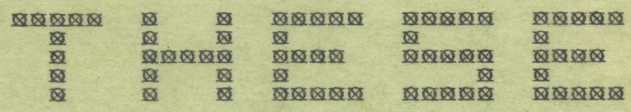
UNIVERSITE DE NANCY FACULTE DES SCIENCES

ACHEVEMENT D'UN COMPILATEUR ALGOL POUR I. B. M. 1620 TRAITEMENT DES PROCEDURES



pour l'obtention du

DOCTORAT de SPECIALITE MATHEMATIQUES (3ème CYCLE)

Soutenue devant le jury le 4 mars 1966

par

Alain F L O C ' H

° ° °

Jury : Mr J. LEGRAS Président

Mr M. DEPAIX

Mr C. PAIR Examinateurs

UNIVERSITE DE NANCY FACULTE DES SCIENCES

ACHEVEMENT D'UN COMPILATEUR ALGOL POUR I. B. M. 1620 TRAITEMENT DES PROCEDURES

pour l'obtention du

DOCTORAT de SPECIALITE MATHEMATIQUES (3ème CYCLE)

Soutenue devant le jury le 4 mars 1966

par

Alain F L O C ' H

° ° °

Jury : Mr J. LEGRAS Président
Mr M. DEPAIX
Mr C. PAIR Examinateurs

UNIVERSITE DE NANCY

FACULTE DES SCIENCES

Doyen : M. AUBRY

Assesseur : M. GAY

Doyens honoraires : MM. CORNUBERT - DELSARTE - URION - ROUBAULT.

Professeurs honoraires : MM. CROZE - RAYBAUD - LAFITTE - LERAY - JOLY - LAPORTE - EICHHORN - GODEMENT - DUBREIL - L.SCHWARTZ - DIEUDONNE - DE MALLEMANN - LONGCHAMBON - LETORT - DODE - GAUTHIER - GOUDET - ULMER - CORNUBERT - CHAPELLE - GUERIN - CHEVALLIER - WAHL.

Maîtres de conférences honoraires : MM. LIENHART - PIERRET.

Professeurs

MM.	URION	Chimie Bio.	FRUHLING	Physique		GARNIER	Agronomie
	DELSARTE	Analyse sup.	SUHNER	Physique expérim.		WEPPE	Minéralogie Appl.
	ROUBAULT	Géologie	HILLY	Géologie		BERNARD	Géologie Appl.
	VEILLET	Biologie animale	LE GOFF	Génie chimique		CHAMPIER	Physique
	BARRIOL	Chimie théorique	CHAPON	Chimie biologique		REGNIER	Physico-chimie
	BIZETTE	Physique	HEROLD	Chimie miné.ind.		GAY	Chimie biologique
	GUILLIEN	Electronique	SCHWARTZ	Exploit.minière		WERNER	Botanique
	GILBERT	Chimie Physique	GAYET	Physio. animale		CONDE	Zoologie
	LEGRAS	Mécanique Ration.	MALAPRADE	Chimie		STEPHAN	Zoologie
	BOLFA	Minéralogie	HADNI	Physique		EYMARD	Calcul dif.et int.
	NICLAUSE	Chimie	BONVALET	Méca. Appliquée		LEVISALLES	Chimie organique
	FAIVRE	Physique Appl.	KERN	Minéralogie		CAPELLE	Méca.Rationnelle
	AUBRY	Chimie minérale	BASTICK	Chimie		MANGENOT	Botanique
	DUVAL	Chimie	DUCHAUFOUR	Pédologie	Mmes	HERVE	Méthodes math.phys.
	COPPENS	Radiogéologie	NEEL	Chimie org.indus.		BASTICK	Chimie MPC Epinal

Maîtres de Conférences

MM.	GOSSE	Mécanique physique	BLAZY	Minéralogie appliquée (ENSG)
	ROCCI	Géologie	BALESDENT	Thermodynamique chimique appliquée
	VUILLAUME	Psychophysiologie	JANOT	Physique MPC (Epinal)
	GUDEFIN	Physique	JACQUIN	Pédologie et chimie agricoles
	HORN	Physique propédeutique	CACHAN	Entomologie appliquée (ENSA)
	FRENTZ	Biologie animale	N...	Probabilités et Statistiques
	AUROUZE	Géologie	N...	Mécanique (ISIN)
	MARI	Chimie (ISIN)	N...	Physique MGP
	LAFON	Physique (ISIN)	N...	Mathématiques
	FELDEN	Physique théor.& nucl.	N...	Mécanique expérimentale
	FLECHON	Physique MPC	N...	Génie chimique
	VIGNES	Métallurgie	N...	Mathématiques propédeutiques
Mle	HUET	Mathématiques SPCN	N...	Phytopathologie
	DEVIOT	Physique du solide	N...	Mathématiques SPCN

Je tiens à exprimer ma profonde gratitude à Monsieur le Professeur J. LEGRAS, Directeur du Centre de Calcul Automatique, pour l'intérêt avec lequel il a toujours suivi mon travail et pour toute l'aide qu'il m'a apportée pendant plus de deux ans. Je l'assure de mon entier dévouement.

J'adresse mes plus vifs remerciements à Monsieur C. PAIR pour la bienveillante attention et l'intérêt constant avec lesquels il a suivi et dirigé ce travail, tant sur le plan théorique que sur le plan pratique.

Je remercie Messieurs les Professeurs qui ont bien voulu me faire l'honneur de composer le jury.

Mes remerciements vont également à Madame Créhange, ainsi qu'à Messieurs André, Cusey et Laporte pour leur collaboration aussi amicale qu'appréciable.

Je remercie Monsieur Procyk, Ingénieur I.B.M. et Monsieur Prince, Chef du Service Mécanographique à la Société Caltex qui ont favorisé mon travail.

Je n'oublierai pas ici Mesdames les Secrétaires du Centre de Calcul pour les nombreux services qu'elles m'ont aimablement rendus.

SOMMAIRE

CHAPITRE I

GENERALITES SUR LE COMPILATEUR

I - LE PROBLEME DE LA COMPILATION

C'est celui de la correspondance sémantique entre deux programmes écrits dans des langages différents. Le problème de la correspondance entre ALGOL et le langage machine I. B. M. 1620 a été traité en détail dans [3] par M. CUSEY.

II - RAPPELS DE QUELQUES POINTS UTILES

A - Lecture du programme-source

Le programme Algol est perforé sur cartes, et ces cartes sont lues suivant le mode alphanumérique à partir de la mémoire SE dans une zone de 211 positions ; leur contenu est analysé par le compteur MEMSE, qui indique à chaque instant l'adresse du dernier caractère lu ; le traitement des blancs, le problème de la continuité du programme entre deux cartes successives, la lecture de nouvelles cartes, sont effectués par le sous-programme DECAL.

B - Analyse syntaxique du programme

Elle s'effectue grâce à deux piles, O et V, dont les sommets respectifs S(O) et S(V) sont indiqués par les compteurs d'adresse MEMSO et MEMSV.

La pile O reçoit les équivalents numériques des symboles de base Algol, elle se remplit suivant les adresses croissantes et se trouve en mémoire en face de la pile V.

La pile V reçoit les adresses α des zones affectées aux nombres, aux variables ainsi qu'aux résultats partiels.

C - Fonctionnement du compilateur - Génération du programme-objet

Lorsqu'un symbole x sort de la pile O, on exécute une séquence du programme de compilation $\sum(x)$. Si un symbole entre sur la pile O, on exécute une séquence $\sum'(y,x)$, y désignant le sommet de la pile.

En particulier, ces $\sum$ ou $\sum'$ construisent une suite d'ordres Γ ou Γ' du programme-objet et modifient l'état des piles.

Les Γ et Γ' sont construits dans une zone de travail (C4) puis transférés dans une zone de stockage (C3) ; le compteur MPGOB indique à chaque instant l'adresse dans C3 à partir de laquelle on peut ranger une nouvelle séquence.

Pour éviter de garder en mémoire tout le programme-objet, on appelle un sous-programme de perforation à chaque entrée dans un $\sum$ ou un $\sum'$, et, au cours du traitement des procédures, chaque fois que l'on risque d'avoir à construire une séquence du programme-objet trop longue pour être contenue dans les zones réservées à cet effet.

D - Les tables d'identificateurs

D. 1 - Renseignements placés dans la table d'identificateurs

xxxxxxxxxxxxxx xxxxx x x
(1) (2) (3) (4)

(1) L'identificateur occupe 14 positions (7 caractères alpha-numériques)

(2) L'adresse correspondante occupe 5 positions

(3) Le type occupe 1 position

(4) Une zone de deux positions est utilisée par les étiquettes et les aiguillages pour garder le "niveau de procédure" (cf. [3]) et par les tableaux pour indiquer leur dimension.

L'adresse α affectée à l'identificateur est placée dans la table au moment des $\sum$ (S(O) = déclarateur) ou des $\sum'$ (S(O) = déclarateur, x=,). Cette adresse d'exécution est prise dans un compteur d'affectation de zone,

- MRINT s'il s'agit d'une variable locale ou d'un tableau local,
- REMAN s'il s'agit d'une variable rémanente.

Le compteur utilisé progressera du nombre des positions nécessaires (12 pour un entier ou un réel, 1 pour un booléen).

En ce qui concerne un aiguillage ou une étiquette, α désignera ou bien l'adresse vraie dans le programme-objet, ou bien une adresse indirecte prise sur REMAN, suivant que l'on rencontre d'abord la déclaration ou l'appel de l'identificateur traité.

Les différents indicatifs de type sont :

0 pour un réel
1 pour un entier
2 pour un aiguillage
3 pour un booléen
4 pour une étiquette
$\overline{5}$ pour un tableau de type réel
5 pour un tableau de type entier
$\overline{6}$ pour un tableau de type booléen.

D. 2 - Cas des identificateurs liés aux procédures

Pour les identificateurs de procédure, les paramètres formels et les identificateurs locaux des procédures, les renseignements mis dans la table sont trop nombreux pour pouvoir tenir sur une seule ligne. Aussi, bien qu'ils ne nécessitent pas exactement 22 positions supplémentaires, leur accorde-t-on une seconde ligne, ceci afin de ne pas alourdir les consultations de table.

Le schéma général de ces deux lignes est le suivant :

première ligne	xxxxxxxxxxxxxx			xxxxx		x	xx			
	1			2		3	4			
seconde ligne	x	xx	xx	xxx	x	xxxx	x	xxxxx	x	xx
	5	6	7	8	9	10	11	12	13	14

Certains points sont communs aux identificateurs de procédures aux identificateurs locaux et aux paramètres formels. Ce sont :

(1) L'identificateur sur 14 positions

(2) Un indicatif spécial de présence de seconde ligne, toujours 8

(9) Un indicatif de début de seconde ligne, toujours 9

(7) Sur deux positions, le niveau de procédure (cf. chapitre 3)

(12) L'adresse réservée pour l'exécution.

D'autre part, (6) permet de distinguer entre elles les différentes catégories traitées ; c'est le "distingo" égal à :

01 pour un identificateur de procédure

03 pour un identificateur local de procédure

04 pour un paramètre formel appelé par valeur

05 pour un paramètre formel appelé par nom.

Pour un identificateur de procédure

(2) représente l'adresse sur REMAN indiquant le début de la procédure dans le programme-objet

(8) indique le nombre de paramètres de la procédure

(9) repère plus loin "position T" (cf. fin chapitre 3)

(13) type de la fonction procédure (donc 0, 1 ou 3), ou indicatif d'instruction procédure, (9) dans ce cas ; cette position est appelée T1.

Pour les paramètres formels, on aura (11), appelé aussi position T2, et en (13) le résultat des spécifications ; pour des spécifications réel, entier, booléen, aiguillage ou étiquette, on aura :

T1 = T2 = 0, 1, 3, 2 ou 4 respectivement.

En ce qui concerne les indicatifs des paramètres tableaux ou procédures, qui peuvent avoir une double spécification, les résultats sont les suivants :

TABLEAU	T1 = 0 et T2 = $\overline{5}$
	de même pour REEL TABLEAU
ENTIER TABLEAU	T1 = 1 et T2 = 5
BOOLEEN TABLEAU	T1 = 3 et T2 = $\overline{6}$
PROCEDURE	T1 = T2 = 9 (9 : indicatif des instructions procédures)
REEL PROCEDURE	T1 = 0 et T2 = 8 (8 : indicatif des fonctions procédures)
ENTIER PROCEDURE	T1 = 1 et T2 = 8
BOOLEEN PROCEDURE	T1 = 3 et T2 = 8

La zone (4) servira, dans le cas des paramètres spécifiés tableaux, à indiquer leur dimension dès que celle-ci pourra être connue.

D. 3 - La première lecture

Le compilateur que nous avons écrit nécessite deux lectures successives du programme Algol. Un premier passage a en effet été rendu nécessaire pour permettre le traitement d'identificateurs avant leur déclaration, ce qui peut se produire dans les cas suivants :

- appels dits croisés entre procédures et entre aiguillages,
- utilisation dans une expression de désignation d'une étiquette avant sa déclaration (si l'on veut bien considérer comme déclaration d'une étiquette la rencontre de celle-ci suivie de "deux points").

La table de première lecture est bâtie à partir de l'adresse TABETI. On construit une ligne de cette table :

- en y rangeant certains renseignements à chaque entrée ou sortie de bloc,
- à chaque déclaration d'aiguillage,
- à chaque rencontre d'étiquette suivie de "deux points".

On ajoute une double ligne à la table :

- pour chaque identificateur de procédure au moment de sa déclaration,
- pour chacun des paramètres formels afférents à cette déclaration.

La structure de la table de première lecture permet de transférer lors du second passage, pendant la compilation proprement dite, dans la deuxième table d'identificateurs, les aiguillages, les étiquettes, les identificateurs de procédures (et leurs paramètres formels éventuellement), et ceci sans avoir à transférer ceux des blocs intérieurs s'ils existent.

Cette façon de procéder oblige à certaines précautions en ce qui concerne les paramètres formels.

En effet, le fait de rattacher les paramètres formels à l'identificateur de procédure (donc au bloc dans lequel a été déclarée la procédure et non pas au bloc intérieur dans lequel ils sont utilisés), peut amener des confusions. Considérons l'exemple suivant :

```
DEBUT ENTIER A,B ;
      ENTIER PROCEDURE F(X) ; ENTIER X ; F:= ε(X) ;
      ENTIER PROCEDURE G(X) ; ENTIER X ; G:= ε(X) ;
      ... F(A) ... G(B) ...
FIN
```

Les deux paramètres formels, X, du fait de leur rattachement au bloc extérieur sont placés dans la table de telle façon que ce serait toujours le même (le second) qui serait traité. Pour éviter cet écueil, les paramètres sont "neutralisés" au moment de leur rangement dans la table en première lecture ; ils ne seront "déneutralisés", au cours de la compilation, que pendant le traitement de la procédure à laquelle ils appartiennent.

La "neutralisation" d'un paramètre formel s'effectue en ajoutant 30 aux deux positions de gauche de l'identificateur dans la table, de façon à ne plus le reconnaître pendant les consultations de tables. La "déneutralisation" est évidemment l'opération inverse. De par le système utilisé pour le traitement en mémoire des caractères alphanumériques, le procédé ne peut amener de confusion.

N. B. : Dorénavant, sauf précision contraire, on désignera par "table d'identificateurs" celle de la seconde lecture.

D. 4 - Construction de la table d'identificateurs

On procède de la façon suivante :

- à chaque entrée dans un bloc, il y a transfert vers cette table de la zone occupée dans la première table par les identificateurs déclarés dans ce bloc et déjà traités : aiguillages, identificateurs de procédures, étiquettes, etc ...

- lors de la rencontre d'un identificateur lorsque le sommet de la pile O est un déclarateur (autre que PROCEDURE ou AIGUILLAGE), il y a construction d'une nouvelle ligne pour cet identificateur, au moment des $\sum$(S(O)) et $\sum'$(S(O), ,). Si l'identificateurs est local à une procédure, il y aura construction d'une double ligne, dans la séquence PDCL (cf. chapitre 3).

D. 5 - Utilisation de la table

A l'entrée de chaque bloc, on met en réserve sur la pile V l'adresse du sommet de la table ; lorsqu'on sort d'un bloc, on transfère cette adresse de la pile V vers le compteur d'adresse CPTR2.

Cette disposition permet de libération des mémoires occupées dans la table par les identificateurs déclarés à l'intérieur du bloc dont on sort, et la "récupération" des places libérées quand on entre dans un nouveau bloc. Ainsi, la portée de la déclaration de tout identificateur est bien celle définie par le rapport Algol [1]. La consultation de table s'effectue en effet comme suit :

A la rencontre dans le programme-source d'un identificateur, on se branche sur le sous-programme CONSUL (si du moins S(0) n'est pas un déclarateur) ; la consultation se fait à l'aide du compteur CPTR3, initialisé par CPTR2, c'est-à-dire à partir du haut de la table. Un identificateur ne pourra donc pas être utilisé à l'extérieur du bloc où il a été déclaré.

A noter que les identificateurs des fonctions usuelles se trouvent placés immédiatement en dessous de la table, on peut donc considérer qu'ils ont été déclarés dans un bloc contenant le programme lui-même.

CHAPITRE II

LE PROBLEME DES PROCEDURES

RESOLUTION THEORIQUE

I - INTRODUCTION

La signification d'un appel de procédure est donnée aux paragraphes 3. 2. 3 et 4. 7 du rapport Algol [1] : l'effet produit par l'exécution de la procédure équivaut au remplacement de l'appel par le corps de procédure.

Dans le cas des procédures non récursives, et bien qu'étant la négation même de la notion de procédure, une telle copie est théoriquement possible.

Il en va tout autrement dans le cas des procédures récursives : la méthode de recopie ne peut être appliquée directement puisqu'on ne sait pas, au moment où le programme est compilé, combien de fois une procédure pourra s'appeler à l'intérieur d'elle-même.

Nous avons utilisé la méthode dite "de saut et retour"
- vers le corps de procédure associé à chaque appel de procédure,
- vers le paramètre effectif associé pour chaque occurrence de paramètre formel.

Avant d'entrer plus avant dans l'exposé du problème, nous allons voir, sur quelques exemples, que certaines notions, en particulier celle de copie, devaient être précisées.

A - Considérons le programme suivant :

```
DEBUT REEL A ;
   PROCEDURE F ; DEBUT REEL B ;
                       B := A↑2 + A*2 ;
                       A := A + B
                  FIN ; . . .
   A := 2 ; F ; ETI : . . .
FIN
```

La copie du corps de la procédure F au moment de son appel ne présente ici aucune difficulté. Le programme devient :

```
DEBUT REEL A ;
   PROCEDURE F ; DEBUT REEL B ;
                    B := A↑2 + A*2 ;
                    A := A + B
                 FIN ;
. . . A := 2 ;   DEBUT REEL B ;
                    B := A↑2 + A*2 ; A := A + B
                 FIN ; ETI : . . .
FIN
```

On voit aisément sur cet exemple simple que la recopie pouvait être évitée et remplacée par un saut au corps de procédure, à condition de connaître l'adresse de retour au programme, (ici l'adresse ETI).

On aura besoin pour cela d'un registre R conservant l'adresse de retour au programme une fois la procédure exécutée.

B - Modifions le programme de l'exemple précédent, de manière à ce que la procédure F devienne récursive.

```
DEBUT REEL A ;
   PROCEDURE F ; DEBUT REEL B ;
                    B := A↑2 + A*2 ;
                    A := A + 1 ;
                    SI B < 10 ALORS F ;
                    A := A + B
                 FIN ; . . .
A := A + 2 ; F ; ETI : . . .
FIN
```

Dans le cas présent, s'effectueront plusieurs appels de F, et leur nombre ne peut être connu à priori. A chacun de ces appels doit correspondre une adresse de sortie, et un index indispensable pour éviter les confusions. Adresse de retour et index sont liés à un appel, et ne doivent en aucun cas, à l'exécution, prendre la place de ceux correspondant à l'appel précédent. D'où l'idée d'une pile P permettant la gestion des registres R et I.

L'ordre d'entrée dans cette pile sera celui des appels de procédure rencontrés à l'exécution du programme, l'ordre de sortie sera celui des sorties de corps de procédures. C'est l'utilisation de la pile P qui permettra l'emploi de la méthode dite de saut et retour.

Il est possible, dès maintenant, d'avoir un aperçu sur le schéma de la compilation d'un appel et d'une déclaration de procédure.

Déclaration de procédure :

- Construction d'un ordre de saut inconditionnel vers l'instruction qui, dans le programme, suit immédiatement le corps de procédure. Le corps de procédure n'est, en effet, jamais exécuté au moment d'une déclaration, il ne l'est qu'après un appel.

Compilation du corps de procédure, avec formation d'instructions renfermant des adresses indexées, l'index utilisé se trouvant, au moment de l'exécution du programme, au sommet S(P) de la pile.

- Préparation du retour à une adresse se trouvant, elle aussi, en S(P), et ordre d'abaisser la pile P.

Appel de la procédure :

- Calcul de l'index actuel.
- Recherche de l'adresse de retour.
- Montée de la pile P et mise sur S(P) des renseignements précédemment calculés.
- Construction d'un ordre de saut vers la procédure.

C - Abordons maintenant le cas des procédures avec paramètres.

La pile P introduite plus haut est nécessaire, que les procédures aient ou non des paramètres ; dans le cas présent, nous verrons cependant que d'autres renseignements sont indispensables pour éviter toute confusion.

La détermination exacte de ce supplément d'informations sera faite au paragraphe 2, lorsque nous serons amenés à préciser les notions de copie d'un programme Algol, et de portée des déclarations des identificateurs.

Nous nous bornerons ici à approcher le problème par quelques remarques.

C1 - Analogie existant entre :

- appel de procédure et occurrence de paramètre formel appelé par nom d'une part ;

- déclaration de procédure et paramètre effectif d'autre part.

Soit le programme :

```
DEBUT REEL A , B ;
   PROCEDURE F(X,Y) ; REEL X , Y ; . . . ;
   F( ℰ1 , ℰ2) ; . . .
FIN
```

où $\mathcal{E}_1$ et $\mathcal{E}_2$ désignent des expressions arithmétiques.
X et Y étant appelés par nom, l'appel de procédure F($\mathcal{E}_1$, $\mathcal{E}_2$) est équivalent, d'après le rapport Algol à la séquence suivante :

```
DEBUT REEL PROCEDURE X ; X := ℰ1 ;
      REEL PROCEDURE Y ; Y := ℰ2 ;
  F(X,Y) ; . . .
FIN
```

Une occurrence de paramètre formel aura donc sur la pile P des répercussions identiques à celle d'un appel de procédure.

C2 - Difficultés d'associer un paramètre formel à un paramètre effectif :

Exemple 1 :

```
. . . PROCEDURE F(X) ; REEL X ;
          DEBUT PROCEDURE G(U) ; SI X = 2 ALORS . . . ;...
```

Le paramètre formel X, appartenant à la procédure F, est appelé dans le corps de la procédure G.

Un paramètre formel n'est donc pas toujours un paramètre de la dernière déclaration de procédure en cours de compilation.

Exemple 2 :

```
. . . ENTIER PROCEDURE F(N) ; F := SI N = 0 ALORS 1
                SINON F(N-1)*N ;
      . . . F(A) . . .
```

On remplacera F(A) par :

SI A = 0 ALORS 1 SINON F(A-1)*A ; soit

SI A = 0 ALORS 1 SINON (SI A-1 = 0 ALORS 1 SINON F(A-1-1)*(A-1)*A

soit encore :

SI A = 0 ALORS 1 SINON (SI A-1 = 0 ALORS 1 SINON (SI A-1-1 = 0 ALORS 1 SINON F(A-1-1-1) * (A-1-1)) * (A-1) * A ; etc...

On voit sur cet exemple que pour les diverses exécutions du même appel de procédure F(N-1), la même occurrence du paramètre formel N n'est pas toujours remplacée par la même expression.

L'association paramètre formel - paramètre effectif devra donc s'effectuer de façon dynamique lors de l'exécution.

On voit de plus que le paramètre effectif remplaçant N n'est pas nécessairement celui de la dernière procédure en cours d'exécution.

Exemple 3 :

Un appel de procédure doit être associé à une déclaration de procédure. Cette association n'est pas définie par Algol de façon stricte.

```
. . . PROCEDURE F(X) ; REEL X ;
          DEBUT REEL PROCEDURE G(Y) ; REEL Y ;
                   G := X * Y ;
                   SI X > 0 ALORS A := X SINON F(G(X)+1)
          FIN ;
      F(A+B) ; . . .
```

On supposera : A+B < 0 et G(A+B)+1 > 0, de manière à ce qu'on ait une exécution finie.

En remplaçant l'appel F(A+B) par le corps de la procédure F, nous obtenons, à la place de F(A+B) :

```
DEBUT   REEL PROCEDURE G(Y) ; REEL Y ;
        G := ( A + B ) * Y ;
  SI A + B > 0 ALORS A := A + B SINON F(G(A+B)+1)
FIN ;
```

Le nouvel appel F(G(A+B)+1) sera lui-même remplacé par :

```
DEBUT   REEL PROCEDURE G(Y) ; REEL Y ;
        G := ( G ( A + B ) + 1 ) * Y ;
  SI G(A+B)+1 > 0 ALORS A := G(A+B)+1 SINON F(G(G(A+B)+1)+1)
FIN
```

Il existe alors deux déclarations pour la procédure G (celles encadrées). A quelle déclaration de G doit-on associer l'appel souligné ? Seule la recopie de la première définition peut conduire à une exécution finie.

Les exemples précédents conduisent à penser que lorsqu'on effectue les copies comme l'indique [1], les règles sur les portées des déclarations (cf. paragraphe 5) doivent être modifiées.

Pour simplifier l'étude qui suivra, nous allons apporter quelques transformations à Algol.

DEFINITION D'ALGOL-P [9]

II. 1 - Règles à ajouter à la grammaire Algol

II. 1. 1 - Syntaxe

N. B. - On indique entre parenthèses les numéros des paragraphes du rapport Algol auxquels se rattachent ces extensions.

1) < Lettre > ::= ω (paragraphe 2. 1)

2) < Entête de bloc > ::= ENTREE | ENTREE < Déclaration > (paragraphe 4. 1)

3) < Indicateur de fonction > ::= < Variable > < bloc non étiqueté > (paragraphe 3. 2. 1)

4) < Type > ::= ETIQUETTE (paragraphe 5. 1. 1)

< Instruction d'affectation > ::= (paragraphe 4. 1. 1)
< liste de parties gauches > := < expression de désignation >

< Expression de désignation simple > ::= < indicateur de fonction > (paragraphe 3. 5. 1)

5) < Déclaration > ::= < déclaration d'équivalence >

< Déclaration d'équivalence > ::= EQUIVALENCE < liste de paires d'identificateurs >

< Liste de paires d'identificateurs > ::= < paire d'identificateurs >
< liste de paires d'identificateurs > , < paire d'identificateurs >

< Paire d'identificateurs > ::= < identificateurs > : < identificateurs >

II. 1. 2 - Sémantique

1) ω permettra d'associer à un identificateur J un autre identificateur ω J. On pourra de cette façon distinguer deux rôles différents joués par un même identificateur du programme Algol.

2) ENTREE joue le même rôle que DEBUT mais permettra de distinguer certains blocs. A signaler que dans les blocs commençant par ENTREE la partie déclaration peut être vide.

3) Pour calculer un indicateur de fonction du type décrit, V β, on exécute le bloc β ; l'indicateur de fonction a alors pour valeur celle de la variable V.

4) Une expression de désignation pourra être une fonction du type ETIQUETTE.

5) Une déclaration EQUIVALENCE :
A : B déclare A ; dans sa portée, A désigne le même élément que B ; la portée de B doit contenir cette déclaration d'EQUIVALENCE.

II. 2 - Restrictions d'Algol-P par rapport au langage de référence.

Certaines règles ont été supprimées :

1) < Déclaration de procédure > ::= < type > PROCEDURE < tête de procédure > < corps de procédure > (paragraphe 5. 4. 1)

< Partie gauche > ::= < identificateur de procédure > := (paragraphe 4. 2. 1).

On ne traitera donc pas les fonctions-procédures en tant que telles, mais on les ramènera à des instructions-procédures.

2) ⟨ Paramètre effectif ⟩ ::= ⟨ expression ⟩ (paragraphe 4. 7. 1)

Un paramètre effectif ne pourra être qu'un identificateur.

Notons que nous avons exclu d'Algol le traitement des chaînes.

3) ⟨ Partie valeur ⟩ ::= VALEUR ⟨ liste d'identificateurs ⟩ (paragraphe 5. 4. 1).

Pour tout paramètre appelé par valeur, on introduit un paramètre formel appelé par nom et un identificateur local à la procédure.

TRANSFORMATION D'UN PROGRAMME ALGOL EN ALGOL-P

III. 1 - Toute déclaration de procédure avec son type, comme

⟨ type ⟩ PROCEDURE F ... ; est remplacée par

⟨ type ⟩ ωF ; PROCEDURE F ... ;

et dans le corps de procédure :

F:= est remplacé par ωF:= ;

l'introduction de la variable ωF permet de ramener la fonction-procédure F à une instruction-procédure.

III. 2 - Pour toute procédure possédant des paramètres X1, X2, ... Xp appelés par valeur, on commence le corps de procédure par :

DEBUT ⟨ type 1 ⟩ X1ω ; ...

⋮

⟨ type p ⟩ Xpω ;

X1ω := X1 ; ... ; Xpω := Xp

FIN

où ⟨ type i ⟩ désigne le type entier, réel ou booléen de Xiω. Ainsi :

VALEUR X,Y,Z ; REEL X,Y ; ENTIER Z ;

sera transformé en :

DEBUT REEL Xω, Yω ; ENTIER Zω ;

Xω := X ; Yω := Y ; Zω := Z

FIN ;

La suppression de la partie valeur revient donc bien à une création de nouvelles variables, ce qui est conforme au rapport Algol.

III. 3 - On remplace tout identificateur de fonction ou instruction procédure F(λ) dont certains paramètres effectifs U1, U2, ... Up, (contenus dans la partie paramètre effectif λ) ne sont par réduits à des identificateurs par :

- ωF si F est indicateur de fonction, puis
- DEBUT ⟨ type 1 ⟩ ω ω1 ; PROCEDURE ω1 ; ω ω1 := U1 ; ...
  ⟨ type p ⟩ ω ωp ; PROCEDURE ωp ; ω ωp := Up ;
  F(λ')

FIN

où ⟨ type i ⟩ est le type ENTIER, REEL, BOOLEEN ou ETIQUETTE de Ui et où λ', partie paramètre formel, est obtenue à partir de λ en remplaçant chaque Ui par ω i.

Cette transformation du programme Algol permet de préciser la notion d'appel par nom d'un paramètre formel.

III. 4 - On remplace les déclarations d'aiguillages par des déclarations de procédures du type ETIQUETTE. Ainsi la déclaration :

AIGUILLAGE A := E1, E2 ; sera remplacée par

ETIQUETTE ωA ; PROCEDURE A(i) ; ENTIER i ;

SI i = 1 ALORS ω A := E1 SINON SI i = 2 ALORS ωA := E2 ;

Lorsque i aura une valeur extérieure à l'intervalle de définition (ici (1,2)), on ira bien en séquence ainsi que le prescrit le rapport Algol.

De plus, on remplace l'appel d'aiguillage A[N] par un appel de procédure ; A[N] devient :

DEBUT ENTIER ω 1 ; ω1 := N ; A(ω1) FIN.

Exemple de transformation d'un programme Algol en Algol-P : n !

Le programme Algol est le suivant :

DEBUT ENTIER A, B ;
ENTIER PROCEDURE F(N) ;
F := SI N = 0 ALORS 1 SINON F(N-1)*N ;
B := F(ℰ 1) ; ...
FIN (où ℰ 1 désigne une expression arithmétique).

Sa transformation en Algol_P s'écrit :

```
DEBUT ENTIER A, B, ωF ;
      PROCEDURE F(N) ;
        F := SI N = 0 ALORS 1 SINON ω F
          DEBUT ENTIER ωω 1 ;
                PROCEDURE ω1 ; ωω1 := N-1 ; F(ω1)
          FIN * N ;
        B := ωF ;
          DEBUT ENTIER ωω1₁ ; PROCEDURE ω1₁ ;
                ωω1₁ := ξ1 ; F(ω1₁)
          FIN ; ...
FIN
```

On a mis des indices à ω1 et ωω1 pour permettre de mieux suivre le processus de remplacement.

- COPIE D'UN PROGRAMME ALGOL-P

IV. 1 - Définitions

* Copie :

Une copie d'un programme Algol-P est un programme obtenu en remplaçant dans le programme Algol-P une instruction procédure ou un indicateur de fonction du type F(A1, A2, ... , Ap) où F a une déclaration de paramètres formels X1, X2, ... , Xp par :

- ωF si F est un indicateur de fonction, puis
- ENTREE EQUIVALENCE X1 : A1, ω X1 : ωA1, ... , Xp : Ap, ωXp : ωAp ;
  .....
  copie du corps de procédure de la déclaration de F
  FIN.

Dans le cas où la procédure est sans paramètres, le bloc commençant par ENTREE ne comportera pas de déclaration. Le symbole ENTREE devra cependant être suivi d'un point-virgule.

* Copie itérée :

On appelle ainsi tout programme obtenu à partir d'un programme Algol-P par copies successives.

- Programmes équivalents ;

Un programme Algol, son transformé en Algol-P et toute copie itérée de celui-ci, sont dits équivalents. (Intuitivement, cela signifie que tous ces programmes auront la même valeur sémantique).

- Exécution :

On peut définir une exécution comme un couple (programme-données), où "données" est une suite finie de nombres et de valeurs booléennes. Deux exécutions de programmes équivalents, avec les mêmes données, sont dits équivalents.

IV. 2 - Pour toute exécution finie d'un programme Algol-P, on peut trouver une exécution équivalente où aucun appel de procédure n'est "effectué". Il resterait à définir ce que sont les instructions "effectuées" dans une exécution ; nous nous contenterons ici d'une idée intuitive.

IV. 3 - Exemple de copie

Reprenons le programme Algol-P de la page précédente (n !) ; la première copie de ce programme s'écrira :

```
DEBUT ENTIER A,B, ωF ;
      PROCEDURE F(N) ;
      ωF := SI N=0 ALORS 1 SINON ωF DEBUT ENTIER ωω1 ; PROCEDURE ω1 ;
                                        ωω1 := N-1 ; F(ω1)
                                  FIN * N ;
      B := ωF DEBUT ENTIER ωω1₁ ; PROCEDURE ω1₁ ; ωω1₁ := ξ1 ;
                ┌──────────────────────────────────────────────┐
                │ ENTREE EQUIVALENCE N: ω1₁, ωN: ωω1₁ ;        │
                │       ωF := SI N=0 ALORS 1 SINON             │
                │         ωF DEBUT ENTIER ωω1 ; PROCEDURE ω1 ; │
                │                   ωω1 := N-1 ; F(ω1)         │
                │             FIN * N                          │
                │ FIN                                          │
                └──────────────────────────────────────────────┘
            FIN ; ...
FIN
```

On fera une seconde copie de ce programme en écrivant :

```
DEBUT ENTIER A,B,ωF ;
      PROCEDURE F(N) ;
       ωF:=SI N=0 ALORS 1 SINONωF DEBUT ENTIERωω1 ; PROCEDUREω1 ;
                                          ωω1:=N-1 ; F(ω1)
                                  FIN ≠N ;
       B:= ωF DEBUT ENTIER ω ω1₁ ; PROCEDURE ωω1₁ ; ωω1₁:= ε1 ;
                ENTREE EQUIVALENCE N:ω1₁ ; ωN:ωω1₁ ;
                     ωF:= SI ωN [ENTREE ; ωω1₁:= ε1 FIN] =0
                          ALORS 1 SINON
                      ω F DEBUT ENTIER ω ω1 ; PROCEDURE ω1 ;
                                ωω1 := N-1 ;
                        ENTREE EQUIVALENCE N:ω1 ; ωN:ωω1 ;
                             SI [N]= 0 ALORS 1 SINON
                              ωF DEBUT ENTIER ωω1 ; PROCEDURE ω1 ;
                                        ωω1 := N-1 ;
                                      F(ω1)
                                 FIN ≠N
                        FIN
                                 FIN ≠ωN [ENTREE ; ωω1₁:= ε1 FIN]
                FIN
          FIN ; ...
FIN
```

I - PROBLEME DE LA PORTEE DES DECLARATIONS [9]

V. 1 - Définitions

A toute occurrence d'un identificateur dans un programme Algol ou Algol-P, on doit savoir associer une déclaration.

Pour préciser la manière d'effectuer cette association, nous préciserons pour chaque déclaration sa portée d'appel et sa portée d'existence.

Ce sont deux parties du programme, telles que :

- pour qu'une occurrence d'un identificateur i soit associée à une déclaration D de i, il faut et il suffit qu'elle soit contenue dans la portée d'appel de D ;

- la portée d'existence d'une déclaration D soit une partie connexe du programme, contenant sa portée d'appel.

V. 2 - Cas d'Algol

V. 2. 1 - Définitions

On appellera partie propre d'un bloc B la suite des signes de B non contenus dans un bloc B' strictement contenu dans B.

La portée d'appel d'une déclaration est une réunion de parties propres de blocs qui peut être définie par la règle suivante :

Règle

> Soit un bloc B et B1 le plus petit bloc contenant strictement B (s'il existe). La partie propre de B est dans la portée d'appel des seules déclarations suivantes :
>
> - celles de B ;
> - celles d'un identificateur ne faisant pas l'objet d'une déclaration dans B, mais dont la partie propre de B1 est dans la portée d'appel.

On prendra comme définition pour la portée d'existence d'une déclaration non rémanente le plus petit bloc la contenant (on ne s'occupe pas ici des déclarations rémanentes).

A deux identificateurs ayant des portées de déclarations non disjointes, on doit affecter à la compilation deux mémoires différentes.

C'est cette affectation de mémoires distinctes à tous les identificateurs du programme ayant des portées de déclaration non disjoints qu'on appelle normalisation du programme (cf. [4]).

V. 2. 2 - Méthode de normalisation d'un programme Algol

On affectera à chaque déclaration d'un identificateur J un identificateur du type Mi où i est un entier (Mi sera dite traduction de J), de manière qu'à deux déclarations ayant des portées d'existence non disjointes soient associés deux identificateurs différents.

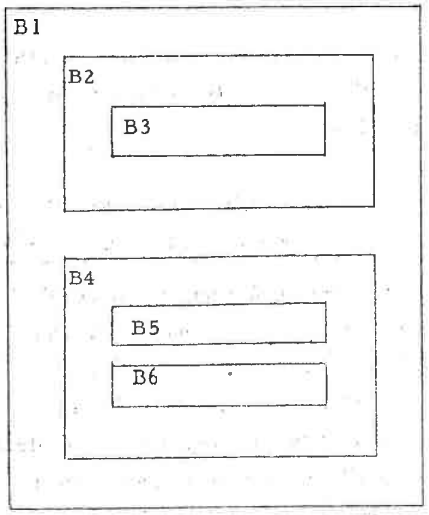
Pour cela, on procède de la manière suivante :

- On construit la pile attachée à l'arborescence des blocs pour la relation d'inclusion (cf. [2]).

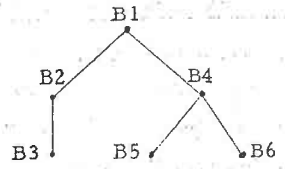
- Pour chaque bloc, on note ses déclarations avec la traduction proposée.

- A la rencontre d'un identificateur $\mathcal{J}$ dans le programme, on "remonte la pile", c'est-à-dire qu'on considère successivement tous les élèments de la pile à partir de son sommet, pour les comparer à $\mathcal{J}$. Ayant trouvé $\mathcal{J}$ dans la pile, on aura en même temps sa traduction.

Supposons pas exemple qu'un programme Algol ait la structure de bloc ci-dessous. A chaque bloc B_k correspond une déclaration D_k.



On peut construire l'arborescence des blocs :



La pile attachée sera :

		D3				D5		D6		
	D2	D2	D2		D4	D4	D4	D4	D4	
D1	D1	D1	D1	D1	D1	D1	D1	D1	D1	D1

V. 3 - Cas d'Algol-P

Remarque 1

La copie d'un programme normalisé n'est pas toujours normalisée. L'exemple de n ! (page II-12) en est une illustration.

La copie itérée comporte plusieurs occurrences de l'identificateur N qui ne désignent pas toujours la même quantité.

Remarque 2

Reprenons le même exemple, et considérons maintenant l'appel de N encadré. Si l'on conserve la même notion de portée d'appel qu'en Algol, on devrait, à la copie itérée suivante, réécrire N sous la forme :

ω N ENTREE ; ω ω1:=N-1 FIN

puis la nouvelle occurrence de N serait réécrite à nouveau

ωN ENTREE ; ω ω1:=N-1 FIN

ce qui conduirait à une suite infinie de réécritures.

On devra donc utiliser une autre notion de portée d'appel qu'en Algol.

V. 3. 1 - Partant d'un programme Algol, transformons-le, et normalisons le programme transformé (cette normalisation s'effectuant de la même manière qu'en Algol). Envisageons une copie itérée Π et appelons procédure réécrite tout bloc commençant par ENTREE. Un tel bloc a été construit grâce à une déclaration qu'on appellera sa déclaration.

Pour deux blocs B et B1 ($B \subset B1$) tels qu'aucune procédure réécrite p ne se trouve entre B et B1 ($\nexists p$ tel que $B \subset p \subset B1$), les déclarations de B et B1 portent sur des identificateurs différents du fait de la normalisation du programme Algol-P.

Aussi, pour définir les portées d'appel, on ne tiendra plus compte que des procédures réécrites. Au lieu d'utiliser les parties propres des blocs comme en Algol, on utilisera les parties propres des procédures réécrites. On appellera partie propre d'une procédure réécrite p la partie de p formée des signes non contenus dans une procédure réécrite p' strictement contenue dans p.

On arrive ainsi à la règle suivante :

Règle

Soit une procédure réécrite p et p_1 la plus petite procédure réécrite contenant strictement la déclaration de p. La partie propre de p est dans la portée des seules déclarations suivantes :

- celles de p, (c'est-à-dire des déclarations D telles que p soit la plus petite procédure réécrite contenant D),
- celles des identificateurs non déclarés par une déclaration de p, mais dont la partie propre de p_1 est dans la portée d'appel.

On prendra alors comme définition de la portée d'existence d'une déclaration la plus petite procédure réécrite contenant la portée d'appel.

Exemple :

Si l'on reprend l'exemple de n !, l'occurrence de l'identificateur N encadrée doit être réécrite :

N ENTREE ; ω ω1 := N-1 FIN et la nouvelle occurrence de N, soulignée ici, devra elle-même être réécrite en tenant compte des deux équivalences $N : \omega 1_1$ et $\omega N : \omega\omega 1_1$, et non plus des équivalences $N : \omega 1$ et $\omega N : \omega\omega 1$; dans ces conditions, on obtiendra comme copie de l'appel ci-dessus :

N ENTREE ; ω ω1 := ξ1 FIN

On parvient donc ainsi à trouver la valeur de N par une suite finie de réécritures, et la copie aura bien la même valeur sémantique que celle attribuée initialement au programme Algol. La réécriture terminale de N a donc été possible en prenant pour portée d'appel une définition liée aux procédures réécrites.

V. 2. 2 - Normalisation d'un programme Algol-P

Pour chaque procédure réécrite p, on devra donc chercher la plus petite procédure réécrite p_1 contenant la déclaration de p : on l'appelle LIEN de p.

On a toujours (cf. [2]) : $p \subset p1$.

Le graphe de la relation "p1 est le lien de p" dans l'ensemble des procédures réécrites est une arborescence ; en effet, le graphe est sans circuit du fait que $p \subset p1$ et tout élément autre que le programme a un lien et un seul.

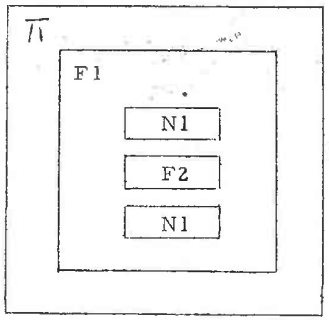
Mais alors que pour normaliser un programme Algol il suffisait de connaître pour tout bloc B le plus petit bloc B1 contenant B, ici p_1 n'est pas la plus petite procédure réécrite contenant p. Or, l'arborescence dont il est facile de construire la pile attachée est celle de la relation : "p1 est la plus petite procédure réécrite contenant strictement p", c'est-à-dire, comme pour les blocs, celles de la relation d'inclusion.

On construira donc cette pile, et pour chaque procédure, on notera dans la pile la position de son lien. Cette pile sera précisément la pile P introduite en début de chapitre.

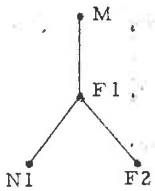
Au lieu de "remonter la pile" élément par élément en les prenant tous successivement, comme on l'avait fait pour normaliser un programme Algol (cf. p II-13), on la remontera grâce aux liens : on dit qu'on "régressera" dans la pile. On ira chercher pour toute procédure réécrite p, la procédure p_1 lien de la première, puis le lien de p_1, et ceci n fois si l'on doit "régresser de n".

Reprenons l'exemple de n !

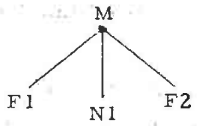
On peut retrouver sur la copie de la page II-12 la structure des procédures réécrites ; on peut schématiser cette structure de la façon suivante, les indices étant mis pour faciliter la compréhension :



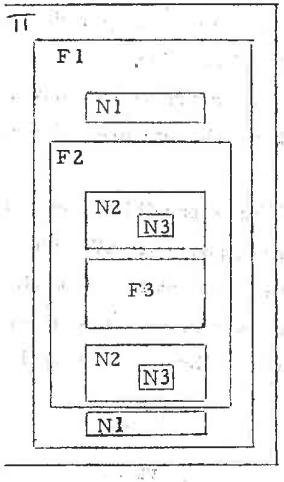
A cette structure, on peut associer l'arborescence des procédures réécrites pour la relation d'inclusion :
"p1 est la plus petite procédure réécrite contenant p" :

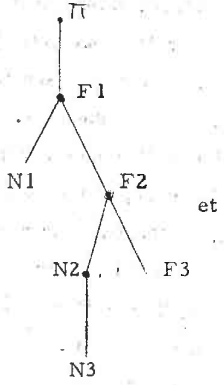


L'arborescence des liens était alors :

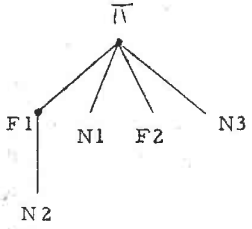


Lors d'une copie itérée suivante, la structure des procédures réécrites, et les arborescences précédentes, seront :





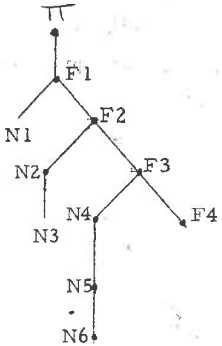
et

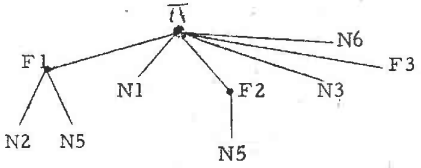


pour le lien

pour la relation d'inclusion

En continuant les recopies, les arborescences deviendront :





Copie et normalisation dynamique

Pour toute exécution d'un programme à procédures, on va se ramener par une suite de copies itérées (suite finie si le programme itéré a un sens) à un programme équivalent ne comportant aucune exécution de procédures.

En fait les copies successives ne s'effectuent pas, on procède par saut et retour.

Pour chaque exécution, il sera nécessaire de connaître l'adresse de retour et l'index qui permettra la normalisation du programme.

Il existe un double problème :

- pour chaque identificateur, retrouver la plus petite procédure réécrite contenant sa déclaration (donc son index),
- chaque fois qu'on entre dans une procédure réécrite, construire l'arborescence des liens (c'est-à-dire retrouver le lien de la procédure).

En réalité, on peut prévoir, dès la compilation, sur le programme Algol primitif, de combien l'on doit remonter dans l'arborescence des liens pour retrouver la plus petite procédure réécrite contenant la déclaration de la procédure en cours d'exécution.

Appelons niveau de déclaration dF d'une procédure F le nombre de déclarations de procédures contenant la déclaration de la procédure F.

Pour trouver la plus petite procédure réécrite contenant la déclaration d'un identificateur J, on devra régresser dans la pile P de n, avec

n = niveau dF - niveau dG,

dans le programme Algol-P, dF étant la plus petite déclaration de procédure contenant l'occurrence de J, et dG étant la plus petite déclaration de procédure contenant la déclaration de J.

Pour trouver le lien d'une exécution de procédure, on devra régresser de n dans la pile P, avec :

n = niveau dH - niveau dF + 1,

dans le programme Algol-P, dH désigne la plus petite déclaration de procédure contenant l'occurrence de la procédure K où l'on entre, et dG la plus petite déclaration de procédure contenant la déclaration de K.

On appellera par la suite le nombre n donné par la compilation "nombre de régression".

CHAPITRE III
TRAITEMENT DES PROCEDURES A LA COMPILATION

PRELIMINAIRES

A - On sait qu'aux identificateurs déclarés à l'extérieur d'une procédure, on attribue à la compilation une adresse, prise sur MRINT ou sur REMAN (cf. chapitre 1). Dans la zone indiquée par cette adresse sera rangée à l'exécution la valeur prise par l'identificateur (cf. [6] pour le cas des identificateurs de tableaux).

On attribuera de même une adresse de zone réservée pour l'exécution aux identificateurs particuliers aux procédures (identificateurs de procédure, identificateurs locaux et paramètres formels). Mais dans ce cas, cette adresse est variable en fonction de l'adresse de la première position libre sur MRINT au moment de l'appel de procédure. La compilation n'attribuera pas pour ces identificateurs des adresses vraies (< 59999), mais des adresses fictives qui devront être "dévirtualisées" à l'exécution par soustraction de l'index.

Ces adresses seront prises arbitrairement à partir de 90000, en décroissant. A l'entrée dans une déclaration de procédure ou à la rencontre d'un appel de procédure avec paramètre, on initialisera MRINT à 90000, (la valeur ancienne étant conservée sur la pile V), et on effectuera les réservations à partir de cette adresse. Comme on réserve 12 positions pour un identificateur de procédure et 17 pour un paramètre formel, l'adresse réservée pour un identificateur de procédure sera toujours 90000, les adresses des paramètres formels éventuels étant 89988, 89971, 89954 etc... Pour les identificateurs locaux, la réservation s'effectue normalement, à partir de la première position libre sur MRINT.

A la sortie du corps de procédure ou après traitement du dernier paramètre effectif, MRINT reprend la valeur mise en réserve sur la pile V.

L'adresse virtuelle affectée à un identificateur particulier à une procédure est rangée dans la table d'identificateurs (seconde ligne, position (12) indiquée sur le schéma du premier chapitre).

Lors du traitement d'un identificateur, l'adresse réservée correspondante sera mise dans le mémoire "ADRES".

B - Pour retrouver sur le programme Algol les niveaux des déclarations de procédure vus à la fin du chapitre précédent, notions définies sur Algol-P, on n'écrira évidemment pas le programme transformé.

On procédera comme suit :

Au cours de la première lecture, on notera, pour tout identificateur X "particulier" à une procédure, le niveau de sa déclaration ; on le placera dans la table, à la position "niveau de procédure", au moment de la déclaration de X.

Il suffit pour cela de disposer d'un compteur (initialisé à zéro), au contenu duquel on ajoute 1 à chaque entrée dans une déclaration de procédure et on retranche 1 à chaque sortie d'un corps de procédure.

A la compilation, un compteur (CTNIV) indiquera le niveau de déclaration de procédure qu'aurait en Algol-P l'appel en cours de traitement. On tiendra compte alors du fait que les appels de paramètres effectifs sont transformés en Algol-P en déclarations de procédure. On ajoutera donc 1 à CTNIV :

- à chaque entrée dans une déclaration de procédure ;
- à chaque rencontre d'une "parenthèse de procédure" (c'est-à-dire de toute parenthèse ouverte amenant un paramètre effectif).

On lui retranchera 1 :

- à chaque sortie d'un corps de procédure ;
- lors des $\sum$ et $\sum'$ de "parenthèses de procédures".

En reprenant les résultats du chapitre II, on aura donc pour le nombre de régressions

1) n = niveau dF - niveau dG = (CTNIV) - (niveau de procédure de J)

qui indiquera de combien l'on devra régresser dans la pile P à l'exécution pour obtenir l'index de l'identificateur J ;

2) n = niveau dH - niveau dF + 1 = (CTNIV) - (niveau de procédure de K) +1

qui permettra de trouver le lien de la procédure K dans laquelle on entre.

Notons qu'au moment d'un appel de paramètre formel N, on devra à la fois calculer l'index du calcul du paramètre effectif associé et trouver le lien de N traité comme procédure.

Le nombre de régression donné par la compilation est dans ce cas :

n = (CTNIV) - (niveau de procédure de N).

On trouvera d'abord l'index en régressant de n dans la pile P ; le lien sera celui qui se trouve immédiatement en dessous dans la pile P.

Le nombre de régression sera placé dans le compteur REGRES.

I - TRAITEMENT DES PROCEDURES LORS DE LA PREMIERE LECTURE

Au cours du premier passage du programme source, on sort de la logique générale à la rencontre du symbole PROCEDURE annonçant une déclaration, en se transférant au programme étiqueté LPROC ; la séquence LPROC a un double but :

- construire de nouvelles lignes dans la table pour les identificateurs de procédure et les paramètres formels, lignes où l'on rangera tous les renseignements qui peuvent être indiqués par les spécifications, ainsi que les niveaux de procédure.

- créer un bloc fictif "corps de procédure" (et ceci même si le corps de procédure est un bloc) ainsi que le préconise le rapport Algol [1] paragraphe 4. 7. 3. 1.

On utilise aussi deux zones de travail, TYPE et TPRIM, dans lesquelles sont préparés les indicatifs à transmettre dans la table aux positions T1 et T2 (cf. chapitre I).

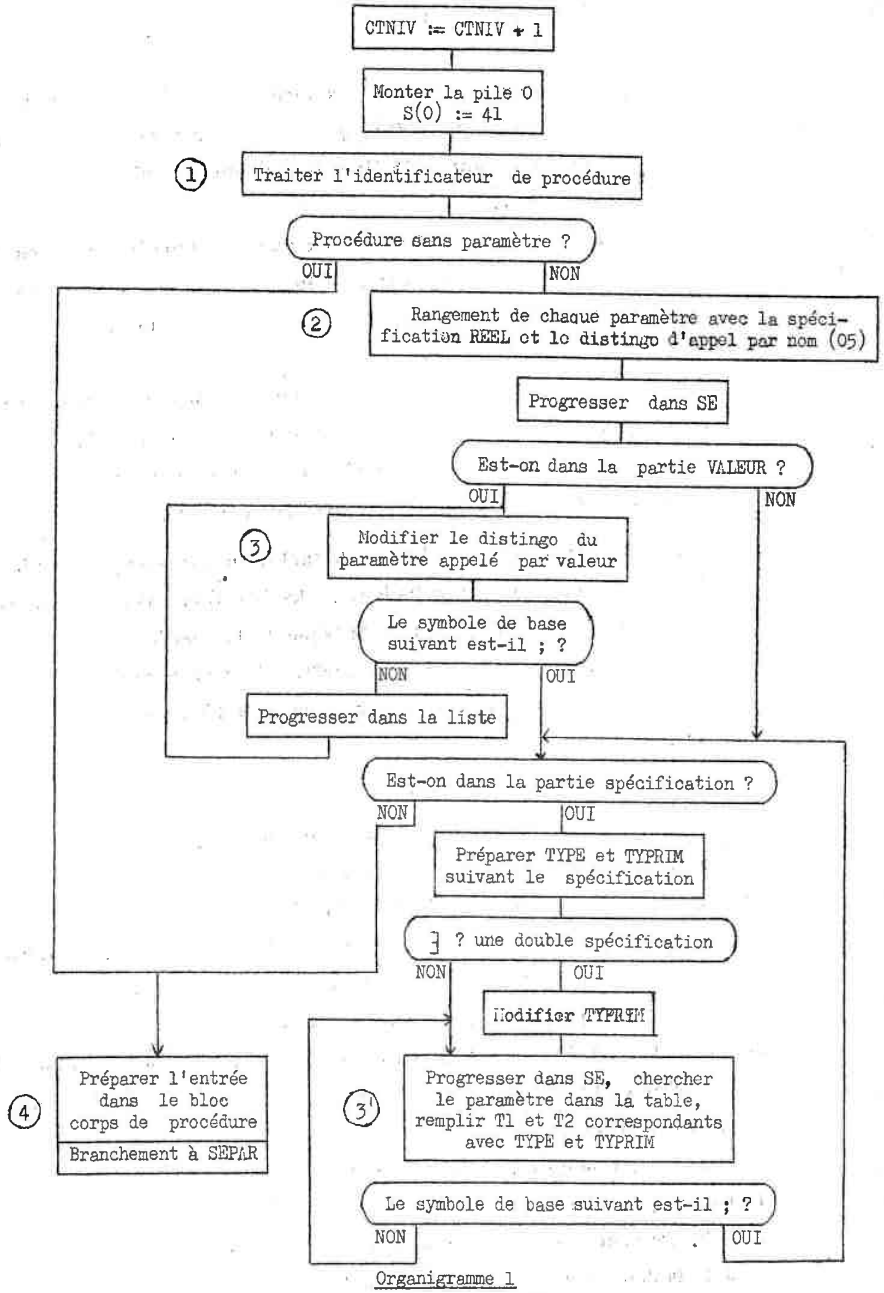
Développement de quelques points de l'organigramme très simplifié de LPROC (voir page suivante) :

1) l'identificateur de procédure est rangé dans la table, avec indicatifs de seconde ligne, distingo égal à $0\overline{1}$; on le considère à priori comme un identificateur d'instruction procédure (T1 = 9) ; on peut dès maintenant lui accorder une adresse pour l'exécution : ce sera l'adresse 90000, ainsi qu'on l'a vu plus haut. Testant ensuite le contenu de LSYMB, on détermine si le dernier symbole de base avant PROCEDURE était ENTIER, REEL ou BOOLEEN, et, dans l'affirmative, on place dans T1 le type correspondant.

Les seuls renseignements encore absents sont :

- l'adresse de début de compilation de la procédure, qui ne peut évidemment être connu qu'à la seconde ligne lecture,

- le nombre de paramètres de la procédure : il sera mis dans la table dès la rencontre du premier "point-virgule" suivant l'identificateur étudié.



Organigramme 1

TRAITEMENT de la TETE de PROCEDURE au COURS de la 1ère LECTURE

2) les paramètres formels sont rangés successivement dans la table, ils sont "neutralisés" (voir chapitre 1), et on leur attribue la spécification REEL (T1 = T2 = 0), le distingo 05 d'appel par nom et le niveau de procédure indiqué par CTNIV.

ADER étant une mémoire indiquant à chaque instant le nombre de paramètres déjà trouvés pour la procédure traitée, on attribue aux paramètres formels l'adresse d'exécution indexée suivante :

90 005 - 17 x (ADER).

La progression dans le programme ALGOL s'effectue jusqu'à la rencontre d'une parenthèse fermée suivie de "point-virgule". Les délimiteurs de paramètres sont considérés comme des commentaires, sauf en ce qui concerne leur listage par la machine à écrire.

3) notons que lors d'une consultation de table, on tient compte du fait que les paramètres sont neutralisés (cf. chapitre 1) en ajoutant également 30 aux positions de gauche de ATENT2 (zone de 14 positions où est formé l'identificateur). Cette consultation s'arrête à la rencontre de l'identificateur de procédure dans la table, ce qui permet la détection de certaines erreurs.

4) le début du corps de procédure est reconnu lorsqu'un point-virgule de la tête de procédure (rencontré par conséquent dans LPROC), n'est pas suivi par un déclarateur. Dans ce cas, on régresse dans la zone d'entrée du programme Algol jusqu'à ce point-virgule, et l'on se branche sur SEPAR, après avoir posé un "flag" sur la position SEPAR (cf. paragraphe 3)

La séquence SEPAR permet, après rencontre des symboles DEBUT ou FIN de placer dans la table d'identificateurs du premier passage une ligne de renseignements "début de bloc" ou "fin de bloc" ; la distinction entre DEBUT et FIN étant faite par un test sur la présence ou l'absence de "flag" sur SEPAR.

Le bloc fictif ainsi ouvert doit être fermé à la fin du corps de procédure par un branchement à SEPAR, ayant ôté le "flag" sur la position SEPAR. On rencontrera ce branchement dans la séquence LPTVG (cf. 3), lors de la détection d'un "point-virgule" dans la zone d'entrée, l'équivalent de procédure 41 étant S(0).

II - TRAITEMENT DES PROCEDURES A LA COMPILATION

Résumons les ordres engendrés à la compilation, en tenant compte de ce qui a été vu au chapitre précédent :

1) Lors d'une déclaration de procédure :

a) ordre de saut de la déclaration ;

b) compilation de la déclaration, avec construction d'ordres comportant des adresses indexées pour les identificateurs locaux et les résultats intermédiaires ; (pour les résultats intermédiaires, l'index à l'exécution se trouvera au sommet de la pile P, pour les identificateurs locaux, on devra créer à la rencontre dans le programme Algol des ordres de branchement à un sous-programme (VALOC) qui permettra la recherche à l'exécution dans la pile P, de l'index de l'exécution associée ;

c) ordres d'abaissement de la pile P à chaque sortie par ALLERA ;

d) à la fin du corps de procédure, ordres d'abaissement de la pile P et RETOUR.

2) Lors d'un appel de procédure :

a) ordre de saut à c) ;

b) compilation des paramètres effectifs, avec, à la fin de chacun d'eux, ordres d'abaissement de la pile P et RETOUR ;

c) ordre de mise sur la pile P de l'index, du lien, de l'adresse de retour, éventuellement de l'adresse de calcul du paramètre effectif ;

d) ordre de saut à la déclaration.

Les paramètres formels

Les occurrences de paramètres formels ont été considérées, nous l'avons vu au chapitre II, comme des appels de procédure. Néanmoins, certaines différences existent au niveau du programme-objet ; si, au moment de la compilation nous avons bien procédé comme il a été dit plus haut pour les procédures, il n'en est pas tout à fait de même pour les paramètres formels : on ne construit pas, en effet, dans ce cas, les

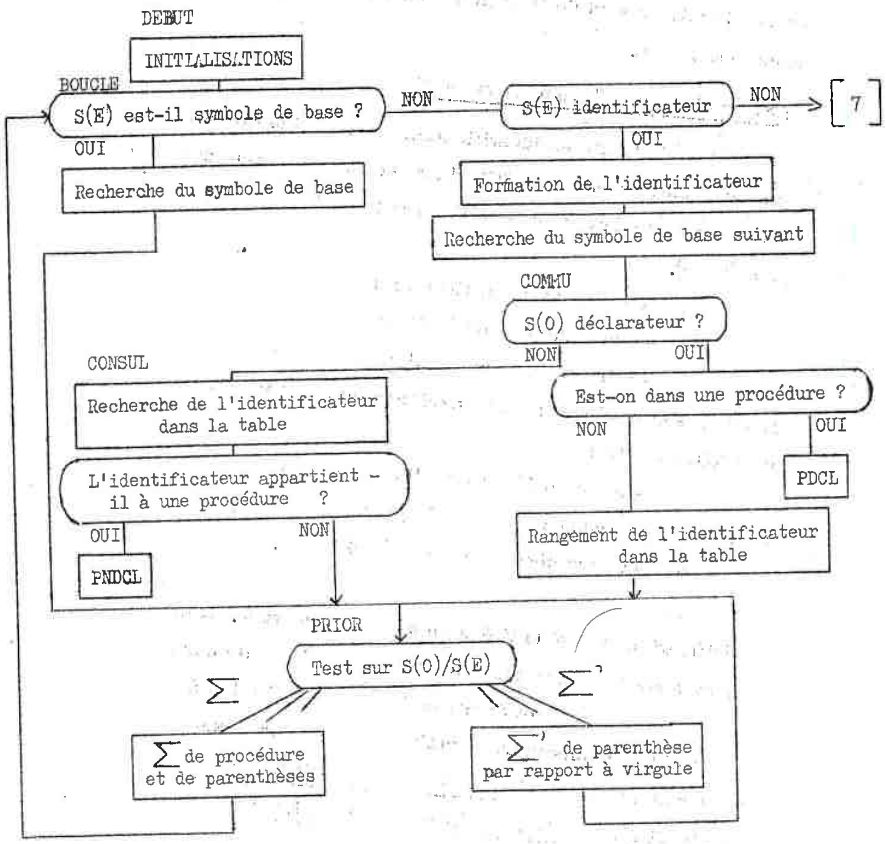
ordres de mise sur la pile P du lien, de l'index,etc... C'est un sous-programme d'exécution qui est chargé de trouver ce lien, l'index de l'exécution, l'adresse de retour, l'adresse du calcul du paramètre effectif associé. En réalité, à la compilation d'un appel de paramètre formel par nom, on se bornera à construire un ordre de branchement à ce sous-programme (PAFOR), en lui transmettant les renseignements nécessaires. (cf. chapitre IV).

En ce qui concerne les appels par valeur de paramètres Xi, on procédera comme suit :

1) Au moment d'entrer dans le corps de procédure dont Xi est paramètre formel, on construit un ordre de branchement au sous-programme PAFOR vu plus haut, qui mettra les renseignements voulus sur la pile P et placera dans les premières positions libres de MRINT le résultat du calcul du paramètre effectif associé. Un ordre de transfert de ce résultat vers l'adresse réservée au paramètre formel Xi sera alors exécuté.

2) Au moment de l'appel de Xi, il suffira de construire un ordre de branchement au sous-programme VALOC qui, ayant calculé l'index de l'exécution précédente, cherchera le résultat placé en 1) à l'adresse réservée et le tranférera sur MRINT, comme pour un identificateur local.

Pour mieux situer dans la logique générale du compilateur les séquences PDCL, PNDCL, $\sum$, $\sum'$ que nous allons étudier, nous donnons ici un organigramme (très incomplet bien sûr) de la deuxième lecture.



I - TRAITEMENT DE LA TETE DE PROCEDURE (cf. organigramme II)

A. 1 - Au moment de la rencontre de PROCEDURE

- addition de 1 à (CTNIV) ;
- saut à la séquence SDEB de [3] qui range dans la table d'identificateurs les déclarations du bloc fictif corps de procédure.

A. 2 - Lorsqu'on traite un identificateur de procédure ou un identificateur local, on sort de la logique générale du compilateur en se branchant sur PDCL.

Explications de l'organigramme II :

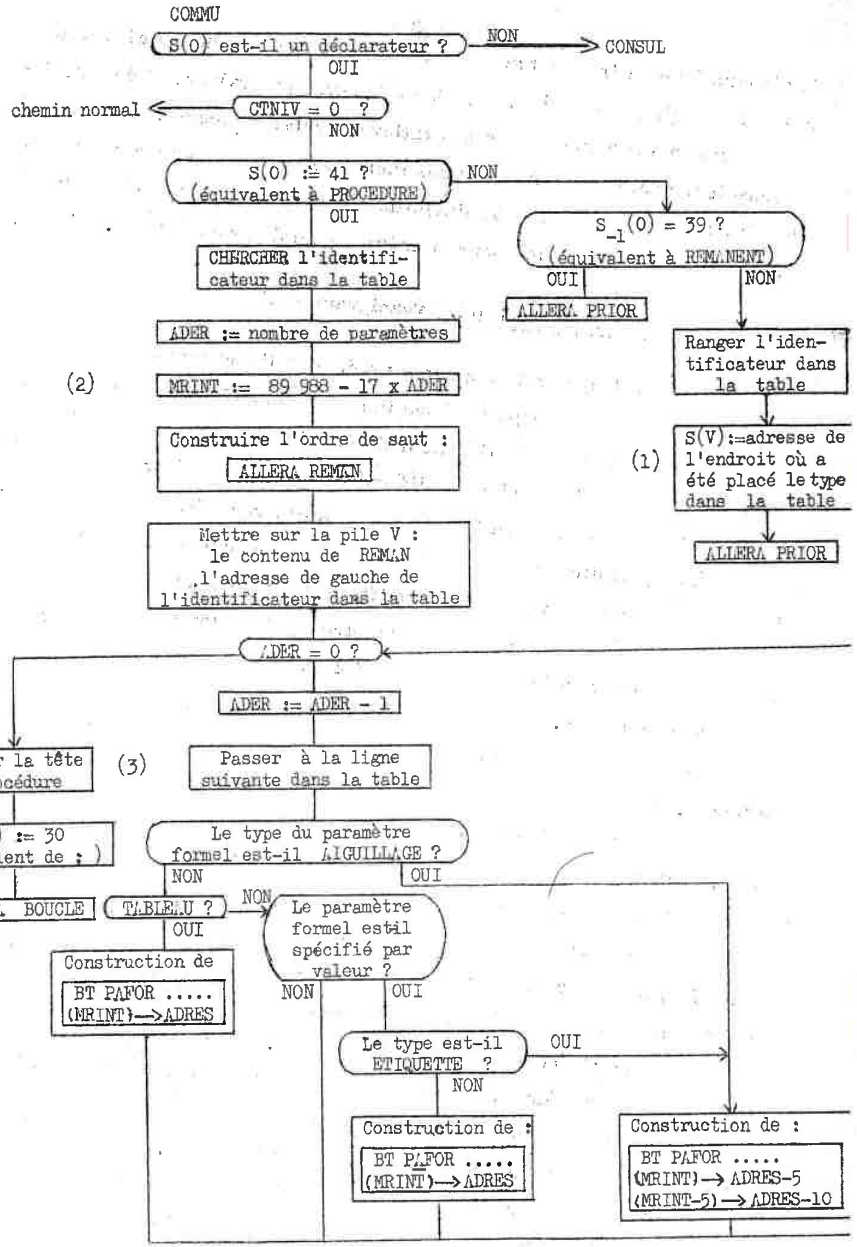
(1) On traite un identificateur local : l'identificateur est rangé dans la table, on place en S(V) l'adresse dans la table du type de l'identificateur (comme dans le cas où l'on traite un identificateur extérieur à une procédure).

(2) On traite un identificateur F de procédure :

- ordre de saut de la déclaration ;
- mise en réserve de l'adresse de F dans la table et de REMAN sur la pile V : ces renseignements seront retrouvés lors du de PROCEDURE ;
- mise en MRINT de l'adresse (virtuelle) de la première position libre, en tenant compte de la place prise par l'identificateur de procédure et les y paramètres formels (12 + 17 * y).

(3) La liste de paramètres formels, la partie valeur et la partie spécification ont été traitées en première lecture. Il importe donc de sauter dans le programme Algol jusqu'au début du corps de procédure.

Mais avant d'entrer dans la compilation de ce corps, il est nécessaire de construire certains ordres pour l'exécution, dans le cas de paramètres formels TABLEAU, AIGUILLAGE, ou de paramètres ETIQUETTE, REEL, ENTIER ou BOOLEEN appelés par valeur. On procédera comme il est indiqué plus haut par un ordre de branchement au sous-programme PAFOR, le résultat de calcul du paramètre effectif étant ensuite transféré dans l'adresse réservée au paramètre formel ;



Organigramme II

TRAITEMENT de la TETE de PROCEDURE et des IDENTIFICATEURS LOCAUX à la COMPILATION

la disparité des ordres de transfert suivant les cas est dûe en fait que le résultat du calcul est couble, (cas des étiquettes par valeur, et des aiguillages) qu'on a son adresse (cas des entiers, réels, booléens), ou directement le renseignement nécessaire (cas des tableaux). On peut dès maintenant chercher l'adresse du tableau ou de l'aiguillage, puisque le paramètre formel désignera toujours le même pour un même appel de procédure (cf. Algol 4.2.1).

APPELS DES PROCEDURES (cf. organigramme III)

Développement de quelques points de l'organigramme :

(1) On a vu au chapitre II que l'on avait supprimé la règle Algol : < partie gauche > ::= < identificateur de fonction > .

On considère bien ici l'identificateur de procédure comme un identificateur local, pouvant être du type entier, réel, booléen ou étiquette.

(2) On traite ici les identificateurs de procédures sans paramètres intervenant comme paramètres effectifs. Ayant vérifié que les types du paramètre formel et de la procédure paramètre effectif étaient en accord, du moins quand les vérifications sont possibles, on transforme dans la table la procédure en type.

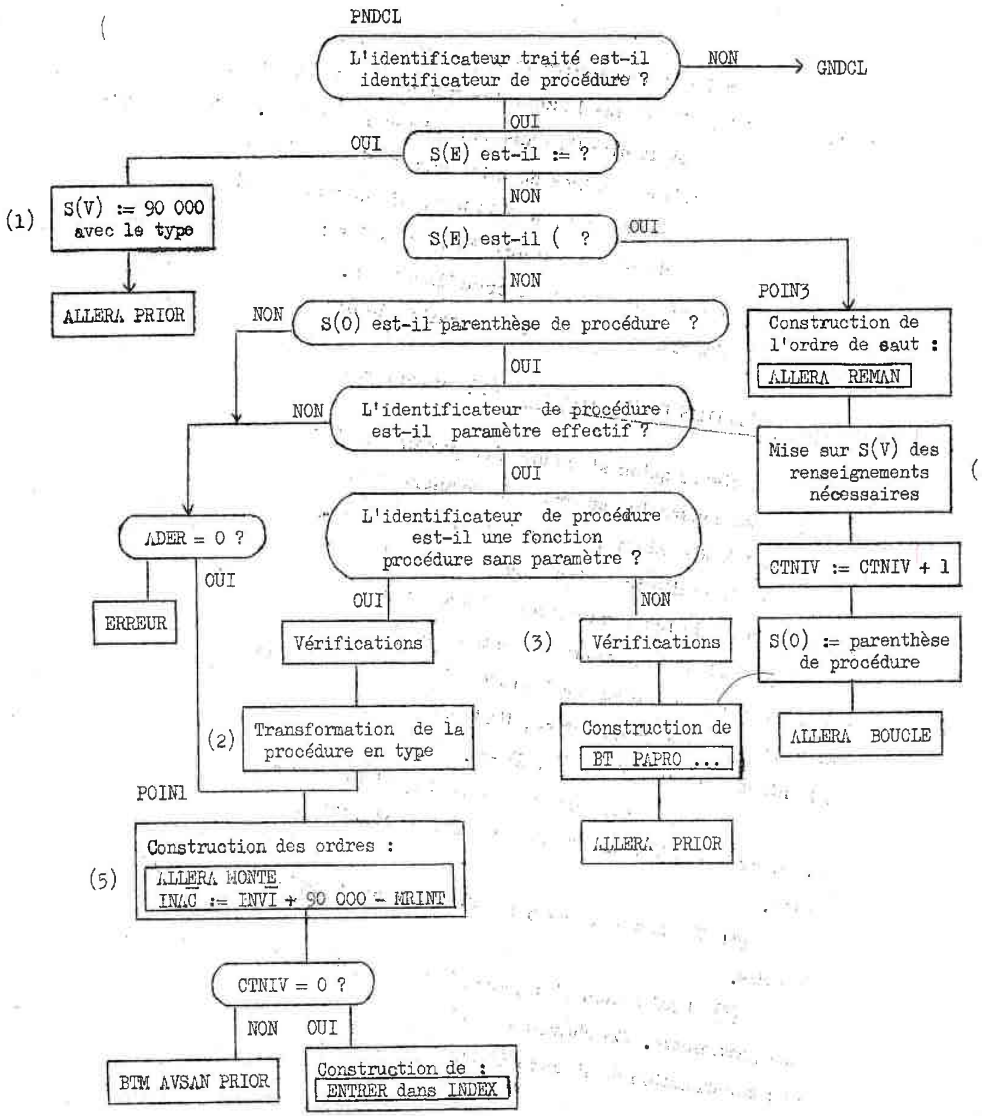
(3) Ici sont traités les identificateurs de procédures paramètres effectifs. Après les vérifications de type, on construit l'ordre de branchement au sous-programme PAPRO, qui cherchera l'index de l'exécution précédente, rangera l'adresse de calcul du paramètre effectif, et abaissera la pile.

(4) Rencontre d'un appel de procédure avec paramètres (POIN3).

On construit l'ordre de saut de la compilation des paramètres effectifs, puis on range sur la pile V un certain nombre de renseignements qui seront utilisés lors du Σ de parenthèse de procédure (adresse de la position T de l'identificateur dans la table (cf. chapitre I), nombre de paramètres de la procédure, contenu des compteurs REMAN et MRINT, adresse actuelle du programme objet). Avant d'entrer dans la compilation des paramètres effectifs, on affecte à (MRINT) l'adresse 90000.

(5) Traitement de l'appel de procédure proprement dit :

On construit les ordres de montée de la pile P, de calcul du nouvel index à partir de l'ancien, et l'on entre dans la séquence AVSAN.



Organigramme III

APPEL D'UN IDENTIFICATEUR DE PROCEDURE

Ce programme :

a) cherche si nécessaire grâce au programme PORTE de [6] la première position libre dans la zone les tableaux à l'exécution, et range cette adresse sur la pile P, dans la position TABLO (cf. chapitre IV).

b) construit les ordres de mise sur la pile de l'adresse de retour, d'entrée dans le programme REPRO (qui cherchera le lien de la procédure) en indiquant comme nombre de régression :

n = (CTNIV) - (niveau de procédure) + 1).

Construit enfin l'ordre de saut à la déclaration de procédure.

III - OCCURRENCE DES PARAMETRES (cf. organigramme IV)

Suivant qu'on s'occupe d'un paramètre formel appelé par nom ou pas, on va construire un ordre de branchement aux sous-programmes PAFOR ou VALOC ainsi qu'il est indique page III-3.

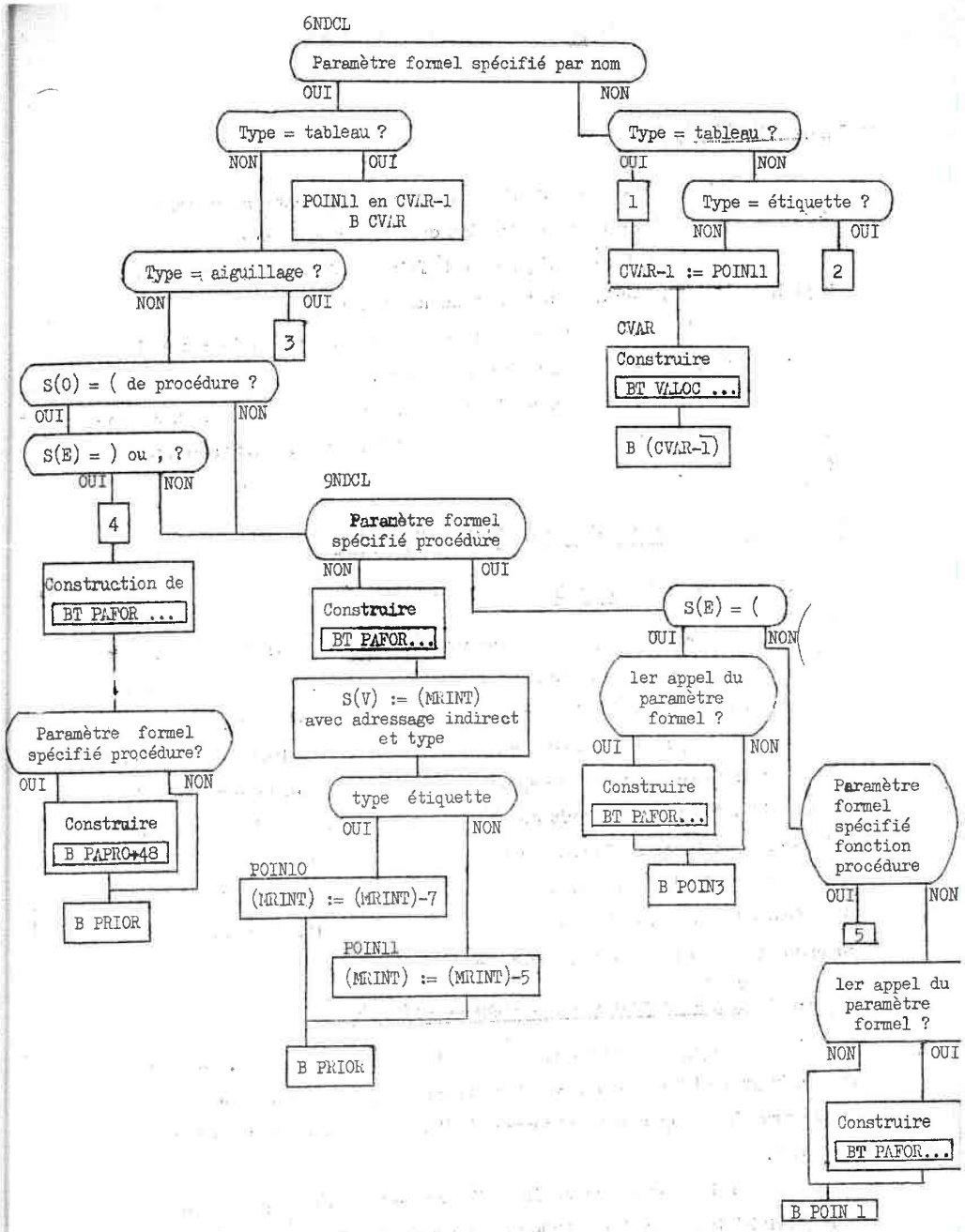
(1) On traite ici uniquement les variables locales tableaux, puisqu'on a exclu les paramètres tableaux appelés par valeur.

(2) (3) On augmente le contenu de REGRES de 50 avant de construire l'ordre de saut à VALOC ; il y a en effet dans ce cas deux renseignements à transmettre, l'adresse de l'aiguillage ou de l'étiquette et le niveau de procédure (cf. IV-5).

Le sous-programme VALOC, si le nombre de régression est supérieur à 50, enverra en MRINT et MRINT-5 ces deux renseignements.

(4) Traitement des paramètres formels pris comme paramètres effectifs.

(5) Traitement d'un paramètre formel spécifié fonction procédure sans paramètre. On charge sa spécification en type et on le traite comme tel : construction de l'ordre de saut à PAFOR, puis branchement à POIN11.



Organigramme IV

TRAITEMENT des APPELS des PARAMETRES FORMELS et des IDENTIFICATEURS LOCAUX

IV - $\sum$ DE PROCEDURE

On retrouve sur la pile V les renseignements qui y avaient été placés au moment de la déclaration de cette procédure :

- REMAN : ce qui permet de faire perforer l'adresse vers laquelle on doit sauter au début de la déclaration ;

- l'adresse de la procédure dans la table : ce qui permet de neutraliser à nouveau ses paramètres formels.

On construit l'ordre d'abaisser la pile BT ABSOR, on retranche 1 à CTNIV, et l'on saute à la séquence SFIN1 de [3] qui traitera la fin du bloc fictif.

V - FIN DE TRAITEMENT D'UN PARAMETRE EFFECTIF

V. 1 - La séquence FINEF

C'est la logique générale du compilateur qui traite les expressions pouvant se trouver comme paramètres effectifs. La séquence FINEF se chargera de construire les ordres de transfert du (ou des) résultat(s) de ce calcul, de la première position libre de MRINT vers l'adresse réservée du paramètre formel correspondant ; selon que le paramètre effectif sera un identificateur de variable simple, de tableau ou d'aiguillage, une étiquette ou une expression, le traitement pourra différer légèrement.

A la fin de FINEF, on construira dans chaque cas l'ordre d'abaisser la pile P, et l'on se branchera à (FINEF - 1), c'est à dire aux programmes SUFI6 ou SUFI7 (voir ci-dessous).

V. 2 - $\sum'$ de parenthèse par rapport à virgule

Ce $\sum'$ existe dans [3] sous la référence PARVIR. Dans le cas où la parenthèse ouverte se trouvant en S(0) est une parenthèse de procédure, il y a branchement vers FINEF, avec SUFI6 comme adresse de sortie.

SUFI6 place sur la pile V l'adresse actuelle du programme objet, réinitialise MRINT à 90000 pour traiter le paramètre suivant et saute à BOUCLE (cf. [3]).

V. 3 - Σ' de parenthèse

C'est la référence PARENT de [3]. Si la parenthèse est une parenthèse de procédure, on se branche à FINEF, avec SUFI6 ou SUFI7 comme adresse de sortie, suivant que la parenthèse fermée indiquée par S(E) est, ou non, une parenthèse de délimiteur de paramètre. Dans le cas du traitement du dernier paramètre d'un appel, on exécute donc la séquence SUFI7.

La séquence SUFI7

- vérifie que le nombre de paramètres de l'appel est bien le même qu'à la déclaration de procédure ;

- affecte à MRINT la valeur qu'avait ce compteur avant l'appel ;

- retranche 1 au contenu de CTNIV ;

- construit les ordres de montée de pile, de calcul de l'index nouveau par rapport à l'ancien ;

- construit pour chaque paramètre formel de F l'ordre de transfert dans l'adresse qui est réservée à l'exécution de l'adresse de calcul du paramètre effectif correspondant dans le programme objet ;

- fait perforer l'adresse du programme où l'on doit sauter au moment de l'appel de procédure pour éviter d'exécuter immédiatement les ordres qu'on a construits lors de la compilation des paramètres effectifs ;

- saute au programme AVSAN vu page III-13 pour traiter l'appel

CHAPITRE IV

TRAITEMENT DES PROCEDURES A L'EXECUTION

INTRODUCTION

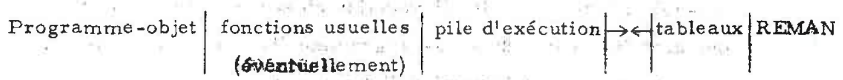
Le programme généré à la compilation des procédures possède une caractéristique : les ordres qui doivent être exécutés comportant des adresses supérieures à 60 000.

Il est donc nécessaire de disposer d'un programme de traduction qui, traitant chaque instruction selon son type, soit susceptible de dévirtualiser éventuellement les adresses en leur soustrayant la valeur de l'index, et bien sûr, d'une zone réservée pour la pile d'exécution qui doit, entre autres choses, permettre précisément de retrouver cet index.

A ce programme d'indexage (INDEX) et à cette pile, viennent s'ajouter les différents sous-programmes auxquels se réfère le programme-objet.

I - LA PILE D'EXECUTION

Cette pile se développe en mémoire derrière le programme-objet si celui-ci n'appelle pas de fonctions usuelles, sinon immédiatement après celles-ci, de sorte qu'on a la disposition suivante :



Chaque niveau de la pile est constitué par quatre compteurs de cinqpositions qui indiquent respectivement :

- le lien
- l'index
- l'adresse de retour
- l'adresse de calcul d'un paramètre effectif.

Pour des commodités de programmation, la pile est gérée par sept registres qui désignent eux-mêmes les adresses des registres des niveaux actuels, précédents ou futurs de la pile. Ces sept compteurs sont appelés :

LIAC, LIVI, INAC, INVI, RETAC, REFU, et TABLO.

Lorsque la pile est à un niveau i, on peut schématiser ainsi les positions désignées par les compteurs :

	lien	index	ad. retour	ad. P.F.
niveau futur			REFU	
niveau actuel i	LIAC	INAC	RETAC	TABLO
niveau précédent	LIVI	INVI		

"Monter" la pile revient à ajouter 20 à chacun de ces 7 compteurs, et réciproquement, "baisser" la pile revient à leur retrancher 20.

II - LE SOUS-PROGRAMME INDEX

Rappel

Une instruction 1620 occupe en mémoire 12 positions se décomposant en trois parties :

- code-opération de 2 positions, $0_0\ 0_1$
- adresse P de 5 positions, $P_2\ P_3\ P_4\ P_5\ P_6$
- adresse Q de 5 positions, $Q_7\ Q_8\ Q_9\ Q_{10}\ Q_{11}$.

Le programme INDEX traite les instructions du programme-objet, lorsque la pile d'exécution n'est pas au niveau zéro, ainsi que certains ordres du sous-programme d'exécution PAFOR (cf. page IV-3). Il se compose de trois parties distinctes :

1°) Gestion des instructions du programme à "dévirtualiser" : contrôle de la progression le long de ce programme, instruction par instruction, et suivant son type, branchement à un sous-programme de traitement correspondant. Cet aiguillage s'effectue grâce à une zone de 100 positions (des mémoires 1600 à 1699) ; selon le code-opération de l'ordre traité $0_0\ 0_1$, il y aura branchement vers le sous-programme dont l'adresse est en $0\ 1\ 6\ 0_0\ 9$.

Selon le code-opération de l'instruction, un ou les deux opérandes seront envoyées vers le sous-programme TRAD de "dévirtualisation" d'adresse. Dans le cas des ordres de sauts, le traitement sera plus particulier.

N. B. - l'adresse de l'instruction traitée est en index-1, l'instruction elle-même se trouvant en AGAIN-12.

2°) Les sous-programmes de traitement correspondant au type de l'instruction à dévirtualiser :

a) $0_0 = 0$ ou $0_0 = 2$: IND1

IND1 envoie les deux opérandes au sous-programme TRAD ; les ordres traités ici sont :

A, S, M, D, opérations arithmétiques
TD, TF, opérations de transmission interne
C, BT, opérations logiques, ainsi que les ordres correspondant à des opérations en virgule flottante.

b) $0_0 = 1$ et $0_0 = 3$: IND2

IND2 envoie seulement le premier opérande vers TRAD ; en effet, si 0_0 est 1, on a affaire aux instructions immédiates et le second opérande doit être transmis intact, si 0_0 est 3, les seuls ordres qu'on est susceptible de rencontrer à l'intérieur d'INDEX sont les instructions 32 et 33 (SF et CF), de pose et retrait de FLAG.

c) $0_0 = 7$: le cas ne peut se rencontrer dans INDEX.

d) $0_0 = 4$: INDEX3

Ce cas correspondant aux instructions de branchement, il a nécessité un nouvel aiguillage ; selon le code-opération de l'ordre traité, on se branche vers le sous-programme dont l'adresse est en $0\ 1\ 6\ 0_1\ 4$.

d1) $0_1 = 1$: AGAIN - 24

Pour exécuter l'ordre 41 (NOP), il suffit de passer au traitement par index de l'instruction suivante du programme-objet.

N. B. - Cet ordre se rencontre lors de l'exécution d'une instruction POUR.

d2) $0_1 = 2$: le cas ne peut se présenter à l'intérieur de INDEX.

d3) $0_1 = 3$ ou $0_1 = 4$: IND5

Les deux opérandes des ordres 43 (BD) et 44 (BNF) doivent être traités par TRAD, mais on doit conserver deux adresses possibles de branchement, l'adresse qui se trouve en AGAIN-6 et l'adresse en séquence du programme-objet.

d4) $0_1 = 5$: le cas ne peut se présenter à l'intérieur d'INDEX.

d5) $0_1 = 6$ ou $0_1 = 7$: IND6

Les ordres 46 (BI) et 47 (BNI) nécessitent des précautions spéciales ; ces ordres testent des indicateurs qui risquent d'être modifiés entre l'instant où a été exécutée l'instruction précédente, dont on veut tester le résultat, et celui où l'on effectue le test 46 ou 47.

Avant d'effectuer aucune opération arithmétique ou logique susceptible de modifier les indicateurs, on met en réserve leur contenu dans une zone de travail, et l'on s'arrange pour que le test 46 soit un test sur un digit, et 47 sur un flag. On est alors ramené au cas d3.

d6) $0_1 = 8$: ne se présente pas à l'intérieur du programme INDEX.

d7) $0_1 = 9$: IND7

C'est le cas de l'ordre 49 de branchement inconditionnel à l'adresse P de l'instruction ; si la partie Q d'une telle instruction n'est pas prise en considération par la machine, il n'en est pas de même pour le programme INDEX. Deux possibilités peuvent se présenter :

i) Pour gagner des positions en mémoire, l'instruction de branchement n'occupe que huit positions (la partie Q de l'ordre étant réduite à Q7). Il s'agit toujours alors d'un branchement vers une adresse de la même procédure, ne nécessitant pas une modification du niveau de la pile d'exécution. Le branchement s'effectue normalement vers l'adresse P éventuellement dévirtualisée, et ce, sans quitter le programme INDEX.

ii) Dès qu'il y a possibilité de changer de niveau de procédure en exécutant le branchement, la partie Q de l'instruction comporte un renseignement qui permet de retrouver le lien entre les deux exécutions en cours :

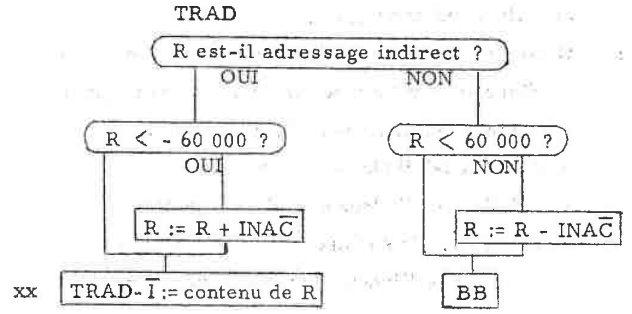
- ou bien l'on a affaire à une étiquette et la différence entre les niveaux de procédure a pu se faire à la compilation ; le nombre de régression se trouve alors en Q_{10} et Q_{11}, avec flag sur Q_{10}. On utilise le sous-programme REGRE (cf. page IV-6) pour régresser dans la pile de la hauteur convenable, et le branchement s'effectue, avec possibilité de sortir de INDEX si le nouveau niveau est nul.

- ou bien l'on a affaire à une expression de désignation et cette différence ne peut se faire qu'au moment de l'exécution ; dans ce cas la partie Q de l'ordre traité indique l'adresse dans MRINT où cette différence est effectuée. On traite donc l'adresse Q par TRAD et, une fois trouvé le nombre de régression, on est ramené au problème précédent. Après avoir effectué le branchement, on sort du programme d'indexage si l'étiquette vers laquelle s'effectue le transfert est au niveau de procédure ZERO.

3°) Le sous programme TRAD :

Si R désigne l'adresse à dévirtualiser, on a en TRAD-1, au moment où l'on entre dans le sous-programme, l'adresse de l'adresse de R : TRAD-$\overline{1}$ R.

Organigramme :



xx

xx Ce transfert nécessite un certain nombre de précautions qui visent à "couper" la chaîne d'adressages indirects qui pourraient conduire la machine à rencontrer une adresse non encore dévirtualisée.

III - LES SOUS-PROGRAMMES D'EXECUTION

On dispose de trois sous-programmes pour la gestion externe de la pile :

MONTE : permet de monter la pile d'exécution,
teste la rencontre possible de la pile avec les tableaux qui lui font face,
fait entrer dans le programme d'indexage.

ABRET : permet de baisser la pile,
En sortant de ABRET, on entre à nouveau dans INDEX, car la pile est toujours au niveau 2 lorsqu'on y entre.

ABSOR : permet de baisser la pile, et d'entrer à nouveau dans INDEX si l'on ne se retrouve pas au niveau zéro.

Deux sous-programmes permettent de trouver le lien : REPRO et REGRE.

REPRO : Ce programme nécessite la connaissance du nombre de régression ; après son exécution le lien cherché se trouve effectivement dans $\overline{LIAC}$.

REGRE : C'est ce programme qui permet de régresser dans la pile.
On dispose en y entrant des renseignements suivants :
- en REGRE-1 le nombre de régression,
- en REGRE-3 l'adresse de sortie,
- en REGRE-8 l'adresse du lien à partir duquel on veut régresser. En sortant de ce sous-programme, REGRE-8 l'adresse de l'index correspondant à l'exécution liée à celle en cours.

Le sous-programme de traitement du paramètre formel appelé par nom : PAFOR.

PAFOR : permet de ranger dans la pile l'adresse du résultat du calcul d'un paramètre formel appelé par nom et de se brancher à l'adresse de ce calcul dans le programme-objet. En entrant dans PAFOR, on dispose des renseignements suivants, fournis par la compilation :

- en PAFOR-1 le nombre de régression,
- en PAFOR-3 l'adresse réservée du paramètre traité,
- en PAFOR-8 90 000-MRINT,
- en PAFOR-13 l'adresse de retour.

Ce sous-programme monte la pile d'exécution grâce à MONTE, et place les renseignements suivants :

- en $\overline{RETAC}$: l'adresse de retour qui se trouvait en PAFOR-13,
- en $\overline{INAC}$: $\overline{INVI}$ + 90 000 - MRINT,
- en $\overline{LIAC}$: le lien de l'exécution de procédure à laquelle appartient le paramètre,
- en $\overline{TABLO}$: l'adresse sur MRINT où se trouvera le résultat du calcul du paramètre effectif ; cette adresse est calculée par la nouvelle valeur de l'index. La recherche de l'adresse de calcul du paramètre effectif dans le programme-objet se fait à l'aide de l'index correspondant à l'exécution précédente.

On sort de PAFOR par un transfert vers cette dernière adresse par l'intermédiaire de INDEX.

Le sous-programme de traitement des paramètres formels appelés par valeur et des variables locales : VALOC.

VALOC : cherche l'index de l'exécution en cours et transfère vers la zone de MRINT réservée à l'exécution en cours les renseignements correspondant à ce paramètre dans l'exécution précédente.

A l'entrée dans le sous-programme, on dispose :

- en VALOC-1 : du nombre de régression ou de celui-ci augmenté de 50 (cas des étiquettes et des aiguillages) ou de l'opposé de celui-ci (cas des tableaux),
- en VALOC-3 : de l'adresse réservée du paramètre ou de la variable,

- en VALOC-8 : de la valeur de MRINT,
- en VALOC-13 : de l'adresse de sortie du sous-programme.

Dans le cas général, c'est l'adresse de la variable qu'on doit transférer vers la zone de MRINT réservée à la nouvelle exécution.

Dans le cas des variables indicées, on a dû prendre certaines précautions pour éviter de transférer l'adresse de cette adresse au lieu de cette adresse elle-même : d'où la présence d'un flag en VALOC-1 pour différencier le cas.

Dans le cas des étiquettes et des aiguillages, ce sont deux renseignements et non plus un seul qui doivent être transmis vers MRINT : outre l'adresse vue plus haut, on a besoin de connaître à l'exécution le niveau de procédure de l'étiquette ou de l'aiguillage traité ; on reconnaîtra ce cas par le fait que

VALOC - 1 > 50.

On sort de VALOC par un branchement vers l'adresse qui se trouve en VALOC-13, en entrant à nouveau dans INDEX.

NOTICE D'UTILISATION DU COMPILATEUR ALGOL
POUR I. B. M. 1620

GENERALITES

Le compilateur a été écrit pour un ordinateur IBM 1620, 60 K, muni des dispositifs spéciaux :

- virgule flottante automatique,
- adressage indirect,
- dernière carte.

Le langage source est ALGOL avec quelques restrictions qui seront indiquées plus loin. Le langage objet est le langage machine 1620.

Le compilateur occupe environ 49500 positions dans sa forme complète, 48000 si l'on utilise la version sans tableaux rémanents, le reste de la mémoire étant employé à bâtir une table d'identificateurs.

Les organes d'entrée-sortie utilisés sont la machine à écrire et le lecteur-perforateur de cartes.

REPRESENTATION DES SYMBOLES DE BASE ALGOL

Pour faciliter les tests d'identification à la compilation, il a été convenu de représenter les symboles de base de trois manières :

- ou bien un caractère spécial 1620,
- ou bien un point suivi d'un caractère spécial,
- ou bien une suite de lettres encadrées par deux points ,

ces suites de lettres désignant des mots français ou anglais.

ALGOL	a	b	...	z	A	B	...	Z	0	1	2	...	9
COMPILATEUR	A	B	...	Z	A	B	...	Z	0	1	2	...	9

ALGOL	,	;	:	(	)	+	-	x	/	[	]	10	‹	›
COMPILATEUR	,	.,	..	(	)	+	-	x	/	.(	.)	.Z	@	$

ALGOL	VRAI	FAUX	≡	⊃	∪	∩	¬
C. Vers. Français	.VRAI.	.FAUX.	.EQUI.	.IMP.	.OU.	.ET.	.NON.
C. Vers. Anglais	.TRUE.	.FALSE.	.EQUI.	.IMPLI.	.OR.	.AND.	.NOT.

ALGOL	:=	÷	↑	<	≤	=	≥	>	≠
C. Vers. Français	.=	.DIVENT.	$	.INF.	.ING.	=	.SUG.	.SUP.	.DIF.
C. Vers. Anglais	.=	.ID.	$	.LT.	.LET.	=	.GET.	.GT.	.DIF.

ALGOL	ALLERA	SI	ALORS	SINON	POUR	FAIRE	PAS
C. Vers. Français	.ALLERA.	.SI.	.ALORS.	.SINON.	.POUR.	.FAIRE.	.PAS.
C. Vers. Anglais	.GOTO.	.IF.	.THEN.	.ELSE.	.FOR.	.DO.	.STEP.

ALGOL	JUSQUA	TANTQUE	REMANENT	BOOLEEN	ENTIER	REEL
C. Vers. Français	.JUSQUA.	.TANTQUE.	.REMANENT.	.BOOLEEN.	.ENTIER.	.REEL.
C. Vers. Anglais	.UNTIL.	.WHILE.	.OWN.	.BOOLEAN.	.INTEGER.	.REAL.

ALGOL	TABLEAU	AIGUILLAGE	PROCEDURE	CHAINE	ETIQUETTE
C. Vers. Français	.TABLEAU.	.AIGUILLAGE.	.PROCEDURE.	.CHAINE.	.ETIQUETTE.
C. Vers. Anglais	.ARRAY.	.SWITCH.	.PROCEDURE.	.STRING.	.LABEL.

ALGOL	VALEUR	COMMENTAIRE	␣	DEBUT	FIN
C. Vers. Français	.VALEUR.	.COMMENTAIRE.	b	.DEBUT.	.FIN.
C. Vers. Anglais	.VALUE.	.COMMENT.	b	.BEGIN.	.END.

A noter qu'on ne doit pas trouver de blancs à l'intérieur d'un symbole de base encadré par deux points. Ceci n'interdit pas de "couper" un tel symbole ; ainsi, par exemple pour PROCEDURE, on peut écrire .PROC dans les cinq dernière colonnes d'une carte et EDURE. dans les six premières de la carte suivante.

RESTRICTIONS PAR RAPPORT AU LANGAGE DE REFERENCE

Entre parenthèses les numéros des paragraphes correspondants du rapport ALGOL.

(3. 5. 1) Les étiquettes (labels) numériques sont interdites.

(2. 4. 1) Les identificateurs peuvent être de longueur arbitraire, mais deux identificateurs dont les sept premiers caractères sont identiques ne sont pas différenciés.

(4. 1. 1) Les identificateurs de variable simple ou de tableau doivent être déclarés avant leur utilisation. Aucune restriction de ce genre n'est imposée aux identificateurs d'aiguillage ou de procédure. Considérons par exemple le programme suivant :

```
DEBUT
        REEL A, B ;
        REEL PROCEDURE F(X) ; REEL X ;
                F := SIN(X) + T[A] / G (A+B) ;
        TABLEAU T[1 : 10] ;
        REEL PROCEDURE G(Y) ; REEL Y ;
                G := COS(X/2) + T[B] ;
                I1 ; I2 ; ... ; IN ;
FIN
```

Dans le corps de la procédure F, on peut effectivement appeler la procédure G bien que celle-ci n'ait pas encore été déclarée, mais il est interdit d'appeler le tableau T ; pour éviter cet écueil, il suffit, en tête d'un bloc, d'effectuer en premier lieu les déclarations de type et de tableaux, en second lieu les déclarations de procédures et d'aiguillages.

(2. 5. 1) Les valeurs numériques rencontrées à l'exécution d'un programme doivent, en valeur absolue, être inférieures à 10^{99} pour les valeurs de type réel, à 10^{12} pour les valeurs entières. La valeur absolue d'une borne de tableau et le nombre des valeurs prises par un indice doivent être inférieures à $10^5/12$ pour un tableau réel ou entier, 10^5 pour un tableau booléen.

4. 7. 5. 5 & 5. 4. 1) Les paramètres des procédures doivent tous être spécifiés, auf, si on le désire, ceux de type réel : l'absence de spécification équivaut la spécification REEL. Les spécifications doivent être strictement respectées, es expressions arithmétiques du genre A↑B (3. 3. 4. 3), ou conditionnelles 3. 3. 4) sont considérés comme étant du type REEL.

4. 7. 5. 3) Un paramètre formel spécifié tableau ne peut pas être appelé par aleur.

3. 2. 1) Un paramètre effectif ne peut pas être réduit à un identificateur de onction usuelle (sinus, racine carrée, etc...) ; ainsi, l'on ne peut pas écrire n programme du type suivant :

```
DEBUT
        PROCEDURE F(X) ; REEL PROCEDURE X ; ... ;
        .....................................
        F(COS) ; ...
FIN
```

5. 4. 6 & 4. 7. 8) Toute procédure doit être écrite en ALGOL.

4. 7. 5. 1) Un paramètre ne peut pas être une chaîne.

On ne peut, enfin, utiliser une fonction procédure comme instruction.

FONCTIONS USUELLES (3. 2. 4)

Ces fonctions sont au nombre de 10 ; leurs identificateurs ne sont pas réservés : ils suivent les règles générales de portée, si l'on considère qu'ils sont déclarés dans un bloc contenant le programme ; ainsi l'on peut écrire le programme suivant :

```
DEBUT ...
        DEBUT REEL COS ;
        ................
        SIGNE : COS := 0. 5 ;
        ................
        FIN
FIN
```

Bien entendu, dans le bloc intérieur où l'on utilise l'étiquette SIGNE et la variable COS, il n'est pas question d'appeler les fonctions SIGNE et COS.

LISTE DES IDENTIFICATEURS DES FONCTIONS USUELLES

RAC 2 (v. anglais SORT)	racine carrée
SIN	sinus
COS	cosinus
ARCTAN	arctg
EXP	fonction exponentielle
ABS	valeur absolue
ENTIER	partie entière
SIGNE (v. anglais SIGN)	pour la fonction qui vaut 1 si son argument est > 0 0 s'il est nul - 1 s'il est négatif
CLE (v. anglais KEY)	fonction booléenne : CLE(E) = VRAI si E vaut 1, 2, 3 ou 4 et si la clé de numéro E sur le pupitre est en position haute ; dans tous les autres cas, CLE(E) = FAUX.

es arguments de ces fonctions peuvent être du type entier ou réel. Les valeurs
:s six premières sont de type réel. Le type de ABS(E) est le type de E. Les
ıleurs des fonctions ENTIER et SIGNE sont du type entier.

COMPILATION D'UN PROGRAMME

Le compilateur étant chargé, le programme Algol est lu deux fois successivement.

Lors du premier passage, aucune perforation n'a lieu.

Au second passage, il y a lecture, compilation et perforation simultanées.

Diverses vérifications sont effectuées et des libellés d'erreur peuvent être imprimés.

Si le compilateur ne détecte aucune erreur, après perforation de la dernière carte du programme-objet, la machine à écrire imprime END OF PASS 2. La machine est alors prête à compiler un nouveau programme.

Dans le cas d'une erreur, il suffit de retirer les cartes des magasins de lecture et de perforation et la machine est prête à compiler un nouveau programme. Cependant, si le libellé est ER 21, on doit taper à la machine à écrire un ordre de branchement en tête du compilateur :

49'05564

Manipulation :

A - Remettre la mémoire à zéro puis charger une fois pour toutes le compilateur ;

B - Après avoir placé le programme Algol dans le magasin de lecture, appuyer sur le bouton START au pupitre. Une fois le programme lu entièrement, la machine s'arrête en affichant le code 48 (HALTE) au pupitre : la machine à écrire tape END OF PASS 1 ;

C - Replacer le programme Algol dans le magasin de lecture pour le second passage ; appuyer sur START au pupitre et sur PUNCH. Le programme-objet est perforé ; après l'émission de END OF PASS 2, il suffit de repartir en B - pour compiler un autre programme (de même qu'après une édition d'erreur autre qu'ER 21).

XECUTION D'UN PROGRAMME

Contrairement à la compilation, le paquet de cartes de l'exécution oit être rechargé pour chaque nouveau programme.

lanipulation :

A - Remettre la machine à zéro et charger le programme-objet ;

B - Charger le paquet EXECUTION ;

C - Appuyer sur START au pupitre. Le programme est alors exécuté.

i) à la fin de l'exécution, la machine à écrire tape EXECUTION TERMINEE (Ang. END OF EXECUTION). Si l'on veut que ce même programme se déroule une seconde fois (avec d'autres données par exemple), il suffit d'appuyer deux fois sur start au pupitre ou de taper à la machine à écrire l'ordre de branchement en tête du programme d'exécution :

49 09068 RS

ii) l'exécution s'arrête sur un message d'erreur de la machine à écrire. Deux cas peuvent se présenter :

- seul le message d'erreur est tapé ; dans ce cas, on a affaire à une erreur d'entrée-sortie ; voir la partie ERREURS de la notice pour les corrections ;
- la machine à écrire a aussi tapé : EXECUTION IMPOSSIBLE (Angl. idem) ; si l'on désire recommencer les calculs, taper :

49 09068 RS.

LA PROCEDURE ENTRER

Un même appel de cette procédure permet d'entrer en mémoire un nombre quelconque de variables ou de tableaux, sans avoir à spécifier de "form

A - APPEL

L'appel est du type :

ENTRER (UNITE, I_1, I_2, ... , I_N)

- où chaque I_i peut être soit une variable, simple ou indicée, soit un identificateur de tableau ;
- où UNITE est une expression arithmétique permettant de choisir l'unité d'entrée désirée. Si au moment de l'appel cette expression vaut 1, l'entrée se fera par la machine à écrire du pupitre. Dans tous les autres cas, elle se fera par le lecteur de cartes.

B - PRESENTATION DES DONNEES

B. 1. SEPARATION DES DONNEES

Chaque donnée, y compris la dernière et même s'il n'y en a qu'une, devra être suivie de deux blancs. A l'intérieur d'une même donnée ne peuvent donc se trouver deux blancs consécutifs.

B. 2. COMMENTAIRES

Chaque donnée peut être précédée de commentaires, c'est-à-dire d'une suite quelconque de caractères soumise aux conditions suivantes, qui évitent de la confondre avec la donnée qui la suit :

i) si cette donnée est un nombre, le premier caractère du commen-
ıire ne peut être ni un chiffre, ni un point, ni l'un des signes + et -. Il ne peut
tre un E que s'il est suivi d'une lettre ;

ii) si la donnée est booléenne, le premier caractère du commentaire
précédant ne peut être ni un V, ni un F, ni l'un des chiffres 0 ou 1 ;

iii) chaque commentaire doit être suivi de deux blancs.

emarque : la présence de deux blancs ou plus en séquence à l'intérieur d'un
ommentaire revient à considérer ce dernier comme étant en fait constitué de
eux commentaires. Le second doit donc lui aussi respecter les trois règles
finies plus haut.

3. DONNEES NUMERIQUES

Aux variables du type ENTIER ou REEL devront correspondre des
nnées écrites sous forme de nombre ALGOL avec signe éventuel.

Le symbole Algol $_{10}$ pourra indifféremment être écrit . Z ou E .

Un nombre entier devra être inférieur à 10^{12}, un réel à 10^{99} en
leur absolue.

A une variable de type ENTIER pourra correspondre une donnée
ise sous forme de nombre réel ou inversement.

4. DONNEES BOOLEENNES

Chaque valeur logique peut être mise indifféremment sous l'une
s trois formes :

VRAI	FAUX
1	0
V	F

C - ENTREE PAR CARTES

Les cartes données sont placées dans le magasin de lecture après le chargement en mémoire du programme-objet. A chaque appel de ENTRER, la machine lit un certain nombre de cartes ; on ne peut donc pas faire lire deux données se trouvant sur une même carte par deux appels différents.

A la fin de l'appel, les caractères restant éventuellement sur la carte sont ignorés. D'autre part, la fin d'une carte n'a aucune signification. En conséquence :

- une donnée pourra être perforée "à cheval" sur deux cartes ;
- une donnée se terminant en colonne 80 d'une carte nécessite une nouvelle carte dont les deux premières colonnes seront vierges (même s'il s'agit de la dernière donnée) ;
- après les deux blancs obligatoires, la dernière donnée d'un appel peut être suivie d'une suite quelconque de caractères.

D - ENTREE PAR MACHINE A ECRIRE

Une fois le clavier libéré, on tape la ou les données, en séquence, puis on enfonce la touche RS.

Il est interdit de frapper une séquence de plus de 80 caractères.

Au cas où les données nécessitent plus de 80 caractères, on les sépare en lignes de 80 caractères au plus en appuyant après chacune sur la touche RS.

Remarque : la tabulation n'introduit aucun caractère en mémoire, pas même un blanc .

E - ORDRE D'AFFECTATION DES DONNEES AUX ELEMENTS D'UN TABLEAU

Si le paramètre I_i de l'appel est un tableau, il y aura lecture d'une suite de données. Les éléments du tableau seront entrés dans l'ordre tel que ce soit le dernier indice qui varie d'abord.

Supposons que l'on désire entrer en mémoire le tableau booléen suivant :

```
V V F F
V F V F
F F V V
```

En cours de programme, on aura déclaré :

```
BOOLEEN TABLEAU B[1:4 , 1:3]
```

On pourra entrer ce tableau par : ENTRER (2,B) en disposant les données comme suit :

- sur une seule carte :

```
VbbVbbFbbFbbVbbFbbVbbFbbFbbFbbVbbVbbbbbb ... bb
```

- sur plusieurs cartes, par exemple :

```
VbbVbbFbbFbbbb ... bb
VbbFbbVbbFbbbb ... bb
FbbFbbVbbVbbbb ... bb .
```


LA PROCEDURE SORTIR

Cette procédure est inspirée, notamment en ce qui concerne les formats, par un rapport du SOUS-COMITE ALGOL du COMITE des LANGAGES de PROGRAMMATION de l'A. C. M.

La procédure SORTIR est une procédure de composition, c'est-à-dire chargée de remplir progressivement en mémoire une ligne qui sera éditée sous certaines conditions.

A - APPEL

L'appel est de l'un des types :

1) SORTIR (UNITE, CHAINE-FORMAT, LISTE)

- où UNITE a le sens déjà défini lors des entrées,
- où LISTE est la liste des expressions arithmétiques ou booléennes à sortir. Toutes ces expressions sont sorties suivant la chaîne-format indiquée au second paramètre.

Cette chaîne-format est une silhouette conventionnelle des résultats qui seront imprimés. La manière d'écrire la chaîne-format est décrite en notations de BACKUS aux paragraphes B. 1. 1 et B. 3. 1.

2) SORTIR (UNITE, LISTE)

permettant de sortir les résultats sous forme normalisée (cf. paragraphe E) .

3) SORTIR (UNITE, CHAINE-FORMAT)

permettant de sortir les commentaires indiqués dans la chaîne-format. (cf. paragraphe C. 3. 1).

4) SORTIR (UNITE)

permettant de sortir un commentaire, préalablement entré par un appel convenable de ENTRER (cf. paragraphe D).

B - LES FORMATS

B. 1. FORMATS DE NOMBRES

B. 1. 1. Syntaxe

Composantes de base

⟨Multiplicateur⟩ ::= ⟨chiffre⟩ | ⟨chiffre⟩ ⟨chiffre⟩

⟨Insertion⟩ ::= B | ⟨multiplicateur⟩ B | ⟨chaîne⟩

⟨Séquence d'insertion⟩ ::= ⟨vide⟩ | ⟨séquence d'insertion⟩ ⟨insertion⟩

⟨Z⟩ ::= Z | ⟨multiplicateur⟩ Z

⟨Partie Z⟩ ::= ⟨Z⟩ | ⟨partie Z⟩ ⟨Z⟩ | ⟨partie Z⟩ ⟨insertion⟩

⟨D⟩ ::= D | ⟨multiplicateur⟩ D

⟨Partie D⟩ ::= ⟨D⟩ | ⟨partie D⟩ ⟨D⟩ | ⟨partie D⟩ ⟨insertion⟩

⟨Partie T⟩ ::= ⟨vide⟩ | T ⟨séquence d'insertion⟩

⟨Partie signe⟩ ::= ⟨vide⟩ | ⟨séquence d'insertion⟩ + | ⟨séquence d'insertion⟩

⟨Partie entier⟩ ::= ⟨partie Z⟩ | ⟨partie D⟩ | ⟨partie Z⟩ ⟨partie D⟩

Structure des formats

⟨Format d'entier sans signe⟩ ::= ⟨séquence d'insertion⟩ ⟨partie entier⟩

⟨Format de partie décimale⟩ ::= . ⟨séquence d'insertion⟩ ⟨partie D⟩
⟨partie T⟩

⟨Format de partie exposant⟩ ::= E ⟨partie signe⟩ ⟨format d'entier sans signe⟩

⟨Format de nombre décimal⟩ ::= ⟨Format d'entier sans signe⟩ ⟨partie T⟩ |
⟨séquence d'insertion⟩ ⟨format de partie décimale⟩ | ⟨format d'entier sans signe⟩ ⟨format de partie décimale⟩

⟨Format de nombre⟩ ::= ⟨partie signe⟩ ⟨format de nombre décimal⟩ |
⟨format de nombre décimal⟩ + ⟨séquence d'insertion⟩ |
⟨format de nombre décimal⟩ - ⟨séquence d'insertion⟩ |
⟨partie signe⟩ ⟨format de nombre décimal⟩
⟨format de partie exposant⟩ | ⟨format normalisé⟩

Remarques : i) chaîne correspond à la définition d'Algol ;
ii) format normalisé sera défini plus bas .

B. 1. 2. Sémantique

B. 1. 2. 1. Multiplicateurs : Un entier sans signe N utilisé comme multiplicateur signifie que le caractère qui le suit est répété N fois. Ainsi, 3B est équivalent à BBB. Si N=0, ce caractère est supprimé. Par exemple BB0ZDD = BBD!

B. 1. 2. 2. Insertions : La syntaxe du A. 1. 1 a été écrite de façon que l'on puiss insérer, n'importe où dans un format de nombre, une chaîne délimitée par ses guillemets ouvert et fermé. Le contenu de ces chaînes apparaîtra à la sortie au niveau externe à la même place dans le nombre (les guillemets étant supprimés). De même la lettre B peut être placée en tout endroit d'un format de nombre. Au niveau externe apparaîtra alors un blanc.

B. 1. 2. 3. Suppression des zéros et des signes : Dans un nombre, à gauche du point décimal, on admet éventuellement un signe, une séquence de Z et une séquence de D (et des insertions).

— La convention pour le signe est la suivante :

a) s'il n'y a pas de signe dans le format, le nombre doit être positif ou nul ;

b) si un signe + est indiqué, le signe, + ou -, apparaîtra au niveau externe en accord avec le signe du nombre ;

c) s'il y a un signe - dans le format, on aura à la sortie un caractère - si le nombre est négatif, un blanc s'il est positif ou nul.

— La lettre Z signifie "suppression des zéros", et la lettre D "impression d'un chiffre, sans suppression des zéros". Chaque D ou Z compte pour un seul digit. Tout chiffre spécifié par un Z sera supprimé, c'est-à-dire remplacé par un blanc, quand tous les chiffres à sa gauche sont déjà des blancs au niveau externe. Un chiffre spécifié dans un format par un D sera toujours imprimé. A noter que le nombre zéro spécifié par un format ne comprenant que des Z donnera au niveau externe une zone de blancs. Un D serait nécessaire pour faire apparaître au moins le chiffre zéro.

— Si des suppressions de zéros sont effectuées, le signe (s'il existe) sera imprimé à la place du zéro supprimé le plus à droite.

B. 1. 2. 4. Point décimal : Il est indiqué par le caractère "." ; il apparaît toujours au niveau externe à la même place que dans le format. A sa droite, il est impossible d'indiquer des Z, seuls des D et des T sont admis.

B. 1. 2. 5. Arrondi et troncature : Les réels sortiront au niveau externe après arrondi sur le chiffre correspondant au dernier D. Mais si la lettre T est utilisée dans le format, cet arrondi n'est pas effectué. Arrondi et troncature d'un nombre A dont le format demande d chiffres décimaux sont définis comme suit :

Arrondi	10^{-d} ENTIER (10^{d} A + . 5)
Troncature	10^{-d} SIGNE (A) x ENTIER (10^{d} ABS(A)) .

B. 1. 2. 6. Format de nombre : On peut grouper les formats de nombres en deux rubriques :

1) Formats de nombre décimal : le nombre est alors aligné en fonction de la place du point décimal (ou s'il n'existe pas, en fonction du dernier chiffre à droite).

Exemples : le nombre à sortie est supposé être 123. 456

D	D	D
1	2	3

-	D	D	D	.	D	D
	1	2	3	.	4	6

+	Z	Z	Z	D	D	.	D
		+	1	2	3	.	5

+	Z	Z	Z	D	D
		+	1	2	3

-	D	D	D	D	D
	0	0	1	2	3

D	D	D	.	D	T
1	2	3	.	4	5

2) Formats de nombre avec partie exposant : dans ce cas, le nombre est sorti suivant le format demandé, sans zéro ni blanc à gauche de sorte que le nombre de chiffres significatifs soit maximum.

Exemple, pour sortir le nombre $-0.123456789 \; 8_{10}26$

-	Z	Z	D	D	.	D	D	D	D	E	+	Z	D
-	1	2	3	4	.	5	6	7	9	E	+	2	2

Remarques :

i) cas où le nombre est nul : tous les chiffres spécifiés par des D sortent sous forme de zéros. Le nombre est considéré comme positif.

Exemples :

Z	Z	D
		0

-	Z	Z	Z	Z

+	Z	Z
		+

-	Z	D	D	.	D	D	D	E	-	D
		0	0	.	0	0	0	E		0

ii) insertions et blancs dans un nombre : ils sortent à l'emplacement indiqué.

Exemples :

-	Z	D	B	D	D	D	.	D	D	D	B	D	D	D
	-	1		2	3	4	.	5	6	7		8	9	5

A=		B	-	D	D
A=			-	1	2

D	D	E	M	E		C	A	S
5	6	E	M	E		C	A	S

B. 2. AUTRES FORMATS

B. 2. 1. Syntaxe

⟨Partie booléenne⟩ ::= L | 4F | FFFF | F

⟨Format booléen⟩ ::= ⟨séquence d'insertion⟩ ⟨partie booléenne⟩
⟨séquence d'insertion⟩ | ⟨format booléen⟩

⟨Format normalisé⟩ ::= ⟨séquence d'insertion⟩ N ⟨séquence d'insertion⟩

⟨Marque d'alignement⟩ ::= S | /|⟨multiplicateur⟩ S | ⟨multiplicateur⟩

⟨Format de titre⟩ ::= ⟨insertion⟩ | ⟨format de titre⟩ ⟨insertion⟩

⟨Détail de format⟩ ::= ⟨format de nombre⟩ | ⟨format booléen⟩ |
⟨format de titre⟩ | ⟨marque d'alignement⟩ ·
⟨détail de format⟩

⟨Elément de format⟩ ::= ⟨détail de format⟩ | ⟨marque d'enregistrement⟩ |
⟨élément de format⟩ ⟨marque d'enregistrement⟩

3. 2. 2. Sémantique

3. 2. 2. 1. Format booléen : une quantité booléenne pourra être mise sous orme externe avec l'un des aspects suivants :

Format	L	F	4F ou FFFF
VRAI	1	V	VRAI
FAUX	0	F	FAUX

3. 2. 2. 2. Format normalisé : suivant le type de l'expression lui correspondant dans la liste de sortie, N sera équivalent à l'un des formats :

Expression de type ENTIER -11ZD

Expression de type REEL -. 10DE-DD

Expression de type BOOLEEN F

'our plus de détails, cf. paragraphe D.

. 2. 2. 3. Format de titre : tous les formats étudiés jusqu'à présent donnaient ne équivalence entre le niveau interne et le niveau externe pour une variable éelle entière ou booléenne. Par contre, un format de titre ne comprend que es insertions et ne possède donc pas d'équivalent au niveau interne. Ces formats servent à sortir une série de commentaires, (titres par exemple) définis n insertions.

. 2. 2. 4. Marques d'alignement : les caractères S et / dans un élément de ormat indiquent des opérations d'alignement. En première approximation, signifie "impression et commencement d'une nouvelle ligne de composition", : S "tabulation". Ce dernier n'a aucune valeur lorsque la sortie se fait sur artes. Il est possible de faire plusieurs opérations d'alignement en séquence. insi, //// commandera l'impression de la ligne en cours de composition, uis de trois lignes blanches.

B. 3. CHAINES-FORMATS

Les éléments de format tels que nous venons de les décrire peuvent se combiner entre eux pour former une chaîne-format, à condition de suivre les règles suivantes :

B. 3. 1. Syntaxe

⟨Format primaire⟩ ::= ⟨élément de format⟩ |
⟨multiplicateur⟩ (⟨format secondaire⟩) |
(⟨format secondaire⟩)

⟨Format secondaire⟩ ::= ⟨format primaire⟩ , ⟨format primaire⟩

⟨Chaîne-format⟩ ::= '⟨format secondaire⟩'

B. 3. 2. Exemples

4F et 'X=' 3Z. 2DE-ZD sont des formats primaires.

4F, 'X=' 3Z. 2DE-ZD est un format secondaire.

3(4F, 'X=' 3Z. 2DE-ZD) est un format primaire.

Exemples de chaînes-formats :

'4(15ZD)/,L,//' '/'

''X=' 5D,8('...')/' '. 5DE-D,X(2(20B. 8DE-D)/,SN)'

''BAISSER LA CLE NUMERO 2' /,S 'N='

B. 3. 3. Sémantique

Une chaîne-format est en fait une suite d'éléments de formats séparés par des virgules et qui sont interprétés de la gauche vers la droite.

La construction ⟨multiplicateur⟩ (⟨format secondaire⟩) signifie que la quantité entre parenthèses est répétée "multiplicateur" fois, une virgule étant insérée chaque fois entre deux répétitions. Si N=0, les parenthèses et leur contenu sont supprimées.

Ainsi 2(20B. 8DE-D) sera équivalente à 20B. 8DE-D, 20B. 8DE-D.

La construction (⟨format secondaire⟩) est utilisée pour permettre un nombre de répétitions quelconque de la partie entre parenthèses.

B.4. REMARQUES SUR LES FORMATS

B.4.1. Limites

En pratique, on ne peut pas mettre de chaînes-formats de plus de :00 caractères (y compris guillemets et autres séparateurs). Cette chaîne, une ois les multiplicateurs supprimés (c'est-à-dire après la répétition), ne devra as dépasser 800 caractères.

Un détail de format ne peut avoir plus de 80 caractère au niveau xterne, c'est-à-dire qu'il ne pourra jamais sortir sur deux lignes.

3.4.2. Remarques

Les expressions suivantes : 4'...' et 4N n'ont pas de sens ; il audrait respectivement les écrire 4('...') et 4(N).

.4.3. Résumé des codes

B	Espace blanc
D	Chiffre, y compris zéro
E	Symbole remplaçant le $_{10}$ d'Algol
F	Booléen (V ou F)
L	Logique (1 ou 0)
N	Format normalisé
S	Tabulation
T	Troncature
Z	Suppression des zéros
-	Impression du signe si c'est -
+	Impression du signe si (+ ou -)
.	Position du point décimal
/	Changement de ligne
( ) , ' b	Délimiteurs

e caractère Algol ␣ est ignoré dans un format.

C - LA PROCEDURE SORTIR

C.1. CARACTERISTIQUES GENERALES

Considérons d'abord le cas où nous voulons sortir, sur cartes, une ligne contenant les valeurs des entiers M et N avec au plus 5 chiffres, puis la valeur de la variable indicée X[M] sous la forme d'un nombre avec signe, avec un seul zéro à gauche du point décimal et indication de l'exposant, et enfin la valeur de COS(T) en utilisant un format de nombre décimal sans partie exposant. Par ailleurs, nous voulons séparer chacune des données par trois espacements. On peut réaliser cela par l'appel suivant :

```
SORTIR(2,'2(BBBZZZZD),3B,+D.DDDDDDE+DDD,3D,
        -Z.DDDDBDDDD/',N,M,X[M],COS(T)) ;
```

Le premier paramètre (2) indique une sortie sur cartes.

Le suivant est la chaîne de formats qui spécifie l'aspect externe des valeurs des 4 paramètres suivants de la procédure SORTIR.

Si N=500, M=0, X[0]=18061579 et T=3.1415926536, on composera la ligne :

bbbbb500bbbbbbb0bbb+1.806158E+007bbb-1.0000b0000 ,

ligne qui sera ensuite imprimée comme le demande la barre / .

Si / n'avait pas été mise, d'autres nombres auraient pu apparaître sur la même ligne lors d'un autre appel de SORTIR.

L'appel de SORTIR étant un appel de composition, on obtient le même résultat par les appels suivants :

```
POUR I:=M,N FAIRE SORTIR(2,'BBBZZZZD',I) ;
SORTIR(2,'3B,+D.DDDDDDE+DDD,3B,-Z.DDDDBDDDD',
       X[M],COS(T)) ;
```

C.2. SYNTAXE D'UN APPEL DE LA PROCEDURE SORTIR

```
<Procédure SORTIR> ::= SORTIR ( <unité> ) |
                       SORTIR ( <unité> , <liste de sortie> ) |
                       SORTIR ( <unité> , <format> ) |
                       SORTIR ( <unité> , <format> , <liste de sortie
```

<FORMAT> ::= <chaîne-format>

<Liste de sortie> ::= <expression arithmétique> |
<expression arithmétique> , <liste de sortie> |
<expression booléenne> , <liste de sortie> |
<expression booléenne>

C. 3. SEMANTIQUE

Les cas particuliers d'appels de SORTIR sans format ont été déjà étudiés. Rappelons que <unité> est une expression arithmétique dont la valeur détermine le numéro de l'organe de sortie à l'exécution.

C. 3. 1. Procédure sans liste de sortie

Dans ce cas, le format ne peut contenir que des insertions ou des marques d'alignement. Ceci mis à part, l'exécution se fait comme pour les appels avec liste.

C. 3. 2. Liste de sortie

Comme pour ENTRER, on admet une séquence quelconque de paramètres, mais cette fois ces derniers peuvent être des expressions et ne peuvent plus être des identificateurs de tableaux. La liste de sortie sert à spécifier les quantités qui seront mises sous la forme externe selon le format leur correspondant dans la partie format. Ces valeurs sortent dans l'ordre où elles sont écrites.

C. 3. 3. Parallèle entre la partie format et la liste de sortie

A chaque élément de la liste de sortie doit correspondre un détail de format du même type.

Les éléments de format seront étudiés les uns après les autres. Ceux ne contenant que des insertions ou des marques d'alignement seront traités immédiatement ; les autres produiront l'appel de la valeur de l'expression ayant le même rang dans la liste de sortie.

Exemple : Si N, R et B sont respectivement des expressions de type ENTIER, REEL et BOOLEEN, l'appel

SORTIR (n, '3D/', 'B EST'BFFFF, 'R='N',E,B,R) ;

sera traité comme les appels successifs :

SORTIR(n, '3D/',E) ;
SORTIR(n, ''B EST'BFFFF',B) ;
SORTIR(n, ''R='') ;
SORTIR(n, 'N',R) ;

D - ENTREE ET SORTIE DE TITRES

On peut utiliser les procédures ENTRER et SORTIR de façon à entrer, puis sortir 80 caractères alphanumériques au plus.

D. 1. APPELS

Les deux appels doivent être ENTRER(U) ; I ; SORTIR(U') .

- où I désigne une suite d'instructions ne contenant pas d'appel de la procédure SORTIR.
- où U et U' sont des expressions arithmétiques. Si U ou U' = 1, l'unité sélectée sera la machine à écrire ; dans tous les autres cas, ce sera le lecteur-perforateur de cartes. U et U' sont indépendants.

D. 2. EXECUTION

A l'appel de ENTRER, une demande de lecture est faite ; selon l'unité U_i sélectée

- faire lire la carte portant le titre ;
- ou taper les caractères du titre (80 au plus) et appuyer sur la la touche RS.

Cette chaîne alphanumérique sera sortie, dès l'appel de SORTIR, avec la même présentation qu'elle avait à l'entrée. Il ne s'agit que de simple reproduction.

E - SORTIE NORMALISEE

On appelle ainsi une façon élémentaire de se servir de la procédure SORTIR. Elle permet la sortie immédiate d'une séquence quelconque d'expressions arithmétiques ou booléennes.

Les sortie normalisées correspondent à l'appel :

SORTIR(UNITE,LISTE) ;

cet appel est équivalent à un appel du type :

SORTIR(UNITE,(N),LISTE) . (cf. B. 2. 2. 2).

E. 1. FORME EXTERNE D'UNE EXPRESSION

Chaque expression est sortie selon son type et occupe une longueur fixe L.

i) pour une expression de type booléen : L = 3 .

Les valeurs VRAI et FAUX sortent respectivement sous la forme V suivi de 2 blancs et F suivi de deux blancs.

ii) pour une expression de type entier : L = 15.

Les entiers sortent suivant le format -11ZD et sont suivis de deux blancs.

Exemples :

```
-1 2 3 4 5 6 7 8 9 0 1 2
 1 2 3 4 5 6 7 8 9 0 1 2
       - 1 2 3 4 5 6 7
             1 2 3 4
               - 1 0
                   1
                 - 1
                   0
```

iii) pour une expression de type REEL : L = 18.

Les réels sortent suivant le format -. 10DE-DD et sont suivis de deux blancs.

Exemples :

```
-. 1 2 3 4 5 6 7 8 9 0 E - 1 2
 . 1 2 3 4 5 6 7 8 9 0 E   1
 . 3 1 4 1 5 9 2 6 5 3 E - 6
-. 9 8 7 6 5 4 3 2 1 0 E   0
 . 0 0 0 0 0 0 0 0 0 0 E   0
```

E. 2. PRESENTATION DES RESULTATS

Les expressions intervenant dans LISTE sont sorties en séquence sur une ou plusieurs cartes (ou lignes de machine à écrire si unité = 1). Si une expression devait être "à cheval" sur 2 cartes (ou 2 lignes), elle serait reportée sur la seconde, la première étant complétée par des blancs).

Remarque : Contrairement à ce qui se passe pour les sorties étendues (avec format), ces expressions sont imprimées dès que la procédure SORTIR est appelée.

F - COMPOSITION ET EDITION

Ainsi que l'on voit, la sortie se fait pratiquement en deux temps composition, puis édition.

Il existe en mémoire deux lignes-composition de 80 caractères, indépendantes, la première pour sortie sur machine à écrire, la seconde pour les sorties sur cartes.

Chaque appel de SORTIR a pour effet de remplir une portion de la ligne choisie (en fonction de la valeur du premier paramètre de SORTIR), immédiatement à droite de la partie déjà remplie, et de l'éditer si cette ligne est pleine.

Il n'existe pas de procédure d'édition.

Cette opération se fera

- soit par l'emploi de formats / et S (cf. B. 2. 2. 4)
- soit lorsqu'elle sera imposée.

L'édition est imposée dans les cas suivants :

- un détail de format ne pourrait pas tenir dans la place qu'il reste dans la ligne de composition ;
- un appel de sortie normalisée ou de titre a été effectué ;
- à la fin de l'exécution du programme-objet (et ce pour les deux lignes de sortie si elles ne sont pas vierges).

LISTE DES ERREURS DETECTEES EN COURS DE COMPILATION

Libellé ER suivi d'un n° d'ordre.

N°	Signification de l'erreur
01	Dépassement de capacité ! nombre dont la valeur absolue est supérieure à 10^{99}
02	Erreur syntaxique dans l'écriture d'un nombre
03	Nombre suivi d'une lettre
04	Erreur dans le nombre d'indices d'une variable indicée
05	Une variable intervenant dans une borne d'une déclaration de tableau n'a pas été déclarée dans un bloc plus grand
11	Un identificateur de procédure n'est pas suivi de (ni de ;
12	Un paramètre formel n'est pas suivi de , ni de)
13	Séparateur incorrect entre deux paramètres formels ou effectifs d'une même liste
14	Symbole de base erroné
15	Une spécification n'est pas suivie de , ni de ;
16	Spécifieur REMANENT
17	Dans le cas de deux spécifieurs le second n'est pas TABLEAU ni PROCEDURE
18	Appel par VALEUR d'un TABLEAU ou d'un AIGUILLAGE
19	Spécification ou appel par VALEUR d'un paramètre n'appartenant pas à la procédure déclarée
20	On retrouve deux fois VALEUR dans une déclaration de procédure
21	Il manque un FIN ou le ; qui suit une déclaration de procédure
22	Erreur dans le nombre de paramètres d'une procédure
25	Spécification non respectée
27	Nombre de dimensions différent pour un paramètre effectif TABLEAU et le TABLEAU correspondant
28	Trop grand nombre d'identificateurs
29	Saturations des piles

N°	Signification de l'erreur
30	Succession incorrecte de symboles Algol
31	Les opérandes de ≡ , ∨ , ∧ , ⊃ , ¬ ; . SI. , . TANTQUE. , ne sont pas du type booléen
32	Types incorrects pour les opérandes des opérateurs arithmétiques ou de relation
33	Affectation incorrecte : entre un BOOLEEN d'une part et un REEL ou un ENTIER d'autre part
34	Une partie gauche d'une instruction d'affectation n'est pas une variable
35	Identificateur non déclaré ou utilisé en dehors de sa portée. Les sept premiers caractères de l'identificateur sont indiqués avant le libellé ER 35
37	. ALLERA. n'est pas suivi d'une expression de désignation
39	Crochet fermé sans être ouvert
40	Un identificateur ne succède pas à . VALEUR. , . PROCEDURE. , . TABLEAU. , . CHAINE. , . AIGUILLAGE. , ou . ETIQUETTE.
41	Expression Algol prise comme instruction

Les erreurs suivantes sont détectées à la compilation des procédures d'E/S. Libellé ER suivi d'un numéro d'ordre.
Dans les exemples nous supposerons que E est un entier, R un réel, T un tableau à 2 dimensions, B un booléen et ETI une étiquette.

N°	Signification de l'erreur	Exemples
79	Nombre incorrect de virgules	ENTRER(2,,E)
80	Paramètre impossible	ENTRER(2,ETI)
81	Le numéro d'organe n'est pas une expression arithmétique	SORTIR(B,E)
82	Un paramètre de la liste d'entrée n'est pas valable	ENTRER(2,COS(R))

N°	Signification	Exemple
83	Un paramètre de la liste de sortie n'est pas valable	SORTIR(2,T)
84	Une chaîne apparaît ailleurs que comme second paramètre	SORTIR(2, A ,'DD',E
88	Il manque au moins un paramètre pour FORMAT	FORMAT ('DD')
89	Le premier paramètre de FORMAT n'est pas une chaîne	FORMAT(2,'XD',E)
91	Le second paramètre de FORMAT n'est pas une expression arithmétique	
92	Un caractère X est utilisé dans une chaîne en dehors de FORMAT	SORTIR(2,'XD',E)
93	Le nombre d'expressions arithmétiques dans la liste de FORMAT n'est pas égal au nombre de X dans la chaîne correspondante	FORMAT('XBXD',E)
94	Une chaîne n'est précédée ni d'une virgule ni d'une parenthèse	
95	Une chaîne n'est suivie ni d'une virgule ni d'une parenthèse	
96	Erreur de syntaxe dans une chaîne : un multiplicateur est suivi d'un caractère non valable	'XB X E'
97	Caractère invalide dans une chaîne	'4DC3B'
98	Chaîne trop longue vu l'état des piles	
99	Chaîne de plus de 200 caractères	

LISTE DES ERREURS DETECTEES EN COURS D'EXECUTION

A - Erreurs détectées à l'exécution de ENTRER

Libellé ER E suivi d'un numéro d'ordre.

N°	Signification	Exemples
2	Faute dans l'écriture d'une donnée booléenne	VRAIbCARTEb
3	La valeur absolue d'un nombre est plus grande que 10^{99} pour un réel, 10^{12} pour un entier	. Z 14 - E 110
4	Faute de syntaxe dans une donnée numérique	- 12. . 34
5	Entrée sur machine à écrire d'une ligne de plus de 80 caractères	
6	Entrée sur machine à écrire d'une ligne de plus de 160 caractères	

Corrections de ces erreurs

Les erreurs E5 et E6

Le programmeur n'a pas suivi la restriction imposée sur le nombre de caractères pouvant entrer par ligne de machine à écrire. Deux cas peuvent se présenter :

- il y a eu entre 80 et 160 caractères tapés : on ignore ceux qui suivent le $80^{ème}$ et l'exécution se poursuit normalement, mais il risque de se produire ensuite une erreur E 2 à E 4.

- il y a eu plus de 160 caractères entrés : les sous-programmes d'exécution peuvent avoir été détruits. Toute exécution est désormais aléatoire, on préfère l'arrêter ; le libellé EXECUTION IMPOSSIBLE est tapé ; il est nécessaire de tout recharger. (Seul cas où une erreur d'entrée-sortie ne peut être corrigée).

Les erreurs E2 à E4

Après l'impression du libellé, la machine à écrire tape la dernière ligne entrée, puis met une astérisque sous le dernier caractère étudié qui est celui erroné dans le cas des erreurs E2 ou E4, le premier qui suit la donnée erronée dans le cas E3.

Enfin, sort l'indication NUMERO COLONNE = .

Deux possibilités sont alors laissées au choix de l'opérateur : ou recommencer toute l'entrée erronée, ou corriger l'erreur.

i) recommencer toute l'entrée erronée

il faut alors :

- taper une marque d'enregistrement (±)
- enfoncer la touche RS (après avoir éventuellement libéré le magasin de l'unité de lecture)
- réintroduire dans le magasin de lecture toutes les données corrigées correspondant à l'appel que l'on reprend.

ii) corriger la donnée erronée

- taper le numéro du caractère à partir duquel on veut introduire de nouveaux caractères. Ce numéro doit être tapé avec deux chiffres
- enfoncer la touche RS
- lorsque le clavier est débloqué, retaper entièrement la fin de la donnée erronée (et pas seulement le ou les caractères erronés)
- enfoncer à nouveau la touche RS.

La lecture reprendra à partir de la colonne dont on a tapé le numéro, C; si l'on a corrigé n caractères, la lecture se continuera à partir du $n+c^{ième}$ caractère de la ligne qui a été recopiée.

B - Erreurs détectées à l'exécution des procédures de SORTIE

Libellé ER S suivi d'un numéro d'ordre.

N°	Signification de l'erreur	Exemples
1	Un détail de format demande plus de 80 caractères au niveau externe	SORTIR(2,'85B')
2	Le nombre de formats numériques ou booléens est supérieur au nombre d'éléments de la liste de sortie	SORTIR(2,'DD,4F',E)
3	Le nombre de formats numériques ou booléens est inférieur au nombre d'éléments de la liste de sortie	SORTIR(2,'DD',E,B)
4	A un format booléen correspond une variable numérique	SORTIR(2,'L',E)
5	A un format numérique correspond une variable booléenne	SORTIR(2,'4D', B)
6	Nombre trop grand pour sortir en respectant le format numérique	SORTIR(2,'DD', 123)
7	Erreur de syntaxe dans un élément de format	SORTIR(2,'BB/N',E)
8	Sortie de titre impossible	
9	Erreur de syntaxe dans la partie décimale d'un format numérique	SORTIR(2,'ZZ.E',E)
10	Erreur de syntaxe dans la partie exposant d'un format numérique	SORTIR(2,'4D.DEDD',E)
11	Erreur dans un format booléen	SORTIR(2,'FF',B)
12	Multiplicateur écrit avec trois chiffres	SORTIR(2,'010D',E)
13	Multiplicateur X associé à une variable numérique	
14	Multiplicateur X associé à un nombre plus grand que 99	
15	Saturation de la machine : trop de multiplicateurs dans la chaîne format	

Correction des erreurs dans les procédures de sortie

Autocorrection par non-opération

Les erreurs S2 et S7 sont signalées, et l'on ne tient pas compte de ce qui est en trop.

Exemple : SORTIR(2,'4D,4F',E) ; E sortira sous le format 4D,et 4F sera ignoré.

Sortie normalisée

Toutes les autres erreurs conduisent à une sortie normalisée : l'élément de liste sort avec un format normalisé ainsi que tous les suivants du même appel.

Exemple : l'appel SORTIR(1,'/4DBB,FBB,3D',1234, -1.0,123) produira les lignes suivantes :

```
ER S4
1234 -.1000000000E 01        123
```

Sortie normalisée de tableau

Dès la détection d'une erreur, les composantes qui ne sont pas encore sorties sont mises sous forme normalisée.

Erreurs de multiplicateurs

S 12 : sortie normalisée

S 13 : on prend la valeur absolue de X

S 14 : X est traité modulo 100

Erreurs locales et définitives

On appelle erreur locale une erreur qui ne peut apparaître qu'au moment d'un appel particulier, et qui ne pourra pas se reproduire. Le libellé d'erreur n'apparaît qu'à chaque appel erroné.

Exemple : .POUR. E:=12, 13, 144, 15 .FAIRE. SORTIR(1, '/DD',E) donnera :

12
13
ER S6
144
15

Lorsqu'au moment du premier appel on détecte une erreur qui se répétera à chaque appel, le libellé correspondant est imprimé, la sortie se fait sous forme normalisée. A chaque nouvel appel, la sortie sera normalisée, mais il n'y aura plus de libellé d'erreur.

Exemple : .POUR. E:=1,2,3 .FAIRE. SORTIR (1, '/4F',E) donnera

ERS4
1
2
3

C - Erreurs signalées dans les sous-programmes fonctionnels

Les erreurs sont signalées après une correction imposée. Le programme se déroule sans discontinuité, mais avec des valeurs qui peuvent être fausses. Cela permet toutefois d'en vérifier le bon déroulement logique. Cette remarque est également valable pour le paragraphe suivant D.

Libellé de ER A suivi d'un numéro d'ordre

N°	Nature de l'erreur	Résultat imposé
1	Un réel est trop grand en valeur absolue pour être converti en entier	$\pm (10^{12} - 1)$
2	Les deux arguments de l'exponentiation sont nuls	1
3	Exponentiation du type $0 \uparrow (-i)$	0
4	Exponentiation du type $A \uparrow R$ où A est négatif et R réel	$ABS(A) \uparrow R$
5	Dépassement de capacité dans l'exponentiation	10^{99}

D - Erreurs signalées lors du traitement des fonctions usuelles

Libellé ER F suivi d'un numéro d'ordre.

N°	Nature de l'erreur	Résultat imposé
1	Perte de toute précision dans SIN ou COS	0
2	LN a un argument nul	-10^{99}
3	LN a un argument négatif	LN(ABS(E))
4	Dépassement de capacité supérieur dans EXP	10^{99}
5	Dépassement de capacité inférieur dans EXP	0
6	RAC2 a un argument négatif	RAC2(ABS(E))

E - Autres erreurs signalées lors de l'exécution d'un programme

Libellé ER suivi d'un numéro d'ordre.

N°	Nature de l'erreur
11	L'indice d'une variable indicée est inférieur à la plus petite valeur qu'il peut prendre
12	L'indice d'une variable indicée est supérieur à la plus grande valeur qu'il peut prendre
13	Borne inférieure trop grande dans une déclaration de tableau
14	Dans une déclaration de tableau, un indice peut prendre trop de valeurs
17	Saturation de la mémoire lors de l'exécution de procédures
20	L'indice d'un aiguillage est négatif, nul ou supérieur au nombre de sorties prévues pour cet aiguillage L'impression de ce message n'arrête pas l'exécution du programme qui continuera en séquence (rapport Algol 4. 3. 5).

Ces erreurs, à l'exception de ER 20, arrêtent l'exécution du programme ; le libellé EXECUTION IMPOSSIBLE est tapé.

---o---

LISTAGE DES PROGRAMMES

CORRESPONDANT AU TRAITEMENT

DES PROCEDURES ET DES AIGUILLAGES

A LA COMPILATION ET A L'EXECUTION

```
*
******    COMPILATION DES PROCEDURES

******    PREMIER PASSAGE


LPROC AM   CTNIV,1,10,                    TRAITEMENT DE
      AM   MEMSO,2,10,                    L'IDENTIFICATEUR
      TFM  MEMSO,41,610,                  DE PROCEDURE
      AM   MEMSO,5,10
      SF   LPROC
      TFM  ADER,0,9
      BTM  SSP7,*+12
      AM   LTAB,5,10
      TR   LTAB,PR1,6
      TF   LTAB,REMAN,6
      SM   REMAN,5,10
      AM   LTAB,8,10
      TF   LTAB,CTNIV,610
      AM   LTAB,3,10
      TF   IDPRO,LTAB
      AM   LTAB,12,10
      CM   LSYMB,36,10
      BNE  *+36
      TDM  LTAB,0,6
      B    1LPROC
      CM   LSYMB,37,10
      BNE  *+36
      TDM  LTAB,1,6
      B    1LPROC
      CM   LSYMB,38,10
      BNE  1LPROC
      TDM  LTAB,3,6
1LPROCAM   LTAB,3,10,                     TRAITEMENT DE LA
      AM   MEMSE,2,10,                    LISTE DE PARAME-
      CM   MEMSE,03,610,                  TRES FORMELS
      BNE  2LPROC
      BTM  DECAL,23,10
      BE   3LPROC-12
      B    ERSY2
2LPROCCM   MEMSE,24,610
      BNE  ERSY1
      BTM  SSP7,*+12
      SM   LTAB,12,10
      AM   LTAB,30,610
      AM   LTAB,17,10
      TFM  LTAB,0,67
      TR   LTAB,PR3,6
      AM   LTAB,8,10
      TF   LTAB,CTNIV,610
      AM   LTAB,14,10
```

```
        AM   ADER,1,9
        MM   ADER,17,1011
        TF   LTAB,99,6
        AM   LTAB,90005,67
        AM   LTAB,4,10
        AM   MEMSE,2,10
        CM   MEMSE,23,610
        BE   2LPROC+24
        CM   MEMSE,4,610
        BNE  ERSY2
        BTM  DECAL,03,10
        BNE  *-12
        BTM  DECAL,23,10
        BE   *+60
        CM   MEMSE,3,610
        BNE  ERSY3
        BTM  DECAL,03,10
        B    2LPROC
        CF   SSP7
        TF   IDPRO,ADER,6
3LPROCTFM  LSYMB,30,10
        BTM  DECAL,03,10
        BE   4LPROC
        SM   MEMSE,2,10,          FIN DE TETE DE
        SF   SSP7,,,              PROCEDURE
        SF   SEPAR
        B    SEPAR
4LPROCBTM  DECAL,03,10
        BTM  SSP3,0,10
        AM   99,2,10
        TF   LSYMB,99,11
        CM   LSYMB,52,10
        BNE  5LPROC
        AM   MEMSE,18,10
        BTM  LCTAIR,3LPROC
5LPROCCM   LSYMB,51,10,           TRAITEMENT PARTIE
        BNE  6LPROC,,,            VALEUR
        BNF  ERVAL,LPROC
        BTM  SSP7,*+12
        BTM  LCONS,00,10
        AM   LTAB1,11,10
        SM   LTAB1,1,610
        AM   MEMSE,2,10
        CM   MEMSE,23,610
        BE   5LPROC+36
        CF   LPROC
LSPEC   CM   MEMSE,03,610,        TRAITEMENT DE LA
        BNE  ERSY5,,,             PARTIE SPECIFICATION
        BTM  DECAL,23,10
        BNE  ERSY5
        B    3LPROC
```



```
6LPROCSF   8LPROC
        SF   9LPROC
        CM   LSYMB,34,10
        BNH  7LPROC
        CM   LSYMB,48,10
        BH   7LPROC
        BL   *+48
        TFM  TYPE,2,1011
        TFM  TPRIM,2,1011
        B    9LPROC
        CM   LSYMB,47,10
        BNE  *+48
        TFM  TYPE,4,10
        TFM  TPRIM,4,10
        B    9LPROC
        CM   LSYMB,41,10
        BH   7LPROC
        BL   *+60
        TFM  TYPE,9,10
        TFM  TPRIM,9,10
        CF   9LPROC
        B    9LPROC
        CM   LSYMB,37,10
        BH   BRT
        BL   *+48
        TFM  TYPE,1,10
        TFM  TPRIM,1,10
        B    8LPROC
        CM   LSYMB,36,10
        BL   *+48
        TFM  TYPE,0,10
        TFM  TPRIM,0,10
        B    8LPROC
        TFM  TYPE,2,10
        TFM  TPRIM,2,10
        CF   9LPROC
        B    9LPROC
BRT     CM   LSYMB,39,10
        BE   ERSY6
        BH   *+48
        TFM  TYPE,3,10
        TFM  TPRIM,3,10
        B    8LPROC
        TFM  TYPE,0,10
        TFM  TPRIM,5,1011
        CF   9LPROC
        B    9LPROC
7LPROCSM   99,4,10
        S    MEMSE,99,11
        SM   MEMSE,8,10
        B    3LPROC+48
8LPROCBTM  DECAL,00,10
        CM   MEMSE,3,610
        BE   *+36
```

```
        SM   MEMSE,2,10
        B    9LPROC
        BTM  DECAL,00,10
        CM   MEMSE,40,610
        BNH  ERSY4
        CM   MEMSE,69,610
        BH   ERSY4
        BTM  SSP3
        AM   99,2,10
        TF   LSYMB,99,11
        CM   LSYMB,41,10
        BNE  *+48
        TFM  TPRIM,8,10
        CF   8LPROC
        B    9LPROC
        CM   LSYMB,40,10
        BNE  ERSY7
        CF   9LPROC
        CM   TYPE,0,10
        BNE  *+36
        TFM  TPRIM,5,1011
        B    9LPROC
        CM   TYPE,1,10
        BNE  *+36
        TFM  TPRIM,5,10
        B    9LPROC
        TFM  TPRIM,6,1011
        B    9LPROC
9LPROCBTM  SSP7,*+12
        BTM  LCONS,00,10
        AM   LTAB1,11,10
        CM   LTAB1,4,610
        BNE  *+36
        BNF  ERSY8,9LPROC
        BNF  PFFP,8LPROC
        AM   LTAB1,11,10
        TD   LTAB1,TPRIM,6
10PROCAM   LTAB1,6,10
        TD   LTAB1,TYPE,6
        AM   MEMSE,2,10
        CM   MEMSE,23,610
        BE   9LPROC
        B    LSPEC
PFFP    AM   LTAB1,11,10
        TD   LTAB1,TYPE,6
        B    10PROC
        DORG*+2
LCONS   TF   LTAB1,LTAB
        SM   LTAB1,31,10
        AM   ATENT2-12,30,10
        C    ATENT2,LTAB1,11
        BNE  *+24
        BB
```



```
        SM   LTAB1,44,10
        C    LTAB1,IDPRO
        BH   LCONS+36
        B    ERSY9
ERSY1   BTM  IMPER,11,9
ERSY2   BTM  IMPER,12,9
ERSY3   BTM  IMPER,13,9
ERSY4   BTM  IMPER,14,9
ERSY5   BTM  IMPER,15,9
ERSY6   BTM  IMPER,16,9
ERSY7   BTM  IMPER,17,9
ERSY8   BTM  IMPER,18,9
ERSY9   BTM  IMPER,19,9
ERVAL   BTM  IMPER,20,9,

******CONSTANTES POUR CONSTRUCTIONS D'ORDRES

BTMIN   DSC  7,1799995
        DC   6,00012'
TFM3    DSC  25,1690000000001689995000000'
AD1     DSC  7,-2177773
        DC   6,90000'
TFIND   DSC  6,267777
        DC   5,37777
        DC   7,4117777
        DC   1,3
        DC   6,90000'
TF1     DSC  25,2600000000002599999499995'
BT1     DSC  17,279999900000000002
        DC   14,9000000000000'
BT2     DSC  12,279999800000
        DC   19,0000290000000000000'
TF2     DS   ,ZPREP1
B1      DS   ,ZORS
TFRE    DSC  6,167777
        DC   16,5000001277771000
        DC   3,20'
TFTA    DSC  6,267777
        DC   7,700000'
TFLI    DSC  10,1177771000
        DC   9,-202677771
        DC   4,9000
        DC   11,01290000000
        DC   13,0526666669000
        DC   7,0496666
        DC   3,60'
BTMRP   DSC  10,1799997000
        DC   11,0149000000'
BABS    DSC  10,1799994000
        DC   3,10'
B4      DSC  9,49660660'
TF3     DSC  9,49999900'
TFM2    DSC  13,169000000000'
```

```
B2     DSC 9,49999930'
BTMON  DSC 13,179999200000'
BTPAP  DSC 21,279999400018000000010'
BTEX   DSC 12,278989800038
       DC  29,000400000000000000000000000000'
BTAIG  DSC 22,16900000000001787878000
       DC  3,10'
PTRAV  DS  5
WORK   DS  5
NORD   DS  5
NPAR   DS  3
REGRESDS   2
TRAV1  DS  5
TRAV2  DS  5
ADRES  DS  5
TRAVA  DS  5

******     COMPILATION DES PROCEDURES

******     DEUXIEME PASSAGE

PDCL   CM  MEMSO,41,610
       BNE PREM
       TF  CPTR3,CPTR2
       SM  CPTR3,8,10
       C   ATENT2,CPTR3,11
       BE  *+60
       SM  CPTR3,22,10
       CM  CPTR3,C13
       BH  *-48
       BTM IMPER,35,9
       AM  CPTR3,16,10
       MM  CPTR3,17,61011
       AM  99,89988
       TF  MRINT,99
       BTM SPP1,0,10
       SM  CPTR3,29,10
       SM  MEMSV,7,10
       TF  MEMSV,CPTR3,6
       SM  REMAN,5,10
       SM  MEMSV,6,10
       TF  MEMSV,REMAN,6
       TF  CALADR,CH0B
       A   CALADR,MPG0B
       SM  CALADR,C3
       BT  PFOSI,CALADR
       AM  CPTR3,18,10
       TF  CPTR3,REMAN,6
       SM  REMAN,5,10
       AM  CPTR3,11,10
       TF  ADER,CPTR3,11
       AM  CPTR3,15,10
```



```
       TF  TRAVA,CPTR3
       TFM REGRES,0,10
       TF  CT,TRAVA
       BTM PF0,0,10
       TF  CPTR3,TRAVA
       TD  ZONE,TRAVA,11
       CM  ADER,0,9
       BH  PAP
1PDCL  CM  MEMSE,30,610
       BE  *+60
       BTM DECAL,03,10
       BNE 1PDCL
       BTM DECAL,23,10
       BNE 1PDCL
       BTM DECAL,03,10
       BE  2PDCL
       SM  MEMSE,2,10
       TFM MEMSE,30,610
       B   VRBCL
2PDCL  TF  WORK,MEMSE
       AM  MEMSE,1,10
       SF  MEMSE,,6
       AM  MEMSE,1,10
       BTM SSP3,0,10
       AM  99,2,10
       CM  99,35,610
       BL  3PDCL
       CM  99,42,610
       BL  1PDCL
       CM  99,51,610
       BE  1PDCL
       CM  99,47,610
       BE  1PDCL
       CM  99,48,610
       BE  1PDCL
3PDCL  TF  MEMSE,WORK
       B   2PDCL-36
PREM   SM  MEMSO,2,10
       CM  MEMSO,39,610
       BE  PRIOR
       AM  CPTR2,14,10
       TF  CPTR2,ATENT2,6
       AM  CPTR2,6,10
       TR  CPTR2,PR2,6
       AM  CPTR2,7,10
       TF  CPTR2,CTNIV,6
       AM  CPTR2,15,10
       SM  MEMSV,6,10
       TF  MEMSV,CPTR2,6
       AM  CPTR2,2,10
       AM  MEMSO,2,10
       CM  MEMSO,40,610
       BNE PRIOR
```

```
        SF   PREM
        B    PRIOR
PAP     AM   ZONE,3,10
        TD   TRAVA,ZONE
        AM   CPTR2,14,10
        AM   CT,13,10
        TF   CPTR2,CT,611
        AM   CT,5,10
        AM   CPTR2,5,10
        TF   CPTR2,CT,6
        AM   CPTR2,1,10
        TDM  CPTR2,9,611
        AM   CPTR2,2,10
        AM   CT,26,10
        SM   ADER,1,10
        AM   TRAVA,40,10
        TF   ADRES,TRAVA,11
        AM   TRAVA,4,10
        AM   CPTR3,35,10
        TD   ZONE,CPTR3,11
        CM   ZONE,2,10
        BNE  1PAP
        BTM  PAFOR,0,10
        SM   ADRES,5,10
        TR   C4,TF1
        TF   C4+6,ADRES
        TF   C4+11,MRINT
        A    C4+18,ADRES
        A    C4+23,MRINT
        TR   MPG0B,C4,6
        AM   MPG0B,24,10
        B    1PDCL-60
1PAP    CF   ZONE
        CM   ZONE,5,10
        BE   *+36
        CM   ZONE,6,10
        BNE  2PAP
        BTM  PAFOR,0,10
        TR   C4,TF2
        TF   C4+6,ADRES
        TF   C4+11,MRINT
        TR   MPG0B,C4,6
        AM   MPG0B,12,10
        B    1PDCL-60
2PAP    SM   CPTR3,11,10
        CM   CPTR3,4,610
        BNE  1PDCL-48
        AM   CPTR3,11,10
        CM   ZONE,4,10
        BE   PAP+240
        BTM  PAFOR,0,10
        TR   C4,TF2
        TF   C4+6,ADRES
```



```
        TF   C4+11,MRINT
        SF   C4+11
        TR   MPG0B,C4,6
        AM   MPG0B,12,10
        B    1PDCL-60
        DORG*-2
SSP1    TR   C4,81
        TF   C4+6,REMAN
        SF   C4+6
        TR   MPG0B,C4,6
        AM   MPG0B,8,10
        BB
        DORG*-7
PAFOR   TR   C4,BT1
        TF   CALADR,CH0B
        A    CALADR,MPG0B
        SM   CALADR,C3-28
        TF   C4+11,CALADR
        A    C4+16,CALADR
        SF   C4+12
        S    C4+21,MRINT
        CF   C4+17
        TF   C4+26,ADRES
        CF   C4+22
        A    C4+28,REGRES
        TR   MPG0B,C4,6
        AM   MPG0B,30,10
        BB
AVSAN   BTM  PF0,0,10
        BD   1ASAN,CPTR3,11
        BT   PORTE,CPTR2
        TR   C4,TFTA
        TF   C4+11,PORTE-1
        TR   MPG0B,C4,6
        AM   MPG0B,12,10
1ASAN   TF   REGRES,CTNIV
        WATYA1
        SM   CPTR3,4,10
        S    REGRES,CPTR3,11
        TF   CALADR,CH0B
        A    CALADR,MPG0B
        SM   CALADR,C3
        BNF  2ASAN,AVSAN
        CF   VALOC
        AM   CALADR,110,9
        TR   C4,TFRE
        TF   C4+11,CALADR
        TR   MPG0B,C4,6
        AM   MPG0B,24,10
        AM   CPTR3,14,10
        TF   ADRES,CPTR3,11
        SM   ADRES,5,10
        SM   CALADR,58,10
```

```
        BTM VALOC,0,10
        TR  MPGOB,TFLI,6
        AM  MPGOB,56,10
        CF  AVSAN
        AM  CPTR3,1,10
        B   3ASAN
VALOC   TR  C4,BT2
        TF  C4+11,CALADR
        A   C4+16,CALADR
        BNF *+36,VALOC
        TF  C4+21,MRINT
        CF  C4+17
        A   C4+26,ADRES
        A   C4+28,REGRES
        MF  C4+28,CVAR-2
        TR  MPGOB,C4,6
        AM  MPGOB,30,10
        BB
        DORG*-9
2ASAN   AM  CALADR,32,10
        TR  C4,TFRE
        TF  C4+11,CALADR
        TR  C4+12,BTMRP
        A   C4+23,REGRES
        SM  CPTR3,8,10
        TF  C4+30,CPTR3,11
        SF  C4+6
        SF  C4+30
        TR  MPGOB,C4,6
        AM  MPGOB,32,10
        AM  CPTR3,23,10
3ASAN   TD  ZONE,CPTR3,11
        CM  ZONE,9,10
        BNE *+36
        AM  MEMSV,7,10
        B   AVSAN-1,,6
        TF  MEMSV,MRINT,6
        AM  MEMSV,1,10
        TD  MEMSV,ZONE,6
        SF  MEMSV,,6
        SM  MRINT,1,10
        CM  ZONE,3,10
        BE  AVSAN-1,,6
        SM  MRINT,11,10
        B   AVSAN-1,,6
PROC    AM  MEMSV,1,10
        TF  CPTR3,MEMSV,11
        AM  CPTR3,29,10
        MM  CPTR3,44,610
        TF  PTRAV,MEMSV,11
        A   PTRAV,99
        C   PTRAV,MEMSV,11
        BE  *+60
```



```
        AM  PTRAV,1,10
        AM  PTRAV,30,610
        SM  PTRAV,45,10
        B   *-60
        AM  CPTR3,1,10
        BNF *+24,CPTR3,11
        TDM CPTR3,1,6
        SM  MEMSV,6,10
        WATYA3
        AM  CALADR,12,10
        AM  MEMSV,5,610
        BT  PFOSI,CALADR
        AM  MEMSV,18,10
        SM  CTNIV,1,10
        TR  MPGOB,BABS,6
        AM  MPGOB,12,10
        B   SFIN1
FICH7   BTM DECAL,40,10 ,
        SM  MEMSV,5,10,
        CM  MEMSE,40,610
        BH  *+24
        BTM FINEF,SUF17
        BTM DECAL,24,10
        BNE *-12
        BTM FINEF,SUF16
SUF17   TF  TRAV1,MPGOB
        A   TRAV1,CHOB
        SM  TRAV1,C3
        MM  NPAR,44,1011
        SM  99,6,10
        TF  TRAV2,99
        A   TRAV2,NORD
        BNF 1SUF17,TRAV2,11
        TF  TRAV2,NPAR,6
        B   2SUF17
1SUF17C TRAV2,NPAR,6
        BNE ERSY15
2SUF17BNF *+24,MEMSV,11
        SF  AVSAN
        CF  MEMSV,,6
        TF  MRINT,MEMSV,11
        AM  MEMSV,6,10
        SM  CTNIV,1,10
        BTM SPP2,0,10
        MM  NPAR,17,1011
        AM  99,90005
        TF  TRAV2,99
        TR  C4,ZADR
        TF  C4+6,TRAV2
        TF  C4+11,MEMSV,11
        TR  MPGOB,C4,6
        AM  MPGOB,12,10
        BTM PFO,0,10
```

∑ DE PARENTHESE DE PROCEDURE

```
        AM   MEMSV,6,10
        AM   TRAV2,17,10
        CM   TRAV2,90005
        BNE  *-108
        TF   NPAR,MEMSV,11
        AM   MEMSV,6,10
        TF   NORD,MEMSV,11
        AM   MEMSV,6,10
        BT   PFOS1,TRAV1
        AM   MEMSV,6,10
        TF   CPTR3,MEMSV,11
        BTM  AVSAN,FIFI7
FIFI7   SM   MEMSE,2,10
        B    DEBPAR+12
SUFI6   AM   NORD,44,10,          ∑' DE PARENTHESE DE PROCEDURE
        WATYA2,,,                    PAR RAPPORT A  ,
        AM   NPAR,1,10
        TF   WORK,MEMSV,11
        TF   CALADR,CH0B
        A    CALADR,MPG0B
        SM   CALADR,C3
        TF   MEMSV,CALADR,6
        SM   MEMSV,6,10
        TF   MEMSV,WORD,6
        TFM  MRINT,90000
        AM   MEMS0,2,10
        B    BOUCLE
FINEF   AM   MEMSV,6,10,          FIN DU TRAITEMENT DES
        WATYA4,,,                    PARAMETRES EFFECTIFS
        BNF  1FINEF,MEMSV,11
        SM   MEMSV,6,10
        CM   MEMSV,99999,67
        BE   ABV+12
        B    CASCAD
1FINEFTF CT,NORD
        SM   CT,11,10
        CM   CT,5,610
        BE   *+36
        CM   CT,4,610
        BNE  ERSY15
        SM   MEMSV,6,10
        CM   MEMSV,99999,67
        BE   ABV+12
        TD   ZONE,MEMSV,11
        CF   ZONE
        CM   ZONE,5,10
        BNE  *+24
        TD   ZONE,MEMSV,11
        TD   TPRIM,NORD,11
        C    TPRIM,ZONE
        BNE  ERSY16
CASCADTD ZONE,MEMSV,11
        CF   ZONE
```



```
        CM   ZONE,4,10
        BE   FINETI
        BH   FINTAB
        CM   ZONE,2,10
        BE   FINAIG
        BNF  2FINEF,MEMSV,11
        TR   MPG0B,TF3,6
        AM   MPG0B,8,10
        B    ABV+12
2FINEFSM MEMSV,1,10
        CM   MEMSV,90000 ,7611
        BE   ABV
        TR   C4,TFM2
        TF   C4+11,MEMSV,11
        TR   MPG0B,C4,6
        AM   MPG0B,12,10
        B    ABV
FINAIGSM MEMSV,1,10
        BNF  ABV,MEMSV,11
        TR   C4,BTAIG
        TF   C4+11,MEMSV,11
        AM   MEMSV,3,10
        A    C4+23,CTNIV
        S    C4+23,MEMSV,11
        TR   MPG0B,C4,6
        AM   MPG0B,24,10
        B    ABV
FINETIBNF *+24,MEMSV,11
        B    ABV+12
        SM   MEMSV,1,10
        TR   C4,TFM3
        TF   C4+11,MEMSV,11
        TF   C4+23,CPTR3,11
        TR   MPG0B,C4,6
        AM   MPG0B,24,10
ABV     AM   MEMSV,1,10
        TR   MPG0B,B2,6
        AM   MPG0B,8,10
        BTM  PF0,0,10
        AM   MEMSV,6,10
        B    FINEF-1,,6
FINTABSM MEMSV,1,10
        CM   MEMSV,90000,6711
        BNE  3FINEF
        BNF  3FINEF+48,CVAR
        B    ABV
3FINEFTR C4,TFM2
        TF   C4+11,MEMSV,11
        TR   MPG0B,C4,6
        AM   MPG0B,12,10
        AM   MEMSV,7,10
        BNF  *+36,MEMSV,11
        SM   MEMSV,6,10
```

```
        B    ABV+12
        SM   MEMSV,6,10
        TF   CT,NORD
        SM   CT,14,10
        BNF  4FINEF,CT,11
        TF   CT,CPTR3,611
        B    ABV+12
4FINEFC CT,CPTR3,611
        BE   ABV+12
        BTM  IMPER,27,9
```



```
******  TRAITEMENT DES IDENTIFICATEURS SPECIAUX
        DES PROCEDURES A LEUR APPEL


PNDCL AM   CPTR3,22,10
      TF   ADRES,CPTR3,11
      SM   CPTR3,14,10
      TF   REGRES,CTNIV
      S    REGRES,CPTR3,11
      SM   CPTR3,8,10
      TF   WORK,CPTR3,11
      AM   CPTR3,6,10
      CM   CPTR3,1,61011
      BNE  6NDCL
      CM   MEMSE,25,610
      BNE  1NDCL
      AM   CPTR3,17,10
      TFM  MEMSV,90000,67
      AM   MEMSV,1,10
      TD   MEMSV,CPTR3,611
      B    PRIOR
1NDCL CM   MEMSE,28,610
      BE   POIN3
      AM   CPTR3,5,10
      SM   MEMSO,2,10
      TF   LSYMB,MEMSO,11
      AM   MEMSO,2,10
      CM   LSYMB,46,10
      BNE  2NDCL
      CM   MEMSO,28,610
      BNE  2NDCL
      CM   MEMSE,42,610
      BE   3NDCL
      CM   MEMSE,45,610
      BE   3NDCL
2NDCL CM   CPTR3,0,69
      BE   POIN1
      BTM  IMPER,22,9
      SM   MEMSV,7,10
POIN1 AM   CPTR3,1,10
      BTM  SPP2,0,10
      BTM  PF0,0,10
      BTM  AVSAN,PRIOR
      DORG*+2
SPP2  TR   C4,BTMON
      TF   C4+11,CHOB
      A    C4+11,MPGOB
      SM   C4+11,C3-12
```

```
        TR   C4+12,TFIND
        S    C4+35,MRINT
        TR   MPGOB,C4,6
        AM   MPGOB,36,10
        CM   CTNIV,0,10
        BE   *+24
        BB
        TR   C4+12,BTMIN
        TF   C4+23,C4+11
        AM   C4+23,36,10
        TR   MPGOB,C4+12,6
        AM   MPGOB,12,10
        BB
        DORG*-7
3NDCL   TD   TPRIM,NORD,11
        TF   TRAVA,NORD
        SM   TRAVA,6,10
        TF   CT,NORD
        AM   CT,6,10
        AM   MEMSV,7,10
        CM   CPTR3,0,69
        BNE  4NDCL
        AM   CPTR3,6,10
        TD   ZONE,CPTR3,11
        CM   ZONE,8,10
        BH   4NDCL+12
        BNF  *+36,MEMSV,11
        SM   CPTR3,6,10
        B    POIN1-12
        AM   CPTR3,6,10
        TD   TYPE,CPTR3,11
        C    TYPE,TPRIM
        BNE  *+36
        SM   CPTR3,12,10
        B    POIN1-12
        CM   TPRIM,8,10
        BNE  ERSY15
        BNF  ERSY15,TRAVA,11
        TD   NORD,CT,611
        B    *-72
4NDCL   AM   CPTR3,6,10
        BNF  *+24,MEMSV,11
        B    5NDCL
        CM   TPRIM,8,10
        BL   ERSY15
        AM   CPTR3,6,10
        TD   ZONE,CPTR3,11
        TD   TYPE,CT,11
        C    TYPE,ZONE
        BNE  ERSY15
        SM   CPTR3,12,10
        BNF  *+36,TRAVA,11
        TF   TRAVA,CPTR3,611
```



```
        B    5NDCL
        C    TRAVA,CPTR3,611
        BNE  ERSY15
5NDCL   TR   C4,BTPAP
        A    C4+11,CHOB
        A    C4+11,MPGOB
        SM   C4+11,C3
        TF   C4+16,WORK
        A    C4+18,REGRES
        TR   MPGOB,C4,6
        AM   MPGOB,20,10
        SM   MEMSV,6,10
        TFM  MEMSV,99999,67
        B    PRIOR
POIN3   AM   CPTR3,6,10
        TF   MEMSV,CPTR3,6
        TR   C4,B1
        TF   C4+6,REMAN
        SF   C4+6
        TR   MPGOB,C4,6
        AM   MPGOB,8,10
        SM   MEMSV,6,10
        TF   MEMSV,REMAN,6
        SM   REMAN,5,10
        AM   CPTR3,49,10
        SM   MEMSV,6,10
        TF   MEMSV,NORD,6
        SM   MEMSV,6,10
        TF   MEMSV,NPAR,6
        TFM  NPAR,1,9
        TF   NORD,CPTR3
        AM   CTNIV,1,10
        SM   MEMSV,6,10
        TF   MEMSV,CHOB,6
        A    MEMSV,MPGOB,6
        SM   MEMSV,C3,6
        SM   MEMSV,6,10
        TF   MEMSV,MRINT,6
        MF   MEMSV,14NDCL,6
        TFM  MRINT,90000 ,7
        AM   MEMSO,4,10
        TFM  MEMSO,4600,68
        TFM  MEMSO,28,610
        BTM  SOSV,40,10
        B    BOUCLE
        DORG*+2
SOSV    TF   PRIOR-1,MEMSV
        S    PRIOR-1,MEMSO
        S    PRIOR-1,SOSV-1
        BNF  *+24,PRIOR-1
        BTM  BUTOIR,40,9
```

```
******    COMPILATION DES PARAMETRES FORMELS


6NDCL TFM CVAR-1,POIN11
      CM  CPTR3,5,610
      BE  7NDCL
      AM  CPTR3,17,10
      TD  ZONE,CPTR3,11
      MF  6NDCL,ZONE
      CM  ZONE,6,10
      BE  *+36
      CM  ZONE,5,10
      BNE *+36
      SM  CPTR3,20,10
      BTM CVAR,POIN11
      CM  ZONE,4,10
      BNE ADRES,5,10
      AM  REGRES,50,10
      TFM CVAR-1,POIN10
      SF  CVAR-2
      B   CVAR
7NDCL AM  CPTR3,11,10
      TD  ZONE,CPTR3,11
      MF  6NDCL,ZONE
      CM  ZONE,6,10
      BE  *+36
      CM  ZONE,5,10
      BNE *+48
      SM  CPTR3,14,10
      SF  CVAR
      B   7NDCL-24
      CM  ZONE,2,10
      BE  8NDCL
      SM  MEMS0,2,10
      TF  LSYMB,MEMS0,11
      AM  MEMS0,2,10
      CM  LSYMB,46,10
      BNE 9NDCL
      CM  MEMS0,28,610
      BNE 9NDCL
      TD  TPRIM,NORD,11
      CM  MEMSE,45,610
      BE  *+36
      CM  MEMSE,42,610
      BNE 9NDCL
      BNF 10NDCL,MEMSV,11
12NDCLBTM PAFOR,0,10
      AM  MEMSV,1,10
      TFM MEMSV,99999,67
      CM  TPRIM,8,10
      BL  *+36
      TR  MPGOB,B4,6
      AM  MPGOB,8,10
```



```
      B   PRIOR
10NDCLTF  CT,NORD
      AM  CT,6,10
      TD  TYPE,CT,11
      AM  CPTR3,6,10
      TD  CP1,CPTR3,11
      C   TPRIM,ZONE
      BNE *+48
11NDCLC   TYPE,CP1
      BNE ERSY16
      B   12NDCL
      CM  TPRIM,8,10
      BE  11NDCL
      CM  ZONE,8,10
      BE  11NDCL
      B   ERSY16
8NDCL TFM CVAR-1,POIN12
      SM  ADRES,5,10
      AM  REGRES,50,10
      B   CVAR
POIN12TF  MEMSV,MRINT,6
      AM  MEMSV,1,10
      TDM MEMSV,4,6
      CM  MEMSE,24,610
      BE  *+24
      TDM MEMSV,2,6
      SM  MRINT,10,10
      B   PRIOR
9NDCL CM  ZONE,8,10
      BNL 13NDCL
      BTM PAFOR,0,10
      CM  ZONE,4,10
      BNE POIN11
POIN10TF  MEMSV,MRINT,6
      SF  MEMSV,,6
      AM  MEMSV,1,10
      TDM MEMSV,4,611
      SM  CPTR3,9,10
      SM  MRINT,7,10
      B   PRIOR
POIN11TF  MEMSV,MRINT,6
      SF  MEMSV, 6
      AM  MEMSV,1,10
      TD  MEMSV,ZONE,6
      MF  MEMSV,6NDCL,6
      SM  MRINT,5,10
      B   PRIOR
13NDCLCM  MEMSE,28,610
      BE  14NDCL
      TF  CT,CPTR3,
      AM  CT,6,10
      CM  ZONE,8,10
      BNE *+60
```

```
       TD   ZONE,CT,11
       TD   CPTR3,CT,611
       BTM  PAFOR,0,10
       B    POIN11
       SF   AVSAN
       SM   CPTR3,6,10
       BNF  2NDCL,CPTR3,11
       TFM  CPTR3,0,69
       BTM  PAFOR,0,10
       B    POIN1
14NDCLSF    14NDCL
       SM   CPTR3,6,10
       BNF  *+24,CPTR3,11
       BTM  PAFOR,0,10
       SM   CPTR3,5,10
       B    POIN3
       DORG*+6
CVAR   TF   CALADR,MPGOB
       A    CALADR,CHOB
       SM   CALADR,C3-28
       SF   VALOC
       BTM  VALOC,0,10
       B    CVAR-1,,6
ERSY16BTM   IMPER,26,9
ERSY15BTM   IMPER,25,9
```



```
*
******    EXECUTION DES PROCEDURES

PILE   DC   35,00020000200002000020000200002000020
TR20   DS   5
LIMIN  DS   5
WORK   DS   5
WORK1  DS   5
DOUZE  DC   10,00012000012
TRAVA  DS   5
LIAC   DS   5
LIVI   DS   5
INAC   DS   5
INVI   DS   5
RETAC  DS   5
REFU   DS   5
TABLO  DS   5
ABC    DSS  100,1600
ERIND  BTM  BUTOIR,5701,8
SAMRT  BTM  BUTOIR,5702,8
ZONEG  DC   2,-11
ZSUP   DC   3,-101
       DORG*+18
REGRE  TF   REGRE-8,REGRE-8,11
       SM   REGRE-1,1,10
       SF   REGRE-12
       BNL  REGRE
       AM   REGRE-8,5,10
       B    REGRE-3,,6
       DORG*+18
PAFOR  SF   PAFOR-2
       SF   PAFOR-7
       SF   PAFOR-12
       TFM  REGRE-3,*+36
       TFM  REGRE-8,LIAC
       BT   REGRE,PAFOR-1
       BTM  MONTE,*+12
       TF   RETAC,PAFOR-13,6
       TF   INAC,INVI,611
       A    INAC,PAFOR-8,6
       TF   LIAC,REGRE-8,6
       SM   LIAC,25,610
       S    PAFOR-3,REGRE-8,11
       TF   TABLO,PAFOR-3,6
       SM   TABLO,5,610
       BT   INDEX,PAFOR-3,6
ABRET  S    TABLO,PILE
       TF   WORK,REFU,11
       B    WORK,,6
       DORG*+12
REPRO  TFM  REGRE-3,*+48
```

```
        TFM REGRE-8,LIVI
        TF  REGRE-1,REPRO-1
        B   REGRE
        TF  LIAC,REGRE-8,6
        SM  LIAC,5,610
        BB
        DORG*+10
VALOC   SF  VALOC-2
        SF  VALOC-7
        SF  VALOC-12
        TFM REGRE-3,*+96
        TFM REGRE-8,LIAC
        MF  VALOC,VALOC-1
        CM  VALOC-1,50,10
        BL  *+36
        SM  VALOC-1,50,10
        SF  REGRE
        BT  REGRE,VALOC-1
        S   VALOC-3,REGRE-8,11
        MF  VALOC-3,VALOC
        SF  VALOC-8,INAC,11
        TF  VALOC-8,VALOC-3,6
        BNF *+60,REGRE
        SM  VALOC-8,5,10
        CF  VALOC-3
        SM  VALOC-3,5,10
        TF  VALOC-8,VALOC-3,611
        BT  INDEX,VALOC-13
        DORG*+12
PAPRO   SF  PAPRO-2
        TFM REGRE-8,LIAC
        TFM REGRE-3,*+24
        BT  REGRE,PAPRO-1
        TF  WORK,TABLO,11
        TF  WORK,REGRE-8,6
        SM  WORK,5,610
        SM  WORK,5,10
        TF  WORK,PAPRO-3,611
        BTM INDEX,ABRET
        DORG*+2
ABSOR   S   TABLO,PILE
        TF  WORK1,REFU,11
        C   LIAC,LIMIN
        BE  WORK1,,6
        BT  INDEX,WORK1,
        DORG*+6
MONTE   A   TABLO,PILE
        C   LIAC,LIMAX
        BH  *+24
        BT  INDEX,MONTE-1
        BTM BUTOIR,5703,8
        DORG*+10
```



```
INDEX   TF  INDEX-6,INDEX-1
        CF  INDEX-5
        AM  INDEX-6,12,10
        B   AGAIN
        A   INDEX-1,DOUZE
        DORG*+12
AGAIN   TD  TRAVA,INDEX-6,11
        TDM INDEX-6,,6
        DC  1,',*
        TR  AGAIN-12,INDEX-1,11
        TDM AGAIN,2
        TD  INDEX-6,TRAVA,6
INDOU   TFM *+30,ABC+9,711
        TD  *+17,AGAIN-12
        B
IND1    SF  AGAIN-5
        BTM TRAD,AGAIN-1
IND2    SF  AGAIN-10
        BTM TRAD,AGAIN-6
        CM  AGAIN-7,5003,8
        BH  *+36
        CM  AGAIN-7,4998,8
        BH  SAMRT
        B   AGAIN-24
IND3    SF  AGAIN-10
        TFM *+30,ABC+4,711
        TD  *+17,AGAIN-11
        B
        DORG*+6
TRAD    BNF 1TRAD,TRAD-1,11
        CM  TRAD-1,60000,6711
        BH  *+24
        A   TRAD-1,INAC,611
        TF  TR20,TRAD-1,11
        CF  TR20
        TF  *+59,TR20
        SM  TR20,4,10
        MF  TRAD,TR20,11
        SF  TR20,,6
        TF  TRAD-1,,6
        MF  TR20,TRAD,6
        B   TRAD
1TRAD   CM  TRAD-1,60000,67
        BL  *+24
        S   TRAD-1,INAC,611
        BB  INDFS+44
BB      DS  ,*-10
        DORG*+6
IND4    TDM AGAIN-11,9
        B   AGAIN-24
IND5    SF  AGAIN-5
        BTM TRAD,AGAIN-1
        B   VABRAN-60
```

```
IND16 TFM ZONE,0,8
      BH  *+48
      BL  *+36

ZONE  DS  ,*
      TF  ZONE,ZONEG
      B   *+24,0,10
TR22  DS  ,*
      TF  ZONE,ZSUP
      SM  AGAIN-11,3,10
      TD  TR22,AGAIN-3
      TFM AGAIN-1,ZONE-3
      A   AGAIN-1,TR22
      SF  AGAIN-10
      BTM TRAD,AGAIN-6
      TF  TRAD-1,AGAIN-6
      TFM AGAIN-6,VABRAN
      B   AGAIN-24
VABRANBT  INDEX,TRAD-1
IND7  BTM TRAD,AGAIN-6
      BNF 1IND17,AGAIN-5
      BNF *+24,AGAIN-2
      B   *+36
      BTM TRAD,AGAIN-1
      TF  AGAIN-1,AGAIN-1,11
      CM  AGAIN-1,0
      BE  1IND17
      TFM REGRE-8,LIAC
      TFM REGRE-3,*+24
      BT  REGRE,AGAIN-1
      SM  REGRE-8,5,10
      C   REGRE-8,LIMIN
      BNE *+36

      B   AGAIN-6,,6
      S   REGRE-8,LIAC
      A   LIAC,REGRE-8
      A   LIVI,REGRE-8
      A   REFU,REGRE-8
      A   RETAC,REGRE-8
      A   INAC,REGRE-8
      A   INVI,REGRE-8
      A   TABLO,REGRE-8
1IND17BT  INDEX,AGAIN-6
      DORG*+6
INDAX C   LIAC,LIMIN
      BE  INDAX-1,,6
      BT  INDEX,INDAX
PP    TF  WORK1,TABLO,11
      TF  WORK1,90000,6
      TF  90000,WORK1
      B   ABRET
```



```
*
******    COMPILATION DES AIGUILLAGES


AFAIG SM  MEMSV,1,10,
      TR  C4,ZORS,,
      TF  C4+6,MEMSV,11
      TR  MPGOB,C4,6
      AM  MPGOB,8,10
      TF  MRINT1,MRINT
      AM  CTNIV,1,10
1AFAIGSM  MEMSV,6,10
      TF  MEMSV,CHOB,6
      A   MEMSV,MPGOB,6
      SM  MEMSV,C3,67
      AM  MEMSV,1,10
      TDM MEMSV,4,6
      TFM MRINT,90000
      B   BOUCLE
VGAIG CF  VGAIG,,,
      BNF 2VGAIG,FLEG,,
      BNF *+36,FL2RP
      AM  MRINT,7,10
      B   3VGAIG
      TR  C4,TFM3
      SM  MEMSV,1,10
      TF  C4+11,MEMSV,11,
      TF  C4+23,CPTR3,11
      TR  MPGOB,C4,6,
      AM  MPGOB,24,10
      AM  MEMSV,1,10
3VGAIGTR  MPGOB,BABS,6
      AM  MPGOB,12,10
      AM  MEMSV,15,10
      BNF 1AFAIG,VGAIG
      B   2AIG
2VGAIGBTM IMPER,42,9
AIG   SF  VGAIG,,,
      SM  MEMSV,5,10
      S   CALADR,MPGOB
      B   VGAIG+12
2AIG  SM  MEMSV,6,10
      TFM C8,0,10
      SM  MPGOB,1,10
1AIG  AM  MEMSV,6,10
      BNF *+24,MEMSV,11,
      B   *+60
```

$\sum$:= PAR RAPPORT A AIGUILLAGE

$\sum$ DE , PAR RAPPORT A AIGUILLAGE

$\sum$ D'AIGUILLAGE

```
        AM   C8,1,10
        AM   MPG0B,5,10
        TF   MPG0B,MEMSV,611
        B    1AIG
        AM   MPG0B,2,10
        TF   MPG0B,C8,6
        TF   MRINT,MRINT1
        SM   CTNIV,1,10
        AM   MPG0B,1,10
        A    CALADR,MPG0B
        MM   CALADR,5,10
        BD   4AIG,99
3AIG    TD   MPG0B,C2+41,6,
        BT   PFOS!,CALADR
        SM   MEMS0,2,10
        AM   MEMSV,10,10
        B    PRIOR
4AIG    TDM  MPG0B,0,611
        AM   MPG0B,1,10
        AM   CALADR,1,10
        B    3AIG
1CROCHCM C7,4,10
        BNE  CROCH
        TR   C4,BTEX
        A    C4+11,CALADR
        A    C4+16,CALADR
        SM   MEMSV,6,10
        TF   C4+38,MEMSV,11
        BNF  *+36,C4+38
        AM   MRINT,5,10
        B    *+36
        BNF  *+24,FL2RP
        AM   MRINT,12,10
        AM   MEMSV,6,10
        TF   C4+26,MEMSV,11
        BNF  *+60,C4+26
        CF   C4+26
        AM   MEMSV,3,10
        A    C4+21,MEMSV,11
        B    *+48
        AM   MRINT,5,10
        A    C4+21,MRINT
        AM   MRINT,5,10
        A    C4+31,MRINT
        A    C4+33,CTNIV
        CF   C4+22
        CF   C4+34
        TR   MPG0B,C4,6
        AM   MPG0B,40,10
        TF   MEMSV,MRINT,6
        AM   MEMSV,1,10
        TDM  MEMSV,4,611
        SM   MRINT,7,10
        SM   MEMS0,2,10
        CF   SYMBOL
        B    SYMBOL
```



```
*
******    EXECUTION DES AIGUILLAGES


ERAIG DAC 6,ER 40'
      DORG*+100
EXEC  DS  ,402
RCOV  DS  ,600
COV   DS  ,1000
EXAIG SF  EXAIG-22
      SF  EXAIG-5
      BTM TRAD, EXAIG-1
      SF  EXAIG-17
      SF  EXAIG-7
      SF  EXAIG-12
      SM  EXAIG-1,1,10
      BNF *+48,EXAIG-1,11
      AM  EXAIG-1,1,10
      BTFLCOV,EXAIG-1,11
      B   *+36
      AM  EXAIG-1,1,10
      TF  RCOV,EXAIG-1,11
      BTM TRAD,EXAIG-13
      BD  *+24,EXAIG-22
      B   *+24
      BTM TRAD,EXAIG-18
      A   TABLO,PILE
      TF  INAC,INV1,611
      S   INAC,EXAIG-8,6
      AM  INAC,90000,67
      TF  RETAC,EXAIG-23,6
      BD  1EX,EXAIG-22
      SF  EXAIG-19
      SF  EXAIG-18
      A   EXAIG-18,EXAIG-6
      AM  EXAIG-18,1,10
      BT  REPRO,EXAIG-18
      B   *+60
1EX   TF  LIAC,EXAIG-18,611
      TF  EXAIG-13,EXAIG-13,11
      BNF *-12,EXAIG-13
      CF  EXAIG-13
      TF  EXAIG-13,EXAIG-13,11
      SM  EXAIG-13,2,10
      BNF *+24,EXAIG-13,11
      AM  EXAIG-13,1,10
      C   RCOV,EXAIG-13,11
      BH  2EX
      CM  RCOV,0
      BE  2EX
      BNF *+24,RCOV
      B   2EX
      AM  EXAIG-13,3,10
```

```
      MM  RCOV,5,10
      SF  95
      S   EXAIG-13,99
      BT  INDEX,EXAIG-13,11
2EX   WATYERAIG
      CF  EXEC
      S   TABLO,PILE
      CM  EXAIG-6,0,10
      BE  EXAIG-23,6
      BT  INDEX,EXAIG-23
      DORG*+30
EFAIG TFM REGRE-8,LIAC
      TFM REGRE-3,*+36
      TF  REGRE-1,EFAIG-1
      B   REGRE
      TFM *+54,89995
      S   *+42,INAC,11
      TF  *+35,REGRE-8
      SM  *+23,5,10
      TFM
      BB
```

BIBLIOGRAPHIE

[1] P. NAUR (Editor) - Revised Report on the algorithmic language Algol 60 - Communication of the A. C. M. 6 (1963) pages 1 à 17

[2] C. PAIR - Séminaire de programmation - Centre de Calcul Automatique - NANCY (1963-1964)

[3] M. CUSEY - Construction d'un compilateur Algol - Thèse de troisième cycle de Mathématiques - Centre de Calcul Automatique - NANCY (1964)

[4] M. NIVAT - L. NOLIN - Sur un procédé de définition de la syntaxe Algol - Institut Blaise Pascal - PARIS (1963)

[5] C. PAIR - Etude de la notion de pile. Application à l'analyse syntaxique - Thèse d'Etat - NANCY (1965)

[6] J. M. LAPORTE - Traitement des tableaux Algol ; extension à Algol matriciel - Thèse de troisième cycle de Mathématiques - Centre de Calcul Automatique - NANCY (1964)

[7] J. ANDRE - Participation à la construction d'un compilateur Algol pour 1620 IBM - Thèse de troisième cycle de Mathématiques - Centre de Calcul Automatique - NANCY (1964)

[8] L. BOLLIET - N. GASTINEL - P. J. LAURENT - Un nouveau langage scientifique : Algol - Hermann - PARIS (1964)

[9] Communication personnelle de Monsieur C. PAIR (1965)

[10] Unités composantes du 1620 IBM - 10 1939 - Mai 1962

Vu et Approuvé
Nancy, le
Le Doyen de la Faculté
des Sciences de NANCY,

M. AUBRY

Vu et Permis d'imprimer
Nancy, le
le Recteur :
Président du Conseil de
l'Université,

P. IMBS