coef
    V.Exp := Valeur(L1).Exp
    Affval(W, V)
    L1:= Suivant(L1)
    L2:= Suivant(L2)
   FSI
  FSI
  SI R#NIL : Affadr(R,W)
  SINON L3 := R  FSI
  R := W
 FINTANTQUE
 TANTQUE L1 # NIL :
  Allouer(Typemaillon, W)
  Aff-Val(W, Valeur(L1))
  SI R# NIL : Affadr(R, W)
  SINON  L3 := W FSI
  L1 := Suivant(L1)
 FINTANTQUE
```

```
TANTQUE L2 # NIL :
 Allouer(Typemaillon, W)
 Aff-Val(W, Valeur(L2))
 SI R# NIL : Affadr(R, W)
 SINON L3 := W FSI
 L2 := Suivant(L2)
FINTANTQUE
```

Module Produit

On utilise l'action suivante Prod(P, M, O) qui effectue le produit du polynôme P par le monôme M et qui range le résultat dans le polynôme O. Elle est définie comme suit :

```
 ACTION Prod(P, M, O)
VAR  P, O : Polynome
 M : T
DEBUT
 R := NIL
 L := P
 TANTQUE L # NIL :
  V.Coef := Valeur(L).Coef * M.Coef
  V.Exp :=  Valeur(L).Exp  + M.Exp
  Allouer(Typemaillon, W)
  Aff-Val(W, V))
  SI R# NIL : Affadr(R, W)
  SINON  O := W FSI
  R := W
  L := Suivant(L)
 FINTANTQUE
FIN
```

Pseudo-algorithme du produit :

```
1)  R = P * Premier terme de Q
 POUR I = 2, Q.Nbre :
2)  M := I-ième terme de Q
3)  R := R + P*M
 FINPOUR
```

Module Produit :

```
ACTION Produit(P, Q, R)
VAR  P, Q, R, T, U  : Polynôme
 I    : ENTIER
DEBUT
1) L:= Q
 M.Coef := Valeur(L).Coef
 M.Exp  := Valeur(L).Exp
 Prod(P, M, R)
 L := Suivant(L)
```

```
 TANTQUE L # NIL :
2)  M.Coef := Valeur(L).Coef
  M.Exp := Valeur(L).Exp
3)  Prod(P, M, T)
  Somme(R, T, U)
  R := U
  L := Suivant(L)
 FINTANTQUE
FIN
```

Exercice 2 : Accès par fonction

```
ALGORITHME Hcode
TYPE T = STRUCTURE
 Val  : Typeqq
 Vide  : BOOLEEN
 Lien  : Pointeur ( Typmaillon)
FIN
VAR V  : TABLEAU(1..M)  DE T
DEBUT
 POUR I=1, N
  LIRE(Val)
  H := F(I)
  SI V(H).Vide :
   V(H).Val := Val
  SINON
   SI V(H).Val = V :
    ECRIRE('La Donnée Existe')
   SINON
    {Parcours DE La Liste}
    P := V(H).Lien ; Trouv := FAUX
    TANTQUE P # NIL ET NON Trouv :
     SI Valeur(P) = V :
      Trouv := VRAI
     SINON
      Q := P
      P := Suivant(P)
     FSI
    FINTANTQUE
    SI NON Trouv :
     Allouer(R, Typmaillon)
     Affval(R, Val); Affadr(R, NIL)
     Affadr(Q, R)
    SINON
     ECRIRE('La donnée Existe')
    FSI
   FSI
  FSI
 FINPOUR
FIN
```

* * * * *

Partie A

A.1 Nombre de mots commençant par 'm' et se terminant par's' :

```
ALGORITHME Ms
VAR Compt  : ENTIER
 C, Sauv  : CAR
DEBUT
 Compt := 0 ; LIRE(C)
 TANTQUE C # '.' :
  TANTQUE C = ' ' : LIRE(C) FINTANTQUE
  SI C = 'M' :
   TANTQUE C # ' ' ET C # '.' :
    Sauv := C
    LIRE(C)
   FINTANTQUE
   SI Sauv = 'S' : Compt := Compt + 1 FSI
  SINON
   TANTQUE C # ' ' ET C # '.' :
    LIRE(C)
   FINTANTQUE
  FSI
 FINTANTQUE
 ECRIRE(Compt)
FIN
```

A.2 : Longueur, nombre de voyelles et dernier caractère de chaque mot

```
ALGORITHME Divers
VAR Longueur, Nbre  : ENTIER
 Mot,Dernier , C : CAR
DEBUT
 LIRE(C)
 TANTQUE C # '.' :
  TANTQUE C = ' ' : LIRE(C) FINTANTQUE
  Dernier := ' '
  Mot:='' ; Longueur := 0 ; Nbre := 0
  TANTQUE C # ' ' ET C # '.' :
   Mot := Mot + C
   Longueur := Longueur + 1
   SI Voyelle(C) : Nbre := Nbre + 1 FSI
   Dernier := C
   LIRE(C)
  FINTANTQUE
  SI Dernier # ' ' { Le mot existe}
   ECRIRE(Mot, Longueur, Nbre, Dernier)
  FSI
 FINTANTQUE
FIN
```

Le prédicat Voyelle(C) est vrai si C est une voyelle, faux sinon.

Partie B

B.1 Recherche du premier nombre dont le carré est égal à la somme des 3 précédents

```
ALGORITHME Carré
VAR  A, B, C, D, I, N  : ENTIER
 Trouv   : BOOLEEN
DEBUT
 LIRE(N) ;  LIRE(A, B, C)
 Trouv := FAUX  ;  I := 3
 TANTQUE I < N ET NON Trouv :
  LIRE(D)
  I := I + 1
  SI D2 = A + B + C
   Trouv := VRAI
  SINON
   A := B ; B := C ;  C := D
  FSI
 FINTANTQUE
 SI Trouv : ECRIRE(D) SINON ECRIRE( ' Inexistant' ) FSI
FIN
```

B.2 Calcul de la somme (-1)i . xi / i

```
ALGORITHME Suite
VAR  I, Signe  : ENTIER
 X, S   : REEL
DEBUT
 LIRE(N) ;  LIRE(X)
 Signe := 1
 S := 0 ; I := 0
 POUR I = 1, N
  S := S + ( X**I / I) * Signe
  Signe := - Signe
 FINPOUR
 ECRIRE(S)
FIN
```

* * * * *

Exercice 1 : Analyse des constantes virgule flottante de la forme

Dans l'algorithme qui suit, Mot désigne une variable chaîne de caractères, W et D des variables entières. W désigne le nombre total de caractères de la constante et D le nombre de chiffres après le point décimal.

On utilise aussi les modules Erreur et Chiffre définis comme suit :

chiffre (c) = vrai si c est chiffre faux sinon

Erreur :

```
TANTQUE  C <>' ' ET C <>'$' :
 Mot := Mot + C ;  LIRE(C)
FINTANTQUE
```

C et Mot sont des variables globales pour ce module.

```
LIRE(C)
TANTQUE C <>' $' :
 Mot := ''  ;  W := 0;  D := 0
 TANTQUE  C= ' '   :  LIRE(C)   FINTANTQUE
 SI C= '+' OU C= '-' :
  W := 1 ; Mot := Mot + C;  LIRE(C)
 FSI

 TANTQUE Chiffre(C) :
  W := W + 1;  Mot := Mot + C ; LIRE(C)
 FINTANTQUE

 SI C = '.' :
  W := W + 1 ; Mot := Mot + C ; LIRE(C)
  TANTQUE Chiffre(C) :
   W := W + 1 ; D := D + 1; Mot := Mot + C;
   LIRE(C)
  FINTANTQUE
 FSI

 SI C = 'E' :
  W := W + 1 ; Mot := Mot + C ; LIRE(C)

  SI C= '+' OU C= '-' :
   W := W + 1 ; Mot := Mot + C;  LIRE(C)
  FSI

  SI Chiffre(C) :
   W := W + 1  ; Mot := Mot + C;  LIRE(C)
   SI Chiffre(C) :
    W := W + 1  ; Mot := Mot + C;  LIRE(C)
    SI C = ' ' OU C = '$' :
     ECRIRE ( Mot , W, D)
    SINON Erreur FSI
   SINON Erreur  FSI
  SINON Erreur FSI
 SINON
  SI C <> '$' : Erreur FSI
FIN
```

Exercice 2 : Classement

```
ALGORITHME Moyenne
VAR  Note : TABLEAU (1..30, 1..9) DE REEL
 Coef : TABLEAU(1 ..9) DE ENTIER
 I, J, Somcoef : ENTIER
 T, M : TABLEAU[1..30] DE REEL
 S : REEL
DEBUT
 LIRE(Coef)
 LIRE(Note)
 Somcoef := 0
 POUR I=1, 9
  Somcoef := Somcoef + Coef(I)
 FINPOUR

 POUR I=1, 30
  S := 0
  POUR J=1, 9 :
   S := S + Note(I, J) * Coef(J)
  FINPOUR
  T(I) := S
  M(I) := S / Somcoef
 FINPOUR
 Trier(M)
 ECRIRE(M)
   FIN
```

* * * * *

Exercice 1 : Couples parfaits

```
ALGORITHME Couples_parfaits
VAR I, J : ENTIER
DEBUT
 POUR I = 1, 1000
  POUR J = 1, 1000 :
   SI Parfait(I + J)
    ECRIRE(I, J)
   FSI
  FINPOUR
 FINPOUR
FIN
```

Module parfait :

```
PREDICAT  Parfait( N )
VAR
 N, S, Quot, I : ENTIER
DEBUT
 S := 1
 POUR I =2,  (N DIV 2) :
```

```
 Quot := N DIV I
 SI N = Quot * I :
  S := S + I
 FSI
FINPOUR
Parfait := ( S = N)
FIN
```

Exercice 2 : Mots de la forme X....Y.....Z

```
ALGORITHME  Xyz
VAR Sauv, C, Mot : CAR
DEBUT
 LIRE(C)
 TANTQUE C # '.' :
  TANTQUE C = ' ' : LIRE(C) FINTANTQUE
  SI C = 'X' :
   Mot := '' ; Trouv := FAUX
   TANTQUE C # ' ' ET C # '.' :
    Mot := Mot + C
    SI C = 'Y' : Trouv := VRAI FSI
    Sauv := C
    LIRE(C)
   FINTANTQUE
   SI Sauv = 'Z' ET Trouv :
    ECRIRE(Mot)
   FSI
  SINON
   TANTQUE C # ' ' ET C # '.' :
    LIRE(C)
   FINTANTQUE
  FSI
 FINTANTQUE
FINTANTQUE
FIN
```

* * * * *

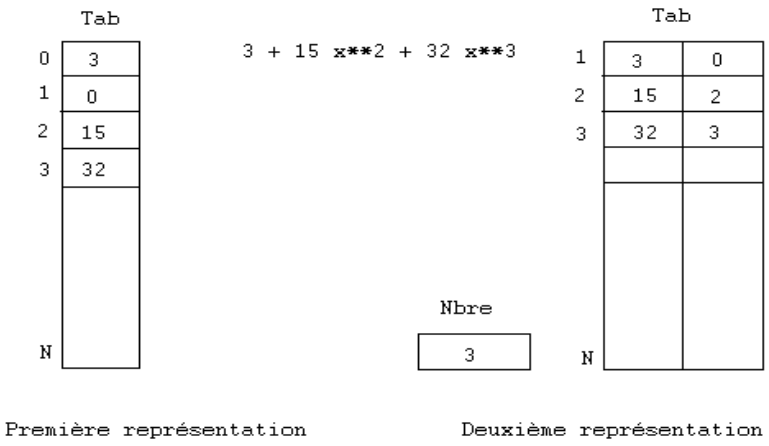
Exercice 1 : Tri par insertion

```
ALGORITHME TRI
VAR I, K, L, A : ENTIER
 T : TABLEAU[1..N]  DE ENTIER
DEBUT
 POUR I = 2, N
  A := T(I)
  K := 1 ; Trouv := FAUX
  TANTQUE K <  I ET NON Trouv :
   SI A <= T(K) :
    Trouv := VRAI
```

```
 SINON
  K := K + 1
 FSI
FINTANTQUE
SI Trouv :
 POUR L := I, K+1, -1 :
  T(L) := T(L-1)
 FINPOUR
 T(K) := A
FSI
FINPOUR
FIN
```

Exercice 2 : Représentation des polynômes

(i)Représentations :
(ii)



Première représentation :

Le degré est l'indice et l'élément représente le coefficient.

```
TYPE Polynome = TABLEAU [0..N] DE ENTIER
```

Deuxième représentation :

```
      TYPE Polynome = STRUCTURE
 Nbre  : ENTIER
 Tab  : TABLEAU(40..2) DE REEL
FIN
```

(ii) Ecriture des modules :

Première représentation :

Module Point :

```
FONCTION Point( T, V) : Réel
VAR T : Polynome
```

```
 V : REEL
 I : ENTIER
DEBUT
 Point := 0
 POUR I= 0, N :
  Point := Point + I * ( X ** T(I))
 FINPOUR
FIN
```

Module Dérivée

```
 ACTION Dérivé( P, R)
 VAR P, R   : Polynome
  I, K, Coef  : ENTIER
 DEBUT
  POUR I= 0, N-1 :
   R(I) := P(I+1)  * (I+1)
  FINPOUR
 FIN
```

Deuxième représentation :

Module Point :

```
 FONCTION Point( P, V) : Réel
 VAR P : Polynome
  V : REEL
  I : ENTIER
 DEBUT
  Point := 0
  POUR I=1, P.Nbre :
   Point := Point + (P.Tab(I, 1) * ( V ** P.Tab(I, 2) ))
  FINPOUR
 FIN
```

Module dérivée

```
 ACTION Dérivé( P, V)
 VAR P, R   : Polynome
  I, K, Coef  : ENTIER
 DEBUT
  K := 0
  POUR I=1, P.Nbre :
   Coef := P.Tab(I, 1) *  P.Tab(I, 2)
   SI Coef # 0 :
    K := K + 1
    R.Tab(K, 1) := Coef
    R.Tab(K, 2) := P.Tab(I, 2) - 1
   FSI
  FINPOUR
  R.Nbre := K
```

```
FIN
```

(iii) Calcul de p(x) en utilisant la factorisation de Horner

Deuxième représentation :

```
        S := T(1)*X + T(2)
POUR I =2, Nbr -1 :
 S := (S * X) + T(I+1)
FINPOUR
```

* * * * *

Exercice 1 : Interclassement de deux vecteurs ordonnés

```
   ALGORITHME Interclasser
VAR  K, J1, J2, N1, N2 : ENTIER
 V1 : TABLEAU[1..N1] DE ENTIER
 V2 : TABLEAU[1..N2] DE ENTIER
 V3 : TABLEAU[1..N1+N2] DE ENTIER
DEBUT
 LIRE(N1, N2)
 K := 0
 J1, J2 := 1
 TANTQUE J1 <= N1 ET J2 <= N2 :
  K := K + 1
  SI V1(J1) < V2(J2) :
   V3(K) := V1(J1)
   J1 := J1 + 1
  SINON
   V3(K) := V2(J2)
   J2 := J2 + 1
  FSI
 FINTANTQUE
 TANTQUE J1 <= N1 :
  K := K + 1
  V3(K) := V1(J1)
  J1 := J1 + 1
 FINTANTQUE
 TANTQUE J2 <= N2 :
  K := K + 1
  V3(K) := V2(J2)
  J2 := J2 + 1
 FINTANTQUE
   FIN
```

Exercice 2 : Permutation de 2 sous-vecteurs

L'algorithme qui suit permute les sous-vecteurs T(1..P) et T(P+1..N) pour P compris entre 1 et N

```
  ALGORITHME Permuter
VAR
 P, N, J, I, N : ENTIER
 T : TABLEAU[1..N] DE ENTIER
 C : ENTIER
DEBUT
 LIRE(P)
 SI P > N DIV 2 :
  POUR J := P + 1, N
   C := T(N)
   POUR I=N-1, 1, -1
    T(I+1) := T(I)
   FINPOUR
   T(1) := C
  FINPOUR
 SINON
  POUR J := 1, P
   C := T(1)
   POUR I= 1, N-1
    T(I) := T(I+1)
   FINPOUR
   T(N) := C
  FINPOUR
 FSI
FIN
```

Exercice 3: Insertion par fonction

Recherche/insertion d'un élément

```
       ALGORITHME Inserer
VAR
DEBUT
 LIRE(K) ;   N := 0
 POUR L = 1, K
  LIRE(Donnee) ;  I := H(Donnee)
  Trouv, Vide := FAUX
  TANTQUE NON Trouv ET NON Vide :
   SI Occ(I) :
    SI Val(I) = E
     Trouv := VRAI
    SINON
     I := I+1;SI I>M : I := I-M FSI
    FSI
   SINON
    Vide := VRAI
   FSI
  FINTANTQUE
  SI NON Trouv :
   SI N = M-1 :
    " Débordement "
```

```
  SINON
   N := N + 1
   Aff-Val(I, E) ; Aff-Occ(I, VRAI)
  FSI
 SINON
  " Existe "
 FSI
FINPOUR
FIN
```

* * * * *

Exercice 1 : X ?, Y ?

Au niveau de l'algorithme, il manque la définition de la variable N. L'algorithme X calcule la somme S = 1 + 7 + 21 + 35.... en prenant exactement (N div 2) + 1 termes.

Pour N=1 , S est 1
Pour N=2 ou N=3 , S est 1 + 7
Pour N=4 ou N=5 , S est 1 + 7 + 21
Pour N=6 ou N=7 , S est 1 + 7 + 21 + 35
etc..

L'algorithme Y délivre la valeur 1, puis les N premiers multiples de i.

Pour n=6 et i =4, y délivre 1, 4, 8, 12, 16, 20 et 24.
Pour n=3 et i=1, y délivre 1, 1, 2, 3.

Exercice 2 : Calcul de la somme 1/2 - 1/3 + 1/4 - .....

Une simple analyse nous permet de trouver la formule de récurrence suivante :

S0 = 0
Sn := Sn-1 + (-1)n+1 / (n+1)

Donc S1 = 1/2 ; S2 = S1 -1/3 ; S3 = S2 + 1/4 ; .....
On calcule successivement les termes S1, S2, ...
On s'arrête quand la différence (en valeur absolue) de deux termes consécutifs devient inférieure à la précision.

```
ALGORITHME Serie
VAR  Precision, A, B  : REEL
 N : ENTIER
DEBUT
 LIRE ( Precision )
 A  := 0
 B := 1/2
 N  := 1
 TANTQUE  Valeur Absolue ( A-B) >= Precision :
  N := N + 1
  A := B
```

```
  B :=   A + (-1)**(N+1) / (N+1)
 FINTANTQUE
 ECRIRE (B)
FIN
```

Exercice 3 : Nombre parfait

```
ALGORITHME Parfait
VAR  N, I, S, Q : ENTIER
DEBUT
 LIRE(N) { On Suppose N > 1 }
 S  := 1
 I  := 2
 TANTQUE I <= N/2 :
  Q := N / I  % Division entière %
  SI N = Q*I : S := S + I FSI
  I := I + 1
 FINTANTQUE
 SI S = N :
  ECRIRE ( N , 'Est Parfait')
 SINON
  ECRIRE(N,  n''est pas parfait' )
 FINSI
FIN
```

Tous les nombres parfaits de 1 à 1000 :

```
ALGORITHME Parfait1000
VAR J, S, I, Q : ENTIER
DEBUT
 J := 2
 TANTQUE J <= 1000 :
  {Corps de l'algorithme précédent }
  S  := 1
  I  := 2
  TANTQUE I <= J/2 :
   Q := J / I { Division Entière }
   SI J = Q*I : S := S + I FSI
   I := I + 1
  FINTANTQUE
  SI S =J :
   ECRIRE(J)
  FSI
  J := J + 1
 FINTANTQUE
FIN
```

4. Recherche de couples avec conditions

```
ALGORITHME Couple
VAR A, B, N, M, Q1, Q2  : ENTIER
```

```
DEBUT
 { On Suppose M >= N }
 LIRE (N, M)
 A := N
 TANTQUE A <= M :
  B := N
  TANTQUE B <= M :
   Q1 := (A**2 + B**2) / 5 { Division entière }
   Q2 := (A**2 + B**2) / 2 { Division entière }
   SI  Q1 * 5 =  (A**2 + B**2) OU
    Q2 * 2 <> (A**2 + B**2) :
    ECRIRE( A, B)
   FSI
   B := B + 1
  FINTANTQUE
  A := A + 1
 FINTANTQUE
FIN
```

* * * * *

Exercice 1 : Impression d'un V en PASCAL

```
PROCEDURE Imprimerv ( C : CHAR; X : INTEGER );
 VAR I : INTEGER;
 BEGIN
  WHILE ( X > 1) DO
   BEGIN
    WRITE(C);
    FOR I := 1 TO X DO WRITE(' ');
    WRITELN(C);
    X := X - 2
   END
 END
```

Exercice 2 : Mots de la forme ....XY....., mots de la forme .....X.....Y.....

a) Mots de la forme ....XY.....

```
ALGORITHME Aaaaxyaaa
VAR N,M  : ENTIER
 C, Mot : CAR
LIRE(C)
TANTQUE C <> '#' :
 { Parcours Des Blancs }
 TANTQUE C= ' ' : LIRE(C) FINTANTQUE
 N := 0
 Mot := ''
 Trouv :=  FAUX
```

```
TANTQUE NON Trouv ET C <>' ' ET C<>'#' :
 TANTQUE C <>'X' ET C <>' ' ET C <> '#' :
  N := N + 1
  Mot := Mot !! C
  LIRE(C)
 FINTANTQUE
 SI C = 'X':
  Mot := Mot + C
  LIRE(C)
  SI N > 0 :
   Trouv:=(C='Y')
  SINON N:= N + 1 FSI
 FSI
FINTANTQUE

SI Trouv :
 M := -1
 TANTQUE C <>' ' ET C <>'#' :
  Mot := Mot !! C
  M := M + 1
  LIRE(C)
 FINTANTQUE
 SI M > 0 : ECRIRE( Mot ) FSI
FSI
FINTANTQUE
FIN
```

b) Mots de la forme .....X.....Y.....

```
ALGORITHME Aaaaaxaaayaaaa
VAR  N, M, P : ENTIER
 C, Mot : CAR
LIRE(C)
TANTQUE C <> '#' :
 { Parcours des blancs }
 TANTQUE C = ' ' : LIRE(C) FINTANTQUE

 { Recherche de  'X' dans le mot }
 Mot := ''
 N := 0
 TANTQUE C <> ' ' ET C <> '#' ET C <> ' X' :
  N := N + 1
  Mot := Mot !! C
  LIRE(C)
 FINTANTQUE

 SI C = 'X' : { SI' Y' suit 'X'  , ne pas le considérer}
  Mot := Mot !! C
  M := 0
  LIRE(C)
  SI C='Y' :
   M := 1
```

```
  Mot := Mot !! C
  LIRE(C)
 FSI

 { Rechercher le 'Y' }
 TANTQUE C <> ' ' ET C <>'#' ET C <> 'Y' :
  Mot := Mot !! C
  M := M + 1
  LIRE(C)
 FINTANTQUE

 SI C='Y' { Rechercher la fin du mot }
  P := - 1
  TANTQUE C <>' ' ET C <> '#' :
   Mot := Mot !! C
   P := P + 1
   LIRE(C)
  FINTANTQUE
  SI N * M * P <> 0 : ECRIRE( Mot) FSI
 FSI
FSI
FINTANTQUE
FIN
```

Exercice 3 : Variétés

1) Modules nécessaires :

Predicat Carre(a, b , c) :
en entrée : 3 entiers a, b et c.
en sortie : vrai si a2 = b + c, faux sinon.

Action Placer(a, b, c, d) :
en entrée : 4 entier a, b, c et d
en sortie : les 3 plus grands parmi les 4 rangés dans b, c et d tels que b < c < d

Prédicat Estilfact(a) :
en entrée : un entier a
en sortie : vrai s'il existe un entier x tel que x!=a, faux sinon

2) Corps de L

```
ALGORITHME Principal
VAR  I, N1, N2, N3, Dernier : ENTIER
 Exist     : BOOLEEN

LIRE(N) { Le Nombre D'éléments }
LIRE(N1, N2, N3)

{Ordonner N1, N2, N3 dans Max1, Max2, Max3 }
Max1 := N1
SI N2 > Max1 : Max2 := N2
```

```
SINON Max2 := Max1    Max1 := N2  FSI

SI N3 > Max2 : Max3 := N3 :
SINON
 SI N3 > Max1 : Max3:= Max2  ; Max2:= N3
 SINON Max3 := Max2    Max2:= Max1   Max1 := N3  FSI
FSI
SI Estilfact(N1) : ECRIRE(N1) FSI
SI Estilfact(N2) : ECRIRE(N2) FSI
SI Estilfact(N3) : ECRIRE(N3) FSI

Exist := FAUX
SI Carré(N3, N1, N2):Exist:= VRAI Dernier := N3  FSI
N1 := N2
N2 := N3

POUR I=1, N :
 LIRE(Nbr)
 SI Carré (Nbr, N1, N2) :
  Exist := VRAI
  Dernier := Nbr
 FSI
 Placer( Nbr, Max1, Max2, Max3)
 SI Estilfact(Nbr) : ECRIRE(Nbr) FSI
 N1 := N2
 N2 := Nbr
FINPOUR
ECRIRE(N1, N2, N3)
SI Exist ECRIRE( Dernier) FSI
FIN
```

3) Ecriture des modules

a)

```
        Predicat Carré( A, B, C)
VAR  A, B, C : ENTIER
Carré := ( A2 = B + C )
FIN
```

b)

```
        Action Placer ( A, Max1, Max2, Max3)
VAR  A, Max1, Max2, Max3 : ENTIER
SI A > Max3
 Max1 := Max2  Max2 := Max3
 Max3 := A
SINON
 SI A > Max2
  Max1 := Max2
  Max2 := A
 SINON
```

```
  SI A > Max1
   Max1 := A
  FSI
 FSI
FSI
FIN
```

c)

```
      Prédicat Estilfact ( N )
VAR F, I : ENTIER
F := 1
I := 1
TANTQUE F < N :
 F := F * I
 I := I + 1
FINTANTQUE
Estilfact := ( F=N)
FIN
```

* * * * *

Exercice 1 : Visibilité des objets

Portée des objets :

A (P) = { P, P2 }

B(P) = { P, P1, P2 }

C(P) = { P, P1, P2 }

D(P) = { P, P1, P2, P3 }

I, J, A (P1) = { P1 }

A, B, C = { P3 }

Trace

**B = 0
Apres P2, A = 0
***C = 2
Apres P3, D = 2
*I = 0
Apres P1, D = 0

Exercice 2 : Opérations sur chaîne de caractères

Module Longueur

```
FONCTION Longueur ( Mot)
VAR
 I, N : ENTIER
 Mot : TABLEAU[1..N] DE ENTIER
DEBUT
 I := 0
 TANTQUE I < N ET T(I) # ' ' :
  I := I + 1
 FINTANTQUE
 SI T(N) # ' ' : I := I + 1 FSI
 Longueur := I
FIN
```

Module Extract

```
FONCTION  Extract (Mot, I, J)
VAR
 I, N : ENTIER
 Mot : TABLEAU[1..N] DE ENTIER
BEGIN
 SI I >= 1 et I <= N et I + J - 1 <= N
  L := 0
  POUR K := I, I+J - 1 :
   L := L + 1
   Ex(L) := Mot(K)
  FINPOUR
  Extract := Ex { Affectation globale }
 SINON
  Extract := '' { Impossible }
 FSI
END
```

Module Index

```
FONCTION Index( Mot2, Mot1)
VAR
 I, N : ENTIER
 Ok : BOOLEAN
 Mot1, Mot2 : TABLEAU[1..N] DE ENTIER
BEGIN
 I := 1
 Ok := FAUX
 TANTQUE I <= N ET NON Ok :
  SI Mot1(1) # Mot2(I) :
   I := I + 1
  SINON
   SI Mot1 = Extract(Mot2, I, Longueur(Mot1) )
    Index := I
```

```
   Ok := VRAI
  SINON
   I := I + 1
  FSI
 FSI
FINTANTQUE
SI NON Ok : Index := 0 FSI
FIN
```

* * * * *

A : Exercice 1 : Mise à jours sur vecteurs

Structure de données

Un élément du vecteur est défini comme suit :

```
      TYPE Type_element = STRUCTURE
 Val : Typeqq
 Sup : BOOLEEN
FIN
VAR V : TABLEAU[1..M] DE Type_element
```

Typeqq désigne un type quelconque.

Initialisation

Vecteur :

```
      POUR I = 1, M DIV 2 :
 LIRE( V(I).Val )
 V(I).Sup := FAUX
FINPOUR
```

Variables globales :

N := M div 2 Taille initiale du sous vecteur
NSup := 0 Nombre de Suppressions logiques (trous)
Ndon := M div 2 Nombre de données présentes

Module de recherche d'une valeur v donnée :

Nous l'écrivons sous la forme d'une fonction : Si v existe, R est son indice dans V sinon R = 0.

```
I := 1 ; Trouv := FAUX
TANTQUE I <= N ET NON Trouv :
 SI NON V(I).Sup ET V(I).Val = V : Trouv := VRAI
 SINON I := I + 1 FSI
```

```
FINTANTQUE
SI Trouv : R := I SINON R := 0 FSI
```

Module d'insertion d'une valeur v après une Valeur v' de V .

```
      SI R(V') = 0 OU N = M :
 " Insertion impossible"
SINON
 POUR I = N, R(V')+1 , -1 :
  V(I+1) := V(I)
 FINPOUR
 V( R(V')+1) := (V, FAUX)
 N := N + 1
 Ndon := Ndon + 1
FSI
```

Module de suppression d'une valeur v .

```
      SI R(V) = 0 :
 " Suppression Impossible"
SINON
 V( R(V) ).Sup := VRAI /* Suppression logique */
 Nsup := Nsup + 1
 Ndon := Ndon - 1
 SI Nsup > Ndon/3 : Recuperer ( V ) FSI
FSI
```

Module Récupérer : éliminer les trous.

```
I := 1
TANTQUE I <= N :
 SI V(I).Sup :
  POUR J = I+1, N : V(J-1) := V(J) FINPOUR
   N := N - 1
 SINON I := I + 1 FSI
FINTANTQUE
Nsup := 0
```

Mesure

t(R) : N/2 (comparaisons)
t(I) : t(R) + N/2 affectations

Gain après le lancement du module Récupérer :

Avant le lancement, parmi les N éléments du vecteur il y a 3/4 de données réellement présentes et 1/4 de données supprimées (puisque Nsup = 1/3 (Ndon) + 1) :

N/2 - ((3/4)N) / 2 = N / 8

Exercice 2 : Représentation d'un tableau à 3 dimensions en mémoire

Ordre de rangement des éléments :

```
        POUR I = 1, N
  POUR J = 1, M
   POUR K = 1, P
    ECRIRE ( I, J, K)
   FINPOUR
  FINPOUR
 FINPOUR
```

Formule donnant la position de T(i, j, K) dans le tableau T(1..N, 1..M, 1..P) :

(i-1)*(M*P) + (j-1)*P + k

Généralisation : Position de T(i1, i2, ...in) dans le tableau T(1..N1, 1..N2, ...) :

(i1-1) * (N2*N3....*Nn) + (i2-1) * (N3*N4*.....*Nn) + ...(in-1 -1)*Nn + in.

* * * * *

Exercice 1 : Palindromes

```
ALGORITHME Palindrome
VAR  C, Mot1, Mot2  : CAR
 Compt   : ENTIER
DEBUT
 LIRE(C)
 Compt := 0
 TANTQUE C # '.' :
  TANTQUE C = ' ' : LIRE(C) FINTANTQUE
  Mot1, Mot2 := ''
  TANTQUE C # ' ' ET C # '.' :
   Mot1 := Mot1 + C
   Mot2 := C + Mot2
   LIRE(C)
  FINTANTQUE
  SI Mot1 # '' : {Les mots existent}
   SI Mot1 = Mot2
    Compt := Compt + 1
   FSI
  FSI
 FINTANTQUE
 ECRIRE(Compt)
FIN
```

Exercice 2 : Epreuve de sélection

```
ALGORITHME Epreuve
CONST
 N=10
 Nreponses = 3
 Nquestions = 5
VAR  I, J, K, K1, K2, N, Rep, Total : ENTIER
 Reponse  : TABLEAU(1..Nquestions, 1..Nreponses) DE ENTIER
 Admis : TABLEAU(1..N) DE ENTIER  { Admis }
 Tref : TABLEAU(1..N) DE ENTIER  { Refusés }
DEBUT
 K1 := 0
 K2 := 0
 LIRE( Reponse )  { Introduire les bonnes réponses }
 POUR I=1, N :
  Total := 0
  {Lecture des réponses du I-ième candidat }
  POUR J=1, Nquestions
   POUR K=1, Nreponses
    LIRE(Rep)
    SI Rep = 1
     SI Reponse(J, K) = Rep
      Total := Total + 5
     SINON
      Total := Total - 4
     FSI
    FSI
   FINPOUR
  FINPOUR
  SI Total > 8 :
   K1 := K1 + 1
   Admis(K1) := I
  SINON
   K2 := K2 + 1
   Tref(K2) := I
  FSI
 FINPOUR
 ECRIRE(Admis)
 ECRIRE(Tref)
FIN
```

Programme PASCAL correspondant :

```
PROGRAM Epreuve;
 CONST
   N = 10;
   Nreponses = 3;
   Nquestions = 5;
 VAR
   I, J, K, K1, K2, Rep, Total : INTEGER;
   Reponse : Array(.1..Nquestions, 1..Nreponses.) OF INTEGER ;
```

```
  Admis  : Array(.1..N.)  OF INTEGER  ; { Admis }
  Tref    : Array(.1..N.)  OF INTEGER ;  { Refusés }
  Fs, Fe : TEXT;
BEGIN
  ASSIGN(Fe, 'D_select.Pas');
  ASSIGN(Fs, 'R_select.Pas');
  { Introduire les bonnes réponses  }
  RESET(Fe);
  REWRITE(Fs);
  READLN(Fe);
  FOR I:= 1 TO Nquestions DO
    BEGIN
      FOR J:= 1 TO Nreponses DO
        READ(Fe, Reponse(. I, J.) );
      READLN(Fe)
    END;

  K1 := 0 ;
  K2 := 0;
  FOR I:=1 TO N DO
    BEGIN
      Total := 0;
      {lecture des réponses du I-ième candidat }
      READLN(Fe);  READLN(Fe);
      FOR J:=1 TO Nquestions DO
        BEGIN
          FOR K:= 1 TO Nreponses DO
            BEGIN
              READ(Fe, Rep);
              IF Rep = 1   { Réponse Cochée }
              THEN
              IF Reponse(.J, K.) = Rep
      THEN
                Total := Total + 5
              ELSE
                Total := Total - 4
            END;
          READLN(Fe)
        END;
      IF Total > 8
      THEN
        BEGIN
          K1 := K1 + 1;
          Admis(.K1.) := I
        END
      ELSE
        BEGIN
          K2 := K2 + 1;
          Tref(.K2.) := I
        END
    END;
  WRITELN(Fs, 'Admis : ');
```

```
  FOR I:= 1 TO K1 DO WRITELN(Fs, Admis (.I.));
  WRITELN(Fs, 'Refusés : ');
  FOR I:= 1 TO K2 DO WRITELN(Fs,  Tref(. I.));
  CLOSE(Fs);
  END.
```

Ce programme admet comme fichier de données (D_Select.pas)

Bonnes réponses
1 0 1
0 0 1
1 1 1
1 0 0
0 0 0

Candidat 1
0 0 1
1 0 0
1 1 1
0 0 0
0 0 1

Candidat 2
0 1 1
1 0 0
1 0 1
0 1 0
0 0 1

Candidat 3
0 1 0
1 0 0
0 1 1
1 0 0
0 1 1

Candidat 4
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0

Candidat 5
1 1 1
1 1 1
1 1 1
1 1 1
1 1 1

Candidat 6
1 1 1

1 1 0
1 0 1
0 1 0
0 1 1

Candidat 7
1 0 1
0 0 0
1 1 1
0 0 1
1 0 1

Candidat 8
0 1 1
1 1 0
1 0 1
1 0 1
0 0 0

Candidat 9
0 0 0
0 1 0
1 1 1
1 0 0
1 0 1

Candidat 10
1 0 1
1 0 0
1 1 1
1 1 0
0 1 1

et comme résultats (R_Select.pas) :

Admis :
1
7
10
Refusés :
2
3
4
5
6
8
9

* * * * *

Exercice 1 : Losange

```
PROGRAM LOSANGE;
VAR
 X : INTEGER;
 I, J : INTEGER;
BEGIN
 READ(X);
 FOR I :=1 TO (X DIV 2 + 1) DO WRITE(' '); WRITELN('O');

    FOR I := 1 TO (X DIV 2 + 1) DO
      BEGIN
        FOR J := 1 TO (X DIV 2 + 1) - I DO WRITE(' ');
        WRITE('O');
        FOR J:=1 TO 2*(I-1) + 1 DO WRITE(' ');
        WRITELN('O');
      END;

    FOR I := (X DIV 2 ) DOWNTO 1 DO
      BEGIN
        FOR J := 1 TO (X DIV 2 ) - I + 1 DO WRITE(' ');
        WRITE('O');
        FOR J:=1 TO 2*(I-1) + 1 DO WRITE(' ');
        WRITELN('O');
      END;

    FOR I :=1 TO (X DIV 2 + 1) DO WRITE(' '); WRITELN('O');
 END.
```

Exercice 2 : Mots de la forme X.....XY.....Y

Les mots de la forme X...XY....Y, c'est _ dire qui commencent par X suivi de N caractères (N <= 3) , suivi de la chaîne 'Xy', suivi de M caractères. M >=30 , suivi de Y.

```
 ALGORITHME X_xy_y
 VAR
  N, M : ENTIER
  Mot, C, Prec : CHAR
  Mot : STRING[40];
 DEBUT
  LIRE(C)
  Prec := ' '
  TANTQUE ( C <> '#' ) :
   SI C <> 'X'
    Prec := C
    LIRE(C)
   SINON
    SI Prec <> ' ' :
     LIRE(C)
```

```
SINON
 LIRE(C)
            Mot := 'X';
 N := 0;
 { Recherche de 'X' s'il existe}
 TANTQUE (  (C<>' ') ET ( C <>'#') ET (C<>'X') )
  N  := N + 1;
             Mot := Mot + C ;
             Prec := C;
  LIRE(C)
 FINTANTQUE

 SI (C = 'X') AND (N <= 3)
  Mot := Mot + 'X';
          LIRE(C);
  SI C = 'Y'
   Mot := Mot + 'Y'
   LIRE(C)
   M := 0;
   TANTQUE (  (C<>' ') ET ( C <>'#')  )
    M  := M + 1;
    Mot := Mot + C ;
    Prec:= C;
    LIRE(C)
   FINTANTQUE
   SI (Prec = 'Y') ET  (M >= 3)
    ECRIRE( Mot, N, M)
   FSI
  FSI
 FSI
FSI
FSI
FINTANTQUE
FIN
```

Exercice 3 : Reconnaissance de constantes arithmétiques

On utilisera les modules Chiffre et Erreur définis comme suit :

Module Chiffre

```
Chiffre := FAUX
SI Chiffre >= 0  ET Chiffre <= 9 : Chiffre := VRAI  FSI
```

Module Erreur :

```
       TANTQUE (C <>' ')  ET  ( C <>'$' )  :
 Mot := Mot + C
       LIRE(C)
FINTANTQUE
ECRIRE ( Mot, ' Constante erronée' )
```

L'algorithme est le suivant :

```
ALGORITHME Chiffre
VAR
   C, MOT : CAR
 ERR : BOOLEEN

DEBUT
 LIRE(C)
 TANTQUE C <> '$' :
  Err := False
   Mot := ''
   W := 0  D := 0

  TANTQUE C=' ' :  LIRE(C)   FINTANTQUE

  TANTQUE Chiffre(C)
   W := W + 1   ;  Mot := Mot + C
   LIRE(C)
  FINTANTQUE

  SI C = '.' :
      W := W + 1     ;     Mot := Mot + C
             LIRE(C)
             TANTQUE Chiffre(C)
    W := W + 1 ; D := D + 1 ;  Mot := Mot + C
    LIRE(C)
   FINTANTQUE

   SI (C<>' ') ET ( C<>'$' )
    Err :=  VRAI
    'Erreur '
   FSI
           SINON
             SI (C<>' ') ET ( C<>'$' )
    Err := VRAI
    'Erreu'r
   FSI
  FSI

  SI (Mot <> '') ET (NON  err)
   ECRIRE (  'Mot=', Mot, ' W=', W, ' D=', D)
  FSI
 FINTANTQUE
FIN
```

* * * * *

Exercice 1 : Un = 3Un-1 + 2Un-2 + Un-3

```
ALGORITHME Suite
```

```
VAR U0, U1, U2, I, J : ENTIER
U0, U1, U2 := 1 {Affectation multiple}
DEBUT
 I := 3
 TANTQUE I # N :
  U3 := 3*U0 + 2*U1 + U2
  U0 := U1
  U1 := U2
  U2 := U3
  I := I + 1
 FINTANTQUE
 ECRIRE(U3)
FIN
```

Exercice 2 : Nombres premiers

On utilise le prédicat Prem défini comme suit :

Fonction Prem (i)

en entrée : un nombre impair
en sortie : prem est vrai si le nombre impair est premier faux sinon

```
FONCTION Prem ( I ) : BOOLEEN
VAR   Divisible  : BOOLEEN
 I, J, A  : ENTIER
DEBUT
 J := 2
 Divisible := FAUX
 A := I DIV 2
 TANTQUE J <= A ET NON Divisible :
  Quotient := I DIV J
  SI Qoutient * J = I :
   Divisible := VRAI
  SINON  J := J + 1 FSI
 FINTANTQUE
 Prem := NON Divisible
FIN


ALGORITHME Premier
VAR I, N : ENTIER
DEBUT
 LIRE(N)
 ECRIRE(1, 2)
 I := 3
 TANTQUE I <= N
  SI Prem(I) : Imprimer(I) FSI
  I := I + 2
 FINTANTQUE
FIN
```

* * * * *

Exercice 1 : Mots commençant par 'M'

```
ALGORITHME M
VAR
 Im : ENTIER
 C, Sauv : CAR
DEBUT
 LIRE(C)
 Im := 0
 TANTQUE C <> '#' :
  TANTQUE C = ' ' : LIRE(C) FINTANTQUE
  Sauv := C
  TANTQUE C <> ' ' ET C <> '#' : LIRE(C) FINTANTQUE
  SI Sauv = 'M' : Im := Im + 1 FSI
 FINTANTQUE
 ECRIRE(Im)
   FIN
```

Exercice 2 : Mots se terminant par 'MME'

```
ALGORITHME MME
VAR
 Imme : ENTIER
 C : CAR
DEBUT
 LIRE(C)
 Imme := 0
 TANTQUE C <> '#' :
  SI C = 'M':
   LIRE(C)
   SI C = 'M':
    LIRE(C)
    TANTQUE C = 'M' : LIRE(C) FINTANTQUE
    SI C = 'E' :
     LIRE(C)
     SI C = ' ' :
      Imme := Imme + 1
      LIRE(C)
     FSI
       FSI
   SINON LIRE(C) FSI
  SINON  LIRE(C) FINSI
 FINTANTQUE
 ECRIRE( Imme )
   FIN
```

Exercice 3 : Mots de longueur L contenant 3 'E' :

```
ALGORITHME 3E
VAR
 Ie, Cpt, I, L : ENTIER
 C : CAR
DEBUT
 LIRE(L)
 Ie := 0
 LIRE(C)
 TANTQUE C <> '#' :
  TANTQUE C = ' ' : LIRE(C) FINTANTQUE
  Cpt := 0 ,  I := 0
  TANTQUE C <> ' ' ET C <> '#' :
   I := I + 1
   SI C = 'E' : Cpt := Cpt + 1 FSI
   LIRE(C)
  FINTANTQUE
  SI I = L ET Cpt = 3 : Ie := Ie + 1 FINSI
 FINTANTQUE
 ECRIRE(Ie)
FIN
```

* * * * *

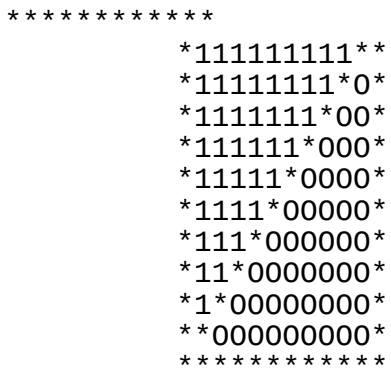
Partie 3.

Exercices programmés en Pascal

PROGRAMME 1.

Réalisation d'un dessin

Ecrire le programme PASCAL qui réalise le dessin suivant :




```
PROGRAM Dessin;
VAR
  I,N,L, K : INTEGER;
  F : TEXT;
BEGIN
   ASSIGN( F,'R_dessin.Pas');
   REWRITE(F);
   N := 12;
   WRITE(F,'                ');
   FOR I := 1 TO N DO WRITE(F,'*') ; WRITELN(F) ;
   FOR K := 2 TO N-1 DO
    BEGIN
      WRITE(F,'                ');
      WRITE(F,'*');
      FOR L:=2 TO N-K DO WRITE(F,'1');
      WRITE(F,'*');
      FOR L:=1 TO K-2 DO WRITE(F,'O');
      WRITE(F,'*');
      WRITELN(F)
    END;
   WRITE(F,'                ');
   FOR I := 1 TO N DO WRITE(F,'*') ;  WRITELN(F) ;

   CLOSE(F)
 END.
```

Contenu du fichier R_dessin.pas

```
***********
*111111111**
*11111111*0*
*1111111*00*
*111111*000*
*11111*0000*
*1111*00000*
*111*000000*
*11*0000000*
*1*00000000*
**000000000*
***********
```

PROGRAMME 2.

Etablissement d'une facture

En vue d'établir une facture, on a relevé sur chaque ligne d'un fichier TEXT PASCAL les informations que l'on organisé comme suit :
col 1 à 5 : numéro du produit ou désignation
col 10 à 14 : quantité (entier)
col 20 à 28 : prix unitaire

La dernière ligne ne contient que des zéros.

Ecrire l'algorithme puis le programme PASCAL qui édite la facture selon le format suivant :

```
 <-----15------> <---10---> <---10---> <---10--->
*----------------*----------*----------*----------*
*  DESIGNATION   * QUANTITE *    PU    *   TOTAL  *
*----------------*----------*----------*----------*
*      ???       *   ???    *   ???    *   ???    *
*                *          *          *          *
*                *          *          *          *
*                *          *          *          *
*                *          *          *          *
*                *          *          *          *
*                *          *          *          *
*                *          *          *          *
*                *          *          *          *
*                *          *          *          *
*----------------*----------*----------*----------*
*                                                  *
*                   NET A PAYER :   ???            *
*                                                  *
*----------------*----------*----------*----------*
```

```
PROGRAM Facture;
 VAR Qte, Numero : INTEGER;
  Pu, Total, Cumul : REAL;
  Fdon, Fsort : TEXT;

 PROCEDURE Entete;
  BEGIN
   WRITELN(Fsort,'*----------------*----------*----------*----------*');
   WRITELN(Fsort,'*  Désignation  * Quantit_ *    Pu    *   Total  *');
   WRITELN(Fsort,'*----------------*----------*----------*----------*');
  END;

 PROCEDURE Netapayer ( Cumul : REAL);
  BEGIN
   WRITELN(Fsort,'*----------------*----------*----------*----------*');
```

```
    WRITELN(Fsort,'*                                    *');
    WRITELN(Fsort,'*            Net A Payer  : ',Cumul:8:2,'*':8);
    WRITELN(Fsort,'*                                    *');
    WRITELN(Fsort,'*---------------*----------*----------*----------*');
  END;

 PROCEDURE Ligne ( Numero, Qte :INTEGER;Pu,Total : REAL);
  BEGIN
    WRITELN(Fsort,'*',' ':5,Numero:5,' ':5,'*', ' ':3,Qte:4,' ':3,
              '* ', Pu:8:2,' * ',Total:8:2,' *');
  END;

 { Initialisation}
 BEGIN
  ASSIGN(Fdon,'D_factur.Pas');
  ASSIGN(Fsort,'R_factur.Pas');
  RESET(Fdon);
  REWRITE(Fsort);
  Entete;
  Cumul := 0;
  READLN( Fdon, Numero, Qte, Pu);
  WHILE Numero <> 0 DO
    BEGIN
      Total := Pu * Qte;
      Ligne(Numero, Qte, Pu, Total);
      Cumul := Cumul + Total;
      READLN( Fdon, Numero, Qte, Pu);
    END;
  Netapayer(Cumul);
  CLOSE(Fsort)
 END.
```

Contenu du fichier D_facture.pas :

```
12654   24      35.3
785     31      67.12
12      18      27
00      000     0000
```

Contenu du fichier R_facture.pas :

```
*---------------*----------*----------*----------*
*  Désignation  * Quantité *    Pu    *   Total  *
*---------------*----------*----------*----------*
*     12654     *     24   *    35.30 *   847.20 *
*       785     *     31   *    67.12 *  2080.72 *
*        12     *     18   *    27.00 *   486.00 *
*---------------*----------*----------*----------*
*                                                *
*                  Net A Payer  :  3413.92       *
*                                                *
*---------------*----------*----------*----------*
```

PROGRAMME 3.

Course de Ski (I)

Pendant une course de SKI, on a relevé après un essai les temps mis par N concurrents. On désire, après avoir mis les informations sur fichier, obtenir d'abord le tableau d'affichage des temps mis par les N concurrents, puis le tableau d'affichage concernant uniquement les trois premiers.

Les données sur fichier ont la forme suivante :
Première ligne : nombre de concurrents sur 3 positions
dans les N lignes suivantes, on a recensé les informations concernant les concurrents.
Chaque ligne est organisée comme suit :
Col 1 à 4 : numéro du concurrent
Col 12 à 19 : Nom du concurrent
Col 30 à 31 : minutes
col 32 à 33 : secondes
Col 34 à 35 : centièmes de seconde

Le tableau d'affichage aura la forme suivante :

```
 <---10---> <-----15------> <-----14-----> 
*----------*---------------*--------------*
*          *               *              *
*  NUMERO  *      NOM      *    TEMPS     *
*          *               *----*----*----*
*          *               * MM * SS * CC *
*----------*---------------*----*----*----*
*          *               *    *    *    *
    * numéro 1 *    nom 1      * mm * ss * cc *
*          *               *    *    *    *
*----------*---------------*----*----*----*
*          *               *    *    *    *
Etc ..
```

Ecrire le programme PASCAL correspondant.

```
PROGRAM Ski;
 VAR
  Mm, Ss, Cc : LONGINT;
  M1, M2, S1, S2, C1, C2, C : CHAR;
  T1, T2, T3, T : LONGINT;
  I, J, N : INTEGER;
  Num1, Num2, Num3, Num : STRING;
  Nom1, Nom2, Nom3, Nom : STRING;
  Fdon, Fsort : TEXT;

 PROCEDURE Entete;
  BEGIN
```

```
    WRITELN(Fsort,'*----------*--------------*--------------*');
    WRITELN(Fsort,'*          *              *              *');
    WRITELN(Fsort,'*  Numero  *      Nom     *    Temps     *');
    WRITELN(Fsort,'*          *              *----*----*----*');
    WRITELN(Fsort,'*          *              *-Mm-*-Ss-*-Cc-*');
    WRITELN(Fsort,'*----------*--------------*----*----*----*');
  END;

 PROCEDURE Ligne ( Num, Nom : STRING; Mm, Ss, Cc : INTEGER);
  BEGIN
    WRITELN(Fsort,'*          *              *    *    *    *');
    WRITELN(Fsort,'*',Num:10,'*', Nom:15,'*', Mm:4,'*',Ss:4,'*',Cc:4,'*');
    WRITELN(Fsort,'*          *              *    *    *    *');
    WRITELN(Fsort,'*----------*--------------*--------------*');
  END;

 BEGIN
  ASSIGN(Fdon,'D_ski.Pas');
  ASSIGN(Fsort,'R_ski.Pas');
  RESET(Fdon);
  REWRITE(Fsort);
  READLN(Fdon,N);
  WRITELN(Fsort, 'Nombre de concurrents : ', N);
  Entete;

  T1:= 300000; T2:=300000; T3 := 300000;
  Num1:=''; Num2:=''; Num3 := '';
  Nom1:=''; Nom2:=''; Nom3 := '';
  FOR I:=1 TO N DO
    BEGIN
 num:='';
      FOR J:=1 TO 4 DO
        BEGIN READ(Fdon, C);Num := Num+C  END;
      FOR J:=1 TO 7 DO READ(Fdon,C);
      Nom:='';
      FOR J:=1 TO 8 DO
        BEGIN READ(Fdon, C);Nom := Nom+C  END;
      FOR J:=1 TO 10 DO READ(Fdon, C);
      READ(Fdon,M1);READ(Fdon,M2);
      Mm := (ORD(M2)-48) + (ORD(M1)-48)*10;
      READ(Fdon,S1);READ(Fdon,S2);
      Ss := ORD(S2)-48 + (ORD(S1)-48)*10;
      READ(Fdon,C1);READLN(Fdon,C2);
      Cc := ORD(C2)-48 + (ORD(C1)-48)*10;
      Ligne( Num, Nom, Mm, Ss, Cc);
T:= (60*Mm + Ss)*100 + Cc;
{ C'est le temps exprimé en Cc }
IF T < T1
      THEN
        BEGIN
   T3 := T2; num3 := Num2; nom3 := Nom2;
   T2 := T1; num2 := Num1; nom2 := Nom1;
```

```
    T1 := T; num1 := Num; nom1 := Nom;
       END
 ELSE
      IF T < T2
  THEN
        BEGIN
          T3 := T2;Num3 := Num2; nom3 := Nom2;
     T2 := T;Num2 := Num; nom2 := Nom;
   END
      ELSE
        IF T < T3
        THEN
          BEGIN
            T3 := T; Num3 := Num; Nom3 := Nom
          END
    END;

   { Ecriture des trois premiers}
   WRITELN(Fsort);
   WRITELN(Fsort);
   WRITELN(Fsort, 'Les trois premiers : ');
   Entete;
   Ss := T1 DIV 100;
   Cc := T1 - 100*Ss;
   Mm := Ss DIV 60;
   Ss := Ss - 60*Mm;
   Ligne(Num1, Nom1, Mm, Ss, Cc);

   Ss := T2 DIV 100;
   Cc := T2 - 100*Ss;
   Mm := Ss DIV 60;
   Ss := Ss - 60*Mm;
   Ligne(Num2, Nom2, Mm, Ss, Cc);

   Ss := T3 DIV 100;
   Cc := T3 - 100*Ss;
   Mm := Ss DIV 60;
   Ss := Ss - 60*Mm;
   Ligne(Num3, Nom3, Mm, Ss, Cc);
   CLOSE(Fsort)
 END.
```

Contenu du fichier D_ski.pas :

```
4
234       Hakim          223389
567       Fairouz        215623
6789      Rachida        241267
12        Hakima         232299
```

Contenu du fichier R_ski.pas :

Nombre de concurrents : 4

Numéro	Nom	Temps		
		-Mm-	-Ss-	-Cc-
234	Hakim	22	33	89
567	Fairouz	21	56	23
6789	Rachida	24	12	67
12	Hakima	23	22	99

Les trois premiers :

Numéro	Nom	Temps		
		-Mm-	-Ss-	-Cc-
567	Fairouz	21	56	23
234	Hakim	22	33	89
12	Hakima	23	22	99

PROGRAMME 4.

Les mots de la forme ....XY.....

Ecrire un programme qui détermine sur la machine-caractères tous les mots de la forme .....XY..... , c'est à dire n caractères suivi de la chaîne 'XY' suivi de m caractères. n et m sont des entiers quelconques et strictement supérieurs à 0.

On assimilera la machine-caractères à un fichier TEXT PASCAL.

```
PROGRAM Xy (Fe, Fs);
  VAR N,M : INTEGER;
    C, Sauv : CHAR; Mot : STRING[40];
    Mememot : BOOLEAN;
    Fe, Fs : TEXT;
  BEGIN
    ASSIGN ( Fe, 'D_xy.Pas' );
    ASSIGN ( Fs, 'R_xy.Pas' );
    RESET(Fe);
    REWRITE(Fs);
    READ(Fe,C);
    Mememot := FALSE;

    WHILE ( C <> '#' ) DO
      BEGIN
        { Parcours des blancs }
        WHILE ( C = ' ' ) DO READ(Fe,C);
         IF NOT Mememot
         THEN
           BEGIN
             N := 0;
             Mot := '';
           END;

         { Recherche de 'Y' s'il existe. Soient Mot la chaîne de caractères avant
le 'Y',
            N sa longueur et  Sauv le caractère qui précède 'Y' }
         WHILE (  (C<>' ') AND ( C <>'#') AND (C<>'Y') ) DO
           BEGIN
             N  := N + 1;
             Mot := Mot + C;
             Sauv := C;
             READ(Fe, C)
           END;

         IF C = 'Y'
         THEN
           IF (Sauv = 'X') AND ( N > 1)
           { Au moins un caractère avant le 'X' }
           THEN
```

```
          BEGIN
            { Rechercher le prochain blanc en comptant le nombre de
caractères après le  Y, soit M. On continue à former le mot}
            M := - 1;
            WHILE ( (C<>' ') AND ( C<>'#') ) DO
              BEGIN
                Mot := Mot + C;
                M := M + 1;
                READ(Fe,C)
              END;
            IF   M > 0
            THEN
              WRITELN(Fs, ' > Mot = ', Mot, '  ( Avant : ', N-1 : 2,' , ',  '
Apr_s : ',  M:2, ' )' );
            Mememot := FALSE;
          END
        ELSE
          { Le caractère qui précède 'Y' est différent de 'X' ou bien il n'y a pas
de caractère      avant le 'X'  ====>  Rechercher prochain Y  dans le même
mot }
          BEGIN
            Mot := Mot + C;
            READ(Fe, C);
            IF C <>' '
            THEN Mememot := TRUE;
            N := N +1
          END
      ELSE Mememot := FALSE;
    END;
    CLOSE(Fs)
  END.
```

Contenu du fichier D_xy.pas :

```
abbbbbbbbXYynnnnnn    aXYxxxxxxx      annnnyssssssssXYyyyyyyy XY
aaaaaaayaaaaaaayaaaaaaaaXYaaaa   bbbbbbbXY  XYzzzz XYXYXY #
```

Contenu du fichier R_xy.pas :

```
 > Mot = abbbbbbbbXYynnnnnn  ( Avant :  9 ,  Après :  7 )
 > Mot = aXYxxxxxxx  ( Avant :  1 ,  Après :  7 )
 > Mot = annnnyssssssssXYyyyyyyy  ( Avant : 14 ,  Après :  7 )
 > Mot = aaaaaaayaaaaaaayaaaaaaaaXYaaaa  ( Avant : 24 ,  Après :  4 )
 > Mot = XYXYXY  ( Avant :  2 ,  Après :  2 )
```

PROGRAMME 5.

Les mots de la forme ....X....Y....

Ecrire un programme qui détermine sur la machine-caractères tous les mots de la forme ...X....Y... c'est à dire c'est à dire n caractères suivis de la lettre 'X' suivi de m caractères suivis de la lettre 'Y' suivi de p caractères. n ,m et p sont des entiers quelconques et strictement supérieurs _ 0.

On assimilera la machine-caractères à un fichier TEXT PASCAL.

```
PROGRAM Xy (Fe, Fs);
  VAR N,M,P : INTEGER;
    C : CHAR; Mot : STRING[40];
    Fe, Fs : TEXT;
  BEGIN
    ASSIGN ( Fe , 'D_x_y.Pas' );
    ASSIGN ( Fs , 'R_x_y.Pas' );
    RESET(Fe);   REWRITE(Fs);
    READ(Fe,C);
    WHILE ( C <> '#' ) DO
      BEGIN
        { Parcours des blancs }
        WHILE ( C = ' ' ) DO READ(Fe,C);

        { Recherche de 'X' s'il existe
          Soit mot la chaîne de caractère avant le 'X', N sa longueur      }
        Mot := '';
        N := 0;
        WHILE (  (C<>' ') AND ( C <>'#') AND (C<>'X') ) DO
          BEGIN
            N  := N + 1;
            Mot := Mot + C;
            READ(Fe, C)
          END;

        IF C = 'X'
        THEN
          BEGIN
            Mot := Mot + C;
            M :=0;
            READ(Fe,C);
            IF C = 'Y' THEN BEGIN M:=1; Mot := Mot+C; READ(Fe,C) END;
            WHILE (  (C<>' ') AND ( C <>'#') AND (C<>'Y') ) DO
              BEGIN
                Mot := Mot + C;
                M := M+1;
                READ(Fe, C)
              END;
```

```
        IF C = 'Y'
        THEN
          BEGIN
            P := -1;
            WHILE ( (C<>' ') AND ( C<>'#') ) DO
              BEGIN
                Mot := Mot + C;
                P := P + 1;
                READ(Fe,C)
              END;

            IF M*P*N <> 0
            THEN
              WRITELN(Fs, ' > Mot = ', Mot, ' ( Avant : ', N:2,
                        ' Milieu : ', M:2, ' Apr_s : ', P:2, ' )');
          END;
        END;
    END;
    CLOSE(Fs)
  END.
```

Contenu du fichier D_x_y :

abcXaaYde abxXxaYy   bXYya aXYXYyyd XYy XY aXYyyyyyyaax XYyyzt #

Contenu du fichier R_x_y :

```
 > Mot = abcXaaYde ( Avant :  3 Milieu :  2 Après :  2 )
 > Mot = abxXxaYy ( Avant :  3 Milieu :  2 Après :  1 )
 > Mot = aXYXYyyd ( Avant :  1 Milieu :  2 Après :  3 )
```

PROGRAMME 6.

Analyse des constantes

Soit sur le ruban de la machine caractères une suite de constantes réelles représentées en virgule flottante. La suite est terminée par le caractères '$'. Les constantes sont séparées par une combinaison de blancs et de virgules. Ecrire un programme qui donne pour chaque constante correcte ses attributs W et D. Une constante en virgule flottante a la forme suivante :

[+/] [chchch...] [.][chchch....] E [+/]chch

Les crochets désignent les parties facultatives.

```
PROGRAM Virguleflottante;
 VAR Mot : STRING[30];
  Fe, Fs : TEXT;
  C : CHAR; W, D : INTEGER;

 FUNCTION Chiffre( C : CHAR) : BOOLEAN;
  BEGIN
   IF (C >='0' ) AND ( C <= '9')
   THEN Chiffre := TRUE
   ELSE Chiffre := FALSE;
  END;

 PROCEDURE Erreur;
  BEGIN
   WHILE (C <>' ')  AND  ( C <>'$' )  DO
    BEGIN
     Mot := Mot + C;
     READ(Fe,C);
    END;
   WRITELN(Fs, Mot : 35, ' Constante erronée' : 30);
  END;

 BEGIN
  ASSIGN(Fe, 'D_vf.Pas');
  ASSIGN(Fs, 'R_vf.Pas');
  RESET(Fe);
  REWRITE(Fs);
  READ(Fe, C);

  WHILE C <> '$' DO
   BEGIN
    Mot := '';
    W := 0; D := 0;

    WHILE C=' ' DO  READ(Fe,C);
     IF (C = '+') OR( C ='-')
     THEN
```

```
    BEGIN
      W := 1; Mot := Mot + C; READ(Fe,C);
    END;

WHILE ( ( C<>'.')  AND (C<>'E') AND ( C<> ' ') AND (C<>'$')
        AND ( Chiffre(C) )  ) DO
  BEGIN
    W := W + 1;
    Mot := Mot + C;
    READ(Fe, C)
  END;

IF C = '.'
THEN
  BEGIN
    W := W + 1;
    Mot := Mot + C;
    READ(Fe,C);
    WHILE ( ( C<>' ') AND ( C <>'$') AND (Chiffre(C)) AND
          ( C<>'E') ) DO
      BEGIN
        W := W + 1; D := D + 1; Mot := Mot + C;
        READ(Fe,C)
      END;
  END;

IF C = 'E'
THEN
  BEGIN
    Mot := Mot + C;
    W := W + 1;
    READ(Fe, C);

    IF (C='+')  OR (C='-')
    THEN
      BEGIN
        W := W +1; Mot := Mot + C; READ(Fe, C)
      END;

    IF Chiffre(C)
    THEN
      BEGIN
        W := W + 1; Mot := Mot + C; READ(Fe, C);
        IF Chiffre(C)
        THEN
          BEGIN
            W := W + 1;
            Mot := Mot + C;
            READ(Fe,C);
            IF( C =' ') OR( C='$')
            THEN
              WRITELN (Fs, Mot: 35, 'W=':10 ,W:2, '  D=':10, D:2);
```

```
            END
          ELSE Erreur
        END
      ELSE Erreur
    END
  ELSE
    IF C <> '$' THEN Erreur
  END;
  CLOSE(Fs);
END.
```

Contenu du fichier D_vf.pas

```
123.. 234.543E02    12E21    23.6754E-23  EEE   231E.23  E+12   1.1E-11 .345E04      -
.23E00   34ADSE $
```

Contenu du fichier R_vf.pas

```
                                  123..       Constante erronée
                               234.543E02      W=10        D= 3
                                    12E21      W= 5        D= 0
                              23.6754E-23      W=11        D= 4
                                      EEE      Constante erronée
                                  231E.23      Constante erronée
                                     E+12      W= 4        D= 0
                                  1.1E-11      W= 7        D= 1
                                  .345E04      W= 7        D= 3
                                  -.23E00      W= 7        D= 2
                                   34ADSE      Constante erronée
```

PROGRAMME 7.

Recherche dichotomique dans un vecteur

Ecrire un programme qui remplit un vecteur à partir d'un fichier TEXT et recherche dichotomiquement un ensemble de données également lues sur le fichier d'entrée.

On utilisera les modules Remplir et Recherche définis comme suit :

- Remplir : remplir un vecteur à partir d'un fichier de données
- Recherche : recherche dichotomique d'une donnée dans un vecteur

```
PROGRAM Dicho;
 VAR I :INTEGER;
    Tab : ARRAY(.1..10.) OF INTEGER;
    Fs : TEXT;  { Fichier de sortie }
    Fe : TEXT;  { Fichier d'entrée  }
    N, D : INTEGER;

 { Remplissage du tableau }
 PROCEDURE Remplir;
  VAR
   Nombre, I : INTEGER;
  BEGIN
   READLN(Fe, Nombre);
   FOR I:=1 TO Nombre DO
    READLN(Fe, Tab[I] );
  END;

 { Recherche dichotomique }
 FUNCTION Recherche ( Cle : INTEGER ) : BOOLEAN;
  VAR
   M, Bi, Bs : INTEGER;
   Trouv : BOOLEAN;
  BEGIN
   Bi := 1;
   Bs := 10;
   Trouv := FALSE;
   WHILE (Bi <= Bs) AND NOT Trouv DO
    BEGIN
     M := (Bi + Bs) DIV 2;
     IF Tab(.M.) = Cle
     THEN Trouv := TRUE
     ELSE IF Tab(.M.) > Cle
         THEN Bs := M -1
         ELSE Bi := M + 1
    END;
```

```
    Recherche := Trouv
  END;

 BEGIN
  ASSIGN( Fe,'D_dicho.Pas');
  ASSIGN( Fs,'R_dicho.Pas');
  RESET(Fe);
  REWRITE(Fs);
  Remplir;
  READLN(Fe, N ); {nombre de données à rechercher}
  FOR I:= 1 TO N DO
   BEGIN
    READLN(Fe, D);
    IF Recherche(D)
    THEN WRITELN(Fs, 'L''élément ', D, ' existe')
    ELSE WRITELN(Fs,'L'élément ', D, ' n''existe pas');
   END;
  CLOSE(Fs)
 END.
```

Contenu du fichier D_dicho.pas :

```
10   Nombre d'éléments
23
32
34
45
54
72
76
76
78
89
12   Nombre de données à rechercher
34
76
23
89
5
99
74
32
45
54
34
72
```

Contenu du fichier R_dicho.pas :

```
L'élément 34 existe
L'élément 76 existe
```

L'élément 23 existe
L'élément 89 existe
L'élément 5 n'existe pas
L'élément 99 n'existe pas
L'élément 74 n'existe pas
L'élément 32 existe
L'élément 45 existe
L'élément 54 existe
L'élément 34 existe
L'élément 72 existe

PROGRAMME 8.

Course de ski (II)

On désire contrôler par ordinateur l'affichage d'une course de ski. Ecrire l'algorithme et le programme PASCAL qui après chaque arrivée d'un concurrent affiche le classement partiel sous la forme :

```
 <---10---> <-------20---------> <-4> <----14------>
*************************************************
*CLASSEMENT*  NOM DU CONCURRENT *PAYS*    TEMPS     *
*          *                   *    *              *
*          *                   *    * MN  SEC CT   *
*************************************************
*          *                   *    *              *
*    1     *     xxxxx         * yy *  5   10   86 *
*          *                   *    *              *
*************************************************
*          *                   *    *              *
```

Remarques :

1) A chaque arrivée, il faudra insérer le concurrent à la place correspondante au classement fait sur le temps de parcours.
2) description des données :
Pays, Nom prénom, Temps(en minutes, secondes et centièmes de seconde). Les données sont disposées à raison d'une ligne par concurrent comme suit :
paysBB nom et prénom BBMnScCs
< 4> < 20 > < 6 >

3) L'affichage se fera sur imprimante suivant le format ci-dessus. Chaque tableau sera imprimé au milieu d'une page du listing.

4 Le jeu d'essai comportera 10 concurrents, on n'oubliera pas de traiter les ex-æquo.

```
PROGRAM Ski1;
 TYPE Typ = RECORD
  Nom , Pays : STRING;
  Temps : INTEGER
 END;
 VAR  mm, Ss, Cc : INTEGER;
  Trouv : BOOLEAN;
  T, I, J, K, N, Nbr : INTEGER;
  Pays, Nom : STRING;
  Fdon, Fsort : TEXT;
  Tab : ARRAY[1..10] OF Typ;

 PROCEDURE Entete;
  BEGIN
```

```
   WRITELN(Fsort);
   WRITELN(Fsort);
   WRITELN(Fsort,
    '*****************************************************';
   WRITELN(Fsort,'*Classement*  Nom Du Concurrent *Pays*    Temps     *');
   WRITELN(Fsort,'*          *                *    *            *');
   WRITELN(Fsort,'*          *                *    * Mn   Sec  Ct *');
   WRITELN(Fsort,
   '*****************************************************');
 END;

PROCEDURE Ligne ( Rang : INTEGER; Pays,Nom : STRING; Mm, Ss, Cc : INTEGER);
 BEGIN
   WRITELN(Fsort,'*          *                *    *            *');
   WRITELN(Fsort,'*   ',Rang:4,'*':4, Nom:15,'*', Pays:4,'*',Mm:4,' ',
              Ss:4,' ',Cc:4,'*');
   WRITELN(Fsort,'*          *                *    *            *');
   WRITELN(Fsort,
   '*****************************************************');
 END;

PROCEDURE Lire(VAR Pays,Nom : STRING; VAR Mm, Ss, Cc :INTEGER);
 VAR J : INTEGER;
   M1, M2, S1, S2, C1, C2, C : CHAR;
 BEGIN
   Pays:='';FOR J:=1 TO 4 DO BEGIN READ(Fdon, C);Pays:= Pays+C END;
   FOR J:=1 TO 2 DO READ(Fdon,C);
   Nom:='';FOR J:=1 TO 20 DO BEGIN READ(Fdon, C);Nom := Nom+C  END;
   FOR J:=1 TO 2 DO READ(Fdon, C);
   READ(Fdon,M1);READ(Fdon,M2);
   Mm := (ORD(M2)-48) + (ORD(M1)-48)*10;
   READ(Fdon,S1);READ(Fdon,S2);
   Ss := ORD(S2)-48 + (ORD(S1)-48)*10;
   READ(Fdon,C1);READLN(Fdon,C2);
   Cc := ORD(C2)-48 + (ORD(C1)-48)*10;
 END;

 {initialisation}
 BEGIN
   ASSIGN(Fdon,'D_ski1.Pas');
   ASSIGN(Fsort,'R_ski1.Pas');
   RESET(Fdon);
   REWRITE(Fsort);
   READLN(Fdon,N);
   WRITELN(Fsort, 'Nombre de concurrents : ', N);
   Nbr := 0;
   FOR I:=1 TO N DO
     BEGIN
       Lire(Pays, Nom, Mm, Ss, Cc);
       T := (60*Mm + Ss)*100 + Cc;
       J:= 1; Trouv := FALSE;
       WHILE ( J <= Nbr) AND (NOT Trouv) DO
```

```
        IF Tab(.J.).Temps >= T
        THEN    Trouv := TRUE
        ELSE    J := J + 1;
      IF Trouv
      THEN
        FOR K:=Nbr DOWNTO J DO  Tab(.K+1.) := Tab(.K.);
      Tab(.J.).Temps := T;
      Tab(.J.).Nom := Nom;
      Tab(.J.).Pays := Pays;
 Nbr := Nbr + 1;
      WRITELN(Fsort);
      WRITELN(Fsort, ' >>> Classement après le concurrent ', I);
      WRITELN(Fsort);
      Entete;
      FOR K := 1 TO Nbr DO
       BEGIN
        T := Tab(.K.).Temps;
        Ss := T DIV 100;
        Cc := T - 100*Ss;
        Mm := Ss DIV 60;
        Ss := Ss - 60*Mm;
        Nom := Tab(.K.).Nom;
        Pays := Tab(.K.).Pays;
        Ligne(K, Pays, Nom, Mm, Ss, Cc)
       END
 END;
    CLOSE(Fsort)
 END.
```

```
4
ALGE   Hakima                014579
GERM   Rachida               025480
URSS   Hakim                 015987
BRES   Fairouz               013410
```

Nombre de concurrents : 4
>>> Classement après le concurrent 1

```
*****************************************************
*Classement*  Nom Du Concurrent *Pays*    Temps      *
*          *                    *    *               *
*          *                    *    * Mn   Sec  Ct *
*****************************************************
*          *                    *    *               *
*      1   * Hakima              *ALGE*   1   45   79*
*          *                    *    *               *
*****************************************************
```

>>> Classement après le concurrent 2

```
*****************************************************
*Classement*  Nom Du Concurrent *Pays*    Temps      *
*          *                    *    *               *
*          *                    *    * Mn   Sec  Ct *
*****************************************************
```

```
*           *                       *    *                *
*       1   * Hakima                *ALGE*   1   45   79*
*           *                       *    *                *
*****************************************************
*           *                       *    *                *
*       2   * Rachida               *GERM*   2   54   80*
*           *                       *    *                *
*****************************************************
```

>>> Classement après le concurrent 3

```
*****************************************************
*Classement*  Nom Du Concurrent *Pays*    Temps      *
*           *                       *    *                *
*           *                       *    * Mn   Sec  Ct *
*****************************************************
*           *                       *    *                *
*       1   * Hakima                *ALGE*   1   45   79*
*           *                       *    *                *
*****************************************************
*           *                       *    *                *
*       2   * Hakim                 *URSS*   1   59   87*
*           *                       *    *                *
*****************************************************
*           *                       *    *                *
*       3   * Rachida               *GERM*   2   54   80*
*           *                       *    *                *
*****************************************************
```

>>> Classement après le concurrent 4

```
*****************************************************
*Classement*  Nom Du Concurrent *Pays*    Temps      *
*           *                       *    *                *
*           *                       *    * Mn   Sec  Ct *
*****************************************************
*           *                       *    *                *
*       1   * Fairouz               *BRES*   1   34   10*
*           *                       *    *                *
*****************************************************
*           *                       *    *                *
*       2   * Hakima                *ALGE*   1   45   79*
*           *                       *    *                *
*****************************************************
*           *                       *    *                *
*       3   * Hakim                 *URSS*   1   59   87*
*           *                       *    *                *
*****************************************************
*           *                       *    *                *
*       4   * Rachida               *GERM*   2   54   80*
*           *                       *    *                *
*****************************************************
```

PROGRAMME 9.

Parties d'un ensemble

Ecrire le programme qui liste toutes les parties d'un ensemble donné.

```
{ Génération automatique de toutes les parties d'un ensemble }

PROGRAM Parties;
 TYPE
  Pointeur = ^Typemaillon;   Typemaillon = RECORD
   Val : INTEGER;
   Adr : Pointeur
  END;
  T = RECORD
   Tete : Pointeur;
   Nb : INTEGER
  END;
 VAR Liste1, Liste2 : ARRAY[1..10] OF T;
  Fs  : TEXT;
  N, N1, N2, I, J, K, Nb, Long, Compt : INTEGER;
  Q, L : Pointeur;
  V : ARRAY[1..10] OF INTEGER;
  Aig : BOOLEAN;

 PROCEDURE Allouer( VAR P : Pointeur );
  BEGIN NEW(P) END;

 PROCEDURE Affval( VAR P : Pointeur; Val :INTEGER );
  BEGIN P^.Val := Val END;

 PROCEDURE Affadr( VAR P : Pointeur; Q : Pointeur );
  BEGIN P^.Adr := Q END;

 FUNCTION Suivant( P : Pointeur ) : Pointeur;
  BEGIN Suivant := P^.Adr END;

 FUNCTION Valeur( P : Pointeur ) : INTEGER;
  BEGIN Valeur := P^.Val END;

 PROCEDURE Imprimer(L : Pointeur);
  BEGIN
   WRITE(Fs, '+ {');
   WHILE (L <> NIL) DO
    BEGIN
     WRITE(Fs, Valeur(L), ',' );
     L := Suivant(L)
    END;
```

```
    WRITELN(Fs, '}');
  END;

 { Recherche de la valeur Val dans la liste L}

 FUNCTION Recherche ( Val:INTEGER; L:Pointeur)  : BOOLEAN;
   VAR Trouv : BOOLEAN;
   BEGIN
     Trouv := FALSE;
     WHILE ( (L <> NIL) AND (NOT Trouv )) DO
IF Valeur(L) = Val
THEN Trouv := TRUE
       ELSE L := Suivant(L);
     Recherche :=  Trouv
   END;

 { Union des I-ième et J-ième listes contenues dans le tableau Liste1
  ou Liste2 selon la valeur de Aig. L est la liste résultante, Nb
  son nombre d'éléments.
  Remarque : L contient tous les maillons de la I-ième liste.      }

 PROCEDURE Union ( Aig : BOOLEAN; I, J: INTEGER; VAR L:
                Pointeur; VAR Nb : INTEGER);
   VAR L1, L2, Q : Pointeur;
     Nb1 : INTEGER;
   BEGIN
     IF Aig
     THEN
       BEGIN
         L1 := Liste1[I].Tete;
         Nb1 :=Liste1[I].Nb;
         L2 := Liste1[J].Tete
       END
     ELSE
       BEGIN
  L1 := Liste2[I].Tete;
    Nb1 :=Liste2[I].Nb;
  L2 := Liste2[J].Tete
       END;
     L := L1;
     Nb := Nb1;
     WHILE ( L2 <> NIL) DO
       BEGIN
         IF NOT Recherche(Valeur(L2), L1)
  THEN { Ajout au d_but de L1}
           BEGIN
             Nb := Nb + 1;
             Allouer(Q);
      Affval(Q, Valeur(L2));
      Affadr(Q, L);
             L := Q
           END;
```

```
 L2 := Suivant(L2)
     END
END;

 { Teste l'égalité  entre 2 listes }

FUNCTION Egal (L1:Pointeur; Nb1:INTEGER; L2:Pointeur; Nb2:INTEGER) :
           BOOLEAN;
  VAR Trouv : BOOLEAN;
  BEGIN
    IF Nb1 = Nb2
    THEN
      BEGIN
       Trouv := FALSE;
        WHILE ( (L2 <> NIL) AND (NOT Trouv )) DO
   IF NOT Recherche( Valeur(L2), L1)
   THEN Trouv := TRUE
   ELSE  L2 := Suivant(L2);
 Egal := NOT Trouv
     END
    ELSE  Egal := FALSE
  END;

{ Recherche de la liste L dans le tableau Liste1 ou Liste2 selon la valeur de Aig. }

FUNCTION Exist ( Aig : BOOLEAN; L:Pointeur ) : BOOLEAN;
  VAR   I : INTEGER;
    Trouv : BOOLEAN;
  BEGIN
    I := 1;
    Trouv := FALSE;
    IF Aig
    THEN
      WHILE ( (I <= N2) AND (NOT Trouv )) DO
        IF Egal(Liste2[I].Tete, Liste2[I].Nb, L, Nb)
        Then trouv := TRUE
        ELSE    I := I + 1
    ELSE
      WHILE ( (I <= N1) AND (NOT Trouv )) DO
        IF Egal(Liste1[I].Tete, Liste1[I].Nb, L, Nb)
 Then trouv := TRUE
 ELSE    I := I + 1;
     Exist := Trouv
  END;

BEGIN
  N := 5;
  { Initialisation de V}
  FOR I:=1 TO 10 DO V[I] := I;
  ASSIGN(Fs, 'R_parties.Pas');
  REWRITE(Fs);
  WRITELN(Fs, '      Ensemble de toutes les parties ');
```

```
   WRITELN(Fs);
   WRITELN(Fs, '{ }');
   WRITELN(Fs,' Ensembles d''un  élément');
   FOR I:=1 TO N DO
    BEGIN
     Allouer(Q);
Affval(Q, V[I] );
Affadr(Q, NIL);
Liste1[I].Tete := Q;
Liste1[I].Nb := 1;
Imprimer(Q)
    END;

   N1 := N;
   N2 := 0;
   Aig := TRUE;
   Long := 2;
   Compt := 1 + N;
   WHILE ( N <> 1 ) DO
    BEGIN
     WRITELN(Fs,' Ensembles de ',Long, '  éléments');
for I:=1 TO N DO
       FOR J:=I+1 TO N DO
        BEGIN
          Union(Aig, I, J, L, Nb);
          IF NOT Exist(Aig, L) AND(Long = Nb)
          THEN
            IF Aig
            THEN
              BEGIN
                N2 := N2 + 1;
                Liste2[N2].Tete := L;
                Liste2[N2].Nb := Nb;
              END
            ELSE
              BEGIN
                N1 := N1 + 1;
         Liste1[N1].Tete := L;
 Liste1[N1].Nb := Nb;
       END
        END;
       IF Aig
       THEN
        BEGIN
          N1 := 0; N:= N2;
          FOR K:= 1 TO N DO Imprimer( Liste2[K].Tete)
        END
       ELSE
        BEGIN
          N2 := 0; N:= N1;
          FOR K:= 1 TO N DO Imprimer( Liste1[K].Tete)
        END;
```

```
        Aig := NOT Aig;
        Long := Long + 1;
        Compt := Compt + N;
      END;
    WRITELN(Fs,'Nombre de parties : ', Compt);
    CLOSE(Fs);
  END.
```

Contenu du fichier R_parties.pas

```
     Ensemble de toutes les parties

{ }
 Ensembles d'un élément
+ {1,}
+ {2,}
+ {3,}
+ {4,}
+ {5,}
 Ensembles de 2  éléments
+ {2,1,}
+ {3,1,}
+ {4,1,}
+ {5,1,}
+ {3,2,}
+ {4,2,}
+ {5,2,}
+ {4,3,}
+ {5,3,}
+ {5,4,}
 Ensembles de 3  éléments
+ {3,2,1,}
+ {4,2,1,}
+ {5,2,1,}
+ {4,3,1,}
+ {5,3,1,}
+ {5,4,1,}
+ {4,3,2,}
+ {5,3,2,}
+ {5,4,2,}
+ {5,4,3,}
 Ensembles de 4  éléments
+ {4,3,2,1,}
+ {5,3,2,1,}
+ {5,4,2,1,}
+ {5,4,3,1,}
+ {5,4,3,2,}
 Ensembles de 5  éléments
+ {5,4,3,2,1,}
Nombre de parties : 32
```

PROGRAMME 10.

Inventaire

On dispose d'un fichier TEXT PASCAL décrivant la liste des produits en stock dans les différents magasin d'une chaîne de supermarchés.
Chaque ligne de ce fichier a le format suivant :
n°produit prix unitaire nombre d'unités en stock
<-- 4 -->BB<---- 7 ---->BB<---------4------------>

La dernière ne contient que des zéros. B désigne un blanc.
1) Créer à partir de ce fichier un autre fichier STOCK (de type FILE) où chaque enregistrement est une structure à 3 champs : numéro du produit, prix unitaire et nombre d'unités en stock.
2) Pour l'inventaire de fin d'année et à partir du fichier STOCK défini en 1), éditer un état ayant la forme suivante :

MAGASIN 1
RAYON 1 : MONTANT DU STOCK ???
RAYON 2 : MONTANT DU STOCK ???
.....
MONTANT TOTAL DU STOCK ???

MAGASIN 2
RAYON 1 : MONTANT DU STOCK ???
.....

MONTANT DU STOCK POUR L'ENSEMBLE DES MAGASIN ???

où les '?' sont remplacés par les valeurs calculées par le programme.
- Les numéros de magasins et de rayons dans chaque magasin sont supposés tous présents à partir de 1. Le fichier est tri par numéro de magasin croissant et pour un magasin donné par numéro de rayon croissant.
- Pour un produit donné, le numéro de magasin est constitué par les deux premiers chiffres du numéro de produit, et le numéro du rayon par les deux chiffres suivants.
Ex : O4174043 produit vendu au magasin n° 4, rayon n° 17

```
PROGRAM Stock;
 TYPE
  T = RECORD
    Nm : INTEGER;{numéro de magasin}
    Nr : INTEGER;{numéro de rayon}
    Num: STRING
  END;

  Typearticle = RECORD
    Numprod : T;
```

```
   Pu : REAL;
   Qte : INTEGER
  END;

  Tfile = FILE OF Typearticle;
VAR
  Im, Ir : INTEGER;
  Mmag, Mray, Mtotal : REAL;
  E : Typearticle;
  Fsort : TEXT;
  Fstock : Tfile;
  Ftext : TEXT;

{ Création du fichier Fstock à partir d'un fichier TEXT}

PROCEDURE Creer(VAR Ft : TEXT; VAR Fs : Tfile );
  VAR Num : STRING;
   Pu : REAL;
   Qte : INTEGER;
   Nm, Nr : INTEGER;
   Art : Typearticle;

  PROCEDURE Lire( VAR Nm, Nr:INTEGER;VAR Num : STRING;VAR Pu:REAL;
              VAR Qte : INTEGER );
   VAR N_m, N_r: STRING;
    J : INTEGER;
    C : CHAR;
   BEGIN
    N_m:='';
    FOR J:=1 TO 2 DO
     BEGIN READ(Ft, C);N_m := N_m+C  END;
    Nm := ORD(N_m(.2.))-48 + (ORD(N_m(.1.))-48)*10;
    N_r:='';
    FOR J:=1 TO 2 DO
     BEGIN READ(Ft, C);N_r := N_r+C  END;
    Nr := ORD(N_r(.2 .))-48 + (ORD(N_r(.1.))-48)*10;
    Num:='';
    FOR J:=1 TO 4 DO
     BEGIN READ(Ft, C);Num := Num+C  END;
    FOR J:=1 TO 2 DO READ(Ft,C);
    READ(Ft, Pu);
    READLN(Ft, Qte);
   END;

  BEGIN
   ASSIGN(Ft,'D_stock.Pas');
   ASSIGN(Fs,'R1_stock');
   RESET(Ft);
   REWRITE(Fs);
   Lire(Nm, Nr, Num, Pu, Qte);
   WHILE ( Nm <> 0 ) DO
    BEGIN
```

```
      Art.Numprod.Nm := Nm;
 Art.Numprod.Nr := Nr;
 Art.Numprod.Num := Num;
      Art.Pu := Pu;
 Art.Qte := Qte;
 WRITE(Fs, Art);
      Lire(Nm, Nr, Num, Pu, Qte);
    END;
   Art.Numprod.Nm := 0;
   WRITE(Fs,Art);
 END;

{ Inventaire}
BEGIN
  ASSIGN(Fstock,'R1_stock');
  Creer(Ftext, Fstock);
  ASSIGN(Fsort,'R_stock.Pas');
  REWRITE(Fsort);
  RESET(Fstock);
  READ(Fstock, E);
  Im := 1; Ir := 1; Mray := 0;
  Mmag := 0;  Mtotal := 0;
  WRITELN(Fsort, ' ':5,'Magasin N_',Im:3);
  WHILE (NOT EOF(Fstock) ) DO
    IF E.Numprod.Nm = Im
    THEN
      IF E.Numprod.Nr = Ir
   then
       BEGIN
         Mray := Mray + E.Qte * E.Pu;
         READ(Fstock, E)
       END
     ELSE
       BEGIN
   WRITELN(Fsort,' ':10,'Rayon N°', Ir:3,':':10, Mray:8:2 );
         Mmag := Mmag + Mray;
         Mray := 0;
         Ir := Ir + 1
 END
    ELSE
     BEGIN
       Mmag := Mmag + Mray;
       Mtotal := Mtotal + Mmag;
       WRITELN(Fsort,' ':10,'Rayon N°', Ir:3,':':10, Mray:8:2 );
       WRITELN(Fsort);
       WRITELN(Fsort,'*':10,' Montant total du rayon : ', Mmag:8:2);
       WRITELN(Fsort);
       WRITELN(Fsort,' ':5,'Magasin N°', Im+1:3);
       Im:=Im + 1;
 Ir := 1;
       Mmag := 0;
       Mray := 0;
```

```
    END;
  WRITELN(Fsort,' ':10,'Rayon N°', Ir:3,':':10, Mray:8:2 );
  Mmag := Mmag + Mray;
  WRITELN(Fsort,'*':10,' Montant total du rayon : ', Mmag:8:2);
  WRITELN(Fsort);
  Mtotal := Mtotal + Mmag;
  WRITELN(Fsort,'*':5,' Montant total du stock : ', Mtotal:8:2 );
  CLOSE(Fsort)
END.
```

Contenu du fichier D_stock.pas

```
01014532  13.23  24
01014536  45.34  12
01022343  23.23  16
01025643  15.8   42
01036798  12    87
02016754  6.76   12
02016543  5.4    98
02017845  1.65   34
03014444  3.56   12
03025555  7.89   8
03036666  7.54   23
04012222  77.2   34
00000000  0000   00
```

Contenu du fichier R_stock.pas

```
     Magasin N°  1
          Rayon N°  1         :  861.60
          Rayon N°  2         : 1035.28
          Rayon N°  3         : 1044.00

         * Montant total du rayon :  2940.88

     Magasin N°  2
          Rayon N°  1         :  666.42

         * Montant total du rayon :   666.42

     Magasin N°  3
          Rayon N°  1         :   42.72
          Rayon N°  2         :   63.12
          Rayon N°  3         :  173.42

         * Montant total du rayon :   279.26

     Magasin N°  4
          Rayon N°  1         : 2624.80
         * Montant total du rayon :  2624.80

    * Montant total du stock :  6511.36
```

Partie 4.

Annexes

Langage algorithmique

Variables et constantes

Un algorithme opère sur des objets. A tout objet est associé un **nom** qui permet de l'identifier de façon unique. C'est généralement une suite de caractères alphanumériques dont le premier est alphabétique.

On distingue deux types d'objets :
- des objets qui peuvent varier durant le déroulement d'un algorithme **Variables** (ardoises).
- des objets qui ne peuvent pas variés par le déroulement d'un algorithme **Constantes**.

On peut répartir l'ensemble des objets en sous ensembles appelés **classes** ou types. Il existe 4 types standards :
- ENTIER : l'ensemble des entiers relatifs
- REEL : l'ensemble des réels
- BOOLEEN : les valeurs VRAI et FAUX
- CAR : l'ensemble des chaînes de caractères

En langage algorithmique, on définit les objets comme suit :
CONST Type Nom = valeur
VAR Nom1, Nom2, ... : Type

Objets composés

Définition de type

Au niveau du langage algorithmique, un type est construit comme suit :

```
    TYPE   Nomdutype = STRUCTURE
Idf1 : Type1
Idf2 : Type2
 ....
FIN
```

Typei peut être un type standard ou construit.

Déclaration des variables d'un type donné

On définit les objets d'un type construit comme suit:

Var Nom : Type

Nom est le nom de l'objet composé et Type un type quelconque préalablement construit.

On peut aussi si nécessaire définir une relation d'ordre entre les éléments appartenant à une même classe d'objets.

Accès à un champ d'un objet composé

Si X est une variable d'un type donné, X.Nom permet d'accéder au champ Nom de la variable composée X.

Expressions sur les objets

Une expression est une suite d'opérandes reliés par des opérateurs.

Une expression peut être :
- arithmétique. On utilisera les opérateurs suivants +,-,/, *.
Exemple A + B/C, A/B/C

- logique. Les opérateurs les plus utilisés sont : ET, OU et NON.
Exemple X et Y, NON X OU Z

- relationnelle. Les opérateurs utilisés sont : <, <=, >, >=, =, <>.
Exemple : A < B, A+5 = B/C

Vecteurs

Description algorithmique d'un vecteur

On décrit un vecteur en langage algorithmique comme suit :

VAR Nom : TABLEAU[1..N] DE Type

Nom est le nom du tableau, et Type est un type quelconque.

Description algorithmique d'un tableau à d dimensions

On décrit un tableau à plusieurs dimensions en langage algorithmique comme suit :

VAR Nom : TABLEAU[1..N1, 1..N2, ..., 1..Nd] DE Type

Nom est le nom du tableau, Type est un type quelconque et d est la dimension du tableau.

Accès à un élément d'un tableau

Il se fait par l'opération d'indexation. Si T est le nom d'un tableau à une dimension, T[I] permet l'accès à son I-ième élément. Si T est le nom d'un tableau à deux dimensions, T[I,J] permet l'accès à l'élément de ligne I et de colonne J.

Listes linéaires chaînées

Définition d'un maillon d'une liste linéaire chaînée

Dans le langage algorithmique, on définira le type d'un maillon comme suit :

```
    TYPE Typedumaillon = STRUCTURE
 Valeur  : Typeqq   { désigne un type quelconque }
 Adresse  : POINTEUR(Typedumaillon)
FIN
```

POINTEUR est un mot-clé du langage algorithmique servant à définir la classe des adresses.

Définition d'une liste linéaire chaînée

Une liste linéaire chaînée est définie par l'adresse de son premier élément.

VAR Liste : POINTEUR(Typedumaillon)

Modèle sur les listes linéaires chaînées

Afin de développer des algorithmes sur les listes linéaires chaînées, on construit une machine abstraite avec les opérations suivantes :

Allouer, Libérer, Aff_Adr, Aff_Val, Suivant, Valeur

définies comme suit :

Allouer(T, P) : allocation d'un espace de taille spécifiée par le type T. L'adresse de
cet espace est rendue dans la variable POINTEUR P.
Libérer(P) : libération de l'espace pointé par P.
Valeur(P) : consultation du champ Valeur du maillon d'adresse P.
Suivant(P) : consultation du champ Adresse du maillon d'adresse P.
Aff_Adr(P, Q): dans le champ Adresse du maillon d'adresse P, on range l'adresse Q.
Aff_Val(P,Val): dans le champ Valeur du maillon d'adresse P, on range la valeur Val.

Fichier

Description algorithmique d'un fichier

On définit un fichier comme suit :

VAR Nom : FICHIER DE Type

Nom est le nom du fichier et Type le type d'un article ou bloc du fichier.

Le fichier peut être vu comme
- un ensemble d'articles
- un ensemble de blocs

Dans le cas où le fichier est vu comme un ensemble de blocs, on peut considérer le cas où les articles sont de longueur fixe et le cas où ils sont de longueur variable.

Dans le premier cas, le bloc est une structure contenant au moins un tableau d'objets. Dans le second cas, le bloc est considéré comme un tableau de caractères. La récupération des différents champs des articles est à la charge du programmeur.

Opérations

Ouvrir(Nom) : ouverture du fichier
Fermer(Nom) : ferméture du fichier

Positionner(Nom, I) : positionnement au I-ième article (bloc) du fichier Nom.

Lirefichier(Nom, Zone) : lecture dans le buffer Zone de l'article (ou le bloc) courant.
Ecrirefichier(Nom, Zone) : écriture du contenu du buffer Zone dans le fichier à la position courante.

Les opérations de lecture et d'écriture permettent d'avancer la position courante dans le fichier d'une unité. A l'ouverture du fichier, la position courante est à 1.

Article ou bloc d'en-tête

Au sein d'un même fichier, on peut avoir des articles (ou blocs) de types différents : un type pour les articles (bloc) et un type pour l'article (bloc) d'en-tête contenant les caractéristiques, c'est à dire toutes les informations utiles pour l'exploitation du fichier.

```
    TYPE Nomdutype = STRUCTURE
 CAS Select DE
 1 : ( définitions des champs des articles ou des blocs )
 2 : ( définition des champs de l'article ou du bloc d'en-tête )
FIN
```

Afin de modifier le type courant, on positionne le selecteur à 1 ou 2 selon que l'on fait une lecture (écriture) d'un article du fichier ou d'un article d'en-tête.

Si Zone est une variable de type Nomdutype, la selection se fait l'opération

Select(Valeur)

avec Valeur dans l'ensemble { 1, 2 }

La selection courante reste valide jusqu'à la nouvelle opération de sélection.

Structures de contrôle

Tous les algorithmes peuvent être exprimés par les "tournures" suivantes qu'on appelle structures de contrôle du fait qu'elles permettent de contrôler le déroulement des algorithmes :

Le séquencement

Exprime qu'un ensemble d'actions est exécuté dans un ordre bien précis.

La répétitive " TANTQUE"

Elle exprime qu'un ensemble d'actions se déroule plusieurs fois tant qu'une condition reste vérifiée.

TANTQUE condition :
Action 1
Action 2
....
Action n
FINTANTQUE

La condition constitue le critère d'arrêt.

La répétitive " POUR"

On utilise cette structure quand on connaît à l'avance le nombre d'itérations d'une boucle. La formulation algorithmique est la suivante :

POUR Variable := Expression1, Expression2, Expression3
Action 1
Action 2

.....

Action n
FINPOUR

Variable désigne une variable entière.
Expression1, Expression2 et Expression3 désignent des expressions entières.
Expression1 est la valeur initiale de Variable, Expression2 la valeur finale et expression3 l'incrément.

Toute boucle "POUR" peut se traduire en boucle "TANTQUE". Par contre, l'inverse n'est pas toujours possible.

La conditionnelle

Elle exprime qu'un ensemble d'actions est exécuté que si une condition est vérifiée.

SI Condition :
Action 1
Action 2
....
Action n
FSI

L'alternative

Elle exprime un choix entre deux ensembles d'actions à exécuter selon la valeur d'une condition.

SI condition :
Action 1
Action 2
....
Action n
SINON
Action n+1
Action n+2
....
Action n + p
FSI

Remarque : Chaque action peut être à son tour une répétitive, une conditionnelle, une alternative.

Autres actions du langage algorithmique

Affectation

C'est l'opération de base. Elle permet d'attribuer la valeur d'une expression calculable à une variable. On utilisera le symbole :=.

Lecture

Elle permet d'introduire des données dans des variables. Nous utiliserons l'opération

LIRE(X, Y, Z, ...)

Ce qui est équivalent à

X := première donnée
Y := deuxième donnée
Z := troisième donnée

Ecriture

Elle permet de restituer des résultats. Nous utiliserons l'opération

ECRIRE(Expression1, Expression2,.....)

Modules

Actions composées

Quand une séquence d'actions se répète plusieurs fois dans un algorithme, il est avantageux et même nécessaire de créer une **action composée** (ou **module**). En langage algorithmique, on définit un module de la façon suivante :

ACTION Nom (V1, V2,)

Définition des paramètres d'entrée et de sortie
Définition des objets locaux
DEBUT
Corps
FIN

Au niveau de la définition de l'action , les paramètres sont dits **formels**.
L'appel se fait par Nom (P1, P2,....). Les paramètres sont dits **réels** ou **effectifs**.

Fonctions

Quand le résultat de l'action composée est unique et de type arithmétique (ou caractère) , il est préférable d'écrire l'action composée sous la forme d'une *fonction*.

Une fonction est définie comme suit :

FONCTION Nom (Paramètres d'entrée) : Type
Définition des paramètres et des objets locaux
DEBUT
Corps
FIN

Une fonction est utilisée directement dans une expression.

Dans le corps de la fonction il doit toujours exister une affectation du genre
" Nom de la fonction := valeur ".
C'est cette valeur qui est retournée par la fonction.

Prédicat

Quand le résultat de l'action composé est unique et de type booléen, il est préférable d'écrire l'action composée sous la forme d'un *prédicat*.

PREDICAT Nom (Paramètres)
Définition des paramètres et des objets locaux
DEBUT
Corps
FIN

Dans le corps du prédicat il doit toujours exister une affectation du genre
" Nom du prédicat := valeur ".

Structure d'un algorithme

Un algorithme est composé de deux parties :
- définition des données et des modules : en-tête,
- corps de l'algorithme.

L'en-tête précise
- le nom de l'algorithme,
- la définition des objets manipulés et des modules utilisés.

Le corps est un ensemble d'actions ordonnées composé généralement de 3 parties :
- initialisations (lectures, ..),
- itérations (parcours d'un ensemble de données),
- opérations finales (écritures, ..).

La description algorithmique est la suivante :

ALGORITHME Nom
Définition des objets
Définition des modules
DEBUT
Corps
FIN

Présentation générale du langage PASCAL

Eléments de base

1 Vocabulaire

C'est l'ensemble
- des identificateurs (nom des variables et constantes),
- mots clés (BEGIN, END, WHILE, ...),
- séparateurs ('.', ';', ',',...),
- opérateurs ('+', '-', '/',...),
- chaînes de caractères.

2 Objets

Les objets sont répartis en deux classes : les constantes et les variables. Les objets peuvent être de type : entier (INTEGER) , réel (REAL), caractère (CHAR), booléen (BOOLEAN)

Les constantes sont définies par :
CONST Identificateur = valeur

Les variables sont déclarées par*:*
VAR Identificateur1, Identificateur2,.... : Type

3 Expressions

Les expressions peuvent être arithmétiques (+, -, *, /, Div, ...), logiques (Not, And, OR,...) ou relationnelles (<, <=, >, >=, <>).

4 Instructions

Une instruction PASCAL peut être simple ou composée.

Dans ce qui suit, Instruction désigne une instruction simple ou composée, Variable une variable et Expression une expression.

Instructions simples

Donnons dans un premier temps les instructions simples suivantes :

Affectation : *Variable := Expression*
Boucle While : *WHILE Expression DO instruction*

Conditionnelle : *IF Cond THEN Instruction*
Alternative : *IF Cond THEN Instruction ELSE Instruction*

Lecture : *READ(Variable)*
Ecriture : *WRITE (Expression)*

Instruction composée

Une instruction composée a la forme suivante :

BEGIN Instruction1; Instruction2; .....END

Exemples

Affectation : *A := (A + B) / C*

Boucle While : *WHILE X > 5 DO X := X + 5*

```
WHILE X+Y  >= 1000 DO
 BEGIN
                              X := X + 10 ;
      Y := Y - 2
 END
```

Conditionnelle : *IF A > B THEN READ(X, Y)*

Alternative : *IF X < Y THEN WRITE(X) ELSE WRITE(Y)*

Lecture : *READ(X, Y, Z)*
Ecriture : *WRITE (' Résultats : ', Res)*

5 Structure d'un programme

Un programme PASCAL a la forme suivante :

```
PROGRAM Nom;
 définition des constantes  ( Const .... )
 définition des variables  (Var .... )
BEGIN
 Corps du programme
END.
```

6 Exemples

Reprenons nos exemples et donnons les programmes PASCAL correspondants.

Equation du second degré

```
PROGRAM Equation;
VAR A, B, C, Delta : REAL;
BEGIN
 READ(A B, C);
 Delta := B              *B - 4 A*C;
 IF  Delta  > 0
 THEN
  BEGIN
  WRITE( 'la première racine est',  - B+RAC(Ardoise)/4A*C);
```

```
  WRITE( 'la deuxième racine est',  -B- RAC(Ardoise)/4 A*C)
  END
 ELSE
  IF Delta = 0
   WRITE( ' Une racine double',  - B / 2*A )
  ELSE
   WRITE( ' Pas de racine réelle' )

END.
```

Racine cubique

```
PROGRAM Racine_cubique;
VAR  A, X0,  Mu, Ancien, Nouveau, Difference : REAL;
BEGIN
 READ(A, X0, Mu);
 Ancien := X0;
 Nouveau := ( 2* Ancien +  A/  (Ancien*Ancien) ) / 3 ;
 Difference := Abs (  Nouveau - Ancien  ) ;
 WHILE  Difference < Mu DO
  BEGIN
   Ancien := Nouveau;
   Nouveau :=   ( 2*Ancien +  A/ Ancien*Ancien ) / 3;
   Difference :=  Abs ( Nouveau - Ancien )
  END;
 WRITE (' Racine carrée de', A , " est ", Nouveau )
END.
```

Entrées/Sorties PASCAL

1 Lecture

Toute introduction de données se fait par l'ordre :

READ[LN](V1, V2, ...Vn)

[ ] désigne une partie facultative.

Vi est une variable de type INTEGER, REAL ou CHAR.

Cette instruction provoque la lecture de n données à partir de l'écran. Pour les nombres, les données sont séparées par des blancs. Pour les caractères, le lecture se fait toujours caractère/caractère.

Si l'option LN est présente, il y a positionnement à la ligne suivante après la lecture des données.

2 Ecriture

Un restitution de résultats se fait par l'ordre

WRITE[LN] (P1, P2, ..., Pn)

P1, P2, ..., Pn sont des expressions suivies éventuellement du mode d'écriture.

On peut avoir les 3 formes :
- E
- E : E1
- R : E1 : E2

avec E, E1, E2, R des expressions.

E : expression de type INTEGER, CHAR, REAL, BOOLEAN
R : expression de type REAL
E1, E2 de type INTEGER indique une largeur de champ (E1 : nombre total de caractères de la valeur écrite, E2 : nombre de chiffres après le point décimal)

Première forme : E

Si E1 n'est pas spécifiée, une valeur (pour E1) par défaut est assurée dépendant du type c'est à dire :
CHAR : 1
INTEGER : 11
BOOLEAN : 8
REAL : 20

Ces valeurs peuvent changer selon la version PASCAL considérée.

Deuxième forme : E : E1

E1 désigne le nombre total de caractères à écrire. Si E1 ne suffit pas, le nombre ne sera pas tronqué sinon cadré à gauche par des blancs. Si E est réelle, le nombre est écrit en virgule flottante.

Troisième forme : R : E1 : E2

La partie fractionnaire de R doit comporter E2 caractères. Le nombre est écrit en format virgule fixe sans exposant.

Exemple

```
PROGRAM  Exemple;
CONST Tab = 5;
VAR I, I1, I2  : INTEGER;
 R1  : REAL;
 B : BOOLEAN;
BEGIN
 B := TRUE ;
```

```
 I1 := -3 ;
 R1 := 4.5;
 I2 := 6 ;
 WRITELN(' Exemple de sortie');
 WRITELN; { saut de ligne }
 WRITELN(' ':Tab, 'I1=',I1:I2,'R1=',R1:13);
 WRITELN(I1:2, I1, 'B=', B : 3 );
 WRITELN( R1:8, R1:4:1)
END.
```

Ce programme donne en sortie :

```
 I1=       -3R1=  4.500000E+00
-3-3B=TRUE
 4.5E+00  4.5
```

3 Les fichiers TEXT

Au lieu de lire les données à partir de l'écran, ce qui peut être fastidieux lors de la mise au point des programmes, il est préférable et même très avantageux de lire les données à partir d'un fichier TEXT construit préalablement par un éditeur de texte.

La déclaration d'un tel fichier se fait comme suit

VAR Fe : TEXT

où Fe désigne le nom logique du fichier.

Tout fichier déclaré de la sorte doit être lié à un fichier physique. Ce lien est établit grâce à l'instruction ASSIGN définie comme suit

ASSIGN (Nom logique, Nom physique)

De plus, le fichier doit être ouvert par l'opération

RESET(Nom logique)

Les opérations de lectures sont faites par :

READ[LN] (Nom logique, V1, V2, ...Vn)

De même, il est conseillé de récupérer les résultats sur un fichier TEXT, puis d'aller vers un éditeur de texte et d'exploiter calmement les résultats de votre programme. Il faudra donc déclarer le fichier et le lier avec le fichier physique comme précédemment.

Par contre le fichier doit être ouvert par

REWRITE(Nom logique)

et les opérations d'écriture se font par

WRITE[LN] (Nom logique, P1, P2, ...Pn)

Il faut toujours rajouter l'instruction CLOSE(Nom logique) afin de ne pas perdre les dernières écritures sur le fichier.

Exemple

Le programme qui suit effectue la somme des données lues sur le fichier Entree.pas et écrit les sommes temporaires sur le fichier Sortie.pas.

```
PROGRAM SOMME;
VAR
 Fe, Fs : TEXT;
 I, S, Nombre, Val : INTEGER ;
BEGIN
   ASSIGN(Fe, 'Entree.pas');
 ASSIGN(Fs, 'Sortie.pas');
 RESET(Fe); REWRITE(Fs);
 READLN(Fe, Nombre);
 S := 0;
 FOR I:= 1 TO Nombre DO
  BEGIN
   READLN(Fe, Val);
   S := S + Val;
   WRITELN(Fs, 'Somme temporaire = ', S);
  END;
 WRITELN(Fs, '> Somme = ', S);
 CLOSE(Fs)
  END.
```

Contenu du fichier Entree.pas :

```
12     Nombre d'éléments
34
65
87
34
23
64
93
88
12
54
34
33
```

Contenu du fichier Sortie.pas :

```
Somme temporaire = 34
Somme temporaire = 99
Somme temporaire = 186
Somme temporaire = 220
```

```
Somme temporaire = 243
Somme temporaire = 307
Somme temporaire = 400
Somme temporaire = 488
Somme temporaire = 500
Somme temporaire = 554
Somme temporaire = 588
Somme temporaire = 621
> Somme = 621
```

Proverbes de programmation

- Définissez les problèmes complètement.

- Réfléchissez d'abord, vous programmerez plus tard.

- Choisissez bien vos identificateurs.

- Evitez les astuces.

- Employez les commentaires.

- Soignez la présentation.

- Fournissez une bonne documentation.

- Testez le programme à la main avant de l'exécuter.

- Ne vous occuper pas d'une belle présentation des résultats avant que le programme soit correct.

- Quand le programme tourne soignez la présentation des résultats.

- N'ayez pas peur de tout recommencer.

- Divisez pour régner.

- ***Ne supposez jamais que l'ordinateur suppose quelque chose.***

Comprendre progressivement l'art de la programmation, maîtriser les algorithmes de base. Tels sont les objectifs recherchés à travers cet ouvrage.

_ Ce livre – en deux tomes - décrit d'une manière très succincte et originale les concepts de base de l'algorithmique et d'une manière générale de la programmation.
_ De nombreux algorithmes sont développés sur la machine de Turing permettant de s'expérimenter sur le formalisme algorithmique.
_ Une méthode de conception d'algorithmes qu'est l'analyse descendante est exposée en mettant en évidence ses caractéristiques.
_ On y trouvera également des notions de quelques structures de données élémentaires telles que les objets composés, les tableaux et les listes linéaires chaînées.
_ Une introduction aux fichiers et aux structures de fichiers est également exposée et étoffée de nombreux programmes.
_ Un éventail de sujets d'examens avec des corrigés-type portant sur tous les cours est proposé. Ainsi, plus d'une centaine d'algorithmes sont proposés et solutionnés dans un langage algorithmique clair et concis.
_ Enfin, une série d'exercices programmés en PASCAL est aussi fournie.

Ce livre s'adresse à des étudiants désirant apprendre l'art de la programmation. Il est aussi destiné aux enseignants, principalement comme un guide.