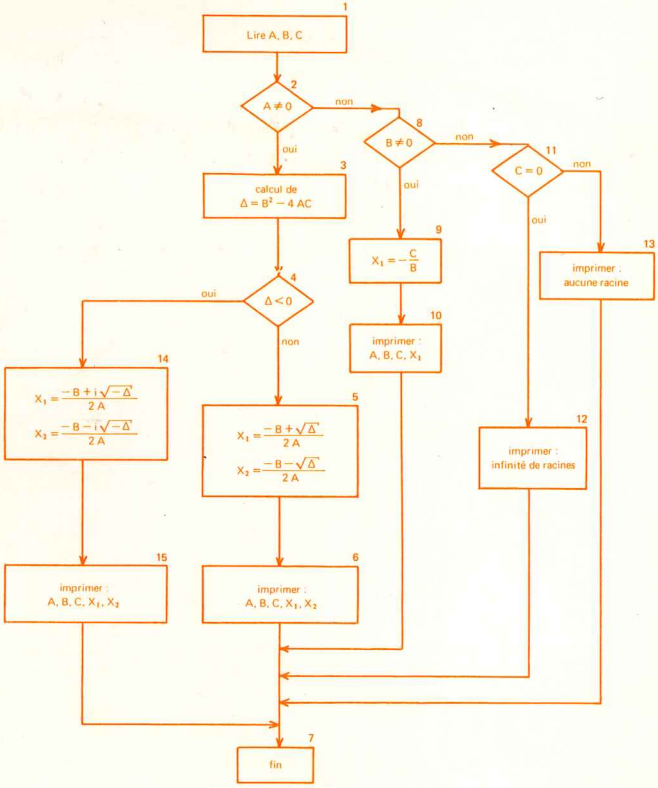
MONOGRAPHIES DE LA COMMISSION ROMANDE DE MATHÉMATIQUE 6
Société Suisse des professeurs de Mathématique et de Physique

Gérald Haury
Raymond Morel

éléments de programmation et de langage fortran



spesdunod

Dans la même collection

Th. BERNET et G. REUSSER	équations équations-exercices calcul littéral
J.-Cl. PONT et M.-A. PICHARD	méthodes modernes en mathématique élémentaire
R. LANG, G. HAURY, A. OLZA	éléments de logique

ISBN 2-602-00016-7

Gérald Haury
Raymond Morel

éléments de programmation et de langage fortran

MONOGRAPHIES DE LA COMMISSION
ROMANDE DE MATHÉMATIQUE
Société Suisse des Professeurs
de Mathématique et de Physique

ÉDITIONS SPES, LAUSANNE
DUNOD, PARIS

PRÉFACE

En ajoutant le présent ouvrage à sa collection de monographies, la C.R.M. répond à un réel besoin de nombreux maîtres et de leurs élèves. A plusieurs reprises la Société suisse des professeurs de mathématique et de physique a organisé des cours ayant pour objet principal l'informatique. De nombreuses questions touchant à l'introduction au gymnase de la programmation et à l'utilisation de tel ou tel type d'ordinateur se sont alors posées aux participants. Le problème de la finalité d'un tel enseignement a aussi été soulevé et discuté. Il semble admis que l'informatique doit, au gymnase, être considérée non pas comme un but en soi mais bien comme un moyen à disposition des enseignants et des enseignés. Grâce à elle, en effet, nombre de problèmes élémentaires, impossibles à résoudre pratiquement naguère, pourront maintenant être traités en classe apportant ainsi un nouveau type d'illustration des matières figurant aux programmes scientifiques des gymnases.

Le livre de MM. G. Haury et R. Morel a été conçu dans cet esprit. L'expérience des auteurs dans le domaine de l'enseignement du calcul électronique assure à leur ouvrage le réalisme et l'efficacité pédagogiques. C'est en effet depuis plusieurs années que MM. G. Haury et R. Morel poursuivent l'œuvre de pionniers qu'ils ont entreprise au Collège de Genève : les cours facultatifs de calcul électronique dont le succès grandit d'année en année.

Les "Eléments de programmation et de langage FORTRAN" viennent donc à leur heure : outil pédagogique de grande valeur ils contribueront à promouvoir un puissant moyen d'investigation scientifique dans les gymnases tout en lui assignant des limites raisonnables étant donné la nature de ces écoles. Pour le futur étudiant qui voudrait se spécialiser par la suite ces Eléments constitueront une solide base de départ.

Avec "Eléments de programmation et de langage FORTRAN" la C.R.M. est heureuse d'étendre le domaine réservé à ses monographies tout en respectant les buts qu'elle a fixés à cette collection.

A. Olza
Président de la C.R.M.
Juillet 1974

AVANT-PROPOS

"Si l'enseignement général vise à permettre aux lycéens d'aujourd'hui de vivre et de travailler dans le monde de demain, alors il faut inscrire l'informatique au programme des écoles secondaires."

C'est la conclusion du séminaire, organisé par le Centre pour la recherche et l'innovation dans l'enseignement (CERI) de l'OCDE, réuni à Sèvres en mars 1970. Plusieurs cantons suisses constatent avec satisfaction que les expériences menées jusqu'à ce jour dans le domaine de l'informatique vont exactement dans le sens des conclusions du séminaire du CERI. C'est le cas notamment du Collège de Genève (classes gymnasiales) où un cours de programmation et de langage FORTRAN est enseigné à plus de 200 élèves chaque année.

Pourquoi le langage FORTRAN ? Il est évident que pour de futurs universitaires le choix ne peut se porter que sur un langage évolué : le COBOL ne s'imposant pas dans des écoles gymnasiales, le FORTRAN a été préféré à l'ALGOL uniquement pour des raisons pratiques. En effet, l'Université de Genève et l'Ecole polytechnique fédérale de Lausanne travaillent essentiellement dans ce langage.

A qui s'adresse cet ouvrage ?

Si le but premier consiste à remettre dans les mains des gymnasiens un ouvrage relativement simple pour l'apprentissage de la programmation, les nombreux exercices corrigés permettront également aux **autodidactes** de se plonger dans un des domaines les plus révolutionnaires de notre époque. Enfin cet ouvrage, de par sa présentation, permettra aux personnes qui ne disposent pas d'un ordinateur de "faire un programme" et de le comparer à celui qui est testé sur l'ordinateur CDC 3800 du Centre cantonal d'informatique à Genève ou sur l'ordinateur time-sharing HB 1642 du Collège de Genève. Le lecteur ne s'étonnera donc pas de trouver quelques imperfections typographiques dues au fait que les "listing" de l'ordinateur ont été fidèlement reproduits. Ce choix implique naturellement l'utilisation parfois du FORTRAN 3800 ou du FORTRAN HB 1642 qui est parfois légèrement différent du FORTRAN standard. C'est la raison pour laquelle le sigle ▲ est utilisé pour renvoyer le lecteur à l'annexe page 172 qui met en évidence les quelques différences ▲1. La numérotation des exercices apparaît dans un rectangle. Le premier nombre correspond au numéro du chapitre

et le second à celui de l'exercice. Ainsi 3.12. signifie chapitre 3, exercice 12. D'autre part, les exercices dont la réponse, sinon une indication qui facilite la résolution (organigramme notamment), est donnée au chapitre 14, sont marqués d'un carré ■.

Cet ouvrage est conçu de telle sorte que la personne qui veut seulement se faire une idée de ce qu'est la programmation peut parfaitement se contenter de lire les sept premiers chapitres. Elle pourra déjà **tester** des programmes présentant un intérêt certain et avoir une connaissance suffisante du FORTRAN pour se faire une idée de ce langage. Le lecteur averti constatera que cet ouvrage ne constitue nullement un cours avancé d'informatique. Il est destiné essentiellement à l'étude d'un langage : c'est pour cette raison que l'aspect de la connaissance des ordinateurs et de l'informatique a été en partie écarté.

Toute personne qui enseigne la programmation s'achoppe à une difficulté : celle de devoir présenter presque toutes les notions nouvelles à la fois. C'est la raison pour laquelle un exemple concret de programme a été traité au chapitre 2 déjà, dans le but d'amener des notions nouvelles avant même de les avoir développées.

Les auteurs tiennent à exprimer leur très vive gratitude à M. F. Louis, mathématicien au CERN, responsable du cours de programmation, qui a bien voulu collaborer avec l'enseignement secondaire genevois en échangeant de nombreux documents tirés d'une part du cours donné dans le cadre du programme d'enseignement académique au CERN et d'autre part du cours de programmation donné au Collège Calvin. Ils tiennent à rendre un hommage particulier au travail fourni par les professeurs de calcul électronique et les élèves du Collège de Genève, à M. D. Thalmann qui a collaboré à la rédaction du chapitre 12, à M. le professeur Levrat, titulaire de la Chaire de calcul électronique de l'Université de Genève, pour ses conseils avisés et enfin à la Direction de l'enseignement secondaire pour sa bienveillante compréhension et ses encouragements constants.

Il leur reste enfin l'agréable devoir de remercier la Commission romande de mathématique de son soutien et de ses remarques judicieuses pour l'élaboration de cet ouvrage, notamment MM. B. Calame et F. Taillard qui ont eu la délicate tâche de se prononcer sur le contenu de cet ouvrage.

G. HAURY
Président du Groupe informatique
de l'Enseignement secondaire genevois

R. MOREL
Professeur au Collège Calvin
Responsable du Centre de calcul électronique
du Collège de Genève

TABLE DES MATIÈRES

CHAPITRE 1

GÉNÉRALITÉS

1.1. INTRODUCTION

Si la première machine à traiter l'information a été inventée par PASCAL en 1643, en revanche la première "machine à programme" n'a été réalisée qu'en 1800. Il s'agit en réalité de l'Orgue de Barbarie. Il faudra attendre le milieu du XXe siècle pour voir apparaître l'ordinateur qui est en fait un mariage entre ces deux systèmes.

A partir de ce stade l'évolution des ordinateurs subira une accélération extraordinaire : en effet la première génération d'ordinateurs (1952 – 1957) utilise des lampes ou des tubes. Dès 1958 et jusqu'en 1963 la deuxième génération est construite notamment à l'aide de transistors; enfin dès 1964 apparaît la troisième génération grâce aux circuits intégrés.

L'ordinateur se compose d'une **unité centrale** et **d'éléments périphériques.** Parmi les éléments périphériques on distingue essentiellement trois catégories :

- *les éléments d'entrée* (lecteur de cartes, lecteur-perforateur de cartes, lecteur de documents, lecteur de bandes perforées, console, etc.);
- *les éléments de sortie* (imprimante, console, perforateur de cartes, perforateur de bandes, etc.);
- les éléments qui permettent de stocker l'information appelés *mémoire auxiliaire* (bandes magnétiques, disques magnétiques, etc.).

L'unité centrale est le centre nerveux de l'ordinateur, elle comprend :

- *la mémoire centrale* qui permet de mémoriser les programmes et les données;
- *l'unité arithmétique* qui exécute toutes les opérations arithmétiques et logiques;
- *l'unité de contrôle* qui permet de contrôler et commander l'exécution du programme, de coordonner les autres éléments.

SCHÉMA D'UN ORDINATEUR "TIME-SHARING"
(traitement à distance)

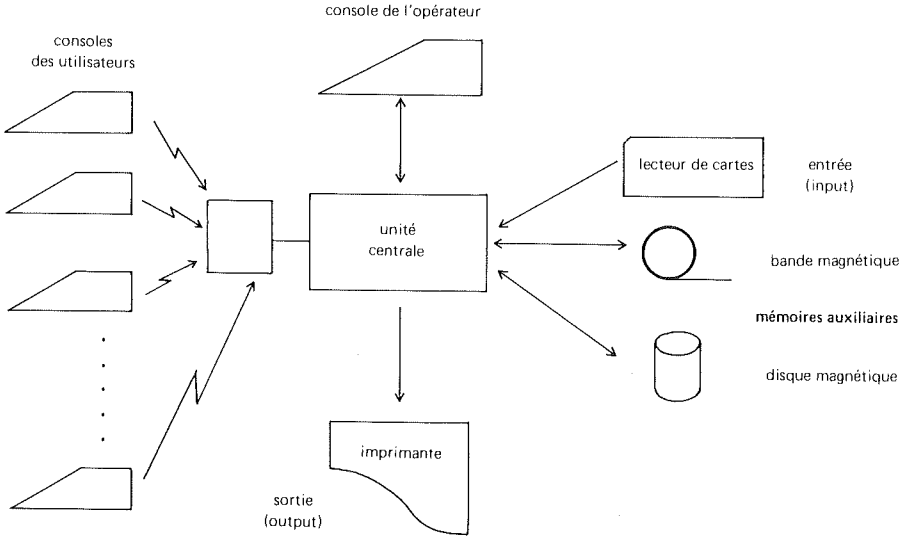


Fig. 0

Les premiers ordinateurs exigeaient de la part de l'utilisateur d'écrire des programmes dans un langage proche de la machine ("langage machine"). Par la suite, afin d'éviter un travail fastidieux, des langages plus proches du langage humain ont été inventés : ce sont les **langages de programmation**. Parmi ces langages on relèvera les suivants :

- BASIC (BEGINNERS ALL-PURPOSE SYMBOLIC INSTRUCTION CODE),
  langage évolué, mais très simple, développé pour General Electric au Dartmouth College en 1966.
- COBOL (COMMON BUSINESS ORIENTED LANGUAGE),
  le langage COBOL a été créé en 1960 aux USA. Ce langage évolué est avant tout orienté vers les problèmes de gestion.
- FORTRAN (FORMULAR TRANSLATION),
  le langage FORTRAN a également été créé aux USA en 1956. A l'instar du COBOL ce langage évolué est orienté vers les problèmes scientifiques. Il en est de même du langage ALGOL (ALGORITHMIC LANGUAGE).

— PL 1 (PROGRAMMING LANGUAGE No 1),
le PL 1 est un langage évolué qui fait la synthèse du FORTRAN et du COBOL.

Il est évident que si l'utilisateur peut disposer d'un langage proche du langage humain cela signifie que l'ordinateur doit traduire ce langage évolué en langage machine. Le **compilateur** se charge de ce travail de traduction et ce n'est qu'à ce stade que le programme peut être **exécuté**.

1.2. CARTE PERFORÉE

Il convient de distinguer deux sortes de cartes :

a) les cartes FORTRAN qui permettent d'écrire le programme

b) les cartes DONNÉE qui permettent de transcrire les données numériques.

Nous verrons par la suite que dans la plupart des cas chaque ligne d'un programme est perforée sur une seule carte.

1.2.1. CARTE FORTRAN

La carte FORTRAN se compose de 80 colonnes comprenant chacune 12 lignes de perforation. Les lignes imprimées sont numérotées de 0 à 9. En revanche, les lignes 12 et 11 ne sont pas imprimées (voir fig. 1).

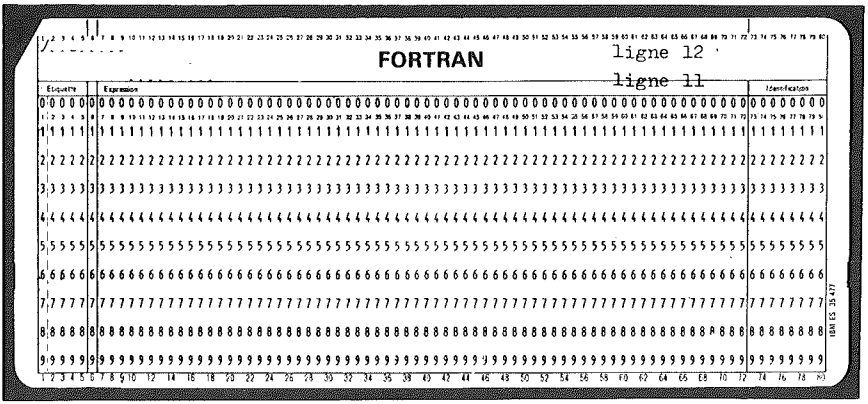


Fig. 1

On remarquera qu'à chaque **chiffre** correspond **une** perforation. A chaque **lettre** correspondent **deux** perforations par colonne (une perforation sur les lignes 12, 11 ou 0 ainsi qu'une perforation sur les lignes 1 à 9).

Enfin certains **symboles** seront parfois réalisés à l'aide de **trois** perforations par colonne.

Exemple :

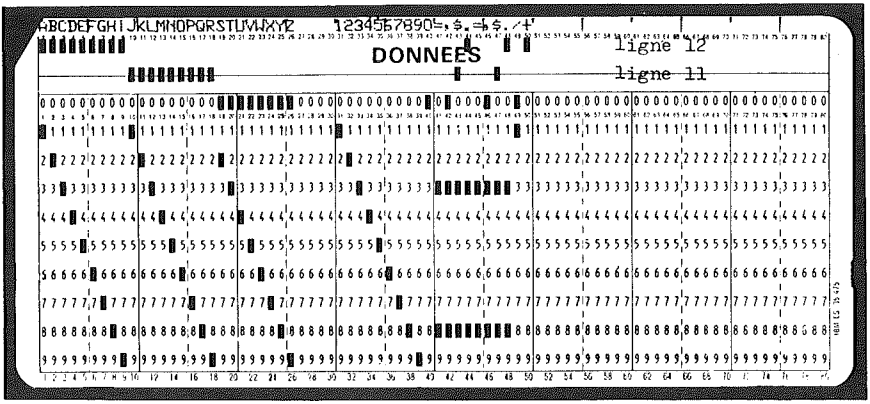


Fig. 2

Le code IBM pour les lettres est le suivant :

	1	2	3	4	5	6	7	8	9
12	A	B	C	D	E	F	G	H	I
11	J	K	L	M	N	O	P	Q	R
0		S	T	U	V	W	X	Y	Z

Remarque :

La lettre O est toujours représentée en FORTRAN par un 0 barré : Ø pour éviter toute confusion dans la perforation avec le chiffre zéro (0).

Exercice :

1.1. Représenter sur une carte les perforations correspondant à votre
NOM PRENOM ANNEE DE NAISSANCE

On distingue 4 zones dans une carte FORTRAN :

1. Colonnes 1 à 5

Les colonnes 1 à 5 permettent de repérer certaines instructions. ▲17

Exemple :

1 2 3 4 5	6	7 8 9 . . . . .
		R E A D (9 , 1 0 0) A , B
1 0 0		F Ø R M A T (. . . . .)

En écrivant la lettre C dans la colonne 1, la totalité des colonnes 1 à 80 est retranscrite. On utilise fréquemment ce procédé pour insérer des **commentaires** dans un programme. ▲23

Exemple :

1	2 3 4 5	6	7 8 9 . . . . .
C	C A L C	U L	D U R A Y Ø N

2. Colonne 6

Pour indiquer qu'une ligne d'un programme est la suite de la précédente on écrit un chiffre de 1 à 9 dans la colonne 6 appelée "**colonne suite**" ▲24

3. Colonnes 7 à 72

Les colonnes 7 à 72 sont réservées aux instructions du programme.

4. Colonnes 73 à 80

Les caractères inscrits dans les colonnes 73 à 80 ne sont pas interprétés et permettent par exemple de numéroter les cartes d'un programme.

1.2.2. CARTES DONNÉES

Voir chapitre 5.

1.3. PERFORATRICE

La plupart des centres de calcul sont équipés de perforatrices alphanumériques imprimantes. Ces machines transforment habituellement les données alphabétiques et numériques sous forme de perforations de 1,4 sur 3,2 mm effectuées sur les cartes. Ces perforatrices sont prévues aussi bien pour la perforation que l'interprétation.

Exercices :

1.2. Perforer sur une carte FORTRAN

a) les chiffres de 0 à 9
b) les lettres de l'alphabet
c) les signes spéciaux

1.3. Perforer sur une carte FORTRAN

a) 1 , I
b) 0 , Ø
Comparer les caractères.

1.4. Etablir la carte suivante :

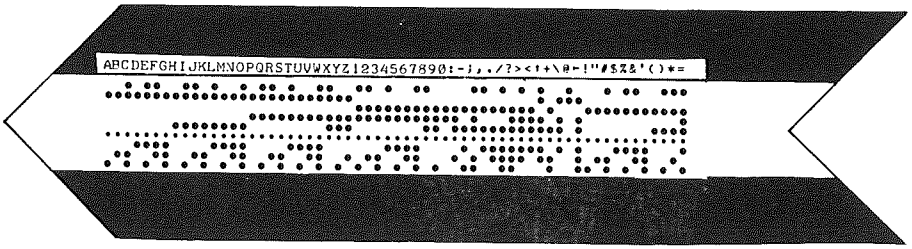
1 2 3 4 5 ... Nom et Prénom	27 ... Sexe	30 ... Adresse	56 ... N° postal	61 ... N° tél.	66 ... Classe	72 ... Ecole
DEbDARDELbbJACQUES	M	32, bRØUTEbDEbCHENE	1206	268293	3LA	CALVIN

Un signe **b** représente un espace laissé en blanc.

1.4. LA BANDE PAPIER

Sur de nombreux terminaux (type ASR, Olivetti, etc.), le dialogue avec l'ordinateur se fait au moyen du clavier d'une machine à écrire reliée à la machine du centre de calcul par ligne téléphonique aux extrémités de laquelle se trouve deux "modem" (**mo**dulateur-**dém**odulateur de fréquences).

Parallèlement au clavier existe souvent un lecteur-perforateur de bande papier; cet appareil facilite la préparation des fichiers à traiter (programmes ou données) et donne également une possibilité de stockage et d'archivage des informations. Voici un exemple de ruban avec la correspondance entre les divers caractères et leur représentation sur la bande papier :



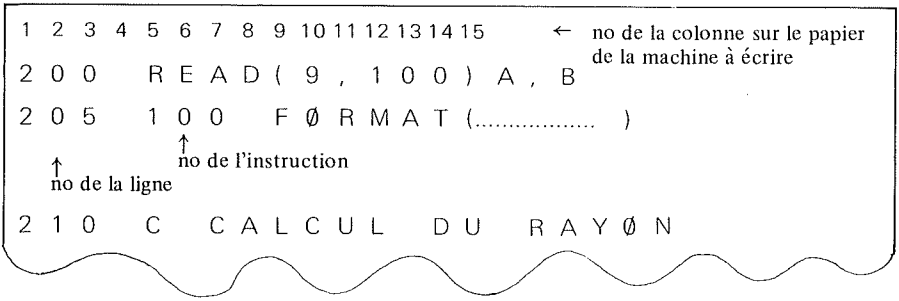
Le désavantage relatif du traitement séquentiel des bandes papier est largement compensé lorsque l'on possède dans un ordinateur un éditeur de fichiers performant (langage traitant des caractères ou des suites de caractères). En effet toutes les corrections et modifications peuvent être effectuées en mode conversationnel sur le fichier conservé sur disque magnétique après la lecture du ruban papier qui ne sert généralement qu'une fois. Il faudra cependant avoir un numéro pour chaque ligne d'un fichier afin de repérer le lieu exact d'une insertion, suppression ou modification ultérieure; là encore l'éditeur est très utile. En revanche les contraintes des sept colonnes de la carte perforée sont plus souples lors de la création d'un programme FORTRAN sur la console.

Reprenons les exemples de la page 5 :

a) pour le cas où les numéros de chaque ligne seraient ajoutés après la lecture du fichier par l'éditeur

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15      ← no de la colonne sur le papier
R E A D ( 9 , 1 0 0 ) A , B                de la machine à écrire
1 0 0   F Ø R M A T (.....)
C   C A L C U L b D U b R A Y Ø N
```

b) pour le cas où les numéros de chaque ligne sont donnés par l'utilisateur dès le départ (ici de 5 en 5 à partir de 200) avec la possibilité naturellement de modifications ultérieures par l'éditeur



CHAPITRE 2

NOTION DE PROGRAMME

2.1. EXEMPLE DE PROGRAMME

Comme indiqué dans l'avant-propos, nous allons développer entièrement un problème afin de mettre en évidence les différentes phases du traitement d'un problème sur ordinateur.

Voici le problème que nous voulons résoudre : dans le but de représenter graphiquement l'application

$$F : x \mapsto \frac{20\,x}{x^2+4} \quad \text{de } [-10;+10] \text{ dans } \mathbb{R},$$

nous aimerions connaître les images par F des éléments contenus dans l'intervalle $[-10;+10]$ en les prenant par pas de 0.5 [16]. En d'autres termes, nous désirons une table des valeurs numériques que prend l'application $y = \frac{20\,x}{x^2+4}$ pour des valeurs de x variant de -10 à $+10$ par saut de 0.5.

x	y
-10.0	.....
-9.5	.....
.....	.....
.....	.....
$+10.0$	.....

L'allure de la courbe est représentée en figure 3.

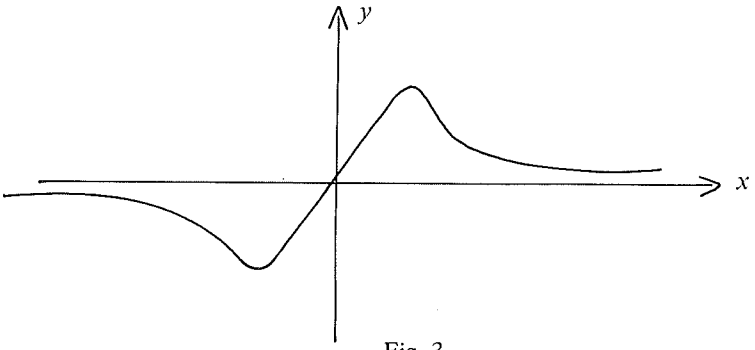


Fig. 3

Ce problème correspond au schéma de la figure 4 (appelé aussi **organigramme**).

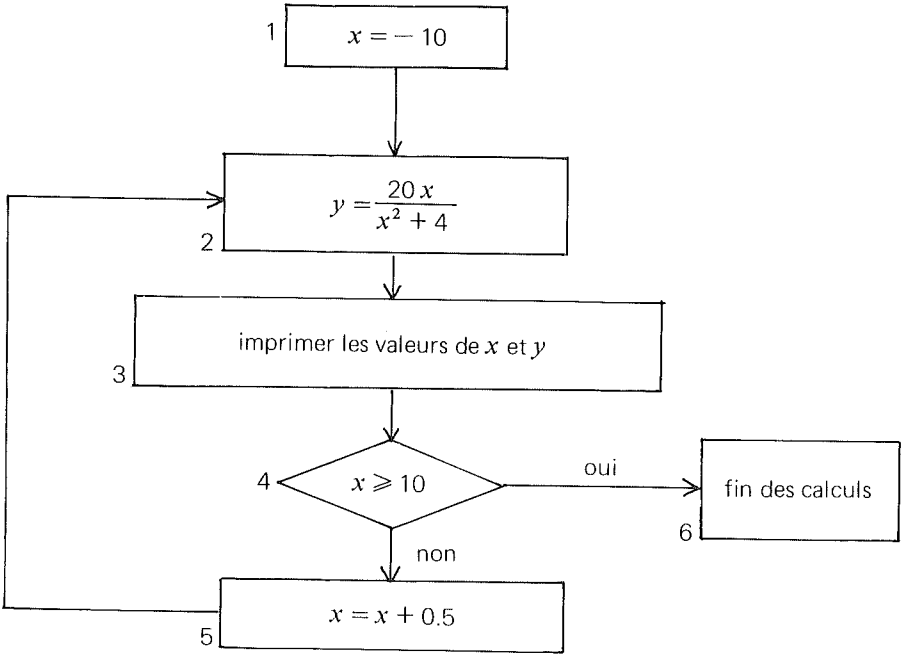


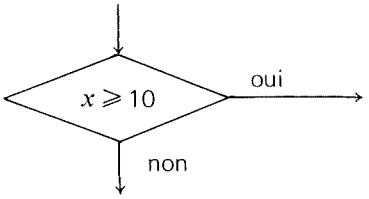
Fig. 4

Deux remarques s'imposent :

a) le losange qui se trouve dans l'organigramme traduit l'idée de comparaison et signifie :

si $x \geqslant 10$, suivre l'embranchement oui

si $x < 10$, suivre l'embranchement non



b) l'expression x = x + 0.5 n'a pas le même sens que dans le cours de mathématique (ce n'est pas une équation). C'est davantage une manière de dire : "augmenter x de la valeur 0.5".

Nous renvoyons le lecteur pour de plus amples explications au chapitre 6 (ordre de replacement).

Le problème présenté sous forme d'organigramme n'est pas assimilable par la machine, aussi convient-il de le transcrire dans un langage de programmation (ici le FORTRAN). (Voir fig. 5).

1	5	6	7 72	73
			PRØGRAM EXMATH	
C			TABULATIØN DE LA FØNCTIØN Y=20X/(X**2+4) DE X=-10 A X=10 PAR	PAS DE 0.5
			X=-10.	
	1		Y=20.*X/(X**2+4.)	
			PRINT 100, X, Y	
			IF (X-10.)2, 3, 3	
	2		X=X+0.5	
			GØTØ 1	
	3		CALL EXIT	
	100		FØRMAT (F 10.1, F 15.6)	
			END	

▲34

Fig. 5

Pour l'instant, il n'est pas question pour le lecteur de comprendre tous les détails de ce programme. Tout au plus lui suffira-t-il de découvrir une certaine analogie entre l'organigramme et le programme.

Ainsi à $y = \frac{20x}{x^2 + 4}$ correspond `Y = 20.*X/(X**2+4.)`

à imprimer les valeurs x et y correspond `PRINT 100, X, Y`

au test $x \geqslant 10$ correspond `IF(X−10.)2, 3, 3`

à $x = x + 0.5$ correspond `X = X + 0.5`

à fin des calculs correspond `CALL EXIT`

enfin à la flèche reliant 5 à 2 correspond `GO TO 1`

Le programme de la figure 5 doit être désormais transcrit sur carte perforée pour être transmis à la machine. D'autres possibilités de transcription sont possibles (bande papier, etc.). A chaque ligne correspond une carte. L'ensemble des cartes constitue le **programme**.

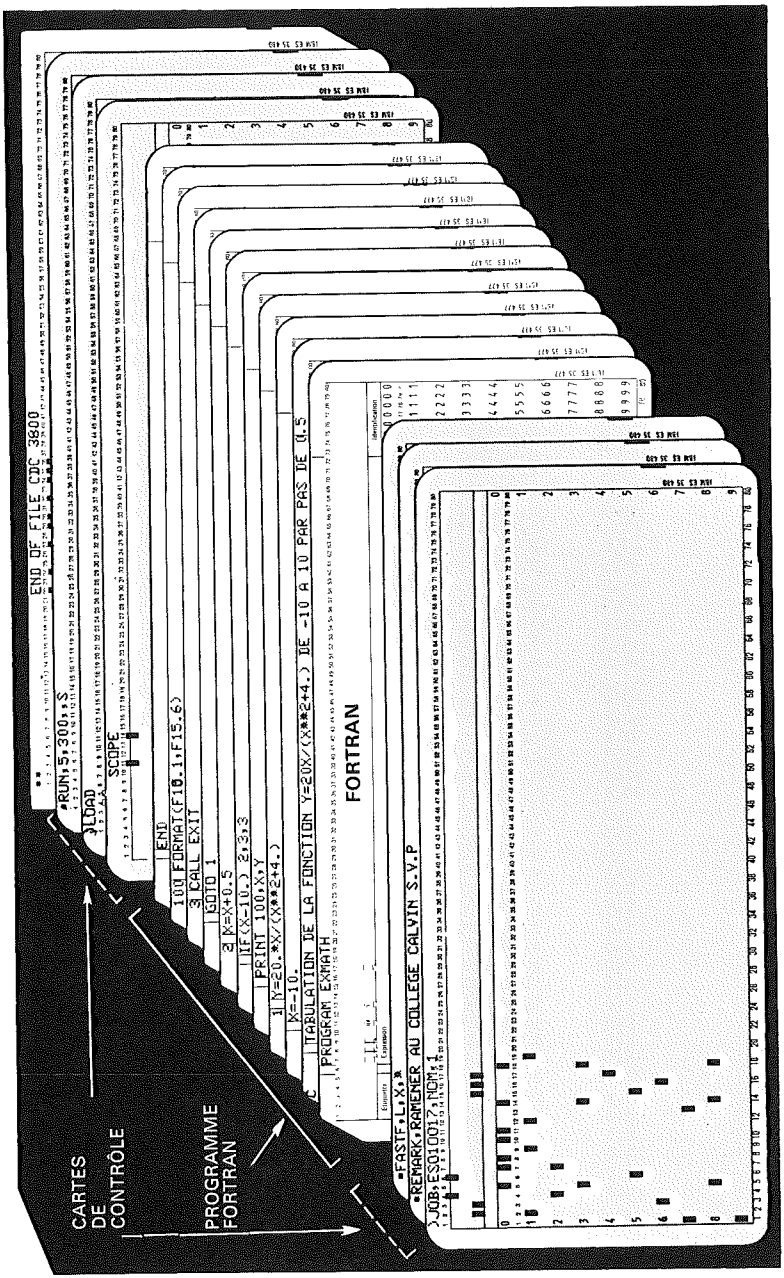
Ce programme, muni d'un certain nombre de cartes de contrôle, s'appelle **deck** et sera lu par le lecteur de cartes.

▲2

Le déroulement des opérations à l'intérieur de la machine se fait en deux phases :

1. traduction du programme en "langage machine" (compilation),
2. chargement du programme, calculs, impression, etc. (exécution).

Les résultats de notre problème se trouvent à la page 14.


CARTES
DE
CONTRÔLE
PROGRAMME
FORTRAN
END OF FILE CDC 3800
FORTRAN
PROGRAM EXMATH
TABULATION DE LA FONCTION Y=20X/(X**2+4.) DE -10 A 10 PAR PAS DE 0.5
FASTF,L,X,
*REMARK,RAMENER AU COLLEGE CALVIN S.V.P.

```
JOB,ES010001,NOM    ,1
**    CCI GENEVE  ***    DISK  SCOPE  2,1 V,3 / 1,    AOUT  71  ****
FASTF,L,X                BEGIN JOB AT   1012 - 37          09/21/71


            PROGRAM EXMATH
      C     TABULATION DE LA FONCTION Y=20X/(X**2+4)
      C     DE X=-10 A X=10 PAR PAS DE 0,5
            X=-10.
          1 Y=20.*X/(X**2+4.)
            PRINT 100,X,Y
            IF(X-10.)2,3,3
          2 X=X+0.5
            GOTO 1
          3 CALL EXIT
        100 FORMAT(F10.1,F15.6)
            END
LOAD
RUN,5,300,,S
EXECUTION STARTED AT   1012 -46

              -10.0     -1.923077
               -9.5     -2.015915
               -9.0     -2.117647
               -8.5     -2.229508
               -8.0     -2.352941
               -7.5     -2.489627
               -7.0     -2.641509
               -6.5     -2.810811
               -6.0     -3.000000
               -5.5     -3.211679
               -5.0     -3.448276
               -4.5     -3.711340
               -4.0     -4.000000
               -3.5     -4.307692
               -3.0     -4.615385
               -2.5     -4.878049
               -2.0     -5.000000
               -1.5     -4.800000
               -1.0     -4.000000
               -0.5     -2.352941
                0.0      0.000000
                0.5      2.352941
                1.0      4.000000
                1.5      4.800000
                2.0      5.000000
                2.5      4.878049
                3.0      4.615385
                3.5      4.307692
                4.0      4.000000
                4.5      3.711340
                5.0      3.448276
                5.5      3.211679
                6.0      3.000000
                6.5      2.810811
                7.0      2.641509
                7.5      2.489627
                8.0      2.352941
                8.5      2.229508
                9.0      2.117647
                9.5      2.015915
               10.0      1.923077
END JOB,NOM,DATE 09/21/71 TIME 1012 - 47 ELAPSED TIME 00 HRS00 MIN 10 SEC
```

A la fin des calculs d'un programme terminé, il est nécessaire d'arrêter la machine (pas physiquement), pour pouvoir traiter le prochain programme; c'est l'ordre CALL EXIT qui remplit cette mission. ▲4

Après l'exécution de l'exemple EXMATH sur un ordinateur travaillant en "batch processing" ou traitement par lots (les "deck" ou paquets de cartes passent l'un après l'autre dans le lecteur de cartes pour être ensuite traités), il est intéressant de reprendre le même problème mais dans un environnement "time-sharing" (temps partagé), cas où l'ordinateur est à la disposition de plusieurs utilisateurs simultanément.

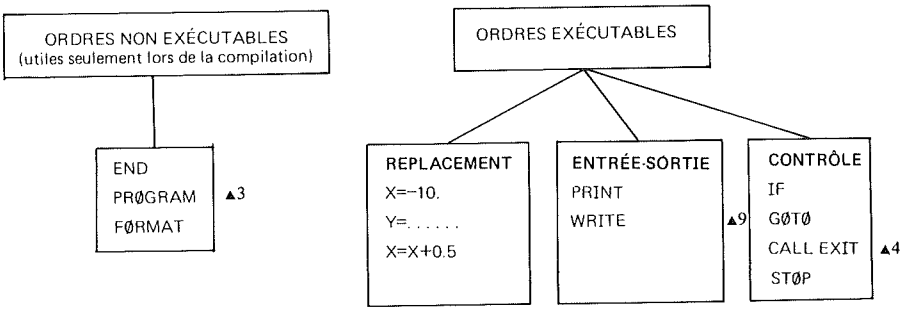
```
                                        Commentaires

PLEASE SIGN ØN                          Début de la liaison avec l'ordinateur
?ID 33MØREL                             Identification de l'utilisateur
H1640 SERIES TIME-SHARING               Heure et date
ØN AT 10 : 14 09/04/72
? TAPE                                  Entrée de la bande de papier
? CREATE EXMATH                         Création du fichier EXMATH, qui
BEGIN                                   correspond au programme FORTRAN
                                        traitant l'exemple
100 * TABULATIØN DE LA FØNCTIØN Y=20X/(X**2+4)
110 * DE X=—10 A X=10 PAR PAS DE 0,5
120 X=—10.
130 1 Y=20.*X/(X**2+4.)
140 WRITE(9,100)X,Y                     Ici le nombre 9 indique que l'impression
150 IF(X—10.)2,3,3                      se fera sur la console
160 2 X=X+0.5
170 GØTØ 1
180 3 STØP                                                          ▲4
190 100 FØRMAT(F10.1,F15.6)
200 END
? TYPE                                  Retour au clavier de la console
? FØRTRAN EXMATH,EXCØMP                 Compilation du fichier source EXMATH
                                        et création du fichier EXCØMP (version
                                        compilée)
```

```
? LØAD EXCØMP                    Chargement de la version compilée EXCØMP
 -10.0        -1.923077
  -9.5        -2.015915          Exécution
  -9.0        -2.117647
  -8.5        -2.229508
  -8.0        -2.352941
  -7.5        -2.489627
  -7.0        -2.641509
                .
                .
                .
                .
                .
                .
   7.5         2.489627
   8.0         2.352941
   8.5         2.229508
   9.0         2.117647
   9.5         2.015915
  10.0         1.923077
STØP,
? ØFF                            Fin de la liaison avec l'ordinateur
OFF AT 10:19 09/04/72
COMPUTE SEC. - 06.5              Nombre de secondes de calcul du processeur
CONNECT MIN. - 05                Nombre de minutes de connection avec l'ordinateur
```

(Les caractères gras sont ceux tapés par l'utilisateur, les autres par l'ordinateur.)

En comparant les deux exécutions du même problème (page 13 et ci-dessus), on trouve une certaine ressemblance entre les cartes de contrôle du "deck" de cartes et les commandes d'un système conversationnel (ici toutes précédées par un point d'interrogation). La différence essentielle réside dans le fait que l'intervention de l'utilisateur, au cas où tout ne se déroulerait pas de manière satisfaisante, n'est possible que dans le cas du système conversationnel (ici un système time-sharing); les cartes de contrôle d'un système d'exploitation "batch" sont évidemment très rigides et contraignantes pour l'utilisateur, sans parler en plus du temps de réponse ou "turn-around" (délai entre la soumission du programme et la remise des résultats) qui est très variable, conséquence des files d'attente inévitables.

Les ordres qui sont intervenus dans notre programme peuvent se classifier de la façon suivante :

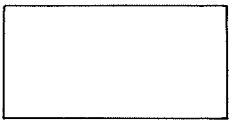


2.2. ORGANIGRAMME

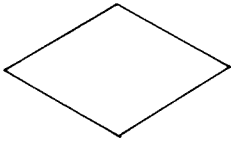
Dans l'exemple introductif, nous avons associé au problème un plan détaillé des opérations. Ce plan s'appelle **organigramme** (en anglais : block diagram ou flow chart). Nous ne saurions trop insister sur l'utilité d'un tel organigramme; c'est en effet un point essentiel de la programmation. Il permet de prévoir une suite d'opérations dans un programme avant de l'avoir écrit.

Faire un organigramme, c'est donc schématiser par un dessin la suite des opérations que doit exécuter l'ordinateur. Cet organigramme aura tout son intérêt dans la mesure où rien ne sera oublié, et où il représentera exactement ce que l'on veut programmer. C'est en quelque sorte la logique du problème. (cf. [8] et [34])

Nous introduisons les notations suivantes pour faire un organigramme



un rectangle représente n'importe quelle opération excepté une décision



un losange représente une décision

Les différents rectangles et losanges d'un schéma sont réunis entre eux par des flèches indiquant l'ordre de succession des opérations.

Remarquons encore que dans certains cas d'erreur dans un programme, il est agréable d'avoir un organigramme à sa disposition.

Exemple 1

Résoudre l'équation du deuxième degré $A x^2 + B x + C = 0$ [16]

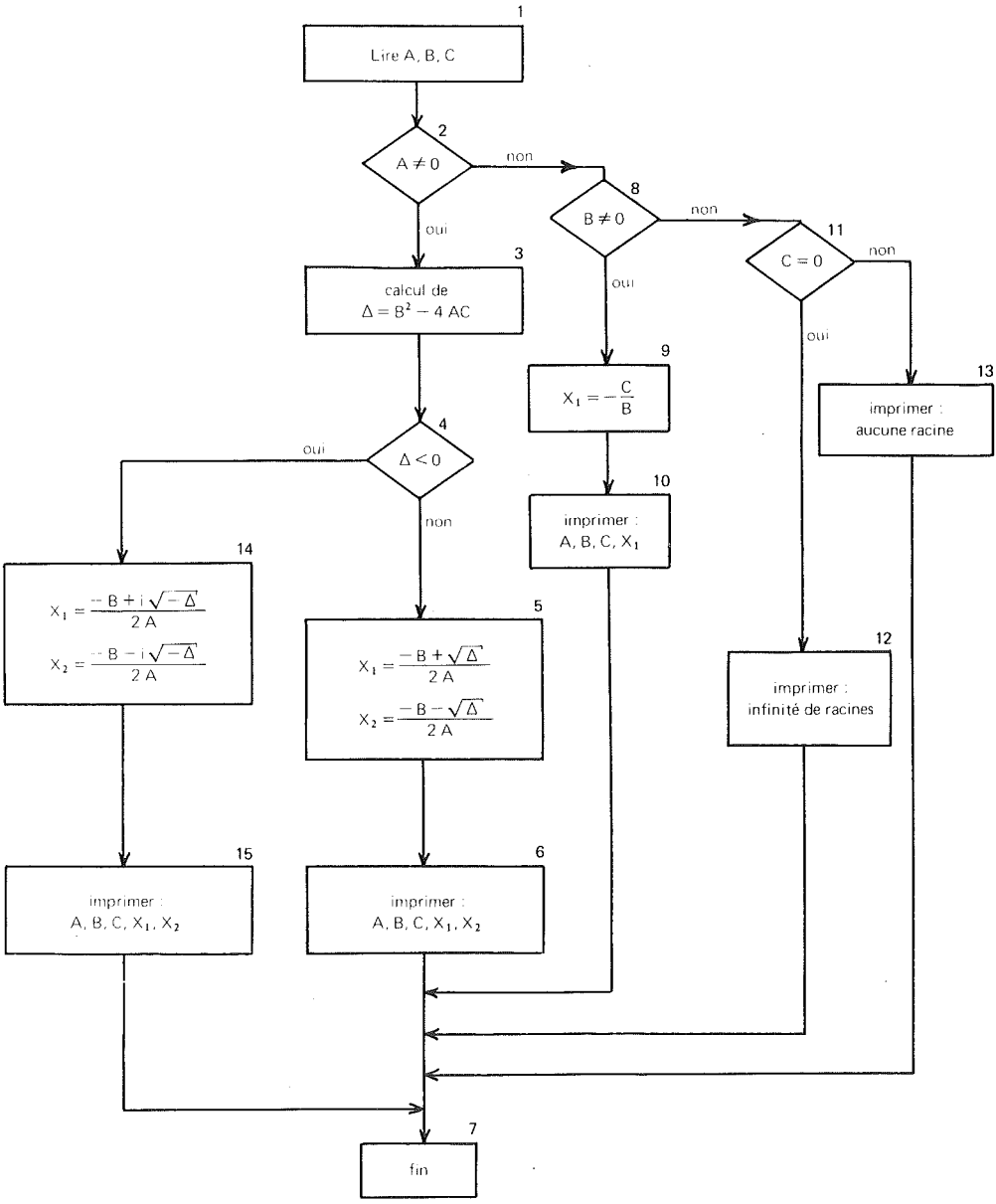
A ce problème correspond l'organigramme ci-contre que nous allons établir ensemble.

Voici les principales opérations :

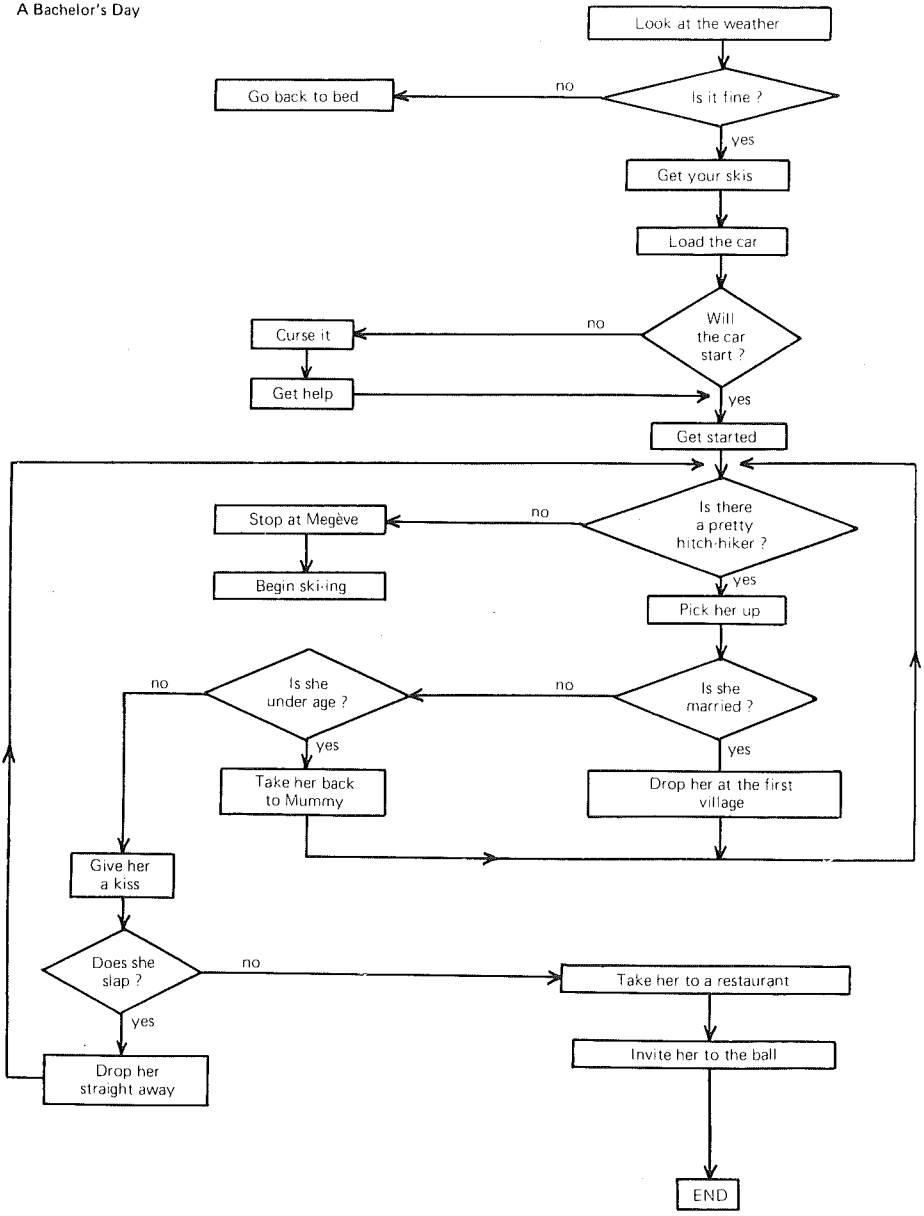
1. Lecture des coefficients de l'équation (A, B, C).
2. Test assurant que nous avons affaire ou non à une équation du deuxième degré.
3. A étant différent de zéro l'équation est du deuxième degré.
   Il faut calculer le discriminant Δ.
4. Test assurant que cette équation a des racines réelles ou non.
5. Δ étant positif ou nul, calcul des deux racines (éventuellement deux racines confondues) de l'équation.
6. Impression des coefficients et des racines de l'équation.
7. Indication que le problème est terminé.
8. A étant nul (ce n'est donc pas une équation du deuxième degré), l'équation est-elle du premier degré ?
9. L'équation étant du premier degré, calcul de sa racine.
10. Impression des coefficients et de la racine de l'équation.
11. A et B étant nuls, C est-il nul ?
12. Les trois coefficients A, B, C sont nuls, il y a une infinité de racines. On imprimera le message : "infinité de racines".
13. $A = 0$, $B = 0$, $C \neq 0$; dans ce cas on imprimera le message : "aucune racine".
14. Δ étant négatif, calcul des racines imaginaires en prenant $\sqrt{-\Delta}$.
15. Impression des coefficients et des deux racines imaginaires de l'équation.

Exemple 2

Les auteurs espèrent que les connaissances du lecteur en anglais seront suffisantes pour apprécier l'organigramme de la page 20 :


1
Lire A, B, C
2
A ≠ 0
non
oui
8
B ≠ 0
non
11
C = 0
non
oui
oui
3
calcul de
Δ = B² − 4 AC
9
X₁ = − C/B
13
imprimer :
aucune racine
4
Δ < 0
oui
non
10
imprimer :
A, B, C, X₁
14
X₁ = (−B + i√−Δ)/2A
X₂ = (−B − i√−Δ)/2A
5
X₁ = (−B + √Δ)/2A
X₂ = (−B − √Δ)/2A
12
imprimer :
infinité de racines
15
imprimer :
A, B, C, X₁, X₂
6
imprimer :
A, B, C, X₁, X₂
7
fin

A Bachelor's Day



Exercices

2.1. On donne l'application $T : x \longmapsto \frac{5}{9}(x - 32)$ qui permet de convertir des degrés Fahrenheit en degrés Celsius. Représenter l'organigramme qui permettra d'obtenir une table de conversion pour tous les degrés Fahrenheit entre 32°F et 120°F.

2.2. Même application qu'à l'exercice 2.1., mais on demande de représenter l'organigramme qui donnera pour certaines valeurs lues en degrés Celsius, les valeurs correspondantes en degrés Fahrenheit.

2.3. Monsieur Taillard est à Genève de passage pour un jour. Pendant ce temps, il décide de rendre visite à son ancien ami Monsieur Calame, mais il ne se souvient pas de son prénom et il a perdu son numéro de téléphone. En revanche, il se rappelle que son ami habite au bord du lac et que dans son adresse, il y avait le nom d'un personnage suisse célèbre.

Dessiner l'organigramme qui permettra à Monsieur Taillard de retrouver son ami Calame en n'utilisant que les moyens suivants :

a) le plan de Genève
b) l'annuaire téléphonique
c) les quelques précisions dont il se souvient.

2.4. Dessiner l'organigramme qui permet de calculer

a) la moyenne arithmétique de 100 nombres
b) n !

2.3. ÉLÉMENTS DU LANGAGE

Ici commence réellement le début de l'étude du langage FORTRAN.

On distingue quatre sortes de caractères :

a) **Les caractères alphabétiques**
Les caractères alphabétiques ne sont formés que des 26 lettres majuscules.

Exemple : A , N , W , I , etc.

b) **Les caractères numériques**
Les caractères numériques sont formés des 10 chiffres du système décimal.

Exemple : 3 , 1 , 0 , 5

c) **Les opérateurs arithmétiques**

Les opérateurs arithmétiques sont :

+	pour l'addition,
*	pour la multiplication,
−	pour la soustraction,
/	pour la division,
**	pour l'exponentiation.

Exemples :

3*5 signifie 3×5 ; 4**3 signifie 4^3 ; 5/3 signifie $5 : 3$

d) **Les caractères spéciaux**

Nous utiliserons un certain nombre de caractères spéciaux : ( ) . , − etc.

Exemple : (2**3)**5 ≠ 2**(3**5)

en effet $(2^3)^5 \neq 2^{(3^5)}$

Remarque :

En FORTRAN la partie décimale d'un nombre réel est précédée d'un point et non d'une virgule (conformément à la notation anglo-saxonne).

2.4. QUELQUES RÈGLES D'USAGE

a) **Usage des parenthèses**

1. X*(−5) est correct
   X*−5 n'est pas correct
2. (A+B) est correct
   (A+B n'est pas correct
3. (X*(Y+Z))−Y*(X−Z)**A*B signifie
   $x(y+z)-y(x-z)^a \cdot b$

On constate donc que l'usage des parenthèses est le même qu'en algèbre.

b) Usage des opérateurs arithmétiques

La lecture des expressions se fait toujours dans l'ordre suivant :		
1.	**	
2.	* ou /	(en lisant de gauche à droite)
3.	+ ou −	(en lisant de gauche à droite)

Ainsi :

A+B/C	signifie	$a + \frac{b}{c}$
A−B**C	signifie	$a - b^c$
(A−B)**C	signifie	$(a-b)^c$
A/B−B/C	signifie	$\frac{a}{b} - \frac{b}{c}$

(((A*X+B)*X+C)*X+D)*X+E

signifie $\{[(a\,x + b)\,x + c]\,x + d\}\;\; x + e$

ou encore $[(a\,x^2 + b\,x + c)\,x + d]\,x + e$

c'est-à-dire $(a\,x^3 + b\,x^2 + c\,x + d)\,x + e$

enfin $a\,x^4 + b\,x^3 + c\,x^2 + d\,x + e$

Attention :

On évitera d'écrire A/B*C qui peut signifier indifféremment $\frac{a}{b} \cdot c$ ou $\frac{a}{b\,c}$

Exercices :

2.5. Traduire en algèbre les expressions suivantes :

a) A+B*C

b) A−(B*C)/D

c) A−B**C

d) (A*B)/C**D/A−B

e) ((A*X+B)*X+C)*X+D

f) A**B/C+D*E*F−G

g) A**B/((C+D)*(E*F−G))

2.6. Ecrire les expressions suivantes en langage FORTRAN :

a) $\dfrac{a+b}{c+d}$

b) $(a-b^c)\,(c-d)^e$

c) $\left(\dfrac{a}{b}-\dfrac{c}{d}\right)^e$

d) $a\,x^2+b\,x+c$

e) $\dfrac{a\,x+b}{c\,x}+\dfrac{d}{e}$

CHAPITRE 3

ÉLÉMENTS DE BASE DU LANGAGE

En langage FORTRAN les quantités numériques sont appelées **constantes** et les quantités littérales sont appelées **variables.**

3.1. CONSTANTES

On distingue 2 sortes de constantes :

a) **les constantes entières** (mode fixe)

Une constante entière est un nombre entier. Sa valeur absolue appartient à l'intervalle

$$[0;\ 2^{47}-1]$$ ▲5

pour l'ordinateur CDC 3800, ce qui correspond donc à une précision de 15 chiffres.

Exemple :	*Contre-exemple :*
+ 19691	22.00
− 17	1012.
0	123456789123456789

b) **les constantes réelles** (mode flottant)

Une constante flottante est formée d'une suite de **10 chiffres significatifs** qui peuvent être séparés par un point. Sa valeur absolue appartient à l'intervalle

$$[0;\ 10^{307}]$$ ▲5

pour l'ordinateur CDC 3800. On peut également utiliser la notation exponentielle si l'on a un exposant entier.

Exemples : + 3.000 3. 03.0
0.
.47814
− 475.671

41.3 signifie 41,3
4.13 E 1 signifie $4{,}13 \cdot 10^{1}$

4.13 E − 1 signifie $4{,}13 \cdot 10^{-1}$
.413 E + 2 signifie $0{,}413 \cdot 10^{2}$

Contre-exemples : 53
17 . 28 E 400
5,25

Remarque :

5 est une constante entière
5. est une constante flottante

3.2. VARIABLES

Les variables sont représentées en langage FORTRAN par des **noms** qui comportent au maximum 6 caractères.

Le premier caractère d'une variable doit être alphabétique, les autres pouvant être des lettres ou des chiffres (caractères alphanumériques). ▲6

Exemples :

RACINE
X
POLYNO
T14
B15121

Contre-exemples :

31416
POLYNOMES
12BAC

On distingue deux sortes de variables :

a) **les variables entières**

Les variables dont le nom commence par I, J, K, L, M, N sont des variables entières.

Exemples :

ISOMME
K24
NPOLYN
J3

b) **les variables flottantes**

Les variables dont le nom ne commence pas par I, J, K, L, M, N sont des variables flottantes.

Exemples :

PROGR
AMAX
BFONCT
X24

Remarque :

En langage FORTRAN, les espaces ne sont pas pris en considération. Ainsi X 24 est interprété comme X24

Exercices :

3.1. A quelles sortes de variables appartiennent les noms suivants ?

OPERA	PRODUI	JASS	SIJKLM
IDIFF	C128	N2	M207PR

3.3. EXPRESSIONS ALGÉBRIQUES

Une expression algébrique peut être une constante, une variable ou une fonction (voir chapitre 4).

On peut obtenir également des expressions en combinant des constantes et des variables.

Cependant, toutes les combinaisons concernant le type de l'expression ne sont pas permises. On respectera scrupuleusement le tableau suivant :

	ENTIER	RÉEL
ENTIER	ENTIER	
RÉEL		RÉEL

▲35

Exemples d'expressions :

1. Type entier

```
I + 5
K * 3
L - 7
16/2
17/3          (résultat 5)
-19/4         (résultat -4)
```

Contre-exemples :

```
J + B
R * 5
3.4 + I
```

2. Type réel

```
A + 5.
S * 4.
(B + C) * 7.5
7.5/2.5       (résultat 3.)
7. /2.        (résultat 3.5)
```

Contre-exemples :

```
3. /5
D * 4
A + 3.14 * I
2 * 5.75
```

L'EXPONENTIATION

Le cas de l'exponentiation doit être traité en prenant soin de bien observer la règle suivante :

		exposant	
		ENTIER	RÉEL
base	ENTIER	ENTIER	RÉEL
	RÉEL	RÉEL	RÉEL*

*cf. remarque 2 p. 29.

▲7

Exemples :

J**7	entier
(B**2.−4.*A*C)**3	réel
B**3	réel

Remarques :

1) L'exposant peut être négatif.

2) L'exponentiation est effectuée par une méthode différente pour une base flottante selon que l'exposant est entier ou flottant :

X**2	x.x
X**2.	2.*lnx

(si x est négatif, le calcul de X**2. entraîne une erreur fatale)

Exercices :

3.2. Souligner les constantes entières

23.4	17	1000	−.5
NAN	−10501	0.0	12345678
JA7	5K00	1/2	274.1

3.5. Souligner les constantes flottantes

−.017	40193.	1.001	B52
.K14	0.	517	1.0
27	2734.4712	S0X	−12

3.3. Souligner les variables entières

5K	BAKER	ITR35	N
NOW	L0ØKOUT	M5M5M5M5M	Y
KING7	M−1	NY	VU

3.4. Souligner les variables flottantes

V8	.5K	DØG	2.4
SUM	IB17	DRØGUE517	X
LTOTAL	M5	A5523	DS21

3.6. Changer les variables suivantes de fixe en flottant ou de flottant en fixe en permutant deux lettres ou en ajoutant au maximum une lettre.

BETA	NUMBER	MØNEY	J5000
X	AJ3	KING62	SLØAD

CHAPITRE 4

FONCTIONS MATHÉMATIQUES

Les fonctions mathématiques peuvent avoir un ou plusieurs arguments. Il convient de relever cependant qu'il n'y a qu'un seul résultat. Le ou les arguments peuvent être des constantes, variables ou expressions algébriques. ▲8

Exemple :

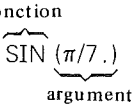


Tableau des principales fonctions mathématiques

Fonction	Nom	Signification	Exemple
Racine carrée	SQRT	$(Arg)^{1/2}$ avec Arg > 0	SQRT(3.)
Sinus	SIN	sin(Arg) Arg en radians	SIN(π/4.)
Cosinus	CØS	cos(Arg) Arg en radians	CØS(3*π/11.)
Logarithme décimal	ALØG10	log(Arg) Arg > 0	ALØG10(3.726)
Valeur absolue	ABS	|Arg|	ABS(−4.7)
Modulo	AMØD	AMØD(Arg 1, Arg 2)	AMØD(23.7) (réponse 2)
Logarithme naturel	ALØG	ln(Arg) Arg > 0	ALØG(2.9)
Exponentielle	EXP	e^{Arg}	EXP(13./4.)
Arc sinus	ASIN	arc sin(Arg)	ASIN(1/7.)
Arc cosinus	ACØS	arc cos(Arg)	ACØS(0.)
Arc tangente	ATAN	arc tg(Arg)	ATAN(1.)

Exercices :

4.1. Transcrire en langage FORTRAN les expressions suivantes :

$\sqrt{15}$	$\lvert -7{,}56\rvert$	Log(0,053)
Modulo(37,6)	$\sin(1/\pi)$	$\cos(3\pi/2)$
ln(17.)	$e^{3{,}45}$	arg tg(2.5)

4.2. Etablir l'organigramme et le programme qui permettent de reconstituer une partie de la table "Voellmy" [35].
(Cet exercice exige la connaissance des ordres d'entrée-sortie, cf. chapitre 5.) ■

CHAPITRE 5

ORDRES D'ENTRÉE-SORTIE

Il convient de distinguer nettement les ordres qui donnent la possibilité de mettre en mémoire l'information contenue sur une carte perforée, sur une bande papier ou magnétique ou sur une zone d'un disque (ordres d'entrée) de ceux qui permettent de restituer l'information contenue dans la mémoire en l'imprimant sur papier (imprimante ou console), en la stockant sur une bande magnétique ou une zone d'un disque, en la perforant sur une carte ou une bande papier, ou éventuellement en l'affichant sur un écran (ordres de sortie).

5.1. ORDRES GÉNÉRAUX (READ et WRITE)

L'ordre de lecture (entrée) est READ. Sa forme générale est :

```
READ (u,n) A,B,C,.....,X,Y,Z
```

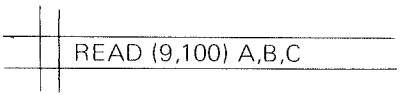
▲9

n est l'étiquette de l'instruction FØRMAT (patron de lecture) qui est liée à l'ordre d'entrée.
A, B, C, ..., X, Y, Z est la liste des variables.
u est une variable ou une constante **entière**, qui donne le numéro d'unité logique associée au périphérique utilisé (console, disque, lecteur de cartes, dérouleur de bande magnétique, etc.).

Remarques :

1. Les numéros d'unités logiques dépendent étroitement de l'installation. ▲25
2. Le cas de lecture, indépendamment d'un FØRMAT, ne sera pas évoqué ici ("mode binaire pur").

Exemple 1 :



signifie lire depuis l'unité logique no 9 les variables A, B, C selon le FØRMAT 100.

L'ordre d'écriture (sortie) est WRITE. Sa forme générale est :

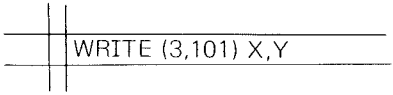
```
WRITE (u,n) A,B,C,.....,X,Y,Z
```

▲9

n est l'étiquette de l'instruction FØRMAT (**patron** d'écriture) qui est liée à l'ordre de sortie.
A, B, C, ..., X, Y, Z est la liste des variables.
u est une variable ou une constante **entière**, qui donne le numéro d'unité logique associée au périphérique utilisé (imprimante, disque, perforateur de cartes, console, dérouleur de bande magnétique, etc.).

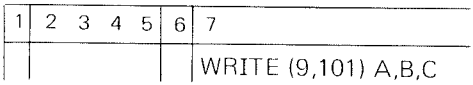
Les deux remarques ci-dessus sont également valables.

Exemple 2 :



signifie écrire sur l'unité logique no 3 les variables X et Y selon le FØRMAT 101.

Exemple 3 :



signifie imprimer selon le FØRMAT 101 les variables A,B,C.

Remarque :

Aux ordres généraux READ et WRITE il convient d'associer des ordres qui ne sont pas exécutables et qui font l'objet du paragraphe 5.2. : les FØRMAT.

Exercice :

5.1. A-t-on le droit d'écrire : WRITE (9,110) A,I,K,S?

5.2. ORDRE NON EXÉCUTABLE FØRMAT (F,I,X,H)

Reprenons l'exemple 1 de la page 34 : READ (9,100) A,B,C. 100 est l'étiquette de l'instruction FØRMAT qui pourra être par exemple :

100 FØRMAT(F3.5,F4.1,F2.1)

Nous verrons plus loin que F3.5,F4.1 et F2.1 donnent des indications sur la façon d'imprimer les trois variables A,B,C.

La forme générale de l'ordre FØRMAT est :

n FØRMAT (C_1 ,C_2 ,C_3 ,.....,C_n)

n est l'étiquette de l'instruction WRITE ou READ

C_1 , C_2 , C_3 , ... , C_n sont appelés codes du FØRMAT.

Exemple :

1	2 3 4 5	6	7
			READ (9,100) A,B,C,K
			FØRMAT (F10.3,F12.3,F14.3,I5)

Nous allons envisager deux types de FØRMAT :

a) le FØRMAT F grâce auquel il est possible de transmettre des nombres réels (on parle alors de "mode flottant");

b) le FØRMAT I grâce auquel il est possible de transmettre des nombres entiers (on parle alors de "mode fixe").

a) FØRMAT F

La forme générale du FØRMAT F est : a F ℓ d

a est une constante entière qui précise le nombre de fois que le code FØRMAT est répété. Si la constante a vaut 1, il n'est pas nécessaire de l'indiquer.

ℓ est une constante entière qui précise la longueur totale du champ (exemple : F 5.1 a une longueur de champ égale à 5).

d est également une constante entière qui précise le nombre total de décimales.

Exemple 1 :

1		6	
	1 0 0		FØRMAT(3FI.d)

est équivalent à :

1		6	
	1 0 0		FØRMAT(FI.d,FI.d,FI.d)

Exemple 2 :

		$\overbrace{\quad\quad\quad\quad}^{l=10}$		
En FØRMAT	F$\underbrace{10}_{l}$.$\underbrace{3}_{d}$	−53412.$\underbrace{714}_{d=3}$	représente	−53412,714
	F8.4	bb1.534b*		1,534
	F10.7	b3.1415926		3,1415926

Remarques :

1. Les symboles − et . comptent pour une position dans le champ.
2. Le symbole + est omis.
3. Il est important de se rappeler la notion de chiffres significatifs (cf. p. 25) et de relire la remarque ▲ 5
   En effet, il serait absurde de prévoir dans un FØRMAT une zone flottante F20.8 pour une machine n'ayant que 6 chiffres significatifs par exemple.

Exemple 3 :

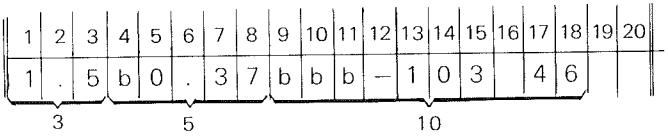
Dans la résolution de l'équation du second degré on est amené à introduire 3 variables A, B, C. Admettons que ces variables aient respectivement les valeurs 1,5 0,37 −103,46.

Si nous avons les deux instructions :

	READ (9,100) A,B,C
1 0 0	FØRMAT (F3.1,F5.2,F10.2)

*b représente un espace en blanc.

la carte de données numériques est :



Si nous avons les deux instructions :

```
      |  READ (9,100) A,B,C
1 0 0 |  FØRMAT (F10.1,F10.2,F10.3)
```

la carte de données numériques est :

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
b	b	b	b	b	b	b	1	.	5	b	b	b	b	b	b	0	.	3	7	b	b	–	1	0	3	.	4	6	b

Exemple 4 :

Soit une variable X ayant pour valeur 3. En écrivant les deux instructions :

```
      |  WRITE (9,103) X
1 0 3 |  FØRMAT (F8.5)
```

l'imprimante écrira : b3.00000

Exercices :

5.2. Sachant que A = 0,0037, B = 1024,5, C = –100,4, D = 1200, remplir une carte numérique sur la base des indications suivantes :

a) FØRMAT(F10.4, 2F10.1, F10.0)

b) FØRMAT(F8.4, F8.1, F9.3, F10.5)

5.3. Quelles sont les valeurs des variables A, B, C, D dans un programme contenant les trois cartes suivantes :

```
      READ (9,100) A,B,C,D
100   FØRMAT(4F5.2)
```

```
b 3 . 1 4 b 1 . 2 4 b 0 . 3 7 b 0 . 1 1
    5         5         5         5
```

b) FØRMAT I

La forme générale du FØRMAT I est : $\boxed{a\,I\,\ell}$

a est une constante entière qui précise le nombre de fois que le code FØRMAT est répété. Si la constante a vaut 1, il n'est pas nécessaire de l'indiquer.

ℓ est une constante entière qui précise la longueur totale du champ (exemple : I3 a une longueur de champ égale à 3).

Exemple 1 :

```
1     6
  100   FØRMAT(3I2)
```

est équivalent à :

```
1     6
  100   FØRMAT(I2,I2,I2)
```

Exemple 2 :

En FØRMAT	I4	b521	représente	521
	I6	bbb−32		−32
	I5	3bbbb		30000
	I5	bbbb3		3
	I8	−4716281		−4716281 ▲5
	I2	−5		−5

Exemple 3 :

Soit une variable K dont la valeur est –3127. En écrivant :

```
      WRITE (9,101) K
101   FØRMAT(I5)
```

nous lisons sur l'imprimante : –3127

Si, par erreur, nous écrivons :

```
      WRITE (9,101) K
101   FØRMAT(I4)
```

l'ordinateur signalera lors de l'impression que le code FØRMAT est trop petit. ▲21

Enfin, si nous écrivons :

```
      WRITE (9,101) K
101   FØRMAT(I10)
```

nous lisons : bbbbb–3127

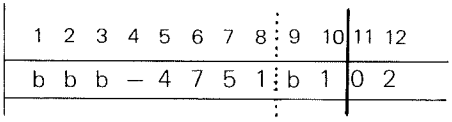
Exemple 4 :

Soit les variables L et M dont les valeurs sont respectivement –4751 et 102

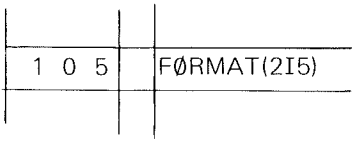
Si nous avons les deux instructions :

```
      READ (9,105) L,M
105   FØRMAT(I8,I4)
```

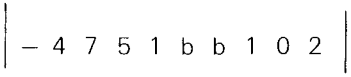
la carte de données numériques devra être



Si le FØRMAT utilisé est :

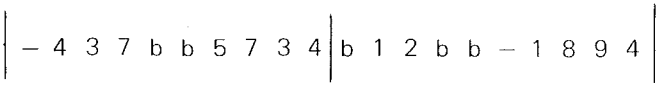


la carte de données numériques sera :

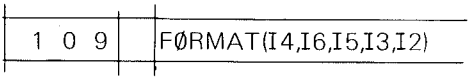


Exercices :

5.4. Soit une carte de données numériques

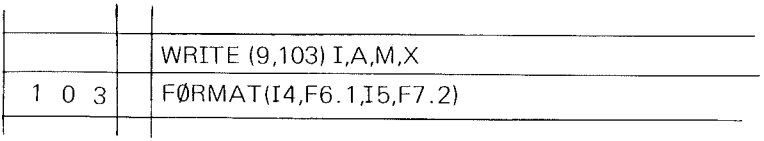


Cette carte est lue par l'instruction READ (9,109) I,J,K,L,M
Le FØRMAT utilisé est :



Quelles valeurs obtiendra-t-on pour I, J, K, L, M ?

5.5. Soient 4 variables I, A, M, X dont les valeurs sont respectivement 432, 32.5, –15, –0.17
Si l'on écrit les deux instructions :



que lira-t-on sur l'imprimante ?

c) **Mise en page**

Pour la mise en page, nous allons utiliser un certain nombre de codes :

1. Code X

Le code X est utilisé pour insérer des espaces blancs consécutifs entre deux zones imprimées sur une même ligne.

La forme générale du code X est : `n X`

n est une constante entière indiquant le nombre d'espaces blancs consécutifs.

Exemple :

Etant donné A = 3.1, B = 52.4 et les deux instructions :

```
      WRITE (9,100) A,B
  100 FØRMAT(1XF3.1,3XF4.1)
```

nous lisons sur l'imprimante : b 3 . 1 b b b 5 2 . 4

Remarque :

En utilisant l'ordre WRITE pour une imprimante il est conseillé de commencer l'impression par 1X au moins de façon à éviter d'avoir une impression en colonne 1.

▲10

2. Code H

Le code H est surtout utilisé pour écrire des titres ou insérer des textes. La forme générale du code H est :

`n H` texte à imprimer (n caractères)

n est une constante entière sans signe.

Les n caractères qui suivent le H sont imprimés textuellement.

Exemple :

```
17HCØLLEGEbDEbFRANCE
7HDELTAb=
```

Remarque :

On peut également utiliser sur certains ordinateurs le code "astérisque" ou le code "apostrophe" qui évite de compter les lettres. ▲11

Exemple :

```
*CØLLEGEbDEbFRANCE*        ou        'CØLLEGEbDEbFRANCE'
*DELTAb=*                            'DELTAb='
```

3. Code /

Le slash (/) stoppe l'impression d'une ligne et reprend l'impression à la ligne suivante. Si l'on désire sauter une ou plusieurs lignes, il suffit d'écrire deux ou plusieurs "slashes" dans un FØRMAT. Ainsi, pour avoir un saut de deux lignes, il faut écrire 3 "slashes" (///).

Exemple :

```
      |   | WRITE (9,100)
1 0 0 |   | FØRMAT(1X17HCØLLEGEbDEbGENEVE/1X17(1H*)////1X14HCØLLEGEbCALVIN//
      | 1 | 1X16HCØLLEGEbVØLTAIRE//1X16HCØLLEGEbRØUSSEAU//1X19HCØLLEGE
      |   | bDEbCANDØLLE)
```

On lit sur l'imprimante :

```
COLLEGE DE GENEVE
**********
.
.
.
COLLEGE CALVIN
.
COLLEGE VOLTAIRE
.
COLLEGE ROUSSEAU
.
COLLEGE DE CANDOLLE
```

Remarque :

Dans les FØRMAT le "slash" sert en même temps de virgule.

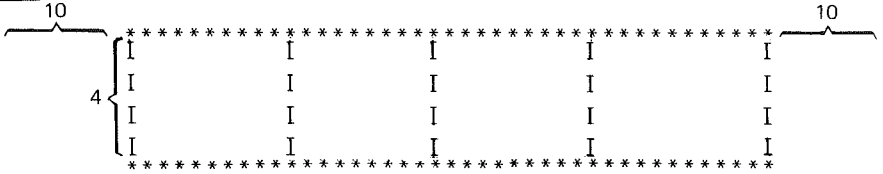
Exercices :

5.6. Donner les instructions qui permettent d'imprimer le titre centré :

PROBLEME DE MOYENNE
****

Indication : la feuille comporte (136–1) positions.

5.7. Donner les instructions qui permettent de réaliser le tableau suivant :



5.8. Compléter l'instruction 100 pour obtenir le texte :

La solution du problème est 3.41

	WRITE (9,100) A
1 0 0	.....

5.9. Compléter l'instruction 101 pour obtenir le texte :

Les racines sont A = 1.28 et B = −5.4415

	WRITE (9,101) A,B
1 0 1	.....

CHAPITRE 6

ORDRE DE REPLACEMENT

(signification du signe "=")

Dans un cours de mathématique, la relation d'égalité entre objets $a = b$ signifie que a et b représentent le même objet dans l'ensemble de base.

Cette relation est symétrique : $a = b \Rightarrow b = a$

En programmation, "=" a un sens très différent. On parle plus volontiers **d'ordre de replacement** qui est défini de la manière suivante :

```
V = E
```

où V est une variable
E est une expression arithmétique ou logique. (cf. chap. 11)

Il est bien entendu que si E est une expression arithmétique, V doit être une variable arithmétique; de même si E est une expression logique, V sera une variable logique.

Cette instruction de replacement signifie :

> l'expression située à droite du signe "=" est évaluée numériquement et sa valeur est rangée dans la mémoire de l'ordinateur affectée à la variable située à gauche du signe "=".

Exemples :

1. I = I + 1

   ou plus généralement

2. X = X + Y

   Cette instruction de replacement représente une sommation successive (fréquemment utilisée dans le système du compteur). La valeur de X, majorée de celle de Y est rangée dans la mémoire correspondant à la variable X. Elle y remplace la précédente valeur de X.

3. A = B–D/F
4. DIST=RATE*TIME
5. I = J + K

   On ne peut pas définir C comme suit :

1. BE + C = A*D
2. AX*XI + C = 72.

   Il faudra donc écrire :

1. C = A*D – BE
2. C = 72.– (AX*XI)

Remarques :

1. Le calcul dans le membre de droite n'est réalisable que si toutes les variables de l'expression (arithmétiques ou logiques) ont été préalablement définies, soit issues d'une précédente instruction de replacement, soit comme "donnée" provenant d'un ordre READ. L'expression "précédente instruction de replacement" ne signifie pas forcément que l'instruction est écrite avant celle qui est envisagée dans le programme.

Exemple :

```
      |     |  |
      |     |  | A=7.8
      |     |  | READ (9,100) BJ
      | 100 |  | FØRMAT (F10.2)
      |     |  | Y=A*BJ+BJ**3
      |     |  |
```

2. Très souvent le signe "=" sera encadré par des expressions du même mode, c'est-à-dire membres de gauche et de droite en fixe ou en flottant.

Cependant, si nous écrivons :

a) B = J, la valeur de B est remplacée par la valeur de J convertie en flottant,

Exemple :

```
      J=7
      B=J
```

d'où B = 7.0

b) J = B, la valeur de J est remplacée par la valeur de la partie entière de B,

Exemple :

```
      B=9.6
      J=B
```

d'où J = 9

Ce genre d'écriture permet donc :

a) de convertir en virgule flottante un nombre entier

b) de prendre la partie entière d'un nombre positif exprimé en virgule flottante.

3. Le sens mathématique du signe "=" est donné en FORTRAN par ".EQ." (equal).

Exemple :

A.EQ.B signifie A = B au sens mathématique.

A.EQ.B est une expression logique. (cf. chap. 11)

Exercices :

6.1. Ecrire en FORTRAN les expressions algébriques suivantes :

a) $x = a + b + c - 3$

b) $S = \dfrac{ar - ar^n}{1 - r}$

c) $x = ar^{n-1}$

d) $w = 3(x + y)$

e) $x = \dfrac{1}{100}(z^3 + y)^{-3}$

f) $a = \begin{cases} \dfrac{n+3}{n-2} & \text{si } n \neq 2 \text{ et } n \text{ pair} \\ \dfrac{n-3}{n+2} & \text{si } n \neq -2 \text{ et } n \text{ impair} \end{cases}$

$n \in \mathbb{Z}_+^*$

g) $s = 2\,pr \sin(\pi/r)$

h) $h = 3\sqrt{x + (5\,y^2/2)}$

6.2. Ecrire l'équivalent en formules mathématiques des expressions FORTRAN suivantes :

a) 6.*B**5

b) X-Y-(Z-W)

c) ((X-Y)*(Y-Z))**2

d) Q=X/(Y*Z)

e) A=17*M/I-J/3

f) Q=(Y+X*X)/(B+E*A)

g) BETA=-2./(3.*A)+X**4/(5.*A**3)

h) Y=X*(Y**2-X**2)/(X**2+Y**2)

i) FC=1.615*D*Q1*Q2/(Q1-S2)

j) J=6*K+4*I1*M2

k) I=I+1

l) M=12

m) PI=3.1415927

n) N=2*M+18*I

6.3. Trouver dans les instructions suivantes au moins une erreur :

a) Y=3.X+B

b) 3.14=B-A

c) C=((X+Y)B**3+(P-Q)**2

d) Y=1,624009.*DEL

e) -K=I**2.

f) BX6=1./-2.*B**2

g) DERIVE=N*X**(N-1)

h) +W=A+B-C

i) 4=I*7.

j) V-3.96=Y**2.74

k) Y=(B+6)**3

l) K9=J**B

6.4. Montrer que X=Y n'est pas égal à Y=X en FORTRAN en exécutant les deux fragments de programme ci-dessous.

```
      X=3.1
      Y=5.2
      X=Y
```

a) Que vaut X ?

```
      X=3.1
      Y=5.2
      Y=X
```

b) Que vaut Y ?

6.5. Quelle valeur est attribuée à X après l'exécution des deux ordres suivants :

```
      A=2.2
      A=A+0.4
```

6.6. Donner la valeur de A, B... J etc. en indiquant si elle est en fixe ou en flottant.

a) A=7*4+2
b) A=2/7
c) J =2/6
d) B=2.*(12./8.)
e) K=2.*(12./8.)
f) B=2*(12/8)
g) J=2*(12/8)
h) K=2*(20/8)
i) C=3*(20/8)
j) A=1./3.+1./3.+1./3.
k) Z=(16.0)**(6./4.)
l) C=(8.0)**6./4.
m) B=8**(6/4)
n) K=8**(6/4)
o) M=18/4+7/4

6.7. La somme d'une progression arithmétique est donnée par $S = \frac{n}{2}(2a + (n-1)r)$ où S = somme; n = nombre de termes; a = premier terme; r = raison.

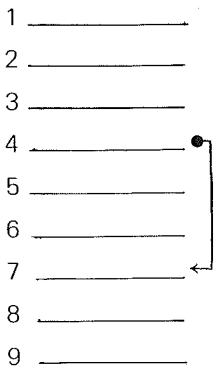
Ecrire l'instruction FORTRAN qui permet de trouver la somme des cent termes de la progression arithmétique dont le premier terme est 7.9 et la raison DELTA.

CHAPITRE 7

ORDRES DE CONTRÔLE

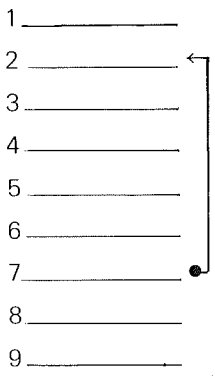
7.1. INTRODUCTION

Les ordres de contrôle arrêtent l'exécution séquentielle des instructions du programme; ils donnent la possibilité, soit d'avancer et d'éviter ainsi une séquence (fig. 1), soit de reculer et de répéter une suite d'instructions (fig. 2).



évitement des instructions 5 et 6

Fig. 1

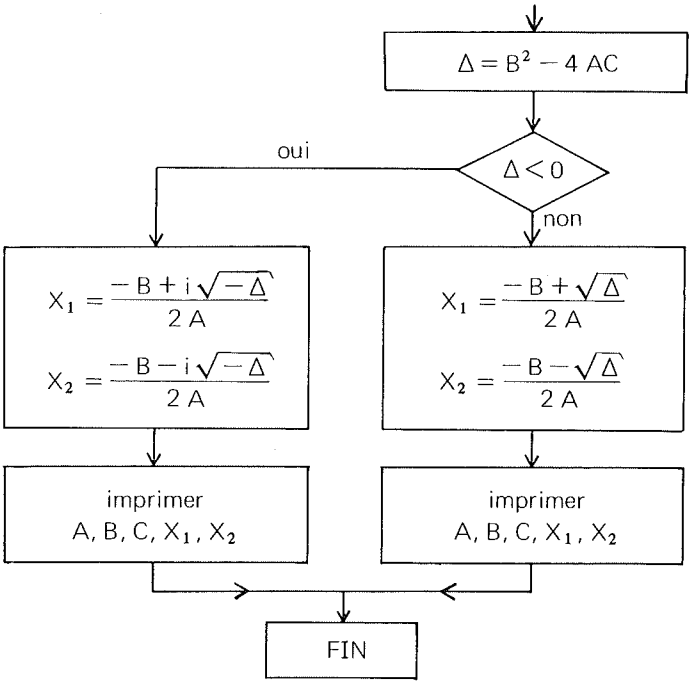


répétition des instructions 2 à 7

Fig. 2

Exemple :

Reprenons une partie de l'organigramme du problème de la résolution de l'équation du deuxième degré:



Si $\Delta \geqslant 0$, nous avons des racines réelles. Si $\Delta < 0$, nous avons des racines imaginaires. Suivant la valeur de Δ (idée de test), les racines seront réelles ou imaginaires. Si nous suivons l'un des cheminements de l'organigramme, il n'est pas question de parcourir l'autre. Mais les instructions qui décrivent ces deux possibilités sont rangées séquentiellement. Il importe, après avoir choisi un chemin, de ne pas emprunter l'autre. Une rupture de séquence **inconditionnelle** permet de réaliser ce projet; la rupture de séquence **conditionnelle** donne la possibilité de décider du chemin à suivre selon la réponse donnée à une condition (ici le signe du discriminant).

De cette manière, nous avons abordé intuitivement deux des trois types principaux d'ordres de contrôle :

a) rupture de séquence inconditionnelle (GØTØ)

b) rupture de séquence conditionnelle (IF) ou ordre de comparaison (test)

c) ordres pour opérer des boucles (cf. chap. 8).

Ces différentes instructions sont appelées aussi **ordres de transfert**. Un autre exemple qui montre bien le rôle des ruptures de séquence est celui du compteur (cf. programme EXMATH).

7.2. NUMÉRO D'INSTRUCTION (statement identifier)

Dans presque tous les exemples présentés jusqu'ici, les instructions étaient exécutées séquentiellement, d'où l'inutilité de reconnaître les différentes instructions, la succession des opérations allant d'elle-même. Mais maintenant, avec les ruptures de séquence, il s'avère nécessaire de pouvoir repérer un ordre FORTRAN, c'est ce que permet le numéro d'ordre (numéro d'instruction ou étiquette).

Le numéro d'ordre obéit aux règles suivantes :

1. c'est un entier sans signe
2. il est compris entre 1 et 99999 ▲17
3. il est écrit et perforé dans les colonnes 1 à 5 pour les cartes perforées
4. les espaces laissés en blanc et les zéros à gauche sont ignorés
   (ainsi : 132, 00132, b1b3b2 représentent le même numéro d'ordre).

Remarques :

a) Un numéro d'instruction n'est pas accepté par la machine si un caractère différent de zéro est perforé dans la sixième colonne, en d'autres termes on ne peut pas numéroter une carte de continuation.

b) Les numéros d'ordre ne doivent pas nécessairement être dans l'ordre croissant.

c) Nous avons déjà rencontré des numéros d'ordre pour les formats.

Exemple :

		READ (9,102) A
	102	FØRMAT (F10.2)

Convention :

Pour séparer nettement les formats des autres instructions, nous numérotons les formats à partir de 100. Il s'agit d'un conseil qui n'a aucun caractère impératif.

Exemples de numéros d'ordre :

132		
9 24		
1		
274		
294		
32767		

Contre exemples :

K		
+72		
0		
− 7		

7.3. RUPTURES DE SÉQUENCE INCONDITIONNELLES (GØTØ)

Ce sont des ordres qui interrompent l'exécution séquentielle du programme en un point précis de celui-ci sans répondre à une condition (cf. fig. 1 et 2, page 49).

7.3.1. DÉBRANCHEMENT INCONDITIONNEL

```
GØTØ n
```

où n est le numéro d'ordre qui existe effectivement dans le programme avant ou après le GØTØ considéré. L'exécution du programme continuera à l'instruction indiquée par l'étiquette n.

Exemple :

```
        GØTØ72
71      XA=BC+CD
        GØTØ48
72      XA=BC−CD
        SX=SQRT(XA)
        GØTØ49
48      SX=SQRT(XA)
49      W=SX*SX
```

Contre-exemples :

1.

70	GØTØ70

2.

	S=(A+B)/2.
	GØTØ24
	GØTØ94

GØTØ94 ne sera jamais exécuté car il est précédé d'un ordre GØTØ et ne porte pas lui-même d'étiquette.

7.3.2. DÉBRANCHEMENT CALCULÉ OU A PLUSIEURS VOIES

GØTØ (n_1, n_2, n_3, ..., n_m), E

où $n_1, n_2, \ldots, n_m$ sont des numéros d'ordre

E est une expression arithmétique qui sera réduite à une valeur entière i.

si $i \leqslant 1$ alors nous aurons le transfert à n_1

si $i \geqslant m$ alors nous aurons le transfert à n_m

si $i = 2$ (respectivement 3, 4, ..., m), alors nous aurons le transfert à n_2 (respectivement $n_3, \ldots, n_m$) ▲13

Exemples :

1.

	I=3
	GØTØ(4,37,248,574,22),I

Ici I=3 implique que l'ordre exécuté sera celui qui a l'étiquette 248.

2.

	GØTØ(11,4,204),KLM

ce qui signifie que si : KLM $\leqslant$ 1 nous continuerons en 11

KLM = 2 nous continuerons en 4

KLM $\geqslant$ 3 nous continuerons en 204

Nous avons parlé de débranchement calculé, car la variable E doit être définie par :

a) un ordre de lecture

```
      READ (9,101) KLM
```

b) par un ordre de replacement

Exemples :

```
      KLM=2
```

ou

```
      KLM=(6*I-KLM)/3
```

Remarque :

En général l'expression arithmétique E dans GØTØE est souvent une variable entière; voici cependant un exemple (cf. [1] où E doit être réduit comme variable entière :

```
      AA=1
      BB=2
      CC=1
      GØTØ(10,20,30),AA*BB-CC
      .
      .
      .
      .
      .
10    AA=AA+1
      GØTØ(11,21,32),AA*BB-CC
```

Le contrôle sera transmis en 32 pour la suite de l'exécution.

Voir exercice *7.19.* (polynômes de Legendre).

7.4. RUPTURES DE SÉQUENCE CONDITIONNELLES IF

Ces instructions donnent la possibilité de faire des tests arithmétiques ou logiques. L'opération qui suivra ce test dépend du résultat de celui-ci.

7.4.1. TEST ARITHMÉTIQUE

IF (expression ou variable) n_1, n_2, n_3

où l'expression (ou la variable) est **arithmétique**

n_1, n_2, n_3 sont trois numéros d'ordre

Cet ordre IF signifie : aller en n_1 si l'expression est négative

aller en n_2 si l'expression est nulle

aller en n_3 si l'expression est positive

Exemples :

1. Soient à calculer dans $\mathbb{C}$ les racines de l'équation $ax^2 + bx + c = 0$.

L'instruction

```
      IF(B**2-4.*A*C)4,7,2
```

conduira à partir de l'instruction 4 à calculer les racines imaginaires,

conduira à partir de l'instruction 7 à calculer la racine réelle,

conduira à partir de l'instruction 2 à calculer les deux racines réelles.

2.

```
      IF(INDICE-7)2,9,6
```

où INDICE est une variable entière, signifie :

- continuer en 2 si INDICE $<$ 7
- continuer en 9 si INDICE $=$ 7
- continuer en 6 si INDICE $>$ 7

Remarques :

1. Dans un IF arithmétique les **trois** numéros d'ordre doivent toujours être présents même si un des branchements n'est jamais parcouru.

```
      |  IF (ABS(X*X-1.)) 13,13,52
```

2. Comme l'illustre l'exemple ci-dessus, deux des trois numéros d'ordre peuvent être les mêmes.

7.4.2. TESTS LOGIQUES

Le lecteur a intérêt à ignorer en première lecture la fin de ce paragraphe et à le reprendre après le chapitre 11.

a) Test logique à deux embranchements

```
IF(L)n1, n2
```

▲13

où L est une expression **logique**,

n_1, n_2 sont des numéros d'ordre.

Cet ordre signifie :

- aller en n_1 si L est vrai (non zéro)
- aller en n_2 si L est fausse (zéro)

Exemples :

```
      |  .
      |  .
      |  .
      |  IF(A.GT.B.ØR.I.EQ.0)5,10
      |  IF(L)1,2          (L est de type LØGICAL)
      |  .
      |  .
      |  IF(A*B-C)1,2      (A*B-C est de type arithmétique)
```

b) Test logique à un embranchement

```
IF(L) s
```

où L est une expression **logique**,

s est n'importe quelle **instruction** (sauf DØ, un autre IF à un embranchement, END, FØRMAT).

Cet ordre signifie :

- si L est vraie (non zéro) exécuter l'instruction s puis continuer séquentiellement (à moins que s ne soit un GØTØ),
- si L est fausse (zéro) ignorer l'instruction s et poursuivre séquentiellement.

Exemples :

1.

```
      .
      .
      .
      IF (X.EQ.Y) WRITE (9,110) X
      X=X+Y
      .
      .
      .
```

permet, si X = Y, l'impression de X suivant le FØRMAT 110, puis l'exécution de la suite des instructions (c'est-à-dire X = X + Y);

si X ≠ Y, l'exécution de la suite des instructions et l'ordre WRITE est dans ce cas ignoré (c'est-à-dire X = X + Y).

2.

```
      .
      .
      .
      IF(ALPHA.GE.BETA)GØTØ6
      ALPHA=PI-ALPHA
    6 SINA=SIN(ALPHA)
      .
      .
      .
```

Contre-exemple :

```
      |  |
      |  | IF(A.EQ.B)IF(A.EQ.C)
```

n'a pas de sens.

Remarque :

Lorsque nous faisons suivre un IF logique à un embranchement d'un IF arithmétique, nous obtenons un aiguillage à quatre directions.

Exemple :

Si $X = -Y$ aller à 18
si $X = Y$ aller à 72
si $X < Y$ aller à 16
si $X > Y$ aller à 6

sera traité de la manière suivante :

```
      |  |  .
      |  |  .
      |  |  .
      |  | IF(X.NE.(-Y))IF(X-Y)16,72,6
   18 |  | suite du programme
```

7.5. COMMENT TERMINER LA LECTURE DE DONNÉES DANS UN PROGRAMME EN UTILISANT L'INSTRUCTION IF ?

Après avoir étudié les instructions GØTØ et IF, nous proposons deux manières (parmi d'autres) de terminer un programme.

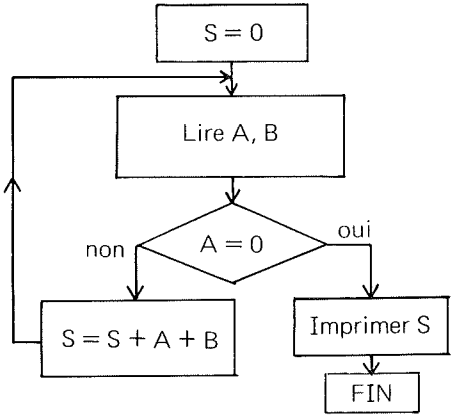
Problème :

Lire des données sur plusieurs cartes (par exemple deux nombres par carte) mais sans en connaître leur nombre, et additionner ces nombres.

1ère méthode :

Nous ajoutons une carte blanche avant la carte de contrôle END ØF FILE

Organigramme :



Programme :

```
      |  S = 0
    3 |  READ (9,100) A,B,
      |  IF(A)1,2,1
    1 |  S = S + A + B
      |  GØTØ3
    2 |  WRITE (9,100) S
      |  STØP
  100 |  FØRMAT (2F10.4)
      |  END
```

Le test permet de savoir si la carte lue est blanche; si oui la lecture est terminée.

Remarques :

1. Nous pourrions finir aussi avec une autre valeur, par exemple 99999. Cela donnerait :

```
      |  IF(A–99999.)1,2,1
      |  ou
      |  IF(A.EQ.99999.)GØTØ 2
```

2. Au lieu de

```
      |  IF(A)1,2,1
      |  . . .
    2 |  CALL EXIT
```

nous aurions pu tester la fin avec

```
      |  IF(A.EQ.0.)GØTØ 2
```

3. **Le lecteur devra se méfier des tests d'égalité avec des variables flottantes.** Il est souvent préférable d'avoir recours aux compteurs ou de faire des tests sur des variables entières.

```
                PROGRAM TEST                                    ▲34
                A=1.
                WRITE (61,100) A,A
                X=0.
          1     X=X+0.1
                WRITE (61,100) X,X
                IF(X-A)1,2,3
          2     WRITE (61,101)
                GØTØ 4
          3     WRITE (61,100) X,X
          4     CALL EXIT
        100     FØRMAT(5XF5.1,5X016)
        101     FØRMAT(*C.Q.F.D.*)
                END
```

Le programme met bien en évidence le problème d'arrondis dans la représentation des mots machine.
La 1ère colonne représente le nombre en écriture décimale.
La 2e colonne représente son équivalent en ØCTAL.

Nombre exact généré par le programme :

```
1.0     2001400000000000 ←
0.1     1774631463146315
0.2     1775631463146315
0.3     1776463146314632
0.4     1776631463146315
0.5     2000400000000000
0.6     2000463146314632   ——— différence
0.7     2000546314631464
0.8     2000631463146316
0.9     2000714631463150
1.0     2001400000000001
```

Nombre obtenu après addition de 10 fois 0.1

```
1.0     2001400000000001 ←
```

Il ressort de cet exemple que le développement octal de la fraction 0.1 étant illimité, on n'obtient pas la valeur 1.0 après avoir additionné 10 fois de suite 0.1 (comparaison faite d'octal à octal). De ce fait, si l'on effectue un test arithmétique du type de celui qui est fait dans le programme ci-dessus, on ne se branchera jamais à l'ordre 2.

2e méthode

La dernière carte du travail est la carte END ØF FILE, qui suit immédiatement la fin des données. Dès lors il apparaît plus commode de tester la fin physique du paquet de cartes qu'un nombre fixé arbitrairement. C'est ce que permet l'instruction

IF(EØF,u) n_1 . n_2	▲14

où : u est le numéro d'une unité logique (60 pour le lecteur de cartes)

n_1 , n_2 sont deux étiquettes.

Cette instruction signifie :

si l'on a rencontré un EØF sur l'unité logique u, aller en n_1, sinon aller en n_2.

Exemple :

Test de la fin du programme exécutant la résolution de l'équation du deuxième degré :

		PRØGRAM DEUSIØ
13		READ (60,100) A,B,C ▲34
		IF(EØF,60)99,31
31		IF(A)...,...,...
		...
		...
		GØTØ13
99		CALL EXIT
		END

Exercices :

7.1. Ecrire un programme qui permette de calculer la somme de n nombres donnés. (n est quelconque.) ■

7.2. Ecrire un programme qui permette de calculer la moyenne arithmétique de n nombres donnés (n quelconque). ■

7.3. Ecrire un programme qui permette de calculer la moyenne géométrique de n nombres donnés (n quelconque). ■

7.4. Ecrire un programme qui permette de résoudre l'équation du second degré en envisageant tous les cas prévus dans l'organigramme de la page 19. ■

Applications

Résoudre les équations suivantes :

a) $x^2 - 5x + 6 = 0$

b) $x^2 - 9x + 20{,}25 = 0$

c) $3x^2 - 11x - 18 = 0$

d) $-2{,}7x^2 - 0{,}64x + 30{,}9 = 0$

e) $37{,}9x^2 - 12{,}14 = 0$

f) $17{,}9x - 18{,}12 = 0$

g) $1237x^2 + 5x + 12 = 0$

h) $19x^2 + 5x + 12 = 0$

7.5. Déterminer la valeur du nombre π (rappel : $\pi = 3{,}14159265358979323846\ldots$) en envisageant les différentes méthodes suivantes : ▲5

a) Calcul de π par la formule de Wallis (modifiée)
Convergence rapide – 17 décimales exactes après 23 itérations.

$$\pi = 6\left(\frac{1}{2} + \frac{1}{(2).3.2^3} + \frac{1.3}{(2.4).5.2^5} + \frac{1.3.5}{(2.4.6).7.2^7} + \ldots\right)$$

b) Calcul de π par la formule alternée d'Euler
Moins bonne convergence – 11 décimales exactes après 20 itérations

$$\pi = 2\sqrt{3}\left(1 - \frac{1}{3.3} + \frac{1}{5.3^2} - \frac{1}{7.3^3} + \frac{1}{9.3^4} - \ldots\right)$$

c) Calcul de π par la formule de Bernoulli

$$\pi = \sqrt{6\left(1 + \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \ldots\right)}$$

d) Calcul de π par la méthode de la tangente
Convergence très lente

$$\pi = 4\left(\frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots\right)$$

1. en prenant 100 termes
2. en arrêtant les calculs lorsque la valeur absolue d'un terme de la série est inférieur à 10^{-3}

e) Calcul de π par la formule de Viète

$$\frac{2}{\pi} = \frac{\sqrt{2}}{2} \cdot \frac{\sqrt{2+\sqrt{2}}}{2} \cdot \frac{\sqrt{2+\sqrt{2+\sqrt{2}}}}{2} \cdot \ldots$$

f) Calcul de π au moyen de fractions approchées

1. $\pi = \frac{22}{7}$

2. $\pi = \frac{355}{113}$

3. $\pi = \frac{167}{80} + \frac{\sqrt{10}}{3}$

g) Calcul de π par la formule de Wallis

$$\pi = 2\left(\frac{2.2}{1.3} \cdot \frac{4.4}{3.5} \cdot \frac{6.6}{5.7} \cdot \frac{8.8}{7.9} \cdot \ldots\right)$$

h) Autre formule

$$\pi = 2\left(1 + \frac{1}{3} + \frac{1.2}{3.5} + \frac{1.2.3}{3.5.7} + \ldots\right)$$

■

7.6. Déterminer la valeur du nombre e (rappel : e = 2,718281828459...) en envisageant les deux méthodes suivantes : ▲5

a) Calcul de e par la limite d'une suite
Convergence moyenne

Il s'agit de la définition mathématique de e qui fut donnée par Neper et calculée par le binôme de Newton.

$$e = \lim_{n \to \infty}\left(1 + \frac{1}{n}\right)^n$$

b) Calcul de e par un développement en série
Bonne convergence – 17 décimales exactes après 20 itérations.
Il s'agit du développement de e^x avec $x = 1$

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \ldots$$

Pour les exercices *7.7.* à *7.18.* écrire chaque fois le fragment de programme (séquence d'instructions) que nécessite le problème posé.

7.7. Placer dans la variable GRAND la plus grande des deux variables X et Y (dans le cas où X = Y prendre soit X soit Y) sans utiliser la fonction spéciale AMAX1(X_1, X_2, ...).

7.8. Placer dans la variable XMAX la plus grande des trois variables X, Y et Z (toujours sans employer AMAX1, mais en utilisant deux fois l'instruction IF). Indication : comparer d'abord X et Y et placer temporairement le plus grand dans une variable auxiliaire, puis comparer cette nouvelle valeur avec Z pour trouver la plus grande des trois.

7.9. X et Y étant des variables qui peuvent avoir des valeurs négatives ou positives, placer dans XMAX celle qui a la plus grande valeur absolue. Indication : utiliser ABS(X).

7.10. Un angle appelé ALPHA est donné; il est positif et plus petit que 40 radians. Soustraire 2 π (un tour) de ALPHA autant de fois qu'il est possible pour le réduire à un angle strictement positif plus petit que 2 π. On désire que le résultat de l'angle réduit reste dans la variable ALPHA.

7.11. Soient Y_1, Y_2 et Y_3 trois variables. Si $Y_2 > Y_1$ et $Y_2 > Y_3$, aller à l'instruction 6, sinon aller à l'instruction 7.

7.12. Si XNØM est plus petit que 7 en valeur absolue, aller en 80; si XNØM est plus grand ou égal à 7 en valeur absolue, aller en 71. Trouver trois méthodes différentes :

a) avec la fonction ABS(X)
b) sans la fonction ABS(X), mais avec un IF logique
c) sans la fonction ABS(X), mais avec un IF arithmétique

7.13. Si A≤X≤B aller en 43, sinon aller en 96. Trouver deux méthodes différentes :

a) sans utilisation de ABS
b) avec utilisation de ABS

7.14. Si K=2 et P<Q aller en 17
Si K=2 et P≥Q aller en 19
Si K≠2 aller en 69

a) ne donner la séquence qu'avec des IF arithmétiques

b) facultatif : essayer avec des IF logiques

7.15. Si L=3,6 ou 9 aller en 15
Si L=3 ou 18 aller en 35
Si L=4,5 ou 29 aller en 47
Si L∉ 1; 2; 3; 4; 5; 6; 7; 8 aller en 9999

7.16. Calculer $y = 16.7x + 9.2x^2 - 1.02x^3$ pour x variant de 1.0 à 9.9 par pas de 0.1; imprimer x et y pour chacune des nonante valeurs de x.
(**attention !** cf. remarque 3, p. 60).

7.17. Exécuter l'ordre 79 si X∈[a; b] et l'ordre 42 dans le cas contraire.

■

7.18. Calculer et imprimer $y = \sqrt{1 + x^2} + \cos 2x$ pour un certain nombre de valeurs de x en progression arithmétique. Lire d'abord trois nombres XDEBUT, XPAS et XFIN

où XDEBUT sera la première valeur de X
XFIN sera la dernière valeur de X
XPAS étant le pas d'accroissement (raison de la progression arithmétique)

Donner le programme complet ainsi que l'organigramme.

■

7.19. Donner la séquence d'instructions qui permet de calculer l'un des cinq premiers polynômes de Legendre définis mathématiquement ainsi :

$$P_0(x) = 1$$
$$P_1(x) = x$$
$$P_2(x) = \frac{3}{2}x^2 - \frac{1}{2}$$
$$P_3(x) = \frac{5}{2}x^3 - \frac{3}{2}x$$
$$P_4(x) = \frac{35}{8}x^4 - \frac{15}{4}x^2 + \frac{3}{8}$$

x et l'indice du polynôme désiré sont supposés déjà donnés dans le programme. (notation : X pour x et LEGI pour l'indice).

■

7.20. Ecrire un programme pour rechercher le plus grand nombre et le plus petit nombre d'une suite de n nombres (n quelconque).

7.21. Ecrire un programme pour calculer les factorielles de 1 à n. Suivant la valeur de n utiliser la double précision. (cf. chap. 10)

CHAPITRE 8

BOUCLES DØ

8.1. INTRODUCTION

Dans de nombreux programmes nous avons déjà employé la notion de compteur ou de boucle. Nous avons toujours utilisé une variable comme indice et une instruction IF (arithmétique ou logique) pour tester si la boucle était terminée.

Exemples :

a)

```
      |  IK=15
   12 |  IK=IK+5
      |  WRITE (9,100) IK          impression de IK entre 20
      |  IF(IK-117)12,25,25        et 120 par pas de 5.
   25 |  ...
```

b)

```
      |  X=+10.
    1 |  Y=X*X+3.*X-7.             calcul de x² + 3x − 7 = y
      |  WRITE (9,100) X,Y         et impression de x et y
      |  X=X+2.                    pour x entre 10 et 20 par
      |  IF(X-20.)1,1,18           pas de 2
   18 |
```

c)

```
      INDEX=0
1     INDEX=INDEX+1
      READ (60,100) A,B,C
      IF(EØF,60)99,2
2     WRITE (61,101) A,B,C
      IF(INDEX-10)1,5,5
5     .
      .
      .
      .
99    CALL EXIT
```

▲34

lecture de dix cartes et impression (chaque carte ayant les valeurs de A,B,C)

Une des notions les plus importantes et les plus intéressantes du FORTRAN est certainement l'instruction DØ (boucle DØ en anglais DØ looping).

8.2. INSTRUCTION DØ

Cette instruction permet la répétition d'une séquence d'instructions un certain nombre de fois, avec à la fin de chaque itération (boucle) un indice qui varie. Elle se présente sous la forme suivante :

DØ n i = m_1 , m_2 , m_3

où n est un numéro d'instruction qui marque la dernière instruction de la séquence à répéter (fin de la boucle)

i est une variable entière **simple**, non dimensionnée (cf. chapitre 9)

m_1 , m_2 , m_3 sont des constantes non nulles entières et sans signe ou des variables entières simples et $m_3 > 0$.

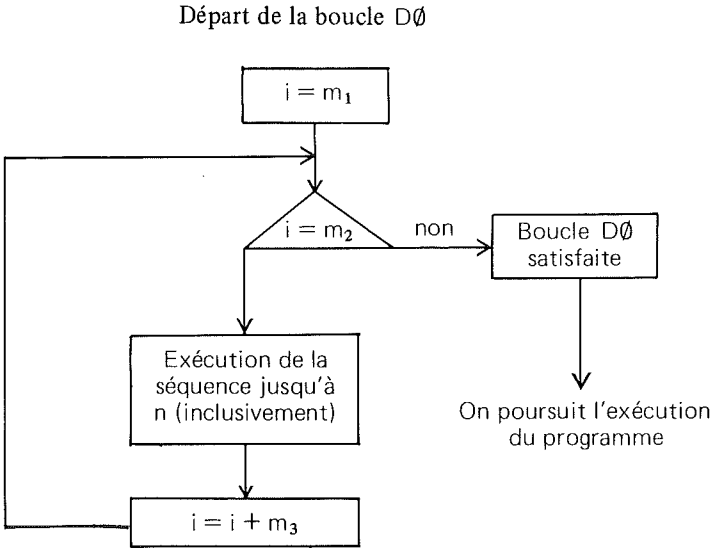
▲15

La boucle DØ est constituée par l'instruction DØ, de l'instruction avec l'étiquette n et de toutes les autres instructions intermédiaires éventuelles.

Cette instruction signifie que :

- la valeur initiale pour i est m_1
- m_2 est la plus grande valeur que pourra prendre i
- m_3 est la quantité qui incrémente i après chaque exécution de la boucle DØ (m_3 est donc le pas).

L'instruction DØ n i = m_1 , m_2 , m_3 est équivalente à l'organigramme suivant :



Exemples :

1) avec indice et IF

```
      IK=15
12    IK=IK+5
      WRITE (9,100) IK
      IF(IK-117)12,25,25
25    ....
```

avec boucle DØ

```
      DØ 19 JK=20,120,5
19    WRITE (9,100) JK
```

} boucle DØ

2) calculer le produit des dix premiers termes d'une progression arithmétique ayant pour premier terme 2 et comme raison 3

avec indice et IF

```
      .
      .
      .
      P=1.
      I=2
1     X=I
      P=P*X
      I=I+3
      IF(I-29)1,1,29
29    ....
      ....
```

avec boucle DØ

```
      .
      .
      .
      .
      .
      .
      P=1.
      DØ 1 I=2,29,3
      X=I
1     P=P*X
      .
      .
      .
      .
      .
      .
      .
```

8.3. REMARQUES SUR LA BOUCLE DØ

1. **si m_3 égale 1**, on peut l'omettre.

2. **dès que i dépasse m_2**, la boucle s'arrête et le contrôle passe à l'instruction qui suit immédiatement l'instruction n.

```
      ....
      DØ 77 JJ=1,50,2
      .
      .
      .
77    ....
```

   Cette boucle se terminera après JJ=49

3. le nombre de fois que les ordres inclus dans la boucle DØ seront exécutés est

$$PE\left(\frac{m_2 - m_1}{m_3}\right) + 1$$

Rappel : PE signifie partie entière.

4. **si $m_1 > m_2$ à l'entrée de la boucle**, la boucle n'est pas exécutée et le contrôle passe à l'instruction qui suit n

▲16

De 15 le programme ira à l'instruction qui suit 14 (la boucle DØ est ignorée).

5. **l'instruction portant l'étiquette n** ne doit pas être :

   a) GØTØ

   b) FØRMAT

   c) une autre boucle DØ

   d) IF (sauf IF(L)s cf. chapitre 7, pour autant que s ne soit pas un GØTØ).

   Pour remédier à cette situation nous introduisons l'ordre fictif "do-nothing instruction" CØNTINUE qui passe le contrôle à l'instruction suivante.

   La séquence suivante, **interdite** :

```
      DØ 12 IND=1,10
      READ (60,100) A,B,C
12    IF (EØF,60)99,2
 2    ....
```

▲14

   est corrigée de la manière suivante :

```
      DØ 12 IND=1,10
      READ (60,100) A,B,C
      IF (EØF,60)99,12
12    CØNTINUE
      ....
```

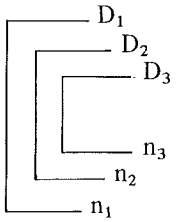
▲14

6. **si la variable i doit être utilisée hors de la boucle** DØ, elle sera préalablement redéfinie à l'intérieur de la boucle.

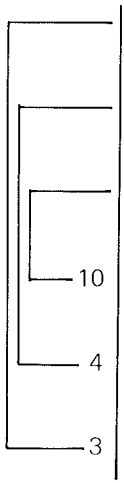
```
      DØ 15 I=1,200
----> K=I
      READ (9,100) A,B,C
      IF (A.EQ.10.)GØTØ19
15    CØNTINUE
      WRITE (9,101) K
101   FØRMAT ('IL N Y A PAS DE CARTE AVEC A=10')
      .
      .
      .
      .
19    WRITE (9,102) K
102   FØRMAT ('LA',I5,'IEME CARTE EST AVEC A=10')
      .
      .
      .
      .
```

7. **notion de** DØ **en cascades** (DØ nests)

A l'intérieur d'une boucle DØ peuvent figurer d'autres ordres DØ.
Exemple schématique : trois boucles en cascades.



Exemple :

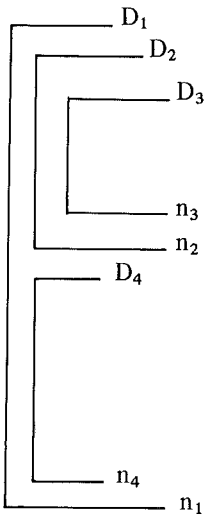


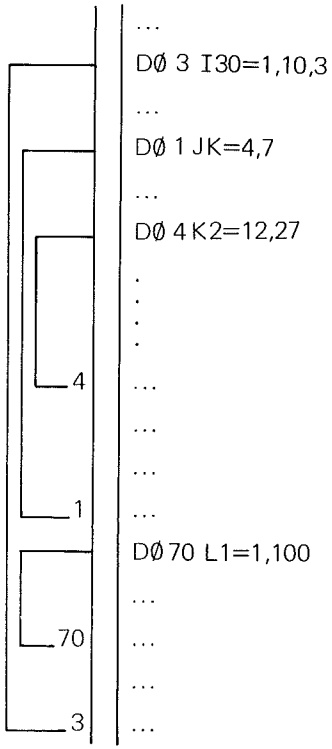

```
DØ 3 I=1,M2
...
DØ 4 J=3,K9,I2
...
DØ 10 K=17,48,11
...
...
...
...
...
...
```

Si D_1 (respectivement D_2 et D_3) est l'instruction DØ de la 1ère (respectivement 2e et 3e) boucle DØ et n_1 (respectivement n_2 et n_3) est la limite de la 1ère (2e et 3e) boucle, il faut que n_3 apparaisse avant (ou coïncide avec) n_2, que n_2 apparaisse avant (ou coïncide avec) n_1.

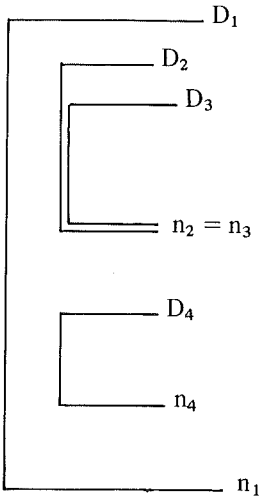
Exemples :

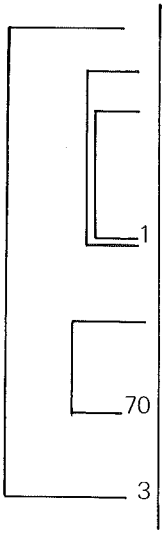
1.





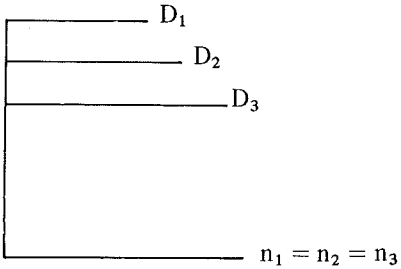
2.

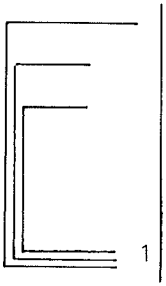





```
DØ 3 I30=1,10,3
DØ 1 JK=4,7
DØ 1 K2=12,27
...
...
...
...
DØ 70 L1=1,100
...
...
...
...
```

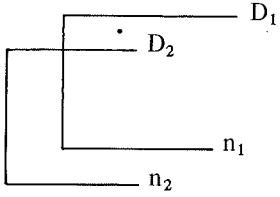
3.

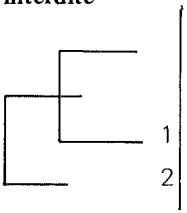





```
DØ 1 J=1,4
DØ 1 J=2,5
DØ 1 IJ=5,19
.
.
.
.
...
```

4. *Contre-exemple :* la situation suivante est **interdite**



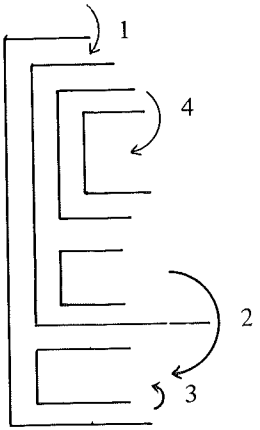



```
DØ 1 I=1,10
DØ 2 J=1,15
...
...
```

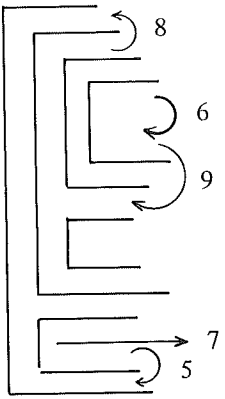
8. **sortie hors de la boucle et entrée dans la boucle**

sortie : Un transfert hors de la boucle est permis à tout moment.

entrée : Une entrée dans une autre boucle est interdite sauf dans le cas de DØ en cascades si le transfert se fait en dehors d'une boucle interne.

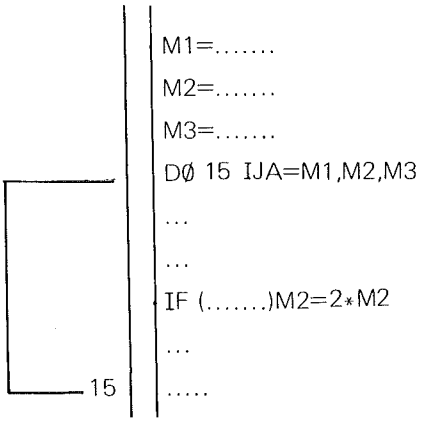


1, 2, 3, 4
sont des entrées
ou sorties non
autorisées.



5, 6, 7, 8, 9
sont des entrées ou
sorties autorisées.

9. si m_1, m_2, m_3 sont des variables alors
   a) m_1 et m_2 peuvent être positives, négatives ou nulles
   b) m_3 doit être positive
   c) m_2 et m_3 peuvent varier pendant l'exécution ▲16



10. on peut mettre jusqu'à 50 DØ en cascades. ▲16

11. d'autres subtilités se trouvent dans [1] et [20].

12. nous reviendrons sur les boucles DØ au chapitre 9 pour les ordres d'entrée-sortie de variables dimensionnées ("DØ - implying segments").

Exercices :

8.1. Reprendre EXMATH avec une boucle DØ (l'indice de la boucle doit être en mode entier). ■

8.2. Reprendre l'exercice moyenne de n nombres avec une boucle DØ. ■

8.3. Tabuler la fonction $F : x \rightarrow x^2 - 28\,x + 447$ pour x variant de 1 à 29 par pas de 2 (avec et sans boucle DØ). ■

8.4. Tabuler la fonction $F(x, y, z) = e^x - 2\,y\,z^2$,
pour x variant de 2 à 3 par pas de 0.1
pour y variant de 10 à 12 par pas de 1
pour z variant de 1,2 à 1,95 par pas de 0.25. ■

8.5. Que va-t-il se passer lorsque l'ordinateur lira la séquence d'instructions suivante :

```
      |  DØ 10 I=4.8
      |  . . .
      |  . . .
10    |  CØNTINUE
```

Rappel : le compilateur FORTRAN

- ignore les espaces laissés en blanc
- permet l'utilisation de tous les noms de variables.

Vérifier votre réponse sur un exemple simple.

CHAPITRE 9

VARIABLES DIMENSIONNEES

9.1. INTRODUCTION

Soient deux matrices A et B telles que $A = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$ et $B = \begin{bmatrix} \lambda & \mu \\ \nu & \sigma \end{bmatrix}$.

Leur produit A · B est :

$$\begin{bmatrix} \lambda\alpha + \beta\nu & \alpha\mu + \beta\sigma \\ \gamma\lambda + \delta\nu & \gamma\mu + \delta\sigma \end{bmatrix}$$

On peut programmer cette multiplication de la façon suivante :

a) Lecture des matrices A et B, c'est-à-dire lecture des huit coefficients (ALPHA, BETA, ..., SIGMA)

b) Calcul des quatre coefficients de la matrice produit :

$\alpha\lambda + \beta\nu$

⋮

.

c) Impression

Remarques :

1. Cet essai bien que correct est très peu rationnel. En effet, le produit matriciel est l'exemple type d'emploi des boucles DØ, mais notre mauvaise notation nous en interdit son utilisation.
2. Ce qui est encore possible avec des matrices de type $< 2; 2 >$ est impensable avec des matrices de type $< 50; 50 >$ par exemple.

3. Le changement de notation

$$\left(A = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} \longrightarrow A = \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} \quad \begin{bmatrix} \alpha_{ij} \end{bmatrix} \right)$$

s'impose également ici pour les noms de variables. Il est en effet intéressant de grouper un certain nombre de variables sous un même nom. Pour les personnaliser on les utilisera sous forme indicée.

9.2. VARIABLES INDICÉES (dimensionnées) (suscripted variables)

Une variable indicée est représentée en FORTRAN par le nom de la variable suivi de un, deux ou trois indices entre parenthèses.

Exemples :

1. β_{32} → BETA (3,2)
2. i_{ij} → I (I,J)
3. C_{10} → C (10)
4. ϵ_{ijk} → EPS (I,J,K)

Remarques :

1. S'il y a plus de trois indices, alors apparaît un diagnostic du compilateur.
2. Les variables dimensionnées peuvent être entières, réelles, en double précision, logiques ou complexes (analogie avec les variables simples).

9.3. FORMES POSSIBLES DES INDICES

Les indices sont de la forme c*I±d,

où I est une variable **simple entière** définie précédemment,

c, d deux constantes entières **sans signe**.

Exemples :

```
AA (I,J)
IDEE (IVER+4,JP+23,14*KC+7)
GRAF (17)
SAM (JUNIØR-6)
BIG (2,LUKE)
```

Contre-exemples :

```
AA (SIN)
IDEE (J2+KQ)
GRAF (4.*MIL*LE+19.7)
SAM (7,-15,18.)
BIG (II(KK))
```

Remarques :

1. La valeur d'un indice doit être toujours plus grande ou égale à 1.

Exemple :

SAM (JUNIØR−6) implique que JUNIØR $\geqslant 7$

Contre-exemples :

GRAF (0) GRAF (−15) GRAF (I−7) si I $\leqslant 7$

2. Le nom d'une variable ne peut pas être employé simultanément comme variable simple et indicée dans le même programme. BIG et BIG (18) par exemple, sont proscrits.
   L'utilisation de BIG et BIG 18 dans le même programme est tout à fait correcte.

9.4. **ORDRE** "DIMENSIØN"

Soit la matrice $A = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \end{bmatrix}$ de type $< 2; 3 >$

Les coefficients $[\alpha_{ij}]$ de cette matrice sont "traduits" par le tableau :

A (1,1)	A (1,2)	A (1,3)
A (2,1)	A (2,2)	A (2,3)

Les caractéristiques (taille et nom) de ce bloc sont déterminées par le nombre d'indices, la valeur maximale de chaque indice et le nom de la variable,

ici A (2,3)

Ces caractéristiques doivent être annoncées au début du programme à l'ordinateur pour réserver à chaque variable dimensionnée la place nécessaire.

C'est l'ordre DIMENSIØN qui permet ces déclarations; il a la forme suivante :

DIMENSIØN $V_1, V_2, \ldots, V_n$

où $V_1, V_2, \ldots, V_n$ sont des variables dimensionnées à un, deux ou trois indices qui sont des constantes **entières**.

Exemples :

a)

```
      DIMENSIØN A(2,3)
```

(pour l'exemple précédent)

b)

```
      DIMENSIØN A(40),D(20,30),E(10),C(3,5,10)
```

permet d'utiliser les variables

a_i $1 \leqslant i \leqslant 40$

d_{jk} $1 \leqslant j \leqslant 20$ et $1 \leqslant k \leqslant 30$

e_ℓ $1 \leqslant \ell \leqslant 10$

c_{mnp} $1 \leqslant m \leqslant 3$ et $1 \leqslant n \leqslant 5$ et $1 \leqslant p \leqslant 10$

c) lecture des six coefficients de la matrice A(2,3) (un coefficient par carte ou par ligne)

```
      DIMENSIØN A(2,3)
      DØ 1 I=1,2
      DØ 1 J=1,3
    1 READ (9,100) A(I,J)
```

Remarques :

1. L'ordre DIMENSIØN est un ordre non exécutable; il précède donc le premier ordre exécutable du programme.
2. Les indices des variables $V_1, V_2, \ldots, V_n$ de l'ordre DIMENSIØN peuvent être des variables entières dans un sous-programme (cf. chap. 12).
3. Certains ordres de déclarations comme LØGICAL, REAL, INTEGER, etc., peuvent jouer le rôle de l'ordre DIMENSIØN.

Ainsi

```
...
DIMENSIØN LUK(30,42),TRUC(10)
LØGICAL LUK,L1,TRUC
...
```

est équivalent à

```
LØGICAL LUK(30,42),L1,TRUC(10)
```

4. Il arrive que dans un programme se trouvent plusieurs ordres DIMENSIØN; cependant chaque variable dimensionnée n'apparaîtra que dans une instruction DIMENSIØN.
5. Le nombre maximum d'éléments pour un tableau dépend de la configuration du système et de l'occupation mémoire (cf. memory map [34]) ▲27

9.5. EXEMPLE DE REPRÉSENTATION INTERNE DE L'ORDRE "DIMENSIØN"

L'instruction

```
DIMENSIØN A(3),B(2,3),C(2,3,2)
```

est représentée en mémoire

par :

	A(1)	A(2)	A(3)	B(1,1)	...

pour une variable unidimensionnelle (vecteur);

par :

...	A(3)	B(1,1)	B(2,1)	B(1,2)	B(2,2)	B(1,3)	B(2,3)	C(1,1,1)	...

pour une variable bidimensionnelle (matrice);

par :

...	B(2,3)	C(1,1,1)	C(2,1,1)	C(1,2,1)	C(2,2,1)	C(1,1,1,)	
C(2,3,1)	C(1,1,2)	C(2,1,2)	C(1,2,2)	C(2,2,2)	C(1,3,2)		
C(2,3,2)	...						

pour une variable tridimensionnelle.

Les blocs sont donc enregistrés en mémoire en ayant le premier de leurs indices qui varie le plus rapidement et le dernier qui varie le moins rapidement.

Remarques :

1. Les valeurs des indices ne doivent jamais dépasser celles attribuées dans l'instruction DIMENSIØN, sinon des erreurs se présenteraient à l'exécution sans avoir été détectées par la compilation.
2. Si l'on écrit DIMENSIØN JAM(500) alors que l'indice de la variable varie de 1 à 100, on a ainsi occupé inutilement 400 mots (situation qui est à éviter aussi souvent que possible).
   (Un **mot** est une unité de mémoire permettant par exèmple de stocker des variables ▲5 et cf. p. 87.)
3. Si l'on écrit DIMENSIØN JAM(200), alors que l'indice INDEX de la variable JAM varie de 171 à 200, on a un gaspillage de 170 mots. Il serait préférable d'écrire :

   NEW=INDEX−170

   et de dimensionner JAM(NEW) à 30. ▲28

9.6. TRANSMISSION DE VARIABLES INDICÉES DANS LES ORDRES D'ENTRÉE-SORTIE (DØ implying segment)

Dans les ordres d'entrée-sortie (READ, WRITE), une ou plusieurs boucles DØ en cascades peuvent être écrites en une seule instruction.

Exemple :

A la place de :

a)

```
      |   | DØ 1 I=1,3
      |   | DØ 1 J=1,3
      | 1 | READ (9,100) A (I,J)
```

on peut écrire :

b)

```
      |   | DØ 1 I=1,3
      | 1 | READ (9,101) (A(I,J),J=1,3)
```

ou encore :

c)

```
      |   |
      | . | READ (9,102) ((A(I,J),J=1,3), I=1,3)
      |   |
```

Remarque :

En ce qui concerne le FØRMAT nous avons :

```
dans a)   100   FØRMAT (F10.2)
```
lecture de un A(I,J) par carte ou par ligne

```
dans b)   101   FØRMAT (3F10.2)
```
lecture des A(I,J) par carte ou par ligne

```
dans c)   102   FØRMAT (8F10.2)
```
lecture des neuf A(I,J) par carte ou par ligne

Comme on ne peut dépasser les 80 colonnes sur une carte, il faut deux cartes de données (une entièrement remplie et l'autre avec la neuvième valeur de A(I,J)). ▲29

Autres exemples :

1.
```
        WRITE (9,100) (A,K=1,10)
```
est équivalent à
```
        WRITE (9,100) A,A,A,A,A,A,A,A,A,A
```
2.
```
        READ (9,100) (A(I),I=1,5,2)
```
est équivalent à
```
        READ (9,100) A(1),A(3),A(5)
```
3.
```
        WRITE (9,100) (B(J),C,A(J),J=1,2)
```
est équivalent à
```
        WRITE (9,100) B(1),C,A(1),B(2),C,A(2)
```
4.
```
        READ (9,100) (CAT,DØG,RAT,I=1,3)
```
est équivalent à
```
        READ (9,100) CAT,DØG,RAT,CAT,DØG,RAT,CAT,DØG,RAT
```

Remarques :

1. On peut emboîter jusqu'à 10 boucles DØ en une seule instruction. ▲16
2. Dans le cas de la transmission complète d'un tableau, on peut négliger les indices et écrire :

```
      ...
      DIMENSIØN A(3,4)
      ...
      READ (9,102) A
```

qui est équivalent à :

```
      DØ 1 J=1,4
      DØ 1 I=1,3
1     READ (9,100) A(I,J)
```

ou encore à :

```
      READ (9,102) ((A(I,J),I=1,3),J=1,4)
```

La transmission de cette matrice A(I,J) se fait **par colonne**.

Exercices :

9.1. Quelle sera la réaction de l'ordinateur devant la séquence suivante :

```
      ...
      DIMENSIØN A(800,9)
      ...
      WRITE (9,100) A
      ...
```

■

9.2. Est-ce que la séquence suivante est correcte ?

```
          ...
          DIMENSIØN NUM (500),A(18)
          ...
17        X=DEN(I)
          ...
          ...
```

(Un message d'erreur montrera que l'ordinateur a cherché en vain la fonction DEN, car la variable DEN n'a pas été dimensionnée.) ▲30

9.3. Ecrire un programme qui stocke N nombres en mémoire avant d'en calculer la moyenne arithmétique.

CHAPITRE 10

FØRMAT A, R, E, D

10.1. FØRMAT A

Le FØRMAT A est utilisé pour imprimer ou lire des textes. La forme générale du FØRMAT A est

aAw

où w est une constante entière sans signe indiquant la longueur du champ. ▲31

a est une constante entière sans signe indiquant le nombre de fois que le code doit être répété. Si a = 1 on peut l'omettre.

1. ENTRÉES

Les w caractères lus sur une carte seront mémorisés comme suit :

a) si w < 8 ▲31

Les w caractères sont transcrits dans la partie gauche du registre et des blancs le complètent à droite.

Exemple :

		READ (60,100) A,B
		. . . . .
100		FØRMAT (A4,A6)

et la carte de données

1	2	3	4	5	6	7	8	9	10	11	12	13	14
F	Ø	R	T	R	A	N	b	C	Ø	B	Ø	L	

le contenu de A est FØRTbbbb,
le contenu de B est RANbCØbb

b) si w = 8 ▲31

Les 8 caractères sont transcrits dans le registre.

Exemple :

	READ (60,100) A
	. . . . .
100	FØRMAT (A8)

et la carte de données

1	2	3	4	5	6	7	8	9	10	11
Ø	R	D	I	N	A	T	E	U	R	

Le contenu de A est ØRDINATE
(8 caractères)

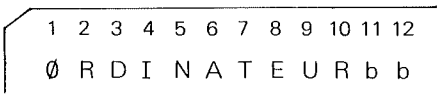
c) si w > 8 ▲31

Les 8 caractères les plus à droite sont transcrits, les autres sont ignorés.

Exemple :

	READ (60,100) A
	. . . . .
100	FØRMAT (A12)

et la carte de données



Le contenu de A est NATEURbb
8 caractères

Remarque :

Un chiffre octal peut être représenté par 3 chiffres binaires (bits) au maximum.

Exemples : 7_8 111_2
5_8 101_2
2_8 010_2

Sur la CDC 3800, chaque caractère est représenté par un double chiffre octal, donc 6 bits. Mais comme la longueur d'un mot est de 48 bits, il y aura au maximum 8 caractères par mot, d'où l'importance du nombre 8 pour les FØRMAT A et R. ▲31

II. SORTIES

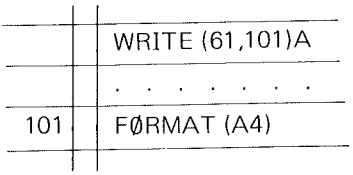
Les 8 caractères sont copiés comme suit :

a) si $w < 8$ ▲31

Seuls les w caractères les plus à gauche sont copiés.

Exemple :

Le contenu de A est FØRTRAN



Le contenu de A est FØRTbbb

b) si $w = 8$ ▲31

Les 8 caractères occupent les 8 positions du champ.

c) si $w > 8$ ▲31

Les 8 caractères du registre sont imprimés dans la partie droite du champ w et des blancs le complètent à gauche.

Exemple :

Le contenu de B est PØLYNØME

```
      WRITE (61, 101) B
      . . . . . .
101   FØRMAT (A15)
```

L'imprimante écrira bbbbbbbPØLYNØME

10.2. FØRMAT R ▲18

Le FØRMAT R est comparable au FØRMAT A à une exception près :

I. ENTRÉES

Pour $w < 8$, au lieu d'avoir les w caractères transcrits à gauche du registre, ils figurent à droite du registre

Exemple :

```
      READ (60, 100) A,B
      . . . . . .
100   FØRMAT (R4,R6)
```

et la carte de données

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14
F Ø R T R A N b C Ø B Ø L
```

Le contenu de A est bbbbFØRT
Le contenu de B est bbRANbCØ

II. SORTIES

Pour $w < 8$, seuls les caractères les plus à droite sont copiés.

Exemple :

Le contenu de A est FØRTRAN

```
      |  | WRITE (61, 101) A
      |  | . . . . . .
  101 |  | FØRMAT (R4)
```

L'imprimante écrira bbbbFØRT

Le FØRMAT R est particulièrement utile pour les problèmes de classement par ordre alphabétique. (cf. chap. 14)

10.3. FØRMAT E

La forme générale du FØRMAT E est

aEw.d

où a est une constante entière sans signe indiquant le nombre de fois que le code doit être répété. Si a = 1, on peut l'omettre.

w est une constante entière sans signe indiquant le nombre total de caractères significatifs du champ.

d est une constante entière sans signe indiquant le nombre de chiffre(s) ▲5 après le point.

La longueur du champ w doit être suffisamment grande pour contenir les chiffres significatifs, les signes, le point et l'exposant. On a généralement $w \geqslant d + 7$.

L'emploi de ce FØRMAT est très utile si l'on ne connaît pas l'ordre de grandeur de la variable A.

Exemple 1 :

A contient –78,388

```
          WRITE (9,100) A
100       FØRMAT (1X,E10.4)
```

Le résultat est : –7.8388+01 ce qui signifie : $-7{,}8388.10^{1}$

```
          WRITE (9,100) A
100       FØRMAT (1X,E12.4)
```

Le résultat sera : bb–7.8388+01

Exemple 2 :

Soit une variable X ayant pour valeur 0,028

```
          WRITE (9,8) X
8         FØRMAT (1X,E14.7)
```

Le résultat est : bb2.8000000–02 soit : $2{,}8.10^{-2}$

Exemple 3 :

Si nous avons l'instruction

```
          READ (60,100) A,B,C
100       FØRMAT (E8.1,2E10.3)
```

et que nous voulons attribuer à A, B, C respectivement les valeurs –8.5, 0.008324 et 20, la carte donnée se présente ainsi :

```
–8.5 + 000 b 8.324 – 003 b 2.000 + 001
   E + 00      E – 03      E + 01
<--------><--------><--------->
    8           10          10
```

10.4. FØRMAT D

Le FØRMAT D permet l'utilisation de variables en double précision, ce qui permet d'augmenter le nombre de chiffres significatifs (qui n'était que de 10 environ pour les FØRMAT F ou E) à 25 environ. ▲26

La forme générale du FØRMAT D est :

aDw.d

Le FØRMAT D se comporte essentiellement comme le FØRMAT E pour ce qui est des constantes w et d.

Toutes les variables utilisées en double précision doivent être déclarées au début du programme au moyen de l'instruction

TYPE DOUBLE liste	ou	DOUBLE PRECISION liste

où liste est la liste des variables que l'on déclare être en double précision. ▲31

Cette instruction se place juste au début du programme (instruction non exécutable).

Exemple 1 :

```
      DOUBLE PRECISION A,B
      .
      .
      .
      .
      READ (9,1012),A,B
1012  FØRMAT (2D15.10)
```

Pour une initialisation dans le programme d'une variable en double précision on aura par exemple :

```
      .
      .
      .
      A = 4.3D+02
      .
      .
      .
```

CHAPITRE 11

EXPRESSIONS LOGIQUES

Dans l'utilisation des IF logiques (cf. chap. 7), nous avons besoin d'opérants logiques qui peuvent être des cinq types suivants :

1. constante logique
2. variable logique
3. relation arithmétique (opérateur de relation)
4. expressions logiques (opérateur logique)
5. expression arithmétique (cas peu fréquent, cf [1]). ▲19

11.1. CONSTANTE LOGIQUE

Une constante logique est de la forme .TRUE. (valeur 1) ou .FALSE. (valeur 0). Elle est traitée en machine comme les entiers.

Exemple :

```
      ....
      RUSE=.TRUE.
      ....
      TEST=.FALSE.
      ....
```

11.2. VARIABLE LOGIQUE

Une variable logique doit toujours être déclarée au début du programme au moyen de l'instruction

```
LØGICAL, liste
```

où liste est la liste des variables que l'on déclare être logique.

LØGICAL est un ordre non exécutable qui doit se trouver avant le premier ordre exécutable du programme.

Exemple :

```
      LØGICAL TEST,RUSE,TRUC
```

Remarque :

Une variable logique occupe un mot. ▲22

11.3. LECTURE ET ECRITURE DES VARIABLES LOGIQUES

La forme générale du FØRMAT L est

```
aLw
```

où a est une constante entière sans signe indiquant le nombre de fois que le code de FØRMAT doit être répété; si a est omis, le code est répété une fois.

w est une constante entière sans signe indiquant la longueur totale du champ (w est généralement égal à 1).

Exemple :

```
      ...
108   FØRMAT (L1,2L2/L3)
      ...
```

a) **en sortie** (c'est-à-dire pour l'écriture)

Si la variable a la valeur VRAI, il y a impression d'un 1 à droite de la zone de w positions, précédé de (w−1) blancs. Sinon, c'est un 0 qui s'imprime dans les mêmes conditions. ▲20

Exemples :

1. Si I est vrai, J fausse, K vraie et L vraie

```
      |  LØGICAL I,J,K,L
      |  WRITE (9,105) I,J,K,L
  105 |  FØRMAT (4L3)
```

Résultat : bb1bb0bb1bb1

2.

```
      |  LØGICAL L,M
      |  L=.TRUE.
      |  M=.FALSE.
      |  WRITE (9,100) L,M
  100 |  FØRMAT (1XL1,L3)
```

Résultat : b1bb0

b) **à l'entrée** (c'est-à-dire pour la lecture)

Un seul caractère par champ (1 pour vrai ou 0 pour faux). ▲32

11.4. RELATION ARITHMÉTIQUE

Les opérateurs de relation s'appliquent à des expressions **arithmétiques** et le résultat est une variable **logique** c'est-à-dire une variable qui ne prend que les valeurs VRAI ou FAUX.

Voici la liste des 6 opérateurs de relation :

$=$	.EQ.	égal	P.EQ.Q	vrai si $P = Q$ faux si $P \neq Q$
$\neq$	.NE.	différent	P.NE.Q	vrai si $P \neq Q$ faux si $P = Q$
$<$	.LT.	strictement plus petit	P.LT.Q	vrai si $P < Q$ faux si $P \geqslant Q$
$\leqslant$	.LE.	plus petit ou égal	P.LE.Q	vrai si $P \leqslant Q$ faux si $P > Q$
$>$	.GT.	strictement plus grand	P.GT.Q	vrai si $P > Q$ faux si $P \leqslant Q$
$\geqslant$	.GE.	plus grand ou égal	P.GE.Q	vrai si $P \geqslant Q$ faux si $P < Q$

Exemples :

```
      ....
      X.LT.1.
      ......
      2*I.EQ.J
      ....
      X.NE.Y
      ....
      R-Q*Z.LE.3.14159
      ......
      B-C.NE.D+E
      ....
      K.GE.16
      .....
```

Remarques :

1. Le mélange de mode est possible. ▲16
2. Deux opérateurs de relation ne peuvent pas se suivre.

	A.GT.B.GT.C

n'a pas de sens.

11.5. EXPRESSIONS LOGIQUES

Les opérateurs logiques s'appliquent à des variables **logiques** et le résultat est une variable **logique.**

Voici la liste des opérateurs logiques :

non	.NØT.	.NØT.P	vrai si *P* est faux faux si *P* est vrai
et	.AND.	P.AND.Q	vrai si *P* et *Q* sont vrais faux si *P* est vrai et *Q* faux faux si *P* est faux et *Q* vrai faux si *P* et *Q* sont faux
ou	.ØR.	P.ØR.Q	vrai si *P* et *Q* sont vrais vrai si *P* est vrai et *Q* faux vrai si *P* est faux et *Q* vrai faux si *P* et *Q* sont faux

On aura remarqué qu'il s'agit en fait des opérateurs logiques **non**, **et**, **ou** (non exclusif) de la logique dont les tables de vérité sont supposées connues. (Le lecteur peut se référer à l'ouvrage intitulé "Eléments de logique" paru aux Editions SPES en 1973 [15]).

Remarque :

Dans l'écriture des opérateurs logiques et des opérateurs de relation, la présence des points (.) qui les précèdent et qui les suivent est obligatoire.

Exemples :

M1, M2, M3, M4 sont des variables logiques.

Les expressions suivantes sont correctes :

```
M1.ØR.M2
M3.AND.SQRT(XIY).NE.W
M4.AND..NØT.M1
```

Contre-exemples :

M1, M2, M3, M4 sont des variables logiques.

```
M1.AND..ØR.M3
.AND.M4
M2.NØT..AND.M1
H.GT.(M1.AND.M2)
```

Remarques :

1. Il n'y a pas d'opérateurs logiques pour la conditionnelle et pour la biconditionnelle mais le rappel suivant permet au lecteur de combler cette lacune :

$$(P \Rightarrow Q) \iff [(\text{non } P) \text{ ou } Q]$$
$$(P \Leftrightarrow Q) \iff [((\text{non } P) \text{ ou } Q) \text{ et } ((\text{non } Q) \text{ ou } P)]$$

2. Les tests logiques sont traités à la page 56.

Exercices :

11.1. L est une variable logique, X, Y, Z étant des variables flottantes telles que $X < Z$ donner la séquence telle que L prend la valeur .TRUE. si Y appartient à [B; D] ■

11.2. P, Q et R étant trois variables logiques, écrire que P est fausse si Q est fausse et R est vraie, et P est vraie si Q est vraie et R est fausse. ■

11.3. Si dans la première séquence d'instructions ci-dessous on introduit une carte blanche ou une ligne sans caractère, que pouvez-vous déduire des valeurs respectives de la variable TEST dans les deux parties de programmes suivantes ?

```
      LØGICAL RUSE,SIØUX, TEST
      READ (9,100) RUSE,SIØUX
      TEST=RUSE.AND..NØT.SIØUX
      .
      .
      .
      .
100   FØRMAT (2L1)
      END
```

```
      LØGICAL RUSE,SIØUX,TEST
      RUSE=.FALSE.
      SIØUX=.FALSE.
      TEST=RUSE.AND.(.NØT.SIØUX)
      .
      .
      .
      .
```

11.4. Avec ou sans boucle DØ, écrire le programme de la table des valeurs de la :

- conjonctive (P **et** Q)
- disjonctive (P **ou** Q)
- conditionnelle ($P \Rightarrow Q$)
- biconditionnelle ($P \Longleftrightarrow Q$) (cf. [15])

■

CHAPITRE 12

SOUS-PROGRAMMES

(Subroutine fonction)

12.1. INTRODUCTION

Soit le problème suivant : lire 3 matrices carrées A, B, C de même type ordre n ($n \leqslant 10$) et montrer l'associativité de la multiplication (A.B).C = A.(B.C).

On pourrait tenter l'essai suivant :

```
      |  DIMENSIØN A(10,10),B(10,10),C(10,10),D(10,10)
*     |  LECTURE DE L ØRDRE N DES TROIS MATRICES
   12 |  READ (9,100)N
  100 |  FØRMAT (I2)
      |  IF (N)99,99,11
*     |  LECTURE DES TROIS MATRICES
   11 |  DØ 1 I=1,N
    1 |  READ (9,101) (A(I,J),J=1,N)
  101 |  FØRMAT (10F8.3)
      |  2 I=1,N
      |  DØ 2 I=1,N
```

```
      2 |   READ (9,101) (B(I,J),J=1,N)
        |   DØ 3 I=1,N
      3 |   READ (9,101) (C(I,J),J=1,N)
*       |   D=A*B
        |   DØ 4 I=1,N
        |   DØ 4 J=1,N
        |   D(I,J)=0.
        |   DØ 4 K=1,N
      4 |   D(I,J)=D(I,J)+A(I,K)*B(K,J)
*       |   D=(A*B)*C
        |   DØ 5 I=1,N
        |   DØ 5 J=1,N
        |   S=0
        |   DØ 6 K=1,N
      6 |   S=S+D(I,K)*C(K,J)
      5 |   D(I,J)=S
*       |   B=B*C
        |   DØ 7 I=1,N
        |   DØ 7 J=1,N
        |   S=0.
        |   DØ 8 K=1,N
      8 |   S=S+B(I,K)*C(K,J)
      7 |   B(I,J)=S
*       |   C=A*(B*C)
        |   DØ 9 I=1,N
        |   DØ 9 J=1,N
        |   C(I,J)=0.
        |   DØ 9 K=1,N
      9 |   C(I,J)=C(I,J)+A(I,K)*B(K,J)
```

```
*        |  TEST DE L'ASSØCIATIVITE
         |  DØ 10 I=1,N
         |  DØ 10 J=1,N
         |  D(I,J)=D(I,J)-C(I,J)
         |  IF (D(I,J)NE.0.)WRITE (9,102) J,K
      10 |  CØNTINUE
         |  WRITE (9,103)
     102 |  FØRMAT ('(AB)C('I2,IH,I2') EST DIFFERENT
         |  DE A(BC) ('I2,IH,I2')')
     103 |  FØRMAT ('LE PRØDUIT EST ASSØCIATIF')
         |  GØTØ 12
      99 |  STØP
         |  END
```

D'emblée les remarques suivantes s'imposent :

1. A plusieurs reprises des séquences semblables avec des variables différentes sont répétées.

Exemples :

les boucles 1, 2, 3 (trois lectures de matrice)

ou les boucles 4, 5, 7, 9 (quatre produits matriciels).

2. Il est avantageux de programmer chaque séquence une fois et de l'utiliser plusieurs fois avec des variables différentes pour exécuter cette suite d'instructions.

3. Le langage FORTRAN offre deux possibilités d'écrire une seule fois de telles séquences (sous-programmes) :
   - en utilisant une **subroutine**,
   - en utilisant une **fonction**.

4. Une différence essentielle entre "fonction" et "subroutine" est que la fonction fournit un résultat unique, alors que la subroutine fournit un, plusieurs ou aucun résultat(s).

5. L'utilisation de subroutines ou de fonctions est également fréquente dans de longs programmes. Il est, en effet, recommandé de fractionner un problème important; chaque sous-programme étant testé, on procède ensuite à la mise au point du programme principal. Cette méthode facilite notamment la recherche d'erreurs éventuelles.

6. L'exemple choisi précédemment est simple; mais il est fréquent d'utiliser 100 ou 1000 fois une séquence.

12.2. SUBRØUTINE

Essai de définition

On appelle **subroutine** une séquence de calcul écrite avec des variables fictives (arguments) qui prennent lors de l'exécution les noms des variables du programme appelant la subroutine.

L'instruction d'une subroutine est de la forme :

```
SUBRØUTINE nom
```

ou

```
SUBRØUTINE nom (P1 ,.,Pi,., Pn) 1 ⩽ n ⩽ 63
```

où nom est le nom de la subroutine (alphanumérique),

P_i l'argument qui peut être :
- le nom d'une variable simple,
- le nom d'un tableau,
- le nom d'une fonction de librairie,
- le nom d'une fonction ou d'une subroutine.

Remarque :

Si un des arguments est le nom d'une variable dimensionnée, il doit être déclaré dans un ordre DIMENSIØN dans la subroutine.

Exemples :

1.
```
SUBRØUTINE X1ABC2
.
.
.
```

2.

```
SUBRØUTINE ORTHO(A,B,C,I,K,Z)
.
.
.
```

3.

```
SUBRØUTINE BIDØN(A,B,C,D)
DIMENSIØN A(10,10),B(5),C(1000)
```

Chaque subroutine doit contenir au moins un ordre RETURN et se terminer par l'instruction END.

L'instruction RETURN indique la fin logique de la subroutine et rend le contrôle au programme (ou sous-programme) qui avait appelé la subroutine.

L'ordre END signale simplement la fin physique de la subroutine.

Exemples :

1.

```
SUBRØUTINE EXERCI (A,Z,MEAN)
.
.
.
RETURN
END
```

2.

```
SUBRØUTINE CALCUL (NØRM,VECT,EPSI)
DIMENSIØN VECT (10,60)
.
.
.
IF (S.GT.EPSI) RETURN
.
.
.
RETURN
END
```

3. Supposons que dans un programme il soit fréquemment nécessaire de trouver le plus grand élément en valeur absolue d'une ligne d'une matrice carrée d'ordre 10.

```
      |  SUBRØUTINE PLGDEL (A,I,GREAT,J)
      |  DIMENSIØN A(10,10)
      |  GREAT=ABS(A(I,1))
*     |  I EST L'INDICE DE LA LIGNE A CONSIDERER
      |  J=1
      |  DØ 17 M=2,10
      |  IF(ABS(A(I,M)-GREAT)17,17,18
   18 |  GREAT=ABS(A(I,M))
      |  J=M
   17 |  CØNTINUE
      |  RETURN
      |  END
```

Les résultats fournis par cette subroutine sont :

GREAT le plus grand élément de la ligne I de la matrice A.

J l'indice de la colonne où se trouve ce plus grand élément. (donc GREAT = A(I,J))

12.3. APPEL D'UNE SUBRØUTINE

Pour appeler une subroutine, on utilise l'instruction

```
CALL nom
```

ou

```
CALL nom (P1 , ..... , Pn) 1 ⩽ n ⩽ 63
```

où nom est le nom d'une subroutine

P_i étant les arguments actuels.

Remarques :

1. L'ordre CALL transfère le contrôle à la subroutine et donne aux arguments fictifs de celle-ci les valeurs courantes des arguments actuels de l'instruction CALL.
2. En principe, les arguments fictifs de la subroutine et les arguments actuels de l'ordre CALL doivent être :
   - de même type,
   - dans le même ordre,
   - en même nombre,
   - également dimensionnés.

   Il faut noter qu'il existe quelques exceptions.
3. Les arguments de l'instruction CALL peuvent être :
   - des constantes,
   - des expressions arithmétiques,
   - des variables simples,
   - des variables dimensionnées,
   - le nom d'un tableau,
   - le nom d'une "fonction" ou d'une "subroutine".

Exemples :

1.

```
      DIMENSIØN B(10,10)
      .
      .
      .
      DØ 1 I=1,N
      K=....
      .
      .
      CALL PLGDEL(B,K,BIG,M)
    1 WRITE....
      .
      .
      STØP
      END
```

```
SUBRØUTINE PLGDEL(A,I,GREAT,J)
DIMENSIØN A(10,10)
.
.
RETURN
END
```

2.

```
DIMENSIØN B(7),C(15,15)
.
.
.
CALL FØRM(L1,M,N,B(1),C,0.5,3)
.
.
.
END
SUBRØUTINE FØRM(I,J,N,A,V,Z,L1)
DIMENSIØN V(15,15)
.
.
.
RETURN
END
```

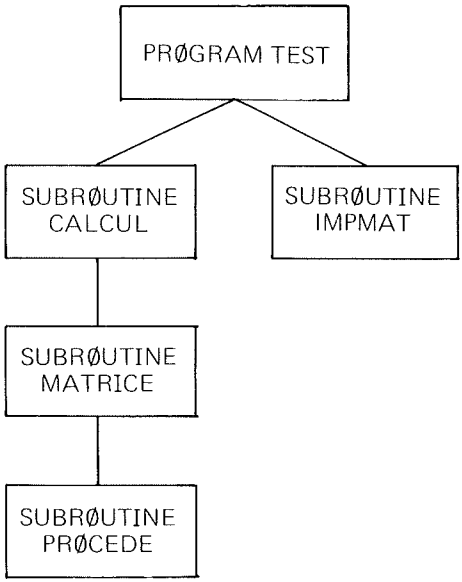
4. Le nom intervenant dans CALL nom ne doit pas être celui d'une fonction de la librairie.

5. Une subroutine peut appeler elle-même une autre subroutine. Le programme qui n'est appelé par aucune subroutine est le **programme principal**.

Exemple :

```
PRØGRAM TEST                                    ▲34
.
.
.
```

```
CALL CALCUL(....)
.
.
.
CALL IMPMAT(....)
.
.
.
CALL EXIT
END
SUBRØUTINE CALCUL(....)
.
.
.
CALL MATRICE(....)
RETURN
END
SUBRØUTINE IMPMAT(....)
.....
RETURN
END
SUBRØUTINE MATRICE(....)
.
.
.
CALL PRØCEDE
RETURN
END

SUBRØUTINE PRØCEDE
.
.
.
RETURN
END
```



6. Une subroutine peut utiliser un ou plusieurs de ses arguments pour des valeurs en retour vers le programme (ou sous-programme) qui l'avait appelée.

Exemple :

```
SUBRØUTINE PLGDEL(A,I,GREAT,J)
```

Exercices :

12.1. Ecrire une subroutine pour l'évaluation de la conditionnelle.

12.2. Ecrire une subroutine pour l'évaluation d'une biconditionnelle. ■

12.3. Ecrire une subroutine générale pour l'évaluation de propositions logiques.

12.4. Ecrire une subroutine pour calculer l'intégrale d'une fonction $F(x)$ sur $[a; b]$:
a) par la sommation de rectangles (méthode de l'escalier),
b) par la sommation de trapèzes (méthode du trapèze) ■
(cf. exercice 13.5.)

12.4. FONCTION

Une **fonction** est un sous-programme calculant, à partir de plusieurs arguments, un seul résultat.

On a utilisé jusqu'à présent des expressions du type SQRT(X), SIN(X), etc. pour désigner des fonctions. Ces expressions sont des fonctions de X.

Lorsque la valeur numérique de X est définie, l'égalité Y=SIN(X) permet de calculer aisément la valeur numérique de Y.

Il en est de même des fonctions CØS(X), ALØG(X), ABS(X), etc. qui sont programmées sur la plupart des ordinateurs.

En revanche, si nous avons besoin de fonctions non programmées, il faut alors les programmer soi-même.

Exemple :

Dans un programme intervient à plusieurs reprises la fonction $Y = \sqrt{1 + X^2}$ qui n'est pas programmée. Si CALC(X) est cette fonction, on a :

```
FUNCTIØN CALC(X)
CALC=SQRT(1.+X**2)
RETURN
END
```

Dans cet exemple très simple, on découvre la structure d'une fonction.

L'instruction d'une fonction est de la forme :

FUNCTIØN nom (liste d'appel)

La liste d'appel est composée des variables intervenant dans le calcul effectué par la fonction. Il est, bien entendu, indispensable d'avoir défini ces variables préalablement.

Exemple :

```
Y=CALC(1.2)
```

Le programme principal fait appel à la fonction CALC qui calcule. $\sqrt{1 + (1.2)^2}$ et met cette valeur en Y.

Dans cet exemple, la fonction a une liste d'appel contenant une seule variable X. C'est un cas particulier.

Remarques :

1. Il y a une grande différence entre les variables de la liste d'une fonction et celles de la liste d'une subroutine.

   Pour une subroutine les variables peuvent être définies dans **le programme appelant et intervenir dans la subroutine** ou **être définies dans celle-ci et revenir ainsi au programme principal**. On a des variables d'entrée et de retour.

   Pour une fonction, toute la liste d'appel doit être définie dans le **programme appelant** et sert ainsi au calcul de la fonction. Un seul résultat est donné par la fonction.

2. Le nom d'une fonction suit les mêmes règles que le nom d'une variable :
   a) il ne doit pas dépasser 6 caractères alphanumériques,
   b) le premier caractère doit être une lettre,
   c) le résultat est entier si le nom de la fonction commence par I, J, K, ... N, et flottant dans les autres cas.

Exemple :

Considérons la fonction : $\left(\frac{X}{Y}\right)^2$. Appelons-la DIV2(X,Y).

```
FUNCTIØN DIV2(X,Y)
DIV2=(X/Y)**2
RETURN
END
```

et dans le programme principal :

```
Z=DIV2(3.,2.)
```

Dans Z on aura donc : $\left(\frac{3.}{2.}\right)^2$, c'est-à-dire 2,25.

Si l'on appelle la fonction IDIV2(X,Y), on aura :

```
FUNCTIØN IDIV2(X,Y)
IDIV2=(X/Y)**2
RETURN
END
```

et dans le programme principal :

```
Z=IDIV2(3.,2.)
```

Dans Z, $\left(\frac{3.}{2.}\right)^2$ sera converti tout d'abord en entier à cause du nom de la fonction, puis reconverti en flottant, car Z est de ce mode. Ainsi dans Z on obtiendra 2.

3. Dans une fonction, le nom doit se trouver **au moins une fois** à gauche du signe d'égalité et **sans argument**. A droite du signe d'égalité, il est essentiel de trouver une expression formée avec les arguments de la fonction ou une partie de ceux-ci.

Exemple :

```
CALC=SQRT(1+X**2)
```

4. Une fonction peut être définie en plusieurs parties comme le montre l'exemple ci-dessous :

Rappel préliminaire : la partie entière d'un nombre a est le plus grand entier inférieur ou ou égal à a.

La partie entière d'un nombre positif est donnée par la fonction de bibliothèque IFIX(X), mais la partie entière d'un nombre négatif n'est pas fournie par cette fonction.

IFIX(−2.3)=−2 et PE(−2.3)=−3

Il faut donc soustraire 1 à IFIX(X).

La fonction PE(X) est donc définie ainsi :

```
FUNCTIØN PE(X)
PE=IFIX(X)
IF(X.GE.0.)RETURN
PE=IFIX(X)-1
RETURN
END
```

5. Une fonction se termine comme une subroutine par les instructions :

```
RETURN
END
```

6. Une fonction ne peut pas être dimensionnée. Aucune variable ne doit porter le même nom qu'une fonction. On ne peut donner à une fonction le nom d'une fonction de bibliothèque.

7. On peut faire figurer dans la liste d'appel une fonction, les mêmes sortes d'arguments que pour une subroutine.
   Cependant si cet argument est une fonction ou une subroutine il faut alors qu'elle figure dans l'instruction

```
EXTERNAL nom
```

   placée dans le programme principal, immédiatement après la carte du nom de programme.

Exemple :

```
FUNCTIØN VAL(X)
VAL=SQRT(X**2+X)
RETURN
END
FUNCTIØN TIM(Y,VAL)
TIM=VAL(Y)/(2.*Y)
RETURN
END
```

Dans le programme principal, l'ordre EXTERNAL VAL est obligatoire :

```
EXTERNAL VAL
.
.
.
W=TIM(12.5,VAL)
```

Exemple 2 :

```
FUNCTIØN FEC(A,B,CALCUL)
CALL CALCUL(A)
FEC=B+A
RETURN
END
```

dans le programme principal :

```
EXTERNAL CALCUL
.
.
.
V=FEC(A,4.2,CALCUL)
```

A est calculé par la subroutine CALCUL avant d'intervenir dans le calcul de FEC.

8. Une variable dimensionnée peut figurer dans la liste d'appel d'une fonction. Elle doit alors être aussi dimensionnée dans la fonction.

Exemple :

```
      FUNCTIØN SUM(A)
      DIMENSIØN A(50)
      SUM=0.
      DØ 1 I=1,50
1     SUM=A(I)+SUM
      RETURN
      END
```

Dans le programme principal :

```
DIMENSIØN A(50)
.
.
.
.
.
Z=SUM(A)
```

9. Il est évidemment pratique dans l'exemple précédent d'avoir les dimensions de A variables dans la fonction, cette possibilité existe en écrivant :

```
      FUNCTIØN SUM(A,N)
      DIMENSIØN A(N)
      SUM=0.
      DØ 1 I=1,N
1     SUM=A(I)+SUM
      RETURN
      END
```

Dans le programme principal :

```
      DIMENSIØN A(50)
      .
      .
      .
      .
      .
      Z=SUM(A,27)
```

Dans Z on a la somme des 27 premiers termes du vecteur A.

La dimension de A **dans le programme principal** doit être numérique et N ne ▲33
doit pas dépasser cette dimension.

Cette remarque s'applique également dans le cas de subroutines (voir exercices).

10. Une fonction peut être logique; elle a alors des arguments logiques. Ces derniers et la fonction doivent être déclarés dans un ordre LØGICAL :

Exemple :

```
FUNCTIØN CONJ(P,Q)
LØGICAL CONJ,P,Q
CONJ=P.AND.Q
RETURN
END
```

dans le programme principal :

```
.
.
.
.
.
LØGICAL CONJ,L,M,K
L=.TRUE.
M=.FALSE.
K=CONJ(L,M)
.
.
.
.
.
```

Exercices

12.5. Ecrire la table des valeurs des propositions logiques composées en utilisant des fonctions. ■

12.6. Calculer les images d'un certain nombre de valeurs pour une fonction.

12.7. Calculer la dérivée en certains points d'une fonction.

12.8. Ecrire un sous-programme d'addition et de produit matriciels **avec des dimensions variables.**

CHAPITRE 13

EXERCICES RÉCAPITULATIFS

13.1. Etudier la convergence du rapport successif de deux termes de la suite de Fibonacci.

Soit la suite : $1 ; 1 ; 2 ; 3 ; 5 \ldots . ; F(n) ; \ldots .$

où $F(n) = F(n-2) + F(n-1)$

a) Ecrire les m premiers termes de la suite.

b) Le rapport $\frac{F(n+1)}{F(n)}$ converge-t-il ? Si oui, vers quelle limite ? ■

13.2. Etablir un programme pour rechercher les n premiers nombres premiers ou les nombres premiers de l'intervalle [1 ; p [.

Rappel : Un nombre est premier si l'ensemble de ses diviseurs est une paire.

Indication :
Admettre 1 comme premier nombre premier; on supposera connus les deuxième et troisième nombres premiers :

$N(1) = 2 , \quad N(2) = 2 , \quad N(3) = 3.$

On élimine ensuite, à priori, tous les éléments impairs et on teste les nombres impairs en les divisant successivement par tous les nombres inférieurs premiers déjà trouvés. ■

13.3. Classer dans l'ordre croissant (respectivement décroissant) n nombres.

Indication :
Comparer le premier nombre A aux suivants :

- dès que l'on rencontre un nombre B plus petit que le premier, c'est B qui devient le premier et qui continue à être comparé aux autres nombres de la liste.
- dès que cette comparaison est terminée, on recommence avec le deuxième nombre, puis le troisième, etc. ■

13.4. Déterminer le noyau d'une application par la méthode de Newton.

Rappel : Dérivée d'une application

Soit une application $f : \mathbb{R} \to \mathbb{R}$ admettant une dérivée $f' : \mathbb{R} \to \mathbb{R}$.
En un point x_0, la valeur $f'(x_0)$ est la pente de la droite tangente à l'application f au point $< x_0 ; f(x_0) >$.

$f'(x_0) = \text{tg}(\alpha)$

$y_0 = f(x_0)$

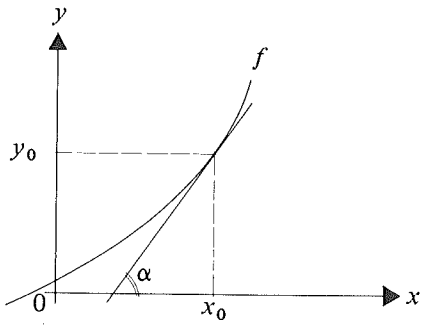


Fig. 7

Méthode de Newton

Indication : on calcule le zéro d'une fonction par approximations successives.
Pour cela :

- choisir un point x_0 et calculer $f(x_0)$;
- déterminer la tangente à la courbe au point $< x_0 ; f(x_0) >$ et son intersection x_1 avec l'axe des premières projections;
- continuer l'opération et s'arrêter lorsque la précision désirée est atteinte.

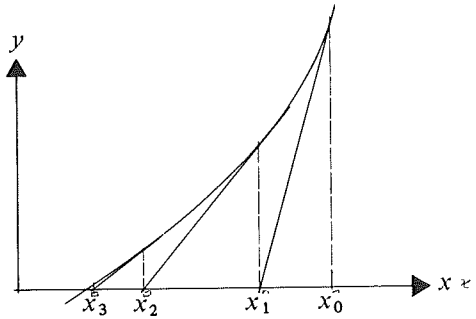


Fig. 8

On obtient ainsi une suite qui tend vers la solution.

Calculer x_{n+1} à partir de x_n (formule générale).

$$\text{tg}\,\alpha = f'(x_n) = \frac{f(x_n)}{x_n - x_{n+1}}$$

On obtient :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

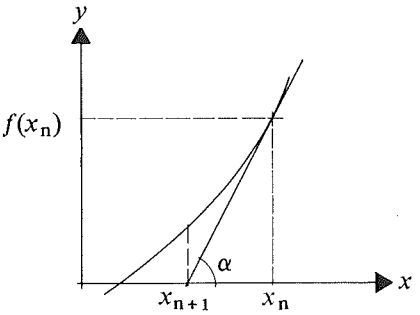


Fig. 9

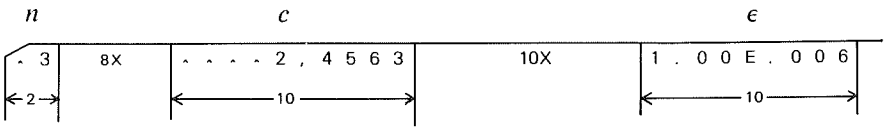
QUESTIONS

a) Calculer $\sqrt[n]{c}$

$\sqrt[n]{c}$ est solution de $x^n - c$ $\quad f(x) = x^n - c$ $\quad f'(x) = nx^{n-1}$

On arrête les itérations lorsque $\dfrac{x^n - c}{c} \leqslant \epsilon$

b) On donne des cartes sur lesquelles figurent :



trouver $\sqrt[n]{c}$ à ϵ près pour un nombre quelconque de données. Prévoir le cas où c est négatif et n est pair.

N.B. Essayer d'optimiser le programme, c'est-à-dire de ne pas calculer plusieurs fois les mêmes expressions ou parties d'expressions, ou encore de ne pas calculer deux fois des puissances d'un même nombre.

Exemple : Pour calculer $A = \dfrac{x^n}{x^{n-1} + D}$

Au lieu de :

```
A = X**N/(X**(N-1)+D)
```

N	multiplication
N−1	multiplication
1	division
1	soustraction
1	addition
2N+2	opérations arithmétiques

Il vaut mieux écrire :

```
XNM1 = X**(N-1)
A = X*XNM1/(XNM1+D)
```

N−1	multiplication
1	soustraction
1	multiplication
1	division
1	addition
N+3	opérations arithmétiques

13.5. Calculer $\int_a^b F(x)\,dx$ sachant que cette intégrale définie mesure l'aire comprise entre l'application F, l'axe des premières projections et deux droites parallèles à Pr_2 contenant les points $<a;0>$ et $<b;0>$.

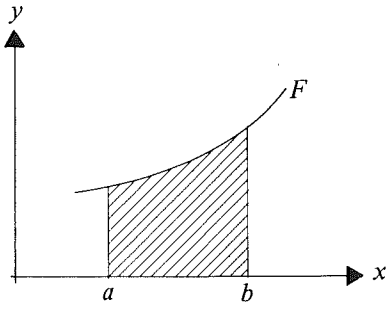


Fig. 10

Indication :
On peut approcher cette surface en remplaçant l'application F par une fonction "escalier" et en additionnant les aires des rectangles ainsi définis :

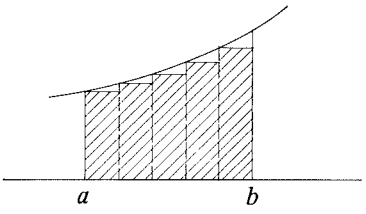


Fig. 11

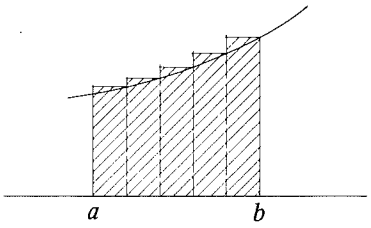


Fig. 12

L'aire calculée à la figure 11 est inférieure à l'aire définie par l'intégrale; celle de la figure 12 est supérieure. Il est évident que plus les rectangles sont nombreux, plus la différence des deux résultats est petite.

QUESTIONS

a) Calculer l'aire du quart de cercle défini par l'application $F : x \rightarrow 1 - x^2$ ($a = 0$ et $b = 1$) voir figure 13;

b) déterminer b de manière à obtenir une aire de 1 avec l'application $G : x \rightarrow \frac{1}{x}$ et la valeur $a = 1$ voir figure 14.

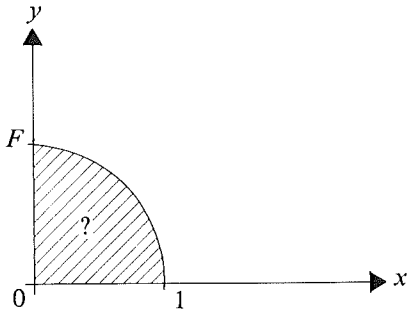


Fig. 13

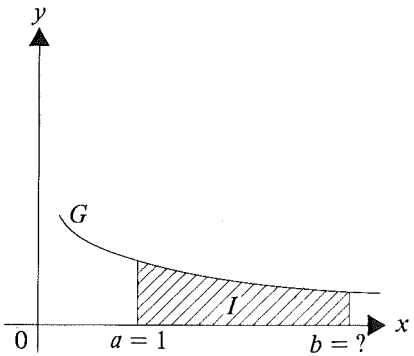


Fig. 14

Voir aussi l'exercice 12.4.

13.6. Représenter graphiquement sur l'imprimante la fonction $F : x \longrightarrow \sqrt[3]{x^3 - 2x^2 - x + 2}$ de $[-3; 4]$ dans $\mathbb{R}$ et son asymptote oblique ($y = x - 2/3$). Il serait peut-être préférable d'étudier complètement l'application F avant d'écrire le programme. ■

13.7. Etant donné la liste et le tableau suivants :

ABOUT
ACTE
ACTUALITE
ALGUE
AMIS
ANGE
ARRACHER
ASSASSINER
ASSEZ
ASSURANCE
ASSURANCE
ASSUREUR
ASTICOTS
ATTRAPER
AVALER
AVANT
AXEE
BATTRE
BETES
BIPLAN
CALCULATRICE
CALER
CALER
CANUT
CAPITULATION
CAQUETER
CARREE
CASER
CASER
CASSE
CESSATION
COQ
CORS
CRIAS
CRIER
CUIT
CUPIDE
CUVER
CYANURE
DIPLODOCUS
ECHECS
ECOLE
ECOUTE
ENNEMIS
ESSAYER
ESTAFILADE
ETERNUE
EXOTISME
FAUNE
FEES
FIXER
FIXER
FUMER
HISSE
IDIOT
INFINITESIMAL
IVRE
LECHEE
LECON
LEURS
LION
LOISIR
LUBRIFIER
LUCRE
LUCRE
LUCRE
LUIRE
LUSTUCRU
MAURE
MEFIANT
MIEUX
MINIUM
MINUIT
MISER
MORT
NATIONSUNIES
NATURE
NATUREL
NATURELLEMENT
NELSON
NERF
NOURRIR
NUIT
NUITEE
OBSTINATION
ODES
ŒUF
ŒUFS
ORAGE
ORDINATEUR
ORTIES
PAPOTER
PARAMETRE
PARIS
PARLE
PARTICULARITE
PLANETES
PRECISER
PRET
PRISER
RARE
RAVI
RETS
RICHESSES
RIRE
RIRE
RITE
ROTI
ROUE
SATURATION
SISAL
SOCLE
SOIR
SOIT
SORTIR
SOT
SUIVRE
SURFACES
TELEVISION
TEMERITE
TERRE
TERREUX
TOTAL
TORT
TOUSSER
TREPIED
TRIS
TRIBUNES
TROIE
TRONE
TROU
TROUVER
TROUVER
TUTIE
VIVRE
VUS

A	S	S	A	S	S	I	N	E	R	A	M	D	E	I	P	E	R	T	E	M	A	R	A	P
R	C	A	S	E	R	O	O	B	S	T	I	N	A	T	I	O	N	C	T	U	O	B	A	A
R	E	T	S	A	U	I	I	E	U	T	S	S	I	R	A	P	O	O	E	R	T	P	L	R
A	S	S	U	R	E	U	R	T	L	R	E	E	H	C	E	L	N	T	R	E	O	I	G	T
C	I	U	R	A	L	E	I	E	C	A	R	R	E	E	E	T	O	E	R	T	R	I	U	I
H	P	I	A	V	L	E	S	S	R	P	E	U	C	B	E	R	I	R	E	V	A	C	E	C
E	R	V	N	A	R	I	E	E	I	E	E	N	A	U	T	T	S	R	E	I	G	T	U	U
R	I	R	C	L	E	F	T	E	A	R	X	A	S	R	L	O	I	E	M	V	E	X	E	L
C	S	E	E	E	M	E	E	E	S	S	A	Y	E	R	T	T	V	U	I	R	U	R	I	A
A	E	R	A	R	U	E	N	F	L	E	S	C	R	C	R	S	E	X	N	E	V	O	S	R
P	R	E	T	Q	F	S	A	N	A	T	E	U	E	R	I	U	L	U	I	I	N	I	R	I
I	E	E	A	E	E	U	L	E	M	U	I	P	R	B	B	R	E	M	U	A	M	I	S	T
T	X	C	S	S	O	C	P	L	I	O	N	R	I	A	U	F	T	E	M	E	R	I	T	E
U	O	E	T	S	E	O	I	S	S	C	U	E	T	T	N	A	V	A	N	T	R	I	O	S
L	T	D	I	A	U	D	B	O	E	E	S	C	O	T	E	C	N	N	L	O	R	A	R	E
A	I	A	C	C	F	O	E	N	T	S	N	I	I	R	S	E	E	A	N	U	C	O	V	D
T	S	L	O	S	S	L	T	A	I	S	O	S	D	E	T	S	R	T	A	T	C	E	U	I
I	M	I	T	O	R	P	N	S	N	A	I	E	I	T	U	R	C	U	T	S	U	L	L	P
O	E	F	S	A	O	I	E	S	I	T	T	R	I	R	I	O	U	R	U	E	G	N	A	U
N	F	A	P	A	D	D	I	E	F	I	A	U	C	O	Q	U	L	E	R	E	T	C	A	C
U	I	T	M	R	L	R	E	Z	N	O	N	A	T	U	R	E	L	L	E	M	E	N	T	C
I	A	S	O	R	T	I	R	S	I	N	F	M	F	V	N	F	I	X	E	R	E	L	A	C
T	N	E	R	I	C	H	E	S	S	E	S	R	U	E	U	F	I	X	E	R	E	I	R	C
E	T	O	T	A	L	L	U	B	R	I	F	I	E	R	I	S	I	O	L	R	E	V	U	C
E	C	H	E	C	S	A	T	U	R	A	T	I	O	N	E	L	C	O	S	E	I	T	R	O

trouver le(s) mot(s) caché(s) dans le tableau en procédant comme suit :

a) barrer chaque mot de la liste que vous avez trouvé en le lisant dans la grille.

b) Les sens de lecture autorisés sont :
   - de gauche à droite ou de droite à gauche **horizontalement**,
   - de haut en bas ou de bas en haut **verticalement**,
   - de droite à gauche ou de gauche à droite **en diagonale**.

   (On remarquera qu'une lettre peut être utilisée par plusieurs mots suivant le sens de lecture.)

c) Quand la liste de mots est épuisée, il reste quelques lettres non biffées qui donnent le(s) mot(s) caché(s) dans le tableau.

Indication :
Avant d'entreprendre le problème, essayer "à la main" de rayer quelques mots de la liste pour se faire une idée d'une méthode susceptible d'être programmée.

■

13.8. Calculer le produit scalaire euclidien dans $\mathbb{R}^3$, l'angle de deux vecteurs de $\mathbb{R}^3$ et la distance dans $\mathbb{R}^3$.

Rappel : Soit $\{x;y\} \subseteq \mathbb{R}^3$ et $x = <\xi_1;\xi_2;\xi_3>$ et $y = <\eta_1;\eta_2;\eta_3>$.

1. $x \cdot y = \sum_{i=1}^{3} \xi_i \eta_i$
2. $\cos\theta = \dfrac{x \cdot y}{\|x\| \, \|y\|}$
3. $\|x\| = \sqrt{\sum_{i=1}^{3} \xi_i^2}$
4. $\delta(x;y) = \sqrt{\sum_{i=1} (\xi_i - \eta_i)^2}$

Indication :
Prévoir la lecture d'un indice à chaque passage permettant de choisir le (ou les) calcul(s) désiré(s).
(L'utilisation du GØTØ calculé est souhaitable.)

13.9. Exprimer au moyen d'un système de numérotation de base quelconque (entre 2 et 9) un nombre représenté dans le système décimal.

Rappel : 125_{10} signifie $1.10^2 + 2.10^1 + 5.10^0$

On lit donc, de droite à gauche, les coefficients des puissances successives de 10. Pour exprimer 125 en base 4, il faut déterminer les coefficients $x, y, z, u \ldots$ tels que $125 = \ldots + u.4^3 + z.4^2 + y.4^1 + x.4_0$. On a $125_{10} = 1331_4$.

Indication :
Lire le nombre entier à transformer et la base dans laquelle on désire l'exprimer. Imprimer le nombre en base 10, la nouvelle base et l'expression du nombre dans cette nouvelle base.

On peut **par exemple** stocker les valeurs de $x, y, z, u \ldots$ dans une variable dimensionnée que l'on imprime en une fois. Dans notre exemple, la plus haute puissance de 4 à traiter est 3. On aura donc :

A (1)	A (2)	A (3)	A (4)
1	3	3	1
4^3	4^2	4^1	4^0

sens du remplissage ⟶

13.10. Imprimer un histogramme..

Le problème est le suivant : on possède un certain nombre de mesures faites lors d'une expérience et on désire connaître la distribution de ces différentes valeurs, c'est-à-dire déterminer combien de valeurs sont situées entre 0 et 1, entre 1 et 2, entre 2 et 3, etc.

Chacune de ces catégories s'appelle un canal.

Sachant que les nombres à classer se situent entre 0 et 50, imprimer un histogramme (ligne par ligne), en faisant apparaître autant de caractères "X" qu'il y a de valeurs contenues dans le canal, par exemple :

Canal	Contenu	Histogramme
1	0	
2	3	XXX
3	5	XXXXX
.	.	.
.	.	.
.	.	.
10	8	XXXXXXXX
		etc.

Indications :

a) Déterminer une variable dimensionnée représentant les canaux de l'histogramme.

b) Lire les valeurs sur cartes ou dans un fichier et les répartir dans les différents canaux.

c) Imprimer ensuite autant de X que le nombre contenu dans le canal et ceci pour chaque canal (mettre sur la même ligne le contenu du canal et son numéro).

d) Une ligne d'imprimante comporte 120, 132 ou 136 caractères. ■

13.11. Déterminer des fréquences.

Le problème est le suivant : déterminer la fréquence d'un certain nombre de caractères (lettres, point, virgule, slash, parenthèses ouvertes et fermées, etc.) dans un texte donné.

Le texte est fourni sur cartes ou dans un fichier (dans le corrigé c'est le fichier TEXTE) – seules les colonnes 1 à 72 contiennent de l'information.

Indications :

a) Lire le texte et l'imprimer.

b) Faire un dénombrement en fréquence absolue et relative de **tous** les caractères.

c) Prévoir une extension du programme de façon à ne dénombrer les fréquences que du (ou des) caractère(s) figurant sur une carte paramètre. ■

13.12. Simuler un appareil permettant de rendre la monnaie.

Depuis quelques années, la plupart des supermarchés ont vu leurs caisses enregistreuses dotées d'appareils à rendre le dû en un minimum de monnaie.

Le problème consiste à écrire un programme fonctionnant de la même manière que ce genre d'appareil; à savoir que celui-ci emploie les trois seuls coefficients 1, 2, 5 (3, 4, 6, 7, 8, 9 étant des sommes de ces trois coefficients de base) et les puissances de dix (positives ou négatives) pour exprimer toutes les valeurs existantes dans la monnaie de tel ou tel pays. Le programme, en effet, doit pouvoir accepter et rendre dans quelques monnaies européennes basées sur le système décimal.

Le programme doit comprendre

à la lecture :
- les différents cours de monnaies européennes (facultatif)
- la date (facultatif)
- le montant à payer
- la monnaie du montant à payer (de même que celle de la somme à rendre)
- la somme reçue
- la monnaie de la somme reçue

à l'impression :
- la date (facultatif)
- le montant à payer
- l'indication de la monnaie dans laquelle on rend le dû
- la somme reçue et dans quelle monnaie elle est reçue
- le nombre de chaque valeur nécessaire pour un minimum de monnaie.

Exemple :

Ayant reçu 100.000 unités de monnaie française pour payer la somme de 39.000 unités de monnaie suisse, il faut rendre au cours du 20.03.1971 :

1 fois 20.00 unités de monnaie suisse
1 fois 10.00 unités de monnaie suisse
1 fois 5.00 unités de monnaie suisse
2 fois 2.00 unités de monnaie suisse
1 fois 1.00 unité de monnaie suisse

Indication :
Faire attention aux "bits" dans les transformations "flottant-entier" et vice-versa. (cf. page 60). ■

13.13. Résolution d'un système linéaire par la méthode de Gauss-Jordan.

Il s'agit d'une diagonalisation complète et systématique de matrices :

du système

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = c_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = c_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = c_3 \end{cases}$$

on passe au système :

$$\begin{cases} a'_{11}x_1 = c'_1 & x_1 = c'_1/a'_{11} \\ a'_{22}x_2 = c'_2 & x_2 = c'_2/a'_{22} \\ a'_{33}x_3 = c'_3 & x_3 = c'_3/a'_{33} \end{cases}$$

Exemple :

Posons $a_{i4} = c_i$ et considérons la matrice suivante :

$$\begin{matrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{matrix}$$

exemple numérique

$$\begin{matrix} 3 & -6 & 7 & 3 \\ 9 & 0 & -5 & 3 \\ 5 & -8 & 6 & -4 \end{matrix}$$

on appelle **pivot** le terme diagonal qui ne changera pas lors du calcul :

1er pivot : a_{11}
on remplace la ligne 2 par $a_{11} \cdot$ ligne 2 $- a_{21} \cdot$ ligne 1
et la ligne 3 par $a_{11} \cdot$ ligne 3 $- a_{31} \cdot$ ligne 1

(a_{11})	a_{12}	a_{13}	a_{14}	3	− 6	7	3
0	a'_{22}	a'_{23}	a'_{24}	0	54	− 78	− 18
0	a'_{32}	a'_{33}	a'_{34}	0	6	− 17	− 27

2e pivot : a'_{22}
on remplace la ligne 1 par $a'_{22} \cdot$ ligne 1 $- a'_{12} \cdot$ ligne 2
et la ligne 3 par $a'_{22} \cdot$ ligne 3 $- a'_{32} \cdot$ ligne 2

a'_{11}	0	a'_{13}	a'_{14}	162	0	− 90	54
0	(a'_{22})	a'_{23}	a'_{24}	0	54	− 78	− 18
0	0	a''_{33}	a''_{34}	0	0	− 450	− 1350

3e pivot : a''_{33}
on remplace la ligne 1 par $a''_{33} \cdot$ ligne 1 $- a'_{13} \cdot$ ligne 3
et la ligne 2 par $a''_{33} \cdot$ ligne 2 $- a'_{23} \cdot$ ligne 3

a''_{11}	0	0	a''_{14}	− 72900	0	0	− 145800
0	a''_{22}	0	a''_{24}	0	− 24300	0	− 97200
0	0	(a''_{33})	a''_{34}	0	0	− 450	− 1350

donc : $x_1 = a''_{14}/a''_{11}$ $\quad x_1 = 2$

$x_2 = a''_{24}/a''_{22}$ $\quad x_2 = 4$

$x_3 = a''_{34}/a''_{33}$ $\quad x_3 = 3$

Indications :

1. Il faut utiliser des variables indicées à double indice.
2. Cette méthode amplifie les nombres très rapidement, on a donc avantage à normaliser en divisant par exemple chaque fois par le pivot (qui sera non-nul, voir 3.).
3. Si le pivot est nul, la méthode n'a plus de sens, pour remédier à cela, il y a plusieurs façons :

a) intervertir des lignes,
b) additionner à la ligne où il y a le pivot nul une autre ligne, de façon à rendre le pivot non-nul.

13.14. Classer par ordre alphabétique une liste de noms.
Pour résoudre ce problème, il est indispensable d'avoir compris le principe du classement de nombres et de connaître l'ordre DIMENSIØN ainsi que les formats A et R.

Structure des "mots"
La mémoire de la CDC est organisée en mots de 48 bits (abréviation de "binary digits").
Ainsi, pour les nombres entiers, 47 positions sont réservées aux chiffres et une position au signe.

Exemple : Le nombre 9 sera représenté comme suit :

Rappel : $9_{10} = 1001_2$

Système octal
Un chiffre octal (0 à 7) occupe 3 positions de mémoire.

0	000	3	011	6	110
1	001	4	100	7	111
2	010	5	101		

Exemple :
Le nombre 3412 sera représenté comme suit :

Représentation des caractères
Un caractère est représenté par 2 chiffres octaux (2 fois 3 bits). On peut donc écrire $\frac{48}{6} = 8$ caractères par mot.

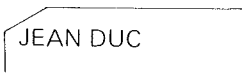
Quelques exemples de représentation du code caractère.

A	21	010	001
B	22	011	011
0	00	000	000
Ø	46	100	110
b	60	110	000

Exemple :

```
      READ (9, 100) A
100   FØRMAT (R10)
      .....
```

et la carte suivante



On a donc

J	E	A	N	b	D	U	C	b	b

en code caractère

41	25	21	45	60	24	64	23	60	60

en octal

100 001	010 101	010 001	100 101	110 000	010 100	110 100	010 011	110 000	110 000

Remarques

On constate donc que PAUL > JEAN PAUL > PAULE car b > E 60 > 25

Classement par ordre alphabétique

En utilisant des variables de type entier, le classement par ordre alphabétique sera comparable à un classement de nombres, à condition de travailler en format R pour éviter d'avoir des nombres négatifs (R7).

Problème posé par le blanc

Comparons les deux noms suivants : DE BOCCARD DEMIERRE

Le blanc étant plus grand que la lettre M, DEMIERRE sera donc classé avant DE BOCCARD. Il faut donc remédier à cette situation en remplaçant, par exemple, les blancs par des zéros. ■

13.15. a) Calculer la somme de deux matrices de type $<n;m>$.

Rappel :

$A = [\alpha_{ij}]$ $\quad B = [\beta_{ij}]$ deux matrices de type $<n;m>$.

$\alpha_{ij} + \beta_{ij} = \gamma_{ij}$ $\quad (i = 1 \text{ à } n,\ j = 1 \text{ à } m)$

$[\gamma_{ij}]$ étant la matrice C telle que $C = A + B$

Indications :

- le programme doit être le plus général possible, avec lecture des dimensions des deux matrices à chaque passage (n et m sont plus petits ou égaux à 10).
- vérifier la commutativité et l'associativité de cette opération.

b) Calculer le produit ligne-colonne de deux matrice de type $<n;p>$ et $<p;q>$.

Rappel :

$A = [\alpha_{ij}]$ $\quad (i = 1 \text{ à } n,\ j = 1 \text{ à } p)$

$B = [\beta_{jk}]$ $\quad (j = 1 \text{ à } p,\ k = 1 \text{ à } q)$

$\gamma_{ik} = \sum_{j=1}^{p} \alpha_{ij}\,\beta_{jk}$, $[\gamma_{ik}]$ étant la matrice C telle que $C = A \cdot B$

Indications :

- le programme doit être le plus général possible, avec lecture des dimensions des deux matrices à chaque passage (n, p et q sont plus petits ou égaux à 10).
- vérifier la commutativité et l'associativité de cette opération.

c) Ecrire un programme général d'algèbre matricielle pour déterminer les matrices suivantes :

1.	$A \cdot B$	4.	${}^tA \cdot {}^tB$	A, B sont 2 matrices
2.	$A \cdot {}^tB$	5.	$\alpha A + \beta B$	α, β sont 2 réels
3.	${}^tA \cdot B$	6.	${}^tA$	

Rappel : Si $A = [\alpha_{ij}]$, alors ${}^tA = [\alpha_{ji}]$

Indications :

- la dimension des matrices peut atteindre 10. A chaque passage, on lira un indice qui permettra de sélectionner l'opération désirée.
- l'emploi des GØTØ calculés et/ou des SUBRØUTINES est recommandé.

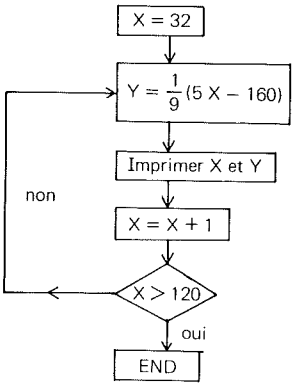
■

CHAPITRE 14

CORRIGÉS D'EXERCICES

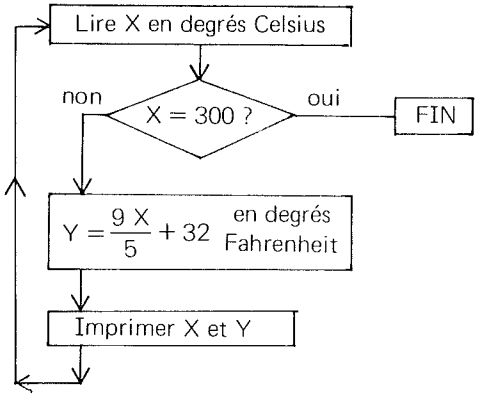
Le lecteur se rendra compte en étudiant les corrigés de certains exercices que des solutions plus rationnelles auraient pu être envisagées. C'est à dessein que les auteurs ont encouragé les élèves à trouver des solutions personnelles qui, si elles ne sont pas toujours les plus concises n'en ont pas moins permis d'arriver à une solution valable. Les auteurs tiennent à exprimer leur très vive reconnaissance aux élèves pour le travail considérable qu'ils ont effectué.

2.1.

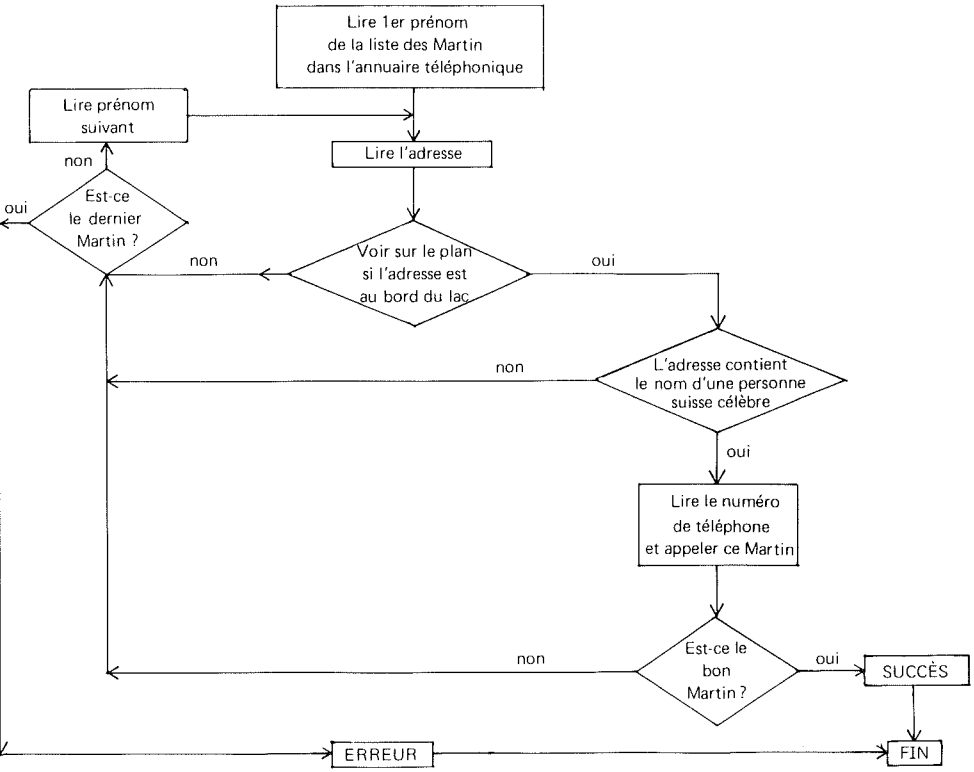


2.2. *Rappel* $F : X \to \frac{5}{9}(X - 32)$ $\qquad F^{-1} : X \to \frac{9X}{5} + 32$

Remarque : pour savoir quand on a épuisé la série de valeurs de X en degrés Celsius à transformer en degrés Fahrenheit, on indiquera une valeur factice supplémentaire (ici 300°C) qui permettra de tester la fin du problème.


Lire X en degrés Celsius
non
X = 300 ?
oui
FIN
Y = 9 X / 5 + 32
en degrés Fahrenheit
Imprimer X et Y

2.3.


Lire 1er prénom de la liste des Martin dans l'annuaire téléphonique
Lire prénom suivant
Lire l'adresse
non
oui
Est-ce le dernier Martin ?
non
Voir sur le plan si l'adresse est au bord du lac
oui
non
L'adresse contient le nom d'une personne suisse célèbre
oui
Lire le numéro de téléphone et appeler ce Martin
non
Est-ce le bon Martin ?
oui
SUCCÈS
ERREUR
FIN

3.2. Les constantes entières sont :

17 1000 −10501 12345678 ▲5

3.3. Les constantes flottantes sont :

−.017 40193. 1.001 0. 1.0 2734.4712 ▲5

3.4. Les variables entières sont :

ITR35 N NØW
KING7 NY

3.5. Les variables flottantes sont :

V8 DØG SUM X A 5523 DS21

4.2.

(travail de l'élève D. Pfenniger)

TABLE TRIGONOMETRIQUE

DEGRE	RADIAN	SINUS	COSINUS	TANGENTE	COTANGENTE	SECANTE	COSECANTE		
0.	0.000000	0.000000	1.000000	0.000000	INFINI	1.00000	INFINI	1.57080	90.
1.	0.017453	0.017452	0.999847	0.017455	57.29028	1.00015	57.29901	1.55334	89.
2.	0.034906	0.034899	0.999391	0.034921	28.63641	1.00061	28.65387	1.53589	88.
3.	0.052360	0.052336	0.998629	0.052407	19.08124	1.00137	19.10743	1.51844	87.
4.	0.069813	0.069756	0.997564	0.069926	14.30075	1.00244	14.33567	1.50098	86.
5.	0.087266	0.087155	0.996194	0.087488	11.43011	1.00382	11.47378	1.48353	85.
6.	0.104719	0.104528	0.994522	0.105104	9.51442	1.00551	9.56683	1.46608	84.
7.	0.122172	0.121869	0.992546	0.122784	8.14439	1.00751	8.20556	1.44862	83.
8.	0.139626	0.139172	0.990268	0.140540	7.11541	1.00983	7.18534	1.43117	82.
9.	0.157079	0.156434	0.987688	0.158384	6.31379	1.01247	6.39249	1.41372	81.
10.	0.174532	0.173647	0.984808	0.176326	5.67132	1.01543	5.75880	1.39626	80.
11.	0.191985	0.190808	0.981627	0.194379	5.14458	1.01872	5.24087	1.37881	79.
12.	0.209438	0.207910	0.978148	0.212555	4.70466	1.02234	4.80976	1.36136	78.
13.	0.226892	0.224950	0.974370	0.230867	4.33150	1.02630	4.44544	1.34390	77.
14.	0.244345	0.241921	0.970296	0.249327	4.01080	1.03061	4.13359	1.32645	76.
15.	0.261798	0.258818	0.965926	0.267948	3.73207	1.03528	3.86373	1.30900	75.
16.	0.279251	0.275636	0.961262	0.286744	3.48744	1.04030	3.62798	1.29155	74.
17.	0.296704	0.292370	0.956305	0.305729	3.27087	1.04569	3.42032	1.27409	73.
18.	0.314157	0.309015	0.951057	0.324918	3.07770	1.05146	3.23609	1.25664	72.
19.	0.331611	0.325566	0.945519	0.344325	2.90423	1.05762	3.07157	1.23919	71.
20.	0.349064	0.342018	0.939693	0.363968	2.74749	1.06418	2.92382	1.22173	70.
21.	0.366517	0.358366	0.933581	0.383862	2.60511	1.07116	2.79044	1.20428	69.
22.	0.383970	0.374604	0.927185	0.404024	2.47510	1.07853	2.66948	1.18683	68.
23.	0.401423	0.390729	0.920506	0.424472	2.35587	1.08636	2.55932	1.16937	67.
24.	0.418877	0.406734	0.913546	0.445226	2.24605	1.09464	2.45861	1.15192	66.
25.	0.436330	0.422616	0.906309	0.466305	2.14452	1.10338	2.36621	1.13447	65.
26.	0.453783	0.438369	0.898795	0.487729	2.05032	1.11260	2.28118	1.11701	64.
27.	0.471236	0.453988	0.891008	0.509522	1.96262	1.12232	2.20270	1.09956	63.
28.	0.488689	0.469469	0.882949	0.531706	1.88074	1.13257	2.13007	1.08211	62.
29.	0.506143	0.484807	0.874621	0.554305	1.80406	1.14335	2.06268	1.06465	61.
30.	0.523596	0.499997	0.866027	0.577346	1.73206	1.15470	2.00001	1.04720	60.
31.	0.541049	0.515035	0.857169	0.600856	1.66429	1.16663	1.94161	1.02975	59.
32.	0.558502	0.529917	0.848050	0.624865	1.60035	1.17918	1.88709	1.01229	58.
33.	0.575955	0.544636	0.838672	0.649403	1.53988	1.19236	1.83609	0.99484	57.
34.	0.593409	0.559190	0.829039	0.674503	1.48257	1.20622	1.78830	0.97739	56.
35.	0.610862	0.573573	0.819154	0.700202	1.42816	1.22077	1.74346	0.95993	55.
36.	0.628315	0.587782	0.809019	0.726537	1.37639	1.23606	1.70131	0.94248	54.
37.	0.645768	0.601812	0.798638	0.753548	1.32706	1.25213	1.66165	0.92503	53.
38.	0.663221	0.615658	0.788013	0.781279	1.27995	1.26901	1.62428	0.90757	52.
39.	0.680675	0.629317	0.777148	0.809778	1.23491	1.28676	1.58902	0.89012	51.
40.	0.698128	0.642784	0.766047	0.839093	1.19176	1.30540	1.55573	0.87267	50.
41.	0.715581	0.656056	0.754712	0.869280	1.15038	1.32501	1.52426	0.85522	49.
42.	0.733034	0.669127	0.743148	0.900396	1.11062	1.34563	1.49448	0.83776	48.
43.	0.750487	0.681995	0.731357	0.932507	1.07238	1.36732	1.46629	0.82031	47.
44.	0.767941	0.694655	0.719343	0.965680	1.03554	1.39016	1.43956	0.80286	46.
45.	0.785394	0.707103	0.707110	0.999991	1.00001	1.41421	1.41422	0.78540	45.
		COSINUS	SINUS	COTANGENTE	TANGENTE	COSECANTE	SECANTE	RADIAN	DEGRE

POUR LES VALEURS DE 0 A 45 DEGRES, LIRE LES TITRES DU HAUT DE LA TABLE.
POUR LES VALEURS DE 45 A 90 DEGRES, LIRE CEUX DU BAS.

TABLE NUMERIQUE DE -20 A +20

X	INVERSE DE X	CARRE DE X	RACINE CARREE DE X	RACINE CARREE DE 10X	CUBE DE X	RACINE CUBIQUE DE X	RACINE CUBIQUE DE 10X	RACINE CUBIQUE DE 100X	LOG NATUREL DE X	LOG DECIMAL DE X
-20.	-0.050	400.			-8000.	-2.7144	-5.8480	-12.5992		
-19.	-0.053	361.			-6859.	-2.6684	-5.7489	-12.3856		
-18.	-0.056	324.			-5832.	-2.6207	-5.6462	-12.1644		
-17.	-0.059	289.			-4913.	-2.5713	-5.5397	-11.9348		
-16.	-0.062	256.			-4096.	-2.5198	-5.4288	-11.6961		
-15.	-0.067	225.			-3375.	-2.4662	-5.3133	-11.4471		
-14.	-0.071	196.			-2744.	-2.4101	-5.1925	-11.1869		
-13.	-0.077	169.			-2197.	-2.3513	-5.0658	-10.9139		
-12.	-0.083	144.			-1728.	-2.2894	-4.9324	-10.6266		
-11.	-0.091	121.			-1331.	-2.2240	-4.7914	-10.3228		
-10.	-0.100	100.			-1000.	-2.1544	-4.6416	-10.0000		
-9.	-0.111	81.			-729.	-2.0801	-4.4814	-9.6549		
-8.	-0.125	64.			-512.	-2.0000	-4.3089	-9.2832		
-7.	-0.143	49.			-343.	-1.9129	-4.1213	-8.8790		
-6.	-0.167	36.			-216.	-1.8171	-3.9149	-8.4343		
-5.	-0.200	25.			-125.	-1.7100	-3.6840	-7.9370		
-4.	-0.250	16.			-64.	-1.5874	-3.4199	-7.3681		
-3.	-0.333	9.			-27.	-1.4422	-3.1072	-6.6943		
-2.	-0.500	4.			-8.	-1.2599	-2.7144	-5.8480		
-1.	-1.000	1.			-1.	-1.0000	-2.1544	-4.6416		
0.		0.	0.0000	0.0000	0.	0.0000	0.0000	0.0000		
1.	1.000	1.	1.0000	3.1623	1.	1.0000	2.1544	4.6416	0.00000	0.00000
2.	0.500	4.	1.4142	4.4721	8.	1.2599	2.7144	5.8480	0.69315	0.30103
3.	0.333	9.	1.7321	5.4772	27.	1.4422	3.1072	6.6943	1.09861	0.47712
4.	0.250	16.	2.0000	6.3246	64.	1.5874	3.4199	7.3681	1.38629	0.60206
5.	0.200	25.	2.2361	7.0711	125.	1.7100	3.6840	7.9370	1.60944	0.69897
6.	0.167	36.	2.4495	7.7460	216.	1.8171	3.9149	8.4343	1.79176	0.77815
7.	0.143	49.	2.6458	8.3666	343.	1.9129	4.1213	8.8790	1.94591	0.84510
8.	0.125	64.	2.8284	8.9443	512.	2.0000	4.3089	9.2832	2.07944	0.90309
9.	0.111	81.	3.0000	9.4868	729.	2.0801	4.4814	9.6549	2.19722	0.95424
10.	0.100	100.	3.1623	10.0000	1000.	2.1544	4.6416	10.0000	2.30258	1.00000
11.	0.091	121.	3.3166	10.4881	1331.	2.2240	4.7914	10.3228	2.39789	1.04139
12.	0.083	144.	3.4641	10.9545	1728.	2.2894	4.9324	10.6266	2.48491	1.07918
13.	0.077	169.	3.6056	11.4018	2197.	2.3513	5.0658	10.9139	2.56495	1.11394
14.	0.071	196.	3.7417	11.8322	2744.	2.4101	5.1925	11.1869	2.63906	1.14613
15.	0.067	225.	3.8730	12.2474	3375.	2.4662	5.3133	11.4471	2.70805	1.17609
16.	0.062	256.	4.0000	12.6491	4096.	2.5198	5.4288	11.6961	2.77259	1.20412
17.	0.059	289.	4.1231	13.0384	4913.	2.5713	5.5397	11.9348	2.83321	1.23045
18.	0.056	324.	4.2426	13.4164	5832.	2.6207	5.6462	12.1644	2.89037	1.25527
19.	0.053	361.	4.3589	13.7840	6859.	2.6684	5.7489	12.3856	2.94444	1.27875
20.	0.050	400.	4.4721	14.1421	8000.	2.7144	5.8480	12.5992	2.99573	1.30103

6.1. a) `X = A + B + C - 3.`

b) `S = (A*R - A*R**N) / (1.-R)`

c) `X = A*R**(N-1)`

d) `W = 3.*(X + Y)`

e) `X = (Z**3 + Y)**(-3))/100.`

f)
```
B = (N + 3 - 6*MØD (N,2))
C = (N - 2 + 4*MØD (N,2))
A = B / C
```

6.2. a) $6\,b^5$

b) $x - y - (z - w)$ ou $x - y - z + w$

c) $(x - y)^2\,(y - z)^2$

d) $q = \dfrac{x}{y \cdot z}$

j) $j = 6\,k + 4\,i_1 \cdot m_2$

k) impossible

l) $m = 12$

6.3. a) `Y = 3*X + B`

b) le membre de gauche doit être une variable

c) `C = ((X + Y)*B**3) + (P - Q)**2`

d) `Y = 1.624009*DEL`

e) `K = I**2.`

f) `BX6 = 1. /-2.*B**2`

6.4. a) `X = 5.2` b) `Y = 3.1`

6.5. `A = 2.6`

6.6.

a)	13.	(flottant)	f)	2.	(flottant)
b)	0.	(flottant)	g)	2	(fixe)
c)	0	(fixe)	h)	4	(fixe)
d)	3.	(flottant)	i)	6.	(flottant)
e)	3	(fixe)	n)	8	(fixe)
			o)	5	(fixe)

6.7. S = 100./2.*(2.*7.9 + 99.*DELTA) ou mieux

S = 50.*(15.8 + 99.*DELTA) ou encore

S = 680.+4950.*DELTA

7.1. 7.2. 7.3. *(travail de l'élève P. Rey)*

a) programme

```
      WRITE (9,150)
      SOMME=0.
      PRODUI=1.
      X1=0.
    1 READ(9,200)DONNEE
      IF(DONNEE-10000.)2,3,2
    2 SOMME=SOMME+DONNEE
      PRODUI=PRODUI*DONNEE
      X1=X1+1.
      GOTO 1
    3 ARITHM=SOMME/X1
      IF(PRODUI)4,5,6
    4 IF(AMOD(X1,2)-1)42,41,42
   41 Y=(-PRODUI)**(1./X1)
      GEOMET=-Y
      WRITE (9,100)X1,SOMME,ARITHM,PRODUI,GEOMET
      GOTO 7
   42 WRITE (9,101)X1,SOMME,ARITHM,PRODUI
      GOTO 7
    5 WRITE (9,102)X1,SOMME,ARITHM,PRODUI
      GOTO 7
    6 GEOMET=PRODUI**(1./X1)
      WRITE (9,100)X1,SOMME,ARITHM,PRODUI,GEOMET
      GOTO 7
    7 STOP
  150 FORMAT(6X,12(1H*)/6X,1H*,10X1H*/6X,1H*,1X'MOYENNES',1X1H*/6X,1H*,1
     +0X1H*/6X,12(1H*)////1X'DONNEES'/1X7(1H*)//)
  151 FORMAT(1XF7.1)
  100 FORMAT(//1X'LE NB. DE DONNEES=',F4.0/1X'LA SOMME DES DONNEES
     +=',F15.3/1X'LA MOYENNE ARITHMETIQUE=',F15.3/1X'LE PRODUI DES
     + DONNEES=',F20.3/1X'LA MOYENNE GEOMETRIQUE=',F15.3)
  101 FORMAT(//1X'LE NB. DE DONNEES=',F4.0/1X'LA SOMME DES DONNEES
     +=',F15.3/1X'LA MOYENNE ARITHMETIQUE=',F15.3/1X'LE PRODUIT DES
     + DONNEES=',F20.3/1X'LE MOYENNE GEOMETRIQUE N EXISTE PAS')
  102 FORMAT(//1X'LE NB. DE DONNEES=',F4.0/1X'LA SOMME DES DONNEES
     +=',F15.3/1X'LA MOYENNE ARITHMETIQUE=',F15.3/1X'LE PRODUIT DES
     + DONNEES=',F20.3/1X'LA MOYENNE GEOMETRIQUE=0')
  200 FORMAT(F7.1)
      END
```

b) résultats

```
      ************
      *          *
      * MOYENNES *
      *          *
      ************


 DONNEES
 *******

! 5.3
! .8
! 5.4
! 4.2
! 4.8
! 5.5
! 10000.


 LE NB. DE DONNEES=  6.
 LA SOMME DES DONNEES      =          26.000
 LA MOYENNE ARITHMETIQUE=           4.333
 LE PRODUI DES       DONNEES=             2538.707
 LA MOYENNE GEOMETRIQUE=          3.693
```

7.4. a) programme *(travail de l'élève M. Portier)*

```
*     RESOLUTION DE L EQUATION DU DEUXIEME DEGRE AX2+BX+C
1     READ (9,100)A,B,C
      IF(A-10000.)10,515,10
10    IF(A)2,5,2
2     D=B**2-4.*A*C
      IF(D)4,3,3
3     U=(-B+SQRT(D))/(2.*A)
      V=(-B-SQRT(D))/(2.*A)
      WRITE (9,200)A,B,C,D,U,V
      GOTO1
4     WRITE (9,300)A,B,C,D
      GOTO1
5     IF(B)6,7,6
6     Z=-C/B
      WRITE (9,400)A,B,C,Z
      GOTO1
7     IF(C)8,9,8
8     WRITE (9,500)A,B,C
      GOTO1
9     WRITE (9,600)
      GOTO1
515   STOP
100   FORMAT(3F10.3)
200   FORMAT(///1X'L EQUATION EST'/
     +16(1H*)/
     +F10.3,' X**2 +',F10.3,' X +',F10.3,3X'=0'/
     +'LE DISCRIMINANT EST',F10.3/
     +21(1H*)/
     +,'LES SOLUTIONS SONT'/
     +19(1H*)/
     +,'A)',F10.3/
     +,'B)',F10.3/)
300   FORMAT(///1X'L EQUATION EST'/
     +16(1H*)/
     +F10.3,' X**2 +',F10.3,' X +',F10.3,3X'=0'/
     +'LE DISCRIMINANT EST',F10.3/
     +21(1H*)/
     +'IL EST NEGATIF,DONC LES DEUX SOLUTIONS'
     +' NE SONT PAS REELLES'/)
400   FORMAT(///1X'L EQUATION EST'/
     +16(1H*)/
     +F10.3,' X**2 +',F10.3,' X +',F10.3,3X,'=0'/
     +,'A EST NUL,L EQUATION DEVIENT BX+C'/
     +,'LA SOLUTION EST ',F10.3/
     +16(1H*)/)
500   FORMAT(///1X'L EQUATION EST'/
     +16(1H*)/
     +F10.3,' X**2 +',F10.3,' X +',F10.3,3X,'=0'/
     +,'A ET B SONT NULS,L EQUATION RESTE DONC C=0'/
     +'SI C N EST PAS NUL,CETTE EXPRESSION N A PAS DE SENS,DONC'/
     +  '/
     +'PAS DE SOLUTION'/)
600   FORMAT(///1X'A,B ET C SONT NULS,L EQUATION RESTE DONC 0=0'/)
      END
```

7.5. a) résultats

(travail de l'élève D. Lavanchy)

```
*******************************
*                             *
* DETERMINATION DU NOMBRE PI  *
* SELON PLUSIEURS METHODES    *
*                             *
*******************************
```

FORMULES UTILISEES	PI AVEC DOUBLE PRECISION	PI SANS DOUBLE PRECISION	NOMBRE D ITERATIONS
FORMULE DE WALLIS MODIFIEE	3.14159265358979323846264 +000	3.14159265358466666	30
FORMULE ALTERNEE D EULER	3.14159265358979330574961+000	3.14159265352645889	30
FORMULE DE BERNOUILLI	3.13682630633096795439279+000	3.13682630640688742	200
METHODE DE LA TANGENTE	3.13909265749601272146624+000	3.13909265759866688	400
FRACTIONS APPROCHEES : 22/7	3.14285714285714285714286+000	3.14285714284051211	
355/113	3.14159292035398230088496+000	3.14159292035037665	
(167/80)+(SQRT(10)/3)	3.14159255338945977733296+000	3.14159255338510745	
FORMULE DE WALLIS	3.13963222192939632747202+000	3.13963222748134291	400
FORMULE DE VIETE	3.14159265358979323734208+000	3.14159265358466666	30
AUTRE FORMULE	3.14159265330115972300802+000	3.14159265335183559	30

VALEUR DE PI SELON LA TABLE	3.14159265358979323846264338327...

7.7.

```
      IF (X-Y) 11,11,12
11    GRAND=Y
      GØTØ 15
12    GRAND=X
15    continuation du programme.
```

ou

```
      GRAND=X
      IF (X-Y) 11,14,14
11    GRAND=Y
14    continuation du programme.
```

ou

```
      GRAND=X
      IF (X.LT.Y) GRAND=Y
      .........
```

7.9.

```
      ABSX=ABS (X)
      ABSY=ABS (Y)
      IF (ABSX-ABSY) 11,11,12
11    XMAX=ABSY
      GØTØ 13
12    XMAX=ABSX
13    continuation du programme.
```

7.10.

```
4     IF (ALPHA-6.2832) 2,1,1
1     ALPHA=ALPHA-6.2832
      GØTØ 4
2     continuation du programme.
```

7.12. a)

```
      IF (ABS(XNØM).LT.7.) GØTØ 80
71    ......
```

b)

```
      IF (XNØM.LT.7..AND.XNØM.GT.-7.) GØTØ 80
71    ......
```

c)

```
      IF (XNØM-7.) 12,71,71
12    IF (XNØM+7.) 71,71,80
80    ......
      ......
      GØTØ 92
71    ......
      ......
92    continuation du programme.
```

7.14. a)

```
      IF (K-2) 69,6,69
6     IF (P-Q) 17,19,19
```

b)

```
      IF (K.NE.2) GØTØ 69
      IF (P.LT.Q) GØTØ17
19    ......
```

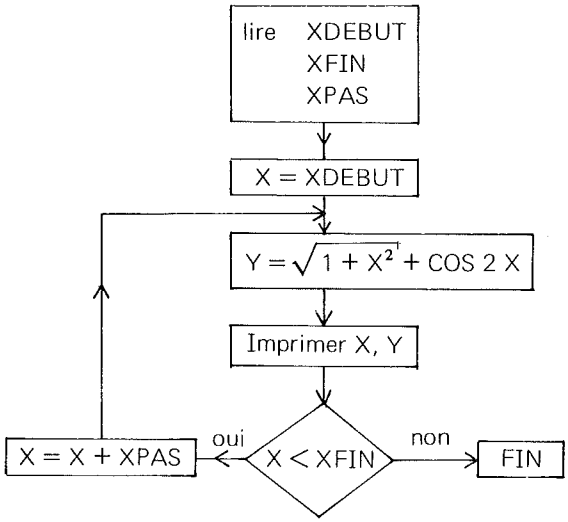
7.17.

```
      IF (X-A) 42,79,41
41    IF (X-B) 79,79,42
```

ou, mieux, sans l'hypothèse que $a < b$

```
      IF ((X-A)*(X-B)) 79,79,42
```

7.18.

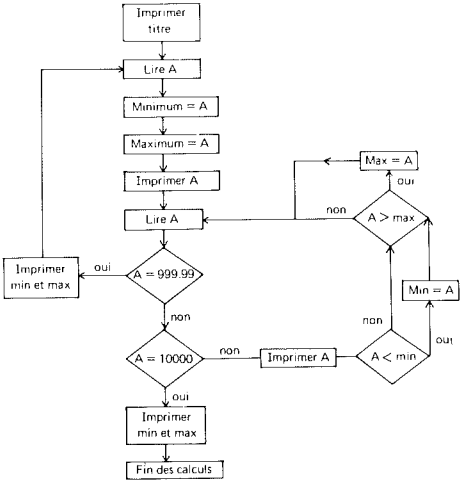


7.19.

LEG est l'indice du polynôme

```
      .
      .
      .
      .
      LEGI=LEG+1
      GØTØ (61,62,63,64,65) LEGI
61    P0=1.
      GØTØ 70
62    P1=X
      GØTØ 70
63    P2=1.5*X*X*-.5
      GØTØ 70
64    P3=2.5*X**3-1.5*X
      GØTØ 70
65    P4=4.75*X**4-3.75*X**2+.375
70    ......
```

7.20. a) organigramme *(travail de l'élève G. Pongratz)*



b) programme

```
*      RECHERCHE DU PLUS PETIT ET DU PLUS GRAND NOMBRES D'UNE SUITE
*      DE NOMBRES
       WRITE (9, 99)
     1 READ(9,100)A
       PIN=A
       PAX=A
     2 READ(9,100)A
       IF(A-999.99)4,3,4
     3 WRITE (9, 102)PIN,PAX
       GOTO 1
     4 IF(A-10000.)7,5,7
     7 IF(A-PIN)11,6,6
    11 PIN=A
     6 IF(A-PAX)2,2,10
    10 PAX=A
       GOTO 2
     5 WRITE (9, 102)PIN,PAX
       STOP
    99 FORMAT('RECHERCHE DU PLUS PETIT ET DU PLUS GRAND NOMBRES'/
      +'D UNE SUITE DE NOMBRES A'///'NOMBRE A',
      +/8(1H*),//)
   100 FORMAT(F6.2)
   102 FORMAT(40X,'LE PLUS PETIT A EST ',F6.2,//40X,
      +'LE PLUS GRAND A EST',F6.2,//72(1H*)///)
       END
```

c) résultats

```
RECHERCHE DU PLUS PETIT ET DU PLUS GRAND NOMBRES
D UNE SUITE DE NOMBRES A


NOMBRE A
********

! 2.50
! 10.00
! -3.75
! 10.00
! 6.55
! 999.99

                                        LE PLUS PETIT A EST  -3.75

                                        LE PLUS GRAND A EST 10.00

*************************************************************************


! 3.
! 6.
! -6.
! 9.
! 10000.

                                        LE PLUS PETIT A EST  -6.00

                                        LE PLUS GRAND A EST  9.00

*************************************************************************
```

7.21. a) programme *(travail de l'élève S. Albertani)*

```
*    ETABLISSEMENT DE LA TABLE DES
*    FACTORIELLES DE 0 A 30
     DOUBLE PRECISION P
     WRITE (9, 100)
     AN=0.
     N=0
     P=1.D0
     WRITE (9,101)N,P
   1 AN=AN+1.
     P=P*AN
     N=AN
     WRITE (9,101)N,P
     IF(AN-30.)3,2,3
   3 GOTO 1
   2 STOP
 100 FORMAT(10X'TABLE DES FACTORIELLES DE 0 A 30'/10X32(1H*)//)
 101 FORMAT(I5,'!   =  '5XD18.12//)
     END
```

b) résultats

```
          TABLE DES FACTORIELLES DE 0 A 30
          ********************************

 0!   =        0.999999999999D 00

 1!   =        0.999999999999D 00

 2!   =        0.200000000000D 01

 3!   =        0.600000000000D 01

 4!   =        0.240000000000D 02

 5!   =        0.120000000000D 03
 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
26!   =        0.4032914611191 27

27!   =        0.108888694502D 29

28!   =        0.304888344605D 30

29!   =        0.884176199353D 31

30!   =        0.265252859805D 33
```

8.1.

```
      PRØGRAM EXMATH
      X=-10.
      DØ 17 K=1,41
      Y=20.*X/(X**2+4.)
      WRITE (9,100)X,Y
17    X=X+0.5
      STØP
100   FØRMAT (F10.1,F15.6)
      END
```

8.2. *(travail de l'élève P. Rey)*

```
      WRITE (9,150)
      SOMME=0.
      PRODUI=1.
      DO 33 I=1,100
      READ (9,200) DONNEE
      IF(DONNEE-10000.)1,2,1
    1 SOMME=SOMME+DONNEE
      PRODUI=PRODUI*DONNEE
   33 COMPTE=I
    2 ARITHM=SOMME/COMPTE
      IF(PRODUI)4,5,6
    4 IF(AMOD(COMPTE,2)-1)42,41,42
   41 Y=(-PRODUI)**(1./COMPTE)
      GEOMET=-Y
      WRITE (9,100)COMPTE,SOMME,ARITHM,PRODUI,GEOMET
      GOTO 7
   42 WRITE (9,101)COMPTE,SOMME,ARITHM,PRODUI
      GOTO 7
    5 WRITE (9,102)COMPTE,SOMME,ARITHM,PRODUI
      GOTO 7
    6 GEOMET=PRODUI**(1./COMPTE)
      WRITE (9,100)COMPTE,SOMME,ARITHM,PRODUI,GEOMET
      GOTO 7
    7 STOP
  150 FORMAT(6X,12(1H*)/6X,1H*,10X1H*/6X,1H*,1X'MOYENNES',1X1H*/6X,1H*,1
     +0X1H*/6X,12(1H*)////1X'DONNEES'/1X7(1H*)//)
  151 FORMAT(1XF7.1)
  100 FORMAT(//1X'LE NB. DE DONNEES=',F4.0/1X'LA SOMME DES DONNEES
     +=',F15.3/1X'LA MOYENNE ARITHMETIQUE=',F15.3/1X'LE PRODUI DES
     + DONNEES=',F20.3/1X'LA MOYENNE GEOMETRIQUE=',F15.3)
  101 FORMAT(//1X'LE NB. DE DONNEES=',F4.0/1X'LA SOMME DES DONNEES
     +=',F15.3/1X'LA MOYENNE ARITHMETIQUE=',F15.3/1X'LE PRODUIT DES
     + DONNEES=',F20.3/1X'LE MOYENNE GEOMETRIQUE N EXISTE PAS')
  102 FORMAT(//1X'LE NB. DE DONNEES=',F4.0/1X'LA SOMME DES DONNEES
     +=',F15.3/1X'LA MOYENNE ARITHMETIQUE=',F15.3/1X'LE PRODUIT DES
     + DONNEES=',F20.3/1X'LA MOYENNE GEOMETRIQUE=0')
  200 FORMAT(F7.1)
      END
```

Remarque : la valeur 100 pour m_2 de la boucle DØ 33 I=1,100 doit être plus grande que le nombre de cartes à lire.

8.3. Comparaison d'un programme avec et sans boucle DØ

```
*       F(X)=X*X - 28*X + 147.
        X=1.
    1   Y=X*X-28.*X+147.
        WRITE (9,100)X,Y
        X=X+2.
        IF(X-29.)1,1,9
    9   STOP
  100   FORMAT(1X,'POUR X =',F3.0,3X'Y = ',F8.2)
        END
```

```
*       F(X)=X*X - 28*X + 147.
        DO 1 I=1,29,2
        X=I
        Y=X*X-28.*X+147.
    1   WRITE (9, 100)X,Y
  100   FORMAT(1X,'POUR X =',F3.0,3X'Y = ',F8.2)
        STOP
        END
```

```
 POUR X =  1.    Y =    120.00
 POUR X =  3.    Y =     72.00
 POUR X =  5.    Y =     32.00
 POUR X =  7.    Y =      0.00
 POUR X =  9.    Y =    -24.00
 POUR X = 11.    Y =    -40.00
 POUR X = 13.    Y =    -48.00
 POUR X = 15.    Y =    -48.00
 POUR X = 17.    Y =    -40.00
 POUR X = 19.    Y =    -24.00
 POUR X = 21.    Y =      0.00
 POUR X = 23.    Y =     32.00
 POUR X = 25.    Y =     72.00
 POUR X = 27.    Y =    120.00
 POUR X = 29.    Y =    176.00

 ?
```

8.4. *(travail de l'élève P. Rey)*

```
      DO 1 I=20,30
      X=FLOAT(I)/10.
      DO 1 J=10,12
      Y=J
      DO 1 K=120,195,25
      Z=FLOAT(K)/100.
      FXYZ=EXP(X)-(2.*Y)*(Z**2)
    1 WRITE (9, 100)X,Y,Z,FXYZ
      STOP
  100 FORMAT(4(E12.5,5X))
      END
```

Remarque : la fonction FLØAT transforme en flottant une variable ou un nombre entier.

9.1. Il y aura impression des 7200 variables de la matrice A (800,9), en prenant d'abord les 800 éléments de la première colonne, puis les 800 éléments de la deuxième colonne, et ainsi de suite jusqu'aux 800 derniers éléments de la neuvième colonne.

11.1.

```
IF (Y.GE.X.AND.Y.LE.Z) L=.TRUE.
```

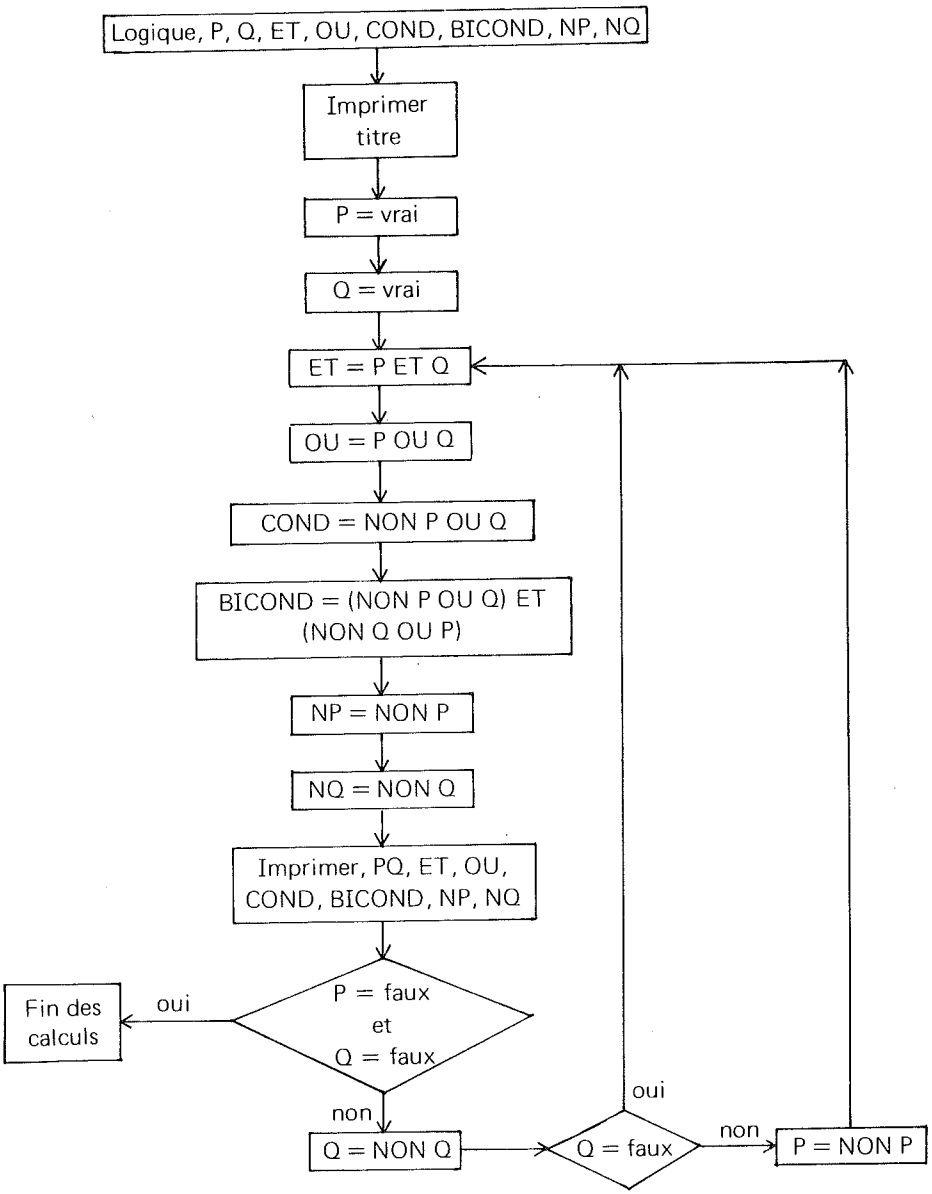
ou

```
IF (X.LE.Y.AND.Z.GE.Y) L=.TRUE.
```

11.2.

```
IF (R.AND..NØT.Q.AND.Q.AND..NØT.R) P=.FALSE.
```

11.4. a) organigramme



b) programme

```
*      TABLE LOGIQUE
       LOGICAL P,Q,ET,OU,COND,BICOND,NP,NQ
       WRITE (9,101)
       P=.TRUE.
       Q=.TRUE.
     1 ET=P.AND.Q
       OU=P.OR.Q
       COND=.NOT.P.OR.Q
       BICOND=(.NOT.P.OR.Q).AND.(.NOT.Q.OR.P)
       NP=.NOT.P
       NQ=.NOT.Q
       WRITE(9,100)P,Q,ET,OU,COND,BICOND,NP,NQ
       IF(.NOT.(P.OR.Q))GOTO 2
       Q=.NOT.Q
       IF(.NOT.Q)GOTO 1
       P=.NOT.P
       GOTO 1
     2 STOP
101    FORMAT(22X,'TABLES DES VALEURS',///1X,69(1H*)/1X,1H*,'   P
      + '
      +'   Q    ',1H*,' P ET Q ',1H*,' P OU Q ',1H*,'  P=>Q  ',1H*,
      +' P<=>Q  ',1H*,' NON P ',1H*,' NON Q ',1H*/1X,69(1H*))
100    FORMAT(1X,1H*,2(L4,3X,1H*),4(L4,4X,1H*),2(L4,3X,1H*),/1X,69(
      +1H
       END
```

c) résultats *(travail de l'élève G. Pongratz)*

TABLES DES VALEURS

```
**************************************************************************
*   P   *   Q   * P ET Q * P OU Q *  P=>Q  * P<=>Q  * NON P * NON Q *
**************************************************************************
*   T   *   T   *   T    *   T    *   T    *   T    *   F   *   F   *
**************************************************************************
*   T   *   F   *   F    *   T    *   F    *   F    *   F   *   T   *
**************************************************************************
*   F   *   T   *   F    *   T    *   T    *   F    *   T   *   F   *
**************************************************************************
*   F   *   F   *   F    *   F    *   T    *   T    *   T   *   T   *
**************************************************************************
```

Remarque : On pourrait reprendre ce programme en utilisant des boucles DØ.

12.4. a) **Intégration par les fonctions en escalier** *(travail de l'élève G. Ineichen)*

```
      SUBROUTINE ESCALI(AMIN,AMAX,PAS,AIRE)
*     (MINIMUM,MAXIMUM,PAS,INTEGRALE(VALEUR DE RETOUR))
*
      AIRE=0.
      X=AMIN
    1 Y=FINT(X)
      AIRE=AIRE+PAS*Y
      X=X+PAS
      IF(X.GT.AMAX)GOTO 1
    2 RETURN
      END
```

Pour l'appel, il faut donner dans l'ordre, en flottant :

1. la valeur minimale, point de départ de l'intervalle d'intégration
2. la valeur maximale de l'intervalle d'intégration
3. le pas
4. une variable dans laquelle sera placée la valeur calculée par le sous-programme.

La fonction pour le calcul est donnée par la FUNCTIØN FINT(X).
A déterminer par l'utilisateur.

b) **Intégration par la méthode des trapèzes**

```
      SUBROUTINE TRAPEZ(AMIN,AMAX,PAS,AIRE)
*     (MINIMUM,MAXIMUM,PAS,INTEGRALE(VALEUR DE RETOUR))
*
      T=FINT(AMIN)
      Z=FINT(AMAX)
      SURF=T+Z
      X=AMIN
    3 X=X+PAS
      IF(X.GT.AMAX-PAS)GOTO 1
    2 SURF=SURF+2.*FINT(X)
      GO TO 3
    1 AIRE=(PAS/2.)*SURF
      RETURN
      END
```

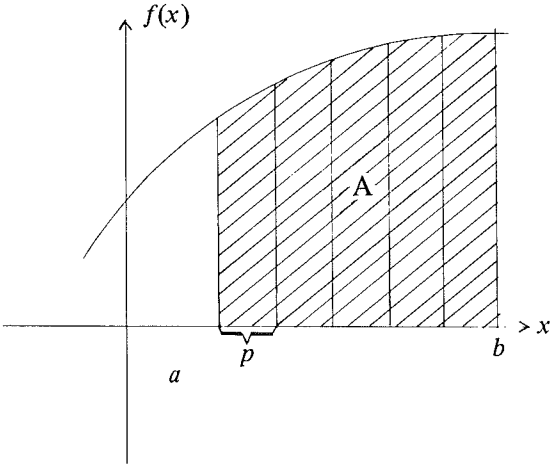
Pour l'appel, il faut donner dans l'ordre, en flottant, (sans contrainte de format) :

1. la valeur minimale, point de départ de l'intervalle d'intégration
2. la valeur maximale de l'intervalle d'intégration
3. le pas
4. une variable dans laquelle sera placée la valeur calculée par la sous-routine.

La fonction pour le calcul est donnée par la FUNCTIØN FINT(X).
A déterminer par l'utilisateur.

Remarques :

1. on pourrait aussi parler de l'intégration par la méthode de Simpson (cf n'importe quel livre élémentaire d'analyse numérique).
2. l'intégration avec la méthode des trapèzes remplace la somme de rectangles par une somme de trapèzes.



L'aire A est donnée par :

$$\int_a^b f(x)\,dx = A = \frac{p}{2}[f(a) + 2f(a+p) + 2f(a+2p) + 2f(a+3p) + \ldots f(a+(n-1)p) + f(b)]$$

où n est le nombre d'intervalles d'intégration.

12.1. 12.5. *(travail de l'élève Th. Musso)*

a) programme

```
      LOGICAL P,Q,R,COND,F
      P=.TRUE.
      Q=.TRUE.
      DO 2 I=1,2
      DO 1 J=1,2
      F=COND(P,Q)
      CALL CONDIT(P,Q,R)
      WRITE (9, 100)P,Q,R,F
  100 FORMAT(1X,L1,2X,L1,5X,L1,3X,L1)
    1 Q=.NOT.Q
    2 P=.NOT.P
      STOP
      END
      SUBROUTINE CONDIT(P,Q,R)
      LOGICAL P,Q,R
      R=(.NOT.P).OR.Q
      RETURN
      END
      FUNCTION COND(P,Q)
      LOGICAL COND,P,Q
      COND=(.NOT.P).OR.Q
      RETURN
      END
```

b) résultats

```
T   T     T   T
T   F     F   F
F   T     T   T
F   F     T   T
```

13.1 a) programme *(travail de l'élève J. Turin)*

```
*  SUITE DE FIBONACCI
      A=0.
      B=1.
      WRITE (9, 100)A,B
      L=2
    3 IF(L-26)1,1,2
    1 C=B+A
      Z=C/B
      WRITE (9, 101)L,C,Z
      A=B
      B=C
      L=L+1
      GO TO 3
    2 E=(1.+SQRT(5.))/2.
      WRITE (9,102)E
      STOP
  100 FORMAT('Y( 0)='F10.0/'Y( 1)='F10.0)
  101 FORMAT('Y(',I2,')='F10.0,10XF11.5)
  102 FORMAT(////'LA LIMITE DE LA SUITE EST '/
     +'EGALE A  (1+SQRT(5))/2  QUI VAUT 'F11.5)
      END
```

b) résultats

```
Y( 0)=          0.
Y( 1)=          1.
Y( 2)=          1.                1.00000
Y( 3)=          2.                2.00000
Y( 4)=          3.                1.50000
Y( 5)=          5.                1.66667
Y( 6)=          8.                1.60000
Y( 7)=         13.                1.62500
Y( 8)=         21.                1.61538
Y( 9)=         34.                1.61905
Y(10)=         55.                1.61765
Y(11)=         89.                1.61818
Y(12)=        144.                1.61798
Y(13)=        233.                1.61806
Y(14)=        377.                1.61803
Y(15)=        610.                1.61804
Y(16)=        987.                1.61803
Y(17)=       1597.                1.61803
Y(18)=       2584.                1.61803
Y(19)=       4181.                1.61803
Y(20)=       6765.                1.61803
Y(21)=      10946.                1.61803
Y(22)=      17711.                1.61803
Y(23)=      28657.                1.61803
Y(24)=      46368.                1.61803
Y(25)=      75025.                1.61803
Y(26)=     121393.                1.61803

LA LIMITE DE LA SUITE EST
EGALE A  (1+SQRT(5))/2  QUI VAUT      1.61803
```

Remarque : Dans le livre de F. Scheid "Numerical Analysis", p. 185-186, on peut trouver d'intéressants développements sur ce sujet relativement aux équations aux différences.

13.2. Version 1

(travail de l'élève Burgermeister)

a) programme

```
*     CALCUL DES 40 PREMIERS NB. PREMIERS
      DIMENSION J(40)
      WRITE(9, 100)
      WRITE(9, 101)
      WRITE(9, 102)
      WRITE(9, 103)
      WRITE(9, 101)
      J(1)=1
      J(2)=2
      J(3)=3
      N=3
      DO 20 K=3,199,2
      DO 21 I=3,N
      IF(K-(K/J(I))*J(I))21,20,21
   21 CONTINUE
      N=N+1
      J(N)=K
      IF(N-40)20,19,20
   20 CONTINUE
   19 WRITE(9,80)(J(N),N=1,40)
      WRITE(9, 101)
      WRITE(9, 100)
      STOP
  100 FORMAT( 40(1H*))
  101 FORMAT( 1H*,38X,1H*)
  102 FORMAT( 1H*,'  LISTE DES 40 PREMIERS NB. PREMIERS  ',1H*)
  103 FORMAT( 1H*,'  ==================================  ',1H*)
   80 FORMAT( 1H*,9X,4I5,9X,1H*)
      END
```

b) résultats

```
****************************************
*                                      *
*  LISTE DES 40 PREMIERS NB. PREMIERS  *
*  ==================================  *
*                                      *
*            1    2    3    5         *
*            7   11   13   17         *
*           19   23   29   31         *
*           37   41   43   47         *
*           53   59   61   67         *
*           71   73   79   83         *
*           89   97  101  103         *
*          107  109  113  127         *
*          131  137  139  149         *
*          151  157  163  167         *
*                                      *
****************************************
```

Remarque : dans les versions 2 et 3, on traite le même problème, mais sans l'emploi de variables dimensionnées.

Version 2

```
*     RECHERCHE DE NOMBRES PREMIERS DE 1 A X  , X=100
      X=100.
      A=1.
    8 IF(A-X)1,1,2
    1 R=SQRT(A)
      M=R
      R=M
      B=2.
    7 IF(B-R)3,3,4
    3 D=A/B
      J=D
      F=J
      IF(F-D)5,6,5
    5 B=B+1.
      GO TO 7
    4 WRITE(9,100)A
    6 A=A+1.
      GO TO 8
    2 STOP
  100 FORMAT(20XF6.1)
      END
```

Version 3 *(travail de l'élève J. Turin)*

```
*   RECHERCHE DES  X PREMIERS NOMBRES PREMIERS   X=40
      X=40.
      Y=0.
      A=1.
    8 IF(Y-X)1,2,2
    1 R=SQRT(A)
      M=R
      R=M
      B=2.
    7 IF(B-R)3,3,4
    3 D=A/B
      J=D
      F=J
      IF(F-D)5,6,5
    5 B=B+1.
      GO TO 7
    4 WRITE(9,100)A
      Y=Y+1.
    6 A=A+1.
      GO TO 8
    2 STOP
  100 FORMAT(20XF6.1)
      END
```

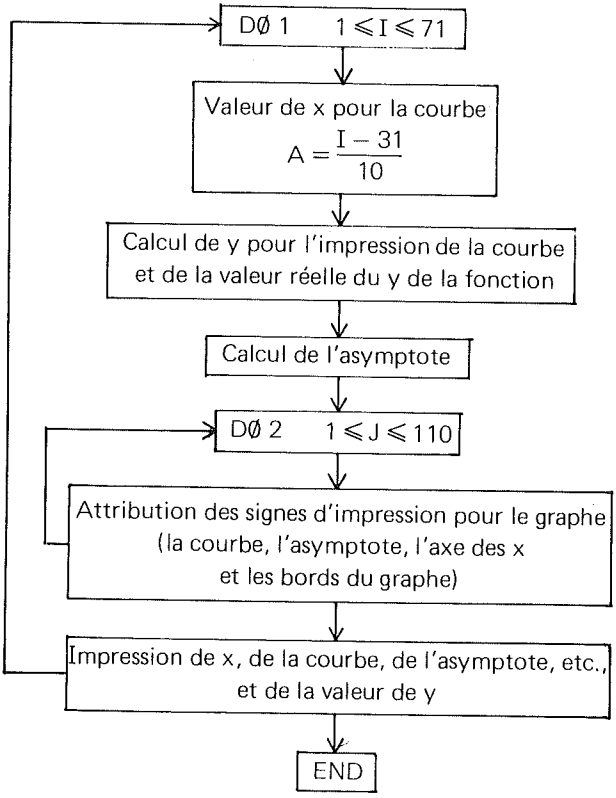
13.3. a) programme *(travail de l'élève S. Albertani)*

```
*      CLASSEMENT DE 50 NOMBRES PAR ORDRE CROISSANT
       DIMENSION X(50)
       READ (2,100)(X(I),I=1,50)
       WRITE(9,101)
       DO 1 J=1,50
       DO 1 I=J,50
       IF(X(J)-X(I))1,1,2
     2 XINTER=X(I)
       X(I)=X(J)
       X(J)=XINTER
     1 CONTINUE
       WRITE(9,102)(X(J),J=1,50)
   100 FORMAT(5F10.3)
   101 FORMAT('CLASSEMENT DANS L ORDRE CROISSANT DE 50 NOMBRES'/48(
      +1H*)////)
   102 FORMAT(5F12.5//)
       STOP
       END
```

b) résultats

```
CLASSEMENT DANS L ORDRE CROISSANT DE 50 NOMBRES
**********************************************

-98765.00000  -456.78900   -79.60001   -78.56001   -56.00000

   -34.00000   -33.50000    -8.00000    -0.89700    -0.09800

     0.00000     0.00900     0.66000     0.67800     1.00000

     2.00000     2.40000     3.00000     3.00000     3.78000

     4.00000     5.00000     5.40000     5.60000     6.00000

     6.00000     7.00000    12.00000    21.22000    23.75000

    34.00000    34.00000    34.00000    34.25999    34.56000

    44.32999    56.00000    56.00200    56.56000    67.00000

    67.09000    67.87999    68.00000    86.50000    89.00000

   345.77997   432.09998   444.55499  1689.00000  8765.00000
```

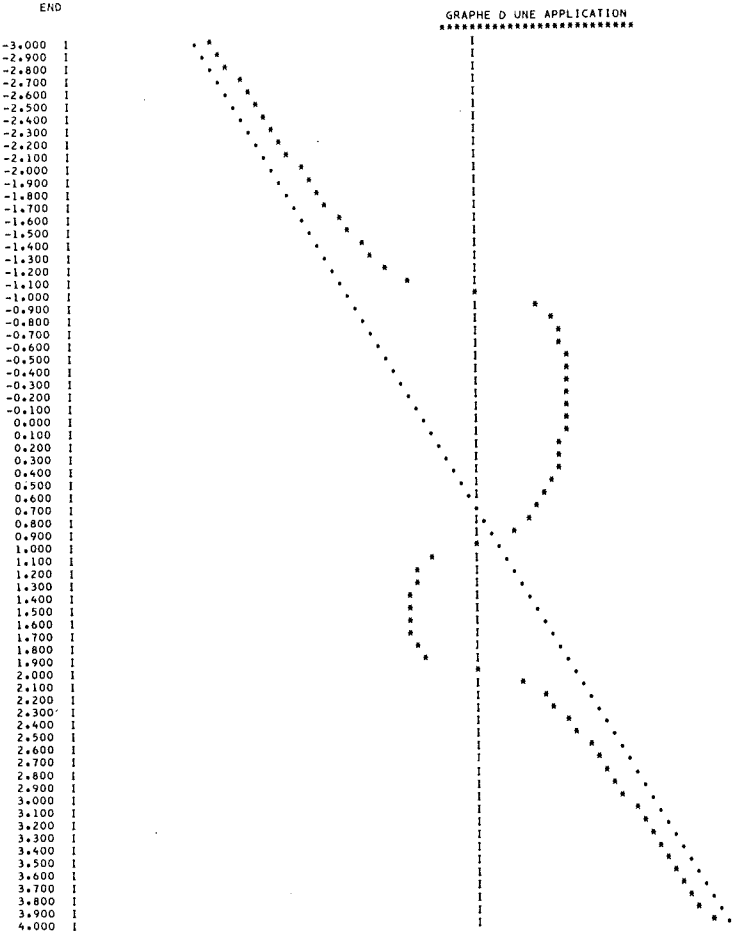
13.6. a) organigramme



b) programme

```
      DIMENSION M(110)
      WRITE(2, 100)
100   FORMAT (60X'GRAPHE D UNE APPLICATION'/59X,26(1H*))
      DO1 I=1,71
      I=I-31
      A=I/10.
      I=I+31
      B=(( CUBERT((A**3)-2.*(A**2)-A+2.))*10. )+54.
      C=(( CUBERT((A**3)-2.*(A**2)-A+2.)))
      N=B
      D=((A-(2./3.))*10.)+54.
      L=D
      DO2 J=1,110
      M(J)=1H
      IF(J.EQ.110)M(J)=1HI
      IF(J.EQ.1)M(J)=1HI
      IF(J.EQ.55)M(J)=1HI
      IF(J.EQ.(N+1))M(J)=1H*
2     IF(J.EQ.(L+1))M(J)=1H.
      WRITE(2, 101)A,(M(J),J=1,110),C
1     CONTINUE
      STOP
101   FORMAT (F7.3,2X,110(A1),2X,F9.5)
      END
```

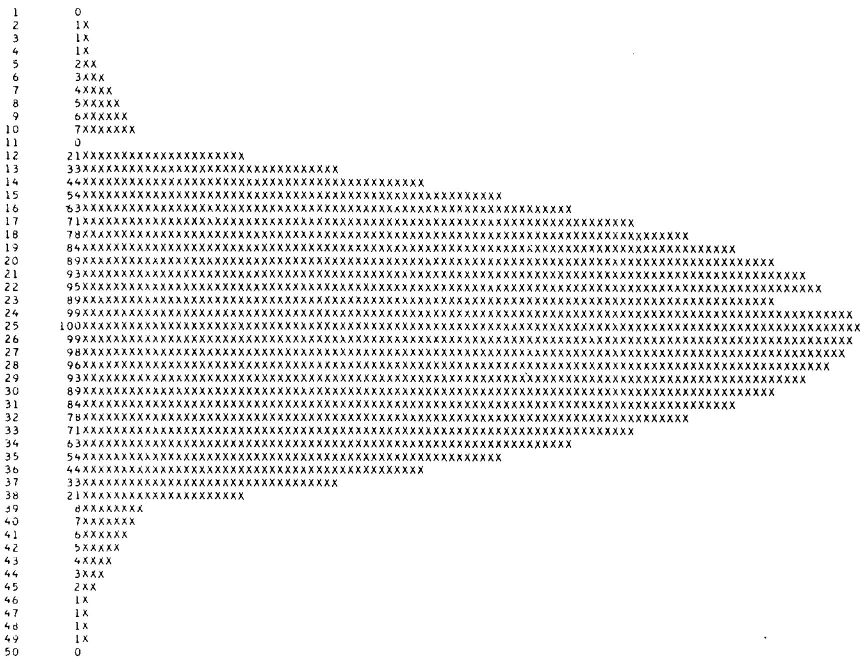
c) résultats



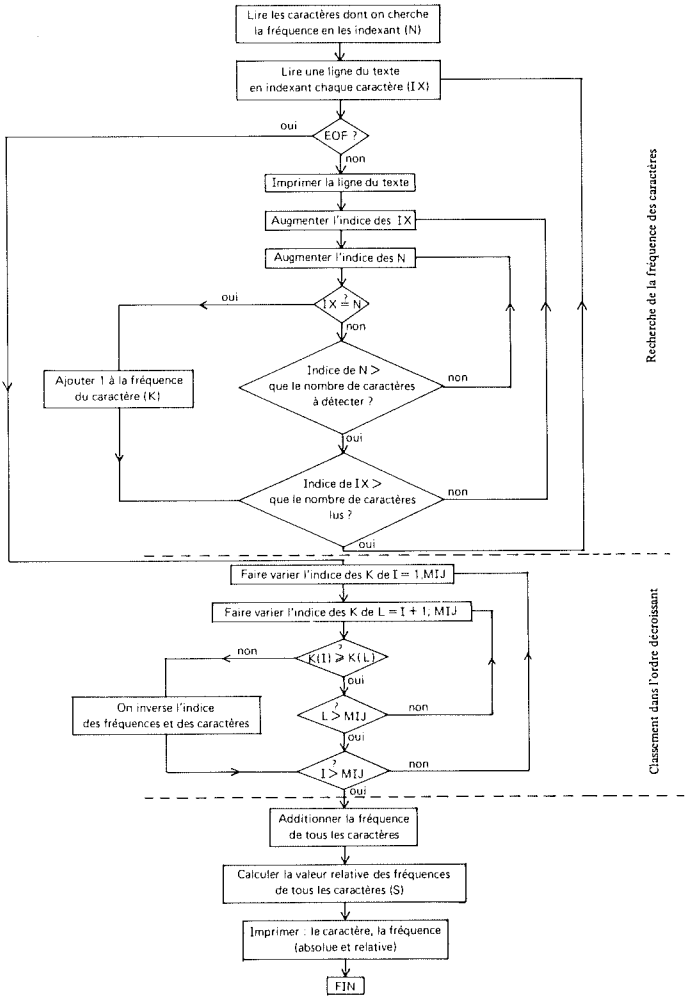
13.7. Albert Einstein

13.10.

(travail de l'élève R. Rosselet)



13.11. a) organigramme *(travail de l'élève J. Turin)*



b) programme

```
      DIMENSION IX(72),N(31),K(31),S(31)
      WRITE(9,106)
      CALL DEFINE(2,'TEXTE ')
      MIJ=31
      DO 1 I=1,MIJ
    1 K(I)=0
      READ (2,100)(N(I),I=1,MIJ)
    5 READ (2,101)(IX(J),J=1,72)
      IF(IEOF(2),EQ.1)GOTO 2
    3 WRITE(9, 102)(IX(J),J=1,72)
      DO4 J=1,72
      DO4 I=1,MIJ
    4 IF(IX(J).EQ.N(I))K(I)=K(I)+1
      GO TO 5
    2 DO20 I=1,MIJ
      JI=I+1
      DO20 L=JI,MIJ
      IF(K(I).GE.K(L))GOTO20
   22 JN=K(L)
      K(L)=K(I)
      K(I)=JN
      JM=N(L)
      N(L)=N(I)
      N(I)=JM
   20 CONTINUE
      MI=0
      DO7 I=1,MIJ
    7 MI=MI+K(I)
      DO8 I=1,MIJ
    8 S(I)=FLOAT(K(I))*100./FLOAT(MI)
      WRITE(9, 200)
      WRITE(9, 103)((N(I),K(I),S(I)),I=1,MIJ)
      WRITE(9, 122)MI
      STOP
  100 FORMAT(31A1)
  101 FORMAT(72A1)
  102 FORMAT(72A1)
  103 FORMAT(A1,10X15,10XF6.2)
  106 FORMAT(' PROGRAMME DETERMINANT LA FREQUENCE DES CARACTERES'
     +/'DANS UNE LANGUE QUELCONQUE'////)
  200 FORMAT('CARACTERE'12X'FREQUENCE'//13X'ABSOLUE'8X'RELATIVE'
     +//////)
  122 FORMAT(///30X'NOMBRES DE CARACTERES IDENTIFIES'3X15)
      END
```

c) données et résultats

```
 PROGRAMME DETERMINANT LA FREQUENCE DES CARACTERES
DANS UNE LANGUE QUELCONQUE


ABCDEFGHIJKLMNOPQRSTUVWXYZ.,/()
  A CE MOMENT, EN UN POINT OU LA FORET ETAIT PLUS PROFONDE ET PLUS DEN-
SE ET OU UNE PISTE TRAVERSAIT NOTRE ROUTE, JE VIS BRUSQUEMENT SURGIR DU
BROUILLARD, LA-BAS DEVANT NOUS, AU CARREFOUR DES DEUX PISTES, UN SOLDAT
ENFONCE DANS LA NEIGE JUSQU AU VENTRE. IL ETAIT LA, DEBOUT, IMMOBILE, LE
BRAS DROIT TENDU POUR INDIQUER LE CHEMIN. QUAND NOUS PASSAMES DEVANT LUI
SCHULTZ PORTA LA MAIN A SON KEPI, COMME POUR LE SALUER ET LE REMERCIER,
PUIS DIT  - EN VOILA UN AUTRE QUI VOUDRAIT ALLER DANS LE CAUCASE  ET SE
MIT A RIRE EN SE RENVERSANT SUR LE DOSSIER DE SON SIEGE.       AU BOUT
D UN AUTRE SEGMENT DE ROUTE, A UN AUTRE CROISEMENT DE PISTE, VOICI QU A
GRANDE DISTANCE, UN AUTRE SOLDAT APPARUT, EGALEMENT ENFONCE DANS LA NEI-
GE, LE BRAS DROIT TENDU POUR NOUS MONTRER LE CHEMIN.       -ILS VONT
MOURIR DE FROID, CES PAUVRES DIABLES, DIS-JE, SCHULTZ SE RETOURNA POUR
ME REGARDER  -IL N Y A PAS DE DANGER QU ILS MEURENT DE FROID DIT-IL,
  ET IL RIAIT. JE LUI DEMANDAI POURQUOI IL PENSAIT QUE CES PAUVRES BOU-
GRES N ETAIT PAS EN DANGER DE MOURIR GELES.  -PARCE QUE DESORMAIS, ILS
SONT HABITUES AU FROID ME REPONDIT SCHULTZ ET IL RIAIT, EN ME TAPANT SUR
L EPAULE. IL ARRETA LA VOITURE ET SE TOURNA VERS MOI EN SOURIANT
 -VOUS VOULEZ LE VOIR DE PRES, VOUS POURREZ LUI DEMANDER S IL A FROID.
 NOUS DESCENDIMES DE VOITURE ET NOUS NOUS APPROCHAMES DU SOLDAT QUI
ETAIT LA, DEBOUT, IMMOBILE, LE BRAS DROIT TENDU POUR NOUS MONTRER LA
ROUTE. IL ETAIT MORT. IL AVAIT LES YEUX HAGARDS, LA BOUCHE ENTROUVERTE.
C ETAIT UN SOLDAT RUSSE MORT.       -C EST NOTRE POLICE DES VOIES ET
COMMUNICATIONS, DIT SCHULTZ. NOUS L APPELONS ,, LA POLICE SILENCIEUSE ,,
 -ETES-VOUS BIEN SUR QU IL NE PARLE PAS.       -QU IL NE PARLE PAS. ACH
SO, ESSAYEZ DE L INTERROGER.       -IL VAUDRAIT MIEUX QUE JE N ESSAIE
PAS, JE SUIS SUR QU IL ME REPONDRAIT, DIS-JE.       -ACH SEHR AMUSANT,
S ECRIA SCHULTZ EN RIANT.       -JA,SEHR AMUSANT, NICHT WAHR. PUIS, J A-
JOUTAI D UN AIR INDIFFERENT       -QUAND VOUS LES AMENEZ LA SUR PLACE,
ILS SONT VIVANTS OU MORTS.       -VIVANTS, NATURELLEMENT, REPONDIT
SCHULTZ.       -ENSUITE, ILS MEURENT DE FROID, NATURELLEMENT, DIS-JE
ALORS.       -NEIN, NEIN,ILS NE MEURENT PAS DE FROID, REGARDEZ LA. ET
SCHULTZ ME MONTRA UN CAILLOT DE SANG, GRUMEAU DE GLACE ROUGIE, SUR LA
TEMPE DU MORT.       -ACH SO, SEHR AMUSANT.       -SEHR AMUSANT , NICHT
WAHR, DIT SCHULTZ. PUIS IL AJOUTA EN RIANT  ,, IL FAUT TOUT DE MEME BIEN
QUE LES PRISONNIERS RUSSES SERVENT A QUELQUE CHOSE ,,.
      EXTRAIT DE  KAPUTT  DE  MALAPARTE
```

CARACTERE	FREQUENCE ABSOLUE	RELATIVE
E	254	13.26
A	151	7.88
S	150	7.83
U	143	7.46.
R	141	7.36
T	137	7.15
I	137	7.15
N	125	6.52
O	109	5.69
L	98	5.11
D	82	4.28
M	55	2.87
,	51	2.66
P	49	2.56
C	44	2.30
.	32	1.67
V	30	1.57
H	26	1.36
Q	19	0.99
G	19	0.99
B	16	0.84
F	14	0.73
Z	13	0.68
J	12	0.63
Y	3	0.16
X	3	0.16
W	2	0.10
K	1	0.05
/	0	0.00
(	0	0.00
)	0	0.00
'		

NOMBRES DE CARACTERES IDENTIFIES 1916

13.12. a) programme *(travail de l'élève Ferrière)*

```
8     WRITE(9,103)
103   FORMAT(//)
      READ (9,101 )DU,PAYE
      IF(DU-10000.)9,7,9
9     A=PAYE-DU+0.001
      WRITE(9, 102)DU,PAYE,A
      A=AMOD(A,10.)
      IF(A.LT.5.) GOTO 1
      WRITE(9, 1500)
      A=A-5.
1     IF(A.LT.2.) GOTO 2
      WRITE(9, 1200)
      A=A-2.
      GOTO 1
2     IF(A.LT.1.) GOTO 3
      WRITE(9, 1100 )
      A=A-1.
3     IF(A.LT.0.5)GOTO 4
      WRITE(9, 1050 )
      A=A-0.5
4     IF(A.LT.0.2)GOTO 5
      WRITE(9, 1020)
      A=A-0.2
      GOTO 4
5     IF(A.LT.0.1)GOTO 6
      WRITE(9, 1010)
      A=A-0.1
6     IF(A.LT.0.05)GOTO8
      WRITE(9, 1005 )
      GOTO 8
7     STOP
101   FORMAT(2(F10.2))
102   FORMAT(//10X'TOTAL'10X'PAYE'10X'RENDU'11X,'RETOUR MONNAIE :
     +5X,F10.2,4X,F10.2,5X,F10.2)
1500  FORMAT(45X'  RENDRE UNE PIECE DE 5.- F')
1200  FORMAT(45X'  RENDRE UNE PIECE DE 2.- F')
1100  FORMAT(45X'  RENDRE UNE PIECE DE 1.- F')
1050  FORMAT(45X'  RENDRE UNE PIECE DE 50 CT')
1020  FORMAT(45X'  RENDRE UNE PIECE DE 20 CT')
1010  FORMAT(45X'  RENDRE UNE PIECE DE 10 CT')
1005  FORMAT(45X'  RENDRE UNE PIECE DE  5 CT')
      END
```

b) résultats

```
9.60,10.

        TOTAL          PAYE          RENDU          RETOUR MONNAIE :

         9.60         10.00          0.40
                                                RENDRE UNE PIECE DE 20 CT
                                                RENDRE UNE PIECE DE 20 CT

1.05,10.

        TOTAL          PAYE          RENDU          RETOUR MONNAIE :

         1.05         10.00          8.95
                                                RENDRE UNE PIECE DE 5.- F
                                                RENDRE UNE PIECE DE 2.- F
                                                RENDRE UNE PIECE DE 1.- F
                                                RENDRE UNE PIECE DE 50 CT
                                                RENDRE UNE PIECE DE 20 CT
                                                RENDRE UNE PIECE DE 20 CT
                                                RENDRE UNE PIECE DE  5 CT

2.45,6.85

        TOTAL          PAYE          RENDU          RETOUR MONNAIE :

         2.45          6.85          4.40
                                                RENDRE UNE PIECE DE 2.- F
                                                RENDRE UNE PIECE DE 2.- F
                                                RENDRE UNE PIECE DE 20 CT
                                                RENDRE UNE PIECE DE 20 CT
```

Remarques :

1. DU = somme due (total)

   PAYE = somme payée (maximum Fr. 10.–)

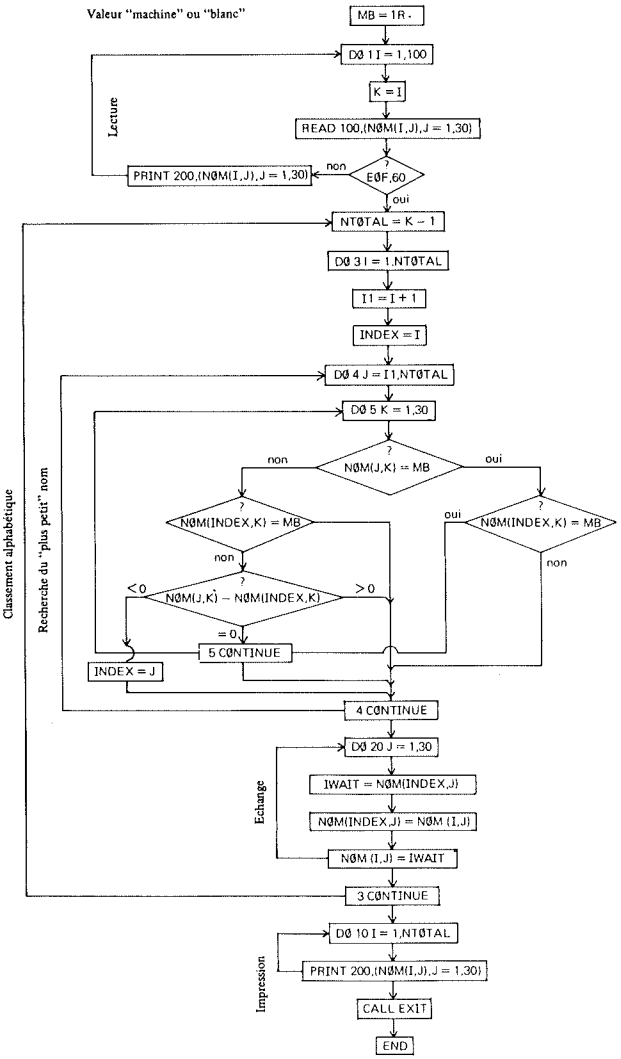
   A = d'abord différence (à imprimer sur le ticket)
   puis somme à rendre en monnaie
   et progressivement solde à rendre

2. On a ajouté un millième à A (instruction no 9) pour éviter des erreurs dans les tests (cf. test logique p. 60)

3. L'ordre WRITE pourrait être remplacé par la commande d'un mécanisme approprié dans la colonne de monnaie correspondante.

4. On a simplifié ici le problème en ne traitant qu'une monnaie pour une somme maximum de Fr. 10.–.

13.14. a) organigramme



b) programme

(travail de l'élève J.-M. Schell)

▲34

```
      PROGRAM CLASSEM
      DIMENSION NOM(100,30)
      MB=1R
      DO 1 I=1,100
      K=I
      READ 100,(NOM(I,J),J=1,30)
      IF(EOF,60)2,1
1     PRINT 200,(NOM(I,J),J=1,30)
2     NTOTAL=K-1
      PRINT 300
      DO 3 I=1,NTOTAL
      I1=I+1
      INDEX=I
      DO 4 J=I1,NTOTAL
      DO 5 K=1,30
      IF(NOM(J,K)-MB)8,6,8
8     IF(NOM(INDEX,K)-MB)9,4,9
9     IF(NOM(INDEX,K)-NOM(J,K))4,5,7
6     IF(NOM(INDEX,K)-MB)7,5,7
7     INDEX=J
      GO TO 4
5     CONTINUE
4     CONTINUE
      DO 20 J=1,30
      IWAIT=NOM(INDEX,J)
      NOM(INDEX,J)=NOM(I,J)
20    NOM(I,J)=IWAIT
3     CONTINUE
      DO 10 I=1,NTOTAL
10    PRINT 200,(NOM(I,J),J=1,30)
100   FORMAT(30R1)
200   FORMAT(1X,30R1)
300   FORMAT(///)
      CALL EXIT
      END
```

c) données et résultats

données à classer

```
 ALBERT PAUL
MATHEMATIQUE COSINUS
MARTIN ZIZI
MARTINE ELIANE
 1ALBERT DEUX
MARTINE
ALBERT JEAN
```

données classées

```
 1ALBERT DEUX
 ALBERT PAUL
ALBERT JEAN
MARTIN ZIZI
MARTINE
MARTINE ELIANE
MATHEMATIQUE COSINUS
```

d) autre programme pour le système HB 1642 et donnant les mêmes résultats que ci-dessus.

```
      DIMENSION NOM(100,15)
      DO 1 I=1,100
      K=I
      IF(IEOF(4).EQ.1)GOTO2
      READ(4,100)(NOM(I,J),J=1,15)
1     WRITE(2,100)(NOM(I,J),J=1,15)
2     NTOTAL=K-1
      WRITE(2,300)
      DO 3 I=1,NTOTAL
      N1=I+1
      DO 3 K=N1,NTOTAL
      DO 4 J=1,15
      IF(NOM(I,J)-NOM(K,J))3,4,5
4     CONTINUE
5     DO 6 J=1,15
      ISWAP=NOM(I,J)
      NOM(I,J)=NOM(K,J)
6     NOM(K,J)=ISWAP
3     CONTINUE
      DO 7 I=1,NTOTAL
7     WRITE(2,100)(NOM(I,J),J=1,15)
300   FORMAT(////)
100   FORMAT(15A2)
      STOP
      END
```

13.15. a) programme

▲34

```
      PROGRAM MAIN
      COMMON/OUF/AMA(10,10),AMB(10,10),AMC(10,10),N1,M1,N2,M2
      COMMON/HELLO/INDEX,K1,K2
      DIMENSION AMD(10,10)
   10 READ 111,MOT,INDEX,N1,M1,N2,M2,K1,K2
      IF(EOF,60)99,11
   11 IF(MOT.NE.2HOK) GOTO 50
      READ 110,((AMA(I,J),J=1,M1),I=1,N1)
      CALL IMPRES(AMA,N1,M1,1)
      IF(INDEX.EQ.6) GOTO 25
    1 READ 110,((AMB(I,J),J=1,M2),I=1,N2)
      CALL IMPRES(AMB,N2,M2,2)
      GOTO(20,21,22,23,24),INDEX
   20 IF(M1.NE.N2) GOTO 30
      CALL PRODM
      CALL IMPRES(AMC,N1,M2,3)
      GOTO 10
   21 CALL TRANSM(AMB,N2,M2)
      GOTO 20
   22 CALL TRANSM(AMA,N1,M1)
      GOTO 20
   23 CALL TRANSM(AMA,N1,M1)
      CALL TRANSM(AMB,N2,M2)
      GOTO 20
   24 CALL REFM(AMA,N1,M1,K1)
      CALL REFM(AMB,N2,M2,K2)
      IF(N1.NE.N2.OR.M1.NE.M2) GOTO 40
      CALL ADDM
      CALL IMPRES(AMC,N1,M1,3)
      GOTO 10
   25 CALL TRANSM(AMA,N1,M1)
      CALL IMPRES(AMA,N1,M1,3)
      GOTO 10
   30 PRINT 113
      GOTO 10
   40 PRINT 114
      GOTO 10
   50 PRINT 112
      GOTO 10
   99 CALL EXIT
  110 FORMAT(8(F10.5))
  111 FORMAT(A2,11,412,2F10.5)
  112 FORMAT(* IL N Y A PAS DE CARTE D INSTRUCTIONS A L ENDROIT VOULU,OU
     1 ELLE NE COMMENCE PAS PAR LE MOT OK*)
  113 FORMAT(* LES DIMENSIONS DE CES 2 MATRICES NE PERMETTENT PAS LEUR M
     1ULTIPLICATION*)
  114 FORMAT(* LES DIMENSIONS DE CES 2 MATRICES NE PERMETTENT PAS LEUR A
     1DDITION*)
      END
      SUBROUTINE IMPRES(AMD,N,M,KOD)
      COMMON/HELLO/INDEX,K1,K2
      DIMENSION AMD(10,10)
      IF(KOD.EQ.1)PRINT 107
      IF(KOD.EQ.2)PRINT 108
      IF(KOD.NE.3)GOTO 8
      GOTO(1,2,3,4,5,6),INDEX
    1 PRINT 101
      GOTO 8
    2 PRINT 102
      GOTO 8
    3 PRINT 103
      GOTO 8
    4 PRINT 104
      GOTO 8
    5 PRINT 105,K1,K2
      GOTO 8
    6 PRINT 106
    8 DO 7 I=1,N
      PRINT 109,(AMD(I,J),J=1,M)
    7 CONTINUE
      IF(KOD.EQ.3) PRINT 110
      RETURN
  101 FORMAT(//* LA MATRICE A*1H**B EST :*//)
  102 FORMAT(//* LA MATRICE A*1H**T(B) EST : *//)
  103 FORMAT(//* LA MATRICE T(A)*1H**B EST :*//)
  104 FORMAT(//* LA MATRICE T(A)*1H**T(B) EST :*//)
  105 FORMAT(//* LA MATRICE*F10.5,1H*,*A + *F10.5,1H*,*B EST :*//)
  106 FORMAT(//* LA MATRICE T(A) EST :*//)
  107 FORMAT(//* LA MATRICE A EST :*//)
  108 FORMAT(//* LA MATRICE B EST :*//)
  109 FORMAT(1HS/7X10(F10.5))
  110 FORMAT(//1X134(1H-))
      END
```

```
      SUBROUTINE TRANSM(AMD,N,M)
      DIMENSION AMD(10,10),AME(10,10)
      DO 40 I=1,N
      DO 40 J=1,M
   40 AME(J,I)=AMD(I,J)
      KASE=N $ N=M $ M=KASE
      DO 50 I=1,N
      DO 50 J=1,M
   50 AMD(I,J)=AME(I,J)
      RETURN
      END
      SUBROUTINE REFM(AMD,N,M,R)
      DIMENSION AMD(10,10)
      DO 50 I=1,N
      DO 50 J=1,M
   50 AMD(I,J)=R*AMD(I,J)
      RETURN
      END
      SUBROUTINE PRODM
      COMMON/OUF/AMA(10,10),AMB(10,10),AMC(10,10),N1,M1,N2,M2
      DO 1 I1=1,N1
      DO 1 J2=1,M2
      AMC(I1,J2)=0.
      DO 1 J1=1,M1
    1 AMC(I1,J2)=AMC(I1,J2)+AMA(I1,J1)*AMB(J1,J2)
      RETURN
      END
      SUBROUTINE ADDM
      COMMON/OUF/AMA(10,10),AMB(10,10),AMC(10,10),N1,M1,N2,M2
      DO 3 I=1,N1
      DO 3 J=1,M1
    3 AMC(I,J)=AMA(I,J)+AMB(I,J)
      RETURN
      END
```

(travail de l'élève A. Schorer)

b) résultats

```
 LA MAIRICE A EST :

          4.00000    5.00000
         -7.00000    8.00000
          9.00000   -2.00000
          6.00000   -3.00000


 LA MAIRICE B EST :

         -5.00000    6.00000   -2.00000   -4.00000    8.00000    1.00000
          0.00000    0.00000   -3.00000    1.00000   -3.00000    4.00000
          5.00000   -7.00000    5.00000    7.00000    3.00000   -2.00000
          3.00000   -1.00000    1.00000    6.00000   -9.00000    7.00000


 LA MAIRICE T(A)*B EST :

         43.00000  -45.00000   64.00000   76.00000   26.00000    0.00000
        -44.00000   47.00000  -47.00000  -44.00000   37.00000   20.00000
```

ANNEXE

Différences entre le FORTRAN standard et le FORTRAN de l'ordinateur CDC 3800 ou de l'ordinateur HB 1642

▲ 1 Cette compatibilité entre ordinateurs pour un même langage existe pour autant que l'on respecte certaines normes. C'est ce que nous voulons souligner en ajoutant cette annexe.

▲ 2 Les cartes de contrôles sont naturellement celles utilisées sur la CDC 3800.

▲ 3 L'instruction PRØGRAM n'est pas reconnue en FORTRAN standard.

▲ 4 L'instruction STØP est standard et a le même effet que CALL EXIT.

▲ 5 La longueur d'un mot (unité de mémoire, voir aussi page 86) est sur la CDC 3800 de 48 bits (binary digits); l'image du tiroir est fort utile pour cette représentation. Le nombre de chiffres significatifs et les valeurs extrêmes que peuvent prendre les données dépendent avant tout de la longueur du mot machine, donc de la structure même de chaque ordinateur. Le système time-sharing HB 1642 a des mots de 16 bits. Le nombre de chiffres significatifs en mode flottant est 6, les entiers appartiennent à l'intervalle $[-2^{15}+1; 2^{15}-1]$, c'est-à-dire [− 32767; 32767]. (cf. [34]). Etant donné un ordinateur avec des mots de p bits, il est possible de connaître immédiatement les limites pour les nombres entiers, soit l'intervalle $[-2^{p-1}+1; 2^{p-1}-1]$. Pour un nombre flottant les limites de la représentation interne sont spécifiées de manière détaillée par le constructeur. Si l'on désire davantage de chiffres significatifs, il convient, suivant le cas, de travailler en double précision (cf. 10.4 et ▲26).

▲ 6 Pour avoir une compatibilité sur la plupart des ordinateurs, il faut envisager au maximum 6 caractères alphanumériques, bien que 8 caractères par variable soient tolérés sur la CDC 3800.

▲ 7 I**R = R n'est pas standard.

▲ 8 Il faut être très prudent pour chaque abréviation des fonctions de librairie qui varient parfois d'une machine à l'autre.

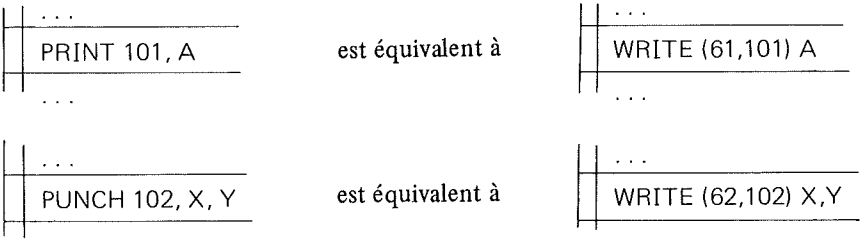
▲ 9 READ n, I où l est une liste de variables (par exemple A, B, C, D, ...)
PRINT n, I n est l'étiquette de l'instruction FØRMAT
ne sont pas standards.

Sur le système CDC 3800 :

```
...
READ 100, A, B, C
...
```

est équivalent à

```
...
READ (60,100) A,B,C
...
```



En revanche les instructions suivantes sont dépourvues de sens :

```
. . .
READ (61,100) Z                    (lire sur l'imprimante !)
. . .
WRITE (60,103) W                   (écrire sur le lecteur !)
```

C'est volontairement que nous avons omis l'instruction PUNCH n, I qui permet de perforer des cartes par la machine.

▲10 Cette restriction n'est pas obligatoire, mais elle permet d'éviter bon nombre d'erreurs avec la bande pilote de l'imprimante qui génère, à l'aide du caractère de contrôle (premier caractère de chaque ligne d'impression), les sauts de page, la surimpression, etc.

▲11 Le code "astérisque" (CDC 3800) et le code "apostrophe" (HB 1642) ne sont donc pas standards.

▲12 Pour que cette instruction soit standard, il faut que E soit une variable entière I avec $1 \leqslant I \leqslant m$. Le résultat n'est pas défini si I est en dehors de ces limites.

▲13 Cette instruction n'est pas standard.

▲14 L'instruction IF (EØF, u) n_1, n_2 n'est pas standard et se rencontre sur la CDC 3800. L'instruction analogue sur le système HB 1642 est IF (IEØF(u)) n_1, n_2 u étant le numéro d'unité logique de 1 à 6 pour la lecture sur disque (cf [34]).

▲15 Pour que l'instruction DØ n i = m_1, m_2, m_3 soit standard, il faut que m_1, m_2, m_3 soient plus grands que zéro.

▲16 Cette remarque n'est pas standard.

▲17 La limite pour les nombres que l'on peut inscrire dans les colonnes 1 à 5 (numéro d'instruction) dépend de l'ordinateur; pour le système HB 1642 c'est 32767.

▲18 Ce FØRMAT n'est pas standard.

▲19 Cette possibilité n'existe pas en FORTRAN standard.

▲20 En FORTRAN standard ce n'est pas 1 ou 0 mais T (true) ou F (false).

▲21 Le nombre est remplacé par des astérisques (CDC 3800) ou précédé par un signe dollar (HB 1642).

▲22 Sur l'ordinateur CDC 3800 une variable logique non dimensionnée occupe 1 bit d'un mot; il y a donc un "gaspillage" de 47 bits. Une variable logique dimensionnée prend au maximum 32 bits par mot.

Exemple : LØGICAL TEST (25,5)
est stocké dans quatre mots (3 x 32 bits + 29 bits).

▲23 Sur l'ordinateur HB 1642 c'est le caractère * qui signifie que la ligne est un commentaire.

▲24 C'est le symbole + qui indique la continuation d'une instruction au système HB 1642.

▲25 Le tableau ci-dessous donne les principaux numéros d'unités logiques pour les deux ordinateurs fréquemment cités dans ce livre et suggère au lecteur de compléter la dernière colonne pour un autre type de machine.

Périphériques	Numéro d'unité logique correspondant pour l'ordinateur :	
	CDC 3800	HB 1642
imprimante	61	3 (en batch)
lecteur de cartes	60	–
perforateur de cartes	62	–
dérouleur de bandes magnétiques	1 à 49 et 50 à 59	0 ou 1 (en batch)
disque	50 à 59	1 à 6
console et écran (entrée et sortie)	–	9
...		
...		
...		

Les numéros d'unités logiques doivent être de type entier; il n'est cependant pas obligatoire de les fournir sous forme de constantes entières, comme l'illustre l'exemple ci-dessous :

```
.
.
.
* DECLARATIØN DES NUMERØS D UNITES LØGIQUES AU DEBUT DU PRØGRAMME
  IØUT=61
  INPUT=60
  MAGTAP=42
  NDISK=53
.
.
.
  READ (INPUT,100) X          lecture sur une carte perforée de la variable X
.
.
.
  WRITE (NDISK,100) X         écriture sur disque de la variable X
.
.
.
  READ (MAGTAP,102) Y,Z       lecture sur bande magnétique des variables Y, Z
.
.
.
  WRITE (IØUT,103) Y,Z        écriture sur l'imprimante des variables Y, Z
```

Afin d'accroître la compatibilité des programmes et de minimiser les corrections pour les utiliser sur d'autres installations, la technique présentée ci-dessus est hautement souhaitable.

▲26 Sur l'ordinateur HB 1642, le nombre de chiffres significatifs passe de 6 en FØRMAT F ou E à 11 en FØRMAT D.

▲27 Cette remarque est valable pour l'ordinateur CDC 3800. Sur le système time-sharing HB 1642 la place disponible pour un utilisateur en mémoire centrale est de 10 K. Les dimensions maximales d'un tableau dépendent de la place laissée disponible par les instructions même du programme, des sous-programmes, etc. L'occupation mémoire est donnée clairement par le "memory map" (cf. [34]).

▲28 Pour le système HB 1642 une variable entière occupe un mot mémoire et une variable flottante deux mots. Les remarques 2 et 3 sont donc correctes, mais si la variable JAM était remplacée par la variable SAM, c'est 800 mots et 340 mots "gaspillés" qui correspondraient à la réalité. Il faut donc être attentif pour chaque ordinateur à la manière dont est représentée l'information à l'intérieur du système (cf. [34]).

▲29 Sur une console on ne peut dépasser 72 caractères par ligne, il faut deux lignes de données (une entièrement remplie et l'autre avec la huitième et la neuvième valeur de A (I,J), le FØRMAT 102 étant :

102	FØRMAT (7F10.2)

▲30 Le message sera : UN EXT DEN pour la CDC 3800
MISSING DEN pour HB 1642

▲31 La représentation interne des caractères dépend de deux facteurs essentiellement :

a) la longueur des mots de l'ordinateur considéré,

b) le code utilisé pour représenter en binaire les caractères.

Pour la CDC 3800, c'est le code BCD qui est employé; chaque caractère est représenté par 6 bits. Les mots ayant une longueur de 48 bits, la remarque de la page 86 montre l'importance du nombre 8 dans les FØRMAT A et R.

Pour le HB 1642, c'est le code ISØ qui est utilisé; chaque caractère est représenté par 8 bits. Les mots ayant une longueur de 16 bits, c'est le nombre 2 qui joue un rôle capital. Le FØRMAT A ne pourra être que A1 ou A2. Une difficulté supplémentaire apparaît : le compilateur FORTRAN ne s'intéresse dans ce cas qu'au mot et non à la variable.

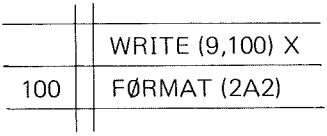
Pour une variable entière,
il y a un mot donc 1 à 2 caractères par variable.

Pour une variable flottante,
il y a deux mots donc 1 à 4 caractères par variable.

Pour une variable en double précision,
il y a trois mots donc 1 à 6 caractères par variable.

Pour une variable complexe,
il y a quatre mots donc 1 à 8 caractères par variable.

Exemple :



(écriture de 4 caractères)

Pour d'autres exemples ou explications cf. [34]

▲32 Cette remarque n'est pas standard. Consulter le manuel de votre système (cf. [3], [20];

▲33 En ce qui concerne la transmission des tableaux entre le programme principal et les sous-programmes (pour les dimensions variables notamment) il faut être très prudent. Une excellente description est donnée dans le chapitre 5 du cours de Monsieur F. Louis, cf. [25].

▲34 Ce programme est destiné à être exécuté sur CDC 3800.

▲35 En fait, sur l'imprimante CDC 3800, le mélange de mode est autorisé (mixed mode), mais nous ne nous arrêterons pas sur cette facilité qui, au début, ne peut qu'engendrer la confusion.

BIBLIOGRAPHIE

1. Amis de Sèvres*, Préparation des enseignants des écoles secondaires à l'informatique, No 3, 1971.
2. Ball, Bloch, Von Eicken, Programming Techniques, CERN 68/V.
3. CDC, FORTRAN Reference Manual 3400 – 3600 – 3800.
4. CDC, SCOPE.
5. CDC FORTRAN (cours de programmation par B. Thompson).
6. CDC Computer Programming Concepts.
7. Datamation*, "Minicomputers", No 5, p. 26, 1971.
8. Doran*, "Structural Programming", Computer Education, No 9, p. 25.
9. Dreyfus, FORTRAN IV, Dunod, 1967.
10. Education*, "Enseigner l'informatique", revue du 13.1.72, p. 24-30.
11. EPFL, Cours de programmation FORTRAN.
12. GRETI, Préparation des enseignants des écoles secondaires à l'informatique, 1971, numéro spécial.
13. Forsythe, Computer Science, a first course, Wiley, 1971.
14. Foster, Computer Architecture, Van Nostrand, 1970.
15. Haury, Lang, Olza, Eléments de logique, Spes-Dunod, 1973.
16. Haury, Lang, Olza, Cours de mathématique, chapitre 5, Fonctions. Département de l'instruction publique de Genève.
17. Hebenstreit*, Informatique et pédagogie, revue de l'Education du 14.10.71, p. 32-37.
18. Hebenstreit*, L'Enseignement et les ordinateurs, résumé d'un séminaire du GRETI, 1971.
19. Honeywell Bull, L'Informatique, 1971.
20. Honeywell Bull, Reference Manuals for HB 1642 system.
21. IBM, Manuel d'opération multiperforatrice alphanumérique imprimante IBM 26.
22. IBM*, Bulletin No 78, p. 3, 1972.
23. IBM*, L'Ordinateur dans l'enseignement, No spécial du Bulletin, 1972.
24. Levrat, Feuilles polycopiées distribuées dans le cadre du cours de calcul électronique de l'Université de Genève.

*articles

25. Louis, Cours de programmation du CERN, 68/69.
26. Louis, Introduction à l'analyse numérique, CERN, 69/70.
27. Mac Cracken, FORTRAN with engineering applications, Wiley.
28. Mac Calla, Introduction to numerical methods and FORTRAN.
29. Mac Cracken, A guide to FORTRAN programming, Wiley, 1961.
30. OCDE, L'Enseignement de l'informatique à l'école secondaire, 1971.
31. Nievergelt, Farrer*, "What machines can and cannot do ?" ACM Computing Surveys 4, p. 81, 1972.
32. Pelletier, Techniques numériques appliquées au calcul scientifique, Masson, 1971.
33. Scheid, Theory and problems of numerical analysis, Schaum's outline series, 1968.
34. Some Bits, Publication interne du Centre de calcul électronique du Collège de Genève, Nos 1 à 15.
35. Voellmy, Extermann, Tables numériques et logarithmes, Payot.

*articles

ALGOL. Abréviation de l'expression "ALGOrithmic Language", langage plus spécialement destiné à résoudre des problèmes purement mathématiques.

BANDE. Ruban en papier ou magnétique pouvant enregistrer des informations utilisées par un ordinateur. La bande magnétique, beaucoup plus rapide que la bande papier, est une mémoire externe, c'est-à-dire utilisable par la machine pour l'enregistrement de ses informations.

BINAIRE. Symbole n'ayant que deux valeurs 0 et 1, système de numération se basant sur ces deux valeurs.

BIT. Abréviation de "BInary digiT" : la plus petite quantité d'information saisissable, pouvant prendre deux valeurs distinctes : 0 ou 1, oui ou non, + ou −, etc.

BOUCLE. Groupe d'instructions devant être exécutées plusieurs fois successivement jusqu'à un nombre donné de fois ou jusqu'à ce qu'un critère donné soit vérifié.

CHARGEMENT. Opération qui introduit en mémoire centrale un programme à partir de son support de stockage (cartes, bande, disque) et lance la première instruction. (En anglais : LØAD.)

COBOL. Abréviation de "COmmon Business Oriented Language", langage universel à vocation commerciale.

COMPILATION. Opération de l'ordinateur consistant à transformer, par un programme spécial appelé compilateur, un programme écrit en langage évolué en programme en "langage-machine".

CONSOLE. (voir Terminal)

DISQUE. Support d'information servant de mémoire auxiliaire ou externe, fonctionnant sur le même principe que les bandes magnétiques; l'accès aux informations est direct pour le disque, alors qu'il est séquentiel pour la bande magnétique.

FERRITE. Matière ayant deux états magnétiques d'équilibre et permettant d'emmagasiner et de récupérer un bit. Les ferrites se présentent sous forme de tores (anneaux minuscules) qui sont les principaux constituants des mémoires centrales.

FORTRAN. Abréviation de "FORmula TRANslation", principal langage de programmation scientifique.

HARDWARE. Littéralement : quincaillerie, en bon français : désigne tout ce qui occupe l'espace et qui "fait le poids", par opposition à software (voir ce mot).

LABEL. Caractères d'identification des fichiers sur bandes magnétiques situés en tête de chaque bande.

LECTEUR. Organe périphérique servant à introduire dans la mémoire centrale des informations contenues sur des supports externes : cartes perforées, bandes perforées, documents optiques, etc.

LOGICIEL. (voir Software)

MÉMOIRE. Système ou procédé permettant l'enregistrement, la conversation et la restitution d'informations par et pour l'ordinateur.

MÉMOIRE-TAMPON. Elément réduit de mémoire servant de zones provisoires de stockage entre deux organes absorbant ou restituant l'information à des vitesses très différentes. En particulier les mémoires-tampons sont utilisées sur les organes périphériques : par exemple sur le lecteur de cartes, une mémoire-tampon peut stocker la totalité du contenu d'une carte avant de la transférer en mémoire centrale.

MOT. Ensemble de caractères directement adressable dans sa totalité. C'est la plus petite quantité d'information à laquelle on puisse accéder en mémoire centrale ou qui puisse être traitée en unité centrale. Sur CDC 3800 un mot est formé de 48 bits, sur HB 1642 chaque mot à 16 bits.

OFF-LINE, ON-LINE. Mode de connexion de certains périphériques, qui peuvent travailler soit directement reliés à l'ordinateur (on-line), soit de façon autonome (off-line).

ORGANIGRAMME. Représentation graphique et symbolique séquentielle d'un programme ou d'une partie de programme traité par un ordinateur.

PÉRIPHÉRIQUE. Organe externe à l'ordinateur proprement dit, mais travaillant sous son contrôle. Ce sont presque toujours des organes d'entrée-sortie ou de stockage des informations : lecteurs, dérouleurs, disques, imprimantes, etc.

PROCESSEUR. Organe de base de l'ordinateur, qui effectue à la fois le contrôle de l'exécution du programme et les opérations de calcul et de logique.

SÉQUENCE. Suite d'instructions effectuées les unes à la suite des autres sans débranchement.

SOFTWARE. Néologisme argotique inspiré du "hardware" signifiant le complément du dur, du solide. C'est tout ce qu'il y a en plus de la machine et de ses périphériques (programmes-système de tris, de contrôle, etc.). Synonyme : logiciel.

SYSTÈME. Ensemble de programmes sans lesquels l'ordinateur ne peut pas fonctionner, et auxquels il doit pouvoir avoir accès en permanence pendant l'exploitation.

TAMBOUR. Support d'information et de mémoire externe offrant peu de volume et d'une grande rapidité, cependant ils tendent à être abandonnés au profit des disques encore plus rapides.

TEMPS RÉEL. Mode de fonctionnement d'un ordinateur dans lequel les informations reçues demandent une réponse immédiate. Ce n'est donc plus l'ordinateur qui prend l'initiative des appels d'informations, mais ce sont celles-ci qui par leur arrivée aléatoire, obligent la machine à effectuer le traitement dont elles ont besoin.

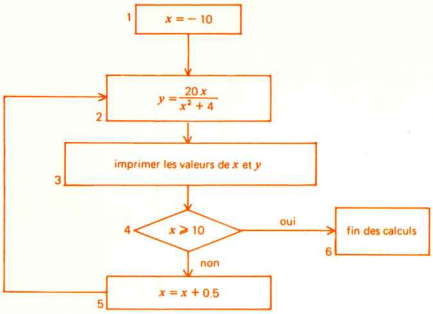
TERMINAL. Organe d'entrée-sortie situé dans une autre pièce, dans une autre ville que l'ordinateur, relié à lui par voie téléphonique. Les terminaux sont utilisés pour l'accès direct à distance (console).

TIME-SHARING. "partage du temps" d'un ordinateur capable de travailler en multiprogrammation et en accès direct à distance. Plusieurs utilisateurs peuvent ainsi simultanément faire exécuter leur programme en le demandant de leur terminal, et avoir l'impression que l'ordinateur leur est exclusivement réservé.

UNITÉ CENTRALE. Partie la plus importante d'un ordinateur composée du processeur et de la mémoire centrale.

Achevé d'imprimer,
pour le compte des Editions Spes S.A., à Lausanne,
en septembre mil neuf cent soixante-quatorze
sur les presses de l'imprimerie Delachaux & Niestlé à Neuchâtel.

éléments de programmation et de langage fortran




```
      PRØGRAM EXMATH
      X=−10.
      DØ 17 K=1,41
      Y=20.*X/(X**2+4.)
      WRITE (9,100)X,Y
17    X=X+0.5
      STØP
100   FØRMAT (F10.1,F15.6)
      END
```

Peu d'ouvrages en informatique s'adressent indifféremment à l'autodidacte, au gymnasien ou à l'étudiant à l'Université. C'est ce tour de force qu'ont réussi MM. G. HAURY et R. MOREL en réalisant ces "Eléments de programmation et de langage FORTRAN". Qu'il s'agisse de la configuration de l'ordinateur, de la réalisation d'un programme ou de l'apprentissage du langage informatique le plus utilisé dans le monde scientifique : le FORTRAN, les auteurs ont réussi à mettre ces notions nouvelles à la portée de chacun. De nombreux exercices corrigés, plusieurs programmes exécutés sur l'ordinateur et reproduits in extenso dans ce livre mettent réellement le lecteur dans la situation du programmeur devant son ordinateur.

Nul ne regrettera l'achat de cet ouvrage de vulgarisation d'un domaine indispensable à l'homme de demain.

spesdunod

ISBN 2-602-00016-7

ORGANISATION EUROPÉENNE POUR LA RECHERCHE NUCLÉAIRE
CERN EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

1211 GENÈVE 23

Téléphone : (022) 41 98 11
Telex : GENÈVE - 2 25 48
Télégramme : CERNLAB-GENÈVE

N. Norel

Votre référence
Your reference

Notre référence
Our reference

A rappeler dans la réponse
Please quote in your reply

Genève, le 10 décembre 1974

Monsieur et cher collègue,

Je vous suis reconnaissant de l'envoi de votre livre que j'ai parcouru avec plaisir. La lecture en est agréable et intéressante et je lui souhaite tout le succès qu'il mérite. A une époque et dans une discipline où l'on s'exprime généralement de plus en plus confusément, la clarté de votre ouvrage surprend ; son efficacité s'affirmera.

Veuillez agréer, Monsieur et cher collègue, l'expression de mes sentiments distingués et, bien que n'ayant pas le plaisir de vous connaître, celle de ma sympathie.

François Louis.