NCR ELLIOTT 4100

ALGOL

NCR

DIVISION ELECTRONIQUE

Reçu le
7-3-67

NCR ELLIOTT 4100

ALGOL

N.C.R. - SCIENTIFIQUE,
3, Rond-Point des Champs Elysées,
75 - PARIS 8e.

Tél. : 225-10-31

SOMMAIRE

		Page
Chapitre 1	INTRODUCTION	
	1.1. BUT	1-1
	1.2. SOMMAIRE	1-1
	1.3. CONFIGURATION	1-2
	1.4. MODE D'UTILISATION	1-2
	1.5. EFFICACITE	1-2
	1.6. FORME DE DISTRIBUTION	1-3
Chapitre 2	LES FONCTIONS	
	2.1. LA REPRESENTATION "HARDWARE"	2-1
	2.1.1. Caractères et symboles de base	2-1
	2.1.2. Représentation des fonctions standard	2-2
	2.1.3. Les Nombres	2-2
	2.1.3.1. Les entiers	2-2
	2.1.3.2. Les nombres réels	2-3
	2.2. COMPATIBILITE AVEC ALGOL 60	2-4

.../...

2.4. ADDITIONS AU LANGAGE 2-6

2.4.1. Les fonctions standard 2-6
2.4.2. Segments en langage d'assemblage 2-6
2.4.2.1. Syntaxe 2-7
2.4.2.2. Entrée et sortie 2-8
2.4.2.3. Identificateurs 2-9
2.4.2.4. Utilisation des identificateurs 2-9

2.4.3. Procédures de contrôle 2-11
2.4.3.1. WAIT 2-11
2.4.3.2. RESTART 2-11
2.4.3.3. STOP 2-11

2.4.4. Procédures relatives au traitement des tableaux 2-12
2.4.4.1. LOWBOUND (A, I) 2-12
2.4.4.2. RANGE (A, I) 2-12

2.4.5. Procédures d'allocation mémoire 2-12
2.4.5.1. STOREMAX 2-12
2.4.5.2. SIZE (A) 2-12

2.4.6. Procédures de contrôle 2-13

2.5. REMARQUES GENERALES 2-14

2.5.1. Paramètres spécifiés comme tableaux 2-14

2.5.2. Paramètres appelés par nom 2-14

2.5.3. Séquence d'opérations 2-14

.../...

Chapitre 3 INDICATIONS D'ERREURS

3.1. PENDANT LA COMPILATION 3-1

3.2. A L'EXECUTION D'UN PROGRAMME 3-6

3.2.1. Erreurs sur la bande donnée 3-6

3.2.2. Dépassement de capacité 3-7

3.2.3. Erreur qu'il est possible d'ignorer 3-8

Chapitre 4 INSTRUCTIONS DE PERFORATION 4-1

4.1. SUPPORT D'ENTREE 4-1

4.2. PREPARATION DES PROGRAMMES ALGOL 4-1

4.3. REMARQUES SUR LA PERFORATION 4-1

4.3.1. Perforation du programme 4-1

4.3.2. Confusion entre lettres et chiffres 4-1

4.3.3. Début de bande 4-2

4.3.4. Erreurs et corrections 4-2

4.3.5. Caractère "Halt" 4-2

4.4. REMARQUES A L'ATTENTION DU PROGRAMMEUR 4-2

4.4.1. Le nom du programme 4-2

4.4.2. Espacement 4-2

.../...

Chapitre 5 INSTRUCTIONS OPERATOIRES 5-1

5.1. MODE D'OPERATION 5-1

5.2. BANDES UTILISEES 5-1

5.2.1. Entrée de la bande SYSTEMS 5-1

5.2.2. Entrée du compilateur ALGOL 5-2

5.2.3. Entrée de la bande DRO 5-2

5.3. COMPILATION D'UN PROGRAMME ALGOL 5-3

5.4. EXECUTION D'UN PROGRAMME ALGOL 5-4

5.5. PROCEDURES OPERATOIRES 5-5

5.5.1. Méthode "load an go" pour petits programmes 5-5

5.5.2. Méthode pour longs programmes 5-5

5.6. AUTRES METHODES 5-6

5.6.1. Méthode "load and go" pour un seul programme 5-6

5.7. MESSAGES D'ERREURS 5-6

Chapitre 6 6.1. ARITHMETIQUE ENTIERE 6-1

6.2. POLYNOMES 6-1

6.3. PARAMETRES DE PROCEDURES APPELES PAR NOM ET PAR VALEUR 6-2

.../...

6.4. VARIABLES INDICEES 6-2

6.5. ENTREE ET SORTIE DES BLOCS 6-2

6.6. LONGUEUR DES BLOCS 6-2

Appendice I VERIFICATION D'UN PROGRAMME ALGOL I-1

CHAPITRE 1 INTRODUCTION

1.1. BUT

Le compilateur ALGOL 4100 a été réalisé pour mettre à la disposition des utilisateurs les facilités d'écriture du langage international ALGOL 60.

1.2. SOMMAIRE

Le système ALGOL 4100 est un sous ensemble d'ALGOL 60 comprenant pratiquement toutes les facilités de ce langage.

Les spécifications qui vont suivre donnent une idée des possibilités du compilateur Algol sur le NCR-ELLIOTT 4100. Aucun détail n'est donné sur le langage lui-même ; le lecteur est supposé connaitre ALGOL 60.

Les publications suivantes sont recommandées :

H. BOTTENBRUCH
Structure and Use of Algol 60
Journal of the ACM April 1962

E.W. DIJKSTRA
A primer of Algol 60 programming
Academic Press 1962

D.D. Mc CRACKEN
A Guide to Algol Programming
Wiley 1962

R. WOOLDRIDGE et J.F. RACTCLIFFE
An Introduction to Algol Programming
English Universities Press Ltd, 1963

J. ARSAC, A. LENTIN, M. NIVAT, L. NOLIN
ALGOL - Théorie et Pratique
Gauthier-Villars - Editeur - Paris

1.3. CONFIGURATION

La version actuelle du compilateur est prévue pour une utilisation sur des systèmes 4100 ayant la configuration suivante :

1 ou 2	Lecteurs de bande perforée
1 ou 2	Perforateurs de bande
1	Machine à écrire de contrôle
1	Imprimante (facultative)

Au minimum 16 384 mots de mémoire.

1.4. MODE D'UTILISATION

L'ALGOL 4100 peut être utilisé avec la méthode "load and go" ; on pourra aussi générer un programme objet sur bande perforée qui pourra être réentré ensuite pour exécution.

Si la place en mémoire est suffisante, plusieurs programmes pourront être compilés en mémoire puis exécutés.

1.5. EFFICACITE

Sur le 4120, les temps d'exécution d'un programme ALGOL contenant une proportion moyenne d'opérations en virgule flottante seront dans un rapport moyen de 3 avec les temps d'un programme correspondant écrit en langage machine.

Cependant, il est possible d'insérer des segments en langage d'assemblage NEAT qui permettront d'accroitre sensiblement l'efficacité des boucles intérieures.

1.6. FORME DE DISTRIBUTION

L'ALGOL 4100 est distribué sous forme de **3 bandes :**

- une bande binaire SYSTEMS avec contrôle de somme à la lecture
- une bande binaire "translatable" pour le compilateur
- une bande binaire translatable pour les Dynamic Routines.

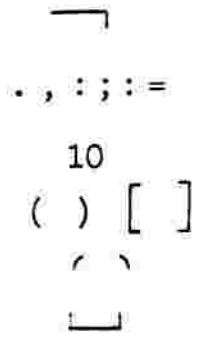
CHAPITRE 2 LES FONCTIONS

2.1. LA REPRESENTATION "HARDWARE"

2.1.1. Caractères et Symboles de base

Les programmes ALGOL 4100 peuvent être perforés en utilisant une machine à écrire Teletype 33 ; le modèle T de Flexowriter permet d'obtenir les lettres minuscules. L'ensemble des symboles ALGOL 60 est plus riche que chacun de ces deux ensembles et une adaptation est nécessaire ; elle sera conforme aux spécifications de l'ECMA.

Symbole ALGOL	Représentation 4100
A-Z	A-Z
a-z	a-z
0-9	0-9
begin end true etc.	"BEGIN" "END" "TRUE" etc.
+ − / ↑	+ − / ↑
x	*
÷	"DIV"
< > =	< > =
≤	"LE"
≥	"GE"
≠	"NE"
∧ ∨	"AND" "OR"
⊂ ≡	"IMPL" "EQUIV"



"NOT"

. , : ; : =

10

( ) [ ]

(espace)

Le symbole "GOTO" peut être écrit avec ou sans espace.

2.1.2. Représentation des fonctions standard

Les identificateurs des fonctions standard sont écrits en majuscule ; par exemple, la fonction

sin(P)

s'écrit SIN(P)

2.1.3. Les Nombres

2.1.3.1. Les entiers

La longueur du mot de mémoire 4100 est de 24 chiffres binaires ; les nombres négatifs sont représentés par leur complément à deux. Ainsi, les nombres entiers pouvant être représentés sont compris entre -8388608 et +8388607.

Dans un programme Algol, les nombres entiers qui dépassent ces limites seront convertis en nombres réels à la compilation.

Cependant, à l'exécution, une instruction du type

I:= 8 999 999 ;

où I est un identificateur du type entier, fera apparaitre le message

FPOLOE

sur la machine à écrire.

D'autre part, si le résultat d'une multiplication ou d'une division de nombre de type entier est à l'extérieur de ces limites, le message

INTOFLO

apparaitra sur la machine à écrire.

2.1.3.2. Les nombres réels

Ils sont représentés sous la forme virgule flottante et occupent deux mots de mémoire. La représentation flottante pour un nombre réel A est constitué par un couple (a, b) tel que :

$$A = a \times 2^b$$

a est appelé la mantisse et b l'exposant.

Sur le 4100, la mantisse vérifie l'une des relations suivantes :

$$\frac{1}{2} \leqslant a < 1 \qquad \text{ou } -1 \leqslant a < -\frac{1}{2} \qquad \text{ou} \quad a = 0$$

Un tel nombre est dit standardisé (sauf si a = 0).

L'exposant b est de type entier. Il peut prendre les valeurs comprises entre -256 et +255. Cependant, il est mémorisé sous la forme b+256 ; ainsi la valeur de l'exposant telle qu'elle mémorisée est comprise entre 0 et 511. Pour A = 0, a = 0 et b = -256 et ainsi l'exposant mémorisé est nul.

L'exposant occupe 9 chiffres binaires et la mantisse 39.

Les nombres qui peuvent être représentés sous cette forme sont compris entre $-5.8_{10}76$ et $+5.8_{10}76$. Tout nombre extérieur à cet intervalle est remplacé par un nombre de signe correct et le plus grand possible en valeur absolue.

Le plus petit nombre positif qui peut être distingué de zéro est $4.3_{10}-78$. Le plus grand nombre négatif est $-4.3_{10}-78$.

La précision est de 38 chiffres binaires soit 11 chiffres décimaux.

Les constantes réelles apparaissant dans un programme ALGOL sont mémorisées avec une précision de 11 chiffres décimaux. Si une constante a plus de 11 chiffres, elle est tronquée.

2.2. COMPATIBILITE AVEC ALGOL 60

(1) Symboles ALGOL et fonctions standard

La forme de représentation des symboles de base et des fonctions standard est donnée plus haut en 2.1.

(2) Longueur des identificateurs

Les six premiers caractères permettent de différencier les identificateurs.

(3) Exponentiation

Le résultat de l'expression P↑Q est toujours réel, quel que soit le type de P et Q.

(4) Récursivité

Dans une procédure récursive, les paramètres de type réel, entier ou booleen ou les paramètres spécifiés comme étiquette ne peuvent être appelés par nom. Cependant, une procédure peut s'appeler elle-même pendant l'évaluation de l'un de ses paramètres.

(5) Etiquettes sous forme d'entiers sans signe

Les entiers sans signe ne peuvent être utilisés comme étiquettes.

(6) Spécification des paramètres

Tous les paramètres formels d'une procédure doivent être spécifiés en tête de la procédure.

(7) Procédures comme paramètre d'une procédure

Un paramètre formel d'une procédure ne peut être spécifié comme procédure. Par conséquent, un identificateur de procédure ne peut apparaitre seul, sans paramètre, à moins que ce ne soit une procédure sans paramètre. Cette restriction peut être évitée en utilisant des paramètres appelés par nom.

(8) Séquence des déclarations

Un nombre quelconque de déclarations peut intervenir dans n'importe quel ordre au début d'un bloc à condition qu'aucun identificateur ne soit utilisé avant d'être déclaré. Le corps d'une procédure peut comprendre des appels d'une procédure déclarée avant, soit dans le même bloc, soit dans un bloc extérieur.

Si un identificateur est utilisé (par exemple, dans la liste des indices d'une tableau) avant qu'il ne soit redéclaré, alors la déclaration faite dans le bloc extérieur est prise en compte et l'erreur n'est pas détectée.

(9) Tableaux remanents

Les indices limites des tableaux remanents doivent être des entiers.

(10) Variables indicées

Une variable indicée ne peut être utilisée comme variable contrôlée d'une instruction "FOR".

(11) Noms des fonctions standard

Les noms des fonctions standard doivent être écrits avec des lettres majuscules.

2.4. ADDITIONS AU LANGAGE

2.4.1. Fonctions standard

Les fonctions standard suivantes, définies dans le rapport ALGOL, peuvent être utilisées :

SIN(E)	SQRT(E)	ENTIER(E)
COS(E)	EXP(E)	ABS(E)
	LN(E)	SIGN(E)

ARCTAN(E) avec résultat dans l'intervalle $-\frac{\pi}{2}$, $+\frac{\pi}{2}$

En supplément, d'autres fonctions peuvent être utilisées :

TAN(E)

ARCCOS(E) avec résultat dans l'intervalle O, π

ARCSIN(E) avec résultat dans l'intervalle $-\frac{\pi}{2}$, $+\frac{\pi}{2}$

Les arguments de SIN(E), COS(E) et TAN(E) sont expirmés en radians.

2.4.2. Segments en langage d'assemblage

Il est possible d'insérer des segments écrits en code NEAT dans un programme ALGOL 4100. Cependant, aucun mnémonique directeur NEAT ne peut être utilisé dans un segment.

Le segment doit être précédé par le symbole "CODE" et doit se terminer automatiquement par "END", "ELSE" ou un point virgule.

Les caractères tabulation et espace sont ignorés mais peuvent être utilisés pour la mise en page. Le caractère doit être utilisé pour la tabulation et $$$ (ou %) pour représenter trois caractères de tabulation. Le retour chariot indique la fin d'une instruction et les retours-chariot redondants sont ignorés.

EXEMPLE ;

```
"BEGIN" "INTEGER" TERM1, TERM2, RESULTAT, INDICATEUR ;
        "COMMENT"  SI  INDICATEUR = 10 RESULTAT VAUT
        TERM1 SINON TERM1 + TERM2 ;
        "READ" TERM1, TERM2, INDICATEUR ;
        "CODE"  %  LDR $ TERM1
                %  LD $ INDICATEUR
                %  COMP:L $ 10
                %  JZ $ SUITE
                %  ADDR $ TERM2
                $$ SUITE $ STR $ RESULTAT ;
        "PRINT" RESULTAT
"END" EXEMPLE ;
```

2.4.2.1. Syntaxe

<instruction de base non étiquetée> ::= <instruction d'affectation > |
<instruction goto >|<instruction fictive> |
<instruction procédure> |<segment code NEAT >

<segment code NEAT> ::= "CODE" <corps en code NEAT>

<corps en code NEAT> ::= <instruction NEAT> | <nde> |
<corps en code NEAT> <retour chariot>
<instruction NEAT> | <corps en code NEAT>
<retour chariot>

<Instruction NEAT> ::= <instruction NEAT étiquetée> |
<instruction NEAT non étiquetée> | <étiquette

<instruction NEAT étiquetée> ::= <étiquette> <instruction de base NEA

<instruction NEAT non étiquetée> ::= $$$ <instruction de base NEAT
% <instruction de base NEAT>

<instruction de base NEAT> ::= <fonction> | <fonction> $
<partie adresse> | GETA$
<identificateur de tableau> | GETS$
<paramètre formel spécifié chaîne>

<fonction> ::= <mnémonique> | <mnémonique> : <mode d'adressage>

<partie adresse> ::= <entier sans signe> | <étiquette à l'intérieur du
segment> | <identificateur déclaré en ALGOL
identificateur déclaré en ALGOL> +
<entier sans signe> | <identificateur déclaré
en ALGOL> - <entier sans signe>

2.4.2.2. Entrée et sortie

(1) Une expression de désignation extérieure à un segment NEAT ne peut se référer à une étiquette du segment ; une instruction de saut à l'intérieur d'un segment NEAT ne peut se référer à une étiquette extérieure au segment. En d'autres termes, une entrée dans un segment NEAT ne peut être faite que par le symbole "CODE" et la sortie ne peut avoir lieu qu'à la fin du segment (indiquée par le symbole "END", "ELSE" ou le point virgule) ou par l'intermédiaire d'une instruction J, JIL, JIR, MTOS ou RTOS.

(2) Le symbole terminant un segment en code NEAT peut être étiqueté.

2.4.2.3. Identificateurs

(1) Aucune déclaration ne pouvant figurer à l'intérieur du segment en code NEAT, tous les identificateurs apparaissant dans les parties adresses des instructions doivent être déclarés dans les blocs extérieurs.

(2) Les lettres minuscules peuvent être utilisées pour les identificateurs.

2.4.2.4. Utilisation des identificateurs

La partie adresse d'une instruction en code NEAT peut être

(1) Un entier sans signe

(2) Un identificateur représentant une étiquette qui spécifie une instruction du segment NEAT.

(3) Un identificateur représentant une variable de type entier.

(4) Un identificateur représentant une variable de type réel.

(5) Un identificateur représentant une variable de type booleen.

(6) Un identificateur représentant un tableau qui peut être utilisé avec le mnémonique GETA.

GÉTA est un extracode pouvant seulement être utilisé à l'intérieur d'un segment en code NEAT ; il a pour effet de placer dans R1 l'adresse du pointeur du tableau spécifié dans la partie adresse. Les registres M et K sont affectés.

(7) Un identificateur, représentant un paramètre formel spécifié "string", qui peut être utilisé avec le mnémonique GETS.

GETS est un extracode pouvant seulement être utilisé à l'intérieur d'un segment en code NEAT ; il a pour effet de placer dans R l'adresse du premier mot de la chaine. Les registres M et K sont affectés.

En ALGOL, les chaines sont groupées à raison de quatre caractères par mot, le dernier caractère étant (symbole de fermeture de chaine). Ainsi, l'instruction :

% GETS $ Z

placera dans R l'adresse de la chaine Z.

(8) Un identificateur de variable suivi d'un complément d'adresse.

Les paramètres formels appelés par valeur sont traités comme des variables ; les paramètres appelés par nom (à l'exception des tableaux et des chaines) ne peuvent être traitées.

2.4.2.5. Restrictions

(1) L'utilisation des mnémoniques JIX, JILX, JIRX n'est pas permise.

(2) On ne peut utiliser de mnémoniques directeurs NEAT.

(3) Les commentaires ne peuvent être insérés dans la zone NOTES comme sur les feuilles de programmation NEAT.

2.4.3. Procédures de contrôle

L'ALGOL 4100 permet l'utilisation de procédures standard facilitant l'exécution des programmes ; la plupart d'entre elles n'ont pas d'effet sur le calcul.

2.4.3.1. WAIT

Cette procédure permet d'arrêter l'exécution d'un programme jusqu'à ce que la position de la clé n° 6 soit changée deux fois. Le message WAIT apparait sur la machine à écrire.

Cette procédure peut être utilisée lorsque des manœuvres doivent être exécutées par l'opérateur (par exemple, monter une bande magnétique).

On écrira :

```
"PRINT" PUNCH(3), "L` PLACER LA BANDE C SUR DEROULEUR
                   2` ;
WAIT;
```

2.4.3.2. RESTART

Cette procédure permet de réexécuter un programme. Elle pourra être utilisée lorsqu'un programme doit être exécuté avec plusieurs jeux de données. Cependant, il est toujours possible de réexécuter un programme qui ne comprend pas d'instruction RESTART.

2.4.3.3. STOP

Cette procédure a pour effet d'arrêter l'exécution d'un programme et de donner le contrôle à l'exécutif NICE. Le message END apparait sur la machine à écrire.

2.4.4. Procédures relatives au traitement des tableaux

2.4.4.1. LOWBOUND(A, I)

La valeur de cette procédure est celle de la limite inférieure du Ième indice à partir de la gauche du tableau A.

2.4.4.2. RANGE(A, I)

Cette procédure donne le nombre de valeurs que peut prendre le Ième indice à partir de la gauche du tableau A.

Si à l'appel de l'une des procédures LOWBOUND ou RANGE, la valeur de I est extérieure à l'intervalle $1 \leqslant I \leqslant$ nombre d'indices de A, le message

SUBOFLO

apparait sur la machine à écrire.

2.4.5. Procédures d'allocation mémoire

2.4.5.1. STOREMAX

Cette procédure donne le nombre de positions de mémoires libres pour le stockage des tableaux.

Le programmeur pourra utiliser cette procédure pour déterminer la dimension maximum permise pour un tableau.

2.4.5.2. SIZE(A)

Cette procédure donne le nombre de positions de mémoires occupées par le tableau A.

2.5. REMARQUES GENERALES

2.5.1. Paramètres spécifiés comme tableaux

Si un paramètre formel est spécifié comme tableau, aucun contrôle n'est effectué sur le paramètre actuel correspondant pour savoir si il est du même type ou s'il a les mêmes dimensions.

2.5.2. Paramètres appelés par nom

Si dans le corps d'une procédure, on affecte une valeur à l'un des paramètres formels alors le paramètre actuel correspondant doit être du même type ; ce paramètre actuel ne pourra être ni une constante ni une expression.

2.5.3. Séquence d'opérations

L'ordre dans lequel les opérations sont effectuées dans une expression n'est pas défini sauf dans le cas où il est déterminé par les règles de priorité suivantes :

- expressions arithmétiques

a) exponentiation

b) multiplication et division

c) addition et soustraction.

Les expressions sont toujours évaluées de la gauche vers la droite.

CHAPITRE 3 INDICATIONS D'ERREUR

3.1. PENDANT LA COMPILATION

Si un caractère erroné est lu (par exemple, une erreur de parité) le message

PARITY

apparait sur la machine à écrire.

En tapant

CONT:

la compilation continue, le caractère erroné étant ignoré.

Si une erreur syntaxique est détectée pendant la traduction, le message

ALG ERR

XX

apparait suivi par le dernier identificateur lu si celui-ci constitue le symbole erroné ; le message apparait également sur le perforateur de bande n° 1, suivi par les 64 caractères subséquents.

TABLE D'ERREURS

XX	Nature de l'erreur
1	Ecriture interdite pour un nombre
2	Erreur dans un symbole de base
3	Début d'instruction non permis
4	Déclaration de procédure non terminée par un point virgule
5	Déclaration d'identificateur non terminée par un point virgule
6	La déclaration qui suit une déclaration de procédure n'est pas une déclaration de procédure
7	Identificateur déclaré deux fois dans le même bloc
8	Etiquette intervenant deux fois dans le même bloc
9	Elément non permis dans une déclaration
10	Premier élément d'une déclaration d'aiguillage non suivi de :=
Dans une déclaration de procédure (XX11 à 21)	
11	L'élément suivant une déclaration de procédure n'est ni un point virgule, ni un symbole (
12	Pas de point virgule ou de) après la liste des paramètres formels N.B. Si l'élément suivant le) n'est pas un point virgule, le compilateur ignorera la chaine de caractères jusqu'au symbole ":"
13	La liste de la partie spécification ou de la partie valeur a une forme non permise.
14	La partie spécification est avant la partie valeur.
15	Paramètre d'un type non permis

16	Trop de paramètres (plus de 24)
17	Paramètre non spécifié
18	**Procédure récursive avec paramètre de type réel, entier,** booleen ou spécifié comme étiquette appelé par nom
19	Un identificateur dans la partie valeur n'est pas un paramètre
20	Un identificateur dans la partie spécification n'est pas un paramètre
21	Paramètre spécifié deux fois
22	Erreur dans le nom du programme
Dans les déclarations de tableaux (XX23 à 26)	
23	Pas de virgule ou de [après un identificateur
24	Pas de : entre les bornes d'indice
25	Pas de virgule ou de] après une paire de bornes
26	Tableau comportant trop de dimensions (plus de 63)
27	Pas de "END" ou de point virgule après une instruction composée
28	Pas de : après une étiquette
29	Début d'instruction non permis
30	Partie gauche d'une instruction d'affectation non suivi par :=
31	Valeur affectée à un identificateur de procédure à l'extérieur du corps de la procédure
32	Identificateur non déclaré ou n'ayant pas de portée dans ce bloc
33	Primaire de type interdit dans une expression arithmétique
34	Expression arithmétique vide

35	Omission d'un opérande dans une expression arithmétique
36	Pas de] après une liste d'indices
37	Parenthèse manquante
38	Nombre erroné d'indices dans une variable indicée
39	Il manque un "ELSE"
40	Il manque un "THEN"
41	Instruction ou expression conditionnelle après "THEN"
42	Opérateur manquant ou erroné dans une expression arithmétique
43	Opérande booleen dans une expression arithmétique
44	Symbole (manquant après un identificateur de procédure avec paramètres
45	Identificateur non permis à gauche du symbole d'affectation
46	Identificateur non permis comme variable contrôlée d'une instruction "FOR"
47	Expression booléenne vide
48	Il manque un opérateur de relation
49	Omission d'un opérande dans une expression arithmétique
50	Primaire booléen non permis
51	Opérateur non permis dans une expression booléenne
52	Symbole non permis au début d'une expression
A l'appel d'une procédure (XX53 à 57)	
53	Pas de (après un identificateur de procédure comportant des paramètres
54	Paramètre actuel non suivi d'une virgule ou de) (cf. erreur n° 12)
55	Erreur dans un délimiteur de la forme) chaine de caractères

56	Pas de paramètre actuel spécifié "ARRAY" ou "LABEL"
57	Paramètre interdit
58	Variable contrôlée dans une instruction "FOR" non suivie de :=
59	Elément d'une liste "FOR" non suivi de virgule ou "DO"
60	Expression de désignation incorrecte
61	Expression arithmétique dans un élément de liste "FOR" non suivi par "STEP" "WHILE" "DO" ou une virgule
62	Il manque "UNTIL"
63	Programme trop important pour être compilé
64	Symbole ' à l'intérieur d'une chaine double
65	"COMMENT" intervenant ailleurs qu'après un point virgule ou un "BEGIN"
66	La partie droite d'une relation est une expression arithmétique conditionnelle
67	Instruction "GOTO" allant dans une instruction d'une boucle "FOR"
68	Opérandes booléens et entiers dans une expression booléenne
69	Nombre erroné de paramètres dans un appel de procédure
70	Paramètre formel spécifié tableau non remplacé par un identificateur de tableau
71	Erreur dans un paramètre
72	Erreur dans un segment en code NEAT
73	Saut à une étiquette non existante dans un segment NEAT
74	Erreur dans une déclaration d'aiguillage
75	Tableau remanent avec bornes variables
76	Le symbole suivant "OWN" n'est ni "REAL", ni "INTEGER", ni "BOOLEAN", ni "ARRAY".

Si un programme comporte une ou plusieurs instructions de saut à une étiquette n'existant pas, le message

NOLABEL

apparait sur la machine à écrire suivi de la liste des identificateurs d'étiquettes.

N.B. L'absence de code Stop à la fin d'une bande ALGOL entrainera le défilement de la bande sous le lecteur.

A la compilation d'un programme, les erreurs suivantes :

(1) le nombre de "END" est insuffisant

(2) pas de point virgule après le dernier "END"

(3) il manque un symbole de fermeture de chaine

feront apparaitre le message RELOAD sur la machine à écrire ; la compilation ne sera pas terminée.

Les erreurs suivantes arrêteront la compilation d'un programme prématurément :

(1) manque d'un "BEGIN"

(2) un "END" ou le commentaire suivant "END" n'est pas suivi par "END", "ELSE" ou point virgule.

3.2. A L'EXECUTION D'UN PROGRAMME

3.2.1. Erreurs sur la bande de donnée

Si l'un des cas d'erreur ci-dessous est rencontré à l'entrée des données

READ ERR

apparait sur la machine à écrire.

L'exécution du programme peut être continuée en tapant :

CONT:

Dans ce cas, la valeur zéro sera prise comme valeur lue.

CAS D'ERREURS

- Un entier comprend un point décimal ou exposant $_{10}$
- Deux points décimaux dans un même nombre
- Deux exposants$_{10}$ dans un même nombre
- Un point décimal à la suite d'un exposant $_{10}$
- Pas de chiffre, ni de signe après un exposant $_{10}$
- Pas de chiffre, ni de point décimal ni d'exposant $_{10}$ après un signe
- Pas de chiffre après un point décimal
- Pendant l'exécution d'une instruction INSTRING, occurence d'un caractère numérique avant le premier symbole d'ouverture de chaine.
- Pendant l'exécution d'une instruction INSTRING, occurence d'un symbole ouverture de chaine à l'intérieur d'une chaine double.

3.2.2. Dépassement de capacité

Les trois erreurs suivantes de dépassement de capacité ne permettant pas de continuer l'exécution d'un programme :

INTOFLO Le résultat d'une opération entière (par exemple, une division par zéro) est à l'extérieur de l'intervalle $(-2^{23}, 2^{23} - 1)$

SUBOFLO(1) (1) L'un des indices d'une variable indicée est à l'extérieur des limites définies pour cet indice.

(2) Dans la déclaration d'un tableau : l'indice inférieur est supérieur à l'indice supérieur

(3) A l'appel de l'une des procédures LOWBOUND ou RANGE, le second paramètre est supérieur au nombre d'indices que comprend le tableau

(4) Un indicateur d'aiguillage est utilisé comme paramètre actuel : voir plus bas SWITOFLO

NOROOM Le programme nécessite plus de place que celle disponible sur le calculateur.

3.2.3. Erreurs qu'il est possible d'ignorer

Les erreurs ci-dessous peuvent être ignorées et l'exécution du programme reprise en tapant

CONT:

sur la machine à écrire ; la valeur de la fonction sera alors prise égale à zéro sauf pour les fonctions marquées d'un astérisque pour lesquelles on prendra la plus grande valeur possible avec pour signe celui de l'argument.

	SQRT ERR	$x < 0$
	SIN ERR	$\|x\| > 8.5_{10}11$ (approximativement)
	COS ERR	$\|x\| > 8.5_{10}11$ (approximativement)
*	TAN ERR	$\|x\| > 4.2_{10}11$ (approximativement)
	LOG ERR	$x \leqslant 0$ ou $p \uparrow q$ avec $p < 0$ et q réel
*	EXP ERR	$x > 254 \log_e 2$ ($\backsim$ 176.0)

	ARCS ERR	$\|x\| > 1$
	ARCC ERR	$\|x\| > 1$
*	FPOLOW	Dépassement de capacité en virgule flottante lors d'une instruction d'affectation; en continuant le plus grand nombre positif est placé en mémoire.
*	FPOLFOE	Dépassement de capacité pendant l'exécution de la fonction ENTIER
*	FPOFLOMD	Dépassement de capacité lors d'une multiplication ou division en virgule flottante ; en continuant le plus grand nombre positif est placé dans le FPA.
	OUTS ERR	La chaine sortie par OUTSTRING contient un caractère inexistant ; en continuant, la suite de la chaine est ignorée.
	PRNT ERR	Le nombre réel non nul à imprimer a une mantisse inférieure à $\frac{1}{4}$ en valeur absolue ; en général cela provient du fait que le programmeur n'a pas assigné de valeur à une variable avant de l'imprimer.

Si on essaie d'imprimer un nombre trop grand pour le format d'impression défini, il y aura "alarme d'impression" (alarm printing) et ceci de façon à ne pas modifier la mise en page. Si le nombre total de caractères spécifié par le format est supérieur à 6, le nombre sera imprimé avec le format "scaled", de façon à ce qu'il occupe le même espace que celui prévu par le programmeur. Si le nombre, n, de caractères est inférieur à 6, la lettre H, suivie d'un code "HALT", est imprimée précédée de (n - 1) espaces.

	SWITOFLO	La valeur de l'indice d'un indicateur d'aiguillage est à l'extérieur de l'intervalle ; en continuant, le programme exécute l'instruction suivante ; dans le cas où l'indicateur d'aiguillage est utilisé comme paramètre actuel, SUBOFLO apparait en même temps sur la machine à écrire.

CHAPITRE 4 INSTRUCTIONS DE PERFORATION

4.1. SUPPORT D'ENTREE

Le support d'entrée est une bande perforée (8 canaux) préparée sur Flexowriter (modèle T) ou Teletype (modèle 33). Après la perforation on devra lister la bande et comparer le texte obtenu avec le manuscrit.

4.2. PREPARATION DES PROGRAMMES ALGOL

Les programmes ALGOL devront être écrits sur du papier à lignes comprenant des séparations verticales ; ces dernières indiquent l'endroit où doit commencer l'impression de la ligne.

En utilisant le modèle 33 de Teletype, huit espaces seront perforés pour chaque tabulation.

4.3. REMARQUES POUR LA PERFORATION

4.3.1. Perforation du Programme

Il faudra perforer exactement ce qui est nécessaire pour obtenir une copie fidèle du manuscrit. Les mots ou les symboles de base (placés entre guillemets) qui se suivent devront être séparés par un espace.

4.3.2. Confusion entre lettres et chiffres

Il faudra prendre soin d'éviter toute confusion entre le chiffre 1 et la lettre l et entre le chiffre 0 et la lettre O.

4.3.3. Début de bande

Toute bande programme devra comporter au début, quelques décimètres de caractères "blanc" ; un ou plusieurs caractères "blanc" pourront d'ailleurs apparaitre n'importe où sur la bande.

4.3.4. Erreurs et Corrections

Un caractère erroné peut être effacé en lui superforant le caractère "Delete" ; ce caractère peut apparaitre n'importe où sur la bande.

4.3.5. Caractère "Halt"

Ce caractère doit être perforé à la fin de chaque bande programme (ou à la fin de chaque partie de bande programme si le programme est perforé en plusieurs parties) et aussi à la fin de chaque bande de données.

4.4. REMARQUES A L'ATTENTION DU PROGRAMMEUR

4.4.1. Chaque programme ALGOL devra comporter un titre. Celui-ci doit commencer par une lettre et ne contenir que des lettres majuscules ou des chiffres ; il doit se terminer par un point virgule.

Le titre ne faisant pas partie du programme, on ne pourra insérer de commentaire immédiatement après le point virgule qui le termine.

4.4.2. Espacement

Les mots ou les symboles de base (placés entre guillemets) qui se suivent devront être séparés par un espace afin d'améliorer la mise en page.

CHAPITRE 5 INSTRUCTIONS OPERATOIRES

5.1. MODE D'OPERATION

Deux modes d'opérations existent pour la compilation et l'exécution d'un programme ALGOL. L'utilisateur peut soit compiler le programme en mémoire et l'exécuter immédiatement (méthode "Load and Go"), soit compiler sur bande perforée de manière à pouvoir réentrer le programme, par la suite, sous le contrôle de la bande SYSTEMS et l'exécuter.

5.2. BANDES UTILISEES

ALGOL SYSTEMS et le compilateur ALGOL doivent être en mémoire lors de la compilation ; à l'exécution du programme, ALGOL SYSTEMS et ALGOL DRO doivent être en mémoire.

5.2.1. Entrée de la bande SYSTEMS

(1) Appuyer sur le bouton poussoir RESET situé sur le pupitre de contrôle.

(2) Placer la bande SYSTEMS sous le lecteur de bande n° 1.

(3) Appuyer sur le bouton INITIAL INSTRUCTIONS.

La bande est lue ; à la fin de la lecture, le message

** MES:

apparait sur la machine à écrire. Un message est alors attendu de l'opérateur.

5.2.2. Entrée du compilateur ALGOL

SYSTEMS doit déjà être en mémoire centrale.

(1) Placer la bande compilateur ALGOL sous le lecteur N° 1.

(2) Taper IN :

La bande est lue ; à la fin de la lecture, le message

ALGOL
END

** MES:

apparait sur la machine à écrire.

5.2.3. Entrée de la bande DRO

SYSTEMS doit être en mémoire centrale.

(1) Placer la bande DRO sous le lecteur n° 1.

(2) Taper IN:

La bande est lue ; à la fin de la lecture, le message

DRO
END

** MES:

apparait sur la machine à écrire.

5.3. COMPILATION D'UN PROGRAMME ALGOL

SYSTEMS et le compilateur ALGOL doivent être en mémoire.

(1) Placer la bande programme sous le lecteur n° 1.

(2) Pour compiler le programme directement en mémoire, taper :

```
ALGOL:
```

ou

```
ALGOL, CH:
```

A la fin de la compilation, le nom du programme apparaitra sur la machine à écrire suivi de :

```
END

** MES:
```

Pour compiler un programme sur bande perforée, on tapera :

```
ALGOL, PT:
```

ou

```
ALGOL, PT, CH:
```

A la fin de la compilation le message :

```
END

** MES:
```

apparaitra sur la machine à écrire.

Si le programme est compilé sur ruban perforé, la bande devra être enroulée à l'envers.

Programmes en Plusieurs Parties

Si le programme est perforé en plusieurs parties, le message

```
RELOAD
```

apparaitra sur la machine à écrire lorsque le caractère "halt" est lu à la fin de la bande.

5.5. PROCEDURES OPERATOIRES

5.5.1. Méthode load and go pour petits programmes

(1) Lire la bande SYSTEMS

(2) Lire le compilateur ALGOL

(3) Lire la bande DRO

(4) Compiler le programme

(5) Exécuter le programme compilé

Pour compiler et exécuter d'autres programmes on retournera en (4).

5.5.2. Méthode pour longs programmes

(1) Lire la bande SYSTEMS

(2) Lire le compilateur ALGOL

(3) Compiler le programme ALGOL sur bande perforée en tapant ALGOL, PT:

Retourner en (3) pour compiler d'autres programmes.

(4) Taper

REMOVE, ALGOL:

(5) Lire la bande DRO

(6) Lire et exécuter le programme compilé.

Retourner en (6) pour réexécuter le même programme ou

Taper

REMOVE,

suivi du nom du programme et de deux points.

Retourner en (6) pour exécuter d'autres programmes déjà compilés.

5.6. AUTRES METHODES

5.6.1. Méthode "Load and go" pour un seul programme

Il est possible par cette méthode de compiler et d'exécuter un seul programme.

(1) Lire la bande SYSTEMS

(2) Lire le compilateur ALGOL

(3) Compiler le programme

(4) Effacer le compilateur ALGOL en tapant

REMOVE, ALGOL:

(5) Lire la bande DRO

(6) Exécuter le programme compilé

(7) Effacer le programme que l'on vient d'exécuter en tapant

REMOVE

suivi du nom du programme et de deux points.

(8) Effacer DRO en tapant

REMOVE, DRO:

Retourner en (2) pour compiler et exécuter un autre programme.

5.7. MESSAGES D'ERREURS

(1) Le message

INERROR

apparait sur la machine à écrire si une erreur se produit lors de la lecture de la bande ALGOL, de DRO ou du programme compilé.

(2) Pendant l'exécution d'un programme, l'un des messages signalés au Chapitre 3 section 2 peut apparaitre sur la machine à écrire. Dans certains cas, il est possible de continuer en tapant

CONT:

CHAPITRE 6

Nous n'avons pas l'intention d'exposer des méthodes d'analyse numérique mais seulement de mettre en évidence certains points relatifs à l'écriture de programmes ALGOL sur le 4100.

6.1. ARITHMETIQUE ENTIERE

Il vaut mieux utiliser des opérations simples chaque fois qu'il s'agit de variables de type entier. Dans les exemples suivants, toutes les variables sont de type "INTEGER"

(1) I + I est à peu près 10 fois pous rapide que 2 * I

(2) I* I est nettement plus rapide que I ↑ 2 puisque I↑ 2 est calculé par un sous-programme et que le résultat est de type "REAL".

(3) L'instruction J:= ABS(I) entraine une double conversion puisque la fonction ABS a un argument de type "REAL" (cf. rapport ALGOL 60). On écrira pour cela :

J:= "IF" I < 0 "THEN" -I "ELSE" I ;

6.2. POLYNOMES

On utilisera la méthode de Horner pour l'écriture des polynômes.

Le polynôme :

17 * X ↑ 5 + 4 * X ↑ 3 + 3 * X ↑ 2 + 2

s'écrira

(((((17 * X) * X + 4) * X + 3) * X) * X + 2)

6.3. PARAMETRES DE PROCEDURES APPELES PAR NOM ET PAR VALEUR

Chaque fois que cela est possible, les paramètres formels d'une procédure devront être spécifiés par valeur.

6.4. VARIABLES INDICEES

Si des variables indicées apparaissent à l'intérieur de boucles "FOR", il est recommandé de s'assurer que leurs indices sont simples, les expressions arithmétiques ayant été évaluées à l'extérieur de la boucle "FOR".

Par exemple,

```
"FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
        A [ B + C/2, I]  := A [B + C/2, I] + 1 ;
```

est moins efficient que

```
K:= B + C/2 ;
"FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
        A [K, I]  := A [K,I] + 1 ;
```

6.5. ENTREE ET SORTIE DES BLOCS

Le nombre d'entrées et de sorties dans un bloc devra être réduit à un minimum.

6.6. LONGUEUR DES BLOCS

En compilant sur ruban perforé, le programme objet n'est perforé que lorsque on rencontre le "END" du bloc correspondant.

APPENDICE I

VERIFICATION D'UN PROGRAMME ALGOL

1 - CONTROLE

(1) Vérifiez que le symbole de multiplication * n'a pas été oublié.

(2) Assurez vous que vous n'effectuez pas de division par zéro, que vous n'évaluez pas le logarithme de zéro ou d'une quantité négative, la racine carrée d'un nombre négatif.

(3) Assurez vous que tous les nombres sont écrits sous une forme acceptable.

(4) Vérifiez que vous n'avez pas deux opérateurs arithmétiques qui se suivent.

(5) En testant l'ordre de grandeur de quantités, pensez à tester la valeur absolue de ces quantités.

2 - VERIFICATIONS PARTICULIERES AU LANGAGE ALGOL

(7) Vérifiez que tous les symboles de base sont placés entre guillemets.

(8) Vérifiez que les symboles de base sont bien orthographiés.

(9) Contrôlez la ponctuation de votre programme ; toute instruction doit être suivie d'un point virgule de "ELSE" ou de "END"

"END" doit être suivi par un point virgule, "ELSE" ou un autre "END". Si le point virgule est oublié après un "END", l'instruction suivante sera interprétée comme un commentaire et donc ne sera ni compilée, ni à fortiori exécutée. Vérifiez que le symbole de base "COMMENT" n'apparait qu'après un point virgule ou un "BEGIN" et que le texte du commentaire est terminé par un point virgule.

(10) A chaque "IF" d'une instruction "IF", il doit correspondre un "THEN". Un "IF" ne doit jamais suivre un "THEN". Se souvenir qu'une expression conditionnelle doit toujours comprendre un "ELSE" et que cela n'est pas nécessaire pour une instruction "IF".

Ainsi l'instruction

```
"IF" A < 0  "THEN"  A := -A ;
```

est équivalente à :

```
A := "IF"  A < 0  "THEN" - A  "ELSE"  A ;
```

et il est faux d'écrire :

```
A :=  "IF"  A< 0 "THEN" - A ;
```

(11) Les parties gauches d'une instruction d'affectation doivent être du même type ainsi

```
A:= B:= C * D
```

est faux si A est de type "REAL" et B de type "INTEGER"

(12) Dans les expressions arithmétiques, il doit correspondre un symbole) à tout symbole (et inversement ; les arguments des fonctions standard doivent être placés entre parenthèses ; les crochets doivent être utilisés dans les déclarations de tableaux et les variables indicées.

(13) Il doit correspondre un symbole "END" à tout "BEGIN".

(14) Toute instruction composée doit être commencée par un "BEGIN" et terminée par un "END" ; il faut vérifier cela particulièrement dans le cas d'une instruction composée suivant le "DO" d'une instruction "FOR".

(15) Toutes les variables utilisées doivent être déclarées et n'ont de portée qu'à l'intérieur du bloc dans lequel elles sont déclarées. Avant d'utiliser une variable dans une expression, il faut lui affecter une valeur. Seuls les 6 premiers caractères d'un identificateur permettent de le différencier.

(16) Les opérandes de la division entière $\div$ doivent être tous les deux de type "INTEGER". En outre, le résultat de I$\uparrow$ J est de type "REAL" même si I et J sont de type "INTEGER".

(17) Une instruction "GOTO" ne peut conduire à l'intérieur d'un bloc ou d'une instruction "FOR".

(18) Effectuez deux tests pour chaque expression ou instruction conditionnelle : l'un répondant "TRUE" et l'autre "FALSE" et vérifiez ainsi que vous avez prévu tous les cas.

(19) A la sortie par épuisement de la liste de l'instruction "FOR" la valeur de la variable contrôlée n'est plus définie et ne peut donc être utilisée.

Par contre, en cas de sortie pas une instruction "GOTO", la valeur de la variable contrôlee sera celle qu'elle avait immédiatement avant l'exécution de l'instruction "GOTO".

(20) L'expression booléenne A = B où A et B sont du type "REAL" peut avoir la valeur "FALSE" même si théoriquement A et B sont égaux ; en effet, les nombres du type "REAL" ne sont pas mémorisés avec une précision complète. Afin d'éviter ce type d'erreur, il vaut mieux effectuer le test

$$\text{abs}(A - B) < \text{epsilon}$$

où epsilon est une constante positive de l'ordre de $_{10}$-10.

Cette liste n'est pas exhaustive.