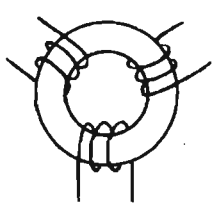
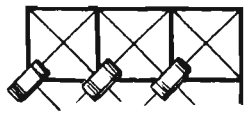
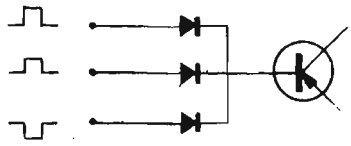
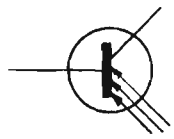
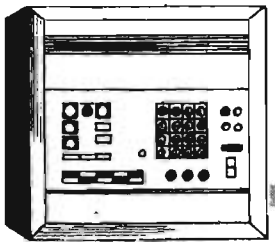








1 + 1 = 10
10 + 10 = 100
1000 - 100 = 100
11 × 11 = 1001



INITIATION AU CALCUL ELECTRONIQUE

BASIC • ALGOL • FORTRAN

(Suite et fin, voir n° 1 347)

ALGOL et BASIC sont deux programmes universels dont l'usage est resté strictement limité, le premier à des calculs scientifiques et mathématiques très particuliers, le second à l'utilisation des ordinateurs en temps partagé : dans ce mode de travail, les programmes de calcul doivent être très « conversationnels » et autoriser des entretiens « téléphoniques » entre l'homme et la machine.

le langage Fortran (FORmule TRANslation) est d'usage courant : il est d'ailleurs quasiment le seul langage universel disponible sur tous les ordinateurs. L'utilisation du Fortran est plus délicate que celle du Basic, et ne peut être envisagée que par des programmateurs déjà expérimentés, au courant du fonctionnement des ordinateurs.

PREMIÈRE DIFFICULTÉ : LE FORMAT

Alors qu'en Basic il était possible d'écrire un nombre arithmétique sans se soucier de sa longueur, le Fortran oblige l'utilisateur à indiquer la forme sous laquelle apparaîtront les nombres arithmetiques, lors de la lecture de données, ou lors de l'impression des résultats.

Ainsi, pour lire le nombre 3.141592, il faudra préciser, à l'ordinateur :

— que le nombre à lire possède 7 chiffres au total ;

— qu'il y a un chiffre avant le point (qui remplace la virgule classique), et,

— qu'il y a 6 chiffres après ce point.

On constate dès à présent une première lacune du Fortran : si un nombre lu à une représentation donnée en un certain instant, il est fort probable qu'en un autre instant la valeur que l'on affectera à ce nombre, lors d'une autre lecture, aura une autre représentation. Ainsi, si on lit d'abord 3.141592, valeur affectée à X, et si, par la suite on veut lire une nouvelle valeur de X : par exemple 10^{-3}, il faudra nécessairement présenter cette dernière valeur par : 0.001000. Le chiffre avant le point, les six chiffres après le point, sont obligatoires.

En Basic, on aurait simplement écrit 3.141592, puis E-3.

Ainsi, en Fortran, une place importante de l'écriture du programme sera consacrée aux formats des nombres lus ou imprimés. Ces formats précisent la mise en page. Ils seront forts utiles en sortie, où l'on pourra créer des tableaux de valeurs, alors que, reconnaissons-le, en Basic, la présentation de résultats de calculs numériques ne varie guère.

LE FORMAT

La syntaxe permettant, en Fortran, de donner un ordre de mise en page, utilise le vocable Format.

Le format n'est pas une instruction de calcul, mais une sorte de gabarit pour les nombres qui sont lus ou imprimés par l'ordinateur. Le format se compose :

— d'une etiquette numerique,

— de l'ordre format,

— d'une liste spécifications entre parenthèses, définissant le gabarit des nombres à lire ou à imprimer.

Il existe 3 types de spécifications comme le présente le tableau V. Ces specifications sont accompagnées d'un « quantificateur » dont le rôle est de spécifier le nombre de colonnes de cartes (lecture par cartes perforées) ou d'états imprimés, concernés par la spécification employée.

Par exemple, 21 X signifie 21 colonnes traitées selon la spécification X.

En lecture de cartes perforees, l'ordinateur « sautera » 21 colonnes.

A l'impression, il restituera 21 espaces blancs sur une ligne.

LECTURE ET IMPRESSION DES DONNÉES

Une carte perforée comporte 80 colonnes consécutives pouvant recevoir, chacune, un chiffre, une lettre ou un caractère spécial.

Un état imprimé contient 128 à 160 colonnes ; l'état le plus classique contient 132 colonnes.

En conséquence, s'il s'agit de cartes perforées, le nombre de caractères lus lors d'une opération d'entrée ne devra jamais dépasser 80. S'il s'agit d'états imprimés, on ne devra pas imprimer plus de 132 caractères par ligne.

LA MISE EN PAGE

La première catégorie de spécifications concerne la mise en page (ou le dessin de cartes perforées).

La spécification X, par exemple, utilisée sous la forme 10 X, permet :

— soit d'ignorer le contenu de 10 colonnes d'une carte perforée, en lecture ;

TABLEAU V — SPÉCIFICATIONS DE FORMAT

Type 1	Type 2	Type 3
Spécifications de mise en page ou de dessin de carte perforée : X : — insertions d'espaces (en impression) — saut de colonnes (en lecture) H : — création de textes alpha-numériques / : — passage à l'enregistrement suivant	Spécifications de traitement de variables : I : pour les nombres entiers E : pour les nombres réels exprimés : (ou D) ● avec une mantisse et ● avec un exposant F : pour les nombres réels exprimés : ● avec une partie entrée et ● avec une partie fractionnaire L : pour les variables logiques A : pour les entrées-sorties en caractères alpha-numériques G : spécification « tout faire »	P : facteur de cadrage

— soit d'insérer *n* espaces « blancs » dans une ligne.

La spécification H permet de créer un texte alphanumérique. On écrit :

n H.....
n caractères

suivi de *n* caractères. Ces *n* caractères vont se retrouver tels quels sur l'état de sortie. Par exemple, " 12HTEMPERATURE : "

Il y a, dans l'exemple précédent, 12 caractères après le H.

Enfin la spécification / est utilisée pour changer de carte (en entrée) ou de ligne (en sortie). Le signe / est appelé « slash ».

En sortie, l'écriture d'un slash dans un format termine la ligne en cours, et l'impression se poursuit au début de la ligne suivante. Si l'on écrit 2 slashes consécutifs, on intercalera alors une ligne blanche entre deux lignes imprimées. Par exemple, le programme suivant :

```
  PRINT 1
1 FORMAT
  (10HRESULTATS ://
  HINTENSITE3X...)
```

permettra d'imprimer :
RESULTATS :
(une ligne est sautée)
INTENSITÉ (trois espaces blancs)

LES SPÉCIFICATIONS DE TYPE 2

Les spécifications de type 2 permettent à l'ordinateur de traiter les nombres qui lui sont présentés en lecture ou d'imprimer, selon les désirs de l'utilisateur, les résultats de calculs.

Ainsi pour lire le nombre entier A, on écrira :

```
      READ 30, A
30    FORMAT (I5)
```

La spécification I (elle provient de l'anglais INTEGER, qui signifie ENTIER), est suivie d'un quantificateur (ici 5). Le nombre A sera lu sur 5 colonnes de la carte perforée.

En impression, l'instruction sera écrite :

```
      PRINT 35, A
35    FORMAT (I5)
```

Si A contient plus de 5 chiffres significatifs, il ne peut pas être imprimé et est remplacé par 5 astérisques. Si A contient moins de 5 chiffres significatifs, le nombre est imprimé, et la zone d'impression est complétée, à gauche du nombre, par des espaces blancs.

Par exemple, si A vaut 101300, l'instruction :

```
      PRINT 35, A, A
35    FORMAT (I4/I7)
```

donnera à l'impression :

```
****
 101300
```

Si le nombre à lire ou à imprimer n'est pas entier, on utilisera le plus souvent,

— la spécification E pour les nombres exprimés en « virgule flottante » ;

— la spécification F pour les nombres écrits sous leur forme ordinaire.

Par exemple, pour imprimer le nombre Pi (Pi = 3.141592), on écrira :

```
      PRINT 100, Pi
100   FORMAT (F9.6)
```

Plus généralement, le format d'une spécification F s'écrira :

FORMAT (F *m* . *n*)

où *m* est le nombre total de positions occupées par le nombre à entrer (signe + ou — et point compris) et *n* le nombre de chiffres de la partie fractionnaire.

Cette spécification ne doit être employée, en sortie, que lorsqu'on est certain, à l'avance, du nombre de chiffres, situés à gauche de la virgule, et contenus dans les nombres à imprimer.

Dans le cas contraire, on utilisera la spécification E.

Cette spécification s'écrit sous sa forme générale :

E *m* . *n*

où *m* représente le total de positions occupées par le nombre, *n* représente le nombre de chiffres à droite du point décimal.

Par exemple pour lire le nombre 1515 sous forme de « virgule flottante », on écrit :

$1515 = 0{,}1515 \times 10^{+4}$

Ici, 1515 représente 4 chiffres de la mantisse et + 4 deux caractères de l'exposant. Plus généralement, un nombre quelconque s'écrira :

$\pm 0.XXXXX\ 10^{\pm YY}$

Les X sont les chiffres de la mantisse, les Y ceux de l'exposant.

Le nombre *m* sera obtenu en faisant la somme :

— du nombre de chiffres de la mantisse (dans 1515, il y a 4 chiffres),

— du nombre de chiffres de l'exposant (dans 10^4, il y a 1 chiffre),

— du nombre de signes nécessaires à la définition de la valeur à entrer ou à imprimer :

* signe de la mantisse,
* signe de l'exposant,
* lettre E représentant l'exposant, symboliquement (10^4 s'écrit E4),
* nombre de chiffres dans la partie décimale,
* point séparant la partie décimale de la mantisse.

Pour lire + $0.1515 \times 10^{+4}$, on définira le format suivant :

F10.4

Le programme suivant :

```
  Z = 1515
  PRINT 1, Z
1 FORMAT (F10.4)
```

donnera à l'exécution :

0.1515 E4

Une spécification E ou F, si elle n'est pas la dernière, ou si elle n'est pas séparée d'un « slash », est séparée de la suivante par une virgule.

LECTURE ET IMPRESSION

L'ensemble des règles concernant le format des nombres déroute souvent le programmeur débutant. Mais une fois ces règles bien admises, l'écriture d'un programme en Fortran paraîtra presque un « jeu d'enfant ».

Les opérations d'entrée-sortie se font le plus souvent avec les ordres suivants :

● Pour lire des données sur des cartes perforées :

READ 100, X, Y, Z

l'instruction de lecture (READ) est suivie d'un étiquette numérique de l'instruction FORMAT correspondante. L'étiquette est suivie de la liste des variables à lire selon le format indiqué.

— Pour imprimer sur papier :

PRINT 110, VITESSE

L'instruction d'impression (PRINT) est suivie de l'étiquette numérique du FORMAT correspondant, puis de la liste des variables à imprimer.

LES NOMBRES EN FORTRAN...

En Fortran, les constantes sont :

— soit des entiers positifs, négatifs ou nuls (en anglais : INTEGER),

— soit des nombres « réels » (en anglais : REAL), exprimés en système décimal, comportant une mantisse, et éventuellement un exposant :

13.5 pourra s'écrire :
0.135 E 2 ou 135 E—1
— 15.10^{-8} pourra s'écrire :
— 15 . E—8 ou —0.15 E—6

L'exposant, en Fortran doit toujours être précédé de la lettre E. Quant aux variables, elles sont également entières ou réelles.

Le langage Fortran considère que toute variable commençant par l'une des lettres,

I, J, K, L, M, N

est une variable entière.

Pour utiliser une variable ne prenant que des valeurs entières et ayant un nom ne commençant pas par l'une des 6 lettres précédentes, on fera la déclaration :

INTEGER SIGMA
BETA 1, BETA 2

Les variables qui suivent la déclaration INTEGER, sont séparées l'une de l'autre par une virgule.

Pour utiliser une variable réelle, dont le nom commence par l'une des lettres I, J, K, L, M, N, il faut la déclarer ainsi :

REAL KAPPA, MU, LAMBDA

Une variable peut, bien entendu, être indicée, et comporter jusqu'à 3 indices. Pour utiliser une telle variable, il faut la déclarer au préalable au moyen de l'ordre :

DIMENSION BETA (J, K, L)
ALPHA (J, K)

Ici, ALPHA et BETA sont les noms des variables ; J, K, L sont des constantes positives entières définissant les valeurs maximales que pourront prendre les indices correspondants. Ainsi :

DIMENSION V(3), A(5,2)

définit, d'une part un vecteur V à 3 composants, d'autre part une matrice A à 5 lignes et 2 colonnes.

... ET LES OPERATEURS ARITHMETIQUES

Le Fortran permet l'utilisation des opérations usuelles sur les nombres :
l'addition +
la soustraction —
la multiplication *
la division /
l'exponentiation **
ainsi que des fonctions utilitaires :

— ABS(X) :
valeur absolue du nombre X
ABS (— 3.14) est égal à 3.14

— INT(X) :
partie entière, d'un nombre réel X, exprimée par un nombre entier
INT(3.14) est égal à 3

— AINT(X) :
partie entière d'un nombre réel X, exprimée par un nombre réel
AINT(3.14) est égal à 3.00

— FLOAT(X) :
conversion en réel d'un nombre entier
FLOAT(3) est égal à 3.00

et des fonctions mathématiques :
SIN(X) pour la fonction sinus
COS(X) pour la fonction cosinus
TAN(X) pour la fonction tangente
ATAN(X) pour la fonction arctangente
EXP(X) pour la fonction exponentielle
ALOG(X) pour le logarithme népérien
SQRT(X) pour la racine carrée d'un nombre X positif.

A partir de ces différentes opérations, on peut construire des formules mathématiques. Par exemple, le discriminant de l'équation : $Ax^2 + Bx + C = 0$ s'écrira :
DELTA = B ** 2. — 4.* A * C
et si DELTA est positif, les racines de l'équation seront égales à :
X1 = (— B + SQRT (DELTA)) /2./A
et X2 = (— B -SQRT (DELTA)) /2./A

On remarque que, dans une telle expression, tous les termes sont, soit entiers, soit réels.

On a multiplié par 4. et non par 4 le terme A * C, produit de deux nombres réels. De même, on a divisé SQRT (DELTA) par 2. et non par 2

C'est la règle d'homogénéité.

Elle est néanmoins de plus en plus abandonnée.

LA LOGIQUE EN FORTRAN

La plupart des compilateurs Fortran autorisent l'emploi d'opérateurs de relation qui s'appliquent à des variables arithmétiques et dont le résultat est une variable logique, prenant les valeurs VRAI ou FAUX. Le tableau VI donne la liste de ces opérateurs.

D'autres operateurs traitent exclusivement des variables logiques.

A titre d'exemple, l'equation du second degré $Ax^2 + Bx + C = 0$ aura deux racines si : A est différent de zéro et si le discriminant est positif ou nul. Il faut donc que la variable logique suivante soit vraie :

A . NE . 0. . AND . B ✳✳ 2. − 4.✳ A ✳ C . GE . 0

Pour qu'il y ait une racine ou une racine double, il faut que :

✳ A = 0 et B ≠ O, ou

✳ $B^2 - 4$ AC = 0 si A ⓣ 0

En FORTRAN, on écrit que la variable logique :

A . EQ . ZER . AND . B . NE . ZER . OR . DELTA . EQ . ZER . AND . A . NE . ZER

Avec ZER = 0.

et DELTA = B ✳✳ 2. − 4 . ✳ A ✳ C

soit vraie.

LA RUPTURE DE SEQUENCE

Les ruptures de séquence interrompent la suite naturelle des instructions symboliques et permettent un branchement vers une autre séquence de calcul.

L'ordre le plus simple est le GOTO, que l'on avait déjà rencontre en Basic, et qu'en Algol, nous aurions écrit ALLERA

GOTO 137

Le calcul se poursuivra à la ligne portant l'étiquette 137.

On peut également établir des aiguillages multi-directionnel au moyen d'un ordre « GOTO calculé » :

GOTO (1, 5, 9, 11), J

Selon que J vaut 1, 2, 3 ou 4, le calcul se poursuit à l'instruction : 1, 5, 9 ou 11. Ces ruptures de séquences inconditionnelles sont complétées par un ordre de rupture conditionnelle : IF. Le IF arithmétique s'écrit :

IF (DELTA) 10, 15, 20

et si DELTA est négatif, le calcul se poursuit à l'étiquette 10
si DELTA est nul, le calcul se poursuit à l'étiquette 15
si DELTA est positif, le calcul se poursuit à l'étiquette 20

Voyons le problème de la résolution de l'équation du second degré

$g = Ax^2 + Bx + C = 0$

dont l'ordinogramme est donné figure 10. La figure 11 en donne le programme FORTRAN.

Une autre possibilité est le IF LOGIQUE : on teste une condition logique, et si celle-ci s'avère vérifiée, une instruction est exécutée :

IF (expression logique) instruction exécutable.

Par exemple :

IF(X . LE . 1.)
C = SQRT (1. − X ✳✳ 2.)

LA BOUCLE DO

L'ordre DO est le plus important de la logique Fortran. Il sert à répéter la séquence de programme qui le suit. Par exemple, pour calculer la somme des N premiers nombres entiers, on écrira le programme suivant :

```
    READ 100, N
    S = 0
    DO 5 I = 1, N, 1
    5 = S + I
    PRINT 100, S
5   CONTINUE
100 FORMAT (I4)
110 END
```

La forme générale de l'ordre DO EST

DO n I = n_1, n_2, n_3.

n est l'étiquette de la dernière instruction de la séquence à répéter.

I est le compteur.

n_1 la valeur initiale du compteur.

n_2 la valeur finale du compteur.

n_3 le pas incrémental de variation de I.

Lorsque n_3 vaut 1, la forme générale de l'instruction DO s'écrit :

DO n I = n_1, n_2

A signaler que l'instruction CONTINUE ne produit aucune instruction particulière. Elle est utile dans l'écriture de bandes DO.

SAVEZ-VOUS PARLER FORTRAN ?

Vous allez prendre connaissance d'un petit problème fort simple. Il n'est absolument pas nécessaire d'avoir un ordinateur pour le résoudre.

Aujourd'hui, M. Martin, du service informatique de la Société Computex doit écrire un programme qui lira 50 valeurs entières quelconques et les rangera dans un ordre croissant.

Pourriez-vous l'aider ?

Marc FERRETTI.

```
      READ 100, A, B, C
100   FORMAT (E 15 . 8)
      IF (A) 110, 120, 110
120   IF (B) 130, 140, 130
130   PRINT 150, A
150   FORMAT (5X,2HX =, 2X,E15.8)
      GOTO 1000
140   IF (c) 160, 170, 160
170   PRINT 175
175   FORMAT (5X,20H PROBLEME INDETERMINE)
      GOTO 1000
160   PRINT 165
165   FORMAT (5X,19H PROBLEME IMPOSSIBLE)
      GOTO 1000
```

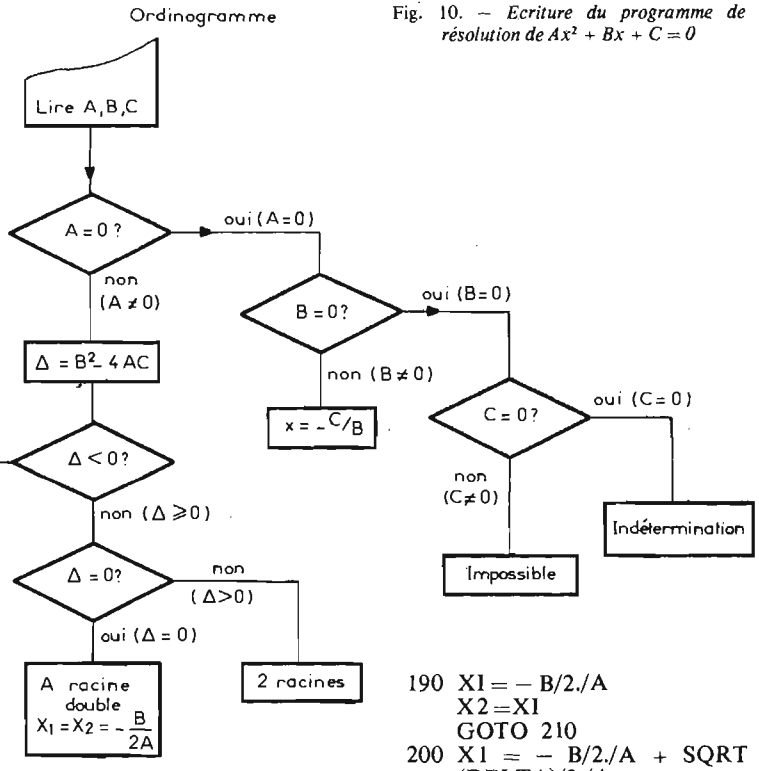


Fig. 10. — *Ecriture du programme de résolution de* $Ax^2 + Bx + C = 0$

```
110 DELTA = B ✳✳ 2. − 4. ✳ A ✳ C
    IF (DELTA) 180, 190, 200
180 PRINT 185
185 FORMAT (5X,20H PAS DE RACINE REELLE)
    GOTO 1000
190 X1 = − B/2./A
    X2 = X1
    GOTO 210
200 X1 = − B/2./A + SQRT (DELTA)/2./A
    X2 = − B/2/A + SQRT (DELTA)/2./A
210 PRINT 220, X1. X2
220 FORMAT (5X, 3HX1=, 2X, E 15.8/5X, 3HX2=. 2X, E15.8)
1000 END
```

TABLEAU VI
Opérateurs de relation et de logique

OPÉRATEURS DE RELATION	
Ecriture	Signification
A = B . EQ . C	A est vrai si B = C faux si B ≠ C
A = B . NE . C	A est vrai si B ≠ C faux si B = C
A = B . LT . C	A est vrai si B < C faux si B ⩾ C
A = B . LE . C	A est vrai si B ⩽ C faux si B > C
A = B . GT . C	A est vrai si B > C faux si B ⩽ C
A = B . GE . C	A est vrai si B ⩾ C faux si B < C

OPÉRATEURS LOGIQUES	
Ecriture	Signification
A = : NOT . B	i A est vrai si B est faux faux si B est vrai
A = B . AND . C	A est vrai si B ET C sont vrais simultanément. Sinon A est faux.
A = B . DR . C	A est vrai si B OU C sont vrais. Si B ET C sont faux, A est faux.