

MANUEL DU PROGRAMMEUR

Fortran

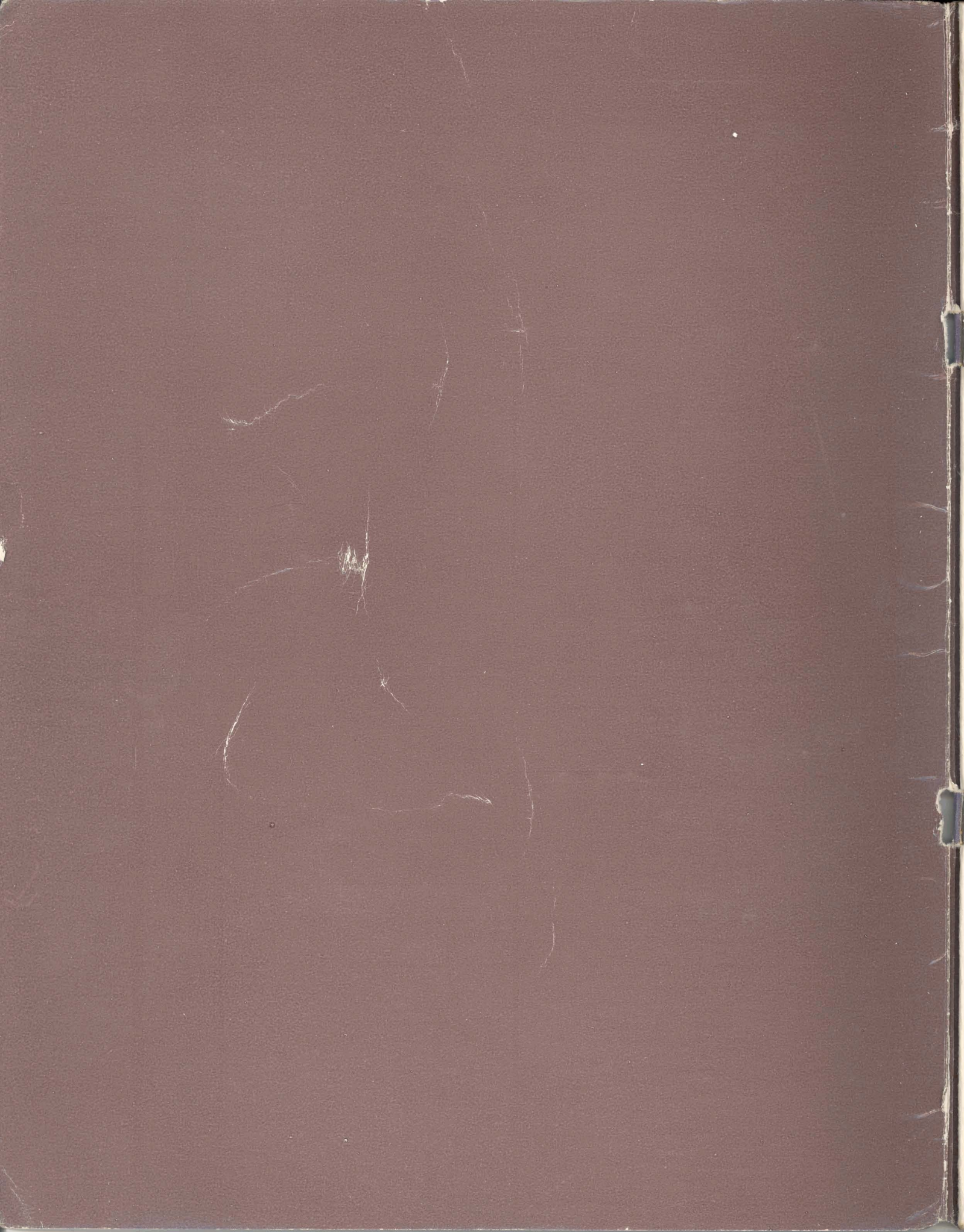
PROGRAMMATION

AUTOMATIQUE

DE

L'ORDINATEUR 704

IBM



LE SYSTÈME FORTRAN

Le système de FORMulation TRANsposée, ou FORTRAN, est un système de programmation automatique de l'Ordinateur Scientifique IBM 704. Il est basé sur un langage — le langage FORTRAN — similaire à celui des mathématiques. A partir d'un programme origine, écrit en ce langage, le « traducteur » FORTRAN crée un programme résultant, en langage machine, prêt à être utilisé par le 704. Ce traducteur est composé d'environ 25.000 instructions qui sont enregistrées sur la bande maîtresse FORTRAN.

FORTRAN transforme donc le 704 en une machine avec laquelle on peut communiquer dans un langage plus familier que le langage 704 lui-même. Il présente l'avantage de réduire considérablement le temps de formation de l'utilisateur, le temps passé à écrire et à mettre au point un programme.

Les principales caractéristiques du système FORTRAN sont exposées ci-après.

Universalité du langage

FORTRAN figure parmi les premiers langages se rapprochant le plus du langage des mathématiques. Le système décrit dans cette brochure s'applique à l'Ordinateur 704. En effet, le premier traducteur FORTRAN a été conçu pour l'Ordinateur IBM 704 composé de quatre unités de bandes magnétiques et d'une unité de tambour. Le programme résultant produit est exécutable pour tout ordinateur 704.

L'utilisation de ce langage a été toutefois étendue aux Ordinateurs IBM 650 et 705 et il le sera aux futures machines IBM. Ainsi, sur le 650 il existe une série de programmes permettant, avec une machine standard, de créer des programmes résultants pour 650 standard ou 650 à bandes.

Efficacité du Pro- gramme résultant

Les programmes résultants obtenus par FORTRAN sont comparables à ceux que réaliseraient de très bons programmeurs.

**Champ
d'application**

Le langage FORTRAN est capable d'exprimer n'importe quel problème numérique. En particulier, il est très commode pour les programmes contenant un nombre important de formules et de variables, y compris les variables indicées à 1, 2 ou 3 dimensions.

Cependant, pour des problèmes dans lesquels les informations enregistrées dans la machine ont une signification logique plutôt que numérique, le langage FORTRAN est moins satisfaisant car certaines opérations logiques ne lui sont pas directement accessibles. Néanmoins, celles-ci peuvent toujours être exprimées en utilisant les sous-programmes de la bibliothèque.

**Bibliothèque de
sous-programmes**

Des sous-programmes préétablis pour évaluer des fonctions uniformes à un ou plusieurs arguments peuvent être mis sur la bande matresse FORTRAN et placés automatiquement par le traducteur FORTRAN dans le programme résultant.

**Souplesse des
Entrées et
Sorties**

Certaines expressions du langage FORTRAN insèrent dans le programme résultant la possibilité de procéder à n'importe quelles entrées ou sorties. Les conversions de « binaire » à « décimal » et de « décimal » à « binaire » sont incluses, donnant ainsi la plus grande liberté dans le choix du modèle des données et des résultats.

**Nature de
l'Arithmétique
Fortran**

L'arithmétique du programme résultant est une arithmétique « simple précision », le plus souvent « virgule flottante ». Les nombres utilisés ont 27 chiffres significatifs binaires (environ 8 chiffres décimaux) et un exposant pouvant varier de 10^{-38} à 10^{+38} , le zéro étant normalisé.

L'arithmétique « virgule fixe » ne traite que des nombres entiers et n'est utilisée en général que pour les indices.

TABLE DES MATIÈRES

	page
LE SYSTÈME FORTRAN	2
CHAPITRE 1. GÉNÉRALITÉS SUR LE PROGRAMME ORIGINE FORTRAN.....	7
2. LE LANGAGE FORTRAN :	
Constantes, Variables et Indices	9
3. LE LANGAGE FORTRAN :	
Fonctions, Expressions et Formules arithmétiques	12
4. LE LANGAGE FORTRAN :	
Expressions de contrôle	19
5. LE LANGAGE FORTRAN :	
Entrées, Sorties	25
6. LE LANGAGE FORTRAN :	
Expressions d'organisation	35
7. DÉTAILS COMPLÉMENTAIRES SUR FORTRAN	38
8. EXEMPLES DE PROGRAMMATION FORTRAN	45
ANNEXE A. TABLE DES CARACTÈRES FORTRAN	50
B. TABLE DES EXPRESSIONS FORTRAN	51

CHAPITRE 1. GÉNÉRALITÉS SUR LE PROGRAMME ORIGINE FORTRAN

Le programme origine FORTRAN consiste en une séquence d'expressions FORTRAN. Il y a 32 types différents d'expressions qui sont décrits en détail dans les chapitres suivants.

Exemple de Programme Fortran

Un programme court servira à illustrer l'aspect et quelques propriétés du programme FORTRAN; il est donné tel qu'il se présente sur la feuille standard FORTRAN.

C ← POUR COMMENT.	SUITE	EXPRESSION FORTRAN	IDENTIFICATION		
			72	73	80
N°					
1	5	6	7		
C		PROGRAMME POUR TROUVER LA PLUS GRANDE VALEUR			
C	1	D UNE SERIE DE NOMBRES			
		AMAX=A(1)			
		FAIRE 20 I=2,N			
		SI (AMAX-A(I))10,20,20			
	10	AMAX=A(I)			
	20	CONTINUER			

Ce programme recherche parmi n valeurs a_i la quantité AMAX qui est la valeur du plus grand des a_i . Il commence, après la désignation du programme, par rendre AMAX égal à a_1 ; ensuite, l'expression FAIRE permet de répéter les expressions qui suivent jusqu'à et y compris l'expression 20, i prenant successivement les valeurs 2, 3, ..., n . A chaque exécution de ce groupe d'opérations, l'expression SI compare AMAX avec a_i . Si AMAX est inférieur à a_i , l'expression 10, qui donne à AMAX la valeur a_i , est exécutée avant de continuer.

Perforation du Programme

Chaque expression est perforée sur une carte séparée. Si une expression est trop longue pour être perforée dans une seule carte, la suite est perforée sur plusieurs autres cartes (9 si nécessaire) appelées cartes *Suite*. Pour chaque expression, la carte initiale doit contenir un 0 ou un blanc en colonne 6. Pour les cartes « Suite », la colonne 6 doit porter un nombre de 1 à 9.

Si une expression est trop longue pour tenir sur une seule ligne de la feuille de programmation, le programmeur peut signaler à la perforatrice qu'il a continué sur la ligne suivante en faisant une marque dans la colonne « Suite ».

L'ordre des expressions est simplement déterminé par l'ordre des cartes. Cependant, un nombre quelconque inférieur à 2^{16} (32768) peut être associé à une expression quelconque par perforation de celui-ci dans les colonnes 1 à 5 de la carte initiale. Ce nombre devient le *numéro de l'expression*. Ces numéros d'expression permettent de référencer le programme origine et aident le programmeur à trouver une correspondance entre ce programme et le programme résultant.

Les cartes portant un C en colonne 1 sont ignorées par FORTRAN. Ces cartes étant néanmoins imprimées, elles permettent l'insertion de commentaires.

Les colonnes 73 à 80 ne sont pas lues par FORTRAN et peuvent être perforées d'une identification quelconque.

Les expressions elles-mêmes sont perforées dans les colonnes 7 à 72, aussi bien dans les cartes initiales que dans les cartes « Suite ». Une expression comporte donc au maximum $10 \times 66 = 660$ caractères.

Les caractères blancs, sauf dans la colonne 6, sont ignorés par FORTRAN et le programmeur peut utiliser des blancs pour faciliter la rédaction de sa liste FORTRAN.

Le programme écrit en FORTRAN peut être introduit dans le traducteur à partir d'un paquet de cartes ou d'une bande DCB écrite sur le périphérique 704.

Expressions Fortran

Les 32 types d'expressions susceptibles d'être employés dans un programme FORTRAN peuvent être classés ainsi :

1. La *formule arithmétique*, qui permet, dans le programme résultant, de calculer une variable. Le chapitre 2 définit les symboles possibles pour les constantes et les variables, le chapitre 3 les combinaisons de ceux-ci dans les formules arithmétiques.
2. Les 15 *expressions de contrôle*, qui permettent d'établir l'ordinogramme du programme résultant. Elles sont décrites au chapitre 4.
3. Les 13 *expressions d'entrée et sortie*, qui sont nécessaires pour fournir au programme résultant ses sous-programmes d'entrée et sortie (Voir chapitre 5).
4. Les 3 *expressions d'organisation*, qui donnent différentes informations nécessaires ou souhaitables pour établir un programme efficace. Elles sont décrites au chapitre 6.

Enfin, les chapitres 7 et 8 fournissent quelques détails complémentaires sur différents sujets ainsi que des exemples de programmation FORTRAN.

CHAPITRE 2. LE LANGAGE FORTRAN : Constantes, Variables et Indices

Comme n'importe quel langage de programmation, FORTRAN permet d'exprimer les constantes numériques et les variables. Mais il donne aussi la possibilité d'employer des variables à 1, 2 ou 3 dimensions.

Constantes

Deux types de constantes sont permis : virgule fixe (sous forme d'entiers) et virgule flottante (caractérisée par l'emploi d'un point décimal).

Constantes en virgule fixe.

FORME	EXEMPLES
Un nombre de 1 à 5 chiffres décimaux précédé ou non du signe + ou - (sans point). La valeur absolue doit être inférieure à 32768.	3 + 1 - 28987

Toute constante positive peut être utilisée comme numéro d'expression.

Constantes en virgule flottante.

FORME	EXEMPLES
Un nombre décimal, avec un point remplaçant la virgule, en tête, à la fin ou entre deux chiffres. Un signe + ou - peut être utilisé. Un exposant qui doit être précédé d'un E peut suivre le nombre.	17. 5.0 -.0003 5.0E3 (= 5.0×10^3) 5.0E + 3 (= 5.0×10^3) 5.0E - 7 (= 5.0×10^{-7})

La valeur absolue du nombre doit être 0, ou un nombre compris entre 10^{-38} et 10^{+38} . Le nombre apparaît dans le programme résultant sous forme d'un nombre flottant normalisé en simple précision.

Variables

Deux types de variables sont permis : virgule fixe (sous forme d'entiers) et virgule flottante. Les variables à virgule fixe sont distinguées par le fait que leur premier caractère doit être I, J, K, L, M ou N.

Variables en virgule fixe.

FORME	EXEMPLES
1 à 6 caractères alphabétiques ou numériques (sauf les caractères spéciaux) dont le premier doit être I, J, K, L, M, ou N.	I M2 JOBNO

Une variable fixe peut prendre une valeur quelconque dont la valeur absolue est inférieure à 32768 (voir au chapitre 7 la section « Arithmétique fixe »).

REMARQUES. Le nom d'une variable ne peut être le même que le nom d'une fonction utilisée dans le programme après l'enlèvement de la lettre F terminale. De même, si une variable indicée a 4 caractères ou plus dans son nom, le dernier ne doit pas être un F (Voir plus loin la signification de fonction et de variable indicée).

Variables en virgule flottante.

FORME	EXEMPLES
1 à 6 caractères alphabétiques ou numériques (sauf les caractères spéciaux) dont le premier est alphabétique mais ne doit pas être I, J, K, L, M ou N.	A B7 DELTA

Une variable en virgule flottante peut prendre toute valeur pouvant être représentée par un nombre flottant 704 normalisé, c'est-à-dire zéro ou un nombre dont la valeur absolue est comprise entre 10^{-38} et 10^{+38} .

REMARQUE. Les restrictions concernant le symbole d'une variable en virgule fixe s'appliquent également aux symboles définissant une variable en virgule flottante.

Indices et Variables indicées

Un nombre quelconque d'un ensemble ou d'un tableau à 1, 2 ou 3 dimensions peut être représenté par une variable affectée de 1, 2 ou 3 *indices*; la variable est dite alors *indicée*. Les indices sont des quantités entières dont la valeur détermine la position du nombre dans l'ensemble.

Indices.

FORME	EXEMPLES
<p>v représentant une variable en virgule fixe et c (ou c') une constante positive en fixe, la forme de l'indice peut être : v</p> <p>c</p> <p>v + c ou v - c</p> <p>c*v</p> <p>c*v + c' ou c*v - c'</p>	<p>1</p> <p>3</p> <p>MU + 2</p> <p>MU - 2</p> <p>5*J</p> <p>5*J + 2</p> <p>5*J - 2</p>

Le symbole * signifie multiplication. La variable v ne peut être indicée. Le coefficient de l'indice est toujours positif.

Variables indicées.

FORME	EXEMPLES
<p>Variable fixe ou flottante suivie de 1, 2 ou 3 Indices, séparés par des virgules, et mis entre parenthèses.</p>	<p>A(1)</p> <p>K(3)</p> <p>BETA (5*J - 2, K + 2, L)</p>

Pour chaque variable qui apparaît sous forme indicée, la grandeur du tableau (c'est-à-dire le maximum des valeurs que peut atteindre le ou les indices) doit être définie dans une expression DIMENSION (Voir chapitre 6) précédant la première apparition de la variable.

La valeur minimum que peut prendre un indice dans un programme résultant est + 1.

Disposition d'un ensemble dans les mémoires.

Dans le programme résultant, un tableau à 2 dimensions A sera dans les mémoires en séquence dans l'ordre suivant : $A_{1,1}$, $A_{2,1}$, ..., $A_{m,1}$, $A_{1,2}$, $A_{2,2}$, ..., $A_{m,2}$, ..., $A_{m,n}$. Il est donc classé par colonne avec le premier des indices variant le plus rapidement et le dernier variant le moins rapidement. Il en est de même pour un tableau à trois dimensions. Pour un tableau à une dimension l'ordre est simplement en séquence. Le premier terme du tableau est mis en mémoire dans la mémoire d'adresse la plus élevée, et les termes suivants sont dans les mémoires d'adresses décroissantes. Donc, les termes sont rangés dans l'ordre décroissant des adresses des mémoires.

CHAPITRE 3. LE LANGAGE FORTRAN :

Fonctions, Expressions et Formules arithmétiques

Parmi les différentes expressions FORTRAN, la *formule arithmétique* définit un calcul numérique que doit exécuter le programme résultant. La formule arithmétique FORTRAN est très voisine des formules arithmétiques classiques; elle est composée d'une variable à calculer, suivie du signe =, suivi lui-même de l'*expression* arithmétique définissant la valeur de la variable. Par exemple, la formule arithmétique :

$$Y = A - \text{SINF}(B - C)$$

signifie « remplacer la valeur de y par la valeur de a—sin(b—c) ».

Fonctions

Comme le montre l'exemple ci-dessus, une expression FORTRAN peut inclure le nom d'une *fonction* (par exemple, la fonction sinus: SINF) à condition que son évaluation puisse être faite soit à l'aide d'une *expression fonction*, soit à l'aide d'un sous-programme préétabli en langage 704 et placé sur la bande maîtresse FORTRAN.

FORME	EXEMPLES
<p>Le symbole d'une fonction comprend 4 à 7 caractères alphabétiques ou numériques (sauf les caractères spéciaux) dont le dernier doit être un F et le premier alphabétique. Un X en tête du symbole donne automatiquement la valeur de la fonction en fixe.</p> <p>Le nom de la fonction est suivi par des parenthèses contenant les arguments (qui peuvent être des expressions) séparés par des virgules.</p>	<p>SINF(A + B) RACF(SINF(A)) ENTF(3.*X) XTANF(3.*X)</p>

Type arithmétique d'une fonction et de ses arguments. Une fonction à un seul argument peut être définie de telle façon que son argument soit en fixe ou en flottant; de même, la fonction peut être, elle aussi, en fixe ou en flottant.

Ainsi, une fonction à un seul argument a quatre types arithmétiques possibles; en général, une fonction de n arguments aura donc 2^{n+1} types.

Un nom spécial doit être donné et un sous-programme doit être disponible pour chaque type employé. Une série complète de sinus pourra être faite de la manière suivante :

SINOF Argument fixe, fonction flottante
 SIN1F Argument flottant, fonction flottante
 XSINOF Argument fixe, fonction fixe
 XSIN1F Argument flottant, fonction fixe

où X et F sont obligatoires, mais où le reste du nom est arbitraire.

Fonctions définies dans la bande maîtresse FORTRAN.

1° Fonctions incluses dans le programme. Le traducteur FORTRAN possède, pour évaluer un certain nombre de fonctions, des sous-programmes ouverts. En voici la liste :

TYPE DE FONCTION	DEFINITION	NOMBRE D'ARG.	ARGUMENT	RÉSULTAT
Valeur absolue	ABSF	1	FL	FL
	XABSF	1	FX	FX
Entier	ENTF	1	FL	FL
	XENTF	1	FL	FX
X Modulo Y	MODF	2	FL	FL
	XMODF	2	FX	FX
Maximum	MAX0F	≥ 2	FX	FL
	MAX1F	≥ 2	FL	FL
	XMAX0F	≥ 2	FX	FX
	XMAX1F	≥ 2	FL	FX
Minimum	d° que ci-dessus avec MIN à la place de MAX			
Flottant	FLOTF	1	FX	FL
Fixation	XFIXF	1	FL	FX
X Signe Y	SIGNF	2	FL	FL
	XSIGNF	2	FX	FX

NOTE. La fonction modulo doit être interprétée de la façon suivante : MODF (X, Y) = X - [X/Y] Y où [a] signifie partie entière de a.

La fonction signe doit être interprétée ainsi : SIGN (X, Y) = |X| signe (Y).

Ce type comprend 18 fonctions toujours assemblées dans le programme comme des sous-programmes ouverts.

2° *Fonctions de la bibliothèque.* En plus des sous-programmes ouverts, des fonctions quelconques uniformes avec un nombre quelconque d'arguments peuvent être incorporées par le programmeur sur la bande maîtresse et être disponibles pour la programmation. De tels sous-programmes seront assemblés dans le programme résultant comme des sous-programmes fermés. La section « Écriture des sous-programmes », au chapitre 7, donne les spécifications d'un sous-programme et le manuel d'utilisation explique comment ce sous-programme est mis sur la bande.

Les fonctions disponibles sur la bande sont les suivantes :

TYPE DE FONCTION		DÉSIGNATION	
Fonctions trigonométriques		SINF(X)	sin x
		COSF(X)	cos x
		TANF(X)	tg x
	$ X \leq 1$	ASINF(X)	arcsin x
	$ X \leq 1$	ACOSF(X)	arccos x
		ATANF(X)	arctg x
Fonction hyperbolique		TANHF(X)	th x
Fonctions logarithmique et exponentielle	$0 < X$	LOGF(X)	Log x
		EXPF(X)	e^x
Racine carrée	$0 \leq X$	RACF(X)	\sqrt{x}
Fonctions de Bessel	$0 \leq X$	BJ0F(X)	$J_0(x)$
	$0 \leq X$	BJ1F(X)	$J_1(x)$
	$0 \leq X \leq 88$	BI0F(X)	$I_0(x)$
	$0 \leq X \leq 88$	BI1F(X)	$I_1(x)$
	$0 < X \leq 88$	BK0F(X)	$K_0(x)$
	$0 < X \leq 88$	BK1F(X)	$K_1(x)$
	$0 < X$	BY0F(X)	$Y_0(x)$
	$0 < X$	BY1F(X)	$Y_1(x)$
Fonction Gamma	$X \neq$ entier négatif	GAMF(X)	$\Gamma(x+1)$

NOTE. La fonction Log x est le logarithme naturel, c'est-à-dire Népérien.

Expressions Fonctions.

Une fonction peut aussi être définie dans le programme origine lui-même au moyen d'une expression fonction et la définition sera conservée tout au cours de ce programme. Ainsi, il est possible d'avoir à sa disposition des fonctions avec les notations que l'on désire, fonctions qui ne sont pas assez importantes pour être placées sur la bande maîtresse FORTRAN et qui n'intéressent que des problèmes particuliers.

DÉFINITION	EXEMPLES
<p>« a = b » où a est un nom de fonction suivi entre parenthèses des arguments (qui doivent être des variables différentes non indicées) séparés par des virgules et b une expression arithmétique (voir la suite de ce chapitre) qui ne comprend pas de variables indicées. Les fonctions qui peuvent être utilisées dans b sont ou des fonctions ouvertes, ou des fonctions de la bande maîtresse, ou des fonctions déjà définies par des expressions fonctions.</p>	<p>PREMF(X) = A*X + B SECONDNF(X,B) = A*X + B TROISF(I,A) = 3.0*A**I QUATF(E,D) = PREMF(E)/D CINQF(J) = J + K SINH(X) = 0,5* (EXPF(X) - 1./EXPF(X))</p>

Comme pour une fonction, le résultat peut être fixe ou flottant suivant que le nom commence ou non par un X.

La partie droite d'une expression fonction doit être une expression qui n'a pas de variable indicée et peut contenir tout ce qui a été défini dans les précédentes sections; en particulier il est possible d'employer les fonctions déjà définies et même d'employer des fonctions précédemment définies par une autre expression fonction.

Il est possible de choisir autant de variables que l'on désire pour une telle expression; elles doivent être placées dans la partie gauche entre parenthèses pour définir les arguments de la fonction qui seront transformés en une autre expression. On obtient ainsi un modèle d'expression extrêmement souple pour les fonctions qui intéressent particulièrement un problème; les noms des arguments n'ont aucune importance (excepté l'indication du mode d'expression fixe ou flottant) et peuvent être les mêmes que des noms déjà employés au cours du programme.

Les variables de la partie droite qui ne sont pas définies comme arguments sont traitées comme paramètres. Ainsi, si PREMF est défini par l'expression fonction

$$\text{PREMF}(X) = A * X + B$$

la référence PREMF (Y) permet de calculer $a y + b$ à partir des valeurs actuelles de a, de b et de y. Toutefois, le nom des paramètres doit suivre les règles d'une seule définition.

Une fonction définie par l'expression fonction peut être utilisée exactement comme une autre fonction. En particulier, ses arguments peuvent être des expressions et peuvent contenir des variables indicées. Suivant la définition précédente, on pourrait donc écrire PREMF (Z + Y (I)) qui calculerait $a (z + y_i) + b$ à partir des valeurs de a, b, y_i et z.

Les fonctions définies par l'expression fonction sont toujours assemblées comme des sous-programmes fermés.

REMARQUE IMPORTANTE. Les expressions fonctions d'un programme doivent précéder la première expression exécutable du programme.

Expressions arithmétiques

Une expression arithmétique est composée d'une suite quelconque de constantes, de variables (indicées ou non), de fonctions séparées par des symboles opérations, des virgules, des parenthèses.

Cependant, une expression FORTRAN peut être soit en fixe, soit en flottant, mais ne peut être une expression mixte. C'est-à-dire qu'une quantité flottante ne peut être incorporée dans une expression en fixe et vice versa. Toutefois, une quantité d'un certain type peut apparaître dans une expression d'un autre type dans certains cas particuliers. Une quantité flottante peut apparaître dans une expression fixe seulement comme argument d'une fonction. Une quantité fixe peut apparaître dans une expression flottante comme argument de fonction, comme indice ou comme exposant.

Règles pour la formation des expressions arithmétiques. De l'application des règles suivantes, on peut déduire toutes les expressions permises.

1. Les constantes, variables, variables indicées en virgule fixe (ou en virgule flottante) sont des expressions du même type. Ainsi, 3 et I sont des expressions fixes et ALPHA et A(I,J,K) sont des expressions flottantes.
2. Si FONCF est une fonction de n variables et si E, F, ..., H est un jeu de n expressions du type correct pour FONCF, FONCF (E, F, ..., H) est une expression du même type que FONCF.
3. Si E est une expression dont le premier caractère n'est ni + ni —, + E et — E sont alors permis et sont du même type que E. Par contre, — A étant une expression, + — A n'est pas une expression.
4. Si E est une expression, (E) est une expression du même type que E. Ainsi (A), ((A)), (((A))), etc, sont des expressions.
5. Si E et F sont des expressions du même type et si le premier caractère de F n'est ni +, ni — :

$$\begin{aligned} E + F \\ E - F \\ E * F \\ E / F \end{aligned}$$

sont des expressions du même type, mais $A - + B$ et $A / + B$ ne sont pas des expressions. Les caractères +, —, *, et / signifient addition, soustraction, multiplication et division.

6. Si E et F sont des expressions, si F n'est en virgule flottante que si E l'est aussi, si le premier caractère de F n'est ni + ni — et si ni E ni F ne sont de la forme $A^{**}B$, alors :

$$E^{**}F$$

est une expression du même type que E. De même, $A^{**}(B^{**}C)$ est une expression, mais $I^{**}(B^{**}C)$ et $A^{**}B^{**}C$ ne sont pas des expressions. Le symbole ** signifie exponentiation, c'est-à-dire que $A^{**}B$ signifie A^B .

Hierarchie des opérations. Quand la hiérarchie des opérations d'une expression n'est pas complètement définie par des parenthèses, il est sous-entendu

qu'elle s'effectue dans l'ordre suivant (en partant des opérations les plus intérieures vers les plus extérieures) :

Exponentiation
 Multiplication et Division
 Addition et Soustraction

Par exemple, l'expression :

$$A + B/C + D**E*F - G$$

signifie :

$$A + (B/C) + (D*E*F) - G$$

Ordre à l'intérieur d'une hiérarchie. Les parenthèses qui ont été omises dans une suite de multiplications et divisions consécutives ou additions et soustractions consécutives, seront sous-entendues comme étant groupées à partir de la gauche. Ainsi, si . représente * ou / (ou bien + ou -), alors

$$A.B.C.D.E$$

signifie :

$$(((A.B).C).D).E$$

Vérification des parenthèses. La règle suivante est suggérée pour la vérification des parenthèses dans une expression compliquée.

Numéroter la première parenthèse 1, puis, en travaillant de gauche à droite, augmenter le numéro de 1 pour chaque parenthèse ouverte et diminuer de 1 pour chaque parenthèse fermée. Le numéro de la dernière parenthèse doit être 0. La parenthèse de fermeture d'une parenthèse ouverte numérotée n doit être la parenthèse suivante numérotée n-1.

Optimisation des expressions arithmétiques. L'efficacité du programme résultant après traduction des expressions arithmétiques dépend particulièrement du mode d'écriture. La section sur l'optimisation des expressions arithmétiques au chapitre 7 mentionne quelques considérations qui influent sur l'efficacité du programme résultant.

Formules arithmétiques

FORME	EXEMPLES
« a = b » où a est une variable (indicée ou non) et b une expression arithmétique.	$A(I) = B(I) + \text{SINF}(C(I))$

Dans la formule arithmétique a = b, le signe = signifie « a doit être remplacé par b ». Une expression arithmétique provoque le calcul de la valeur de la partie droite et la mise en mémoire de cette valeur à la place de la variable définie dans la partie gauche.

Le résultat est mis en mémoire sous forme fixe ou flottante suivant que le type de la variable de gauche est fixe ou flottant.

Si la variable à gauche est fixe et que l'expression à droite est flottante, le résultat sera d'abord calculé en flottant, ensuite tronqué et enfin mis en fixe. Ainsi, si le résultat est $\pm 3,569$ le nombre mis en mémoire ne sera pas ± 4 mais ± 3 .

Exemple.

FORMULE	SIGNIFICATION
A = B	Mettre la valeur de B à la place de A.
I = B	B est tronqué, converti en fixe et mis à la place de I.
A = I	I est converti en flottant puis mis à la place de A.
I = I + 1	1 est ajouté à I et le résultat mis à la place de I.
	Cet exemple montre bien qu'une formule arithmétique n'est pas une équation mais le remplacement d'une valeur par une autre.
A = MAX1F(SINF(B), COSF(B))	A est remplacé par la plus grande des quantités sin B et cos B. Cet exemple montre l'utilisation d'une fonction comme argument d'une fonction.
A = 3.0*B	A est remplacé par 3B.
A = 3*B	Incorrect. Cette expression est mixte.
A = I*B	Incorrect. Cette expression est mixte.

CHAPITRE 4. LE LANGAGE FORTRAN :

Expressions de contrôle

La deuxième série des expressions FORTRAN est formée de 15 expressions de contrôle qui permettent au programmeur d'établir le déroulement de son programme.

ALLER A inconditionnel

FORME	EXEMPLES
« ALLER A n » où n est le numéro d'une expression.	ALLER A 5

L'instruction suivante exécutée est l'expression n.

ALLER A imposé

FORME	EXEMPLES
« ALLER A v, (n ₁ , n ₂ , ..., n _m) » où v est une variable en virgule fixe non indicée qui a été définie au préalable dans une expression IMPOSER EN, et n ₁ , n ₂ , ..., n _m sont des numéros d'expressions.	ALLER A N, (7, 12, 19)

Cette expression provoque le transfert à une expression dont le numéro est égal à la valeur donnée à v par le dernier IMPOSER exécuté. Les n₁, n₂, ..., n_m sont une liste de valeurs que v peut prendre.

Cette expression ALLER A est utilisée pour des embranchements prédéterminés.

REMARQUE. Quand un ALLER A se trouve dans un FAIRE, il faut tenir compte de certaines conditions pour n₁, n₂, ..., n_m (Voir la section FAIRE de ce chapitre).

IMPOSER

FORME	EXEMPLES
« IMPOSER n EN v » où n est un numéro d'expression et v une variable en virgule fixe non indicée.	IMPOSER 12 EN N

Cette expression provoque, par un ALLER A v, (n₁, n₂, ..., n_m) ultérieur, le transfert à l'expression dont le numéro est n; n doit être une des valeurs n₁, n₂, ..., n_m.

L'expression IMPOSER 12 EN N et la formule arithmétique N = 12 ne sont pas équivalentes. Une variable dont on a imposé la valeur ne peut être utilisée que pour un ALLER A imposé jusqu'à ce qu'elle soit rétablie comme une variable ordinaire.

**ALLER A
calculé**

FORME	EXEMPLES
« ALLER A (n_1, n_2, \dots, n_m), i » où n_1, n_2, \dots, n_m sont des numéros d'expression et i une variable en virgule fixe non indiquée.	ALLER A(30, 40, 50, 60), I

Si au moment de l'exécution la valeur de la variable i est k , il y a alors transfert à l'expression n_k . Ainsi, dans l'exemple, si I a la valeur 3 au moment de l'exécution, il y a transfert à 50. Cette expression est utilisée lorsque les différents cas possibles sont calculés.

SI

FORME	EXEMPLES
« SI (a) n_1, n_2, n_3 » où a est une expression arithmétique et n_1, n_2, n_3 des numéros d'expressions.	SI (A(J,K) - B) 10, 20, 30

Il y a transfert à n_1, n_2, n_3 , suivant que l'expression arithmétique est inférieure, égale ou supérieure à 0.

VOYANT

FORME	EXEMPLES
« VOYANT i » où i est 0, 1, 2, 3 ou 4.	VOYANT 3

Si i est 0, tous les voyants du pupitre 704 sont éteints; sinon c'est le voyant i qui est allumé.

**SI
(VOYANT)**

FORME	EXEMPLES
« SI (VOYANT i) n_1, n_2 » où n_1, n_2 sont des numéros d'expressions et i est 1, 2, 3 ou 4.	SI (VOYANT 3) 30, 40

Il y a transfert à l'expression n_1 ou n_2 suivant que le voyant i est allumé ou éteint et le voyant s'éteint.

**SI
(CLE)**

FORME	EXEMPLES
« SI (CLE i) n_1, n_2 » où n_1 et n_2 sont des numéros d'expressions et i est 1, 2, ..., ou 6.	SI (CLE 6) 30, 40

Il y a transfert à l'expression n_1 ou n_2 suivant que la clé i du pupitre de commande du 704 est abaissée ou non.

SI DC AC

FORME	EXEMPLES
« SI DC AC n_1, n_2 » où n_1 et n_2 sont des numéros d'expressions.	SI DC AC 30, 40

Il y a transfert à l'expression n_1 ou n_2 suivant que l'indicateur de dépassement de capacité de l'accumulateur est en fonction ou non, et l'indicateur est restauré.

SI DC MQ

FORME	EXEMPLES
« SI DC MQ n_1, n_2 » où n_1 et n_2 sont des numéros d'expressions.	SI DC MQ 30, 40

Il y a transfert à l'expression n_1 ou n_2 suivant que le dépassement de capacité du multiplicateur-quotient est en fonction ou non, et l'indicateur est restauré.

SI DIV IMP

FORME	EXEMPLES
« SI DIV IMP n_1, n_2 » où n_1 et n_2 sont des numéros d'expressions.	SI DIV IMP 30, 40

Il y a transfert à l'expression n_1 ou n_2 suivant que l'indicateur de division impossible est en fonction ou non, et l'indicateur est restauré.

PAUSE

FORME	EXEMPLES
« PAUSE » ou « PAUSE n » où n est un nombre entier positif exprimé en octal.	PAUSE PAUSE 7777

La machine s'arrêtera avec le nombre octal n dans le registre mémoire au pupitre. (Si n n'est pas donné, il y aura 0 dans le RM). Si on appuie sur le bouton DÉPART le programme continue en prenant l'expression FORTRAN suivante.

ARRÊT

FORME	EXEMPLES
« ARRÊT » ou « ARRÊT n » où n est un nombre entier positif exprimé en octal.	ARRÊT ARRÊT 7777

Cette expression provoque l'arrêt de la machine sans possibilité de départ en appuyant sur le bouton DÉPART. Cet ordre est utilisé chaque fois que le programmeur désire arrêter définitivement la machine. Le nombre octal n apparaît dans le registre mémoire au pupitre de commande. (Si n n'est pas donné, on a 0).

FAIRE

FORME	EXEMPLES
« FAIRE n $i = m_1, m_2$ » ou « FAIRE n $i = m_1, m_2, m_3$ » où n est un numéro d'expression, i est une variable en virgule fixe non indiquée, et m_1, m_2, m_3 sont chacun soit une constante positive en virgule fixe, soit une variable non indiquée en virgule fixe. Si m_3 n'est pas défini, il est pris égal à 1.	FAIRE 30 $i = 1, 10$ FAIRE 30 $i = 1, M, 3$

L'expression FAIRE signifie « FAIRE les expressions qui suivent, jusqu'à et y compris l'expression n , et les répéter la première fois avec $i = m_1$ puis en augmentant i chaque fois de m_3 , c'est-à-dire $m_1 + m_3, m_1 + 2m_3, \dots$ jusqu'à ce que i prenne la plus grande valeur de cette suite qui n'excède pas m_2 ».

La *boucle* d'un FAIRE est le groupe d'expressions qui sont répétées. C'est donc la séquence des expressions qui suivent FAIRE jusqu'à et y compris l'expression n .

L'*index* d'un FAIRE est la variable fixe i contrôlée par FAIRE et qui prend successivement les valeurs $m_1, m_1 + m_3, m_1 + 2m_3, \dots$, dont la dernière n'excède pas m_2 . Au cours de la boucle, i est disponible pour les calculs comme variable en virgule fixe habituelle ou comme variable d'un indice. Après la dernière exécution de la boucle, le FAIRE est dit *satisfait*.

Prenons un exemple :

```
10  FAIRE 11  $i = 1, 10$ 
11  A(I) = I*N(I)
12
```

La boucle de ce FAIRE est réduite à la seule expression 11, et l'index est I. Le FAIRE prend I égal à 1 et on entre dans la boucle. $1 \times N(1)$ est calculé, converti en virgule flottante et mis en mémoire dans A(1). Ensuite, l'expression étant la dernière de la boucle et le FAIRE n'étant pas satisfait, I prend la valeur 2 et on recommence la boucle, c'est-à-dire l'expression 11. $2 \times N(2)$ est calculé et mis dans la mémoire A(2). Et ainsi de suite jusqu'au moment où la boucle est exécutée avec $I = 10$; l'expression suivante est alors 12.

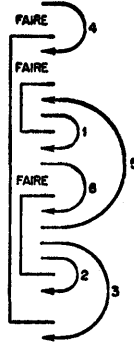
FAIRE à l'intérieur de FAIRE. Parmi les expressions d'une boucle FAIRE, il peut y avoir des expressions FAIRE. Dans ce cas, la règle suivante doit être observée :

Règle 1. Si la boucle d'un FAIRE contient un autre FAIRE, toutes les expressions de la boucle de ce dernier doivent être aussi dans la boucle du premier.

Le deuxième FAIRE est dit inscrit dans le premier.

Transfert de contrôle et expression FAIRE. Les transferts de contrôle par SI ou ALLER A sont sujets à la règle suivante :

Règle 2. Aucun transfert n'est permis dans la boucle d'un FAIRE de l'extérieur de cette boucle. Ainsi, dans la figure suivante 1, 2 et 3 sont des transferts permis mais 4, 5 et 6 ne le sont pas.



EXCEPTION. A partir d'une boucle FAIRE, il est possible de sortir pour exécuter un sous-programme ne changeant pas les index et de rentrer dans cette boucle à une expression quelconque excepté la première.

Règle pour un ALLER A imposé dans la boucle d'un FAIRE. Quand un ALLER A imposé existe à l'intérieur d'une boucle FAIRE, toutes les expressions auxquelles renvoie le transfert doivent être toutes dans la boucle exclusive d'un seul FAIRE. La boucle exclusive d'un FAIRE est constituée par les expressions de la boucle qui ne font pas partie d'une autre boucle.

Conservation des valeurs d'index. Quand le contrôle quitte la boucle d'un FAIRE de la manière normale (c'est-à-dire le FAIRE étant satisfait, le contrôle passant à l'expression suivant la dernière expression de la boucle), la sortie est dite normale. Après une sortie normale d'une boucle FAIRE, la valeur de l'index contrôlée par ce FAIRE n'est pas définie et l'index ne peut être utilisé que lorsqu'il est à nouveau défini (Voir les compléments de FAIRE au chapitre 7).

Cependant, si la sortie s'est faite par un transfert à l'extérieur de la boucle, la valeur actuelle de l'index se trouve disponible pour une utilisation ultérieure; si la sortie a lieu par un transfert situé dans les boucles de plusieurs FAIRE, les valeurs définies des index contrôlés par les FAIRE sont disponibles.

Restrictions aux calculs à l'intérieur d'une boucle FAIRE. La première expression d'une boucle FAIRE ne doit pas être une expression non exécutable. La dernière expression d'une boucle FAIRE ne doit pas être une expression de transfert. Dans ce cas, c'est l'expression CONTINUER, décrite au paragraphe suivant, qui permettra de terminer la boucle.

Tous les calculs sont permis à l'intérieur d'une boucle FAIRE sauf les calculs qui redéfiniraient l'index ou les paramètres d'indexage (m). Il faut donc que l'indexage du FAIRE soit complètement défini avant l'entrée dans la boucle.

CONTINUER

FORME	EXEMPLES
« CONTINUER »	CONTINUER

CONTINUER est une expression qui ne donne naissance à aucune instruction dans le programme résultant. Elle est le plus fréquemment utilisée comme dernière expression d'une boucle FAIRE où elle fournit un numéro d'expression auquel il est possible de se référer pour les transferts.

Comme exemple de programme demandant une expression CONTINUER, on peut considérer la Consultation de table suivante :

```
10 FAIRE 12 I = 1,100
11 SI (ARG - VAL (I)) 12, 20, 12
12 CONTINUER
13
```

Ce programme examinera la table des 100 valeurs VAL jusqu'à ce qu'il en trouve une égale à l'argument ARG. A ce moment, il se transférera à l'expression 20; la valeur de I correspondant à ce point est alors fixée et disponible. S'il n'y a dans la table aucune valeur égale à l'argument, l'expression 13 sera prise en séquence. Le programme :

```
10 FAIRE 11 I = 1,100
11 SI (ARG - VAL (I)) 11, 20, 11
12
```

ne serait pas correct; la boucle FAIRE ne peut être réalisée si la dernière expression est un tranfert.

Sommaire de la séquence Fortran

Les règles précises qui régissent l'ordre dans lequel les expressions d'un programme FORTRAN seront exécutées peuvent être résumées comme suit :

1. Le contrôle commence à la première expression exécutable.
2. Si le contrôle exécute l'expression E, il prendra comme expression suivante l'expression qui fait suite à l'expression E suivant les propriétés de cette dernière (Voir Annexe B).
3. EXCEPTION. Si, cependant, E est la dernière expression de la boucle d'un ou plusieurs FAIRE non satisfaits et n'est donc pas une expression de transfert (SI, ALLER A), le contrôle ne tient pas compte des propriétés particulières de l'expression, il se transfère à la première instruction du FAIRE non satisfait le plus proche et l'index de ce FAIRE est augmenté.

NOTE. Les expressions MODÈLE, DIMENSION, ÉQUIVALENCE et FRÉQUENCE qui sont traitées dans les deux prochains chapitres sont des expressions non exécutables qui sont simplement ignorées au point de vue contrôle.

REMARQUE IMPORTANTE. Chaque expression exécutable du programme origine doit faire partie d'une branche de l'Ordinogramme, sinon des erreurs pourraient se produire au moment de l'assemblage du programme résultant.

La dernière expression exécutable du programme origine doit être une instruction ARRÊT (et non PAUSE) ou une instruction de transfert (SI, ALLER A).

CHAPITRE 5. LE LANGAGE FORTRAN :

Entrées, Sorties

Il existe 13 expressions FORTRAN pour définir la transmission de l'information pendant l'exécution du programme résultant entre les mémoires à ferrites d'une part, et les bandes, les tambours, le lecteur, la perforatrice et la tabulatrice du 704, d'autre part. Ces 13 expressions peuvent se grouper de la manière suivante :

1. Cinq expressions (LIRE, PERFORER, IMPRIMER, INSCR SORTIE BANDE et LIRE ENTRÉE BANDE) permettent la transmission d'une liste de quantités entre les ferrites et un milieu extérieur — cartes, imprimés ou bandes — dans lequel l'information est exprimée respectivement sous forme de perforations Hollerith, d'impressions ou de codes DCB.
2. Une expression (MODÈLE) non exécutable définit la forme de l'information dans le milieu extérieur pour les 5 expressions précédentes.
3. Quatre expressions (LIRE BANDE, LIRE TAMBOUR, INSCR BANDE et INSCR TAMBOUR) permettent la transmission d'informations sous forme binaire d'une liste de quantités.
4. Enfin, 3 expressions (FIN BLOC, REBOBINER et ESP ARR) sont utilisées pour la manipulation des bandes.

Ce chapitre décrit la liste des quantités à transmettre (cette liste devant apparaître dans chacune des 9 expressions des groupes 1 et 3 ci-dessus), la méthode d'écriture de l'expression MODÈLE, la forme sous laquelle doivent être fournies les données au programme résultant et, enfin, les expressions des groupes 1, 3 et 4.

Description de la liste des quantités

Chacune des 9 expressions qui permettent de transmettre des informations comprennent une liste des quantités à transmettre. Cette liste est ordonnée et son ordre doit être le même que celui des mots existants (pour les entrées) ou à sortir (pour les sorties) dans le milieu extérieur.

L'exemple suivant fera mieux comprendre la description et la signification d'une liste.

A, B(3), (C(I), D(I,K), I = 1,10), ((E(I,J), I = 1,10,1), F(J,3), J = 1,K)

Si cette liste est utilisée avec une expression de sortie, l'information est alors écrite dans le milieu extérieur dans l'ordre suivant :

A, B(3), C(1), D(1,K), C(2), D(2,K), ..., C(10), D(10,K),
 E(1,1), E(2,1), ..., E(10,1), F(1,3),
 E(1,2), E(2,2), ..., E(10,2), F(2,3), ..., F(K,3).

De la même manière, si cette liste était utilisée avec une expression d'entrée, les mots successifs qui seraient lus à partir du milieu extérieur seraient placés dans l'ordre précédent des mémoires données.

La liste est donc lue de la gauche vers la droite, avec répétition des variables entre parenthèses. Cette répétition est exactement du même type que les répétitions du type FAIRE. A chaque parenthèse ouverte (excepté les parenthèses de variables indicées), on pourrait associer un FAIRE avec l'indice donné immédiatement avant la fermeture de la parenthèse et avec une boucle sur toutes les informations indicées. Ainsi, l'ordre de la liste ci-dessus aurait été le même que si l'on avait donné le « programme » suivant :

```

1 A
2 B(3)
3 FAIRE 5 I = 1, 10
4 C(I)
5 D(I,K)
6 FAIRE 9 J = 1,K
7 FAIRE 8 I = 1, 10, 1
8 E(I,J)
9 F(J,3)

```

Seules les variables (et non les constantes) peuvent être listées.

A noter que les paramètres d'indexage, comme dans les boucles FAIRE, sont au nombre de 3 (constantes ou variables fixes) et que le dernier, en cas d'omission, est pris égal à 1.

Pour une liste de la forme K, A(K) ou K, (A(I), I = 1,K) dans laquelle l'indice ou le paramètre d'indexage lui-même apparaissait déjà dans la liste des expressions d'entrée, l'indexage est fait avec la valeur qui vient d'être lue.

Quand on désire transmettre un tableau complet dans l'ordre inverse de l'ordre interne ($A_{m,n}, \dots, A_{m,2}, \dots, A_{2,2}, A_{1,2}, A_{m,1}, \dots, A_{2,1}, A_{1,1}$, etc.), la notation abrégée peut être utilisée. Il suffit de donner le nom de la variable indicée et d'omettre l'information d'indexage. FORTRAN recherchera si dans une expression DIMENSION le nom de la variable apparaît. (Voir chapitre 6). Dans l'affirmative, l'information d'indexage pour transmettre le tableau complet dans l'ordre inverse de l'ordre interne sera fournie automatiquement; sinon, seule une variable sans dimension sera transmise. Dans l'exemple précédent, A était une variable simple, mais s'il avait été une variable indicée, le tableau complet aurait été transmis.

REMARQUE. Les indications données dans cette section ne s'appliquent intégralement qu'aux listes données avec les expressions du groupe 1 pour la transmission d'informations sous forme décimale. Pour les expressions transmettant des informations binaires INSCR TAMBOUR et LIRE TAMBOUR, les notations abrégées qui viennent d'être mentionnées sont seules permises; voir à la section « Liste des bandes binaires » du chapitre 7 l'analyse détaillée des expressions LIRE BANDE et INSCR BANDE.

MODÈLE

FORME	EXEMPLES
« MODÈLE (Spécification) » voir ci-dessous.	MODÈLE (I2/(E12.4, F10.4))

Les 5 expressions d'entrée et de sortie décimales du groupe 1 contiennent, en plus de la liste des quantités à transmettre, un numéro d'expression MODÈLE. Ce numéro décrit le modèle suivant lequel l'information existe ou doit être produite dans le milieu extérieur, ainsi que le type de conversion à effectuer entre le binaire et le décimal. L'expression MODÈLE n'est pas exécutable, sa fonction est de fournir un renseignement au programme « résultant ». C'est pourquoi elle peut être placée à un endroit quelconque du programme « origine ».

La spécification. Pour donner plus de précision à la spécification de MODÈLE, nous prendrons comme exemple une expression d'impression. Cependant, la description reste valable pour les autres cas, en généralisant le concept de ligne imprimée à celui d'unité d'enregistrement du milieu extérieur. Ainsi, une unité d'enregistrement peut être :

1. Une ligne imprimée avec 120 caractères.
2. Une carte perforée de 72 caractères.
3. Un enregistrement sur bande décimale DCB avec 120 caractères maximum.

Les 3 types de conversion décimal-binaire ou binaire-décimal sont les suivants :

	INTÉRIEUR	EXTÉRIEUR
E	FL	FL
F	FL	FX
I	Entier	Entier

La spécification MODÈLE décrit la ligne à imprimer en donnant pour chaque zone de la ligne (depuis la gauche jusqu'à la droite, en commençant par la première roue d'impression) :

1. Le type de conversion à utiliser (E, F, ou I).
2. La dimension (w) de la zone.
3. Pour les conversions E et F, le nombre de décimales (d) après la virgule qui doivent être imprimées après arrondi. Si d n'est pas inférieur à 10, il est traité modulo 10.

Les spécifications de base des zones sont donc de la forme :

Iw, Ew.d, et Fw.d

les différentes spécifications de zones étant séparées par des virgules. Ainsi, l'expression MODÈLE (I2, E12.4, F10.4) doit donner la ligne :

27 -0.9321E 02 -0.0076

Comme dans cet exemple, la dimension des zones peut être prise plus grande que nécessaire pour assurer un espacement entre les nombres. Dans ce cas, il y a un blanc après 27, un blanc automatique après E et 3 blancs après 02. A l'intérieur de chaque zone, la partie à imprimer est toujours cadrée le plus à droite possible.

Si l'on désire imprimer n zones successives de la même façon, on place n devant E, F, ou I. Ainsi l'expression MODÈLE (I2, 3E12.4) doit donner :

27 -0.9321E 02 -0.7580E-02 0.5536E 00

Pour permettre la répétition de groupes de spécifications de zones, une expression avec parenthèses est permise. Ainsi, l'expression MODÈLE (2(F10.6, E10.2), I4) est équivalente à MODÈLE (F10.6, E10.2, F10.6, E10.2, I4). Rien n'est prévu dans le cas de parenthèses à l'intérieur de parenthèses.

Pour permettre une utilisation plus générale de la conversion du type F, un facteur de cadrage suivi de la lettre P peut précéder la spécification. Le facteur de cadrage est défini de telle sorte que :

$$\text{Nombre imprimé} = \text{Nombre interne} \times 10 \text{ facteur de cadrage}$$

L'expression MODÈLE (I2, 1P3F11.3) utilisée avec les données du précédent exemple, donnera donc :

27 -932.096 -0.076 5.536

tandis que MODÈLE (I2, -1P3F11.3) donnera :

27 -9.321 -0.001 0.055

Un facteur de cadrage positif peut aussi être utilisé avec la conversion E pour augmenter le nombre et diminuer l'exposant. Ainsi, MODÈLE (I2, 1P3E12.4) donnera pour les mêmes nombres :

27 -9.3210E 01 -7.5804E-03 5.5361E-01

NOTE. Le facteur de cadrage est mis automatiquement à zéro si aucune valeur n'est donnée. Cependant, quand une valeur est donnée, elle est maintenue pour toutes les conversions F,E jusqu'à ce qu'une nouvelle valeur soit donnée. Le facteur de cadrage n'a aucun effet sur la conversion I.

Zones Hollerith. Ce sont des zones contenant des caractères spéciaux, alphabétiques ou numériques, où ne figure aucune spécification de nombres calculés ou destinés à être calculés. La définition est caractérisée par la présence d'un H précédé du nombre de caractères et suivi de ces caractères disposés suivant la présentation désirée. Par exemple, MODÈLE (4H XY = F8.3, 4H Z = F6.2, 7H W/AF = F7.3) donnera :

XY = -93.210 Z = -0.01 W/AF = 0.554

A noter que tous les caractères Hollerith, blancs inclus, peuvent être imprimés. C'est le seul cas où FORTRAN tienne compte des blancs (Chapitre 1).

Il est possible de n'imprimer qu'une information Hollerith en ne donnant aucune liste avec l'expression d'entrée ou sortie et aucune indication de I, E ou F dans l'expression MODÈLE.

Considérons une zone Hollerith dans l'expression MODÈLE au moment de l'exécution du programme résultant. Si l'expression MODÈLE a été utilisée avec une expression d'entrée, le texte Hollerith dans l'expression MODÈLE sera remplacé par le texte lu à partir des zones correspondantes du milieu extérieur. Quand l'expression MODÈLE est utilisée avec une expression de sortie, le texte est normalement imprimé à partir de l'expression MODÈLE. Ainsi un texte peut être créé au moment de l'écriture du programme origine ou au moment de l'exécution en utilisant l'expression MODÈLE avec une expression d'entrée.

Modèles pour enregistrements multiples. Pour indiquer que l'on a à traiter un système d'entrées ou de sorties comportant plusieurs lignes d'enregistrements relatifs à un même ordre (lignes d'impression, cartes, enregistrements sur bandes décimales), une barre de fraction / devra séparer chaque ligne. Ainsi, MODÈLE (3F9.2, 2F10.4/8E14.5) définira un bloc dans lequel les lignes 1, 3, 5, . . . ont le MODÈLE (3F9.2, 2F10.4) et les lignes 2, 4, 6, . . . ont le MODÈLE 8E14.5.

Si l'on désire qu'un bloc d'informations ait les deux premières lignes d'un modèle spécial et toutes les autres d'un autre modèle, cette dernière ligne du MODÈLE sera à mettre entre parenthèses; par exemple: MODÈLE (12, 3E12.4/2F10.3, 3F9.4/(10F12.4)).

En général, s'il reste des quantités à transmettre lorsque la spécification du modèle a été complètement utilisée, le modèle est répété à partir de la dernière parenthèse ouverte y compris son coefficient et, s'il n'y en a pas, depuis le début. Exemple: MODÈLE (6E12.8/14, 2(F5.2, F6.4)). La troisième ligne se répètera suivant le modèle 2(F5.2, F6.4).

Des lignes blanches peuvent être introduites dans un bloc en omettant l'information du modèle; ainsi // et /// donneront respectivement une et deux lignes blanches.

Fin d'une expression MODÈLE. Si une expression d'entrée-sortie ne contient qu'un seul terme dans sa liste et si le MODÈLE correspondant ne comporte pas la spécification numérique de ce terme, l'expression d'entrée-sortie ne parviendra jamais à la fin. Une opération d'entrée-sortie décimale ne peut se terminer que lorsqu'une zone de spécification numérique est rencontrée dans l'expression MODÈLE et qu'il n'y a plus de quantités à transmettre.

Contrôle du saut de papier. Lorsque l'expression INSCR SORTIE BANDE doit être utilisée pour préparer une bande DCB qui sera imprimée par la suite sur l'équipement périphérique, il faut prévoir le saut de papier sur la tabulatrice. En effet, l'équipement périphérique peut être utilisé sous 3 formes: simple espacement, double espacement et espacement programmé, ce dernier offrant la plus grande souplesse.

En espacement programmé, le saut est contrôlé par le premier caractère de chaque enregistrement DCB et ce caractère n'est pas imprimé. Les différents caractères de contrôle avec leurs effets sont les suivants :

Blanc Simple espacement avant impression
 0 Double espacement avant impression
 + Pas d'espacement avant impression
 1-9 Saut aux canaux 1-9
 J-R Saut court aux canaux 1-9

Ainsi, la spécification MODÈLE de l'expression INSCR SORTIE BANDE pour imprimer en espacement programmé débute habituellement par un caractère 1H suivi du caractère de contrôle approprié.

L'impression sur tabulatrice connectée est identique pour l'expression IMPRIMER, FORTRAN simulant le contrôle d'impression périphérique.

Forme des données pour le programme résultant

Les données décimales à lire par LIRE ou LIRE, ENTRÉE BANDE, au moment de l'exécution du programme résultant, doivent être essentiellement du même modèle que celles données dans les exemples précédents. Ainsi, une carte lue avec MODÈLE (I2, E12.4, F10.4) doit être perforée :

27 -0.9321E 02 -0.0076

Dans chaque zone, toutes les informations doivent être cadrées à l'extrême droite. Les signes positifs peuvent être indiqués soit par un blanc, soit par +; le signe — doit être perforé 11 ou 8 et 4 (Voir Annexe A). Les blancs dans une zone numérique sont considérés comme des zéros. Les nombres pour les conversions E et F peuvent comporter un nombre quelconque de chiffres mais 8 chiffres seulement seront conservés. Un nombre pour la conversion I sera traité modulo 32768.

Pour permettre une économie dans la perforation, une certaine souplesse dans la présentation des données d'entrée est permise.

1. Les nombres pour la conversion E ne doivent pas comporter nécessairement 4 colonnes pour la zone exposant. Le début de la zone exposant doit être marqué par un E ou, s'il est omis, par + ou — (pas de blanc). Ainsi, E2, E02, +2, +02, E 02 et E+02, sont tous permis dans la zone exposant.
2. Pour les nombres à convertir du type E et F, il n'est pas nécessaire de perforer le point décimal. Si celui-ci n'est pas perforé, la spécification MODÈLE donne sa position effective; par exemple, — 09321+2 avec E12.4 sera traité comme si le point décimal avait été perforé 4 positions avant le début de la zone exposant, c'est-à-dire entre 0 et 9. Si le point décimal est perforé, sa position doit être située à gauche de la première décimale et il doit être compté dans la dimension w.

LIRE

FORME	EXEMPLES
« LIRE n, Liste » où n est le numéro d'une expression MODÈLE et la Liste telle que décrite précédemment.	LIRE 30, K, A(K)

L'expression LIRE permet de lire des cartes à partir du lecteur du 704. Chaque enregistrement (chaque carte) est lu jusqu'à ce que la liste complète soit entrée et convertie, mise en mémoire dans les adresses spécifiées par la liste. L'expression MODÈLE décrit la disposition des informations sur les cartes et le type de conversion à faire.

Si la liste n'est pas épuisée et qu'il n'y a plus de carte dans le lecteur, le programme résultant s'arrête. Après avoir mis d'autres cartes dans le magasin, en pressant sur le bouton DÉPART le programme continue à lire à partir du point où la liste avait été arrêtée.

Une vérification partielle est faite dans le programme résultant pour les colonnes incorrectement perforées ; une telle colonne provoque un arrêt. En appuyant sur le bouton DÉPART PROGRAMME, la colonne erronée est considérée comme nulle et le programme continue.

**LIRE ENTRÉE
BANDE**

FORME	EXEMPLES
« LIRE ENTREE BANDE i, n , Liste » où i est une constante ou une variable positive en fixe comprise entre 1 et 10, n est le numéro de l'expression MODÈLE et la Liste telle que décrite précédemment.	LIRE ENTRÉE BANDE 3, 30, K, A(K) LIRE ENTRÉE BANDE I, 30, K, A(K)

L'expression LIRE ENTRÉE BANDE permet de lire, au cours du programme résultant, des informations DCB à partir de la bande i , i pouvant être égal à 1, 2, ..., 10. Les enregistrements sont entrés l'un après l'autre dans la machine suivant le MODÈLE jusqu'à épuisement de la liste des quantités à mettre en mémoire.

Une fin de bloc provoque un arrêt dans le programme résultant. En appuyant sur le bouton DÉPART PROGRAMME, le programme continue à lire à partir du point où il avait été arrêté dans la liste.

Le programme résultant vérifie la lecture de la bande. Si un enregistrement est deux fois erroné, le programme s'arrête. En appuyant sur le bouton DÉPART l'information lue la seconde fois est acceptée et le programme continue.

PERFORER

FORME	EXEMPLES
« PERFORER n , Liste » où n est le numéro de l'expression MODÈLE et la Liste telle que décrite précédemment.	PERFORER 30, (A(J), J = 1,10)

L'expression PERFORER permet au programme résultant de perforer des cartes sur la perforatrice du 704. Les cartes sont perforées suivant le MODÈLE jusqu'à ce que toute la liste soit perforée.

Aucune vérification n'est faite, il n'y a donc pas d'ARRÊT dans le programme résultant.

IMPRIMER

FORME	EXEMPLES
« IMPRIMER n, Liste » où n est le numéro de l'expression MODÈLE et la Liste telle que décrite précédemment.	IMPRIMER 30, (A(J), J = 1,10)

L'expression IMPRIMER a pour effet, dans le programme résultant, d'imprimer sur la tabulatrice du 704. Les lignes sont imprimées suivant le MODÈLE jusqu'à épuisement de la liste à imprimer.

L'impression est vérifiée par ÉCHO et une erreur d'impression arrête la machine. En appuyant sur le bouton DÉPART PROGRAMME, le programme continue. En appuyant sur les boutons RAZ CONTR REG et DÉPART, la ligne est imprimée une nouvelle fois et le programme continue.

INSCR SORTIE BANDE

FORME	EXEMPLES
« INSCR SORTIE BANDE i, n, Liste » où i est une constante ou une variable positive en fixe comprise entre 1 et 10, n est le numéro de l'expression MODÈLE et la Liste telle que décrite précédemment.	INSCR SORTIE BANDE 3, 30, (A(J), J = 1, 10) INSCR SORTIE BANDE 1, 30, (A(J), J = 1, 10)

L'expression INSCR SORTIE BANDE permet d'écrire des informations DCB sur la bande i, i pouvant être égal à 1, 2, ..., 10. Les enregistrements seront faits suivant le MODÈLE et jusqu'à épuisement de la liste. Aucune fin de bloc n'est écrite après le dernier enregistrement.

Aucune vérification n'est faite, il n'y a donc pas d'ARRÊT dans le programme résultant.

LIRE BANDE

FORME	EXEMPLES
« LIRE BANDE i, Liste » où i est une constante ou une variable positive en fixe comprise entre 1 et 10 et la Liste telle que décrite au chapitre 7.	LIRE BANDE 3, (A(J), J = 1, 10) LIRE BANDE 1, (A(J), J = 1, 10)

LIRE BANDE a pour effet, dans le programme résultant, de lire des informations binaires à partir de la bande i, i pouvant être égal à 1, 2, ..., 10. Un seul enregistrement est lu et il l'est entièrement si la liste contient autant de mots qu'en contient l'enregistrement. La bande va jusqu'à l'enregistrement suivant.

Si la liste est plus longue que l'enregistrement, le programme résultant s'arrêtera avec une vérification lecture-écriture et le programme ne pourra être continué.

Une fin de bloc provoque un arrêt dans le programme résultant. Si l'on appuie sur le bouton DÉPART, le programme lira l'enregistrement suivant.

LIRE BANDE peut, cependant, être donné sans liste, de façon à sauter un enregistrement ou une fin de bloc.

En raison des temps requis, on doit limiter la complexité des listes. (Voir au chapitre 7 la section « Liste pour bandes binaires »).

Le programme résultant teste les vérifications longitudinales de la bande à lire (la vérification longitudinale est employée si l'enregistrement complet est lu). Si un enregistrement est erroné deux fois de suite, le programme s'arrête. En appuyant sur le bouton DÉPART, l'information lue la seconde fois est acceptée comme correcte et le programme continue.

LIRE TAMBOUR

FORME	EXEMPLES
« LIRE TAMBOUR i, j, Liste » où i et j sont des constantes ou des variables positives en fixe, i étant compris entre 1 et 8 et la Liste telle que décrite ci-dessous.	LIRE TAMBOUR 2, 1000, A, B, C, D LIRE TAMBOUR I, J, A, B, C, D

LIRE TAMBOUR permet au programme résultant de lire des mots binaires à partir d'addresses consécutives sur le tambour i où $i = 1, 2, \dots, 8$, en commençant par le mot portant l'adresse j, où $j = 0, 1, \dots, 2047$ (si j est supérieur ou égal à 2048, il est interprété modulo 2048). La lecture s'effectue jusqu'à ce que la liste complète soit lue.

En raison des temps requis, il y a des limitations très précises à la complexité de la liste. En fait, la liste doit être employée avec les notations abrégées décrites précédemment dans ce chapitre; elle ne doit comprendre que des variables non indicées comme A, B, C, D. Les variables simples seront lues dans les mémoires de la manière ordinaire. Les variables qui ont des dimensions sont lues avec l'indexage donné par l'expression DIMENSION. (Voir chapitre 6).

Aucune vérification n'est faite et il n'y a pas d'ARRÊT dans le programme résultant.

INSCR BANDE

FORME	EXEMPLES
« INSCR BANDE i, Liste » où i est une constante ou une variable positive en fixe comprise entre 1 et 10 et la Liste telle que décrite au chapitre 7.	INSCR BANDE 3, (A(J), J = 1, 10) INSCR BANDE I, (A(J), J = 1, 10)

INSCR BANDE permet au programme résultant d'écrire des informations binaires sur la bande i, où $i = 1, 2, \dots, 10$. Un seul enregistrement est écrit et sa longueur est déterminée par la liste.

En raison des temps requis, il y a des limitations à la complexité d'une liste. (Voir chapitre 7).

Aucune vérification n'est faite et il n'y a pas d'ARRÊT dans le programme résultant.

INSCR TAMBOUR

FORME	EXEMPLES
« INSCR TAMBOUR i, j, Liste » où i et j sont des constantes ou des variables positives en fixe, i étant compris entre 1 et 8 et la Liste telle que décrite dans LIRE TAMBOUR	INSCR TAMBOUR 2, 1000, A, B, C, D INSCR TAMBOUR I, J, A, B, C, D

INSCR TAMBOUR permet au programme résultant d'écrire des mots binaires dans les mémoires consécutives du tambour i, où $i = 1, 2, \dots, 8$, en commençant à partir de l'adresse j, où $j = 0, \dots, 2047$ (s'il était égal ou supérieur à 2048, il serait interprété modulo 2048). L'écriture est continue jusqu'au moment où la liste est épuisée.

La liste est sujette aux mêmes restrictions que LIRE TAMBOUR.

Aucune vérification n'est faite et il n'y a pas d'ARRÊT dans le programme résultant.

FIN BLOC

FORME	EXEMPLES
« FIN BLOC i » où i est une constante ou une variable positive en fixe comprise entre 1 et 10.	FIN BLOC 3 FIN BLOC I

FIN BLOC permet au programme résultant d'écrire une fin de bloc sur la bande i, où $i = 1, 2, \dots, 10$.

REBOBINER

FORME	EXEMPLES
« REBOBINER i » où i est une constante ou une variable positive en fixe comprise entre 1 et 10.	REBOBINER 3 REBOBINER I

REBOBINER permet au programme résultant de rebobiner la bande i, où $i = 1, 2, \dots, 10$.

ESP ARR

FORME	EXEMPLES
« ESP ARR i » où i est une constante ou une variable positive en fixe comprise entre 1 et 10.	ESP ARR 3 ESP ARR I

ESP ARR permet au programme résultant d'effectuer un espacement arrière d'un enregistrement sur la bande i, où $i = 1, 2, \dots, 10$.

Identification des arrêts

Les différents arrêts qui peuvent survenir durant les Opérations d'entrée et de sortie dans le programme résultant peuvent être identifiés par le contenu du registre mémoire sur le pupitre du 704. Le Manuel Opérateur FORTRAN contient une liste de ces arrêts et leurs identifications.

CHAPITRE 6. LE LANGAGE FORTRAN : *Expressions d'organisation*

Le dernier groupe des expressions FORTRAN comprend trois expressions d'organisation : DIMENSION, ÉQUIVALENCE et FRÉQUENCE. Ces expressions ne sont pas exécutables, mais elles fournissent des informations que FORTRAN utilise pour donner un programme résultant plus efficient.

DIMENSION

FORME	EXEMPLES
« DIMENSION v, v, v, ... » où chacun des v est une variable indiquée avec 1, 2 ou 3 constantes positives en fixe. Le nombre de v peut être quelconque.	DIMENSION A(10), B(5,15), C(3, 4, 5)

L'expression DIMENSION permet de réserver les mémoires nécessaires dans le programme résultant aux quantités à plusieurs dimensions.

Chaque variable qui apparaît dans le programme sous forme indicée doit apparaître dans une expression DIMENSION et cette expression DIMENSION doit précéder la première apparition de la variable. Dans l'expression DIMENSION sont données les dimensions désirées pour l'ensemble des valeurs de la variable. Dans le programme exécuté, les indices de cette variable ne doivent jamais être supérieurs à ceux définis dans DIMENSION.

Ainsi, l'exemple stipule que B a 2 dimensions et que les indices de B ne doivent jamais être supérieurs à 5 et à 15 ; 75 mémoires sont réservées pour l'ensemble B.

Une seule expression DIMENSION peut être utilisée pour plusieurs ensembles.

ÉQUIVALENCE

FORME	EXEMPLES
« ÉQUIVALENCE (a, b, c, ...), (d, e, f, ...) » où a, b, c, d, e, f, ... sont des variables parfois suivies d'une constante positive en fixe entre parenthèses.	ÉQUIVALENCE (A, B(1), C(5)), (D(17), E(3))

L'expression ÉQUIVALENCE permet au programmeur, s'il le désire, de commander les attributions de mémoires dans le programme résultant. En particulier, elle lui permet ainsi d'économiser des mémoires pour les données en prenant les mêmes mémoires pour deux ou plusieurs quantités quand la logique de son programme le rend possible. Elle lui permet aussi, s'il le désire, d'appeler une même quantité de différents noms et de faire en sorte que ces noms soient considérés comme équivalents.

Une expression ÉQUIVALENCE peut être placée n'importe où dans le programme origine. Chaque paire de parenthèses contient les noms de deux ou plusieurs quantités dont les mémoires doivent être identiques dans le programme résultant. Le nombre d'équivalences (paires de parenthèses) peut être quelconque.

Dans une expression ÉQUIVALENCE la signification de C (5), par exemple, est la quatrième mémoire dans le programme résultant après la mémoire contenant C ou si C est un ensemble ordonné après C (1) ou C (1,1) ou C (1,1,1). En général, A (p) est défini pour $p \geq 1$ et représente la p-1^e mémoire après A ou le début du tableau A, c'est-à-dire la p^e mémoire du tableau. Si p n'est pas donné, il est pris égal à 1.

Ainsi, dans l'expression donnée comme exemple, les A, B, C ou le début des tableaux A, B ou C sont placés dans des mémoires de telle sorte que la mémoire contenant A, la mémoire contenant B et la quatrième mémoire après celle contenant C soient les mêmes. De même la 16^e mémoire après D et la 2^e après E doivent être identiques.

Une quantité (ou un ensemble de quantités) qui n'apparaît pas dans une expression ÉQUIVALENCE sera dans des mémoires qui lui seront propres.

Des mémoires ne peuvent être réparties qu'entre des variables et non entre des constantes.

L'attribution de mêmes mémoires à plusieurs variables ne peut être faite correctement sans une connaissance de celle des expressions FORTRAN qui, dans le programme résultant, envoient une nouvelle valeur dans une mémoire. Il y a 7 expressions de ce type :

1. L'exécution d'une formule arithmétique met en mémoire une nouvelle valeur pour la variable de gauche.
2. L'exécution d'un IMPOSER i EN n envoie une nouvelle valeur à la place de n.
3. L'exécution d'un FAIRE donne, en général, une nouvelle valeur à l'index. (Il n'en est pas toujours ainsi, comme nous le verrons dans la section « Détails à propos de Faire », au chapitre 7).
4. L'exécution d'un LIRE, LIRE ENTRÉE BANDE, LIRE BANDE ou LIRE TAMBOUR met en mémoire de nouvelles valeurs des variables de la liste.

FRÉQUENCE

FORME	EXEMPLES
« FRÉQUENCE n (i, j, ...), m (k, l, ...), ... » où n, m, ... sont des numéros d'expression et i, j, k, l, ... sont des constantes positives en fixe.	FRÉQUENCE 30(1, 2, 1), 40(11), 50(1, 7, 1, 1)

L'expression FRÉQUENCE permet au programmeur de donner son estimation, pour chaque embranchement contrôlé, des fréquences avec lesquelles les branches seront exécutées dans le programme résultant. Cette information est utilisée pour optimiser l'usage des registres Index dans le programme résultant.

Une expression FRÉQUENCE peut être placée n'importe où dans le programme origine et peut être utilisée pour donner une information de fréquence pour un nombre quelconque d'embranchements. Pour chaque embranchement, l'information est constituée par le numéro de l'expression créant l'aiguillage suivi, entre parenthèses, des fréquences prévues, séparées par des virgules.

Dans l'exemple ci-dessus, il peut s'agir d'une expression FRÉQUENCE dans un programme où l'expression 30 est un SI, 40 est un FAIRE et 50 est un ALLER A calculé. Le programmeur estime que l'argument SI a autant de chances d'être 0 ou non 0 et que s'il est non 0 il a autant de chances d'être positif que négatif. L'expression FAIRE de la ligne 40 est présumée avoir au moins un de ses paramètres d'indexage qui ne soit pas une constante mais une variable; donc le nombre de fois où la boucle doit être exécutée n'est pas connu à l'avance. Le programmeur pense que 11 est une bonne estimation pour ce nombre. Pour l'expression ALLER A en 50 les fréquences des sorties sont estimées pour les quatre branches : 1, 7, 1, 1.

Toutes les fréquences estimées sauf celles de FAIRE sont *relatives*; ainsi, elles peuvent être multipliées par une constante. L'exemple précédent peut être également exprimé sous la forme FRÉQUENCE 30 (2,4,2), 40 (11), 50 (3,21,3,3). Une fréquence peut être évaluée 0 quand le cas a pratiquement peu de chance d'être réalisé.

La table suivante donne la liste des 7 types d'expression pour lesquels une information de fréquences peut être donnée.

TYPE	Nombre de Fréquences	REMARQUES
ALLER A calculé	≥ 2	Ordre des Fréquences : le même que l'ordre des branches
SI	3	— — —
SI (CLE)	2	— — —
SI DC AC	2	— — —
SI DC MQ	2	— — —
SI DIV IMP	2	— — —
FAIRE	1	à ne donner que si m_1 , m_2 ou m_3 est variable.

Il n'est pas nécessaire de donner une information de fréquences pour chaque expression de transfert. Si rien n'est donné, les différentes branches sont considérées comme ayant une probabilité égale.

Une estimation de fréquences à propos d'une expression FAIRE sera ignorée sauf quand au moins un des paramètres d'indexage de ce FAIRE est variable. De plus, la fréquence estimée doit être basée uniquement sur les valeurs attendues de ces paramètres; en d'autres termes, même si la boucle du FAIRE contient un SI ou un ALLER A qui peut transférer à l'extérieur de la boucle, l'estimation de fréquence sera le nombre de fois que cette boucle devra être exécutée pour avoir une sortie normale.

Un FAIRE pour lequel les paramètres d'indexage sont variables et pour lequel aucune expression FRÉQUENCE n'est donnée est traité comme si la fréquence avait été estimée 5.

CHAPITRE 7. DÉTAILS COMPLÉMENTAIRES SUR FORTRAN

Machine origine
Machine résultant Les appellations de machine *origine* et machine *résultant* ont été adoptées pour désigner les machines traduisant le programme origine en programme résultant, d'une part, et utilisant le programme résultant pour l'exécution du calcul, d'autre part.

La machine origine peut être un 704 ne comportant que 4096 mots de mémoire à ferrites, l'instruction CAL (copie et addition logique), une unité de tambour, 4 unités de bandes, une perforatrice connectée, un lecteur de cartes et une tabulatrice connectés ou périphériques.

Les spécifications de la machine résultant sont déterminées par le programme origine.

Organisation
du programme
résultant Les instructions et les constantes du programme résultant sont mises en mémoires à partir de 0. Les données et les autres adresses nécessaires le sont à partir de l'adresse 77777, dans l'ordre décroissant. Ces adresses sont donc toujours les dernières quelle que soit la capacité de la machine résultant.

Les adresses les plus élevées sont celles des variables qui apparaissent dans les expressions DIMENSION ou ÉQUIVALENCE. Les adresses de ces variables font que deux programmes, dont les expressions DIMENSION et ÉQUIVALENCE sont identiques, auront des adresses identiques. Il est donc possible d'écrire une famille de programmes avec les mêmes données.

Les mémoires qui viennent ensuite sont occupées par les variables qui ne sont pas définies par DIMENSION ou ÉQUIVALENCE, des réservations nécessaires au déroulement du programme et enfin une section de mémoires effaçables.

Pour chaque programme résultant, FORTRAN donne la répartition des mémoires.

Arithmétique
fixe L'utilisation de l'arithmétique fixe est soumise aux considérations suivantes :

1. Les constantes fixes définies dans le programme origine ne peuvent être supérieures à 2^{15} .
2. Les données fixes lues par le programme résultant sont traitées modulo 2^{15} .
3. L'arithmétique fixe du programme résultant est une arithmétique modulo 2^{15} .
4. L'indexage dans le programme résultant dépend de la capacité de la machine résultant.

**Ecriture des
sous-programmes
pour la bande
maîtresse**

Les sous-programmes de la bibliothèque existent sur la bande maîtresse sous forme binaire translatable. Pour placer un nouveau sous-programme sur la bande, il est nécessaire d'établir d'abord le programme sous forme de cartes binaires translatables, puis de le transcrire sur la bande au moyen d'un programme spécial.

Dans le programme résultant, le transfert au sous-programme est obtenu par la séquence :

TSX Sous-programme, 4
Retour normal

Le sous-programme lui-même et les constantes nécessaires sont écrits dans les mémoires translatables 0, 1, 2, ... Il peut aussi utiliser un *bloc de mémoires communes* de longueur désirée n , commençant à l'adresse translatable $77777_8 - (n-1)$ et finissant à 77777_8 .

Au moment du transfert au sous-programme, ARG 1 sera placé dans l'AC, ARG 2 (s'il existe) dans le MQ, ARG 3 (s'il existe) dans la mémoire translatable 77775_8 du bloc de mémoires communes, ARG 4 (s'il existe) à l'adresse 77774_8 , etc... Les mémoires de ce bloc peuvent aussi être utilisées comme mémoires de travail par le sous-programme.

Après l'exécution du sous-programme, le résultat est dans l'accumulateur et les Registres d'Index 1 et 2 doivent être restaurés à leur valeur initiale.

Les quantités fixes dans le programme résultant se présentent sous la forme suivante : le signe dans la position signe, la valeur absolue dans la partie décrement, le mot étant complété par des zéros.

Il est souhaitable que les arrêts pour erreur dans les sous-programmes soient codés avec HSE et un nombre permettant d'identifier l'erreur et le sous-programme au pupitre.

Exponentiation

L'expression d'exponentiation programmée sous la forme $E^{**}F$ donnera lieu à une des 5 solutions résultantes suivantes :

1 et 2. Si F est une constante fixe comprise entre 1 et 7 inclus, un des deux sous-programmes ouverts sera utilisé pour le calcul, suivant que E est fixe ou flottant. La méthode employée se ramène à $F-1$ multiplications successives avec un maximum de $F + 2$ instructions.

Le calcul est valable pour E quelconque.

3 et 4. Si F est une constante fixe en dehors des limites 1 à 7 ou une variable fixe, un des deux sous-programmes fermés sera utilisé pour le calcul, suivant que E est fixe ou flottant. Ces sous-programmes n'apparaîtront pas plus d'une fois dans le programme résultant. Ils exécutent l'exponentiation en formant E^F sous forme du produit de E, E^2, E^4, E^8, \dots , suivi du calcul de l'inverse si $F < 0$.

Le sous-programme pour E fixe comprend 34 instructions et demande environ 10 ms. Pour $F \leq 0, E = 0$ et pour $F < 0, |E| > 1$, le sous-programme prend immédiatement zéro; pour tous les autres cas, il calcule E^F .

Le sous-programme pour E flottant comprend 38 instructions et demande environ 9 ms. Pour $F \leq 0$, $E = 0$, il prend zéro; dans tous les autres cas, il calcule E^F .

5. Si F et — en conséquence — E sont flottants, un sous-programme fermé est utilisé. Il n'apparaîtra pas plus d'une fois dans le programme résultant. Il calcule E^F comme étant $e^{F \log_e E}$. Il comprend 100 instructions et demande environ 6 ms. Pour $F \leq 0$, $E = 0$, il prend zéro; dans tous les autres cas, il calcule $|E|^F$.

Optimisation des expressions arithmétiques

La plus grande attention a été apportée dans FORTRAN pour s'assurer que le programme résultant d'une expression arithmétique sera efficient quelle que soit la manière dont cette dernière a été écrite. Par exemple, une séquence de multiplications et de divisions non groupées par des parenthèses sera automatiquement ordonnée, si nécessaire, pour diminuer le nombre de transferts en mémoire dans le programme résultant.

Cependant, une partie importante de l'optimisation qui concerne les *expressions communes* n'est prise en compte que si l'expression est écrite correctement. Comme exemple, prenons la formule arithmétique :

$$Y = A*B*C + \text{SINF}(A*B)$$

Un programme résultant correct ne calculera le produit $A*B$ qu'une seule fois. Cependant, si la formule arithmétique est écrite comme ci-dessus, la multiplication de A par B sera effectuée deux fois. La forme correcte pour écrire cette formule arithmétique est :

$$Y = (A*B)*C + \text{SINF}(A*B)$$

L'expression commune $A*B$ a été mise entre parenthèses et le programme résultant n'effectue qu'une seule fois la multiplication de A par B.

En général, des parenthèses seront utilisées pour déceler les expressions communes à l'intérieur des expressions à calculer.

Il existe un cas où le programmeur n'a pas besoin d'écrire ces parenthèses. Ainsi, les parenthèses qui sont décrites dans la section « Hiérarchie des Opérations » au chapitre 3 n'ont pas besoin d'être écrites :

$$Y = A*B + C + \text{SINF}(A*B)$$

est aussi correct, au point de vue de l'optimisation, que :

$$Y = (A*B) + C + \text{SINF}(A*B)$$

Détails complémentaires à propos de l'expression FAIRE

Cette section contient quelques détails sur l'emploi de l'expression FAIRE.

Indexage triangulaire. Un indexage tel que :

FAIRE I = 1,10
FAIRE J = 1,10

ou :

FAIRE I = 1,10
FAIRE J = 1,1

est permis et simplifie le calcul pour les tableaux triangulaires. Ce sont des cas particuliers du fait que l'index contrôlé par FAIRE est disponible pour l'utilisation générale comme une variable fixe.

Les éléments de la diagonale d'un bloc peuvent être appelés par un indexage du type suivant :

FAIRE I = 1, 10
A(I, I, I) = une expression.

Règles pour le contenu de la mémoire Index. Une boucle FAIRE avec Index I ne doit pas affecter le contenu de la mémoire du programme résultant contenant I sauf dans certaines conditions, en particulier si :

1. Un transfert du type SI ou ALLER A provoque la sortie de la boucle FAIRE.
2. I est utilisé comme variable dans une boucle FAIRE.
3. I est utilisé comme indice en combinaison avec une *constante relative* dont la valeur change à l'intérieur du FAIRE (une constante relative est un indice dont la variable fixe n'est pas normalement sous le contrôle de FAIRE).

C'est pourquoi, si une sortie normale d'un FAIRE auquel les cas 2 et 3 ne sont pas applicables se produit, la mémoire I contient ce qu'il y avait avant sa modification. Après une sortie normale pour laquelle les cas 2 et 3 peuvent être appliqués, la mémoire I contient la première valeur de la séquence I qui est supérieur à m_2 . Après un transfert, la mémoire contient la valeur actuelle de I.

Listes pour les opérations de bandes binaires

La mise entre parenthèses de certaines parties des listes dans le cas des expressions LIRE BANDE et INSCR BANDE est soumise à certaines restrictions qui doivent être observées pour que le temps d'indexage du programme résultant n'excède pas 288 μ s, temps maximum laissé au calcul entre 2 instructions COP en 704.

Du fait de la complexité de ce sujet il n'en sera donné qu'un exposé incomplet. Certaines règles qui seront données permettront la construction des listes utiles au travail. Il y aura pourtant d'autres listes exclues de ces règles qui seront utilisables.

Comment définir le terme *combinaison d'indices*? Considérons chaque indice de chaque variable indicée défini par les quatre symboles c, v, c' et d, où d est la valeur maximum de l'indice donné par l'expression DIMENSION correspondant à cette variable et où $c*v \pm c'$ est la forme entière de l'indice (la forme entière de l'indice I serait $1*I + 0$; celle de l'indice 3 serait $1*0 + 3$, etc.). La combinaison indicée d'une variable indicée peut être :

c_1, v_1
 c_1, v_1, c_2, v_2, d_1
 $c_1, v_1, c_2, v_2, c_3, v_3, d_1, d_2$

suivant que la variable a 1, 2 ou 3 indices. A(I,J,K), B(I,K,J), C(5*I,J,K) et D(I,J) ont tous des combinaisons indicées différentes, mais A(I,J,K) et B(I + 5,J,K) ont la même combinaison d'indices si les deux premières dimensions de A et B sont les mêmes.

D'autre part, on appelle *élément* d'une liste la partie limitée par les virgules principales.

Éléments qui ne contiennent pas de variables indicées. Il n'y a aucune restriction à ce sujet.

Éléments ne contenant pas de variables à 3 indices. Soit :

N_1 = Nombre de combinaisons indicées différentes à 1 dimension
 N_2 = Nombre de combinaisons indicées différentes à 2 dimensions
 $M_1 = 0$ si $N_1 = 0$; autrement $M_1 = 1$
 $M_2 = 0$ si $N_2 = 0$; autrement $M_2 = 5$

La condition est alors : $(4N_1 + 6N_2 + \text{Max}(M_1, M_2)) \leq 18$.

Par exemple, pour l'élément $((A(I,J), B(I,J), J = 1,10), C(I), I = 1,10)$, $N_1 = 1$, $N_2 = 1$ (à condition que les premières dimensions de A et B soient les mêmes); $4N_1 + 6N_2 + \text{Max}(M_1, M_2) = 15$, et l'élément est possible.

Éléments contenant des variables à 3 indices. Considérons une variable A dont les 3 indices s_1, s_2, s_3 font intervenir comme variables v_1, v_2, v_3 . Les éléments schématisés peuvent être :

$(((, v_2 =), v_3 =), , v_1 =)$

ou

$(((, v_2 =), v_1 =), , v_3 =)$

ou

$(((, v_2 =), v_3 =), , v_1 =)$

Les parenthèses les plus intérieures (contrôlant respectivement v_2, v_3, v_1) peuvent contenir $A(s_1, s_2, s_3)$ et un certain nombre d'autres variables avec la même combinaison indicée; de même, les parenthèses extérieures peuvent contenir des variables d'une quelconque combinaison indicée. Les parenthèses médianes ne peuvent contenir aucune variable indicée. Ainsi, par exemple :

$((A(I,J,K), B(I,J,K), K = 1,10), J = 1,5), C(I), D(I), I = 1,10)$

fonctionnera à condition que les deux premières dimensions de A et B soient les mêmes. Cet exemple fait usage du premier des éléments schématisés ci-dessus.

Dans le cas particulier où le prochain élément de la liste n'est ni indicé ni inclus dans des parenthèses, le premier des éléments schématisés peut aussi contenir des variables d'une quelconque combinaison d'indices entre les parenthèses médianes.

Enfin, l'élément schématisé :

$(((, v_1 = 1,d_1), v_2 = 1,d_2), v_3 =)$

contrairement aux autres indexages dans l'ordre habituel, fonctionnera. Remarquons, cependant, que les paramètres indicés pour v_1 et v_2 doivent varier

jusqu'à leurs limites. Les parenthèses les plus intérieures peuvent contenir $A(s_1, s_2, s_3)$ et un nombre quelconque d'autres variables avec la même combinaison indiquée.

Paramètres d'indexage variables et constantes relatives. Une autre restriction concernant les listes pour bandes binaires a trait à l'utilisation de constantes relatives et de paramètres d'indexage qui sont variables (une constante relative est un indice dont la variable n'est pas normalement sous le contrôle d'un FAIRE ou d'une parenthèse).

La restriction réside dans le fait que les variables qui ont un indice comprenant soit une constante relative, soit un index déterminé par des paramètres d'indexage variables, ne peuvent apparaître que dans le premier élément d'une liste. Par là même, toutes les variables de ce type doivent avoir également l'un de leurs indices qui soit contrôlé par la première parenthèse.

Par exemple, dans $((A(I,J), I = 1,L), B(K,J), J = 1,M)$ les deux indices de A sont déterminés par les paramètres d'indexage variables; il en est de même pour un des indices de B, tandis que l'autre est une constante relative (à moins que le LIRE BANDE ou INSCR BANDE soit lui-même dans la boucle du FAIRE portant sur K). Cependant, puisque J, qui est l'index contrôlé par la première parenthèse, est un indice à la fois de A et de B, cet élément fonctionnera s'il est le premier élément de la liste.

**Dimension limite
du Programme
origine**

Au moment de la traduction du programme origine en programme résultant, FORTRAN crée et utilise des tables qui représentent les différents aspects des informations contenues dans le programme origine. Les dimensions de ces tables sont fonction de la quantité d'informations contenue dans le programme origine. En cas de dépassement des limites de ces tables, le programme FORTRAN s'arrêtera à l'une des adresses données dans le manuel opérateur FORTRAN.

Dans ce qui suit l'expression « écriture » signifie que si un même élément est écrit plusieurs fois, il est compté plusieurs fois.

1. (Table TEIFNO). Le nombre d'expressions FORTRAN qui sont numérotées ne doit pas dépasser 1500. (Une expression d'entrée-sortie qui a un numéro d'expression et dont la liste contient des parenthèses de contrôle compte double).
2. (Table FIXCON). Le nombre de constantes fixes différentes ne doit pas dépasser 100 (Les constantes qui ne diffèrent que par le signe ne sont pas considérées comme différentes).
3. (Table FLOCON). Le nombre de constantes flottantes différentes ne doit pas dépasser 450 (Les constantes qui ne diffèrent que par le signe ne sont pas considérées comme différentes, ainsi que les nombres 4., 4.0, 40.E-1, etc..., qui sont réellement identiques).
4. (Table TDO). Le nombre d'expressions FAIRE ne doit pas dépasser 150.
5. Le nombre d'expressions FAIRE à l'intérieur d'une boucle ne doit pas dépasser 50.
6. (Table TIFGO). Le nombre total des expressions IMPOSER, SI et ALLER A ne doit pas dépasser 300.
7. (Table TRAD). Le nombre total de numéros d'expressions mentionnés dans les expressions ALLER A calculés et prédéterminés ne doit pas dépasser 250.

- 8.** (Table FRET). Le nombre total de numéros mentionnés dans les expressions FRÉQUENCE ne doit pas dépasser 750. (Une expression telle que FRÉQUENCE 30 (1,2,1) a 4 numéros).
- 9.** (Tables DIM). Le nombre de variables à 1, 2 ou 3 dimensions qui apparaissent dans les expressions DIMENSION ne doit pas dépasser respectivement 100, 100 et 90.
- 10.** (Table EQUIT). Le nombre d'« écritures » des variables dans les expressions ÉQUIVALENCE ne doit pas dépasser 750.
- 11.** (Table LAMBDA). Cette table et la table BETA qui suit limitent la dimension des expressions arithmétiques dans la partie droite des formules ainsi que les arguments des expressions SI. Soit, dans une expression :
- n = nombre d'« écritures » des variables et constantes, sauf celles des indices;
 - b = nombre de parenthèses ouvertes, sauf celles des indices;
 - p = nombre d'« écritures » de + ou —, sauf dans les indices ou comme signe de symbole (le + dans $A^*(+B)$ est le signe du symbole B);
 - t = nombre de * ou /, sauf dans les indices;
 - e = nombre de **;
 - f = nombre d'« écritures » des noms de fonction; et
 - a = nombre d'arguments de fonctions (Pour SINF (SINF(X)), a = 2).
- $LAMBDA = n + 4b + 4a - 3f + 3p + 2t + e + 3$, ne doit pas dépasser 400.
- 12.** (Table BETA). $BETA = LAMBDA + 1 - n - f$, ne doit pas dépasser 300.
- 13.** (Table CLOSUB). Dans tout le programme, le nombre d'« écritures » des fonctions ne doit pas dépasser 1500.
- 14.** (Table FORVAL). Le nombre de formules arithmétiques dont les parties gauches sont des variables fixes non indicées ne doit pas dépasser 500.
- 15.** (Table FORVAR). Le nombre total d'« écritures » de variables fixes non indicées dans la partie droite des formules arithmétiques et dans les arguments des SI ne doit pas dépasser 750.
- 16.** (Table FORTAG). Le nombre total d'« écritures » de variables indicées ne doit pas dépasser 1500.
- 17.** (Tables TAU). Le nombre total de combinaisons d'indices différentes à 1, 2 ou 3 dimensions ne doit pas dépasser respectivement 100, 90 et 75. (Voir la section précédente pour la définition de combinaison d'indices).
- 18.** (Tables SIGMA). Soit une variable à 3 indices et c'_1, c'_2, c'_3 les parties additives de ces indices (écrits sous forme complète). Le nombre de systèmes distincts c'_1, c'_2, c'_3 ne doit pas dépasser 100. De même, pour des variables à 2 et 1 dimensions, ni le nombre de c'_1, c'_2 ni celui de c'_1 ne doit dépasser 100.
- 19.** Il existe quelques limitations supplémentaires qui sont trop compliquées pour être exposées ici. Dans tous les cas, cependant, un arrêt identifiable se produira dans le programme FORTRAN.

CHAPITRE 8. EXEMPLES DE PROGRAMMATION FORTRAN

Exemple d'un Programme complet

Cet exemple, qui reprend le problème traité dans le chapitre 1, constitue un programme FORTRAN simple mais complet. Le but du programme est de trouver dans l'ensemble des nombres $A(I)$ la plus grande valeur prise par l'un des éléments et de l'imprimer. Les valeurs des $A(I)$ sont entrées par cartes perforées contenant 12 nombres de 6 colonnes. Il ne peut y avoir plus de 999 valeurs; leur nombre exact N est perforé dans une carte contrôle précédant les cartes A.

C ← POUR COMMENT.	SUIVE						IDENTIFICATION			
	N°	1	5	6	7		72	73	80	
		PROGRAMME POUR TROUVER LA PLUS GRANDE VALEUR D UNE SERIE DE								
	1	NOMBRES								
		DIMENSION A(999)								
		FREQUENCE 30(2,1,10),5(100)								
		LIRE 1,N,(A(I),I=1,N)								A
	1	MODELE (I3/(12F6.2))								B
		AMAX=A(1)								C
	5	FAIRE 20 I=2,N								D
	30	SI (AMAX-A(I))10,20,20								E
	10	AMAX=A(I)								
	20	CONTINUER								F
		IMPRIMER 2,N,AMAX								G
	2	MODELE (19H1LE PLUS GRAND DES I3,13H NOMBRES EST F7.2)								H
		ARRET 7777								

La première expression exécutable est l'expression (A); cet ordre de lecture permet d'entrer dans la machine, successivement, N , $A(1)$, $A(2)$, ..., $A(N)$ suivant les indications fournies par le MODÈLE 1. L'indexage de la boucle contrôlant la lecture des $A(I)$ est parfaitement défini, la valeur de N étant lue au préalable.

Le MODÈLE défini dans l'expression (B) indique que la première carte à lire correspond à I3 et celles qui suivent à 12F6.2. Donc N perforé dans les

trois premières colonnes de la première carte est converti en un nombre entier. Les A(I), perforés dans 12 zones : 1 à 6, 7 à 12,, 67 à 72 sous la forme XXXXX ou — XXXXX, sont transformés en nombres flottants, la virgule étant sous-entendue avant les deux derniers chiffres. Cependant, si la virgule décimale figurait dans la carte, sa position l'emporterait sur celle indiquée dans le MODÈLE.

Après la lecture, l'expression (C) est exécutée, la mémoire AMAX est faite égale à A(1).

L'expression (D) définit une boucle dont l'index est I et les paramètres 2 et N.

Dans l'expression (E), SI est exécutée une première fois pour $I = 2$ et renvoie aux expressions 10 ou 20, suivant que $A(2) > AMAX$ ou que $A(2) \leq AMAX$. Dans le premier cas, AMAX devient égal à A(2); dans les autres cas, il reste égal à A(1).

L'expression (F) CONTINUER permet de terminer la boucle et provoque un transfert en début de boucle jusqu'à ce que I ait pris toutes les valeurs de 2 à N inclus. La comparaison va donc se faire successivement entre AMAX et A(3), AMAX et A(4), etc... jusqu'à A(N).

Pour $I = N$, la boucle étant satisfaite, la première expression exécutée sera celle qui suit CONTINUER. AMAX contient maintenant le plus grand des nombres de l'ensemble A; N et la suite des valeurs de A restent inchangés.

L'expression (G) déclenche l'impression directe de N et de AMAX suivant le MODÈLE 2.

Dans l'expression (H) donnant le MODÈLE, le caractère suivant immédiatement H définissant une zone Hollerith contrôle l'espacement; la présence d'un 1 provoque un saut en tête de page. Le texte « LE PLUS GRAND DES » sera imprimé par les roues 1 à 18, la valeur N par les roues 19 à 21 sous forme d'entier décimal, le texte « NOMBRES EST » par les roues 22 à 34 et la valeur de AMAX, convertie en décimale fixe, par les roues 35 à 41. Il faut noter la nécessité de prévoir une position supplémentaire pour la virgule qui s'imprimera avant les deux derniers chiffres de AMAX.

Le programme se termine avec une expression ARRÊT et 7777, apparaît dans la partie adresse du Registre Mémoire au pupitre. Il n'est pas possible de reprendre le calcul en appuyant sur le bouton Départ programme.

La seule variable indicée utilisée dans le programme est A; elle figure donc seule dans l'expression DIMENSION.

Les seules expressions pour lesquelles une fréquence peut être définie sont SI et FAIRE. Le programme prévoit que AMAX sera, en général, plus grand que A(I); cette hypothèse est valable si les A(I) se présentent au hasard, mais elle est fautive si les A(I) sont ordonnés. Il semble également que 100 soit une moyenne raisonnable des valeurs de N et, par suite, du nombre de boucles contrôlées par FAIRE.

Boucle FAIRE avec sortie et retour dans la bande

Étant donnée une matrice carrée $A(N,N)$, trouver les éléments non diagonaux symétriques et les écrire sur une bande binaire.

C ← POUR COMMENT.	SUITE						IDENTIFICATION		
	N°	EXPRESSION FORTRAN							
	1	5	6	7		72	73	80	
		REBOBINER 3							
		FAIRE 3 I=1,N							A
		FAIRE 3 J=I,N							B
		SI(A(I,J)-A(J,I))3,20,3							
	3	CONTINUER							
		FIN BLOC 3							
								
	20	SI(I-J)21,3,21							C
	21	INSCR BANDE 3,I,J,A(I,J)							

Après reboinage de la bande 3, les boucles (A) et (B) recherchent tous les termes $A(I,J)$ de la matrice égaux aux termes $A(J,I)$. Pour des raisons de symétrie, cet examen n'est nécessaire que pour un triangle de la matrice. Chaque fois que la condition $A(I,J) - A(J,I) = 0$ est satisfaite, un transfert est exécuté à l'expression (C) qui renvoie à l'ordre d'impression s'il s'agit d'un terme n'appartenant pas à la diagonale. Trois mots sont alors inscrits en binaire sur la bande 3 : I, J et $A(I,J)$.

Ce sous-programme ne modifie pas les indices I et J et l'on peut donc revenir dans les boucles (A) et (B) et poursuivre ainsi l'examen des termes suivants.

Impression d'une liste triangulaire

Le but du programme ci-après est d'imprimer une matrice symétrique d'ordre 12. L'impression doit être faite dans l'ordre habituel : ligne, colonne; chaque nouvelle ligne de la matrice doit être imprimée en commençant une nouvelle ligne d'impression. Seuls les éléments de la diagonale et ceux au-dessous sont à imprimer.

C ← POUR COMMENT.	SUITE						IDENTIFICATION		
	N°	EXPRESSION FORTRAN							
	1	5	6	7		72	73	80	
		FAIRE 1 I=1,12							
	1	IMPRIMER 50,(A(I,J),J=1,I)							
	50	MODELE (1H07E17.8/1H 7E17.8)							

Chaque expression IMPRIMER provoque l'impression d'un nouveau résultat, donc un retour à la ligne. Le nombre d'expressions IMPRIMER exé-

cutées est déterminé par la boucle FAIRE qui est indexée par le nombre de lignes de la matrice.

Le nombre de termes par ligne est donné par J; une deuxième ligne est imprimée si le nombre de termes est supérieur à 7.

La première ligne commandée par un IMPRIMER provoque un espacement double (0 derrière H) tandis que la seconde provoque un espacement simple (1 blanc derrière H).

Les lignes de la matrice comportant au maximum 7 termes sont imprimées en espacement double; les lignes de la matrice comportant plus de 7 termes sont imprimées sur deux lignes en espacement simple.

Produit de deux matrices

Deux matrices A et B sont respectivement d'ordre l,n et n,m; le produit C est d'ordre l,m, chaque terme étant calculé par :

$$C_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

C ← POUR COMMENT.	SUITE			EXPRESSION FORTRAN	IDENTIFICATION		
	N°	5	6		72	73	80
				FAIRE 1 I=1,L			A
				FAIRE 1 J=1,M			B
				C(I,J)=0.0			
				FAIRE 1 K=1,N			C
	1			C(I,J)=C(I,J)+A(I,K)*B(K,J)			D

Les deux expressions (A) et (B) FAIRE 1 permettent de calculer le terme général en faisant varier les deux indices.

La boucle composée des expressions (C) et (D) effectue la somme des produits partiels. Chacun des produits partiels est cumulé au précédent dans la mémoire C(I,J) dont la valeur finale est celle du terme produit car on a pris la précaution de mettre C(I,J) à zéro avant chaque sommation.

REMARQUE. Les termes sont produits dans l'ordre suivant : C(1,1), C(1,2), ..., C(1,M), C(2,1), ..., C(L,M) car les boucles FAIRE sont satisfaites en commençant par les plus intérieures.

Itérations

Trouver une racine d'une équation du troisième degré par la méthode de Newton avec une précision donnée, connaissant une valeur approchée. A chaque itération, X est remplacé par XN calculé avec la fonction $Y = aX^3 + bX^2 + cX + d$, et sa dérivée, jusqu'au moment où l'écart entre X et XN est inférieur à la valeur maximum permise.

C ← POUR COMMENT	N°	SUITE	EXPRESSION FORTRAN	IDENTIFICATION		
				72	73	80
C			RACINE D EQUATION DU TROISIEME DEGRE PAR LA METHODE DE NEWTON			
			FONCF(X) = ((A*X+B)*X+C)*X+D			A
			DFONCF(X) = (3.*A*X+2.*B)*X+C			B
			XN=VAPPRO			C
	1		X=XN			D
			XN=X-FONCF(X)/DFONCF(X)			E
			SI (ABS(XN-X)-1.0E-5) > 2,1			F
	2				

Les expressions (A) et (B) définissent respectivement la fonction de X et sa dérivée.

La valeur initiale de XN est prise égale à la valeur approchée de la racine dans l'expression (C).

L'expression (D) permet à chaque itération de donner à X la valeur XN précédemment calculée.

L'expression (E) sert au calcul de la nouvelle valeur de XN par la méthode de Newton.

Dans l'expression (F), un test sur la valeur absolue de l'écart entre XN et X arrête l'itération quand cet écart est inférieur à 10^{-5} .

ANNEXE A. TABLE DES CARACTÈRES FORTRAN

CARTE	BANDE DCB	704	CARTE	BANDE DCB	704	CARTE	BANDE DCB	704	CARTE	BANDE DCB	704				
1	1	01	01	A	12 1	61	21	J	11 1	41	41	/	0 1	21	61
2	2	02	02	B	12 2	62	22	K	11 2	42	42	S	0 2	22	62
3	3	03	03	C	12 3	63	23	L	11 3	43	43	T	3 0	23	63
4	4	04	04	D	12 4	64	24	M	11 4	44	44	U	0 4	24	64
5	5	05	05	E	12 5	65	25	N	11 5	45	45	V	0 5	25	65
6	6	06	06	F	12 6	66	26	O	11 6	46	46	W	0 6	26	66
7	7	07	07	G	12 7	67	27	P	11 7	47	47	X	0 7	27	67
8	8	10	10	H	12 8	70	30	Q	11 8	50	50	Y	0 8	30	70
9	9	11	11	I	12 9	71	31	R	11 9	51	51	Z	0 9	31	71
Blanc		20	60	+	12	60	20	-	11	40	40	0	0	12	00
=	8-3	13	13	.	12 8-3	73	33	Fr	11 8-3	53	53	,	0 8-3	33	73
-	8-4	14	14)	12 8-4	74	34	*	11 8-4	54	54	(0 8-4	34	74

NOTE 1. Il existe deux signes — (moins). Dans les cartes Programme origine, seule la perforation 11 peut être utilisée pour le signe « moins ». L'un ou l'autre des signes « moins » peut être employé dans les informations d'entrée du Programme résultant; 8-4 est le signe « moins » de la sortie du Programme résultant.

NOTE 2. Le caractère « Fr » peut être utilisé dans FORTRAN mais uniquement comme texte Hollerith dans une expression MODÈLE.

ANNEXE B. TABLE DES EXPRESSIONS FORTRAN

EXPRESSION	SÉQUENCE NORMALE
a = b	Prochaine expression exécutable
ALLER A n	Expression n
ALLER A n ₁ , (n ₁ , n ₂ , ..., n _m)	Expression du dernier IMPOSER exécuté
IMPOSER n EN v	Prochaine expression exécutable
ALLER A (n ₁ , n ₂ , ..., n _m), i	Expression n _i
SI (a) n ₁ , n ₂ , n ₃	Expression n ₁ , n ₂ , n ₃ suivant que a < = > que 0
VOYANT i	Prochaine expression exécutable
SI (VOYANT i) n ₁ , n ₂	Expres. n ₁ , n ₂ suivant que Voyant i allumé ou éteint
SI (CLÉ i) n ₁ , n ₂	» » suivant que Clé i abaissée ou non
SI DC AC n ₁ , n ₂	» » suivant que Dép. Cap. AC ou non
SI DC MQ n ₁ , n ₂	» » suivant que Dép. Cap. MQ ou non
SI DIV IMP n ₁ , n ₂	» » suivant que Div. impossible ou non
PAUSE ou PAUSE n	Prochaine expression exécutable
ARRÊT ou ARRÊT n	Programme terminé (Arrêt machine)
FAIRE n i = m ₁ , m ₂ ou FAIRE n i = m ₁ , m ₂ , m ₃	Prochaine expression exécutable
CONTINUER	» » »
MODÈLE (Spécification)	Non exécutée
LIRE n, Liste	Prochaine expression exécutable
LIRE ENTREE BANDE i, n, Liste	» » »
PERFORER n, Liste	» » »
IMPRIMER n, Liste	» » »
INSCR SORTIE BANDE i, n, Liste	» » »
LIRE BANDE i, Liste	» » »
LIRE TAMBOUR i, j, Liste	» » »
INSCR BANDE i, Liste	» » »
INSCR TAMBOUR i, j, Liste	» » »
FIN BLOC i	» » »
REBOBINER i	» » »
ESP ARR i	» » »
DIMENSION v, v, v	Non exécutée
ÉQUIVALENCE (a, b, c, .), (d, e, f, .), .	» »
FRÉQUENCE n(i, j, .), m(k, l, .), .	» »

IBM FRANCE
INSTITUT DE CALCUL SCIENTIFIQUE

5, Place Vendôme, PARIS 1^{er}

Tous droits de reproduction réservés

