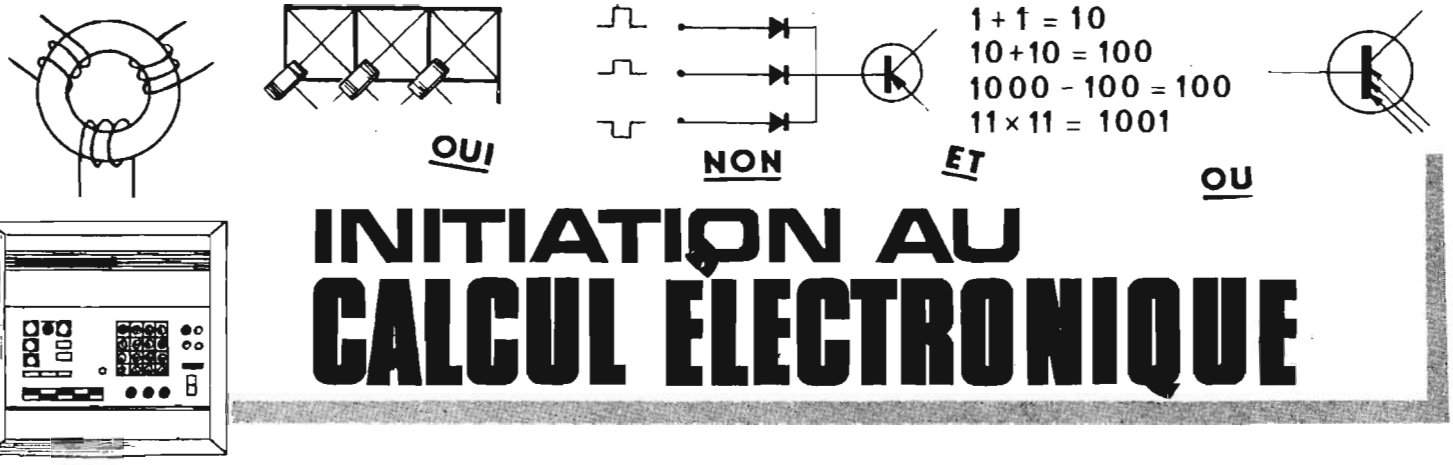

INITIATION AU CALCUL ÉLECTRONIQUE

BASIC • ALGOL • FORTRAN

(Suite. Voir n° 1 334.)

LA IVe Biennale de la langue française, organisée par le Conseil international de la langue française, a pris fin récemment à Menton. M. Jacques Duron, directeur du service des lettres au ministère des Affaires culturelles, y a insisté notamment sur les dangers de la « créolisation » du français.

Les travaux de l'assemblée ont porté essentiellement sur la recherche des termes équivalents susceptibles de remplacer une vingtaine de vocables anglo-saxons communément employés.

Parmi ces vocables, le mot « software ».

53 % des participants ont proposé « programmerie » pour software.

L'AFFECTATION, B.A.BA DE LA PROGRAMMERIE

Comme nous l'avons vu le mois dernier, certaines variables utilisées dans l'ordinateur peuvent changer de valeurs. Pour cela, on utilise une instruction d'affectation :

A : = B × C↑4

Initialement, la machine connait les valeurs respectives de B et de C ; par exemple :

B : = 3 (on affecte à B la valeur : 3) et

C : = 2 (on affecte à C la valeur : 2)

L'ordinateur va alors calculer la puissance quatrième de C (soit 2 × 2 × 2 × 2 = 16), puis va multiplier par 3 le résultat : 3 × 16 = 48.

Le dernier résultat trouvé (48) est affecté à A :

A = 48

Dans la suite du programme, on pourra donc utiliser A à la place du nombre 48 et écrire par exemple :

A : = A + 2

A valait 48, on lui ajoute 2, et A sort du calculateur avec la valeur 50. La valeur que possédait A avant d'entrer dans l'ordinateur est définitivement oubliée : la machine ne sait plus qu'avant d'avoir affecté 50 à A, elle lui avait affecté la valeur 48.

Il est bien évident que toutes les expressions algébriques à la droite du signe d'affectation doivent avoir une valeur. Si vous écrivez, dans un programme ALGOL,

DEBUT
REEL B, A, C, DELTA ;
DELTA : =
RAC2 (B↑ − 4 × A × C) ;
FIN

L'ordinateur, ne connaissant pas les valeurs affectées à A, B, C, ne sera pas en mesure de calculer la racine carrée de l'expression $B^2 - 4.A.C$ et ne pourra pas affecter de valeur à DELTA. Un compilateur bien fait détecte l'erreur, arrête le déroulement du programme et signale l'erreur à l'utilisateur.

LES ORDRES DE BRANCHEMENT

Considérons la suite de nombres $u_1, u_2, u_3, u_4, \ldots, u_{n-1}, u_n$ définis par récurrence de la manière suivante :

$$u_n = \left(u_{n-1} + \frac{A}{u_{n-1}}\right)/2.$$

A étant un nombre positif quelconque, on démontre mathématiquement que cette suite « converge »

Si on calcule :

$$u_2 = \frac{1}{2}\left(u_1 + \frac{A}{u_1}\right),$$

puis successivement :

$$u_3 = \frac{1}{2}\left(u_2 + \frac{A}{u_2}\right),$$

$$u_4 = \frac{1}{2}\left(u_3 + \frac{A}{u_3}\right),$$

...

$$u_n = \frac{1}{2}\left(u_{n-1} + \frac{A}{u_{n-1}}\right).$$

Il arrive un moment où les nombres tendent vers une limite définie. Ainsi, ici, lorsque n est grand, u_{n-1} et u_n ont des valeurs très proches l'une de l'autre, et plus n augmente, plus u_0 se rapproche d'une valeur limite que nous appellerons u. Lorsque n devient infini, on atteint la valeur u, qui, bien sûr, est la solution de l'équation :

$$u = \frac{1}{2}\left(u + \frac{A}{u}\right)$$

On réduit au même dénominateur les deux membres de cette équation :

$$2u^2 = u^2 + A$$

ce qui donne $u^2 = A$

La limite u des nombres u_1, u_2, u_3, u_4, u_5,..., u_{n-1}, u_n,... est donc la racine carrée du nombre positif A.

C'est une méthode de calcul sur ordinateur de la racine carrée d'un nombre positif quelconque.

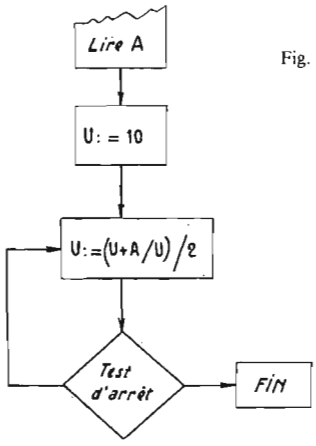

Fig. 2

La figure 2, représente « l'ordenogramme » du programme de résolution de la racine carrée d'un nombre A positif quelconque.

On part d'une première valeur u_0, après avoir lu la valeur de A sur un périphérique quelconque.

Ici $u_0 = 10$.

En appliquant la formule $u_n = \frac{1}{2}\left(u_{n-1} + \frac{A}{u_n}\right)$ on calcule les valeurs successives u_1, u_2,... tant qu'un test d'arrêt n'aura pas mis fin au calcul. Or, on constate que le calcul est toujours le même : on part d'un nombre u, on calcule l'expression arithmétique $\frac{1}{2}\left(u + \frac{A}{u}\right)$ puis on affecte le résultat à la variable u.

Pratiquement, la machine, dans ce programme, ne se souviendra pas des valeurs u_0, u_1, u_2,... prises successivement par la variable u. Elle ne conservera en mémoire que la dernière valeur calculée.

Comme on le voit sur la figure 2, il y a une boucle de calcul, que la machine suit jusqu'au moment où le test d'arrêt aura été satisfait.

L'ECRITURE DE LA BOUCLE

Pour écrire, en ALGOL, une boucle, on a besoin de :

a) une étiquette : c'est le nom que l'on donnera une fois pour toutes à une instruction. Une étiquette, en ALGOL, est constituée d'une ou plusieurs lettres, suivies éventuellement d'un nombre entier.

Par exemple, on pourra choisir comme étiquette :

ITERATION

ou encore

ITERATION 1

Et on écrira :

ITERATION : U : = (U + A/U)/2 ;

b) Un ordre de branchement qui se nomme, en ALGOL : **ALLERA.** Cet ordre est suivi d'une étiquette qui nomme l'instruction à laquelle la machine devra se rendre.

La boucle de calcul de la racine carrée d'un nombre A s'écrira donc :

DEBUT
REEL U ;
U := 10 ;
ITERATION : U := (U + 101300/U)/2 ;
ALLERA ITERATION ;
FIN.

Dans cet exemple, on calcule la racine carrée du nombre 101300.

Les langages BASIC et FORTRAN admettent également des étiquettes. Nous en reparlerons plus tard.

LES INSTRUCTIONS CONDITIONNELLES

Le problème, dans l'exemple précédent est lié au fait que la machine ne s'arrêtera jamais : elle ira constamment à l'étiquette ITERATION. Il faut pouvoir arrêter la boucle. On y parvient avec un test d'arrêt : dans le cas de la racine carrée, la machine demandera dans chaque boucle, si u_n est très voisin de u_{n-1}.

Pour chiffrer la proximité de u_n et de u_{n-1}, on est amené à fixer la précision du calcul. Par exemple, la différence $u_n - u_{n-1}$ en valeur absolue, est-elle inférieure à 0,001 ? Si la différence est inférieure à 0,001 (ce sera la précision du calcul), on arrête la boucle. Sinon, on continue la boucle.

Pratiquement, l'ordinogramme du programme s'écrit comme l'indique la figure 3.

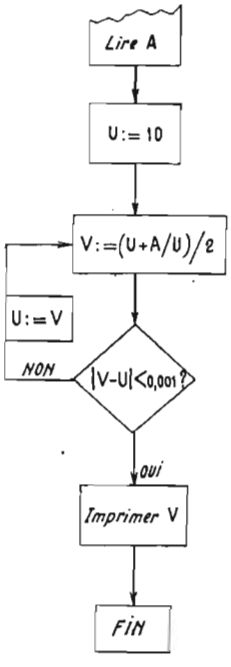

Fig. 3

La première constatation qui s'impose est la nécessité d'introduire deux variables pour pouvoir comparer u_n et u_{n-1}. Ici, on remplace u_{n-1} par U et u_n par V.

Si, en valeur absolue, la différence V-U est supérieure à la précision du calcul (fixée ici à 0,001), on va refaire la boucle. Mais il ne faut pas oublier, en recommençant le calcul, d'affecter à U la dernière valeur trouvée pour V. On y arrive en écrivant :

U := V

V avait au temps $t = 0$, une certaine valeur. Au temps $t = 1$ (l'unité de temps est arbitraire), on donne à U la valeur qu'avait V. Donc au temps $t = 1$, U et V ont la même valeur.

Au temps $t = 2$, on revient à la formule

V := (U + A/U)/2

Ce qui modifie la valeur de V, pour passer dans le test.

Dans le test, on compare la valeur de V au temps $t = 2$, à la valeur de U un temps $t = 2$ (c'est la même valeur qu'au temps $t = 1$). Si U et V sont suffisamment proches l'un de l'autre, la machine imprimera le dernier résultat obtenu de l'itération, sinon, elle recommence la boucle.

La machine arrêtera donc l'itération seulement lorsque les deux valeurs U et V seront suffisamment voisines l'une de l'autre.

En ALGOL, ce programme s'écrira sous la forme :

SI... ALORS... :

Après le **SI,** on trouvera une expression booléenne, c'est-à-dire une relation d'ordre complexe (l'expression booléenne peut inclure des ET et des OU logiques). Après l'instruction **ALORS** se trouvera une instruction inconditionnelle.

Si l'expression booléenne n'est pas vérifiée, la machine travaillera en séquence, c'est-à-dire qu'elle sautera l'instruction **ALORS** et ira à l'instruction qui suit. On peut également envisager, en ALGOL, de réaliser une autre instruction conditionnelle lorsque l'expression booléenne n'est pas vérifiée : on peut alors, si on le désire, écrire l'ordre de branchement sous la forme :

SI (A) **ALORS** (I_1) **SINON** (I_2) ;

Si l'expression booléenne (A) est vraie, on réalise l'instruction (I_1) ;

Si l'expression booléenne (A) est fausse, on réalise l'instruction (I_2).

Comment écrira-t-on alors le programme de la racine carrée ? On suppose que le nombre A dont on recherche la racine carrée vaut 101300 :

DEBUT
REEL U, V ;
U := 10 ;
ITERATION : V := (U + 101300/U)/2 ;
SI ABS (V-U)< 0,001 **ALORS ALLERA** IMPRESSION **SINON ALLERA** ITERATION ;
IMPRESSION : IMPRIMER (V) ;
FIN

L'instruction booléenne est ici une relation d'ordre simple ; les instructions (I_1) et (I_2) sont deux instructions **ALLERA ;** enfin, on a ajouté une étiquette IMPRESSION.

On aurait très bien pû éviter d'ajouter cette étiquette en modifiant légèrement l'écriture du programme.

DEBUT
REEL U, V ;
U:= 10 ;
ITERATION : V := (U + 101300/U)/2 ;
SI ABS (V − U) ⩾ 0,001 **ALORS ALLERA** ITERATION ;
IMPRIMER (V) ;
FIN

Dans cette programmation, on a inversé la relation d'ordre ; tant qu'elle est vérifiée, on va à l'étiquette ITERATION ; quand elle n'est plus vérifiée, on saute l'instruction qui suit **ALORS** et on opère en séquence puisqu'il n'y a pas **SINON.**

UNE INSTRUCTION COMPOSEE

On vient de voir que l'instruction inconditionnelle s'écrit sous sa forme générale :

SI (A) **ALORS** (I_1) **SINON** (I_2)

L'ALGOL est un langage plein de possibilités. (I_1) et (I_2) peuvent en fait être une succession d'instructions. La forme d'écriture sera alors la suivante :

SI (A) **ALORS DEBUT** (I_3) ; (I_4) ; (I_5) ; (I_6) ; **FIN SINON DEBUT** (I_7) ; (I_8) ; (I_9) ; (I_{10}) ; (I_{11}) ; **FIN ;**

Si l'expression booléenne (A) est vraie, la machine réalise les instructions (I_3) à (I_6) en séquence ; si l'expression booléenne est fausse. elle réalise les instructions (I_7) à (I_{11}) en séquence.

LA LOGIQUE BOOLEENNE

Nous reviendrons rapidement sur l'algèbre de Boole qui a été déjà traitée, dans le cadre de la présente série d'articles (tableau I) voir le « Haut-Parleur » nos 1211 et 1215.

Les opérateurs logiques dont on dispose en ALGOL sont :

– la négation : si l'opérande P est vrai, l'opérande « NON-P » est faux, et vice-versa

– les opérateurs : ET, OU, IMPLICATION et EQUIVALENCE.

P	Q	ET	OU	IMPLICATION	EQUIVALENCE	NON-P
VRAI	VRAI	VRAI	VRAI	VRAI	VRAI	FAUX
FAUX	VRAI	FAUX	VRAI	VRAI	FAUX	VRAI
FAUX	FAUX	FAUX	FAUX	VRAI	VRAI	
Symbole	représentatif	∧	∞	⊃	Ξ	Ö

TABLEAU I

Soient deux opérandes P et Q :
P peut être vrai ou faux
Q peut être vrai ou faux
tout comme un contact électrique est ouvert ou fermé.

L'opérande « P ET Q » est vrai si P est vrai en même temps que Q est vrai.

L'opérande « P OU Q » est vrai si l'un ou l'autre (ou les deux) opérandes considérés (P ou Q) est vrai.

L'opérande « P EQUIVALENT A Q » est vrai si P et Q sont vrais ou sont faux simultanément.

L'opérande « P IMPLIQUE Q » est vrai si Q est vrai ou si P et Q sont vrais ou faux simultanément.

Le tableau II donne le tableau de vérité de ces quatre opérateurs.

En ALGOL, on peut définir des variables booléennes qui ne peuvent prendre que les valeurs logiques VRAI ou FAUX. Il faut déclarer, en tête du programme la présence de telles variables, par :

BOOLEEN B ;

B sera ici la variable booléene, qui ne peut qu'être vraie ou fausse.

RETOUR SUR L'INSTRUCTION COMPOSEE

L'instruction inconditionnelle sous sa forme la plus générale, s'écrit, rappelons-le :

SI (A) **ALORS** (I_1) **SINON** (I_2) ;

(A) est une expression booléenne qui peut comporter en fait plusieurs opérations booléennes. Nous allons revenir sur le calcul de la racine carrée pour expliquer ce détail.

On va introduire un compteur dans l'ordinogramme de calcul de la racine carrée. Ce compteur permettra d'arrêter le calcul si la boucle est parcourue un trop grand nombre de fois (Fig. 4).

Le compteur sera obtenu par un test sur un nombre entier N qui augmente d'une unité chaque fois que l'on parcourt la boucle. Si le nombre entier est plus grand que 1000, par exemple, on arrête le calcul.

Ce programme s'écrit alors :

```
DEBUT
REEL U, V ;
ENTIER N ;
U : = 10 ;
N : = 0 ;
ITERATION : V =
      (U + 101300/U)/2 ;
SI ABS (V − U) ⩾ 0,001 ET
N⩽ 1000 ALORS ALLERA
SUITE 1
SINON ALLERA SUITE 2 ;
SUITE 1 : U : = V ;
N : = N + 1 ;
ALLERA ITERATION
SUITE 2 : SI ABS (V − U)
< 0,001 ALORS ALLERA
SUITE 3 ;
IMPRIMER (« RESOLUTION
IMPOSSIBLE ») ;
ALLERA FIN ;
SUITE 3 : IMPRIMER (V) ;
FIN : FIN
```

On constate que l'introduction d'un compteur, souvent nécessaire a alourdi le programme. Il est bien évident que d'autres variantes peuvent être trouvées pour l'écriture du programme : on pourrait très bien trouver d'autres formes qui ne nécessitent pas l'utilisation de quatre étiquettes.

En dernier lieu, on remarquera, dans les figures 2, 3 et 4, le mode de représentation symbolique d'une instruction de lecture, d'une instruction de calcul et d'un test dans un ordinogramme.

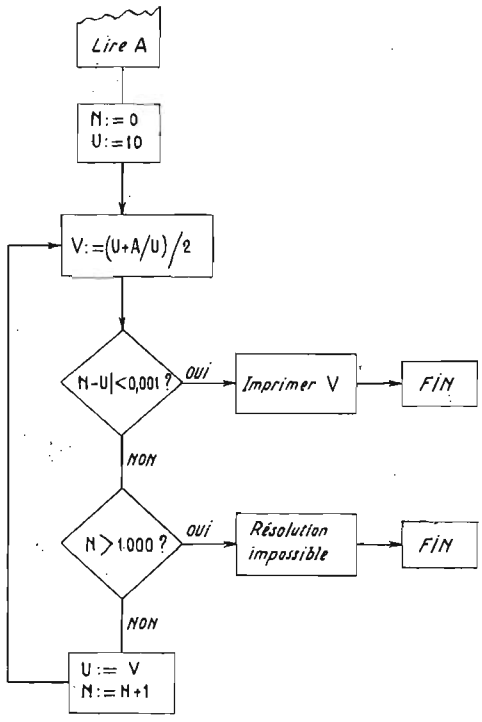

Fig. 4

TABLEAU I
RETOUR SUR L'ALGEBRE DE BOOLE

L'Algèbre de Boole travaille sur des grandeurs qui ne peuvent prendre que deux positions : on dit encore qu'elle traite des propositions vraies ou fausses. Par exemple : « il fait beau aujourd'hui » est une proposition qui est vraie ou fausse ; « la porte est fermée » est une proposition qui est vraie ou fausse.

Soient alors deux propositions P et Q quelconques, qui peuvent être vraies ou fausses. On définit des opérandes qui créent une nouvelle proposition à partir d'une ou plusieurs propositions élémentaires.

NEGATION : si la proposition « il fait beau » est fausse, la proposition « il ne fait pas beau » est évidemment vraie.

Plus généralement on dit que NON-P est vraie si P est fausse. NON-P est noté, en algèbre de Boole : $\overline{P}$

OPERATEUR ET : « il pleut » ET « j'ai un imperméable », donc je sors dehors. Si l'une ou l'autre des propositions « il pleut », « j'ai un imperméable » n'est pas vraie, je ne sors pas.

Plus généralement, l'opérande P ET Q est vrai si P et Q sont vrais simultanément. Sinon cet opérande est faux.

OPERATEUR OU : « il pleut » OU « je suis malade » et je ne sors pas dehors. Ici il suffit que l'un ou l'autre des deux opérateurs (ou les deux) soit vérifié pour que je ne sorte pas.

Donc l'opération P OU Q est vrai si P est vrai, ou si Q est vrai ; ou encore si P et Q sont vrais simultanément. Mais P QU Q est faux si P et Q sont faux simultanément.

On peut encore dire que l'on a NON − (P OU Q) si l'on a NON-P et NON-Q, d'après les définitions qui précèdent soit : $\overline{P \text{ OU } Q} = \overline{P}$ ET $\overline{Q}$.

On démontre aisément que $\overline{P \text{ ET } Q} = \overline{P}$ OU $\overline{Q}$

TABLEAU II

P	*Q*	*ET*	*OU*	*IMPLICATION*	*ÉQUIVALENCE*	*NON-P*
VRAI	*VRAI*	*VRAI*	*VRAI*	*VRAI*	*VRAI*	*FAUX*
VRAI	*FAUX*	*FAUX*	*VRAI*	*FAUX*	*FAUX*	
FAUX	*VRAI*	*FAUX*	*VRAI*	*VRAI*	*FAUX*	*VRAI*
FAUX	*FAUX*	*FAUX*	*FAUX*	*VRAI*	*VRAI*	
Symbole représentatif		∧	*V*	⊃	≡	¬P

SAVEZ-VOUS PARLER ALGOL ?

Vous allez prendre connaissance d'un petit problème fort simple. Il n'est absolument pas nécessaire d'avoir un ordinateur pour le résoudre. Mais nous voulons vous initier au travail de l'informaticien, sans trop vous fatiguer.

Ce premier exercice sera suivi d'un certain nombre d'autres. Bonne initiation !

N'hésitez pas à nous écrire en cas de difficulté !

RESOLUTION DE L'EQUATION DU SECOND DEGRE

Soit l'équation $ax^2+bx+c=0$, dont on cherche les solutions. On doit considérer plusieurs cas de calcul :

1) si $a = 0$, on trouve alors d'autres possibilités

1.1) b n'est pas nul, il y a une solution $x = -\frac{c}{b}$

1.2) b est nul, et

1.2.1.) c est nul, il y a une infinité de solutions puisque quelle que soit la valeur donnée à x, on a toujours $0 = 0$.

1.2.2.) c n'est pas nul, il y a impossibilité car on a à gauche de l'égalité une expression non nulle, et à droite, une expression nulle.

2) Si c n'est pas nul, on doit calculer le discriminant :

$$DELTA = b^2 - 4\,ac$$

2.1) si DELTA = 0, il y a une racine double :

$$x_1 = x_2 = -\frac{b}{2a}$$

2.2) si DELTA est positif, il y a deux racines :

$$x_1 = \frac{-b + RAC2\,(DELTA)}{2a}$$

$$x_2 = \frac{-b - RAC2\,(DELTA)}{2a}$$

bien entendu, RAC2 (DELTA) si l'écriture ALGOL de la racine carrée de DELTA

2.3) Si DELTA est négatif, il n'y a pas de solutions réelles.

Le problème est posé. L'utilisateur aura à introduire les nombres *a, b, c*. La machine donnera, éventuellement les racines.

Pouvez-vous en écrire l'ordinogramme et rédiger le programme de résolution ?

Marc FERRETTI.

(à suivre)