UNIVERSITE DE NANCY FACULTE DES SCIENCES

EMPLOI DE PL/1

POUR LES PROBLEMES DE LINGUISTIQUE

-=-=-=-

THESE

pour l'obtention du

DOCTORAT de SPECIALITE MATHEMATIQUES (3ème CYCLE)

Soutenue devant le Jury le 3 juillet 1969

par

Madame J. VILLARD

-=-=-

Jury : Mr J. LEGRAS Président

Mr C. GILORMINI

Mr C. PAIR Examinateurs

UNIVERSITE DE NANCY FACULTE DES SCIENCES

EMPLOI DE PL/1

POUR LES PROBLEMES DE LINGUISTIQUE

par Madame J. VILLARD

Université de Nancy

Faculté des Sciences

ANNEE SCOLAIRE 1968/69

DOYEN : M. AUBRY

ASSESSEUR : M. GAY

Doyens honoraires : MM. CORNUBERT - ROUBAULT.

Professeurs honoraires : MM. RAYBAUD - LAFFITTE - LERAY - JOLY - LAPORTE - EICHBORN - CAPELLE - GODEMENT - L. SCHWARTZ - DIEUDONNE - DE MALLEMAN - LONGCHAMBON - LETORT - DODE - GAUTHIER - GOUDET - OLMER - CORNUBERT - CHAPELLE - GUERIN - WAHL.

Maîtres de Conférences honoraires : MM. LIENHART - PIERRET - Mle MATHIEU.

PROFESSEURS

MM.	ROUBAULT	Géologie		GAYET	Physiologie
	VEILLET	Biologie animale		HADNI	Physique
	BARRIOL	Chimie théorique	*	BASTICK	Chimie
	BIZETTE	Physique		DUCHAUFOUR	Pédologie
	GUILLIEN	Electronique		GARNIER	Agronomie
	LEGRAS	Mécanique rationnelle		NEEL	Chimie organique industrielle
	BOLFA	Minéralogie		BERNARD	Géologie appliquée
	NICLAUSE	Chimie	*	CHAMPIER	Physique
	FAIVRE	Physique appliquée	*	GAY	Chimie biologique
	AUBRY	Chimie minérale		STEPHAN	Zoologie
	COPPENS	Radiogéologie	*	CONDE	Zoologie
	DUVAL	Chimie	*	WERNER	Botanique
	FRUHLING	Physique		EYMARD	Calcul différentiel et intégral
	HILLY	Géologie			
	LE GOFF	Génie chimique		LEVISALLES	Chimie organique
	SUHNER	Physique expérimentale		FELDEN	Physique
	CHAPON	Chimie biologique	*	GOSSE	Mécanique physique
	HEROLD	Chimie minérale industrielle	*	DAVOINE	Physique (ENSMIN)
	SCHWARTZ B.	Exploitation minière		HORN	Physique (1° cycle)
	MALAPRADE	Chimie	*	ROCCI	Géologie
*	MANGENOT	Botanique	* Mme	LUMER	Mathématiques
				DELPUECH	Chimie physique

N...	Chimie biologique
N...	Mécanique appliquée
N...	Analyse supérieure
N...	Méthodes mathématiques de la physique
N...	Mécanique rationnelle

(*) Professeur titulaire à titre personnel

PROFESSEURS SANS CHAIRE

Mme	BASTICK	Chimie P. C. Epinal	MM.	FLECHON	Physique P. C.
MM.	GUDEFIN	Physique	Mle	HUET	Mathématiques C. B. G.
	VUILLAUME	Psychophysiologie		VIGNES	Métallurgie
	FRENTZ	Biologie animale		BALESDENT	Thermodynamique, Chimie appliquée (ENSIC
	MARI	Chimie (ISIN)		BLAZY	Minéralogie appliquée (ENSG)
	AUROUZE	Géolog...		JANOT	Physique P. C Epinal
	DEVIOT	Physique du solide		CACHAN	Entomologie appliquée (ENSA)

MAITRES DE CONFERENCES

MM	JACQUIN	Pédologie et chimie agricoles	MM.	BAVEREZ	Chimie (ENSIC)
	MAINARD	Physique M. P.		CHAMBON	Exploitation minière (Mines)
	MARTIN	Chimie P. C.		HUSSON	Physique (ENSEM)
	PAULMIER	Mécanique expérimentale		GERL	Physique
	PROTAS	Minéralogie		WEISSLINGER	Physique
	JOZEFOWICZ	Physico-chimie		ROQUES	Chimie minérale
	JURAIN	Géologie C. B. G.		PERRIER	Mathématiques
	RIVAIL	Chimie appliquée (CUCES)		N...	Mécanique des fluides (ISIN)
	VILLERMAUX	Génie chimique		N...	Mécanique (ISIN)
	METCHE	Biochimie appliquée (Brasserie)		N...	Mathématiques
				N...	Mathématiques P. C.
	PAIR	Mathématiques appliquées		N...	Mathématiques C. B. G.
	BAUMANN	Physique 1° cycle		N...	Physiologie animale
	DURAND	Physique		N...	Mathématiques M. P.
	GRANGE	Physique (ISIN)			
	DEPAIX	Probabilités et statistiques			

CHARGES D'ENSEIGNEMENTS

MM. AMARIGLIO, COEURE, DAVRAINVILLE, GILORMINI, GIRARDEAU, HILY, MAURIN, NOVERRAZ, OVAERT, RUYER, WEBER.

Je remercie Monsieur LEGRAS, Directeur de l'Institut Universitaire de Calcul Automatique, qui a bien voulu me faire l'honneur de présider ce jury.

Ce travail a été effectué sous la direction de Monsieur PAIR.

J'assure de ma reconnaissance Monsieur GILORMINI qui a accepté de participer au Jury.

Que Monsieur CHENIQUE, chef de service à la Compagnie de Pont-à-Mousson, veuille bien trouver l'expression de ma gratitude pour m'avoir permis l'accès à l'ordinateur IBM 360/50.

Cette étude a fait l'objet d'une convention de la Délégation Générale de la Recherche Scientifique et Technique (D.G.R.S.T.) - Convention n° 66 00 230 -

TABLE DES MATIERES

2.2. pointeur

2.3. allocation de mémoire à une variable basée

2.4. recherche d'un élément de liste.

3 - Exemples.

3.1. Création d'une liste

3.2. Classement alphabétique d'une liste

3.3. Adjonction d'éléments dans une liste classée alphabétiquement.

3.4. Consultation d'une liste

3.5. Vérification d'une règle de grammaire.

me partie : Intérêt de PL/1 pour les littéraires, comparaison à Cobol et Algol linguistique.

Chapitre I : Eléments de PL/1 adaptés à la programmation des travaux littéraires,

Chapitre II : Comparaison de ce sous-ensemble à Cobol

Chapitre III : Comparaison de ce sous-ensemble à Algol linguistique.

nexe : Exemples de programmes,

Phrases où il existe des mots donnés.

Procédure donnant un mot d'une phrase, de rang donné.

Edition d'un texte.

Vérification de l'orthographe formelle d'un texte hébreu codé et mise sur bande.

INTRODUCTION
SUR LES LANGAGES DE PROGRAMMATION

INTRODUCTION

SUR LES LANGAGES DE PROGRAMMATION

Un langage de programmation est défini par un vocabulaire et une grammaire. Le vocabulaire est composé des lettres, chiffres, caractères spéciaux tels que les séparateurs , . (; : ... les opérateurs + - * | ... et quelques mots appelés mots-clés ou symboles de base tels que GET , LIST , IF , THEN , DØ , GØ , TØ , END...

Les règles de grammaire permettent d'engendrer toutes les phrases permises du langage.

Exemples :

- Règle de définition d'un identificateur :

Un identificateur est une suite d'au plus 31 caractères alphanumériques (y compris le tiret) commençant par une lettre :

MØT , I , I1 , VØYELLE_A

sont des identificateurs.

- Règle définissant l'affectation d'une valeur à une variable simple.:

<identificateur> = <constante> ;

affectation à I de la valeur 1 : I = 1 ;

Le langage de programmation est l'interprète entre la personne qui désire faire résoudre un problème par l'ordinateur et l'ordinateur. La personne établit la liste des instructions à exécuter à partir d'un algorithme de résolution du problème.

Exemple 1 :

Algorithme de recherche de la voyelle a dans un mot.

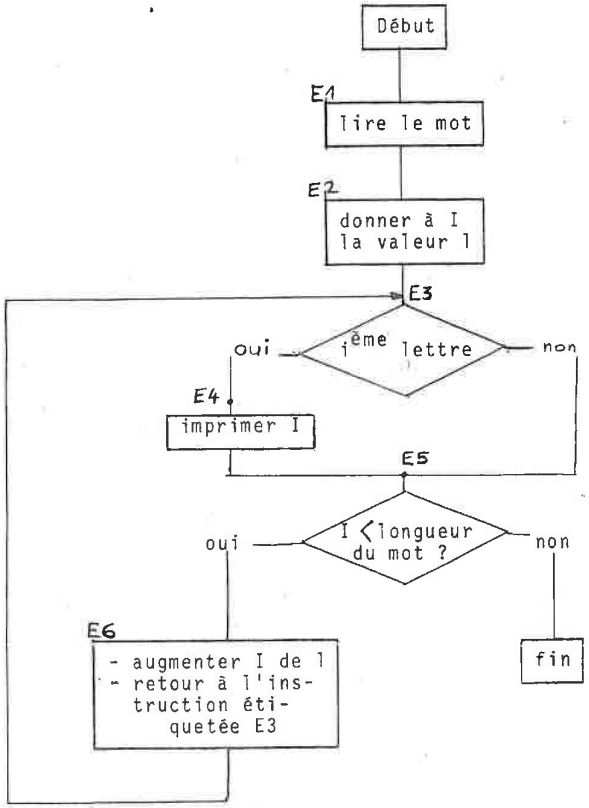
Etant donné un mot, l'on souhaite connaître la ou les positions de a dans le mot.

ex :

1 - a Notion de branchement, de test, de groupe d'instructions.

(le schéma qui représente l'algorithme est appelé organigramme).

Soit I le rang de la lettre désignée dans le mot



Ceci correspond à la suite des instructions suivantes en PL/1 :

```
E1 : GET LIST (MØT);
E2 : I=1;
E3 : IF SUBSTR(MØT,I,1)='a' THEN
E4 : PUT LIST (I);
E5 : IF I<LENGTH(MØT) THEN
E6 : DØ; I=I+1;
     GØ TØ E3;
   END E6;
END;
```

Tous les mots employés dans cette suite d'instructions font partie du vocabulaire PL/1 sauf MØT,I et les étiquettes E1,....E6. Les étiquettes sont reconnues comme telles puisqu'elles précèdent les instructions mais il faut fournir à l'ordinateur des indications sur les identificateurs MØT et I, par exemple : la nature (caractère ou nombre la longueur s'il s'agit d'un mot. Ces indications sont données dans les déclarations.

La suite des déclarations et instructions forme le programme Ainsi le programme de recherche de la voyelle a dans un mot s'écrira :

```
VOYELLE_A1: PRØCEDURE;
     DECLARE MØT CHARACTER(8), I;
     GET LIST (MØT);
     I=1;
   E3: IF SUBSTR(MØT,I,1)='a'
         THEN PUT LIST (I);
       IF I<LENGTH(MØT)
         THEN DØ; I=I+1;
                GØ TØ E3;
            END;
END VØYELLE_A1;
```

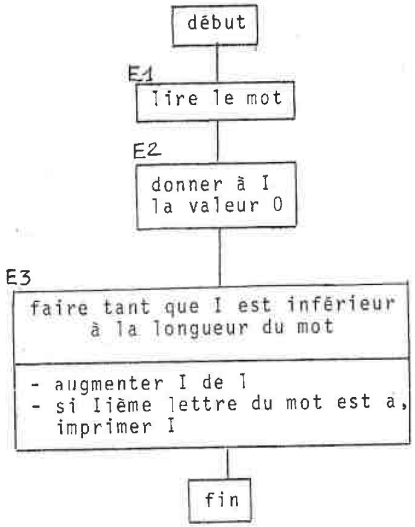
Remarques : 1 - Il est nécessaire d'introduire un blanc entre 2 mots si aucun n'est séparateur ou opérateur.

2 - L'étiquette E3 qui identifie l'instruction à laquelle il y a branchement si I est inférieur à la longueur du mot, de même que VØYELLE A1 qui est le nom du programme élémentaire ou procédure de recherche de la voyelle a dans un mot, sont nécessaires. Les autres sont facultatives.

3 - Les 2 instructions : augmenter I de 1 et retour à l'instruction E3 forment le groupe d'instructions à exécuter dans le cas où I est inférieur à la longueur du mot.

1 - b Notion d'itération :

Nous remarquons qu'il y a comparaison de chaque lettre du mot à a, nous recommençons donc le même processus tant que le rang est inférieur à la longueur du mot, d'où un autre organigramme pour le même problème.



ce qui correspond à la procédure PL/1 :


```
VOYELLE-A2: PRØCEDURE;
     DECLARE MØT CHARACTER(8), I;
 E1: GET LIST (MØT);
 E2: I=0;
 E3: DØ WHILE(I<LENGTH(MØT));
         I=I+1;
         IF SUBSTR(MØT,I,1)='a'
             THEN PUT LIST(I);
 END E3;
END VØYELLE-A2;
```

Exemple 2 :

Algorithme de recherche des voyelles dans un mot.

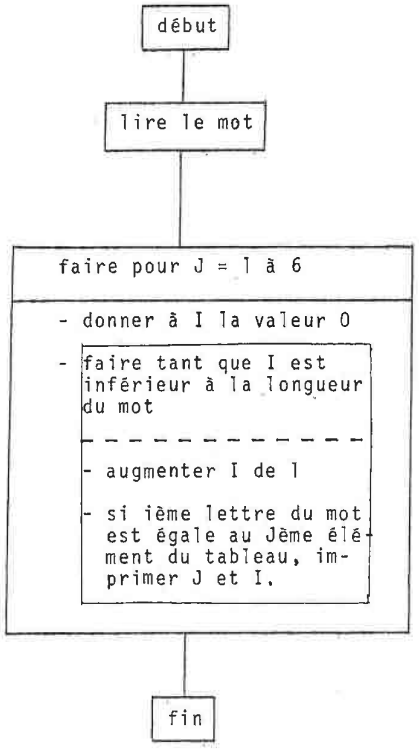
Notion de tableau.

Etant donné un mot, l'on désire connaître la ou les positions des différentes voyelles.

Exemple : voyelle 0 2
y 3
e 4
e 7

Nous pouvons rechercher toutes les voyelles a du mot, les voyelles e, ... les voyelles y, c'est-à-dire répéter le processus présenté dans l'exemple 1 pour chaque voyelle. Nous allons donc ranger les voyelles dans un tableau de 6 éléments dont chaque voyelle sera un élément et répéter le processus pour chaque élément du tableau T (6)

a	e	i	o	u	y



Chaque élément du tableau est repéré par son rang :

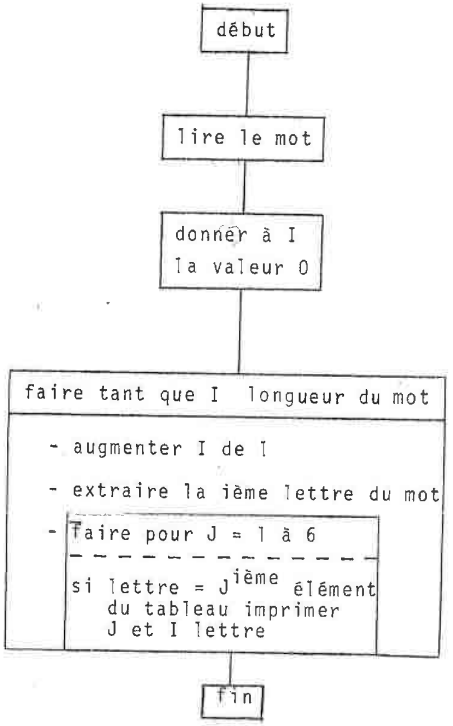
i ⟹ T(3)

```
VØYELLE: PRØCEDURE;
     DECLARE MØT CHARACTER(8),I,J,
        T(6) CHARACTER(1) INITIAL('a','e','i','o','u','y');
     GET LIST(MØT);
     DØ J=1 TØ 6;
```



```
          I=0;
          DØ WHILE (I < LENGTH(MØT));
             I=I+1;
             IF SUBSTR(MØT,I,1)=T(J)
             THEN PUT LIST (J,I);
          END;
       END;
    END VØYELLE;
```

L'extraction de chaque lettre du mot étant plus longue que la recherche d'un élément de tableau, il est préférable de comparer chaque lettre à la liste des voyelles, d'où le nouvel organigramme



```
VØYELLE: PRØCEDURE;
      DECLARE MØT CHARACTER(8), I, J,
         T(6) CHARACTER(1) INITIAL('a','e','i','o','u','y'),
         LETTRE CHARACTER(1);
      GET LIST (MØT);
      I=0;
      DØ WHILE (I < LENGTH(MOT));
         I=I+1;
         LETTRE=SUBSTR(MØT,I,1);
         DØ J=1 TØ 6;
            IF LETTRE=T(J)
               THEN PUT LIST (LETTRE,I);
         END;
      END;
END VØYELLE;
```

Exemple 3 :

Notion de structure.

Une information peut être composée d'éléments de nature différente et pourtant former un tout.

3 - a Définition d'un mot dans le dictionnaire de poche Larousse :

Elle est composée de la suite : mot, nature, définition.

Elément

mot	nature	définition
baliverne	n. f.	discours frivole

Ce qui s'écrit ainsi :

```
1 ELEMENT,
     2 MØT CHARACTER(10),
     2 NATURE CHARACTER(4),
     2 DEFINITION CHARACTER(66),
```

3 - b Algorithme de recherche des débuts de phrases grammaticales

Le manuscrit a été codé par zones :

Zone A : copie du texte manuscrit

Zone B : mentions des abréviations, sigles de correction ...

Zone C : éléments d'analyse morphologique et syntaxique.

La double numérotation de la zone C permet de délimiter les groupes nominaux et verbaux, les locutions adverbiales, les relatifs.

Exemple :

```
2402ZA0.L O R S ,,L E U R ,,A V I N T ,,I ,,G R A N T ,,D O M A C H E ,+++++++
GV1 2402ZB0                          1       *     0 0
    2402ZC0 6        4 R      5 3      3  30  20        10S

GV1 2502ZA0.E*T ,,C E ,,E S T ,,L A ,,O U ,,L I ,,B R A Z ,,S A I N T -++++++++++≠
    2502ZB0                                                  0 0 0 0          ≠
GV1 2502ZC0 8    1 S   5 3  3  6  0, 6  0  20    10S1     *  1     F         ≠
```

Pour chaque début de phrase, on veut obtenir l'information suivante :

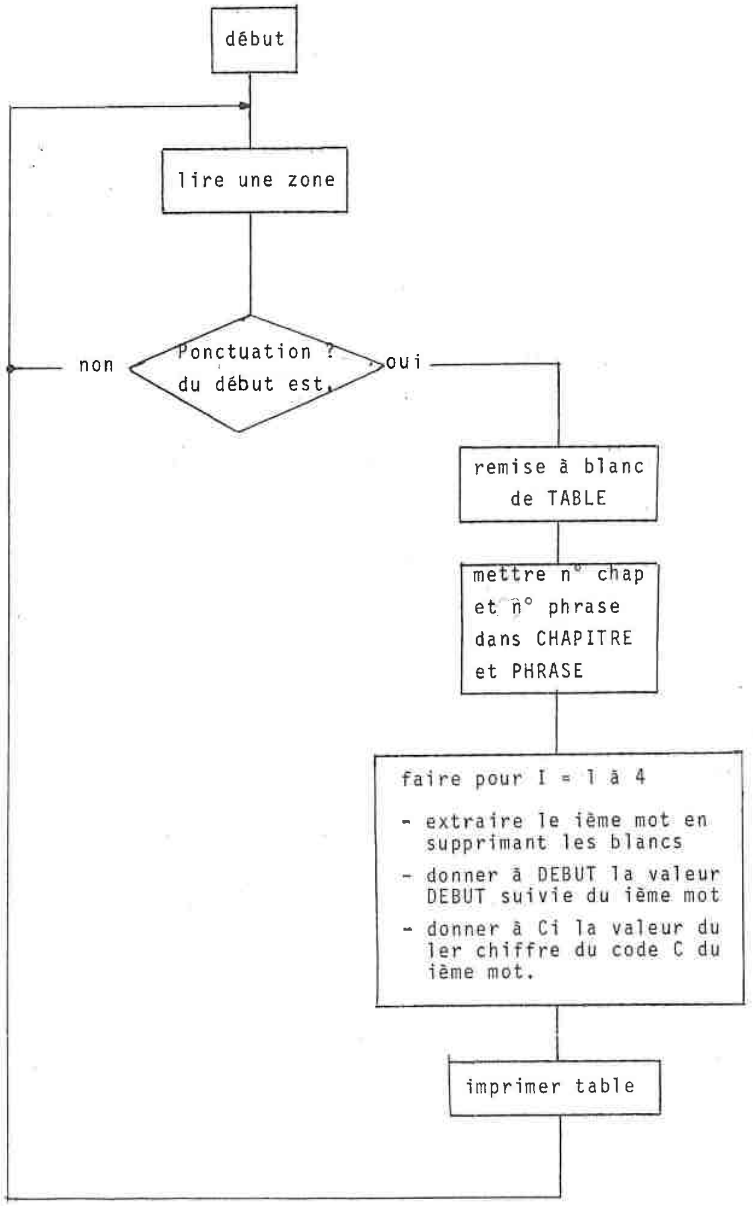
Codes grammaticaux des 4 premiers mots - 4 premiers mots - référence.

C1 C2 C3 C4 — n° chapitre — n° phrase

6 4 5 3 LORS LEUR AVINT I	124	02
8 4 5 6 ET CE EST LA	125	02

Cette information a la structure suivante :

```
1 TABLE,
    2 CODE,
      3 C1,
      3 C2,
      3 C3,
      3 C4,
    2 DEBUT-PHRASE,
    2 REFERENCE,
        3 CHAPITRE,
        3 PHRASE,
```

DEUXIEME PARTIE

DESCRIPTION D'UN SOUS-ENSEMBLE de PL/I

A L'USAGE DES LITTERAIRES ET DES LINGUISTES

CARACTERISTIQUES DES PROBLEMES LITTERAIRES

L'intérêt que portent les linguistes et les littéraires aux ordinateurs a pour but l'automatisation de certains de leurs problèmes tels que l'étude de la forme et de la structure des textes, la documentation et la traduction. Dans la suite, nous faisons une étude rapide de ces problèmes.

I - ETUDE DES TEXTES :

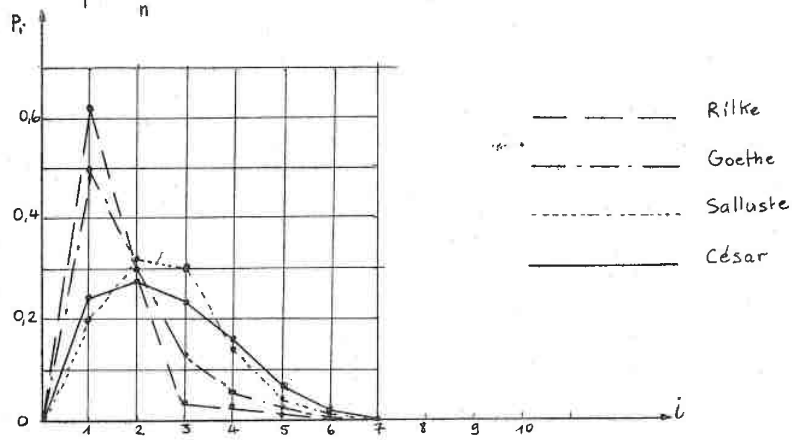
Elle comprend l'analyse statistique d'un texte, la reconnaissance de forme des mots, la vérification de règles de grammaire.
L'analyse statistique d'un texte, s'appuyant sur des comptages d'éléments du texte, définis à priori, précise ainsi par exemple :

- la répartition des voyelles et des consonnes dans un mot, la fréquence des voyelles par rapport aux consonnes,
- la longueur des mots, le nombre moyen de mots par phrase.
- la fréquence relative des groupes de 2, 3, ..., n lettres, leur répertoire et celui des préfixes, désinences ou syllabes.
- la fréquence des classes de mots (la définition de ces classes de mots est du ressort de l'étude grammaticale).

Pour illustrer l'intérêt de l'analyse statistique d'un texte, nous indiquons les résultats d'une étude [4] sur la comparaison des différences stylistiques entre divers auteurs.

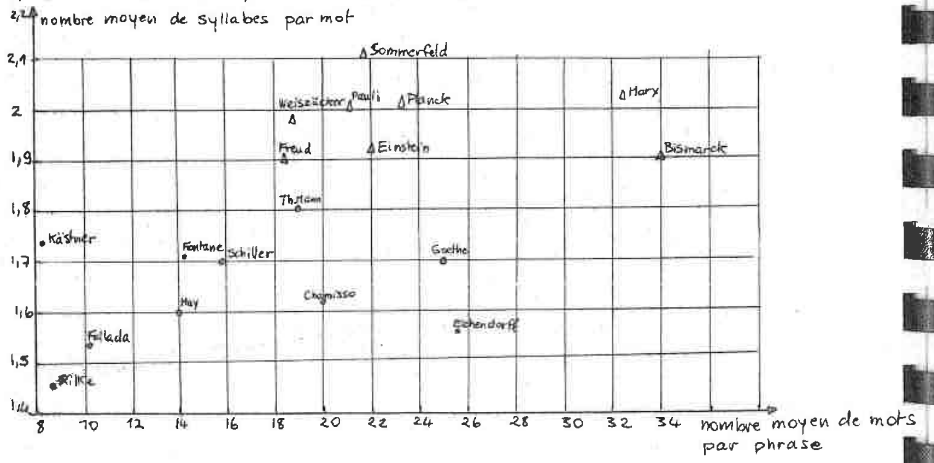
Le premier critère choisi est le nombre Z_i de mots de i syllabes que l'on a compté chez divers auteurs latins et allemands (Salluste, César, Goethe et Rilke).

N désignant le nombre de mots d'un texte, on évalue alors la fréquence relative $P_i = \frac{Z_i}{n}$



L'étude montre que le maximum de la courbe de distribution est atteint pour les monosyllabes chez les auteurs allemands et pour les dissyllabes chez les latins.

Le deuxième critère choisi est le nombre moyen de mots par proposition en fonction du nombre moyen de syllabes par mot, ce qui définit un point caractéristique de l'auteur étudié.



L'étude montre que les points se répartissent en 2 zones disjointes, l'une formée par les écrivains (•), l'autre par les savants, philosophes, hommes politiques ... (Δ).

L'intérêt d'une telle étude réside dans ses applications : ainsi partant d'un texte contesté attribué à l'un des auteurs, étudié par ses oeuvres incontestables, on pourra, au sens habituel de la statistique, dire ou non s'il est de sa main.

L'étude grammaticale permet d'approfondir l'analyse des textes. Dans un premier stade, on effectue une étude morphologique. La partition des éléments d'un texte en classes de mots n'est d'ailleurs pas universelle : certains distinguent les classes de mots à forme invariable (adverbe, préposition, infinitif ...) et les classes de mots à forme variable (forme personnelle du verbe ...), d'autres des classes de mots faisant appel à des critères sémantiques.

Considérons les classes de mots suivantes :

1 substantif	7 conjonction
2 verbe	8 numéral
3 adjectif	9 adverbe
4 pronom	10 nom propre
5 article	11 virgule
6 proposition	12 point

Les fréquences de classes de mots ne donnent pas de renseignements sur la syntaxe de la phrase aussi est-il intéressant de tenir compte de l'ordre de succession des mots appartenant à ces classes de mots. L'ordre de succession peut être représenté par un tableau à 2 dimensions, appelé matrice de transition.

Exemple : Soient les 2 phrases suivantes

Puis-je avoir un fruit ? tu peux prendre une pomme.

Pour ces deux phrases : une question et la réponse à cette question,

la distribution de fréquence est la même :

substantif	1
pronom	1
article	1
verbe	2

mais l'ordre de succession est différent

i \ j	Subst.	Verbe	Pronom	Article	Point
substantif	0	0	0	0	1
Verbe	0	0	1	1	0
pronom	0	1	0	0	0
article	1	0	0	0	0
point	0	0	0	0	0

i \ j	Subst.	Verbe	Pronom	Article	Point
subst.	0	0	0	0	1
verbe	0	1	0	1	0
pronom	0	1	0	0	0
article	1	0	0	0	0
point	0	0	0	0	0

Les différentes valeurs Z_{ij} de la matrice renseignent sur la fréquence de succession des classes de mots Z_i et Z_j.

On peut calculer la fréquence relative

$$P'_{ij} = \frac{Z_{ij}}{\sum_i Z_{ij}} \qquad \sum_j P_{ij} = 1.$$

La méthode des matrices de transition permet de comparer deux textes : pour cela, on note dans un système d'axes perpendiculaires, en abscisse les valeurs P'_{ij} du 1er texte relatives à une transition donnée, en ordonnée les valeurs P'_{ij} du 2ème texte relatives à la même transition.

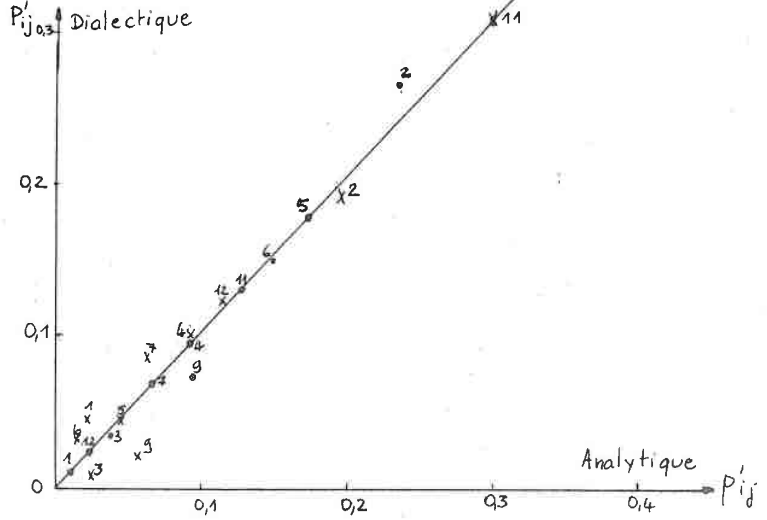
Si les 2 textes sont identiques du point de vue grammatical, tous les points sont situés sur la ligne de 45°.

Nous indiquons les résultats de l'étude 4 faite sur 2 textes de Kant, portant sur un échantillon de 8.000 éléments de phrase choisis dans l' "Analytique transcendantale" et sur un échantillon de même importance choisi dans la "Dialectique transcendantale".

Les classes de mots sont celles indiquées précédemment.

Les transitions des substantifs sont notées par des points., celles des verbes par des croix x.

ainsi : substantif verbe correspond à .2



Après avoir reconnu les diverses structures des phrases d'un texte, nous pouvons aborder le problème inverse :

- définir les règles de grammaire d'un langage
- analyser une phrase pour une grammaire

Exemple : Considérons une règle simplifiée de production d'une phrase : R_i

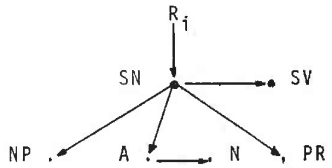
La phrase P est composée d'un syntagme nominal SN suivi d'un syntagme verbal SV. Le syntagme nominal est

soit un nom propre NP

soit un article A suivi d'un nom N

soit un pronom PR

d'où la représentation de la règle R_i

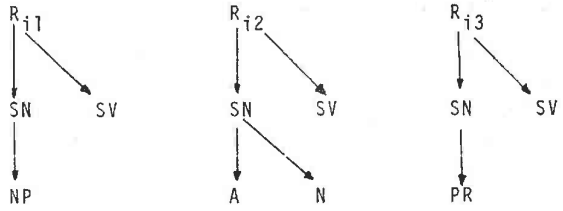


Les flèches horizontales indiquent la succession.

Les flèches verticales indiquent le choix entre plusieurs descendants.

Une phrase correspondant à la suite A - N - SV vérifie la règle R_i.

Dans le graphe représentant la règle R_i, les chemins reliant les noeuds n'ont pas même signification. La définition de la règle et l'analyse d'une phrase seront simplifiées si la règle R_i est décomposée en 3 règles excluant un choix.



II - DOCUMENTATION :

Les tâches habituelles de la documentation sont :

- l'édition de bibliographies
- l'édition de catalogues dont chaque élément est composé d'une référence (nom de l'article ou de l'ouvrage, auteur, éditeur, date de parution) et d'un résumé,
- la recherche de documents répondant à certains critères.

La 1ère tâche peut être automatisée très facilement : lors de l'arrivée d'un nouveau document, on établit une fiche signalétique comprenant le titre du document, le nom de l'auteur, de l'éditeur, la date de parution et éventuellement quelques indications sur le contenu, permettant un classement.

Il est facile d'éditer à partir de ce fichier, la liste des publications d'un auteur, la liste des articles appartenant à une classe donnée .....

La 2ème tâche comporte une partie d'analyse documentaire importante, travail habituel des documentalistes qui est la rédaction d'un résumé.

La 3ème tâche comporte plusieurs phrases : l'indexation du document, l'enregistrement des informations en mémoire et leur consultation. Le but de la consultation est de choisir des documents conformes à certaines spécifications. L'automatisation intervient dans l'enregistrement des informations et le choix des documents, mais l'indexation est faite manuellement.

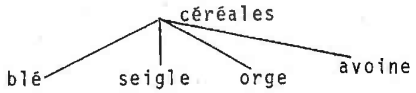
Le documentaliste chargé de l'indexation assigne à chaque document un certain nombre de termes ou mots-clés appartenant au langage documentaire. Le langage documentaire est composé d'un vocabulaire restreint dans lequel sont éliminés les synonymes.

Exemple :

- un texte traitant de la culture des céréales en Europe sera indexé par CULTURE CEREALES EURØPE
- un texte traitant de la culture de blé en URSS sera indexé par CULTURE BLE URSS

Si l'on cherche les documents relatifs à la culture des céréales, le 1er texte sera sélectionné mais le 2ème, donc l'indexation par mots-clés est insuffisante puisque des documents pertinents sont ignorés.

Il est nécessaire d'établir des relations entre les termes du vocabulaire du langage documentaire. Pour cela on dispose d'un lexique documentaire comprenant les mots du langage et les relations qui existent entre eux.



Si les éléments du lexique sont numérotés, chaque élément sera suivi des numéros de ses descendants,
Lors de la recherche d'un texte répondant aux critères précédents CEREALES et EURØPE seront successivement remplacés par leurs descendants et les textes répondant aux nouveaux critères seront sélectionnés.
Cette méthode de sélection permet d'obtenir tous les documents contenant les mots-clés de la question et ses dérivés mais tous ne répondent pas exactement à la question posée aussi introduit-on des relations explicites entre les mots-clés pour préciser la question et l'indexation du document relation d'appartenance
de cause
d'effet .....,
La fiche relative à un document aura la forme suivante :

Référence	
01	01 02 10
02	1 3
mots clés	

Les relations sont numérotées

mot 01 $\xrightarrow{R_1}$ mot 02

mot 02 $\xrightarrow{R_3}$ mot 10

- TRADUCTION : La traduction d'un texte d'une langue dans une autre nécessite une analyse grammaticale et sémantique du texte, un dictionnaire.
Il semble que lorsque la documentation automatique sera capable de détecter la signification d'un texte et de la transposer dans son langage propre, le problème de traduction sera résolu.

IV - CONCLUSION :

Les structures d'information rencontrées sont très variées : mot-phrasé, graphe, tableau, fichier,
Ces informations sont très souvent de longueur variable.
Pour pouvoir traiter les problèmes exposés de manière efficace, le langage de programmation utilisé, doit possèder des entrées-sorties évoluées, des fonctions permettant d'agir sur des chaînes de caractères, des tableaux et des listes, des opérations arithmétiques simples pour l'analyse statistique. Certains traitements se répètent souvent aussi est-il nécessaire de pouvoir appeler une procédure pour effectuer ce travail, d'où la notion de procédures indépendantes rangées en bibliothèque et disponibles à tout instant.

DEUXIEME PARTIE

DESCRIPTION D'UN SOUS-ENSEMBLE de PL/1

A L'USAGE DES LITTERAIRES ET DES LINGUISTES

INTRODUCTION

PL/1 a été créé en juin 1964 par I.B.M., révisé et modifié depuis, la dernière révision date de décembre 1966.

A l'encontre des langages de programmation évolués précédents tels que ALGOL, FORTRAN ou COBOL, PL/1 n'est pas un langage spécialisé dans un domaine ; sa portée est générale. Il est toutefois possible de définir différents sous-ensembles du langage relatifs à des applications et à des niveaux de complexité différents.

Ce rapport définit le sous-ensemble de PL/1 permettant la programmation des travaux de linguistique.

CHAPITRE I

ELEMENTS DU LANGAGE

- Symboles de base :

On dispose pour écrire les programmes d'un jeu de 60 caractères :

- les caractères alphabétiques : les 26 lettres

$
@
#

utilisés pour former les identificateurs, les étiquettes, les commentaires, les mots-clés.

- les chiffres

- les caractères spéciaux qui sont :

les opérateurs arithmétiques + - * /

les opérateurs de relation > = <

les opérateurs logiques ¬ & |

les séparateurs . , ; % ' _ ? : blanc

les parenthèses ( )

Deux mots d'une phrase sont séparés soit par un opérateur, soit par un séparateur.

II - Les valeurs :

Les informations traitées par l'ordinateur peuvent être des nombres, des chaînes de caractères, des chaînes de bits.

2.1. Les nombres :

Exemple : 550 12.1 .1 0.1

$3*10^2$ s'écrit 3E2

Un nombre sans signe est représenté par une partie entière suivie d'une partie fractionnaire, suivie d'un facteur de cadrage qui a la forme suivante : E exposant,

On peut omettre - la partie entière ou fractionnaire si elle est nulle.
- le facteur de cadrage si l'exposant est nul.
- le signe de l'exposant s'il est +.

2.2. Les chaînes de caractères :

'MØT' '500' 'blanc'

(3)'A' représente la chaîne 'AAA'

Une constante de chaîne de caractères consiste en zéro, un ou plusieurs caractères placés entre apostrophes, la chaîne peut être précédée d'un entier entre parenthèses indiquant la répétition .

Si la chaîne contient une apostrophe, celle-ci est remplacée par 2 apostrophes successives.

Exemple : 'C''EST'

2.3. Les chaînes de bits :

La chaîne de 1 bit ou valeur logique est utilisée pour indiquer la présence ou l'absence d'une caractéristique, le résultat d'une comparaison (le résultat de la comparaison est vrai chaîne='1'B, faux chaîne='0'B).

Exemple : les livres d'une bibliothèque sont classés suivant la nationalité de leur auteur - français
- étranger

Cette classification peut être représentée par une chaîne de 1 bit

'1'B —— auteur français

'0'B —— auteur étranger

On peut compléter cette classification en précisant si les livres sont imprimés en français (bit=1) ou non (bit=0).

La classification est alors représentée par une chaîne de 2 bits dont les différentes valeurs correspondent aux cas suivants :

livre d'un auteur français	'11'B
" " " étranger traduit	'01'B
" " " " non traduit	'00'B

Une chaîne de bits est une chaîne des caractères 1 ou 0 suivie de la lettre B.

III - Les variables :

3.1. Introduction :

Considérons un texte dont les phrases ont la structure suivante : groupe sujet, groupe verbal, groupe complément.

Lors de la lecture du texte, chaque groupe prend une nouvelle valeur qui est une chaîne de caractères (ou éventuellement garde la même) à chaque phrase; mais il porte toujours le même nom.

Exemple :

Un langage de programmation est défini par un vocabulaire et une grammaire.

GS GV GC

Les règles de grammaire permettent d'engendrer les phrases.

GS GV GC

GS, GV et GC sont appelées variables simples, les valeurs prises sont des chaînes de caractères.

Les variables peuvent être aussi groupées, en tableaux, structures ou tableaux de structures.

GS suivi de GV suivi de GC forme la PHRASE, la variable PHRASE est une structure.

3.2. Variable simple : est représentée par un identificateur.

Un identificateur est une chaîne de caractères alphanumériques (caractères alphabétiques, chiffres et trait d'union) commençant par une lettre.

Exemple : MØT

G-S

Elle peut prendre pour valeurs des valeurs de l'un des types définis précédemment.

.3. Tableau :

xemple : Terminaison des conjugaisons.

Verbes du 1er groupe à l'indicatif.

INDICATIF(6,4)	Présent	Imparfait	Futur	Passé simple
1ère pers.	e	ais	erai	ai
2ème	es	ais	eras	as
3ème	e	ait	era	a
1ère	ons	ions	erons	âmes
2ème	ez	iez	erez	âtes
3ème	ent	aient	eront	èrent

e tableau identifié par INDICATIF a 6 lignes et 4 colonnes, il est it bidimensionnel. Il est composé de 24 éléments qui ont pour valeur es chaînes de caractères.

Un tableau est un ensemble d'éléments de même type, il est présenté par un identificateur, il peut être à une ou plusieurs mensions, ces dimensions sont associées à l'identificateur de bleau lors de la déclaration : ainsi INDICATIF (6,4).

les éléments sont à valeurs numériques, ils ont même représentation, ce sont des chaînes, elles ont même longueur si la longueur est xe, même longueur maximum si la longueur est variable.

4. Variable indicée : Chaque élément d'un tableau est repéré par nom du tableau et son rang dans le tableau, cet ensemble nom-rang appelle variable indicée.

emple :

iez terminaison de la 2ème personne du pluriel de l'imparfait de l'indicatif est identifié par INDICATIF(5,2).

Une variable indicée est définie par un identificateur de tableau suivi d'une liste d'indices entre parenthèses.

Il est possible d'isoler tous les éléments d'un tableau relatifs à la variation d'un ou plusieurs indices.

Exemple :

Les terminaisons du futur (3ème colonne du tableau) forment un tableau à une dimension, elles seront désignées par INDICATIF(*,3).

Les terminaisons de verbes à la 3ème personne du singulier (3ème ligne du tableau) seront désignées par INDICATIF(3,*).

Exemples d'utilisation d'un tableau à 2 dimensions.

1 - Emploi de l'article défini

ARTICLE(3,2)		nom commençant par une	
		voyelle	consonne
article défini	le ou la	0	1
	l'	1	0
	les	1	1

Ce tableau indique qu'il y a incompatibilité séquentielle entre le ou la et un mot commençant par une voyelle, entre l' et un mot commençant par une consonne, qu'il y a compatibilité séquentielle entre l' et un mot commençant par une voyelle.

2 - Fréquence des voyelles dans une liste de n mots

FREQUENCE(n,6)	a	e	i	o	u	y
arbre	1	1	0	0	0	0
image	1	1	1	0	0	0
caractère	2	2	0	0	0	0
fréquence	0	3	0	0	1	0

3.5. Structure : Une structure est une suite de composantes dont chacune est une variable, simple, un tableau ou une autre structure.

Exemple : Soit un livre perforé sur cartes de la manière suivante : chaque carte comprend une partie référence et une partie texte

MALRAUX	L''ESPOIR	1	3	6	1	Utilisés chaque jour contre Teruel,réparés

Les composantes sont la référence et le texte.

Le texte est une variable simple (chaîne de caractères).

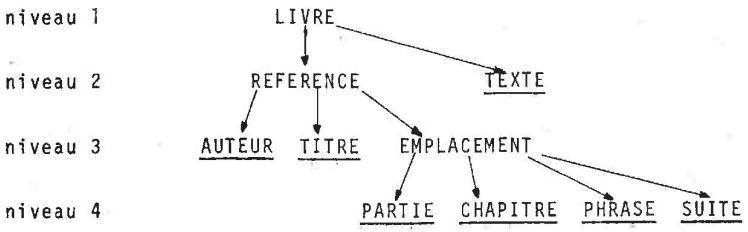
La référence se subdivise en : nom de l'auteur, nom de l'ouvrage, emplacement du texte dans l'ouvrage qui comprend les éléments simples suivants : partie, chapitre, phrase et partie de phrase.

Cette structure peut être décrite ainsi :

```
1 LIVRE,
    2 REFERENCE,
        3 AUTEUR,
        3 TITRE,
        3 EMPLACEMENT,
            4 PARTIE,
            4 CHAPITRE
            4 PHRASE,
            4 SUITE,
    2 TEXTE,
```

LIVRE est la structure principale, REFERENCE et EMPLACEMENT sont des structures secondaires. Seuls les éléments terminaux de la structure prennent une valeur, ce sont soit des éléments simples, soit des tableaux d'éléments simples, ici : AUTEUR, TITRE, PARTIE, CHAPITRE, PHRASE, SUITE, TEXTE.

Représentation de la structure LIVRE sous forme arborescente :



Les éléments terminaux sont soulignés.

A chaque suite de composantes est attribué un nombre entier ou numéro de niveau et à chaque composante un nom. Les numéros de niveau sont attribués par ordre croissant, le passage d'un numéro de niveau au suivant correspond à une subdivision de la composante. La description d'une structure apparaît donc comme une suite de numéros de niveau et d'identificateurs.

3.6. Variable qualifiée : On repère une composante de structure par son nom précédé par les noms des structures dont la composante fait partie, la variable ainsi obtenue est appelée variable qualifiée.

Ainsi 'MALRAUX' sera identifié par : LIVRE.REFERENCE.AUTEUR.

La qualification permet de différencier des variables de même nom employées dans des structures différentes; toutefois il n'est pas nécessaire que la séquence des noms comprenne toutes les structures successives, il suffit qu'elle en contienne suffisamment pour lever toute ambiguité.

Exemple : Considérons la structure suivante :

Malraux	Les conquérants	roman	1928

```
1 EDITIØN
    2 AUTEUR
    2 TITRE
    2 TYPE
    2 PARUTIØN
```

Les conquérants' sera désigné par EDITION.TITRE

L"espoir' " " " LIVRE.TITRE

ne variable qualifiée est définie par une liste d'identificateurs
e niveau croissant, séparés par des points.

.7. Les éléments d'un tableau peuvent être des structures auquel
as on parle de tableau de structures.

Liste des ouvrages par auteur.

haque élément de tableau a la forme suivante :

Auteur	nb d'ouvrages	titre	type	1er ouvrage année de parution	2ème ouvrage...

```
1 LISTE(N)
    2 AUTEUR
    2 NØMBRE
    2 ØUVRAGE(NØMBRE)
      3 TITRE
      3 TYPE
      3 PARUTION
```

'année de parution du 1er ouvrage du 10ème auteur sera désigné
ar PARUTIØN(10,1)

ou LISTE(10).ØUVRAGE(1).PARUTIØN

CHAPITRE 2

INSTRUCTION D'AFFECTATION - EXPRESSION ET FONCTION

L'instruction d'affectation permet de modifier la valeur d'une ou plusieurs variables. La nouvelle valeur est une constante, la valeur d'une autre variable ou le résultat d'un calcul ou expression.
Une expression se présente sous la forme d'une suite de variables et de valeurs appelées opérandes, reliées par des opérateurs.

Exemples :

```
MØT='IMAGE';
```

affectation à la variable de chaîne de caractères MØT de la constante 'IMAGE'. C'est une affectation scalaire simple.

```
I,J=1;
```

donner à I et J la valeur 1. C'est une affectation scalaire multiple.

```
N=I*2+J;
```

affectation de la valeur de l'expression I*2+J à la variable N.
Le type de la (ou des) variable doit être compatible avec le type de la valeur résultante de l'expression ; la valeur est convertie, si nécessaire, suivant les caractéristiques de la variable (par exemple troncature à droite de la chaîne valeur si celle-ci a une taille supérieure à celle de la variable) ; si la conversion est impossible, il y a erreur.

Exemple : A chaîne de caractères ayant la valeur 'RETØURNER'
C chaîne de 6 caractères
D chaîne d'au maximum 6 caractères

```
C=A;      C prend la valeur 'RETØUR'
C='LE';          "        'LE    '
```

```
D=A;              D prend la valeur 'RETØUR'
D='LE';                    "       'LE'
```

Suivant la nature des variables, on distingue :

- affectation scalaire qui modifie la valeur d'un élément simple, et les expressions scalaires.
- affectation de tableau qui modifie la valeur de tous les éléments d'un tableau, et les expressions de tableaux.
- affectation de structure qui modifie la valeur de tous les éléments d'une structure ou de certains éléments d'une structure, et les expressions de structures.

I - AFFECTATION SCALAIRE - EXPRESSION SCALAIRE :

Les opérandes d'une expression scalaire sont des éléments simples : valeurs, variables simples, indicées ou qualifiées. La variable apparaissant dans l'affectation scalaire est simple, indicée ou qualifiée.

Exemples : 1) N variable de type arithmétique

```
N=N+1;
```

2) ETAT variable qui identifie une chaîne de 1 bit

```
ETAT='1'B
```

3) variables de type chaîne de caractères

```
B='TØURNER';
PREF='RE'
A=PREF|| B;
```

Cette instruction affecte à la variable A la valeur 'RETØURNER'.

Le type de l'expression dépend du type des opérateurs :

. opérateurs arithmétiques + - * /

. opérateur de concaténation ||

. opérateurs de comparaison $>, \neg>, >=, <, \neg<, <=, =, \neg=$

. opérateurs logiques & | $\neg$

Le calcul d'une expression se fait de proche en proche, une étape du calcul consiste en une opération élémentaire : opérande opérateur opérande, le résultat de l'opération élémentaire est un opérande pour l'étape suivante. Les opérandes sont convertis, si nécessaire, suivant le type de l'opérateur.

Pour déterminer l'ordre d'évaluation de l'expression, la priorité des opérateurs a été définie comme suit :

$\neg$, signes +, - unaires plus haute priorité
* , /
+ , -

$>=, >, \neg>, =, \neg=, <=, <, \neg<$

&

|

||

plus basse priorité

Dans le cas d'opérateurs de même priorité, les opérations s'effectuent de la gauche vers la droite.

Exemple :

Calcul de l'expression 2*N1+N2/2-1

. Opération 1 2*N1 ⟶ R1+N2/2-1

. Opération 2 N2/2 ⟶ R1+R2-1

. Opération 3 R1+R2 ⟶ R3-1

. Opération 4 R3-1 ⟶ R

On peut modifier la priorité en mettant une partie de l'expression entre parenthèses ; cette partie, calculée avant l'exécution de l'opération associée, est traitée comme une opérande.

Exemple : Comparons les expressions B/2*A et B/(2*A)

B/2*A	B/(2*A)
. Opération 1 B/2 ⟶ R1*A	. Opération 1 2*A ⟶ B/R'1
. Opération 2 R1*A ⟶ R	. Opération 2 B/R'_1 ⟶ R'

Si B a la valeur 8 et A la valeur 2

B/2*A vaut 8 B/(2*A) vaut 2.

1.1. Opération arithmétique :

Les opérandes doivent avoir des valeurs numériques pour que l'opération soit possible.

Exemple : N variable de type arithmétique

C variable de type chaîne de caractères

```
N=N*C;
```


La conversion de C en valeur numérique n'est possible que si la chaîne de caractères est une suite de chiffres.

C = '256' N = 20

N = N * C ⟹ N = 5120

C = N * C C = '256' N = 20

L'expression N * C = 5120 valeur numérique convertie en une chaîne de caractères pour être affectée à C C = '5120'.

1.2. Opération de concaténation :

Les opérandes ont pour valeur des chaînes de caractères.

La conversion de nombres ou chaîne de bits en chaîne de caractères est toujours possible.

Exemple : X a pour valeur la chaîne 'CHANTER'

X, Y, Z variables de type chaîne de caractères

N variable de type arithmétique dont la valeur est 3.

```
Y='ILS '|| X ||'ØNT';
```

Y prend la valeur 'ILS CHANTERØNT'

```
Z=X ||' '||N ||' FØIS';
```

Si Z a été déclaré avec une longueur suffisante

Z prend la valeur 'ILS CHANTERONT 3 FOIS';

1.3. Opération de comparaison ou relation :

Exemple : A > B

Si la valeur de A est supérieure à la valeur de B, la relation A > B est vraie, sinon elle est fausse.

Le résultat d'une comparaison est une valeur logique :

'1'B si la relation est vraie, '0'B si la relation est fausse.

Il existe 3 types de comparaison :

- comparaison algébrique entre valeurs numériques

a + b > c a, b, c ayant pour valeur des nombres

- comparaison alphabétique entre des chaînes alphabétiques

'ARBRE' < 'ARDØISE'

cette relation est vraie, elle signifie que, dans l'ordre alphabétique, ARBRE est placé avant ARDØISE.

La comparaison se fait de gauche à droite par paire, l'opérande le plus court est complété à droite par des blancs.

- comparaison de bits entre chaînes de bits :

B = ETAT = ¬'0' B

B prend la valeur '1'B si la relation est vraie, c'est-à-dire si ETAT a la valeur '1'B.

La comparaison se fait de gauche à droite par paire, l'opérande le plus court est complété à droite par des zéros.

1.4. Opération logique :

Les opérandes ont pour valeur des chaînes de bits, l'opérande le plus court est complété à droite par des zéros.

Une expression logique est, dans la plupart des cas, une suite de relations reliées par des opérateurs logiques. Elle permet de regrouper des comparaisons successives.

Exemple 1 : Une phrase se termine soit par ., soit par ;

en décrivant le texte, caractère par caractère, on sera à la fin d'une phrase si le caractère = . ou le caractère = ;

d'où l'expression logique EL

EL=C='.' | C=';' ;

L'opération est effectuée bit par bit, chaque position de bit a la valeur définie par la table ci-après :

X	Y	¬X	X & Y	X | Y
1	1	0	1	1
1	0	0	0	1
0	1	1	0	1
0	0	1	0	0

Exemple 2 : Enquête sur le niveau des études en fonction du sexe, de l'âge, du domicile.

Chaque fiche a la structure suivante :

numéro	sexe o m 1 f	âge	domicile ville	 campagne	niveau d'études prim.	 Sec.	 Sup.

```
1 FICHE,
    2 NØ CHAR(5),
    2 SEXE BIT(1),
    2 AGE FIXED(2),
    2 DØMICILE,
      3 V BIT(1),
      3 C BIT(1),
    2 NIVEAU,
      3 (PRIM, SEC, SUP) BIT(1);
```

La présence d'une caractéristique dans les structures secondaires DØMICILE et NIVEAU est indiquée par la valeur 1 dans la position

correspondante, la valeur 0 est réservée au cas d'une fiche incomplètement remplie.

On désire connaître le nombre de personnes entre 25 et 60 ans (et) ayant fait des études supérieures ou secondaires ; ce sont les fiches pour lesquelles l'expression logique ci-dessous est vraie.

$$\underbrace{AGE > 25 \ \& \ AGE < 60}_{25 < \text{âge} < 60} \quad \& \quad \underbrace{(SUP \mid SEC)}_{\text{études supérieures ou secondaires.}}$$

1.5. Conversion des opérandes : Elle permet d'opérer sur des variables de types différents dont les valeurs sont comparables.

1.5.1. Conversion d'une chaîne de bits en chaîne de caractères et inversement.

Les bits 1 et 0 deviennent les caractères 1 et 0 et inversement.

Les caractères alphanumériques autres que 1 et 0 ne peuvent être convertis en chaînes de bits.

1.5.2. Conversion d'une chaîne de caractères en nombre et inversement.

Elle n'est possible que si la chaîne de caractères est une suite de chiffres, la conversion inverse est toujours possible.

'256' $\rightleftharpoons$ 256

1.5.3. Conversion d'un nombre en une chaîne de bits et inversement : elle est toujours possible.

- AFFECTATION DE TABLEAU - EXPRESSION DE TABLEAU :

Les opérandes d'une expression de tableau sont des tableaux ou des scalaires. Les tableaux apparaissant dans l'expression et le tableau auquel est affecté le tableau de valeurs doivent avoir des bornes identiques.

L'instruction d'affectation de tableau est équivalente à une séquence d'instructions d'affectation scalaire.

Exemple : présent d'un verbe du 1er groupe

```
PRESENT=PRØNØM || 'CHANT' || INDICATIF(*,1);
```

est équivalent à

```
PRESENT(1)=PRØNØM(1) || 'CHANT' || INDICATIF(1,1);
.....
PRESENT(6)=PRØNØM(6) || 'CHANT' || INDICATIF(6,1);
```

Remarque : Le résultat d'une position peut être affecté par l'évaluation et l'affectation d'une position précédente.

Exemple : Soient les tableaux de 5 éléments A et B.

```
A=B*A(1);
```

La nouvelle valeur de A(1) sera utilisée pour calculer A(2),..., A(5).

III - AFFECTATION DE STRUCTURE - EXPRESSION DE STRUCTURE :

On distingue 2 types d'affectation :

3.1. Affectation normale :

Les opérandes sont des structures de même composition ou des scalaires.

L'instruction d'affectation de structure est équivalente à une suite d'instructions d'affectation scalaire sur tous les éléments terminaux de la structure.

Exemple : Soient les structures

```
1 A, 2 B, 3 C, 3 D,
    2 E,
1 L, 2 M, 3 N, 3 Ø,
    2 P
```

A=A || L; est équivalent à

```
C = C || N;
D = D || Ø;
E = E || P;
```

3.2. Affectation par nom :

Les opérandes sont des structures ou des scalaires.

L'évaluation de l'expression et l'affectation se font sur les éléments terminaux identifiés par la même variable qualifiée (la qualification faisant intervenir toutes les structures secondaires dont fait partie l'élément).

Exemple : considérons un texte perforé sur cartes de la manière suivante :

auteur	titre	partie	n° de chapitre	n° phrase	suite	texte

On lui fait correspondre la structure suivante :

```
1 LIVRE
    2 REFERENCE
        3 AUTEUR
        3 TITRE
        3 EMPLACEMENT
            4 PARTIE
            4 CHAPITRE
            4 PHRASE
            4 SUITE
    2 TEXTE
```

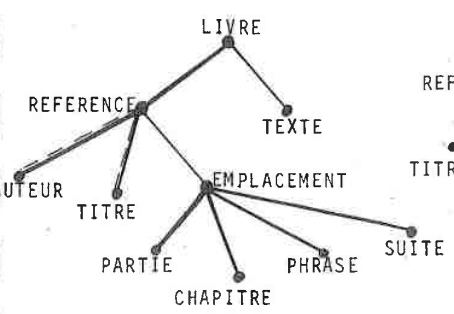
Pour chaque livre, on veut construire une table contenant les éléments suivants :

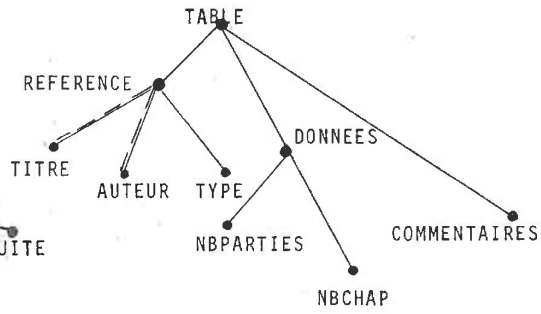
titre	auteur	type d'ouvrage	nombre de parties	nombre de chapitres	commentaires

On lui fait correspondre la structure suivante :


```
1 TABLE
    2 REFERENCE
        3 TITRE
        3 AUTEUR
        3 TYPE
    2 DONNEES
        3 NBPARTIES
        3 NBCHAP
    2 COMMENTAIRES
```

Représentons chaque structure sous forme d'arbre





Les variables qualifiées REFERENCE.AUTEUR et REFERENCE.TITRE se retrouvent dans les 2 structures donc l'instruction

```
TABLE=LIVRE, BY NAME;
```

équivaut aux instructions

```
TABLE.TITRE=LIVRE.TITRE;
TABLE.AUTEUR=LIVRE.AUTEUR;
```

Remarque : si la 2ème structure était définie par

```
1 TABLE
    2 TITRE
    2 AUTEUR
    2 DONNEES
    .........
```

Il n'y aurait pas, dans les 2 structures, des éléments terminaux identifiés par la même variable qualifiée et l'instruction d'affectation précédente n'engendrerait aucune instruction.

Dans l'affectation de structure par nom, il n'est pas nécessaire que la composition de toutes les structures soit la même.

- FONCTIONS INCORPOREES :

Une fonction incorporée est un algorithme de calcul mis en mémoire une fois pour toutes, on indique le nom de la fonction et les éléments permettant le calcul ou arguments.

Exemple : LENGTH (MØT) donnera la longueur de la chaîne identifiée par MØT.

4.1. Fonctions arithmétiques :

Les arguments peuvent être des expressions quelconques dont la valeur est convertie, si besoin est, en nombre. La valeur renvoyée est un nombre

ABS (X) donne la valeur absolue de X

MAX (X, Y) donne la plus grande valeur parmi X, Y

FLOOR (X) donne le plus grand entier n'excédant pas X.
Si X vaut 3,59 FLOOR (X) vaut 3

SIGN (X) prend la valeur 1 si X > 0
0 si X = 0
-1 si X < 0

4.2. Fonctions de chaînes :

Les arguments peuvent être des expressions quelconques, l'argument est converti, si besoin est, en chaîne avant l'appel de la fonction.

- Fonctions définissant un nombre

LENGTH (X) donne la longueur actuelle de la chaîne X

INDEX (X,Y) donne - le rang du caractère de X à partir duquel Y est contenu dans X
- zéro si Y n'est pas contenu dans X ou si l'un des 2 arguments est de longueur nulle.

Exemple : décomposition d'un texte en phrase

texte | phrase 1 | . | phrase 2 | . | phrase 3 | . | .....

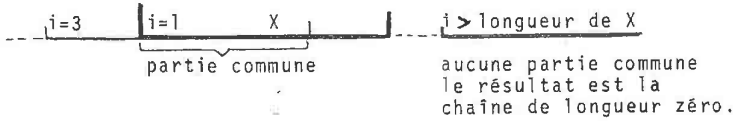
I=INDEX (PHRASE,'.') donne le rang du 1er point.

Fonctions définissant une chaîne

SUBSTR (X,i1,j1) (X chaîne, i1 et j1 expressions quelconques dont les valeurs entières sont i et j).

Complétons la chaîne X à gauche et à droite par des blancs, SUBSTR (X,i1,j1) donne la chaîne des caractères communs à X et à la chaîne de j caractères commençant au $i^{ème}$ caractère de la chaîne X complétée.

Si j1 n'est pas spécifié, SUBSTR (X,i1) donne la chaîne de caractères communs à X et à la chaîne de caractères commençant au $i^{ème}$ caractère de la chaîne X complétée.



REPEAT (X, n) X concaténé n fois avec lui-même.

Exemple 1 : Soit X la chaîne 'REGARDER'

SUBSTR (X,3) ⟹ 'GARDER'

SUBSTR (X,2,5) ⟹ 'EGARD'

SUBSTR (X,1,6) ⟹ 'REGARD'

Exemple 2 : Décomposition d'un texte en phrases

rang du 1er point	I=INDEX (TEXTE,'.')
1ère phrase	PH1=SUBSTR (TEXTE,1,I-1)
texte amputé de la 1ère phrase	TEXTE=SUBSTR (TEXTE,I+1)
rang du 2ème point	I=INDEX (TEXTE,'.')

Exemple 3 : Recherche de mots ayant même radical

INDEX (MØT,RAD) a une valeur non nulle si RAD est contenu dans MØT, si la valeur est supérieure à 1, on pourra isoler le préfixe

```
MØT='RETØURNER'; RAD='TØUR';
I=INDEX(MØT,RAD);
```

SUBSTR(MØT,1,I-1) donne le préfixe RE.

4.3. Fonctions de manipulation de tableaux :

Les arguments sont des expressions de tableaux et les valeurs renvoyées sont scalaires

SUM(X) X tableau de n éléments

La valeur est la somme des n éléments de X

PROD(X) La valeur est le produit des n éléments de X

H BOUND(X,S) donne la borne supérieure actuelle de la n^{ième} dimension du tableau (s converti en un entier n).

Exemple : Calcul de la fréquence des voyelles dans une liste à partir du tableau FREQUENCE (n,6)

SUM(FREQUENCE(*,1)) donne la fréquence de la voyelle a dans cette liste de n mots.

4.4. Fonctions incorporées de tableaux et de structures :

Les fonctions arithmétiques et les fonctions de chaînes peuvent avoir des expressions de tableaux ou de structures comme arguments, sauf

dans le cas où des constantes entières sont nécessaires. La fonction est calculée pour chaque élément.

Exemple : Recherche de la 2ème lettre des mots rangés dans le tableau RAD

$$RAD\begin{pmatrix}'mang'\\'coup'\\'donn'\end{pmatrix}\ SUBSTR\ (RAD^{2}_{i},1) \longrightarrow \begin{pmatrix}'a'\\'o'\\'o'\end{pmatrix}$$

4.5. Utilisation de la fonction SUBSTR comme pseudo-variable :

La fonction SUBSTR est dite utilisée comme pseudo-variable si elle apparaît dans une instruction d'affectation à gauche de =, dans les instructions GET et DØ. Dans le cas de l'affectation, ceci permet de modifier une partie d'une chaîne de caractères.

Exemple :

```
SUBSTR (MØT,I,1)='a'
```

remplace la i^{ème} lettre de la chaîne contenue dans MØT par a.

V - EXEMPLES D'UTILISATION DE L'INSTRUCTION D'AFFECTATION :

Elle est indispensable lorsque l'on veut connaître le résultat d'une expression.

Elle est très utile dans les cas suivants :

- initialisation d'indices

```
I, J, K = 1;
```

- donner un nom à une expression compliquée qui intervient à plusieurs reprises dans le programme.
- donner un nom à la valeur d'une fonction incorporée

Exemple : recherche des voyelles dans un mot.

On compare chaque lettre du mot à la liste des voyelles. La i^{eme} lettre du mot est extraite à l'aide de la fonction SUBSTR (MØT,I,1); si l'on n'affecte pas cette valeur à une variable, il faudra calculer cette valeur chaque fois que l'on change de voyelle, sinon la i^{ème} lettre du mot est transférée dans une zone identifiée et comparée à toutes les voyelles.

CHAPITRE 3

INSTRUCTIONS DE CONTROLE

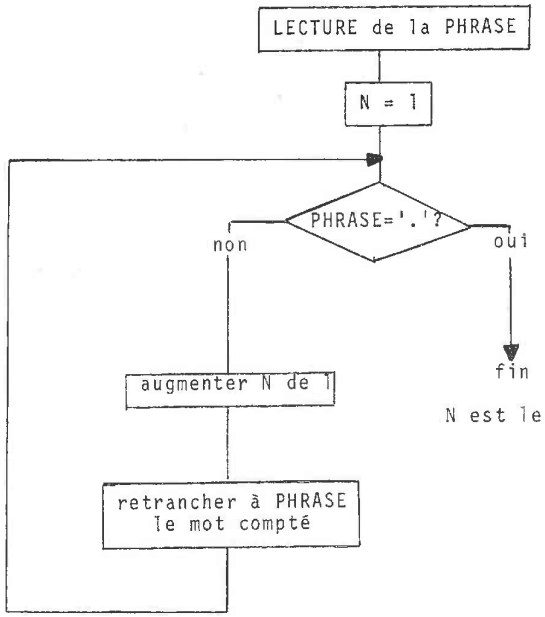
Les instructions de contrôle modifient le déroulement séquentiel du programme.

Cette modification peut être :

- impérative : saut
- conditionnelle : instruction conditionnelle
- faite un certain nombre de fois ou tant qu'une condition est réalisée : itération.

Exemple : Calcul du nombre de mots d'une phrase : N.

La phrase est terminée par un point, les mots sont séparés par des blancs



N est le nombre de mots

Cet organigramme montre que le déroulement séquentiel du programme correspondant doit être modifié.

modification conditionnelle PHRASE='.' ?

modification impérative : lorsque l'on a retranché un mot à la phrase, il est nécessaire de revenir au test PHRASE='.'? donc ce test doit être identifié, il l'est par une étiquette d'instruction.

On remarque que l'on peut faire une seule modification en utilisant la 3ème forme : faire tant que PHRASE ¬='.'.

Une étiquette d'instruction est

- soit un identificateur ou constante étiquette

```
E: I = 1;
```

- soit une variable indicée

```
E(1): I = 1;
```

E(1) appartient à un tableau d'étiquettes.

I - SAUT : L'instruction de saut a la forme suivante :

```
GØ TØ expression de désignation;
```

L'expression de désignation est un identificateur étiquette d'instruction, une variable simple ou une variable indicée, ayant pour valeur une étiquette d'instruction.

L'instruction de saut provoque le transfert du contrôle à l'instruction désignée (celle-ci ne peut être ni une déclaration, ni un format).

Elle permet : - de faire exécuter plusieurs fois une instruction ou une suite d'instructions
- d'empêcher l'exécution d'une ou plusieurs instructions en séquence.

Exemples :

1) l'expression de désignation est un identificateur

```
        ⋮
     GØ TØ A;
        ⋮
   A: X=X+1;
```

2) l'expression de désignation est une variable simple.

Soit L une variable qui prend au cours du programme les valeurs étiquettes L1 et L2. La valeur de L sera modifiée par affectation.

```
   L1: X=Y-1;
       L=L2;
       GØ TØ A;
   L2: Y=X-1;
       L=L1;
   A:  ⋮
       GØ TØ L;
```

1er passage `L=L2`

`GØ TØ L` ∫ `GØ TØ L2;`

2e passage `L=L1`

`GØ TØ L` ∫ `GØ TØ L1;`

3) L'expression de désignation est une variable indicée.

a) Soit AIG un tableau de 3 étiquettes


```
        I=SIGN(RANG-RANGMAX) + 2;
        GØ TØ AIG(I);

(1)     AIG(1):
           ⋮
        GØ TØ EXIT;
(2)     AIG(2):
           ⋮
        GØ TØ EXIT;
(3)     AIG(3):
           ⋮
        EXIT:
```

b) Soit AIG un tableau de 3 étiquettes auxquelles on a donné les valeurs L1, L2, L3.

```
        I=SIGN(RANG-RANGMAX) + 2;
        GØ TØ AIG(I);
        ⋮
(1)   L1:
        ⋮
      GØ TØ EXIT;
(2)   L2:
        ⋮
      GØ TØ EXIT;
(3)   L3:
      EXIT:
```

Suivant la valeur de I, on exécute la séquence d'instructions (1), (2) ou (3).

II - INSTRUCTION CONDITIONNELLE : Le déroulement du programme dépend de la valeur d'une expression scalaire.

Elle permet de faire exécuter une séquence d'instructions dans un cas ou une autre dans le cas contraire.

Exemple : on cherche le nombre d'occurences de a dans un mot

```
E: LETTRE=SUBSTR(MØT,i,1 );
 IF LETTRE='a' THEN NBA=NBA+1;
 I=I+1;
 GØ TØ E;
```

Si la lettre est différente de a, on exécute l'instruction suivante : augmenter I de 1.

L'instruction a la forme suivante :

IF expression scalaire THEN groupe d'instructions
[ELSE groupe d'instructions]

L'expression scalaire est calculée, convertie une chaîne de bits; si tous les bits valent 0, le contrôle est transféré au groupe d'instructions suivant ELSE ou à l'instruction suivante s'il n'y a pas de clause ELSE, sinon au groupe d'instructions suivant THEN puis à l'instruction suivante (sauf si le groupe contient une instruction de saut).

Un groupe d'instructions est :

- soit une instruction exécutable
- soit un groupe DØ d'instructions
- soit un bloc BEGIN.

Une instruction exécutable est une instruction d'affectation, une instruction de saut, une instruction conditionnelle, une instruction d'Entrée/Sortie ou une instruction vide.

Un groupe DØ d'instructions est une suite d'instructions exécutables, placée entre une instruction DØ et une instruction END et considérée comme une seule instruction dans le déroulement séquentiel du programme. Si l'instruction DØ est étiquetée, l'instruction END peut avoir la forme END étiquette; l'étiquette étant la même que celle qui identifie le groupe (elle ne peut pas être une variable indicée).

Le bloc BEGIN, moins utilisé, sera expliqué dans le chapitre : Structure de programme.

Exemples :

1) Calcul du nombre de mots d'une phrase :

PHRASE variable de chaîne de caractères contenant la phrase à étudier.

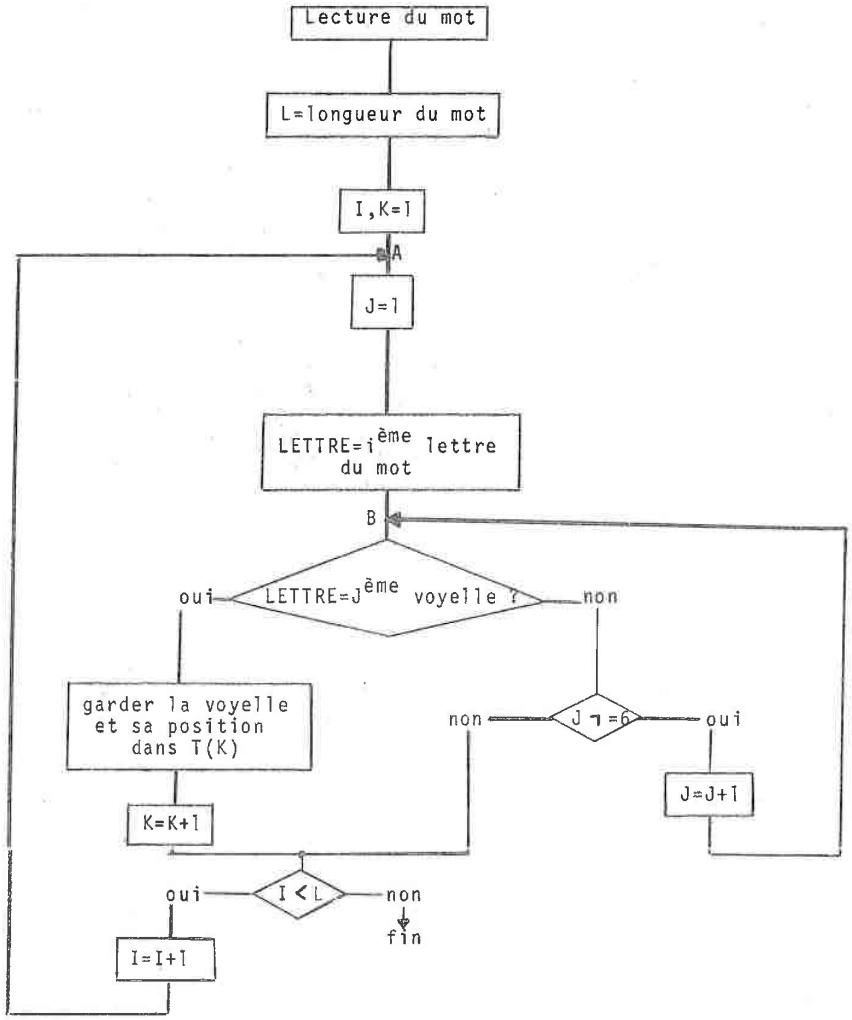
N variable de type arithmétique dont la valeur finale est le nombre de mots de la phrase.

```
 N=-1;
E: IF PHRASE='.' THEN GØ TØ FIN;
 N=N+1;
 PHRASE=SUBSTR (PHRASE, INDEX (PHRASE,' ')+1);
 GØ TØ E;
FIN:
```

2) Recherche des voyelles dans un mot.

Pour chaque voyelle on veut la voyelle suivie de sa position dans le mot.

Organigramme :



On utilise les variables suivantes :

MØT chaîne d'au plus 20 caractères.

VØY tableau de 6 éléments de 1 caractère contenant la liste des voyelles.

T tableau de 15 éléments de 3 caractères, chaque élément est formé de la suite : voyelle, position dans le mot.

LETTRE chaîne de 1 caractère.

3 indices I, J, K.

Programme :

```
      L=LENGTH (MØT);
      I,K=1;
   A: LETTRE=SUBSTR (MØT,I,1);
      J=1;
   B: IF LETTRE=VØY(J) THEN
                DØ; T(K)=LETTRE || I;
                    K=K+1;
                END;
      ELSE IF J  =6 THEN
                      DØ; J=J+1;
                          GØ TØ B;
                      END;
      IF I  L THEN DØ; I=I+1,
                      GØ TØ A;
                END;
```

III - ITERATION :

Elle permet de faire exécuter une séquence d'instructions placées entre une instruction DØ et une instruction END, un certain nombre de fois.

- soit tant qu'une condition est réalisée : itération avec condition
- soit pour des valeurs successives d'un paramétre appelé variable contrôlée : itération avec progression
- soit pour des valeurs quelconques de la variable contrôlée : itération cas général.

3.1. Itération avec condition :

```
DØ WHILE (expression scalaire);
suite d'instructions
END;
```

L'expression scalaire est calculée avant chaque exécution du groupe et convertie en une chaîne de bits; tant que la chaîne de bits n'est pas nulle, le groupe est exécuté sinon le contrôle est passé à l'instruction suivant le groupe.

Exemple : nombre de mots d'une phrase.

```
N=1;
DØ WHILE (PHRASE¬='.');
   N=N+1;
   PHRASE=SUBSTR (PHRASE, INDEX (PHRASE,' ')+1);
END;
```


3.2. Itération avec progression :

La variable contrôlée varie linéairement d'une valeur initiale à une valeur finale avec un pas donné.

DØ variable contrôlée = expression 1 TØ expression 2 BY expression 3
de type numérique
Suite d'instructions
END;

Les expressions sont des expressions scalaires à valeurs numériques

expression 1 : valeur initiale de la variable contrôlée

expression 2 : valeur limite de la variable contrôlée

expression 3 : pas de l'itération.

Si le pas de l'itération est omis, il est égal à 1.

Cette forme d'itération est utile, entre autres, pour comparer un élément aux éléments d'un tableau, pour exécuter un groupe d'instructions un nombre déterminé de fois.

Exemple : Recherche des voyelles dans un mot :

```
      L=LENGTH(MØT);
      K=1;
   A: DØ I=1 TØ L;
      LETTRE=SUBSTR(MØT,I,1);
      B: DØ J=1 TØ 6;
         IF LETTRE=VØY (J) THEN
            DØ;T(K)=LETTRE(I);
               K=K+1;
               GØ TØ C;
            END;
      END B;
C: END A;
```

L'itération A permet de décrire le mot lettre après lettre.
L'itération B permet de considérer les éléments du tableau des voyelles l'un après l'autre et de les comparer à la lettre du mot. Si la lettre est une voyelle, sa valeur et son rang sont rangés dans le tableau T et l'on passe à l'étude de la lettre suivante.

Cas dégénéré de l'itération avec progression :

```
DØ variable contrôlée=expression 1;
```

La variable contrôlée peut être de type arithmétique, chaîne ou étiquette.

Exemples :

```
DØ VØY='a';
DØ LETTRE=SUBSTR (MØT, I, 1);
```

Cette forme dégénérée de l'itération n'offre d'intérêt que combinée avec la forme avec condition ou dans la forme générale.

3.3. Itération avec progression et condition :

Cette itération est la combinaison des 2 premières formes. La valeur limite de la variable contrôlée peut être omise, auquel cas l'itération sera faite jusqu'à ce qu'il y soit mis fin par la clause WHILE ou par une instruction du groupe DØ.
L'itération commence par l'affectation de la valeur initiale à la variable contrôlée, l'examen de la condition a lieu ensuite.

Exemples :

```
1) DØ J=1 TØ 6 WHILE (LETTRE¬=VØY(J));
   J=1 lors du 1er passage sur la condition LETTRE¬=VØY(J).
2) I=1;
   DØ  LETTRE=SUBSTR(MØT, I, 1) WHILE(I<=L);
   I=I+1;
   END;
```


Il est nécessaire de donner à I une valeur avant d'entrer dans la boucle sinon il aurait une valeur indéterminée.

3.4. Itération forme générale :

```
DØ variable contrôlée = liste de spécifications;
  suite d'instructions
END;
```

Les spécifications sont séparées par des virgules. Une spécification a l'une des formes précédentes.
Cette forme d'itération permet de donner à la variable contrôlée des valeurs numériques non successives, des valeurs de chaîne de caractères ou d'étiquettes.

Exemple : Recherche des voyelles dans un mot.
La variable contrôlée VØY prend tour à tour les valeurs a, e, i, ø, u, y, ce qui évite de les ranger dans un tableau et d'aller les rechercher.

```
      I,K=1; L=LENGTH(MØT);
   A: DØ WHILE (I<=L);
      LETTRE=SUBSTR (MØT, I, 1);
      B: DØ VØY='a', 'e', 'i', 'ø', 'u', 'y';
         IF LETTRE=VØY THEN
            DØ;T(K)=VØY ‖ I;
               K=K+1;
               GØ TØ C;
            END;
      END B;
    C: I=I+1;
   END A;
```

Remarques :

1) On ne peut passer de l'extérieur d'un groupe DØ à une instruction intérieure au groupe que s'il n'y a pas itération.

2) Si l'on sort de l'itération par épuisement de la liste des spécifications, la valeur de la variable contrôlée n'est pas définie.

Exemple : Recherche des voyelles dans un mot en utilisant pour l'itération B une itération avec progression et condition.
Que l'on sorte par épuisement de la progression ou par la condition, le contrôle est transféré à l'instruction suivante, il est donc nécessaire de différencier les 2 cas, seul le 2ème correspondant à une voyelle.

```
        K=1; L=LENGTH (MØT);
    A: DØ I=1 TØ L;
       LETTRE=SUBSTR (MØT, I, 1); J1=0;
       B: DØ J=1 TØ 6 WHILE (LETTRE¬=VØY(J));
          J1=J;
       END B;
       IF J1=6 THEN GØ TØ C;
          T(K)=LETTRE‖I;
          K=K+1;
    C: END A;
```

La variable J1 permet de conserver la valeur de la variable contrôlée J.

CHAPITRE 4

DECLARATIONS

Un programme est une suite de phrases du langage séparées par;.
Une phrase ou instruction est une suite de mots-clés, caractères spéciaux et identificateurs.

Exemple :

```
A = B * C ;
```

Cette instruction d'affectation est syntaxiquement correcte, mais si A et B ont pour valeur des nombres et C une chaîne de caractères non numériques, elle n'a aucun sens.
Pour déterminer si l'instruction a un sens, il est nécessaire d'avoir des renseignements sur l'objet représenté par l'identificateur par exemple nature, type, longueur.....
Les propriétés caractérisant l'objet représenté et celles caractérisant l'identificateur lui-même sont indiquées par les attributs.
L'ensemble des attributs d'un identificateur lui est associé au moyen d'une déclaration.
L'instruction de déclaration a la forme suivante :
DECLARE identificateur liste d'attributs, identificateur liste d'attributs, ... ;
Le nombre d'identificateurs déclarés dans une même instruction est quelconque.

Exemple : DECLARE A FIXED (3), B FIXED, C CHARACTER(10);

A identifie un nombre entier de 3 chiffres

B identifie un nombre dont la grandeur n'est pas spécifiée, elle sera égale à la grandeur standard définie par le compilateur, le compilateur PL/1 de niveau F lui attribuera, par défaut,

la grandeur (5,0) qui désigne un nombre entier de 5 chiffres.

C identifie une chaîne de 10 caractères.

Pour les chaînes, le type et la longueur sont nécessaires, mais pour les identificateurs identifiant des nombres, il n'est pas obligatoire d'indiquer tous les attributs, il est prévu pour le ou les attributs manquants un attribut de remplacement correspondant à celui le plus fréquemment utilisé : c'est l'attribution par défaut.

Nous considérerons dans ce chapitre les attributs caractérisant les objets déjà étudiés : nombres, chaînes isolés ou groupés en tableaux ou structures et les étiquettes. Les attributs correspondent aux autres objets tels que fichiers, listes .... seront définis lors de l'étude des objets.

I - DECLARATION D'UNE VARIABLE SIMPLE - ATTRIBUTS DE TYPE :

Une variable simple peut prendre pour valeur un nombre, une chaîne de caractères ou de bits ou une constante étiquette.

On distingue donc 3 types de déclarations :

1.1. Variable à valeurs numériques :

```
DECLARE X FIXED(3,2), Y FLØAT(3), Z FIXED(3),P;
```

X a pour valeur un nombre en virgule fixe de 3 chiffres dont 2 après le point par exemple

1.23

Y a pour valeur un nombre en virgule flottante de 3 chiffres par exemple

.123E5 ou 0.12E5

Z a pour valeur un entier de 3 chiffres par exemple

123

P a pour valeur un nombre en virgule fixe dont la précision est, par défaut, (5,0); par exemple

00123

Pour une variable à valeurs numériques, la déclaration spécifie la représentation et la précision, ou la représentation seule auquel cas la précision est attribuée par défaut, ou rien auquel cas représentation et précision sont attribuées par défaut.

Attributs d'un nombre en virgule fixe

a) FIXED (n,m)

n nombre de chiffres, m nombre de chiffres après le point.

b) FIXED sous_entendu (5,0)

c) rien l'identificateur commence par l'une des lettres de I à N : les valeurs prises par la variable sont des

entiers compris entre - 32768 et 32768.

Attributs d'un nombre en virgule flottante :

a) FLØAT (n)

n nombre total de chiffres

b) FLØAT sous-entendu (5)

c) rien l'identificateur commence par l'une des lettres autres que les lettres de I à N.

Les valeurs prises par la variable sont des nombres en virgule flottante dont le nombre de chiffres est indiqué par défaut.

1.2. Variables à valeurs caténaires :

```
DECLARE A CHARACTER (20) VARYING,
        B BIT(1), C CHARACTER((L1+L2)/2);
```

A représente une chaîne d'au maximum 20 caractères par exemple 'ELEMENT'

B représente une chaîne de 1 bit par exemple '1'B

C représente une chaîne de caractères dont la longueur est évaluée au moment de la déclaration de C.

Pour une variable caténaire, la déclaration spécifie le type de chaîne : CHARACTER ou BIT et la longueur qui peut être fixe ou variable auquel cas on indique la longueur maximum.
La longueur est un entier, une expression ou * auquel cas la longueur doit avoir été définie précédemment.

1.3. Variables à valeurs d'étiquettes :

Les valeurs prises par la variable sont des étiquettes qui apparaissent dans le programme.


```
DECLARE L LABEL(E1,E2,E3), X LABEL
.
.
.
L=E1;
GØ TØ L;
.
.
.
E1:
.
.
.
E2:
.
.
.
E3:
.
.
.
E :
```

Pour une variable de type étiquette, la déclaration comprend l'attibut LABEL suivi ou non de la liste des valeurs prises par la variable.
Si une liste de valeurs est indiquée, la variable ne peut prendre pour valeur que celles de la liste, sinon la variable peut prendre pour valeur n'importe quelle étiquette d'instruction du programme.
Dans l'exemple précédent L peut prendre pour valeur E1, E2, E3 mais pas E alors que X peut prendre pour valeur E1, E2, E3, E .

II - DECLARATION DE TABLEAU - ATTRIBUT DE DIMENSION :

```
DECLARE T (3) CHARACTER (10);
```

Cette déclaration indique que T est un tableau unidimensionnel de 3 éléments qui sont des chaînes de 10 caractères.
(3) est l'attribut de dimension.

La déclaration d'un tableau comprend un attribut de dimension suivi de la déclaration de l'élément de tableau qui peut prendre pour valeur une valeur numérique, une valeur caténaire ou une valeur d'étiquette. L'attribut de dimension comprend une liste de paires de bornes entre parenthèses.

par exemple (2:N,3:10)

Les bornes sont précisées pour chaque dimension, toutefois la borne inférieure peut être omise si elle vaut 1. Une borne est un entier (>0 ou ≤ 0), une expression (évaluée au moment de la réservation de place en mémoire pour le tableau) ou * (cette notation est permise si la borne est connue au moment de la déclaration voir chapitre sur les procédures).

Exemple 1 : Soient 2 tableaux A et B de 10 éléments dont les valeurs sont des nombres en virgule fixe.

```
DECLARE A(10) FIXED, B(10) FIXED;
```

Exemple 2 :

```
DECLARE E(3) LABEL;
```

E tableau de 3 étiquettes E(1), E(2), E(3).

I - DECLARATION DE STRUCTURE :

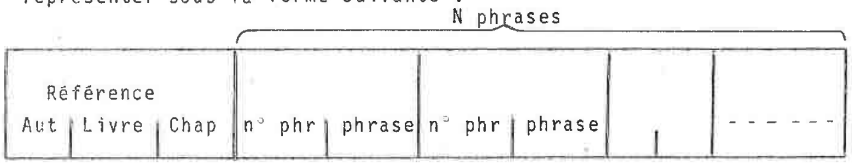
3.1. Une structure principale est composée de structures secondaires et (ou) d'éléments terminaux qui sont des tableaux d'éléments simples ou des éléments simples. Seuls les éléments terminaux prennent une valeur ; donc la déclaration de structure comprend des attributs de type et éventuellement de dimension seulement pour les éléments terminaux.

Exemple : Soit la phrase de 300 caractères enregistrée sous la forme suivante :


```
DECLARE 1 TEXTE,
          2 REF,
              3 AUTEUR CHARACTER(15),
              3 LIVRE CHARACTER(15),
              3 CHAP FIXED(2),
              3 NØPH FIXED(3),
          2 PHRASE CHARACTER(300);
```

3.2. Tableau de structures :

Considérons l'ensemble des N phrases d'un chapitre. On peut le représenter sous la forme suivante :



La référence est suivie d'un tableau de structures. Chaque structure comprend un n° et une chaîne de 300 caractères.

```
DECLARE 1 TEXTE,
          2 REFERENCE,
              3 AUTEUR CHARACTER(15),
              3 LIVRE CHARACTER(15),
              3 CHAP FIXED(2),
          2 T(N),
              3 NØ FIXED(3),
              3 PHRASE CHARACTER(300);
```

Seuls les éléments terminaux : AUTEUR, LIVRE, CHAP, NØ, PHRASE ont un attribut de type, mais l'attribut de dimension est indiqué à la suite de l'identificateur du tableau de structures.

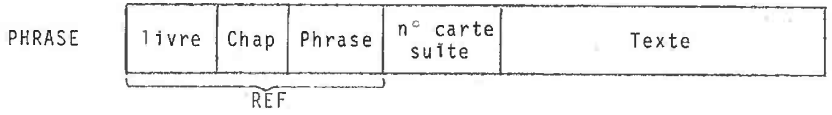
3.3. Attribut LIKE : Il a la forme suivante

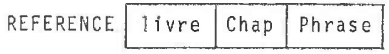
LIKE nom de structure

Il permet de déclarer une structure principale ou secondaire de même composition qu'une structure principale ou secondaire déjà déclarée, dite structure modèle, sans avoir à la décrire.

Exemple :

Soit un texte perforé sur cartes, une phrase ne tient pas toujours sur une seule carte aussi est il nécessaire de garder la référence de la 1ère carte pour la comparer à celle des cartes suivantes :





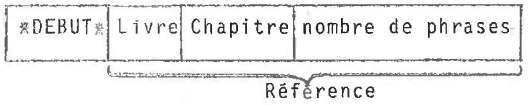

```
DECLARE 1 PHRASE,
          2 REF,
            3 LIV CHARACTER(2),
            3 CHAP FIXED(2),
            3 NØ FIXED(3),
          2 SUITE FIXED(1),
          2 TEXTE CHARACTER(72),
        1 REFERENCE LIKE PHRASE.REF;
```

PHRASE.REF Structure secondaire appartenant à la structure principale PHRASE.

La déclaration de référence équivaut à :

```
1 REFERENCE,
    2 LIV CHARACTER(2),
    2 CHAP FIXED(2),
    2 NØ FIXED(3),
```

Considérons la carte début de chapitre



Bien que le contenu du 3ème élément de référence n'ait pas le même sens, mais il a même configuration, on pourra déclarer la carte début de chapitre ainsi :

```
1 C,
  2 DEB CHARACTER(8),
  2 REFERENCE LIKE PHRASE.REF,
```

C.NØ représente le nombre de phrases dans le chapitre considéré.

Remarques : La structure modèle ne doit ni faire partie d'un tableau de structures, ni contenir l'attribut LIKE.

Les éléments terminaux et secondaires de la structure déclarée avec l'attribut LIKE ont même nom que ceux de la structure modèle, les numéros de niveau sont modifiés si besoin est.

IV - INITIALISATION :

L'attribut d'initialisation permet d'affecter une valeur à une variable simple au moment de sa déclaration, une liste de valeurs à un tableau d'éléments simples. Il a la forme suivante :

INITIAL (liste de valeurs)

Exemple :

```
DECLARE I FIXED INITIAL(0);
```

I nombre entier dont la valeur initiale est 0.

Une valeur de la liste peut être :

- une constante arithmétique
  caténaire
  étiquette.
- * indiquant l'absence d'initialisation pour la variable correspondante.
- une spécification d'itération qui a la forme suivante :

(facteur d'itération) élément itératif

L'élément itératif est soit une constante
soit *
soit (liste d'éléments)

Le facteur d'itération est soit un entier > 0
soit une expression scalaire évaluée

et convertie en un entier.

Si la valeur du facteur d'itération est négative ou nulle, il n'y a pas initialisation.

Exemple 1 : Initialisation d'une variable simple

```
DECLARE ALPHA CHARACTER(2)
INITIAL((2)'A'), BETA(2) CHARACTER(1) INITIAL
((2)(1)'A'), X FLØAT(3) INITIAL(.555E3)
```

La valeur de ALPHA est 'AA'
La valeur de BETA(1) est 'A'
La valeur de BETA(2) est 'A'
La valeur de X est .555E3

Si une seule expression scalaire précède une valeur initiale de chaîne, elle est interprétée comme un facteur de répétition pour la chaîne.

Exemple 2 : Initialisation de tableau

```
DECLARE VØY(6) CHARACTER(1) INITIAL('A','E','I','Ø',
'U','Y'), A(5,5) INITIAL((15)0, (2)((3)5,2,0));
```

Pour un tableau, l'initialisation se fait élément par élément, l'indice final variant le plus rapidement. Si les valeurs sont en excès, les valeurs restantes sont ignorées, si leur nombre est insuffisant, le tableau n'est pas entièrement initialisé.

Tableau A

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
5	5	5	2	0
5	5	5	2	0

Exemple 3 : Tableau d'étiquettes d'instruction

```
DECLARE Z(3) LABEL INITIAL(A,B,C);
:
A:
B:
C:
GØ TØ Z(I);
```

Les valeurs du tableau d'étiquettes doivent apparaître comme étiquettes d'instruction dans le programme.

Exemple 4 : Effacement d'une zone de travail

```
DECLARE PH CHARACTER(100) INITIAL((100)'');
```

V - COMPLEMENTS SUR L'INSTRUCTION DECLARE.

Tous les identificateurs utilisés dans un programme doivent être déclarés sauf les constantes étiquettes d'instruction.

5.1. Ordre des attributs :

5.1.1. Déclaration d'une variable simple :

Si la précision est indiquée, elle suit obligatoirement l'attribut de représentation.

Si la variable est une variable étiquette, les valeurs d'étiquettes, si elles sont spécifiées, suivent obligatoirement l'attribut LABEL.

5.1.2. Déclaration de tableau :

L'attribut de dimension est placé immédiatement après l'identificateur de tableau.

5.2. Mise en facteur d'attributs :

Exemple :

```
DECLARE (A,B) CHARACTER(10),
        (I INITIAL(0), J INITIAL(1)) FIXED;
```

A et B identifient des chaînes de 10 caractères,

I et J des entiers dont les valeurs initiales sont respectivement 0 et 1.

Cette instruction DECLARE est équivalente à :

```
DECLARE A CHARACTER(10), B CHARACTER(10),
I FIXED INITIAL(0), J FIXED INITIAL(1);
```

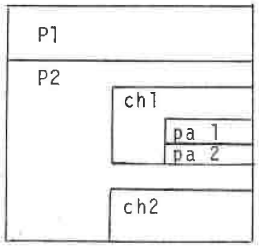

CHAPITRE 5

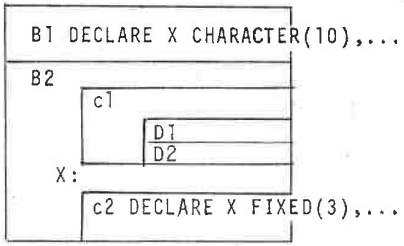
STRUCTURE DE PROGRAMME

1 - NOTION de BLOC :

De même qu'un livre un programme a une certaine organisation. Un livre peut être décomposé en une ou plusieurs parties ; chaque partie comprend des chapitres ; un chapitre peut éventuellement comporter plusieurs paragraphes. De même un programme est composé d'un ou plusieurs blocs externes ; un bloc externe comprend un début de bloc, des déclarations, des instructions, éventuellement des blocs internes et une fin de bloc ; un bloc interne comprend un début de bloc, éventuellement des déclarations et des blocs internes, des instructions et une fin de bloc.

Schéma :





Les blocs remplissent 2 fonctions importantes :

1) ils permettent un encombrement réduit de la mémoire de l'ordinateur puisque la place en mémoire est donnée aux variables seulement pendant l'exécution du bloc sauf dans le cas de variables dites externes.

2) ils définissent le domaine d'application des identificateurs

Exemple :

X représente une chaîne de caractères dans le bloc B1

X " " étiquette dans B2 et les blocs contenus dans B2 à l'exception de bloc C2 où X représente un entier de 3 chiffres.

Un bloc externe est une procédure dite externe.

Un bloc interne est soit un bloc BEGIN

soit une procédure dite interne.

Forme du bloc procédure :

nom de la procédure : instruction PROCEDURE

Corps de procédure

END nom de la procédure ;

Le nom de la procédure est un identificateur.

L'instruction PROCEDURE est soit simple, soit paramétrée.

```
P : PROCEDURE (X,Y);
    DECLARE (X,Y) FIXED(2);
     Y = X * Y;
     :
  END P;
```

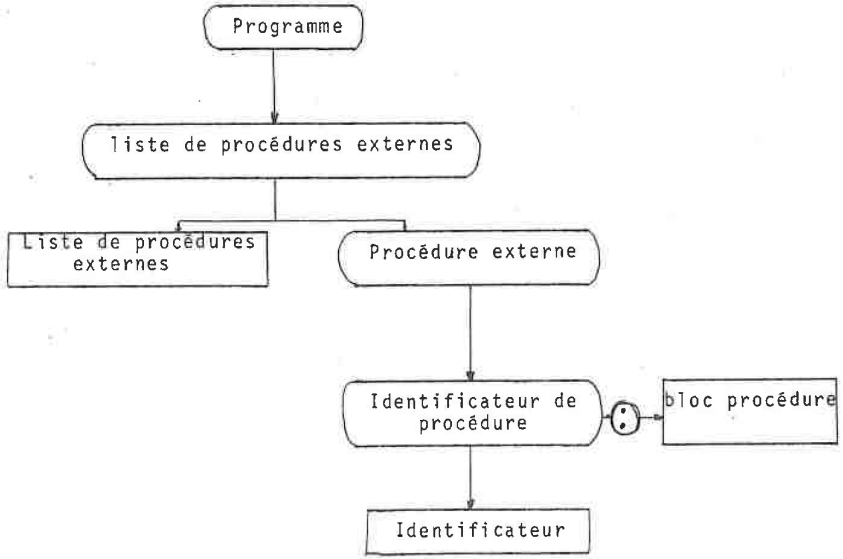
Forme du bloc BEGIN :

```
BEGIN;
  :
END;
```

L'instruction BEGIN peut être étiquetée, l'étiquette qui suit éventuellement END est soit l'étiquette du bloc, soit celle d'un bloc plus grand qui le contient .


```
A : BEGIN;
  :
  :
END A;
```

d'où la forme d'un programme PL/1

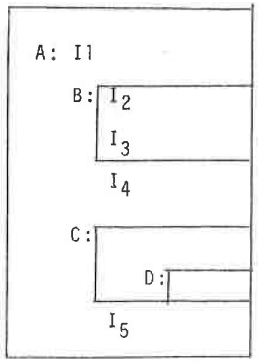


Exemple :

```
A: PRØCEDURE;
   II
  B: BEGIN;
     I2
     I3
   END B;
```

```
      I4
   C: PRØCEDURE;

      D: BEGIN;
      END;
   END C;
    I5
END A;
```

La procédure externe contenant le bloc BEGIN B et la procédure interne C contenant elle-même le bloc D



Le texte d'un bloc sauf les étiquettes ou identificateur précédant le bloc est dit être contenu dans le bloc.

La partie du texte d'un bloc A, non contenue dans un bloc interne à A est dite être interne au bloc A.

Reprenons l'exemple précédent : le texte interne au bloc A est :

```
PRØCEDURE
   I1
   B:
   I4
   C:
   I5
  END A;
```


La notion d'"interne à" est très importante dans la définition du domaine d'application des identificateurs.

Utilisation de l'instruction END - L'instruction END marque la fin d'un bloc ou d'un groupe DO.

```
END  étiquette          ;
     ou id.de procédure
```

Si aucune étiquette est indiquée, l'instruction termine le bloc ou groupe précédant l'instruction, non encore terminé.

Si une étiquette ou identificateur de procédure suit END, l'instruction termine le bloc identifié et tous les groupes ou groupes non terminés internes à ce bloc

Exemple :

```
A: PRØCEDURE;
    B: BEGIN;
      C: DØ ...
END A;
```

L'instruction END A termine les blocs A et B et le groupe C.

L'instruction END peut elle-même être étiquetée.

Exemple :

```
A: PRØCEDURE;
   F X>Y THEN GØ TØ ETI;
ETI: END A;
```

II - PORTEE DES IDENTIFICATEURS :

Tous les identificateurs utilisés dans un bloc doivent (à l'exception des étiquettes et des noms de procédure) être déclarés soit dans ce bloc soit dans un bloc plus grand. L'identificateur est connu dans un domaine bien défini du programme appelé portée de la déclaration ou portée de l'identificateur. En général la portée d'un identi-

ficateur est le bloc dans lequel il a été déclaré ou dans lequel il apparait.

Exemple :

```
A: PRØCEDURE;
   DECLARE (X,Z) FIXED(2);
  B: PRØCEDURE;
     DECLARE Y CHARACTER(6),...
     C: BEGIN;
        DECLARE (A,X) CHARACTER(1),...
     END B;
  D: PRØCEDURE;
       DECLARE X LABEL(X1,X2);
       X2:
       X1:
   END D;
END A;
```

Identificateurs	Signification	Portée
X	nombre de 2 chiffres	Bloc A sauf les blocs C et D.
Z	"	Bloc A
B	identificateur de procédure interne	"
Y	chaîne de 6 caractères	Bloc B
C	étiquette	"
A, X	chaînes de 1 caractère	Bloc C
D	identificateur de procédure interne	Bloc A
X	variable étiquette	Bloc D
X_2, X_1	étiquettes	Bloc D

Il est toutefois possible d'utiliser le même nom pour des déclarations distinctes mais concordantes du même identificateur dans des blocs différents, grâce à l'attribut de portée : EXTERNAL ; la portée est la réunion des portées de toutes les déclarations de cet identificateur avec l'attribut EXTERNAL :

Exemple :

```
A: PRØCEDURE;
   DECLARE X EXTERNAL;
   :
   B: PRØCEDURE;
      DECLARE X CHARACTER(2),
      :
      C: BEGIN;
         DECLARE X EXTERNAL;
         :
      END C;
   END B;
END A;
D: PRØCEDURE;
   DECLARE X EXTERNAL;
END D;
```

La portée de la variable externe X est le bloc A sauf le bloc B, les blocs C et D.

Remarques :

1) un identificateur n'étant connu qu'à l'intérieur de sa portée, toutes les apparitions d'un identificateur doivent se trouver dans la portée de déclaration de l'identificateur d'où la limitation du transfert de contrôle par l'instruction de saut.

Exemple :

```
A: PRØCEDURE;
    B: PRØCEDURE;
       GØ TØ E1;
       C: PRØCEDURE;
          E1:
       END C;
          E2: END B;
          E3: END A;
```

GØ TØ E1 est incorrect : E1 n'est connu que dans le bloc C et non dans le bloc B qui le contient.

Les étiquettes E2 et E3 sont connues dans le bloc B donc GØ TØ E2 ou GØ TØ E3 est correct.

2) à l'entrée dans un bloc, les éléments suivants sont connus :

- variables déclarées dans un bloc plus grand et connues.
- variables externes
- variables déclarées dans le bloc
- arguments passés au bloc
- les plus récentes générations de variables contrôlées connues dans le bloc.

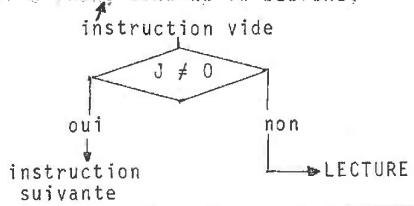
II - INSTRUCTION VIDE :

Elle n'indique aucune action. Elle peut être étiquetée.

Exemple :

1)

```
J = INDEX (TEXTE,MØT(I));
IF J THEN; ELSE GØ TØ LECTURE;
```



2)

```
IF LETTRE=VØY THEN DØ;
                   END;
               ELSE;
```

3)

```
ETI: ;
```

IV - COMMENTAIRE :

Il permet de donner des renseignements sur le travail effectué par le programme mais n'a aucune influence sur son déroulement. Il peut apparaître partout où un blanc est permis (sauf à l'intérieur d'une constante de chaîne de caractères) sous la forme suivante :

```
/* commentaire */
```

Exemple :

```
ELEM: PRØCEDURE (A,B);
 /* Recherche d'un mot dans une phrase */

 END ELEM;
```

Le commentaire ne doit pas contenir la combinaison */

Remarque :

Lors de l'exécution du programme, la 1ère procédure externe à exécuter est dite procédure principale ; c'est une procédure sans paramètres dont l'instruction PROCEDURE a la forme suivante :

```
nom: PRØCEDURE ØPTIØNS (MAIN);
```

CHAPITRE 6

PROCEDURES

I - DEFINITION ET EXEMPLES :

On appelle procédure une partie de programme qui dépend ou non des variables appelées paramètres formels et qui peut être utilisée en différents points du programme en remplaçant les paramètres formels par des expressions appelées paramètres effectifs.

Il est utile de mettre sous forme de procédure une suite d'instructions que l'on est amené à exécuter souvent.

Exemples :

1) nombre de mots d'une phrase

Le paramètre formel X représente la phrase, suite de mots et de séparateurs séparés par des blancs, terminée par un point.

```
RANG: PRØCEDURE(X);
      DECLARE X CHARACTER(*)VARYING, N FIXED(2);
      N=0;
      DØ WHILE (X¬='.');
         X=SUBSTR(X,INDEX(X,'')+1);
         N=N+1;
      END;
      RETURN (N);
END RANG;
```

Appels de la procédure :

a) I=RANG(PHRASE)-1;

b) DØ I=1 TØ RANG(A);

PHRASE et A sont des paramètres effectifs.

La procédure RANG est une fonction procédure : lors d'un appel, la procédure renvoie une valeur qui est le nombre de mots de la phrase.

2) Existence d'un mot dans une phrase

Les paramètres sont la phrase et le mot;

Le résultat est une valeur logique (chaîne de 1 bit)

vrai si le mot est dans la phrase

faux si le mot n'est pas dans la phrase.

Exemple EX:

```
PRØCEDURE (X,Y) BIT(1);
DECLARE X CHARACTER(*), Y CHARACTER(*),
      I, J;
J=0;
DØ WHILE (X¬='.');
   I=J+1;
   J=INDEX(X,'');
   IF SUBSTR(X,I,J-I)=Y THEN.
        RETURN ('1'B);
      ELSE X=SUBSTR(X,J+1);
END;
RETURN ('0'B);
END EX;
```

L'instruction procédure indique que la procédure est une fonction procédure à 2 paramètres dont le résultat est une valeur logique.

Appel de la procédure EX

On range la référence de la phrase si le mot appartient à la phrase

```
IF EX (PHRASE,MØT) THEN   ranger la référence
                   ELSE   passer à la phrase suivante.
```

3) Erreur dans l'écriture formelle d'un texte.

L'erreur peut être une erreur de voyelle, de consonne, de séparateur.

On désire imprimer un message de la forme suivante :

'erreur de''nature de l'erreur':'i'ème caractère de' référence

```
ERREUR: PRØCEDURE (X,Y,Z,T,N);
    DECLARE (X,Y) CHARACTER(*),(Z,T,N) FIXED,
    MESSAGE CHARACTER(80) VARYING;
    MESSAGE='ERREUR DE'||X||': '||N||  EME
        CARACTERE DE LA '||T||'EME PHRASE DU '
       ||Z|| 'EME CHAPITRE DE '||Y;
            PUT LIST(MESSAGE);
END ERREUR;
```

Appel de la procédure ERREUR :

```
CALL ERREUR ('CØNSØNNE',LIVRE,CHAPITRE,PHRASE,I);
```

La procédure ERREUR est un sous-programme de sortie de message, elle est appelée par une instruction CALL.

Le corps de procédure est une suite de déclarations et instructions;

On peut y trouver 3 sortes de variables :

- paramètres formels
- variables locales déclarées dans le corps de procédure
- variables déclarées dans un bloc plus grand (ceci est peu pratique car la procédure ne sera plus indépendante et son utilisation sera plus difficile).

II - <u>PARAMETRES FORMELS</u> :

Ils interviennent comme opérandes dans le corps de la procédure mais les quantités qu'ils représentent ne sont pas définies à l'intérieur de la procédure elle-même, elles le sont à chaque appel.

Dans la procédure EX, X et Y sont des chaînes de longueur non définie, la longueur aura une valeur au moment de l'appel de la procédure EX pour étudier une phrase et un mot donné.

Un paramètre formel peut être soit un identificateur de variable simple, de tableau, de structure, de fichier, de pointeur, une étiquette, un point d'entrée. Il doit être déclaré dans le corps de procédure et non dans un bloc extérieur ; la portée de l'identificateur est uniquement le bloc procédure ; les bornes d'un paramètres tableau ou la longueur d'un paramètre chaîne sont rarement connues, elles sont déclarées en utilisant la notation *.

III - <u>APPEL DE PROCEDURE</u> :

En un point quelconque d'un programme, où un identificateur de procédure est connu, on peut se référer à la procédure de la manière suivante :

nom d'entrée (liste de paramètres effectifs)

Le nom d'entrée est soit un identificateur de procédure, soit un identificateur qui définit une entrée secondaire de la procédure (voir Entrée Secondaire).

Les paramètres effectifs sont les quantités avec lesquelles la procédure est exécutée, ce sont des valeurs, des variables ou des expressions ; ils doivent être en même nombre que les paramètres formels, chaque paramètre effectif est associé au paramètre formel de même rang.

<u>Exemple</u> :

```
EX: PRØCEDURE (X,Y) BIT(1);
.............
IF EX(PHRASE,'EST') THEN .....
```

PHRASE est associée à X

'EST' est associé à Y.

Une procédure peut être utilisée comme :

- fonction
- sous-programme.

3.1. Fonction procédure :

L'utilisation en fonction est possible si la procédure calcule une valeur et une seule.

La référence de procédure apparaît comme opérande dans une expression.

Exemple : Calcul du nombre de mots d'un texte

```
LTEXTE=LTEXTE+RANG(PHRASE);
```

Lorsque l'identificateur de procédure suivi de la liste de paramètres effectifs apparaît dans une expression, la procédure est exécutée, la valeur calculée est utilisée pour évaluer l'expression.

La fonction procédure a la forme suivante :

```
Identificateur: PRØCEDURE (liste de paramètres formels)
      liste d'attributs de la valeur calculée (liste
      éventuellement vide auquel cas les attributs sont
      donnés par défaut).
      :
      RETURN (Expression);
END identificateur;
```

L'expression qui suit RETURN est de type arithmétique, caténaire, logique (ou pointeur) ; si des attributs sont spécifiés la valeur de l'expression est convertie suivant les caractéristiques indiquées.

La procédure EX est un exemple de fonction de procédure, la valeur calculée est de type logique.

Si une instruction de saut met fin à la fonction procédure, l'évaluation de l'expression dans laquelle apparaissait la fonction

procédure ne sera pas terminée et le contrôle ira à l'instruction désignée.

Exemple : Fonction dont les paramètres X, Y, Z sont des nombres.

```
A: PRØCEDURE (X,Y,Z);
      DECLARE (X,Y,Z)FIXED(3);
      IF X  =Y THEN RETURN (Z/(X-Y));
            ELSE GØ TØ ERREUR;
END A;
```

Appel de la procédure A

```
N=N-A(N1,N2,N3);
```

Si N1=N2 l'expression ne sera pas calculée et le contrôle ira à l'instruction étiquetée par ERREUR.

3.2. Sous-programme :

L'utilisation de la procédure en sous-programme permet par exemple, le renvoi de plusieurs valeurs, des opérations d'entrée-sortie.

L'appel de la procédure se fait à l'aide de l'instruction CALL.

Exemple :

```
CALL ERREUR ('SEPARATEUR',LIV,CHAP,PH,I);
```

Lorsque la procédure est appelée comme sous-programme, il y a remplacement des paramètres formels par des paramètres effectifs et transfert du contrôle à la procédure.

La procédure sous-programme a l'une des formes suivantes :

a)
```
Identificateur: PRØCEDURE (liste de paramètres formels);
      :
      RETURN;
END identificateur;
```

b)
```
Identificateur: PRØCEDURE (liste de paramètres formels);
      :
      END identificateur;
```

a . Lorsque l'instruction RETURN est rencontrée au cours de l'exécution de la procédure, le contrôle est renvoyé à la 1ère instruction exécutable suivant l'instruction d'appel.

b . L'instruction END joue le même rôle que l'instruction RETURN de a.

Si le corps de procédure contient une instruction de saut vers l'extérieur de la procédure, le contrôle est transféré à l'instruction désignée. L'étiquette peut être un paramètre de la procédure.
La procédure ERREUR est un exemple de procédure utilisée en sous-programme.

- RELATIONS ENTRE PARAMETRES FORMELS ET PARAMETRES EFFECTIFS :

Les attributs d'un paramètre effectif doivent être les mêmes que ceux du paramètre formel correspondant.
Toutefois, il peut y avoir conversion du paramètre effectif suivant les caractéristiques du paramètre formel si l'identificateur de procédure est déclaré dans la procédure appelante.
La déclaration a la forme suivante :

DECLARE Id. de procédure ENTRY (liste d'attributs de type de chaque paramètre formel).

Exemple : Appel de la procédure EX, si le 2ème paramètre effectif peut être un nombre, il faut déclarer EX dans la procédure appelante pour qu'il y ait conversion du nombre en chaîne de caractères.

```
ETUDE: PRØCEDURE;
   DECLARE EX ENTRY(CHARACTER(*),CHARACTER(*)),
          PHRASE CHARACTER(200),
          N FIXED(2);
   IF EX(PHRASE,N) THEN DØ;
                        :
                        END;
```



```
   END ETUDE;
   EX: PRØCEDURE (X,Y)BIT(1);
      DECLARE X CHARACTER(*), Y CHARACTER(*),I,J;
      :
   END EX;
```

La déclaration de EX dans la procédure ETUDE signifie que EX est un point d'entrée à 2 paramètres :

- le premier est une chaîne de caractères
- le 2ème est une chaîne de caractères.

Remarques : sur la déclaration de procédure

1) Si la liste d'attributs des paramètres n'existe pas, l'identificateur est considéré comme un identificateur de procédure (ou point d'entrée dans une procédure). Les attributs des paramètres formels et effectifs doivent être les mêmes.

   Exemple :

   DECLARE EX ENTRY;

2) Le nombre de listes d'attributs est le même que celui de paramètres formels, une liste peut être vide auquel cas le paramètre effectif de même rang doit concorder avec le paramètre formel.

   Exemple :

   DECLARE EX(, CHARACTER(*));

   Le 1er paramètre effectif de EX doit être une chaîne de caractères, le 2ème sera converti, si besoin est, en une chaîne de caractères.

3) Déclaration de paramètre tableau :

   La dimension est le premier attribut déclaré.

Exemple :

Existence des mots d'une liste dans un texte.

La procédure EXISTENCE a pour paramètres formels le texte (chaîne de caractères), la liste (tableau de chaînes de caractères)

```
EXISTENCE: PRØCEDURE (X,Y)BIT(1);
    DECLARE X CHARACTER(*), Y(*)CHARACTER(*);
    ⋮
END EXISTENCE;
APPEL: PRØCEDURE;
     DECLARE TEXTE CHARACTER(2000),
        LISTE(25) CHARACTER(20) VARYING,
        EXISTENCE ENTRY(25) CHARACTER(20));
        ⋮
END APPEL;
```

Si les bornes du tableau ne sont pas connues, on utilise la notation *.

Les bornes peuvent être une expression qui sera calculée au moment de l'entrée dans le bloc APPEL.

4) Déclaration de paramètre structure :

La composition de la structure est spécifiée en utilisant les nombres de niveau et les attributs des éléments terminaux.

Exemple :

```
P: PRØCEDURE (A,X);
   DECLARE 1 A,
             2 B FIXED(5),
             2 C,
                2 D CHARACTER(10),
```



```
                3 E FIXED(3)
             X FIXED;
        ⋮
        END P;
APPEL: PRØCEDURE;
     DECLARE (1,2 FIXED(5), 2 ,3 CHARACTER(10),
              3 FIXED(3), FIXED),.....
        ⋮
END APPEL;
```

5) La mise en facteur des attributs n'est pas permise.

4.2. Transmission des paramètres effectifs :

Lors de l'exécution de l'appel d'une procédure, le paramètre formel est remplacé par le nom du paramètre effectif correspondant et la procédure est exécutée avec les paramètres effectifs.

Si le paramètre effectif est une constante, une expression ou si ses attributs ne concordent pas avec ceux du paramètre formel, la constante, la valeur de l'expression ou la valeur du paramètre effectif convertie sera affectée à une variable fictive qui remplacera le paramètre formel lors de l'exécution de la procédure.

Exemple :

```
CALL ERREUR('VØYELLE',LIV,CHAP,PH,3*I)
```

'VØYELLE' et la valeur de l'expression 3*I seront affectés à des variables fictives (par exemple V1, V2) qui remplaceront X et N lors de l'exécution de la procédure :

```
MESSAGE='ERREUR DE '||V1||': '||V2||' EME CARACTERE
        DE LA '||PH||' EME PHRASE DU '||CHAP||' EME
        CHAPITRE DE '||LIV;
PUT LIST(MESSAGE);
```

Si le paramètre effectif est une variable indicée, les indices sont calculés avant l'appel, donc les modifications éventuelles de l'indice pendant l'exécution de la procédure n'ont aucun effet sur le paramètre correspondant.

4.3. Correspondance entre paramètres formels et paramètres effectifs.

Paramètre formel	Paramètre effectif
variable simple	variable scalaire expression scalaire
tableau	tableau expression de tableau expression scalaire si la procédure appelée est déclarée et la dimension du tableau spécifié.
Structure	structure expression de structure expression scalaire si la description de la structure est donnée pour le paramètre dans la déclaration de la procédure appelée.
étiquette variable scalaire	variable scalaire constante étiquette
étiquette variable de tableau	variable étiquette de tableau
nom d'entrée	nom d'entrée.

Un paramètre effectif correspondant à un paramètre chaîne de longueur fixe, doit être de longueur fixe, chaîne de longueur variable, doit être de longueur variable, sauf s'il y a création de variables fictives.

V - ENTREE SECONDAIRE DANS UNE PROCEDURE :

L'existence d'entrées secondaires dans une procédure permet de n'exécuter que la partie de la procédure relative au traitement désiré au lieu de sélectionner par des tests à l'intérieur de la procédure la partie désirée.

Exemple : soit une procédure de recherche des divers éléments d'une phrase : nom, verbe, article, préposition, adverbe ....

Cette procédure RECHERCHE est subdivisée en paragraphes nommés NOM, VERBE, ARTICLE, PREPOSITION, ADVERBE, ...

Si l'on désire étudier les adverbes d'une phrase, on exécutera la subdivision ADVERBE.

RECHERCHE, identificateur de la procédure est un point d'entrée particulier dit point d'entrée primaire ou principal.

Chaque subdivision sera précédée d'une instruction ENTRY dont la forme est la même que celle d'instruction PRØCEDURE.

Exemple :

```
SP1 : PRØCEDURE (X,Y,Z);
      DECLARE (X,Y,Z,A,B) FIXED;
      A=X-1;
      B=Y+1;
      GØ TØ SUITE;
SP2 : ENTRY (X,Z);
      A=X-2;
      B=X-3;

      SUITE: Z=A+B;
END SP1;
```

SP1 point d'entrée primaire dans la procédure SP1

SP2 point d'entrée secondaire dans la procédure SP1.

L'appel d'un point d'entrée secondaire se fait de la même manière qu'un point d'entrée primaire.

```
CALL  SP2 (3+L,SØMME);
```

L'instruction ENTRY ne peut être interne ni à un bloc contenu dans la procédure pour laquelle elle définit une entrée secondaire, ni à un groupe DO spécifiant une itération.

CHAPITRE 7

ENTREE - SORTIE

Les instructions d'entrée-sortie permettent les échanges d'information entre la mémoire centrale et les organes périphériques : mémoires auxiliaires ou organes d'accès.

Les informations contenues dans les mémoires auxiliaires : bande magnétique, disque, tambour magnétique .... ne sont pas accessibles directement par l'utilisateur, aussi les échanges entre utilisateur et ordinateur se font-ils, en entrée par carte perforée, ruban perforé, machine à écrire, en sortie par imprimante, perforation de ruban ou carte.

Il est prévu pour chaque ordinateur une entrée standard, en général lecteur de cartes (fichier SYSIN) et une sortie standard : imprimante (fichier SYSPRINT).

Les informations échangées sont :

- des éléments simples : chaîne de caractères ou de bits
  nombre avec ou sans signe
  tableau d'éléments simples
- des informations qui doivent être découpées en une suite d'éléments suivant un modèle, on dit alors que l'information est structurée, c'est un article. Un article n'est en général pas le seul du même modèle, par exemple la définition d'un mot du dictionnaire : à chaque mot est associé un article, le dictionnaire est représenté par une suite d'articles ou fichier.

Un article rangé en mémoire auxiliaire est appelé enregistrement, la variable en mémoire centrale qui a pour valeur un article est une structure.

I - E/S D'ELEMENTS SIMPLES (ET D'ARTICLES) SUR ORGANES D'ACCES STANDARD :

Entrée : Lecteur de cartes

Sortie : imprimante

Les instructions d'E/S d'éléments simples permettent :

- de lire les données d'un programme par exemple :
  - liste de voyelles
  - liste de codes
  - mot
- d'imprimer les résultats par exemple :
  - voyelle suivie du rang dans le mot : I-3
  - nombre de mots d'une phrase
  - .....

1.1. Lecture : on indique les variables qui prennent pour valeur les informations lues.

Exemple : DECLARE MØT CHARACTER(8)

1ère forme : entrée dirigée par liste - information sur carte

'LECTURE' 'DU' 'JØURNAL' 33 'EXEMPLAIRES'

Les valeurs sont séparées par un espace, les chaînes sont mises entre apostrophes.

```
GET LIST (MØT);
```

Cette instruction ordonne le transfert dans la variable MØT du 1er mot de la carte.

MØT | L | E | C | T | U | R | E |

2ème forme : entrée suivant un format - information sur carte

LECTUREDUJOURNAL33EXEMPLAIRES

Les données forment une chaîne continue de caractères.

```
GET EDIT (MØT) (A(7));
```

Cette instruction ordonne le transfert en mémoire centrale des 7 premiers caractères de la carte dans la variable MØT.

MØT | L | E | C | T | U | R | E |

L'exécution d'une instruction de lecture provoque la lecture d'une carte au moins, ce qui nécessite le transfert de toutes les informations utiles d'une carte en mémoire centrale, au moyen d'une seule instruction de lecture.

1.1.1. Entrée d'une liste d'éléments simples :

```
GET LIST (liste de variables);
```

Les valeurs des variables sont à la suite sur le support externe, dans l'ordre des variables séparées par des blancs, ou des virgules. Les variables sont des variables scalaires, des tableaux, des structures, avec spécification de répétition possible pour les tableaux.

Exemple : soit le tableau des lettres sur le support externe

'A' 'B' 'C' 'D' 'E' 'F' .....

```
DECLARE (T(26), CØNS(20), VØY(6)) CHARACTER(1);
GET LIST (T);
```

Cette instruction ordonne le transfert des 26 premières valeurs de la carte dans le tableau T.
Si les 26 valeurs occupent plus d'une carte, il y a lecture de la carte suivante.

```
GET LIST (VØY(1), (CØNS(I) DØ I=1 TØ 3), VØY(2),
    (CØNS(I) DØ I=4 TØ 6), VØY(3), (CØNS(I) DØ I=7 TØ 11),
    VØY(4), (CØNS(I) DØ I=12 TØ 16), VØY(5), CØNS(I) DØ I=
    17 TØ 19), VØY(6), CØNS(20));
```

Cette instruction ordonne le transfert de la 1ère valeur dans le 1er élément du tableau VØY, des 3 valeurs suivantes dans les 3 premiers éléments du tableau CØNS.....

Remarque : si l'on veut garder l'ancienne valeur d'une variable apparaissant dans une instruction de lecture, la valeur de rang correspondant sera des blancs entre 2 virgules.

Exemple :

```
'A', 12,   , 33 .........

GET LIST (X, Y, Z, T);
```

X	A		
Y	-12		
Z	valeur précédente	T	33

1.1.2. Entrée suivant un format :

```
GET EDIT (liste de variables) (liste de formats);
```

Cette instruction d'entrée permet de transférer une partie d'une information qui est sous la forme d'une file de caractères. A chaque variable correspond un format qui indique la nature de la valeur et sa longueur

exemple : A(3)-chaîne de 3 caractères

Comme dans l'entrée par liste, le transfert des valeurs se fait dans l'ordre des variables.

Les formats sont de 2 types :

- format décrivant la valeur : nature, longueur
- format de contrôle.

a) format décrivant la valeur

nature (longueur)

nature : F nombre en virgule fixe
B chaîne de bits
A chaîne de caractères
E nombre en virgule flottante

longueur :
W nombre de caractères occupés par la valeur,
W, d pour les nombres non entiers
d nombre de chiffres après le point.

b) format de contrôle X (W)

Il ne correspond à aucune variable et permet d'ignorer W caractères du support externe.

Les valeurs lues suivant le format indiqué sont transférés en mémoire centrale et converties suivant la représentation indiquée lors de la déclaration de la variable; en particulier une chaîne trop longue est tronquée, une chaîne trop courte complétée à droite par des blancs.

Exemple : Sur le support externe : mot suivi d'une liste d'éléments constitués d'une voyelle du mot suivie de son rang dans le mot

10 lettres	
ANIMAL	A-01,I-03,A-05

```
DECLARE MØT CHARACTER(12), RANG(8) FIXED(2);
GET EDIT (MØT, RANG DØ I=1 TØ 3) (A(10), 3(X(3), F(2)));
```

Cette instruction ordonne le transfert des 10 premiers caractères de la carte dans les 12 caractères de MØT et le transfert des nombres dans les 3 premiers éléments du tableau RANG, les -, espace et voyelle sont ignorés.

Remarques :

1) Un élément de format ou une liste de formats peut être précédé d'un facteur d'itération n qui est soit un entier, soit une expression scalaire entre parenthèses, si n 0 le format n'est pas utilisé.
2) La liste des variables comprend k éléments, le format j éléments.
   $j < k$ les j premières valeurs sont transférées, suivant le format indiqué, dans les j premières variables.
   La j + 1 ième est transférée, suivant le 1er élément de format, dans la j + 1 ième variable .... et ainsi jusqu'à épuisement de la liste de variables.

$j > k$ les éléments de format supplémentaires sont ignorés.

c) Instruction FØRMAT :

La liste des formats peut être remplacée par un élément de format à distance

R (étiquette)

L'étiquette repère l'instruction FØRMAT qui décrit les valeurs correspondantes.

Etiquette : FØRMAT (liste des formats);

Exemple :

Carte début de livre

00001 n°	GV	*DEBUTL*	001	020	Commentaires

Carte début de chapitre

00002	GV	*DEBUTC*	007	125	Commentaires

Les informations de ces cartes sont des valeurs de nature et longueur identiques, aussi est-il intéressant de n'écrire qu'une seule liste de formats et de s'y référer à la lecture des 2 cartes.

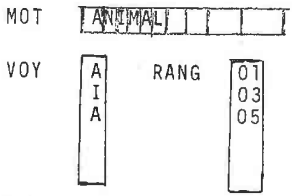
```
DECLARE NØ FIXED(2), TITRE CHAR(2), CØM CHAR(59),
(L,C) CHAR(8), (NØL, NØC, NBC, NBP) FIXED(3);

F: FØRMAT (F(5), A(2), A(8), F(3), F(3), A(59));
GET EDIT (NØ, TITRE, L, NØL, NBC, CØM) (R(F));
⋮
GET EDIT (NØ, TITRE, C, NØC, NBP, CØM) (R(F));
```

Remarque : Les instructions utilisant la liste de formats définie dans l'instruction FØRMAT et l'instruction FØRMAT doivent être dans le même bloc.

1.2. Ecriture : On indique les variables dont les valeurs doivent être transférées sur imprimante.

On distingue les 2 formes décrites pour la lecture.

Exemple : DECLARE MØT CHARACTER(10), RANG(8) FIXED(2), VØY(8) CHAR(1);



1ère forme : PUT LIST (MØT, (VØY(I), RANG(I) DØ I=1 TØ 3));

Cette instruction ordonne l'impression sur imprimante de la valeur de la variable MØT sur 10 caractères, de la valeur du 1er élément du tableau VOY sur 1 caractère, de la valeur du 1er élément du tableau de RANG sur 2 caractères, de la valeur du 2ième élément du tableau de VOY....

Les valeurs sont séparées par un blanc.

A N I M A L A 0 1 I 0 3 A 0 5

1er élément

2ème forme : PUT EDIT (MØT, (VOY(I), RANG(I) DØ
I=1 TØ 3)) (A(10), 3(X(2), A(1), F(2)));

Cette instruction ordonne le transfert des mêmes valeurs que précédemment mais chaque groupe voyelle-rang est séparé du suivant par 2 blancs grâce au format de contrôle X(2)

A N I M A L A 0 1 I 0 3 A 0 5

1.2.1. Sortie par liste :

PUT LIST (liste d'éléments);

Les éléments sont soit des variables, soit des expressions, dont la valeur calculée au moment de l'exécution de l'instruction de sortie, est inscrite sur le support externe.

Les variables sont simples, indicées ou qualifiées, avec spécification de répétition possible pour les tableaux.

Les valeurs sur le support externe sont séparées par un blanc, les apostrophes encadrant les chaînes de caractères ne sont pas imprimées.

Exemple : A = 'LANGAGES'
B = 'PROGRAMMATIØN'
C = 'LES ENTREES-SØRTIES'

PUT LIST (SUBSTR(C,1,4)||A ||' DE', B);

sur imprimante L E S L A N G A G E S D E P R O G R A M M A T I O

1er élément — 2ième élément

1.2.2. Sortie suivant un format :

PUT EDIT (liste d'éléments) (liste de formats);

Cette instruction permet de transférer sur l'organe de sortie une suite de valeurs regroupées en une file de caractères.

La liste d'éléments a même forme que celle de la sortie par liste, la liste des formats a même forme que celle de l'entrée suivant un format. Le format de contrôle X(W) permet d'insérer W espaces entre 2 valeurs.

Exemple :

```
DECLARE A CHARACTER(8), B CHARACTER(13),
        C CHARACTER(19);
PUT EDIT (C, A, 'DE '||B) (A(3), X(1), A(8), X(1), A(16));
```

LES LANGAGES DES LISTES

1er 2ième élément 3ième élément

II - E/S DE FICHIERS :

2.1. Déclaration, ouverture, fermeture d'un fichier.

Un fichier étant en général composé d'un grand nombre d'articles, il n'est pas possible de le ranger en mémoire centrale pour le traiter, aussi le range-t-on dans une mémoire auxiliaire de grande capacité et les articles sont transférés en mémoire centrale au moment de leur traitement.

Pour disposer d'une information d'un fichier, il faut réaliser les étapes suivantes :

- identifier le fichier désiré
- trouver la partie appropriée du fichier
- transférer cette partie du support externe en mémoire centrale.

Un fichier peut être organisé de 2 manières différentes

- suite continue de caractères

fichier STREAM

Exemple : fichier de bibliothèque 1 comprenant en début de fichier un tableau de 200 éléments : auteur et nombre d'ouvrages en bibliothèque, puis les fiches relatives à chaque livre classées alphabétiquement par rapport au nom de l'auteur.

ABOUT 05 ACHARD 11 ..... A B O U T ...

Une fiche comprend le nom de l'auteur, le titre de l'ouvrage, la date de parution et des renseignements.

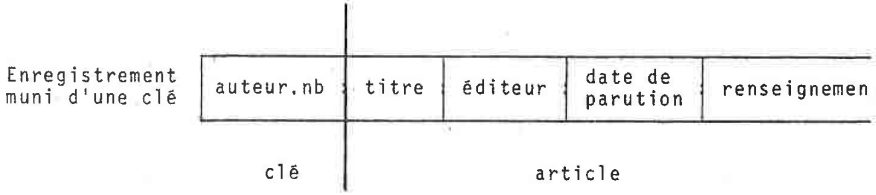
- suite d'enregistrements

fichier RECØRD

Les enregistrements du fichier peuvent être anonymes, c'est-à-dire seulement identifiables par leur contenu, ou particularisés par une clé qui permet de localiser un enregistrement parmi les autres.

Exemple : fichier de bibliothèque 2.

A chaque fiche est affectée un indicatif : clé de l'enregistrement. Cette clé est composée du nom de l'auteur suivi d'un nombre de 2 chiffres.



Les principales utilisations d'un fichier sont :

- la consultation
- la mise à jour par : modification de l'article
  insertion d'articles
  suppression d'articles
- la création.

Ces 3 utilisations impliquent respectivement que le fichier soit lu, le fichier est alors dit fichier d'entrée (INPUT), lu et réécrit le fichier est alors dit fichier de mise à jour (UPDATE), écrit le fichier est alors dit fichier de sortie (ØUTPUT).

Le fichier peut être exploité de 2 manières différentes :

- tous les articles du fichier sont traités les uns à la suite des autres, l'exploitation est dite séquentielle.

- seuls les articles possédant une certaine caractéristique sont traités, cette caractéristique doit apparaître comme clé du fichier pour permettre de localiser l'enregistrement dans le fichier, l'exploitation est dite directe.

Cette méthode d'exploitation nécessite un fichier RECØRD muni de clés, rangé sur un support externe adressable (mémoire à accès direct) : disque, tambour magnétique ....

Un fichier doit être déclaré puis ouvert avant la 1ère utilisation et fermé à la fin du traitement.

2.1.1. Déclaration : Elle est faite dans une instruction DECLARE.

DECLARE nom du fichier 1 FILE liste d'attributs , nom du fichier 2 ..... ;

Les attributs donnent des renseignements sur le fichier :

organisation - utilisation - exploitation.

Exemple : déclaration des fichiers de bibliothèque 1 et 2.

B1B1 fichier d'entrée STREAM

B1B2 fichier d'entrée RECØRD avec une clé

DECLARE B1B1 FILE STREAM INPUT, B1B2 FILE RECØRD INPUT DIRECT KEYED;

2.1.1.1. Fichier STREAM :

Il ne peut être que fichier d'entrée ou de sortie, son exploitation est séquentielle.

La liste d'attributs est
- vide : fichier d'entrée
- INPUT : "
- ØUTPUT : fichier de sortie
- ØUTUT PRINT : fichier de sortie à imprimer

STREAM peut être indiqué.

2.1.1.2. Fichier RECØRD :

Il peut être fichier d'entrée, de sortie, de mise à jour.

Fichier RECØRD non muni de clés : l'exploitation est séquentielle. La liste des attributs est :

RECORD [SEQUENTIAL] { [INPUT] / ØUTPUT / UPDATE }

Exemples :

X FILE RECØRD

X fichier d'entrée dont l'exploitation est séquentielle

Y FILE RECØRD UPDATE

Y fichier de mise à jour dont l'exploitation est séquentielle.

Fichier RECØRD muni de clés : l'exploitation est séquentielle ou directe. La liste des attributs est :

RECØRD { [SEQUENTIAL] / DIRECT } KEYED { [INPUT] / ØUTPUT / UPDATE }

Exemple :

Z FILE RECØRD DIRECT KEYED UPDATE

Z fichier RECØRD de mise à jour à exploitation directe.

2.1.2. Ouverture : Elle associe l'identificateur donné dans le programme au fichier et complète la liste des renseignements :

organisation - utilisation - exploitation.

Exemple : DECLARE B1B1 FILE, LISTING FILE STREAM;
OPEN FILE (B1B1), FILE (LISTING) PRINT;

La déclaration du fichier LISTING indique qu'il s'agit d'un fichier en continu.

L'ouverture précise que le fichier LISTING est destiné à l'impression donc fichier de sortie. L'exploitation est séquentielle d'où la liste complète des attributs : STREAM SEQUENTIAL ØUTPUT PRINT.
De plus les fichiers peuvent être étiquetés par un en-tête ou label de début de fichier. Cet en-tête est le nom propre au fichier, indépendant du programme utilisateur du fichier.
Pour un fichier d'entrée ou de mise à jour cet en-tête peut être connu grâce à l'attribut suivant IDENT (variable caténaire).
L'en-tête est transféré dans la variable caténaire :
Exemple : ØPEN FILE (BIBI) IDENT (ETIQUETTE).
Pour un fichier de sortie, on peut écrire un en-tête grâce à l'attribut : IDENT (expression).
L'expression est calculée puis convertie en une chaîne de caractères qui sera l'étiquette de début de fichier.
Exemple : ØPEN FILE LISTING IDENT ('BIBLIOTHEQUE'||DATE)
ØUTPUT PRINT;
LISTING fichier à imprimer dont l'étiquette de début de fichier est BIBLIOTHEQUE suivi de la date.

2.1.3. Fermeture : Elle dissocie du fichier l'identificateur donné au fichier dans le programme, les attributs déclarés au moment de l'ouverture, seuls restent valables ceux indiqués dans la déclaration du fichier.
Exemple : Soit un fichier muni de clés à remettre à jour; les 50 premiers articles sont à modifier, les autres articles à modifier sont répartis dans le reste du fichier, on remplacera séquentiellement les 50 premiers articles du fichier par les 50 premiers articles du fichier de modification, puis on traitera les autres directement.

Soit X le fichier à remettre à jour
Y le fichier de modification

```
DECLARE X FILE RECØRD KEYED UPDATE,
        Y FILE STREAM INPUT;
  ØPEN FILE (Y), FILE (X) SEQUENTIAL;
  DØ I=1 TØ 50;
       remplacement d'un article de X par l'article correspondant de Y
  END;
  CLØSE FILE (X);
  ØPEN FILE (X) DIRECT;
       remplacement d'un article de X de clé donnée par l'article de Y
       (ceci jusqu'à la fin du fichier de modification)
  CLØSE FILE (X), FILE (Y);
```

La fin du fichier peut être repérée par un label de fin de fichier grâce à l'attribut IDENT
En entrée ou mise à jour :
IDENT (variable caténaire)
L'étiquette de fin de fichier si elle existe est recopiée dans la variable caténaire, sinon la chaîne vide est affectée à la variable caténaire.
En sortie :
IDENT (expression)
L'expression est calculée, convertie en une chaîne de caractères et placée en fin de fichier.

2.2. E/S d'un fichier continu :
Les instructions GET et PUT permettent le transfert des informations du support externe en mémoire centrale et vice-versa; elles ont même forme que les E/S d'éléments simples, il faut seulement

préciser le fichier à traiter.

Exemple : fichier de bibliothèque BIBI

```
DECLARE BIBI FILE       STREAM INPUT, SØR FILE ØUTPUT PRINT,
    1 TABLE (200),
        2 AUTEUR CHAR(8),
        2 NB FIXED(2),
    1 ARTICLE,
        2 AUTEUR CHAR(8),
        2 TITRE CHAR(20),
        2 EDITEUR CHAR(10),
        2 DATE CHAR(4),
        2 RENS CHAR(70);
ØPEN FILE (BIBI), FILE (SØR);
GET FILE (BIBI) EDIT (TABLE, ARTICLE) (200 (A(8), F(2)), R(F));
        ⋮
        traitement du 1er article
PUT FILE (SØR) EDIT (ARTICLE) (R(F));
R: FORMAT (A(8), A(20), A(10), A(4), A(70));
        ⋮
CLØSE FILE (BIBI), FILE (SØR);
```

L'instruction de lecture provoque le transfert des 2112 premiers caractères du fichier, en mémoire centrale et les répartit suivant le format indiqué dans le tableau de structures TABLE et dans la structure ARTICLE.

L'instruction d'écriture provoque le transfert du contenu de la structure ARTICLE dans le fichier SØR et son impression.

2.3. E/S d'un fichier RECØRD :

Les instructions E/S relatives à un fichier RECØRD impliquent la transmission, sans conversion, d'un enregistrement entier et d'un seul.

Les variables dans lesquelles où à partir desquelles sont transmises les informations doivent être de niveau 1 c'est-à-dire structure principale, tableau ou variable simple; elles ne doivent pas contenir de chaînes de longueur variable.

2.3.1. Lecture d'un enregistrement d'un fichier d'entrée ou de mise à jour :

READ FILE (nom du fichier) { INTØ (variable) { [KEYTØ (variable [caténaire)] ; KEY (expression)] } ; IGNØRE (expression) }

2.3.1.1. Lecture séquentielle d'un fichier :

a) L'enregistrement transféré dans la variable est l'enregistrement suivant.

Exemple : Fichier BIB non muni de clés
BIB2 avec clés

```
DECLARE (BIB, BIB2 KEYED) FILE RECØRD INPUT SEQUENTIAL,
   1 FICHE, 2 LIVRE CHAR(20), 2 AUTEUR CHAR(10),....
   1 FICHE2, 2 TITRE CHAR(20), 2 EDITEUR CHAR(10), 2 DATE CHAR(4),
   ............
   CLE2 CHAR(12);
```

```
OPEN FILE(BIB), FILE (BIB2);
READ FILE (BIB) INTØ (FICHE);
READ FILE (BIB2) INTØ (FICHE2) KEYTØ (CLE2);
```

La 1ère instruction de lecture provoque le transfert de l'enregistrement suivant de BIB dans FICHE.
La 2ème instruction de lecture provoque le transfert de l'enregistrement suivant de BIB2 dans FICHE2 et de la clé de cet enregistrement dans CLE2.

b) Saut de n enregistrements en lecture

```
READ FILE (nom du fichier) IGNØRE (expression);
```

L'expression est calculée et convertie en un entier n.
n enregistrements du fichier séquentiel sont ignorés.

Exemple : saut de 50 enregistrements du fichier BIB2 à partir du dernier lu.

```
READ FILE (BIB2) IGNØRE (50);
```

2.3.1.2. Lecture d'un enregistrement de clé donnée

```
READ FILE (nom du fichier) INTØ (variable) KEY (expression);
```

L'enregistrement transféré dans la variable est celui dont la clé est fournie par le calcul de l'expression suivant KEY, la clé n'est pas transmise.

Exemple :

```
DECLARE BIB2 FILE RECØRD KEYED,
    1 FICHE2 ...., AUTEUR CHAR(12);
ØPEN FILE (BIB2) DIRECT;
    AUTEUR='ZOLA....01'
READ FILE (BIB2) INTØ (FICHE2) KEY (AUTEUR);
```


L'instruction de lecture provoque le transfert de l'enregistrement de clé ZOLA....01 dans FICHE2.

2.3.1.3. Lecture séquentielle d'un fichier à partir d'un enregistrement de clé donnée.

Exemple : on désire connaître les fiches relatives aux oeuvres de Zola.
On lira donc l'enregistrement de clé ZOLA 01 puis les enregistrements suivants jusqu'à ce que la partie de la clé nom de l'auteur ne soit plus ZOLA.

```
DCL BIB2 FILE RECØRD KEYED SEQUENTIAL INPUT,
AUTEUR CHAR (10), CLE CHAR(12);
ØPEN FILE (BIB2);
CLE='ZOLA 01'; AUTEUR='ZOLA';
READ FILE (BIB2) INTØ (FICHE2) KEY (CLE);
DØ WHILE (AUTEUR='ZOLA');
     PUT LIST (FICHE2);
     READ FILE (BIB2) INTØ(FICHE2) KEYTØ(CLE);
     AUTEUR=CLE;
END;
```

2.3.2. Ecriture de mise à jour à accès direct sur un fichier de sortie

```
WRITE FILE (nom du fichier) FRØM (variable)  KEYFRØM (expression)  ;
```

L'enregistrement dont la valeur est contenue dans la variable est ajouté au fichier.
L'option KEYFRØM permet d'adjoindre à l'enregistrement une clé dont la valeur est celle de l'expression après conversion en une chaîne de caractères.

Exemple : Création du fichier de sortie ZOLA à partir de BIB2

```
DCL BIB2 FILE RECØRD KEYED SEQUENTIAL INPUT,
  ZOLA FILE RECØRD ØUTPUT, AUTEUR CHAR(10),
  CLE CHAR(12);
  ØPEN FILE (BIB2), FILE (ZØLA);
  CLE='ZOLA 01'; AUTEUR='ZOLA';
  READ FILE (BIB2) INTØ (FICHE2) KEY (CLE);
  DØ WHILE (AUTEUR='ZØLA');
     WRITE FILE (ZØLA) FRØM (FICHE2);
     READ FILE (BIB2) INTØ(FICHE2) KEYTØ(CLE);
     AUTEUR=CLE;
  END;
```

2.3.3. Mise à jour d'un fichier de mise à jour par modification ou suppression d'enregistrements.

2.3.3.1. Modification d'un enregistrement :

REWRITE FILE (nom du fichier) FRØM (variable);

L'enregistrement qui vient d'être lu est remplacé par la valeur de la variable.

REWRITE FILE (nom du fichier) KEY (expression) FRØM (variable);

L'enregistrement dont la clé est donnée par l'expression est remplacé par la valeur de la variable.

2.3.3.2 Suppression d'un enregistrement de clé donnée d'un fichier de mise à jour à accès direct.

DELETE FILE (nom du fichier) KEY (expression);

Exemple : L'ouvrage de clé ZØLA 02 a disparu, la mise à jour du fichier BIB2 est faite par suppression de l'enregistrement correspondant,

```
DCL BIB2 FILE RECØRD KEYED;
ØPEN FILE (BIB2) UPDATE DIRECT;
DELETE FILE (BIB2) KEY ('ZØLA 02);
CLØSE FILE(BIB2);
```

Remarque : L'exploitation séquentielle d'un fichier de mise à jour permet de lire, modifier ou ignorer les enregistrements existants, mais le nombre d'enregistrements est invariant. L'exploitation directe permet, outre les possibilités précédentes, d'ajouter des enregistrements ou d'en supprimer.

III - EDITION D'UN FICHIER STREAM SUR IMPRIMANTE.

Le fichier est déclaré avec les attributs suivants :

STREAM ØUTPUT PRINT

Au moment de l'ouverture du fichier on peut préciser la dimension de la ligne et de la page.

Exemple : ØPEN FILE (LISTING) PAGESIZE (20) LINESIZE (100);

Le fichier LISTING sera édité à raison de 20 lignes par page et de 100 caractères par ligne.

Si ces options ne sont pas indiquées, le format standard est appliqué

La mise en page des valeurs à éditer se fait grâce à des formats de contrôle et des options de l'instruction PUT.

3.1. Formats de contrôle :

PAGE positionnement en haut de la page suivante

LINE (n) espacement vertical absolu

Il y a positionnement à la $n^{ième}$ ligne avant l'écriture des valeurs à éditer.

Si n lignes ont déjà été écrites ou si n est supérieur au nombre de lignes par page le programmeur est informé du dépassement.

SKIP [(n)] espacement vertical relatif

Il y a saut de n-1 lignes, si n n'est pas indiqué, il y a positionnement à la ligne suivante.

CØLUMN (n) espacement horizontal absolu

la valeur de l'élément suivant est écrit à partir de la $n^{ième}$ colonne. Si n caractères ont déjà été écrits sur la ligne, la valeur sera écrite à partir de la $n^{ième}$ colonne de la ligne suivante.

Si n est supérieur à la dimension de la ligne, la valeur sera écrite à partir du début de la ligne.

Exemple : Soit le fichier LISTING à imprimer, à la fin de chaque page le libellé 'suite à la page suivante' est imprimé, en début de page le numéro de page est imprimé en colonne 90.

```
DCL LISTING FILE STREAM PRINT, NØ, L;
ØPEN FILE (LISTING) PAGESIZE(30), LINESIZE(100);
:
IF L=30 THEN DØ; PUT FILE (LISTING)
EDIT ('SUITE A LA PAGE SUIVANTE',
'PAGE'  NØ) (SKIP, A(24), PAGE, CØLUMN(90), A(7));
NØ=NØ+1; L=0;END;
```


3.2. Options de l'instruction PUT :

PAGE

LINE (expression)

SKIP (expression)

L'expression est calculée et convertie en un entier n.

Ces options ont même signification que les formats de contrôle de même nom.

Exemple : Positionnement à la ligne 10 de la page suivante

```
PUT PAGE LINE(10);
```

IV - MESSAGES ENTRE L'OPERATEUR ET L'ORDINATEUR :

L'instruction DISPLAY provoque la transmission d'un message à l'opérateur et lui permet de communiquer avec le programme.

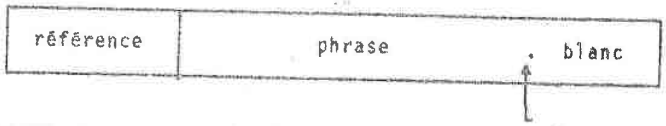
DISPLAY (exp. scalaire) REPLY (identificateur de chaîne) ;

La valeur de l'expression scalaire est convertie en un chaîne de caractères qui est le message à transmettre.

L'option REPLY permet à l'opérateur de répondre, l'exécution du programme est suspendue jusqu'à la réception de la réponse. Si cette option n'est pas spécifiée, l'exécution continue sans interruption.

Exemples :

1) Etude d'un texte qui se présente de la manière suivante :




```
IF SUBSTR(TEXTE,L,1)  ='.' THEN
  DISPLAY ('ERREUR DE PONCTUATION A LA FIN DE LA PHRASE NO'  I);
```

2) Demande de listing :

```
DISPLAY ('LISTING DEMANDE?') REPLY (REPØNSE);
IF REPØNSE='ØUI' THEN ...
```

TROISIEME PARTIE

COMPLEMENTS

CHAPITRE 1

GESTION DE LA MEMOIRE CENTRALE

La mémoire centrale de l'ordinateur est de dimension limitée, aussi essaie-t-on de l'utiliser de la manière la plus efficace possible pendant l'exécution du programme. Les variables ne sont pas toutes utilisées au même moment et elles ne le sont pas pendant toute l'exécution du programme ; il suffit que la variable ait un emplacement en mémoire au moment de son utilisation. PL/1 prévoit plusieurs modes de réservation de mémoire :

réservation statique

réservation dynamique qui est soit automatique,

soit contrôlée par le programmeur.

Le mode est indiqué lors de la déclaration de la variable par un attribut de classe de mémoire. Il ne peut être donné qu'à des variables de niveau 1 : structure principale, tableau, variable simple.

RESERVATION STATIQUE STATIC

Un emplacement en mémoire est réservé à la variable pour toute l'exécution du programme. Si c'est un tableau, les bornes sont des constantes. Si c'est une chaîne, sa longueur est une constante.

Exemple :

```
A: PRØC OPTIØNS (MAIN);
   .
   .
   .
   CALL P(X, Y);
   .
   .
   .
```

```
END A;
P: PRØC (I,J);
   DCL N STATIC INITIAL (0), I ,,,,,;
   N=N+1;
   :
END P;
```

Ce programme se compose de 2 procédures externes : la procédure principale A et la procédure P.

N variable statique permet de compter le nombre d'appels de la procédure P lors de l'exécution du programme.

II - RESERVATION DYNAMIQUE AUTOMATIQUE AUTØMATIC

Un emplacement en mémoire est réservé à la variable au moment de l'entrée dans le bloc où figure la déclaration de la variable, puis libéré à la sortie du bloc.

Exemple :

```
A: PRØC OPTIØNS (MAIN);
  :
  CALL P(X, Y);
  :
  CALL P(B, C);
  :
END A;
P: PRØC (I, J);
  DCL N STATIC INITIAL(0),I,J,L AUTOMATIC
  INITIAL(0) ,,,,;
  :
END P;
```

L variable à laquelle est alloué un emplacement en mémoire à chaque entrée dans la procédure P, sa valeur à chaque entrée dans P sera 0.

I - RESERVATION DYNAMIQUE CONTROLEE CØNTRØLLED

Le programmeur spécifie lui-même a quel moment il désire une place en mémoire pour la variable et à quel moment il la libère.

La variable est déclarée avec l'attribut de classe CØNTRØLLED.

L'allocation de mémoire se fait à l'aide de l'instruction ALLØCATE, la libération à l'aide de l'instruction FREE.

Exemple :

```
A: PRØC ØPTIØNS (MAIN);
   DCL C CHAR(10)CØNTRØLLED,,,,,, ;
   ⋮
E1: ALLØCATE C;
   ⋮
    C='ALLØCATIØN';
   ⋮
E2: FREE C;
   ⋮
END A;
```

La variable C n'a d'existence en mémoire centrale qu'entre l'exécution des instructions étiquetées E1 et E2.

3.1. Instruction ALLØCATE :

ALLØCATE liste de quantités;

Les quantités sont séparées par des virgules et ont la forme suivante

[niveau] identificateur [dimension] [{BIT / CHARACTER}] [INITIAL (valeur initiale)]

L'identificateur est un identificateur d'élément de structure (structure secondaire ou élément terminal), la structure principale correspondante doit être déclarée avec l'attribut CØNTRØLLED.

La dimension doit indiquer le même nombre de dimensions que celui indiqué dans la déclaration.

Les bornes de tableaux et longueur de chaine sont fixées au moment de l'exécution de l'instruction ALLØCATE ; si elles sont explicitement spécifiées, elles remplacent toute déclaration antérieure ; si la borne ou la longueur est *, sa valeur est celle de la plus récente génération, si elle n'est pas spécifiée, elle doit l'être dans la déclaration.

3.2. Instruction FREE

FREE liste d'identificateurs ;

L'identificateur est un identificateur de - variable simple
- tableau
- structure principale.

3.3. Exemples :

1) allocation d'un tableau

```
P: PRØC;                                  Bornes de A
     DECLARE A (N1,N2) CØNTRØLLED;
     N1,N2=10;
     ALLOCATE A;                          } A (10,10)
     ⋮
     FREE A;
```

```
   ALLØCATE A (K1,K2);                    ┐
   :                                      │ A (K1,K2)
   FREE A;                                ┘
   N1=N1+1;
   ALLØCATE A;                            ┐
   :                                      │ A (11,10)
   FREE A;                                ┘
   ALLØCATE A (*,3);                      ┐
   :                                      │ A (11,3)
   FREE A;                                ┘
END P;
```

2) Allocation de chaîne :

```
DECLARE B CHARACTER (*) VARYING CØNTRØLLED;
ALLØCATE B; /* allocation incorrecte puisque la longueur de B
               n'est pas définie */
ALLØCATE B CHARACTER (N); /* B chaîne d'au plus N caractères */
ALLØCATE B CHARACTER (2) INITIAL ('XY');
            /* B chaîne d'au plus 2 caractères dont la valeur
               est XY */
```

La réservation dynamique contrôlée est intéressante pour des tableaux et des chaînes de caractères qui tiennent beaucoup de place en mémoire centrale. Elle permet de créer ou supprimer de l'information à tout instant.

- ATTRIBUTION PAR DEFAUT D'UNE CLASSE DE MEMOIRE :

Portée	Classe de mémoire attribuée
EXTERNAL	STATIC
non spécifiée	AUTOMATIC

Les attributs EXTERNAL et AUTØMATIC sont incompatibles puisqu'une variable externe a pour portée tous les blocs où elle est déclarée comme variable externe.

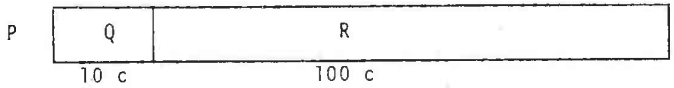
COMPLEMENT SUR LES DECLARATIONS : REDEFINITION D'UNE ZONE DE MEMOIRE

PL/1 permet de définir une variable scalaire, un tableau ou une structure occupant la même zone de mémoire qu'une autre variable (tableau ou structure) appelée base. Cette possibilité offre de l'intérêt lorsqu'une zone de mémoire contient une information qui n'est pas toujours formée des mêmes éléments (définition par correspondance) ou lorsque l'on ne veut traiter qu'une partie de l'information contenue dans la zone de mémoire (définition par recouvrement).

La redéfinition de la zone se fait dans la déclaration sous la forme suivante :

élément défini DEFINED base

Exemple : zone de mémoire appelée P se composant de 2 éléments Q et R




```
DECLARE 1 P,
          2 Q CHARACTER(10)
          2 R CHARACTER(100),
        A CHARACTER(110) DEFINED P;
```

A correspond à la chaîne des caractères de P.
On peut considérer que cette zone de mémoire contient soit un élément simple A, soit une structure P de 2 éléments Q et R.

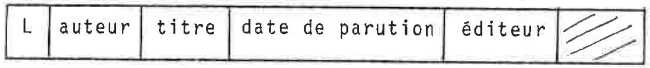
La base doit être connue du bloc dans lequel l'élément défini est déclaré, elle ne peut être elle-même un élément défini.
Un élément défini ne peut avoir ni valeur initiale, ni classe de mémoire, ni les attributs EXTERNAL, VARYING. Sa taille est au plus égale à celle de la base.
Un élément défini sur une base composée de caractères ne peut contenir des éléments déclarés numériques.

- DEFINITION PAR CORRESPONDANCE :

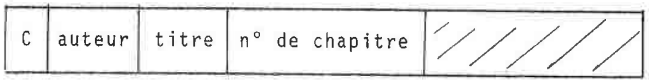
Elle permet d'utiliser une même zone de mémoire découpée différemment, à des moments différents.

Exemple : soit un fichier sur bande comprenant 3 types d'enregistrement :

1er type : enregistrement début livre



2e type : enregistrement début de chapitre



3e type : enregistrement texte n° de séquence des enregistrements texte.

Lors d'une lecture, l'enregistrement suivant est transféré en mémoire centrale dans la zone de mémoire occupée par l'enregistrement précédent. Suivant le code L, C ou n°, on déduit le type donc le découpage de la zone.


```
DECLARE 1 ENR,
          2 CØDE CHAR (1),
          2 TEXTE CHAR(199),
        1 ENRL DEFINED ENR,
          2 CØDE CHAR (1),
          2 AUTEUR CHAR (20),
          2 TITRE CHAR (30),
          2 DATE CHAR (4),
          2 EDITEUR CHAR (20),
        1 ENRC DEFINED ENR,
          2 CØDE CHAR (1),
          2 AUTEUR CHAR (20),
          2 TITRE CHAR (30),
          2 NØ CHAR(2);
LEC: READ FILE (X) INTØ (ENR);
     IF ENR.CØDE=L THEN GØ TØ LIV; /* traitement de l'enregis-
                                      trement début livre */
     IF ENR.CØDE=C THEN GØ TØ CHAP; /* traitement de l'enregis-
                                       trement début chapitre */
  /* traitement du texte */
     :
     GØ TØ LEC;
  LIV:
     :
     GØ TØ LEC;

  CHAP:
     :
     GØ TØ LEC;

  FIN;
```

Remarque : DATE et NØ sont définis comme chaînes de caractères puisque la base ne contient que des caractères.

II - DEFINITION PAR RECOUVREMENT :

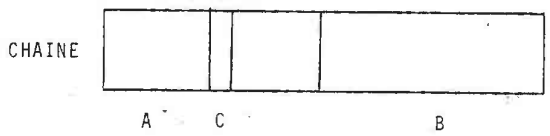
Elle permet de donner un nom à une partie de la base, repérée par sa position par rapport au début de la zone.

élément défini DEFINED base PØSITIØN (entier)

si PØSITIØN n'est pas spécifié, PØSITIØN (1) est sous-entendu.

Exemple :

```
DECLARE CHAINE CHAR (40),
        A CHAR (10) DEFINED CHAINE,
        B CHAR (20) DEFINED CHAINE PØSITIØN (21),
        C CHAR (1) DEFINED CHAINE PØSITIØN (11);
```



A est constitué des 10 premiers caractères de CHAINE

C " du 11ème caractère de CHAINE

B " des 20 derniers caractères de CHAINE.

C='K'; est équivalent à SUBSTR (CHAINE,11,1)='K';

La définition par recouvrement permet de ne pas avoir recours à la fonction SUBSTR (outil puissant mais assez lent) si l'on connait les parties de la chaine utilisées.

En général élément défini et base sont des éléments simples.

CHAPITRE 3

TRAITEMENT DES INTERRUPTIONS

Au cours du déroulement d'un programme, certains évènements peuvent entraîner une interruption de l'exécution du programme, par exemple tentative de lecture au delà de la fin du fichier, division par zéro ,...,

Le traitement des interruptions permet d'imposer les décisions à prendre quand un tel évènement se produit. Deux choses sont à spécifier : l'évènement (condition) et la conduite à adopter (action) ; elles sont précisées dans une instruction ØN ; de plus on peut préciser pour certaines parties du programme si tel évènement provoquera ou non une interruption grâce aux préfixes de condition.

I - DIFFERENTES CONDITIONS :

1.1. Conditions de calcul :

Elles interviennent dans la manipulation des données, le calcul des expressions.

CØNVERSIØN : conversion illégale sur une chaîne de caractères soit interne, soit en entrée-sortie.
Exemple : si Z='LIVRE' et N est un nombre, N=N+Z; fera apparaître une erreur de conversion.

SIZE : la taille déclarée pour l'élément n'est pas suffisante, elle apparait lors de l'affectation d'une valeur à une variable.

Exemple : DECLARE Z CHAR(8), N FIXED(2);

Z='LIVRE';

N=100;

Les 2 instructions font apparaître une erreur de dimension.

ZERØDIVIDE : tentative de division par zéro.

Pour ces conditions de calcul, l'action standard du système consiste à générer la condition ERRØR et à sortir un message. Sur la console de l'opérateur.

1,2. Conditions d'entrée-sortie :

Le traitement de l'interruption est relatif à un fichier donné spécifié après la condition. Les causes d'interruptions pouvant apparaître lors d'une entrée-sortie ne pourront être masquées par programme.

ENDFILE : tentative de lecture sur un fichier séquentiel au delà du délimiteur de fichier ; cette condition peut apparaître lors de l'exécution de GET ou de READ.

KEY : cette condition peut apparaître pour un fichier muni de clés, lors de la lecture d'un enregistrement de clé inexistante dans le fichier, ou lors de l'écriture d'un enregistrement déjà existant.

ENDPAGE : apparaît pour un fichier STREAM PRINT s'il y a tentative d'écriture d'une nouvelle ligne au delà de la limite spécifiée pour la page. Action standard : saut de la page.

TRANSMIT: Erreur de transmission entre la mémoire et le support externe,

RECØRD : apparaît lors de l'exécution d'une instruction READ ou REWRITE si la taille de la variable diffère de la taille de l'enregistrement.

Pour toutes ces conditions sauf ENDPAGE, l'action standard est la génération de la condition ERRØR et la sortie d'un message.

1.3. Conditions de mise au point :

Elles sont utilisées pour faciliter la mise au point des programmes.

SUBSCRIPTRANGE : cette condition se présente quand la valeur d'un indice n'appartient pas au domaine de variation déclaré

Exemple :

```
N=10
DECLARE T (N);
DØ I=1 TØ 20;
   IF T(I)> 0 THEN ...;
END;
```

Action standard : génération de la condition ERRØR et sortie d'un message.

CHECK (liste d'identificateurs)

Les identificateurs sont :

- étiquette d'instruction exécutable
- identificateur de variable scalaire, de tableau, de structure, de variable étiquette.
- étiquette de point d'entrée.

Cette condition n'a aucun effet sur le déroulement du programme, mais permet de suivre son exécution. Chaque fois qu'il y a modification de la valeur d'un identificateur ou exécution de l'instructio dont il est l'étiquette, l'identificateur est imprimé avec sa nouvelle valeur.

Exemple : DECLARE 1 P, 2 Q, 2 R, 2 S;

CHECK(P) la validation de cette condition entraîne l'impression

des éléments terminaux de P Q = ,.....

R = .,,,,,,,,

S = ,,,,,,,,.

1.4. Conditions du système :

FINISH : apparait immédiatement avant l'exécution d'une instruction STØP, RETURN, END ou EXIT, marquant la fin d'une procédure principale.
L'action standard met fin à l'exécution du programme.

ERRØR : apparait lorsque l'exécution du programme est obligée de s'arrêter à cause d'une erreur.

- INSTRUCTION ØN

ØN condition [SNAP] action

L'action est soit une instruction exécutable non étiquetée,
soit un bloc BEGIN non étiqueté,
soit SYSTEM auquel cas l'action standard est exécutée.

Après exécution de l'action, il y a transfert du contrôle soit au point d'interruption, soit à un autre point du programme si l'action contient une instruction de saut. L'option SNAP provoque, de plus, l'écriture du contenu de tous les registres.

Portée de l'instruction ØN : L'instruction ØN a comme domaine d'application le bloc dans lequel elle est déclarée et tous les blocs qui en dépendent (blocs internes ou procédures appelées) sauf si une autre action est spécifiée pour la même condition :

Exemple :

```
A: PRØCEDURE
   ØN1: ØN CØNVERSIØN (FILEX) CØNVQ=9999;
        .
        .
        .
   L1: GET FILE (FILEX) EDIT (A, B) (2 F(2));

   ØN2: ØN CØNVERSIØN (FILEX) SNAP GØ TØ FIN;

   L2: GET FILE (FILEX) EDIT (PHRASE)(A(300));

      FIN:
   END A;
```

Portée de ØN1

Portée de ØN2

Si un caractère illégal est lu sur FILEX pendant l'exécution de L1, l'instruction CØNVQ=9999 est exécutée.
S'il y a erreur de conversion au moment de la lecture de FILEX après l'instruction étiquetée ØN2, le contrôle est transféré à l'instruction étiquetée FIN et il y a écriture de tous les registres.

III - OPERATIONS D'INTERRUPTIONS :

3.1. Validité des conditions :

Une condition apparaissant pendant l'exécution provoque une interruption si et seulement si elle est validée à ce moment.
Les conditions d'entrée-sortie et du système sont toujours validées.
Les conditions SIZE, SUBSCRIPTRANGE et CHECK ne sont validées que si le programmeur l'a prévu explicitement. Les conditions de calcul sauf SIZE peuvent être masquées.

3.2. Préfixe de condition : c'est une condition ou la négation d'une condition placée entre parenthèses devant une instruction exécutable, un groupe DØ ou un bloc. Il valide la condition.

Exemple :

```
(SIZE): A: PRØCEDURE;
          ⋮
        ØN SIZE BEGIN;
             ⋮
           END;
      END A;
```

La condition SIZE étant validée pour la procédure A, il y aura interruption si elle se présente au cours de l'exécution de la procédure A et des procédures appelées par A.

Les divers préfixes sont : CØNVERSIØN, ZERØDIVIDE, SIZE, CHECK, SUBSCRIPTRANGE.

Les préfixes peuvent être précédés de NØ, auquel cas il y a annulation de la validité de la condition.

Portée du préfixe de condition :

- s'il est placé devant un bloc, sa portée est ce bloc et les blocs internes à l'exception des blocs, groupes et instructions pour lesquels la condition est respécifiée ; elle n'inclut pas les procédures appelées,
- s'il est placé devant un groupe ou une instruction exécutable autre que IF ou ØN, sa portée est le groupe ou l'instruction à l'exclusion des procédures appelées.
  S'il est placé devant une instruction IF ou ØN, sa portée ne comprend pas les blocs ou groupes qui font partie de l'instruction.

3.3. Exemples :

3.3.1. Condition provoquant normalement une interruption

```
(NØZERØDIVIDE): A: PRØDEDURE;
                     ⋮
                   B: ØN ZERØDIVIDE SYSTEM;
                     ⋮
                   C: ØN ZERØDIVIDE SYSTEM;
                     ⋮
                      Z=X/(N1-N2);
                   END A;
```

Interruption masquée de A à B grâce au préfixe action standard du système de B à C s'il y a une tentative de division par zéro. Affectation à Z de la valeur 0 s'il y a une interruption dûe à une tentative de division par zéro après l'étiquette C.

3.3.2. Condition normalement ignorée

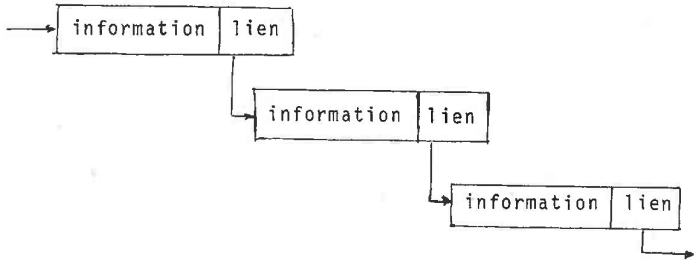
```
CHECK (E, L): A: PRØCEDURE;
                 DECLARE CARTE CHAR(80) VAR;
N=0;
ØN CHECK (E)N=N+1;
ØN CHECK (L)SYSTEM;
E: GET LIST (CARTE);
L=LENGTH(CARTE);
⋮
GØ TØ E;
PUT LIST (N);
END A;
```

A chaque lecture de carte, le compteur N est incrémenté et la longueur de la variable CARTE est imprimée par exemple L=75;

CHAPITRE 4

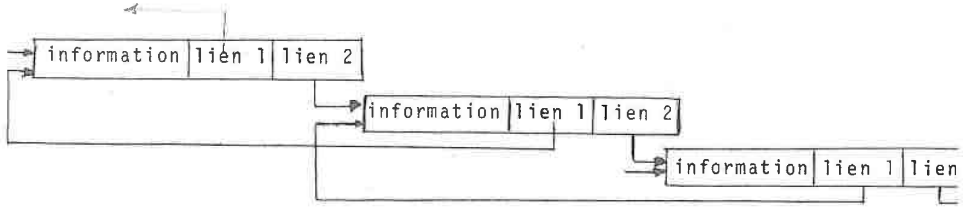
UTILISATION ET TRAITEMENT DES LISTES

Une liste est un ensemble d'informations placées en mémoire centrale et liées par un ou plusieurs liens qui permettent de passer d'un élément à un autre. Le cas le plus simple est celui où le lien est unique, on représente schématiquement la liste ainsi :



Le lien est l'adresse de la zone de mémoire occupée par l'élément suivant.

Dans de nombreuses applications, il est utile de pouvoir identifier outre le successeur, le prédécesseur.



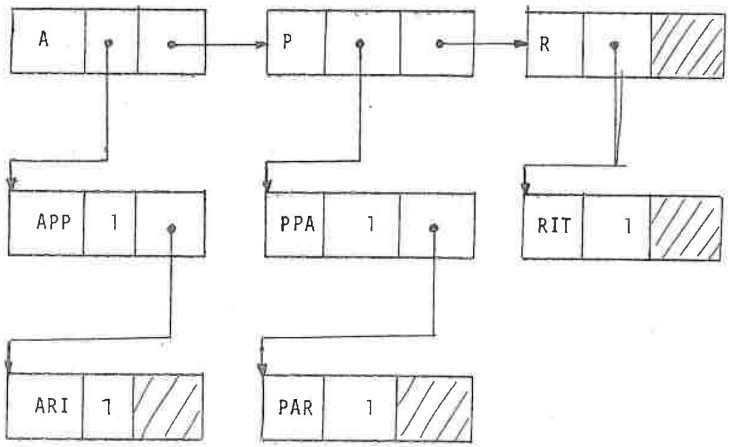
I - UTILISATION DES LISTES :

La technique de liste permet :

- la création des éléments au fur et à mesure des besoins.

Exemple : on veut connaître tous les groupes de 3 lettres apparaissant dan

un texte et leur fréquence ; l'on pourrait prévoir un tableau dont le nombre d'éléments serait le nombre de possibilités de grouper 27 caractères (26 lettres + le blanc) soit 27^3, or il est évident que toutes ces possibilités ne sont pas présentes dans une langue, donc il vaut mieux créer les éléments au fur et à mesure de leur apparition, une 1ère liste INITIALE aiguillerait vers une 2ème liste comprenant les divers groupes de 3 lettres déjà trouvés,

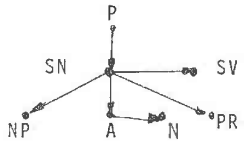


Cet exemple correspond au début du mot apparition.

- la représentation de règles de grammaire.

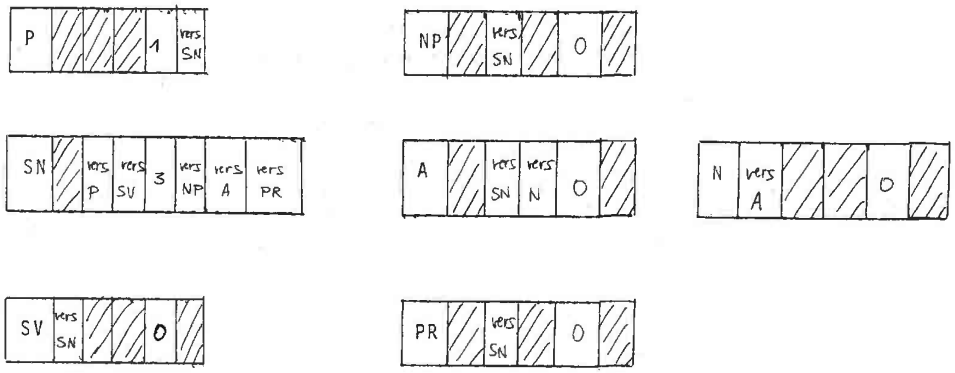
Un langage étant défini par un ensemble de règles de grammaire, on peut ainsi vérifier si une phrase appartient au langage.

Exemple : Reprenons la structure de phrase donnée dans la 1ère partie.



Cette règle de grammaire est représentée par une liste dont chaque élément correspondant à un noeud du graphe a la forme suivante :

noeud	lien vers élément gauche	lien vers prédécesseur	lien vers successeur	n	tableau des liens vers n descendants 1 1 1



- TRAITEMENT DES LISTES :

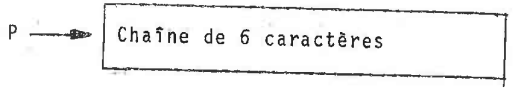
La variable qui permet de se refèrer à un élément de liste (appelé en PL/1 variable basée) n'a d'existence en mémoire que lorsque qu'une adresse lui a été attribuée. Cette adresse lui est attribuée soit par une réservation explicite, soit par un pointeur (un lien étant un pointeur particulier).

2.1. Variable basée :

Une variable basée est une variable scalaire, un tableau ou une structure.

Déclaration : elle comprend un attribut suivi de l'attribut BASED (pointeur)

Exemple :

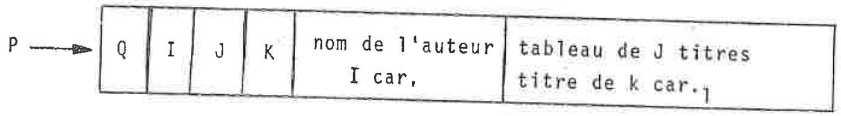


DECLARE CHAINE CHAR (6) BASED (P);

Les dimensions de tableau ou longueur de chaîne déclarées avec la variable basée sont évaluées à chaque référence à la variable basée donc l'attribut VARYING et la notation * sont interdits.

Une structure basée peut contenir des tableaux et des chaînes dont la dimension ou la longueur est un élément de la structure.

Exemple : DECLARE 1 TABLE BASED (P),
2 Q PØINTER,
2 I,
2 J,
2 K,
2 AUTEUR CHAR(I),
2 TITRES(J) CHAR(K);



Toutefois le compilateur de niveau F impose des restrictions à cette possibilité : une structure basée ne peut avoir qu'un seul élément déclaré ainsi (soit une chaîne, soit un tableau à 1 seule dimension variable) et cet élément est le dernier de la structure.

La variable longueur ou dimension prend une valeur au moment de la réservation de mémoire pour la structure, cette valeur lui est fournie par une variable externe à la structure grâce à l'attribut REFER.

Exemple : DECLARE L, 1 TABLE BASED (P),
2 Q PØINTER,
2 I,
2 INF CHAR(L REFER (I));
L=80;
ALLØCATE TABLE;
/* INF à 80 caractères, I=80 */

Lorsque l'élément a été créé, la variable externe à la structure est inutilisée.

2.2. Pointeur : Un pointeur est une variable qui ne peut prendre comme valeur qu'une adresse, cette valeur ne peut pas être fixée par le programmeur.

- déclaration : 1) déclaration explicite
DECLARE P PØINTER;

Un pointeur peut avoir les attributs suivants :
- dimension
- classe de mémoire
- initialisation (non accepté par le compilateur actuel)

2) déclaration par le contexte

Une variable est considérée comme un pointeur lorsqu'elle apparait dan une déclaration de variable basée à la suite de BASED ou dans un ordre ALLOCATE.

- Attribution d'une valeur à un pointeur
  - par affectation
  - lors d'une réservation explicite de mémoire pour la variable basée associée
- Opérations sur les pointeurs : les seules opérations permises sont les tests d'égalité ou d'inégalité

- Fonctions incorporées donnant une valeur à un pointeur

NULL La valeur de cette fonction n'identifie aucun élément. Elle sert à initialiser un pointeur ou sa valeur est le lien du dernier élément d'une liste,

ADDR(X) X est la variable à identifier.
Si X est une variable basée, la valeur est celle du pointeur associé à X sinon sa valeur est l'adresse de X.

Exemple : DECLARE A CHAR(10) BASED (P),
B CHAR(10);
P=ADDR (B);

à P est affectée la valeur de l'adresse de B donc il y a recouvrement de B par A.

2.3. Réservation de mémoire pour une variable basée :

- Réservation explicite par une instruction ALLØCATE

ALLØCATE variable basée SET (pointeur) ;

Il y a allocation d'un emplacement en mémoire à la variable basée et affectation de l'adresse de l'emplacement au pointeur associé à la variable basée si l'option SET n'est pas spécifiée, sinon au pointeur indiqué dans l'option SET.

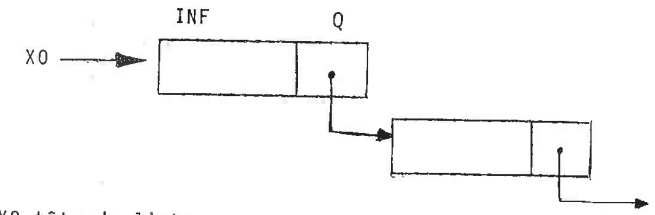
Exemple : DECLARE 1 LISTE BASED (P),
2 INF CHAR(20),
2 Q PØINTER,
X PØINTER;
ALLØCATE LISTE;
/* équivalent à ALLØCATE LISTE SET (P) */
ALLØCATE LISTE SET (X);
/* emplacement réservé à LISTE et adresse transférée dans X */

- Réservation par recouvrement d'une autre variable :
on affecte au pointeur associé à la variable basée l'adresse d'un élément de même composition (voir ADDR).

2.4, Recherche d'un élément de liste :

Une liste étant créée, un élément de la liste est rendu disponible en le qualifiant par son adresse,

Exemple :



X0 tête de liste

le 1er élément a pour adresse X0

X0 ⟶ INF identifie l'information contenue dans le 1er élément de la liste

INF est dit qualifié par X0

P=X0 ⟶ Q; affectation à P du lien contenu dans le 1er élément donc de l'adresse du 2ème élément

P ⟶ INF identifie l'information contenue dans le 2ème élément de la liste.

2.5. Libération d'un emplacement alloué à un élément de liste :

FREE [pointeur ⟶] variable basée;

Il y a libération de l'emplacement mémoire identifié par le pointeur spécifié ou par le pointeur associé à la variable basée si aucun pointeur n'est indiqué.

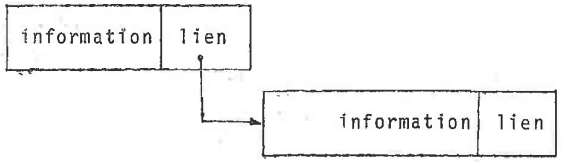
Exemple :

FREE X0 ⟶ LISTE, LISTE;

Il y a libération de l'emplacement mémoire donc suppression des éléments de liste identifiés par X0 et P.

III - EXEMPLES :

1. Création d'une liste unidirectionnelle :




```
LISTE: PROCEDURE (PE);
       DECLARE 1 ELEMENT BASED(PE),
                 2 P POINTER,
                 2 MOT CHAR(10),
       (XO,Q) POINTER STATIC EXTERNAL INITIAL (NULL);
       P=NULL;
       IF Q=NULL THEN XO,Q=PE; /* 1er élément de la liste */
                 ELSE Q ----> P, Q=PE; /* élément suivant */
END LISTE;
```

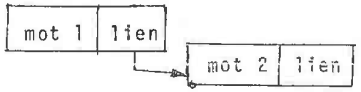
Considérons un appel de la procédure

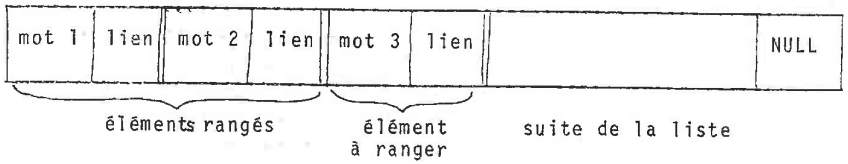
```
       DECLARE 1 X STATIC
                 2 Y POINTER,
                 2 Z CHAR(10);
       CALL LISTE (ADDR(X));
```

ELEMENT recouvre X

2. Classement alphabétique des mots d'une liste :

Chaque élément de la liste est lié au suivant :





On dispose des pointeurs suivants :

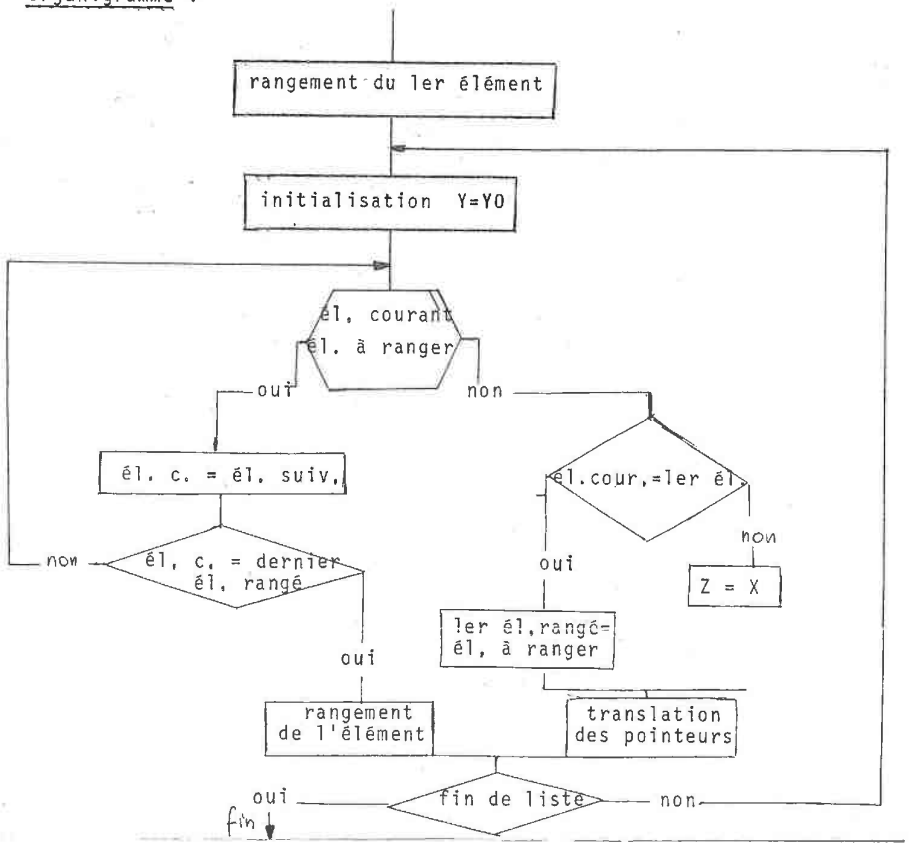
XO ——> tête de la liste à ranger et 1er élément rangé

X ——> élément à ranger

Y ——> élément courant comparé à l'élément à ranger

Z ——> élément précédant l'élément courant.

Organigramme :



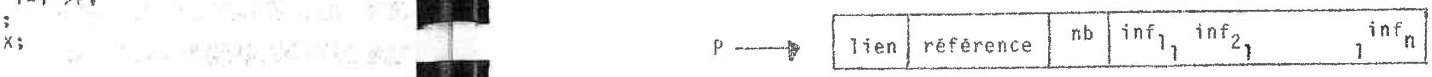
```
CLASSEMENT: PROCEDURE (XO);   /* XO TETE DE LISTE */
   DCL 1 ELEMENT BASED (X),
         2 P POINTER,
         2 MOT CHAR (15),
   (XO, Y , Z, TEMP) PTR;
   X=XO->P;
   XO->P=NULL;
A: DO WHILE (X¬=NULL);
         Y=XO;
      B: DO WHILE (Y¬=NULL);
         IF X->MOT>Y->MOT THEN DO; Z=Y;
                                   Y=Y->P;
                                END;
         ELSE DO; IF Y=XO THEN XO=X;
                  ELSE Z->P=X;
                  TEMP=X;
                  X=X->P;
                  TEMP->P=Y;
                  GO TO E;
                END;
      END B;
         Z->P=X;
         X=X->P;
         TEMP=Z->P; TEMP->P=NULL;
E: END A;
   P=XO; /* IMPRESSION DE LA LISTE CLASSEE*/
   DO WHILE (P¬=NULL);
   PUT EDIT (MOT) (SKIP, A);
   END;
END CLASSEMENT;
```

3 - ADJONCTION D'ELEMENTS DANS UNE LISTE CLASSEE ALPHABETIQUEMENT.

```
ADJONCTION: PROC (TERME, XO);
   DCL   TERME CHAR (*),    (XO, P, PREC) PTR,
      1 ELEMENT BASED (X),
           2 Q PTR,
         2 MOT CHAR (15);
   X=XO; /* XO TETE DE LISTE */
   DO WHILE (X->MOT < TERME & X¬=NULL);
        PREC=X;
        X=X->Q;
   END;
        ALLOCATE ELEMENT SET (P);
        P->MOT=TERME;
        PREC->Q=P;
        P->Q=X;
END ADJONCTION;
```

4 - CONSULTATION D'UNE LISTE :

Cette procédure permet de trouver les titres des ouvrages d'un auteur dans une table classée alphabétiquement. La table est formée d'éléments de structure suivante :



XO désigne la tête de liste.

```
CONSUL: PROCEDURE (AUTEUR, TITRES);
   DCL 1 TABLE BASED (P),
            2 Q POINTER,
            2 REF CHAR (10),
            2 NB,
          2 INF (N REFER (NB)) CHAR (20), N,
   XO STATIC POINTER, (TITRES(*), AUTEUR) CHAR (*);
   P=XO;
   DO WHILE (P->REF < AUTEUR);
           P=P->Q;
           IF P=NULL  THEN GO TO E;
   END;
   IF P->REF=AUTEUR THEN  TITRES=INF;
   ELSE E: TITRES(1)=' ';
   RETURN;
END CONSUL;
```

QUATRIEME PARTIE

INTERET DE PL/1 POUR LES LITTERAIRES

COMPARAISON A COBOL ET ALGOL LINGUISTIQUE

INTERET de PL/1 POUR LA PROGRAMMATION DES TRAVAUX LITTERAIRES

PL/1 est adapté à la programmation des travaux littéraires par les éléments suivants :

1 - LA NOTION DE CHAINE ET LES FONCTIONS DE TRAITEMENT DE CHAINES :

Une chaîne est une suite de caractères appartenant au jeu de caractères, donc une phrase peut être considérée comme une chaîne de caractères. La chaîne peut être de longueur fixe ou de longueur variable, auquel cas on fixe la longueur maximum.

Exemple : DECLARE MØT CHARACTER(25) VARYING ;

MØT='MAXIMUM'

la longueur actuelle de MØT est 7.

Les divers traitements possibles sur les chaînes sont l'assemblage de plusieurs chaînes par concaténation, et, grâce aux fonctions incorporées, la création de sous-chaînes, la reconnaissance d'une suite de caractères dans une chaîne donnée, le remplacement d'un groupe de caractères par un autre.

Exemples : DECLARE MØT CHARACTER(25) VARYING ;

MØT='C''EST';

1) accès au caractère :

LETTRE=SUBSTR(MØT,1,1) ⟹ C

2) accès au groupe de caractères :

3 dernières lettres du mot : EST

GRØUPE=SUBSTR(MØT,LENGTH(MØT)-2)

3) concaténation de plusieurs chaînes :

```
L='LANGAGE';
PHRASE=MØT || ' UN ' || L;
```

PHRASE prend la valeur 'c'est un langage'

4) Reconnaissance d'une suite de caractères dans un mot, d'un mot dans une phrase.

- mot contenant le préfixe 're'

```
IF INDEX(MØT,'re')=1 THEN RAD=SUBSTR(MØT,3);
```

Si mot identifie 'retour', RAD prend la valeur 'tour'.

- phrase contenant la négation NE ... PAS

```
J=INDEX(PHRASE,' NE ');
IF J THEN
  DØ; PH=SUBSTR(PHRASE,J+4);
     IF INDEX (PH,' PAS ') THEN
      /* la phrase contient la négation */
       PUT EDIT (PHRASE) (A);
  END;
  /* la phrase ne contient pas la négation
```

5) Remplacement d'un groupe de caractères :

Le verbe laver en vieux français a pour radical lef aux personnes du singulier du présent.

```
Si VERBE = 'je lave'
   SUBSTR (VERBE,4,3) = 'lef' ;
```

2 - LA NOTION DE STRUCTURE :

Elle permet de décomposer une information en éléments qui peuvent

être traités séparèment et différemment.

Exemple : un texte se présente sous la forme suivante :

Auteur	Titre	Chapitre	Phrase	Suite	Texte

```
DECLARE 1 ØUVRAGE,
          2 REF,
            3 AUTEUR CHARACTER(10),
            3 TITRE CHARACTER(2),
            3 CHAPITRE FIXED(2),
            3 PHRASE FIXED(3),
          2 SUITE FIXED(1),
          2 TEXTE CHARACTER(62);
```

La référence sert à situer le morceau de texte, la suite situe le morceau de texte dans la phrase.

Il est possible de définir une structure de même modèle qu'une partie d'une autre structure ou qu'une autre structure ; ainsi dans l'exemple précédent il sera nécessaire, à chaque début de phrase, de garder la référence pour la comparer à la référence de chaque morceau de texte appartenant à la phrase.

```
1  REFERENCE LIKE ØUVRAGE.REF
```

Référence :

Auteur	Titre	Chapitre	Phrase

ce qui équivaut à la déclaration suivante :

```
1 REFERENCE,
    2 AUTEUR CHARACTER(10),
    2 TITRE CHARACTER(2),
    2 CHAPITRE FIXED(2),
    2 PHRASE FIXED(3),
```

3 - LA NOTION DE SOUS-TABLEAU ET LES FONCTIONS DE MANIPULATION DE TABLEAUX :

Il est possible d'isoler dans un tableau, les éléments correspondant à la variation d'un ou plusieurs indices, ainsi dans le tableau à 2 dimensions des terminaisons de l'indicatif des verbes du 1er groupe, on peut considérer :

- le sous-tableau d'un temps : présent, futur, ...
- ou le sous-tableau des terminaisons d'une personne.

Les fonctions de manipulation de tableaux permettent, entre autres, de faire la somme ou le produit de tous les éléments d'un tableau.

Exemple : Nombre d'occurences des voyelles dans les mots d'une liste.

ØCCUR(N,6) où N représente le nombre de mots,
6 nombre des voyelles

ØCCUR	a	e	i	o	u	y
mot 1						
mot 2						
mot 3						

ØCCUR(*,1) est le tableau unidimensionnel donnant les nombres d'occurences de a dans chaque mot

SUM(ØCCUR(*,1)) donne le nombre d'occurences de a dans la liste des n mots

SUM(ØCCUR) donne le nombre de voyelles dans la liste des n mots.

4 - LES EXPRESSIONS MIXTES :

PL/1 admettant toute combinaison de symboles ayant un sens en soi, les expressions mixtes sont autorisées.

Exemple : 'LA PHRASE EST COMPOSEE DE ‖L-NBSEP+1‖ ' LETTRES.'

Cette expression caténaire est composée de 2 chaînes et d'une expression arithmétique qui est calculée en premier lieu d'après l'ordre de priorité des opérateurs, puis convertie en une chaîne de caractères.

5 - LES EXPRESSIONS DE TABLEAUX OU DE STRUCTURES :

L'AFFECTATION DE TABLEAU OU DE STRUCTURE :

Elles permettent de modifier, en une seule instruction, toutes les valeurs d'un tableau ou d'une structure, par exemple pluriel d'une liste de mots, conjugaison .....

```
PRESENT='CHANT'‖ INDICATIF(*,1);
```

Cette instruction est équivalente à la suite d'instructions :

```
PRESENT(1)='CHANT'‖ INDICATIF(1,1);
PRESENT(2)='CHANT'‖ INDICATIF(2,1);
.....
PRESENT(6)='CHANT'‖ INDICATIF(6,1);
```

Insertion de * entre les éléments d'une structure en vue d'une sortie.

```
1 REFERENCE,
    2 AUTEUR CHARACTER(10),
    2 TITRE CHARACTER(20),
    2 NB CHAPITRES,
    ..............
STRUCT = REFERENCE ||'*';
```

STRUCT	Auteur	*	Titre	*	nb de chapitres	*...

Les tableaux intervenant dans les expressions et l'affectation doivent avoir des bornes identiques, les structures même composition. Toutefois, l'affectation par nom permet d'agir sur les éléments de même nom de structures quelconques, ceci est utile dans la technique de liste.

Exemple : Construction d'une arborescence

```
          A: PROCEDURE (ELEMENT);
DECLARE     1 ELEMENT,
              2 NIVEAU FIXED,
              2 NØM CHARACTER(*),
            1 NØEUD CØNTRØLLED(P),
              2 NØM CHARACTER(LENGTH(ELEMENT.NØM)),
              2 NIVEAU FIXED,
              2 (DESCENDANT,EL DRØIT,PREDECESSEUR)PØINTEUR,
             ..............
  NØEUD=ELEMENT,BY NAME;
  ..........
```

Cette instruction transfère dans NØEUD.NØM le nom de l'élément et dans NØEUD.NIVEAU, le niveau de l'élément.

6 - ITERATION :

Les valeurs prises par la variable contrôlée sont celles d'expressions scalaires quelconques, en particulier des chaînes.

Exemple : Recherche des voyelles dans un mot

```
DØ VOY='a','e','i','o','u','y';
```

7 - ENTREE-SORTIE :

PL/1 offre des possibilités très étendues d'entrée-sortie. L'utilisateur communique au programme les opérations qu'il désire voir réaliser à l'aide d'ordres accompagnés d'options et d'attributs qui indiquent la description du fichier et la manière dont il doit être traité.

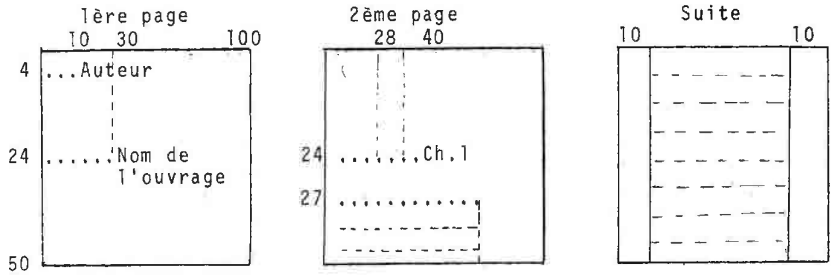
La transmission par liste permet de lire ou écrire rapidement un grand nombre de données placées en séquence : lecture de tableaux, sortie de résultats sans souci de présentation.

La transmission avec édition permet de lire des informations se présentant sous n'importe quelle forme, de sauter des éléments, ou de sortir des résultats sous la forme désirée. Elle permet de générer un rapport grâce aux options qui permettent de décrire une page imprimée : titre, numérotation des pages, saut à la page suivante ou à une ligne pré-déterminée, positionnement en un point quelconque d'une ligne, espacement des lignes, dimension d'une ligne ou d'une page.

Exemple :

On veut publier le texte correspondant à la structure OUVRAGE de

la manière suivante :

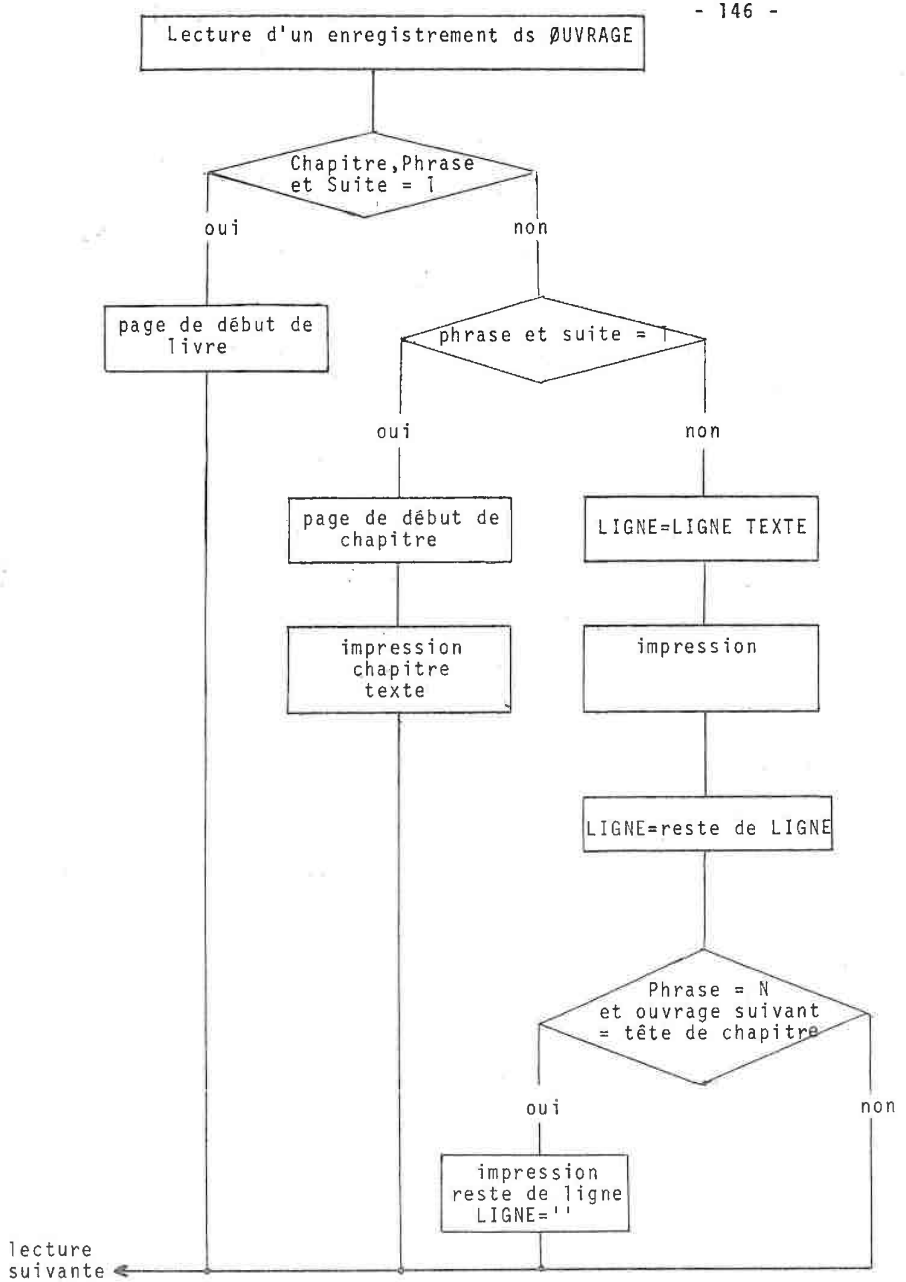


Une page contient 50 lignes de 100 caractères

```
DECLARE 1 ØUVRAGE,
          2 REF,
             3 AUTEUR CHARACTER(10),
             3 TITRE CHARACTER(2),
             3 CHAPITRE FIXED(2),
             3 PHRASE FIXED(3),
          2 SUITE FIXED(1),
          2 TEXTE CHARACTER(62),
    LIGNE CHARACTER(120) VARYING INITIAL(''),
    LISTING FILE STREAM PRINT
    PAGESIZE(50);
```

La carte chapitre donne le nombre N de phrases dans le chapitre

lecture et traitement du texte.



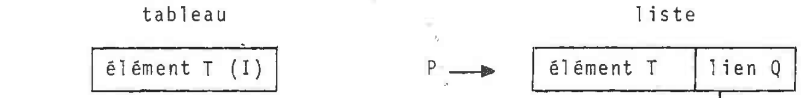
```
IF CHAPITRE=1 & PHRASE=1 & SUITE=1 THEN /*1ère page*/
    PUT FILE (LISTING) EDIT (AUTEUR,TITRE)(SKIP(4),
    COLUMN(10),A(10),SKIP(20),COLUMN(30),A);
IF PHRASE=1 & SUITE=1 THEN /* imprimer la fin du chapitre
précédent, saut à la page suivante et impression de la page
début de chapitre*/
   DO; PUT FILE (LISTING) EDIT (LIGNE)(R(F));
       LIGNE='';
       PUT FILE (LISTING) EDIT ('CHAPITRE ',CHAPITRE)
       (PAGE,LINE(24),COLUMN(40),A,F(2),SKIP(3));
       /* impression de la 1ère ligne */
       PUT FILE (LISTING) EDIT (TEXTE)(COLUMN(28),A);
   END;
   LIGNE=LIGNE TEXTE;
   IF LENGTH(LIGNE)> =100 THEN
       DO; PUT FILE (LISTING) EDIT (SUBSTR(LIGNE,1,100))(R(F));
           LIGNE=SUBSTP(LIGNE,101);
       END;
```

PL/1 prévoit la transmission d'enregistrements entiers d'un fichier. Le fichier est soit à accès séquentiel auquel cas il est possible de le lire du dernier enregistrement au premier, ou de sauter un nombre déterminé d'enregistrements, soit à accès direct auquel cas une clé permet de sélectionner l'enregistrement désiré.
Des instructions de mise à jour permettent d'effacer un enregistrement, de le modifier ou de le remplacer par un autre si le fichier est déclaré fichier de mise à jour.

8 - LE TRAITEMENT DES LISTES :

Une liste est une suite d'éléments de même composition reliés par un ou plusieurs liens qui permettent de cheminer dans la liste. Le traitement des listes permet une manipulation plus souple des éléments qu'un tableau en dissociant l'ordre logique des éléments de leur organisation physique ; toutefois, la place occupée en mémoire par une liste est plus grande que celle occupée par un tableau, la recherche d'un élément particulier se fait en cheminant dans la liste et en comptant jusqu'à l'élément désiré, ce qui est moins efficace que la variation de l'indice d'un tableau, mais on gagne en efficacité en parcourant la liste à partir de l'élément utilisé précédemment au lieu de revenir en tête de liste.

Exemple :




```
DO I=1 TO N;                    P=P0;/*P0 tête de liste*/
   .......                      DO WHILE (P1=NULL);
   T(I).....                    T .......
END;                            P=Q;
                                END;
```

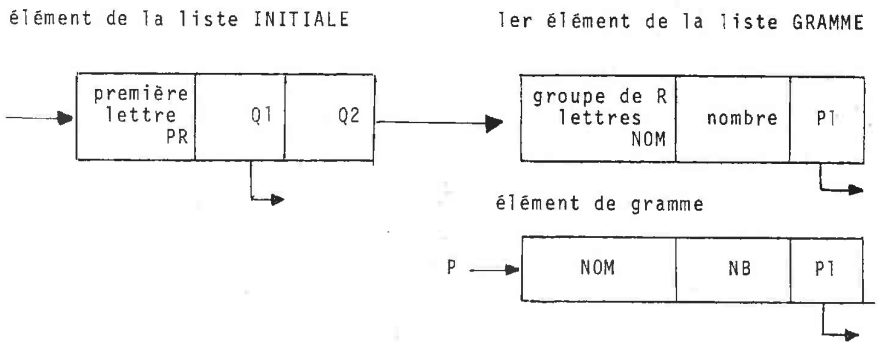
Les arrangements des éléments d'une liste se font sans déplacement.
Les éléments d'une liste peuvent être créés au fur et à mesure de leur apparition.

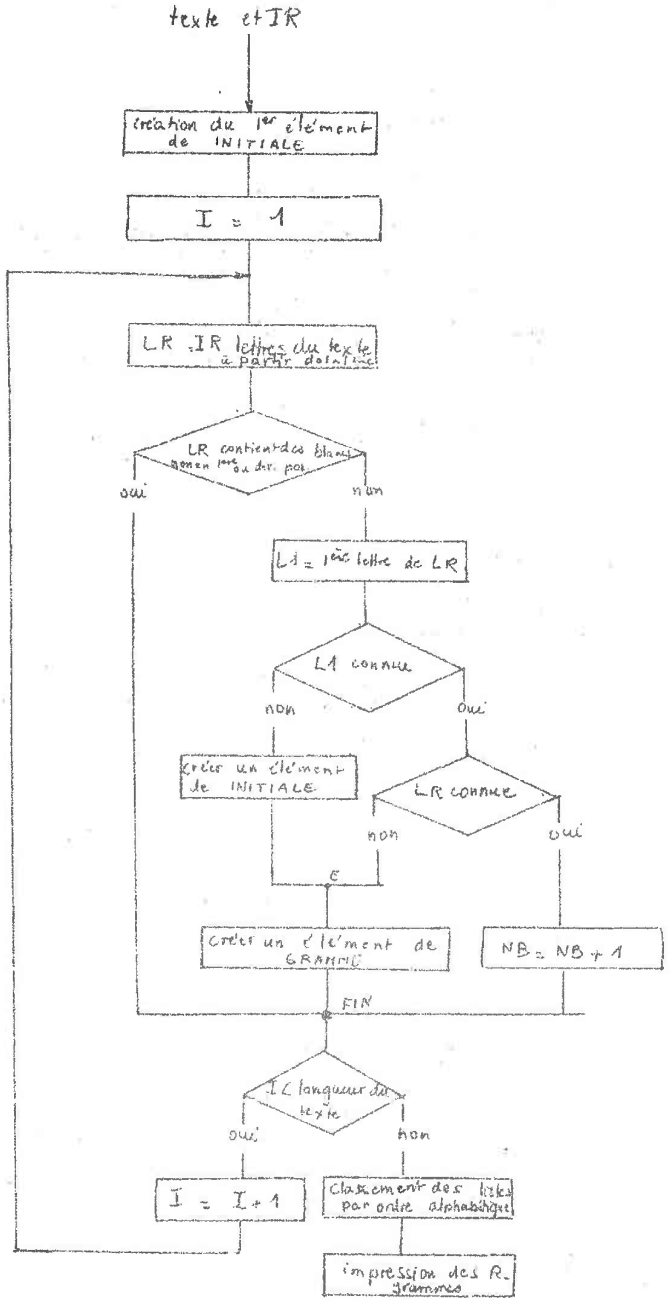
Exemple : Enumération et comptage des R-grammes d'un texte.

Un R-gramme est un groupe de R lettres consécutives. Cette étude donne les groupes de lettres les plus fréquemment utilisées, les mots de longueur R-2, les groupes les plus fréquents en début et fin de mot.

La technique de liste est utile dans ce cas, les éléments sont créés au fur et à mesure de leur apparition d'où un gain de place par rapport à un tableau de grandes dimensions où de nombreux éléments seraient vides.

On utilise 2 listes pour permettre un cheminement plus rapide





```
\PPEL_G     : PROC OPTIONS (MAIN);
```

```
        APPEL_G     : PROC OPTIONS (MAIN);
           DCL TEXTE CHAR (150) VAR;
           TEXTE='UNE LISTE EST UN ENSEMBLE D''INFORMATIONS PLACEES EN MEMOIRE
                 ET LIEES PAR UN OU PLUSIEURS LIENS QUI PERMETTENT DE PASSE
                 D''UN ELEMENT A UN AUTRE';
           PUT LIST (TEXTE) SKIP;
           CALL R_GRAMME (3, TEXTE);
        R_GRAMME: PROCEDURE(IR, TEXTE);
           DCL TEXTE CHAR (*), IR, L1 CHAR (1), LR CHAR (IR),
               1 GRAMME BASED (P),
                   2 NOM CHAR (3),
                   2 NB,
                   2 P1 POINTER,
               1 INITIALE BASED (Q),
                   2 PR CHAR (1),
                   2 Q1 PTR,
                   2 Q2 PTR,
              (Q0 STATIC EXTERNAL, X, Y, TEMP)PTR;
           ALLOCATE INITIALE SET (Q0); Q=Q0;
           Q1, Q2=NULL;
           PR=' ';
        ITER: DO I=1 TO LENGTH (TEXTE)+1-IR;
                   LR= SUBSTR (TEXTE, I, IR);
                   DO J=2 TO IR-1;
                       IF SUBSTR (LR, J, 1)=' ' THEN GO TO FIN;
                   END;
                   L1=SUBSTR (LR, 1, 1);
                   Q=Q0;
                   DO WHILE (L1¬=PR);
                       IF Q1=NULL THEN /* CREATION D'UN ELEMENT DE INITIALE */
                               DO; ALLOCATE INITIALE SET (X);
                                   Q->Q1=X;
                                   PR=L1;
                                   Q1=NULL;
                                   ALLOCATE GRAMME SET (Y);
                                   Q2=Y;
                                   GO TO E;
                               END;
                           Q=Q1;
                   END;
                   /* INITIALE CONNUE, RECHERCHE DE LR */
                   P=Q2;
                   DO WHILE (P¬=NULL);
                       IF LR=NOM THEN DO; NB=NB+1;
                                          GO TO FIN;
                                      END;
                           TEMP=P;
```

```
PPEL_G     : PROC OPTIONS (MAIN);

                    P=P1;
                END;
                /* CREATION D'UN ELEMENT DE GRAMME */
                ALLOCATE GRAMME SET (Y);
                TEMP->P1=Y;
            E: NOM=LR;
               NB=1;
               P1=NULL;
        FIN: END ITER;
        /* IMPRESSION */
          Q=Q0;
          DO WHILE (Q¬=NULL);
                PUT EDIT (PR) (SKIP (5), COLUMN (20), A(1));
                P=Q2;
                DO WHILE (P¬=NULL);
                   PUT EDIT (NOM, NB) (SKIP, COLUMN(25), A, X(5),F(5));
                   P=P1;
                END;
             END;
        END R_GRAMME;
        END APPEL_G;
```

CONCLUSION :

PL/1 satisfait les exigences des problèmes littéraires. Les programmes s'écrivent facilement, toutefois l'adjonction de quelques fonctions de traitement des chaînes telles que celle qui permettrait d'isoler un mot dans une chaîne, ou celle qui donnerait le nombre de mots d'une chaîne, faciliterait l'étude des textes. La notion de structure, la notion de sous-tableau, les expressions et affectation de tableaux et de structures sont très utiles et simplifient l'écriture du programme. Les déclarations complètes peuvent paraître lourdes mais le programmeur novice peut ignorer certaines des possibilités offertes par PL/1. La structure de programme en blocs permet de libèrer les mémoires qui ne sont plus utilisées, ce qui est avantageux pour le traitement des problèmes littéraires dont les données sont en général des chaînes nécessitant une grande place en mémoire. Lorsque la structure de chaîne est trop rigide, (analyse syntaxique et traduction automatique) la technique de listes permet de traiter ces problèmes ; mais contrairement aux langages de listes, PL/1 permet d'éviter l'emploi de listes dans les cas simples. De plus les possibilités très étendues d'Entrée-Sortie permettent de lire ou sortir des résultats, sous n'importe quelle forme.

COMPARAISON DE CE SOUS-ENSEMBLE DE PL/1 A ALGOL LINGUISTIQUE

L'Algol linguistique est une extension d'Algol adaptée au traitement des informations non numériques.

I - NOTION DE CHAINE ET TRAITEMENT DES CARACTERES :

La chaîne est définie de même manière qu'en PL/1 ; c'est une suite de caractères de longueur fixe ou variable.

Exemple :

chaîne explicite "CHAINE"

Les opérateurs de concaténation et de coupure permettent d'assembler des chaînes, d'isoler les N premiers ou derniers caractères d'une chaîne.

Exemples : on définit la chaîne : mot CHAINE MOT [25]

1) accès au caractère

MOT [I] désigne le $i^{ème}$ caractère de la chaîne identifiée par MOT.

2) accès au groupe de caractères

L'accès aux N premières ou dernières lettres du mot est immédiat grâce aux opérateurs de coupure

PR := N $1^{ère}$ LETTRE DE MOT ;

DER:= N DERNIERE LETTRE DE MOT ;

L'accès à un groupe de caractères interne à la chaîne nécessite le découpage de la chaîne.

J caractères de MOT à partir du ième.

GROUPE := J DERNIERE LETTRE DE (I+J-1)
PREMIERE LETTRE DE MOT;

La fonction SUBSTR de PL/1 permet toutes ces opérations

MOT [I]	≤	SUBSTR(MØT,I,1)
PR	≤	SUBSTR(MØT,1,N)
DER	≤	SUBSTR(MØT,LENGTH(MOT)+1-N)
GROUPE	≤	SUBSTR(MØT,I,J)

3) Concaténation de plusieurs chaînes

MOT SUIVI DE "S"

4) Reconnaissance d'une suite de caractères dans une chaîne et rang dans la chaîne.

Soit MØT la chaîne, RAD la suite de caractères et R le rang

```
J = LØNG (RAD);
POUR I=1 PAS 1 JUSQUA N FAIRE
 SI J DERNIERE LETTRE DE (I+J-1 1ère LETTRE DE
   MØT) = RAD ALORS DEBUT R=I;
                        ALLERA ETI;
                  FIN;
R = 0;
ETI :
```

Algol linguistique possède la notion de chaîne à structure de suite ; c'est une chaîne formée d'éléments séparés par S. La fonction ELEM (CHAINE,I) permet d'isoler le Ième élément de CHAINE, la fonction RANG (CHAINE) donne le nombre d'éléments de CHAINE. Cette notion est très utile :

1°) pour l'étude des textes, une phrase est considérée comme une chaîne à structure de suite dont les éléments sont les mots.

Exemple : phrase contenant la négation NE ... PAS

```
PØUR I:=1 PAS 1 JUSQUA RANG(PHRASE) -1 FAIRE
  SI ELEM (PHRASE,I) = "NE" ALØRS
     DEBUT COMMENTAIRE la phrase contient la négation;
     ..............
     FIN
     SINØN;
     ALLERA E1;
  FIN
SINØN;COMMENTAIRE la phrase ne contient pas la négation;

E1: ..........
```

2°) pour ranger les articles d'un dictionnaire

mot 1 S indications 1 S mot 2 S indications 2 S ....

les éléments impairs sont les mots, les éléments pairs les renseignements attachés au mot précédent.

Exemple : procédure de consultation de dictionnaire

MØT	mot à rechercher dans le dictionnaire
RENS	renseignements relatifs au mot trouvés dans le dictionnaire
B	booléen qui prend la valeur VRAI si le mot est dans le dictionnaire

```
PRØCEDURE CØNSUL (MØT,DIC,B,RENS);
   CHAINE MØT,DIC,RENS; BØØLEEN B;
   DEBUT ENTIER I;
         B:=FAUX;
         RENS:=VIDE;
   POUR I:=1 PAS 2 JUSQUA RANG(DIC) FAIRE
        SI MOT = ELEM(DIC,I) ALORS
              DEBUT B:=VRAI;
                    RENS:=ELEM(DIC,I+1);
                    ALLERA ETI;
              FIN;
ETI:FIN
```

PL/1 ne possède pas cette notion mais les procédures ELEM et RANG sont faciles à écrire.

II - GROUPEMENT DES DONNEES :

Dans les 2 langages, les données peuvent être des éléments simples ou des éléments groupés en tableaux, mais Algol exclut la possibilité de tableau de chaînes, PL/1 définit un nouveau groupement d'éléments : la structure qui permet de décomposer une information en parties distinctes ou de regrouper divers éléments se rapportant à la même chose.

La chaîne à structure de suite d'Algol linguistique correspond à une structure PL/1 composée d'éléments terminaux mais non de structures secondaires.

Exemple :

Auteur	titre	chapitre	phrase	suite	texte

Structure PL/1

```
1 OUVRAGE,
   2 REF,
      3 AUTEUR CHAR(10),
      3 TITRE CHAR(2),
      3 CHAPITRE FIXED(2),
      3 PHRASE FIXED(3),
   2 SUITE FIXED(1),
   2 TEXTE CHAR(62)
```

Chaîne à structure de suite correspondante

Auteur	S	Titre	S	Chapitre	S	Phrase	S	Suite	S	Texte

CHAIN [84]

Les différents éléments sont séparés par S

Le n° de phrase sera repéré par ELEM (CHAIN,4)

Le $i^{ème}$ caractère du texte sera identifié par

CHAIN [22 + I]

22 étant le nombre de caractères précédant l'élément texte.

La référence sera gardée dans une chaîne à structure de suite :

REFERENCE [20]

REFERENCE := 20 $1^{ère}$ LETTRE DE CHAIN;

On peut donc représenter une information décomposée en ses éléments en Algol linguistique, mais ils perdent leur signification et deviennent des éléments anonymes de la chaîne à structure de suite aussi est-il nécessaire d'avoir par ailleurs l'image de l'information.

III - LES EXPRESSIONS :

Les expressions simples d'Algol linguistique sont semblables aux expressions scalaires de PL/1. Algol linguistique possède de plus des expressions de la forme :

⟨Proposition SI⟩⟨expression simple⟩ SINON ⟨expression⟩

Exemple :

```
RENS:= SI MØT = ELEM(DIC,I) ALØRS ELEM(DIC,I+1)
       SINØN VIDE;
```

Cette extension des expressions peut être obtenue en PL/1 dans le cas d'expressions arithmétiques et booléennes en se servant des chaînes de bits, mais non dans le cas d'expressions caténaires.

```
X = (Y>0) * Y + ¬(Y>0) * 0 ;
ou  X:= SI Y>0 ALORS Y SINON 0;
```

Algol est moins souple que PL/1 parce qu'il n'y a pas conversion entre chaînes de caractères et nombres même si elle est possible.

Exemple :

```
PHRASE || 'NB DE MOTS:' || N
```

est une expression caténaire valide en PL/1 alors que

```
PHRASE SUIVI DE "NB DE MOTS :" SUIVI DE N
```

n'est pas une expression caténaire valide en Algol.
Elle sera valide par convention si l'expression entière est placée entre crochets.
Les expressions de tableaux et de structure n'existent pas.

IV - LES INSTRUCTIONS :

Elles ont une forme à peu près identique dans les deux langages. Toutefois dans le cas de l'itération, l'instruction POUR

3ème forme d'Algol : POUR var. = expression TANT QUE (exp. booléenne) ne correspond pas à l'utilisation la plus fréquente.
En PL/1 la clause WHILE peut être
- seule DO WHILE (expression scalaire);
- associée à la variation de la variable contrôlée

```
DO var. contrôlée = variation de la variable clause WHILE
```

V - ENTREE-SORTIE :

Les E/S ne figurent pas dans le langage de base ALGOL. Les possibilités offertes par PL/1 sont très étendues :
- transmission en continu
- transmission par enregistrements.

VI - STRUCTURE DE PROGRAMME :

Un programme ALGOL se compose d'un seul bloc externe alors qu'un programme PL/1 se compose d'un ou plusieurs blocs externes dits procédures externes.
Un bloc externe peut dans l'un et l'autre langage contenir des blocs internes et des procédures internes.
Les deux langages offrent la même possibilité de réservation dynamique de mémoire, PL/1 offre, de plus, la réservation contrôlée de mémoire.
En Algol les déclarations sont toujours placées en tête de bloc, elles sont faites par type, chaque type est suivi de la liste des variables de ce type. En PL/1, elles peuvent se trouver n'importe où dans le bloc où elles sont utilisées, chaque variable est suivie de sa liste d'attributs. Les déclarations complètes de PL/1 sont plus complexes que celles d'Algol mais l'attribution par défaut permet de réduire le nombre des attributs attachés à chaque variable.

Algol linguistique permet une représentation linéaire des textes satisfaisante, la notion de chaîne à structure de suite étant adaptée à la description d'un texte. Mais cette notion est très inefficace pour faire l'analyse syntaxique d'une phrase. De plus les entrées-sorties ne sont pas assez développées et le traitement de listes est inexistant.

COMPARAISON DE CE SOUS-ENSEMBLE DE PL/1 A COBOL

Le langage de programmation Cobol est destiné au traitement des problèmes commerciaux ; mais parce qu'il accepte des données de type alphabétique, il peut être utilisé pour le traitement des problèmes littéraires.

I - NOTION DE CHAINE ET TRAITEMENT DES CARACTERES :

La chaîne telle qu'elle est définie en PL/1 ou en Algol linguistique n'existe pas en cobol, il est question de zone alphabétique ou alphanumérique de longueur fixée. On n'a pas accès de manière naturelle à un caractère ou à un groupe de caractères internes à la zone. Les opérations sur les chaînes et les fonctions de traitement de chaînes n'existent pas. Toutefois il est possible de comparer deux zones même si elles sont de longueur différente, de reconnaître un caractère donné, de compter ses occurences dans une zone, de le remplacer par un autre caractère.

Exemple : Si TEXTE est "langage de programmation"

- Compter le nombre de A de TEXTE

  EXAMINE TEXTE TALLYING ALL"A"

  la valeur résultante du registre TALLY sera 4

- Compter le nombre de lettres du 1er mot

  EXAMINE TEXTE TALLYING UNTIL FIRST SPACE

- Remplacer les espaces du texte par $

  EXAMINE TEXTE REPLACING ALL SPACE BY "$"

- Supprimer le 1er mot de TEXTE

```
EXAMINE TEXTE REPLACING UNTIL FIRST SPACE BY SPACE
```

On a accès au caractère en considérant chaque enregistrement du texte comme un tableau unidimensionnel ayant pour borne la longueur de l'enregistrement, dont les éléments sont des zones de un caractère alphabétique ou alphanumérique.

Exemple :

```
01 TABLEAU.
   02 ELEMENT X OCCURS 24 TIMES.
```

ELEMENT(I) désigne la ième lettre du texte.

Un mot sera isolé en transférant dans un autre tableau les éléments du tableau initial compris entre 2 éléments vides.

II - GROUPEMENT DES DONNEES-STRUCTURES :

En PL/1 et Cobol, les données peuvent être des éléments simples ou des éléments groupés en tableaux ou en structures. Les structures sont définies de la même manière dans les deux langages, mais en PL/1, l'option LIKE permet d'alléger l'écriture.

Les 2 langages offrent la possibilité de désigner par le même nom des éléments identiques appartenant à des structures différentes, ils possèdent la notion de tableau d'éléments ou de structures.

L'enregistrement suivant se définit ainsi :

Auteur	Titre	Chapitre	Phrase	Suite	Texte

```
01 ØUVRAGE.
   02 REF.
      03 AUTEUR PICTURE X(10).
```



```
      03 TITRE PICTURE XX.
      03 CHAPITRE PICTURE 99.
      03 PHRASE PICTURE 999.
   02 SUITE PICTURE 9.
   02 TEXTE PICTURE X OCCURS 62 TIMES.
```

En Cobol les tableaux sont considérés comme des structures

```
01 TABLEAU.
   02 LIGNE OCCUR I TIMES.
      03 ELEMENT PICTURE X OCCURS J TIMES.
```

correspond au tableau PL/1

```
TABLEAU(I,J) CHARACTER(1)
```

On peut avoir accès à

un élément : en COBOL ELEMENT(I1,J1), en PL/1 TABLEAU(I1,J1)

une ligne : en COBOL LIGNE(I1), en PL/1 TABLEAU(I1,*)

en PL/1 on peut aussi avoir accès à une colonne TABLEAU(*,J1)

III - LES EXPRESSIONS : La présence des opérateurs et la possibilité d'écrire des expressions arbitrairement compliquées en PL/1 rend plus aisée le traitement des données. Les expressions caténaires, les expressions de tableaux et les expressions de structures n'existent pas en Cobol. Les 2 langages offrent les mêmes possibilités de relations et d'expressions logiques.

IV - LES INSTRUCTIONS :

L'affectation simple, multiple ou par nom, l'instruction de saut, l'instruction conditionnelle et l'itération existent dans les deux langages.

Cependant en Cobol, l'affectation de tableau ou de structure n'est pas permise, l'affectation par nom est limitée aux opé-

rations de mouvement et d'addition ; une nouvelle instruction conditionnelle ne peut pas apparaître dans la clause ELSE.

L'ordre DØ de PL/1, tout comme l'ordre PERFØRM de Cobol permet :

- soit d'indiquer le début d'un groupe en lui affectant un nom
- soit de commander l'exécution de ce groupe plusieurs fois, en y faisant varier la valeur d'une variable, jusqu'à réalisation d'une condition.

Les valeurs prises par la variable contrôlée d'un ordre PERFØRM sont numériques alors que celles prises par la variable contrôlée de l'ordre DØ sont quelconques, en particulier des caractères.

La variable contrôlée, en PL/1, peut prendre les valeurs d'une liste de spécifications.

V - ENTREE-SORTIE :

Les E/S Cobol se font à l'aide de fichiers décrits entièrement dans la DATA DIVISION. Ceci correspond aux E/S par enregistrements de PL/1. La manipulation des fichiers à accès séquentiel est amélioré en PL/1 par les éléments suivants : lecture ou écriture à partir d'un enregistrement déterminé, saut à la lecture d'un nombre donné d'enregistrements, possibilité de lecture du dernier enregistrement au premier.

Un fichier peut être dans les 2 langages à accès direct, les enregistrements sont alors repérés par des clés qui permettent de sélectionner l'enregistrement désiré.

De plus PL/1 offre la possibilité de transmission en continu : transmission par liste, transmission avec édition.

VI - STRUCTURE DE PROGRAMME :

Un programme COBOL a le niveau d'un bloc PL/1, il comprend 4 divisions :

IDENTIFICATION DIVISION

ENVIRONMENT DIVISION

DATA DIVISION qui décrit les données et les variables utilisées dans le programme.

PROCEDURE DIVISION qui comprend les instructions.

La déclaration de toutes les variables en tête de programme exclut la possibilité de zones dont la longueur est définie au cours du programme et de tableaux de bornes variables. Il y a réservation statique de mémoire pour toutes les variables.

L'ordre PERFØRM permet d'appeler un groupe d'instructions et non une procédure au sens habituel, c'est-à-dire sous-programme dépendant de paramètres.

Un programme PL/1 est beaucoup moins rigide. Il peut comprendre plusieurs procédures indépendantes ou imbriquées. Les réservations de mémoire sont statiques, dynamiques ou contrôlées, ce qui permet une meilleure utilisation de la mémoire et un encombrement moindre. La possibilité de fractionner un programme en plusieurs blocs indépendants permet de se servir de procédures déjà écrites et de les adapter au problème particulier à l'aide de paramètres.

Cobol tant que l'on reste au niveau du caractère permet de traiter facilement certains problèmes littéraires tels que vérification d'écriture, comptages de lettres ou de groupes de lettres, mais n'est pas adapté à la souplesse d'un texte ou du mot.

L'absence de gestion automatique ou contrôlée de la mémoire ne permet pas une bonne utilisation de la mémoire. L'impossibilité de fractionner un programme en plusieurs blocs indépendants oblige à réécrire chaque bloc pour différents programmes avec les variables actuelles.

A N N E X E

EXEMPLES DE PROGRAMMES

A N N E X E

I - RECHERCHE DES PHRASES OU IL EXISTE DES MOTS DONNES :

Les phrases, de longueur variable, ont la forme suivante :

Référence	Texte
1 8	

Une phrase est une suite de mots et de signes de ponctuation séparés par des blancs.

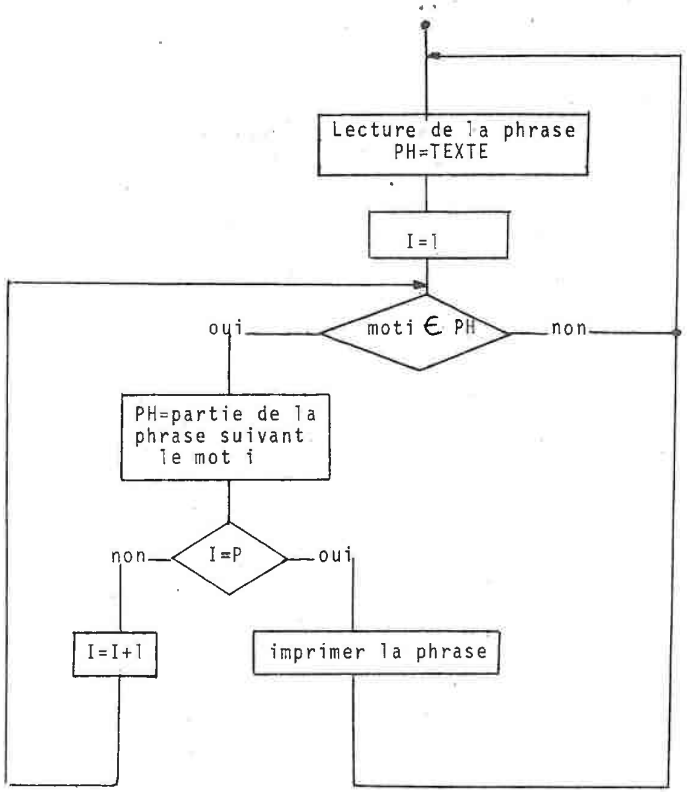
Les mots donnés sont encadrés par un blanc

' mot '

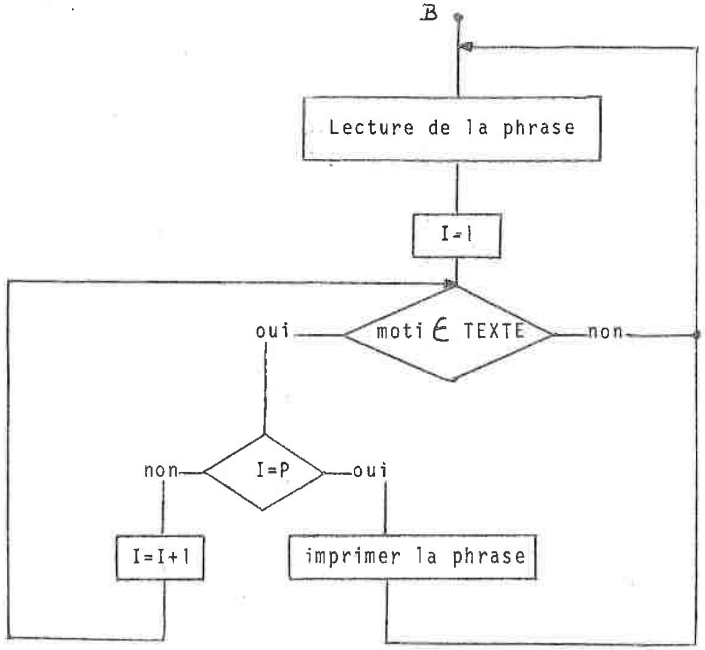
Les phrases sont lues sur un fichier d'entrée X.

On demande d'imprimer les phrases dans lesquelles apparaissent les mots donnés.

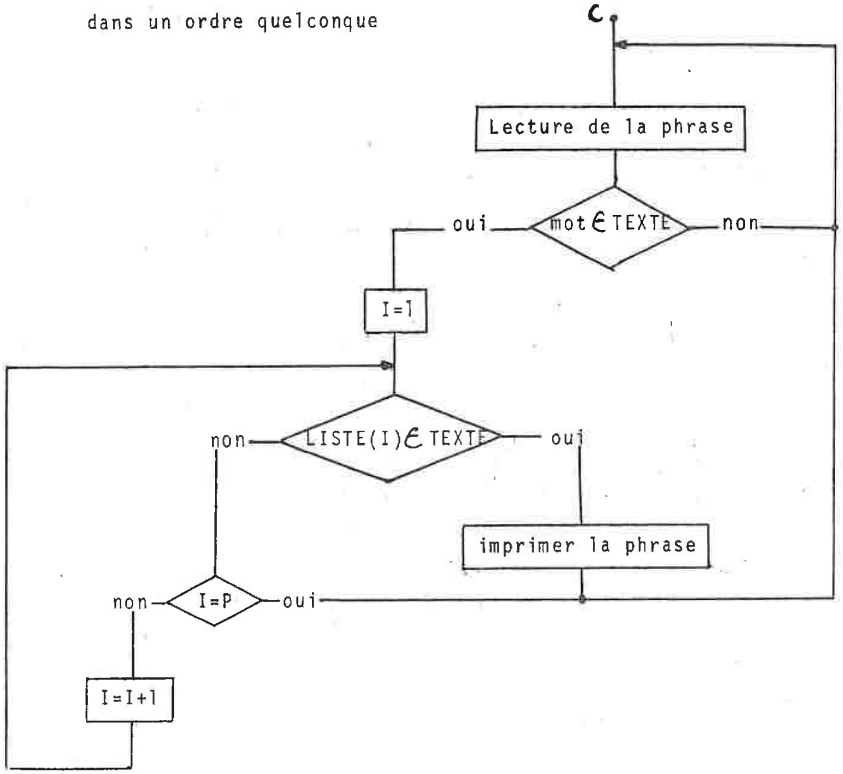
1°) Phrases où il existe p mots donnés consécutifs ou non, dans l'ordre .



2°) Phrases où il existe p mots dans un ordre quelconque



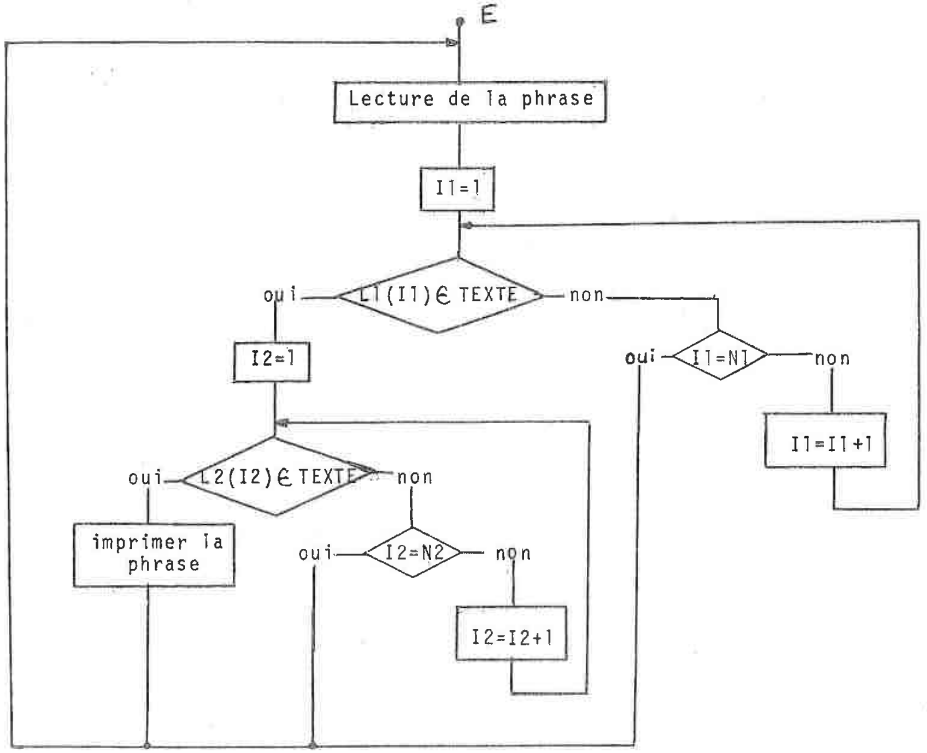
3°) Phrases où il existe un mot donné et un mot d'une liste de p mots dans un ordre quelconque



4°) Phrases où il existe 2 fois le même mot,

Il suffit d'appeler la procédure A (1°) en donnant à P la valeur 2 et aux 2 éléments du tableau MØT la valeur du mot.

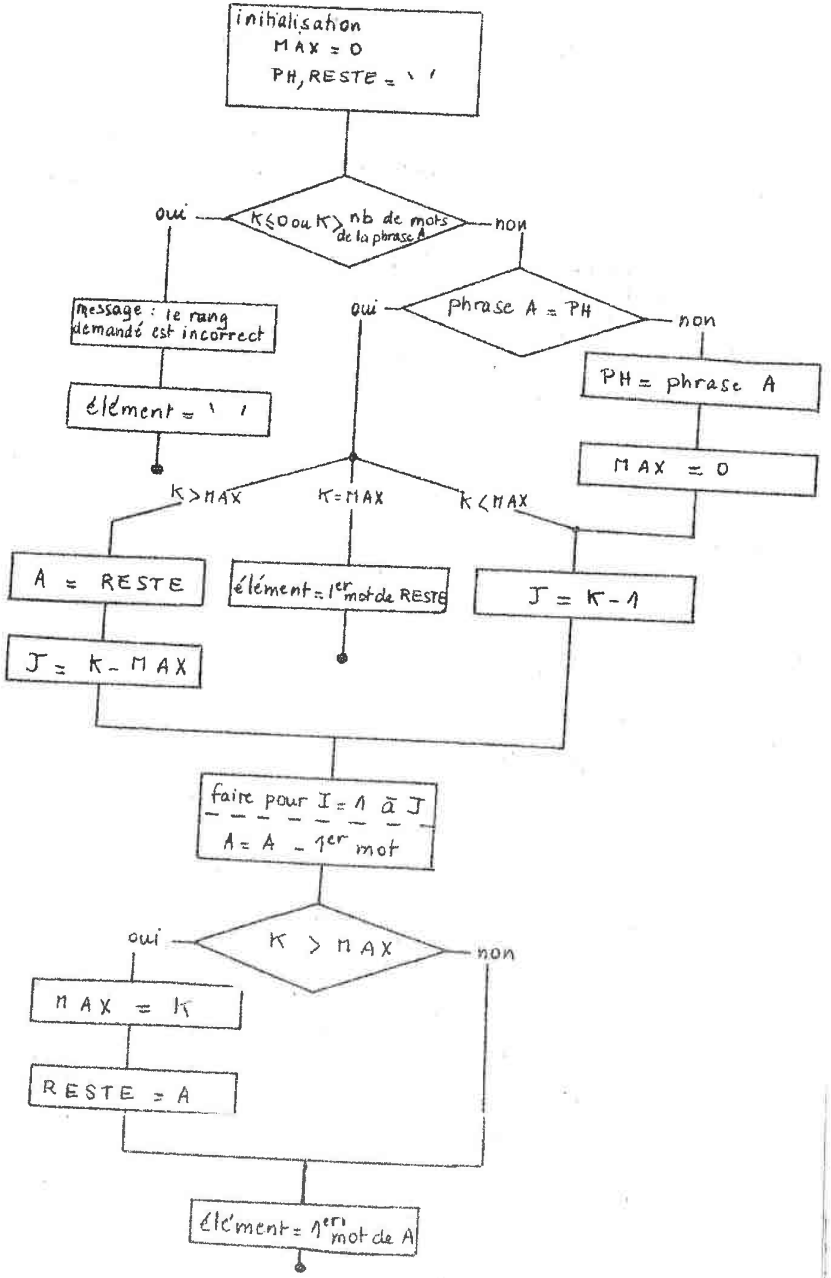
5°) Phrases où il existe un mot d'une liste L1 de N1 mots et un mot d'une liste L2 de N2 mots



II - PROCEDURE DONNANT UN MOT D'UNE PHRASE, DE RANG DONNE

La phrase est conservée, ainsi que le rang maximum demandé et la sous-phrase à partir de ce rang ; ceci permet de ne pas étudier la phrase à partir du début si la procédure est appelée plusieurs fois consécutives pour rechercher des éléments de rang croissant d'une même phrase.

Soient A la phrase, K le rang
PH la phrase conservée, MAX le rang maximum
RESTE la sous-phrase.



```
'EL: PROC OPTIONS (MAIN);


      APPEL: PROC OPTIONS (MAIN);
         DCL LEC FILE INPUT, L(2) CHAR(6)VAR, L1(2) CHAR(5) VAR, N, L2(3)
             CHAR(5) VAR, CHAINE CHAR (80), MOT CHAR (15) VAR;
          DCL ELEM RETURNS (CHAR(15)), RANG RETURNS (FIXED BINARY);
         L(1),L1(1)=' NE '; L(2), L2(1)=' QU'' '; L1(2)=' N'' ';
         L2(2)=' PAS '; L2(3)=' PLUS ';
         CALL D (2, L, LEC);
         CALL B (2, L, LEC);
         CALL C (' NE ', L2, 3, LEC);
          L(1), L(2)=' SOIT ';
         CALL D (2, L, LEC);
         GET FILE (LEC) EDIT (CHAINE)(A(80));
         MOT=ELEM (CHAINE, 3);
         PUT EDIT(MOT) (SKIP, A);
         N=15;
         MOT=ELEM (CHAINE, N);
         PUT EDIT (MOT) (SKIP, A);
      D:PROC (P, MOT, X);
                /* PHRASES OU IL EXISTE P MOTS DONNES DANS L'ORDRE */
         DCL X FILE INPUT, MOT (P) CHAR (*), PH CHAR (72) VAR, I,J,
             1 PHRASE,
                2 REF CHAR (8),
                2 TEXTE CHAR (72) VAR;
         ON ENDFILE (X) GO TO FIN;
      LECTURE: GET FILE (X) EDIT (PHRASE) (A(8),A(72));
         PH=TEXTE;
         DO I=1 TO P;
                J=INDEX (PH, MOT (I));
                IF J THEN PH=SUBSTR (PH,J+2);
                     ELSE GO TO LECTURE;
         END;
         PUT LIST (PHRASE) SKIP;
         GO TO LECTURE;
      FIN: CLOSE FILE (X);
         RETURN;
      END D;
      B: PROCEDURE (P, MOT, X);
                /* PHRASES OU IL EXISTE P MOTS DONNES NON DANS L'ORDRE */
         DCL X FILE INPUT, MOT (P) CHAR (*), I, J,
             1 PHRASE,
                2 REF CHAR (8),
                2 TEXTE CHAR (72) VAR;
         ON ENDFILE (X) GO TO FIN;
      LECTURE: GET FILE (X) EDIT (PHRASE) (A(8),A(72));
         DO I=1 TO P;
                J=INDEX (TEXTE, MOT(I));
                IF J THEN ;
```

```
      APPEL: PROC OPTIONS (MAIN);

                        ELSE GO TO LECTURE;
         END;
         PUT LIST (PHRASE) SKIP;
         GO TO LECTURE;
      FIN: CLOSE FILE (X); RETURN;
      END B;
      C: PROCEDURE (MOT, LISTE, P, X);
                /* PHRASES OU IL EXISTE UN MOT DONNE ET UN MOT D'UNE LISTE*/
         DCL MOT CHAR (*), LISTE (P) CHAR (*), X FILE INPUT, I, J,
             1 PHRASE,
                2 REF CHAR (8),
                2 TEXTE CHAR (72);
         ON ENDFILE (X) GO TO FIN;
      LECTURE: GET FILE (X) EDIT (PHRASE) (A(8),A(72));
         J=INDEX (TEXTE, MOT);
         IF ¬J THEN GO TO LECTURE;
         DO I=1 TO P;
                J=INDEX (TEXTE, LISTE (I));
                IF J THEN DO; PUT LIST (PHRASE) SKIP;
                               GO TO LECTURE;
                          END;
         END;
         GO TO LECTURE;
      FIN: CLOSE FILE (X); RETURN;
      END C;
      E: PROCEDURE (L1, N1, L2, N2, X);
                /* PHRASES OU IL EXISTE UN MOT D'UNE LISTE ET UN MOT D'UNE
                   AUTRE LISTE */
         DCL X FILE INPUT, L1 (N1) CHAR (*), L2 (N2) CHAR (*), I1, J1, I2,J2,
             1 PHRASE,
                2 REF CHAR (8),
                2 TEXTE CHAR (72) VAR;
         ON ENDFILE (X) GO TO FIN;
         LECTURE: GET FILE (X) EDIT (PHRASE) (A(8), A(72));
                DO I1=1 TO N1;
                   J1=INDEX (TEXTE, L1(I1));
                   IF J1 THEN DO;
                   DO I2=1 TO N2;
                      J2= INDEX (TEXTE, L2(N2));
                      IF J2 THEN DO; PUT LIST (PHRASE) SKIP;
                                     GO TO LECTURE;
                                END;
                   END;
                   GO TO LECTURE;
                              END;
                END;
                GO TO LECTURE;
```

```
PEL: PROC OPTIONS (MAIN);

      FIN: CLOSE FILE (X);
           RETURN;
   END E;
   ELEM: PROCEDURE (A,K)CHAR (15);
              /* RECHERCHE DANS UNE PHRASE D'UN MOT DE RANG DONNE */
      DCL A CHAR (*),(PH,RESTE) CHAR (500) STATIC INITIAL (' '),
          MAX STATIC INITIAL (0), AIG(3) LABEL, I,J,K ;
      IF K <= 0 | K > RANG(A) THEN  /* RANG INCORRECT */
              DO ; PUT EDIT ('LE RANG DEMANDE EST INCORRECT' || K)(A);
                   RETURN (' ');
              END;
      IF A ¬= PH THEN /* CHANGEMENT DE PHRASE */
              DO; PH=A;
                  MAX=0;
                  GO TO AIG(1);
              END;
      I=SIGN (K-MAX)+2;
      GO TO AIG (I);
   AIG(1): J=K-1;
           GO TO E;
   AIG(2): RETURN(SUBSTR (RESTE, 1, INDEX (RESTE, ' ')-1));
   AIG(3): A= RESTE;
           J= K-MAX;
   E: DO I=1 TO J;
           A=SUBSTR (A, INDEX(A, ' ')+1);
   END E;
      IF K > MAX THEN
           DO; MAX=K;
               RESTE=A;
           END;
      RETURN(SUBSTR(A, 1, INDEX(A, ' ')-1));
   END ELEM;
   RANG: PROCEDURE (X)BINARY FIXED ;
              /* CALCUL DU NOMBRE DE MOTS D'UNE PHRASE */
      DCL X CHAR (*), I;
      I=0;
      DO WHILE (X¬='.');
              X=SUBSTR (X, INDEX (X, ' ')+1);
              I=I+1;
      END;
      RETURN (I);
   END RANG;
   END APPEL;
```

III - PROBLEME ELEMENTAIRE D'EDITION AVEC JUSTIFICATION

Ce problème n'est qu'une approche du problème concrêt de l'édition.

On appelle mot une suite de lettres

séparateur soit un blanc

soit un signe de ponctuation (. ! ? , ; ') suivi d'un blanc

texte une suite finie de mots, chacun étant suivi d'un séparateur.

Problème : On lit un texte qui a été perforé sur cartes de façon

- qu'aucun mot ne soit coupé (des colonnes restant éventuellement vides dans la partie droite de la carte).
- qu'il n'y ait jamais de séparateur en première colonne (le blanc suivant soit un mot se terminant en colonne 80, soit un signe de ponctuation se trouvant également en colonne 80 étant supprimé).

On demande d'imprimer le texte en lignes de 100 caractères de façon que le premier et le dernier caractère de chaque ligne (sauf la dernière) ne soit pas un blanc. Pour cela on pourra :

- couper les mots selon les règles de césure indiquées ci-dessous.
- remplacer dans la ligne un blanc séparant 2 mots par 2 blancs.
- supprimer le blanc suivant un signe de ponctuation en fin de ligne.

et ce de façon qu'il y ait toujours le maximum de lettres dans une ligne. Ainsi la ligne (de 23 caractères) :

ET QUAND IL PLEUT A PA-

sera préférée à

ET QUAND IL PLEUT A

qui a moins de lettres.

Règles de césure :

- Quand on coupe un mot, on met un tiret (-) à la fin de la ligne
- On ne coupe jamais entre l'avant-dernière et la dernière lettre d'un mot.
- On peut couper un mot.

i) entre 2 lettres identiques consécutives

ii) après une voyelle si elle est suivie d'une consonne puis d'une voyelle

iii) après une voyelle si elle est suivie de ER ou EUR...

iv) après l'une des lettres R,S,N,C,X si elle est suivie d'une consonne.

v) avant l'une des lettres B,C,D,F,G,J,K,P,Q,T,V,W,X,Z.

La priorité des cas allant en décroissant de i à v (par ex. on coupera EX-TERIEUR et non E-XTERIEUR).

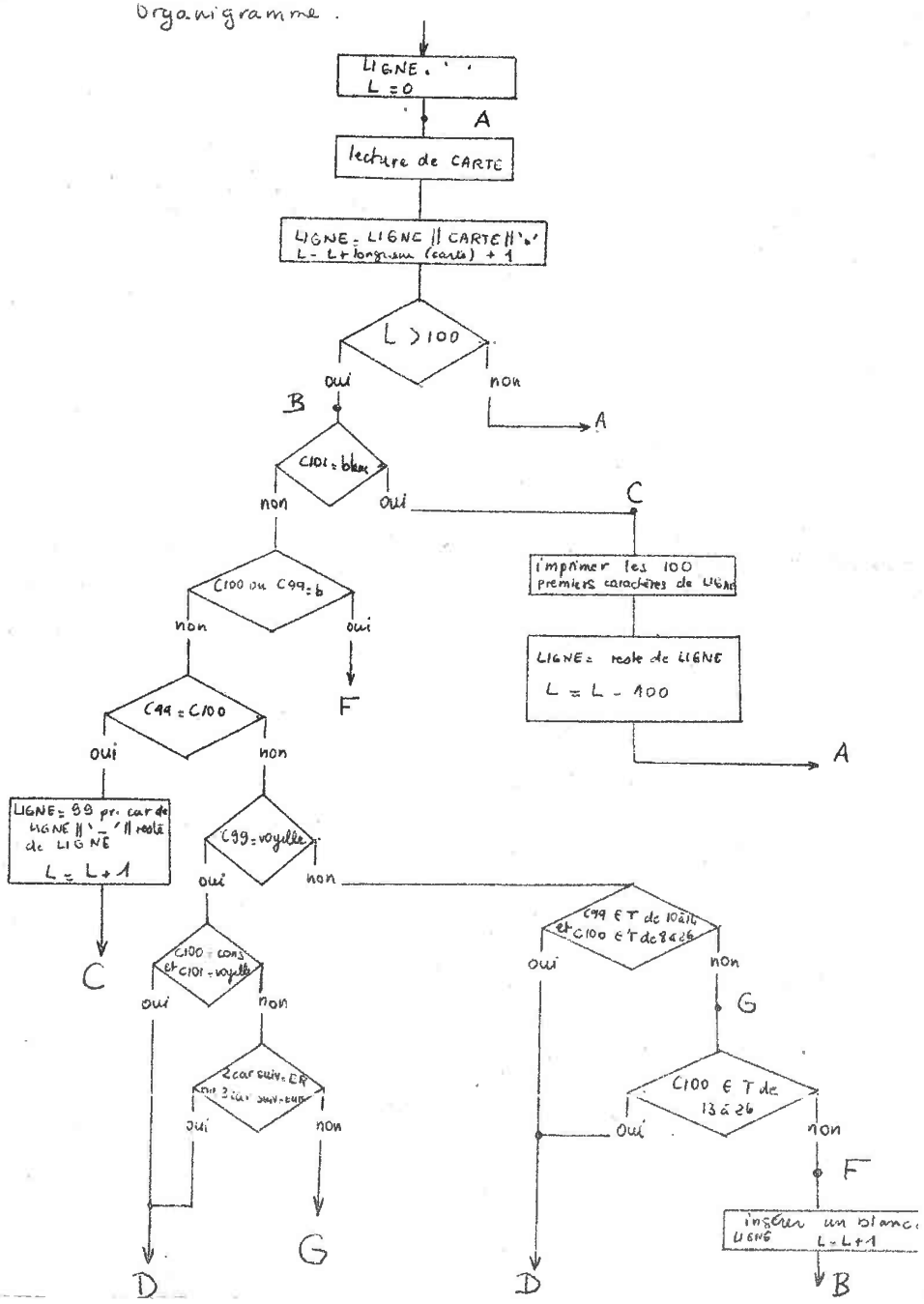
N.B. N'attacher aucune importance à la validité de ces règles, ce qui compte c'est le genre des règles que l'on pourra avoir à traiter en pratique

Tableau T :

AEIOUYHLMRSNXCBDFGJKPQTVWZ

Ci est le i^{ème} caractère de la chaine LIGNE : C99,C100

Organigramme.



```
DITION: PROCEDURE OPTIONS (MAIN);

EDITION: PROCEDURE OPTIONS (MAIN);
         /*EDITION D'UN TEXTE A RAISON DE 100 CARACTERES PAR LIGNE*/
         DECLARE (LIGNE              , SLIGNE) CHAR (181) VAR, J, L,
                 CARTE CHAR (80) VAR, I(26) CHAR (1), SYSIN FILE INPUT,
                 (C99, C100) CHAR (1);
         L=0; LIGNE=(0)'A';
         ON ENDFILE (SYSIN) GO TO FIN;
         GET EDIT (I) (26 (A(1)));
    A:   GET FILE (SYSIN) EDIT (CARTE) (A(80));
         LIGNE=LIGNE||CARTE||' ';
         L=L+LENGTH(CARTE)+1;
         IF L->=100 THEN GO TO A;
    B:   IF SUBSTR (LIGNE, 101, 1)=' ' THEN
    C:   DO; PUT EDIT (SUBSTR (LIGNE, 1, 100))(SKIP, A(100));
             LIGNE=SUBSTR (LIGNE, 101);
             L=L-100;
             GO TO A;
         END C;
         C99=SUBSTR (LIGNE, 99, 1);
         C100=SUBSTR (LIGNE, 100, 1);
         IF C99=' '|C100=' ' THEN GO TO F;
         IF C99=C100 THEN
    D:   DO; LIGNE=SUBSTR (LIGNE, 1, 99) || '-' || SUBSTR (LIGNE, 100);
             L=L+1;
             GO TO C;
         END D;
         DO J=1 TO 6; IF C99=I(J) THEN GO TO E; END;
         DO J=10 TO 14; IF C99=I(J) THEN
                 DO K=8 TO 26; IF C100=I(K) THEN GO TO D; END;END;END;
    G:   DO J=13 TO 26; IF C100=I(J) THEN GO TO D; END G;
    F:   SLIGNE=LIGNE;
         DO WHILE (INDEX (SLIGNE,'  '));
                 SLIGNE=SUBSTR (SLIGNE, INDEX (SLIGNE, '  ')+2);
         END;
         SLIGNE=SUBSTR (SLIGNE, INDEX (SLIGNE, ' ')+1);
         LIGNE=SUBSTR (LIGNE, 1, L-LENGTH (SLIGNE)) ||' ' ||SLIGNE;
         L=L+1;
         GO TO B;
    E:   DO J=7 TO 26; IF C100=I(J) THEN
                 DO K=1 TO 6; IF SUBSTR (LIGNE, 101, 1)=I(K) THEN GO TO D;
                 END;
         END E;
         IF SUBSTR (LIGNE, 100, 2)='ER' | SUBSTR (LIGNE, 100, 3)='EUR'
                 THEN GO TO D;
                 ELSE GO TO G;
    FIN: PUT EDIT (LIGNE)(SKIP, A); CLOSE FILE (SYSIN);
END EDITION;
```

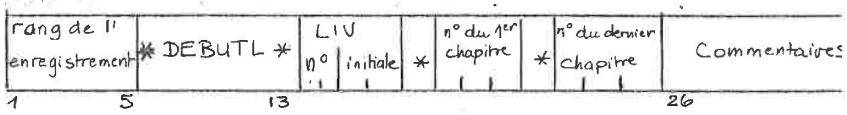

IV - VERIFICATION DE L'ORTHOGRAPHE FORMELLE D'UN TEXTE HEBREU CODE.

Ce programme vérifie la séquence des enregistrements, la validité des caractères du texte, l'obéissance à un certain nombre de règles du fichier LEC et crée un texte sur bande. Les messages d'erreur ainsi que les enregistrements erronés sont sortis sur imprimante.

I - Fichier LEC :

Le fichier LEC comprend des blocs de 20 enregistrements de 85 caractères. On distingue les enregistrements suivants :

1 - Enregistrement de début de livre : LV



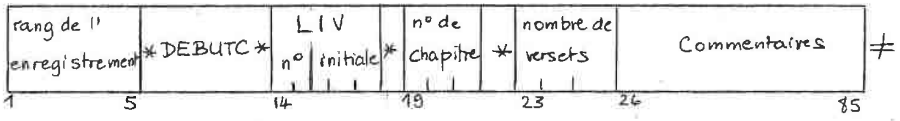
Placé avant les enregistrements du texte du livre, cet enregistremer donne les caractéristiques de l'ouvrage : numéro attribué au livre et abréviations du nom du livre, numéros du 1er et du dernier chapitre.

NUMEROS & ABREVIATIONS des LIVRES

Genèse	01 GN	Nahum	21 NH
Exode	02 EX	Habaquq	22 HB
Lévitique	03 LV	Sophonie	23 ZE
Nombres	04 NB	Haggée	24 HG
Deutéronome	05 DT	Zacharie	25 GK
		Malachie	26 ML

Josué	06 JS		
Juges	07 JU	Psaumes	27 PS
I Samuel	08 1S	Job	28 JB
II Samuel	09 2S	Proverbes	29 PR
I Rois	10 1R		
II Rois	11 2R	Ruth	30 RT
		Cantique	31 CT
Isaïe	12 IS	Qohélet	32 EC
Jérémie	13 JR	Lamentations	33 LM
Ezechiel	14 EZ	Esther	34 ES
Osée	15 ØS	Daniel	35 DN
Joël	16 JL	Ezra	36 EA
Amos	17 AM	Nehemie	37 NE
Obadiah	18 ØB	I Chroniques	38 1C
Jonas	19 JN	II Chroniques	39 2C
Michée	20 MI		

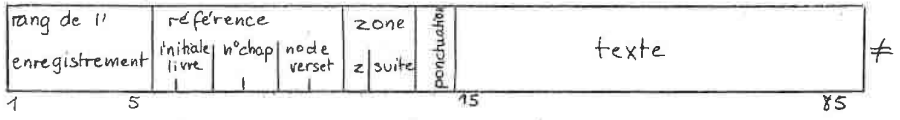
2 - Enregistrement de début de chapitre : C



L'enregistrement C donne le N° du chapitre étudié et le nombre de versets contenus dans le chapitre.

3 - Enregistrement du texte : T

Le texte hébreu est codifié sur 5 zones A, B, C, D, E, les 5 enregistrements correspondants à chaque partie du texte sont groupés en un tableau T de 5 élèments



Le texte est composé d'un nombre entier de mots, la partie du texte qui suit le dernier séparateur est vide.

L'enregistrement T comprend le rang de l'enregistrement, une partie référence, une partie zone qui indique le nom de la zone et le numér de séquence dans le verset (un verset comprend au maximum 6 enregistrements de chaque type), la ponctuation de début d'enregistremen et le texte. Les zones B, C, A reproduisent le texte étudié, elles sont codifiées sur 2 positions par caractère.

3.1. Zone B ou zone des consonnes

Cette zone contient les consonnes et les séparateurs de mots.

On trouve les codes suivants :

1ère position : consonnes, les chiffres 6 et 3 ou le caractère %
les séparateurs , . 1 0 - $

2ème position : les caractères *
les séparateurs , 1 0 - $

3.2. Zone C ou zone des voyelles

1ère position : voyelles , ou 0 ou $

2ème position : ou 0 ou $

3.3. Zone A ou zone de cantilation

C'est une zone numérique à 2 positions où l'on peut trouver les nombres suivants : de 00 à 04
"12 à 49
90 à 99

Ces codes obéissent aux règles suivantes :

- Si 00 apparaît dans A, il est aussi dans C, A, D et E.
- tout mot, dont la zone de cantilation est vide, est nécessairement suivi d'un tiret qui le lie au mot suivant.
- si le code est un nombre de 01 à 04 sur un mot de 2 lettres, le mot est obligatoirement suivi d'un tiret.

- si le code est 12, le mot correspondant est le dernier du verset.
- si le code est 21, le code du dernier caractère du mot est aussi 21, ou le caractère correspondant est le dernier.
- si le code 99 est en fin de mot, le code du caractère précédent est 40 et le séparateur suivant 1.
- si le code 45 apparait dans un mot, le mot suivant contient 25 et inversement.

3.4. Zone D ou zone grammaticale,

Le code relatif à un mot occupe les 2 premières positions du mot :

D	dieu	la fin du mot est vide
DV	divinité	
ET	nom ethnique	
HB	hébraïcisme	
IT	interjection	
MO	mois	
NL	nom de lieu	
NP	nom propre	
P	particule isolée	
PP	pronom personnel	
R*		
S*	relatif lié au nom	
SB	substantif	
TD	temple de Dieu	
TV	temple des divinités	
/	indication de direction	
A	article	la fin du mot contient des caractères.
F	copule	
I	interrogatif	
PA	particule + article	
P*	particule construite	
NB	participe	

OR	nombre ordinal	la fin du mot contient au moins un chiffre.
NM	nombre cardinal	

De plus si un séparateur de la zone D est $, le mot suivant est vide en zone D,

3.5. Zone E :

Elle permet d'indiquer les modifications à apporter à l'écriture de la zone B,

La codification est faite sur 2 positions et les caractères qui peuvent apparaître sont ceux de la zone B.

4 - Enregistrement de fin de chapitre F,C,

5 - Enregistrement de fin de livre F.L.

6 - Enregistrement CYCL

Peut se trouver en fin de verset,

IV - MESSAGES D'ERREUR : Si une ou plusieurs erreurs sont détectées

dans un des 5 enregistrements d'une partie de texte, tous les messages d'erreur relatifs aux 5 enregistrements apparaissent en séquence. Ils sont suivis du listing des 5 enregistrements.

Messages d'erreur correspondant au texte qui suit

```
Référence  B 1
           C 1
           A 1
           D 1
           E 1
```

1 85

Messages d'erreur correspondant au texte qui suit

```
Référence  B 4
           C 4
           A 4
           D 4
           E 4
```

1 85

Forme des messages d'erreur

ERREUR DE CONSONNE : 20ième CARACTERE DE B1, 20ième CARACTERE DE L'ENREGISTREMENT.

Différents messages d'erreur :

consonne pour les zones B et E

cantilation pour la zone A

voyelle " C

code " D

séparateur pour toutes les zones.

V - CREATION DE LA BANDE VERIFIEE : GELILAH

La bande GELILAH est créée à partir du fichier de sortie SØR, elle comprend des blocs de 4 enregistrements de 510 caractères.

On distingue les enregistrements suivants :

1) Enregistrement de début de livre :

n° du livre	0 0 0	0 0 0	0	*DEBUTL*		*		*		*commentaires
1	1 1	1 1			1 1		1 1		1 1	

2) Enregistrement de début de chapitre :

n° du livre	n° chap	0 0 0	0	*DEBUTC*	n°	*		*		*commentaires
1	1 1	1 1			1 1		1 1		1 1	

3° Table des séparateurs :

Les séparateurs sont recopiés dans les zones A, C, D et E; leur posit est repérée à partir du début du verset.

Cette table donne le nombre de mots dans le verset et la place des séparateurs.

n° livre	n° chap	n° verse	0	0000	nb de mots	1° sép	2° sép	3° sép	. . . .
1	1 1	1 1			1	1 1	1 1	1 1	

4°) Enregistrement d'une zone de verset :

n° livre	n° chap	n° verset	n° zone		texte
1	1 1	1 1			

La zone n° 1 correspond à la zone A

. . . . . .

" n° 5 " E

5°) Enregistrement de fin de chapitre :

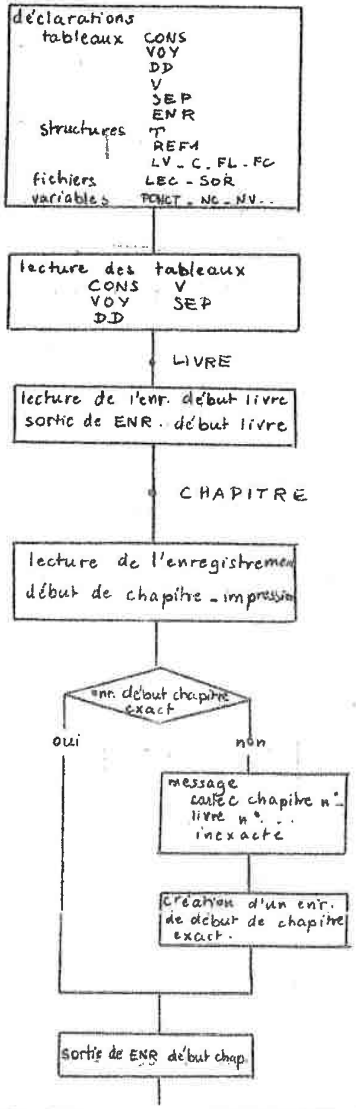
n° livre	n° chap	9 9 9	0	

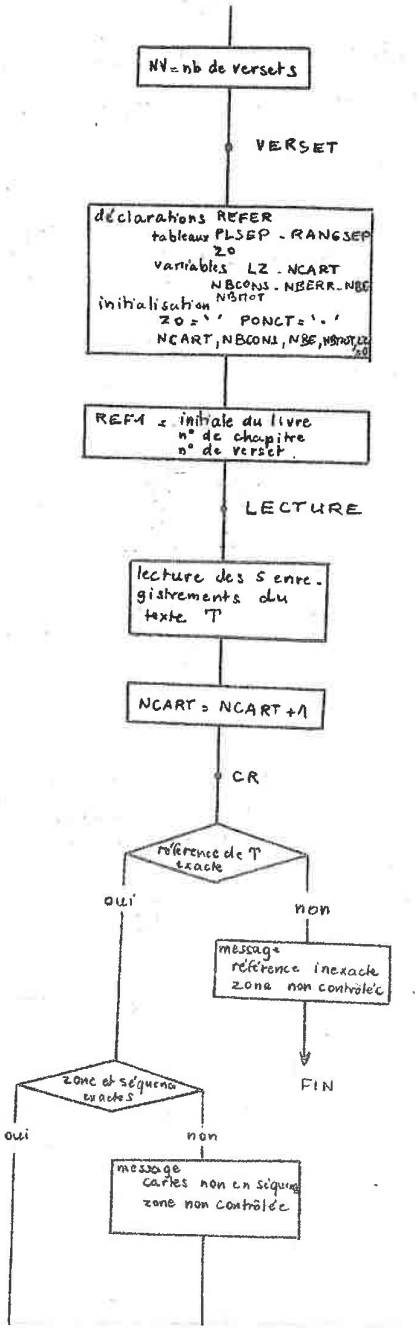
6°) Enregistrement de fin de livre :

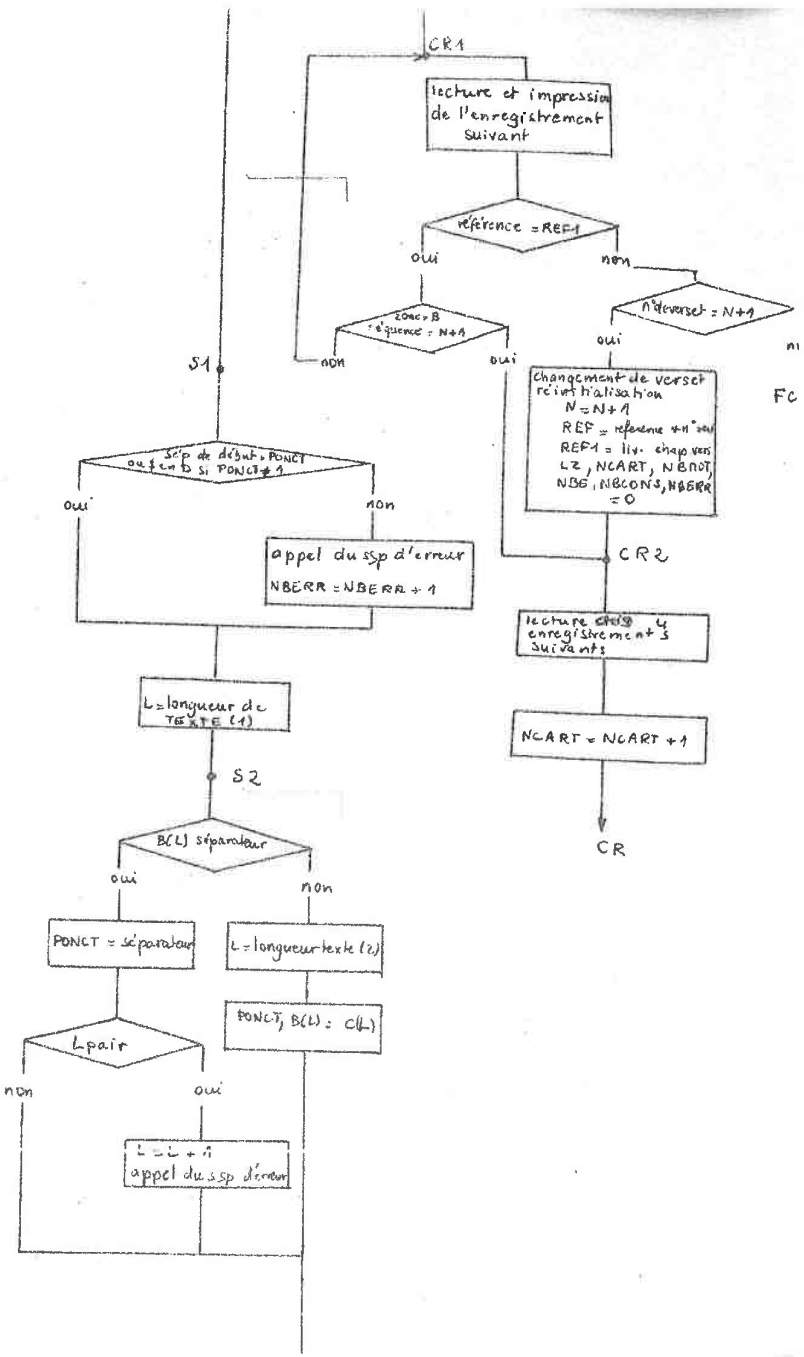
n° livre	9 9 9	9 9 9	0	

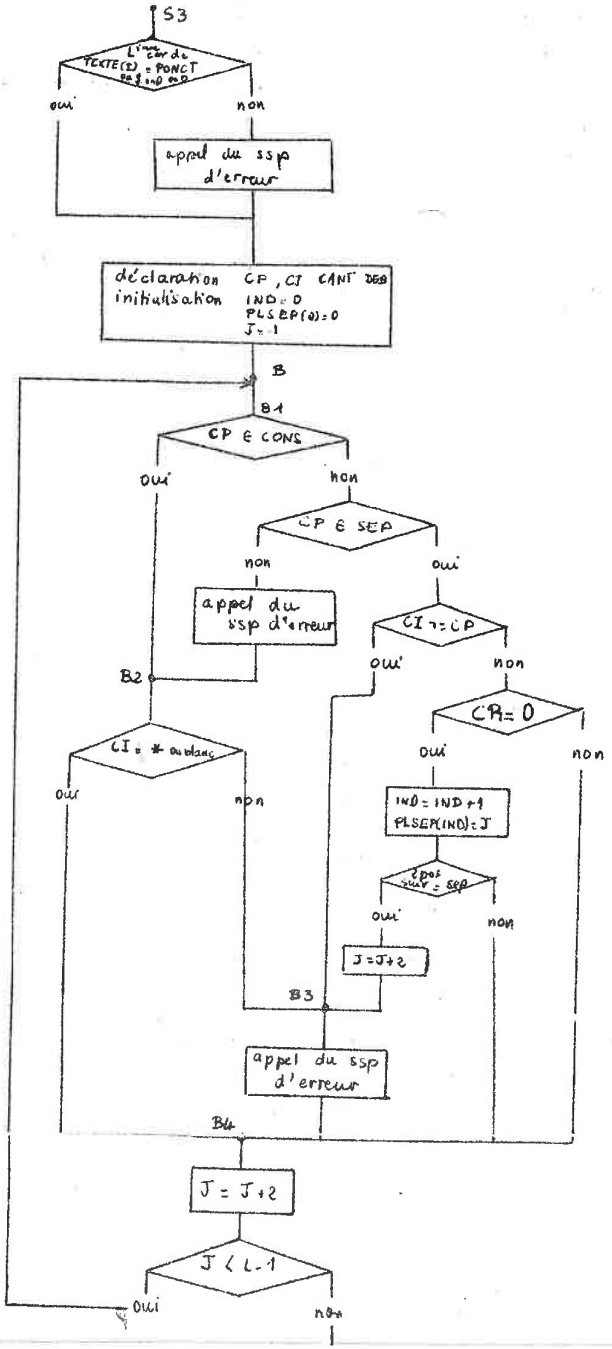
II

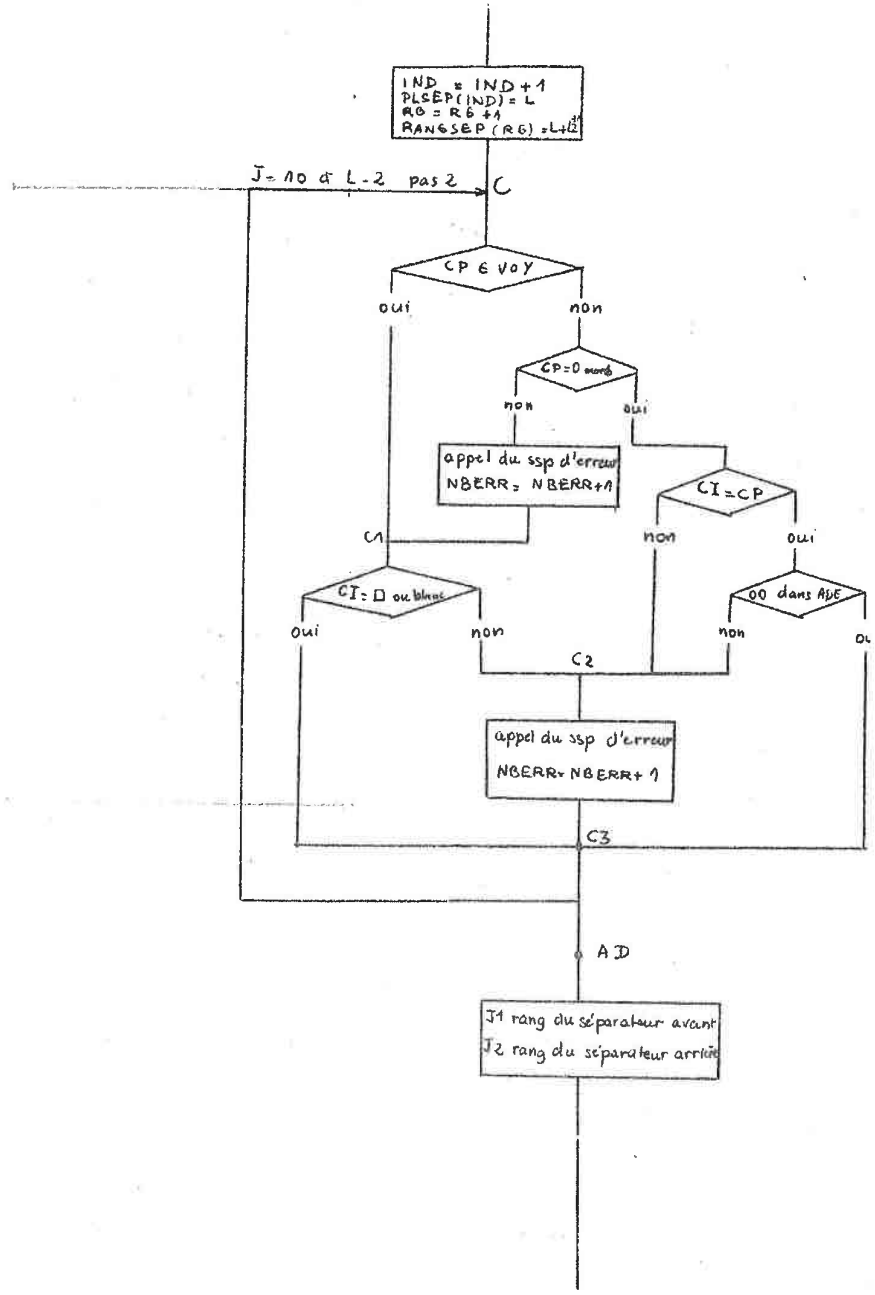
ORGANIGRAMME DE VERIFICATION DE L' ORTHOGRAPHE FORMELLE.

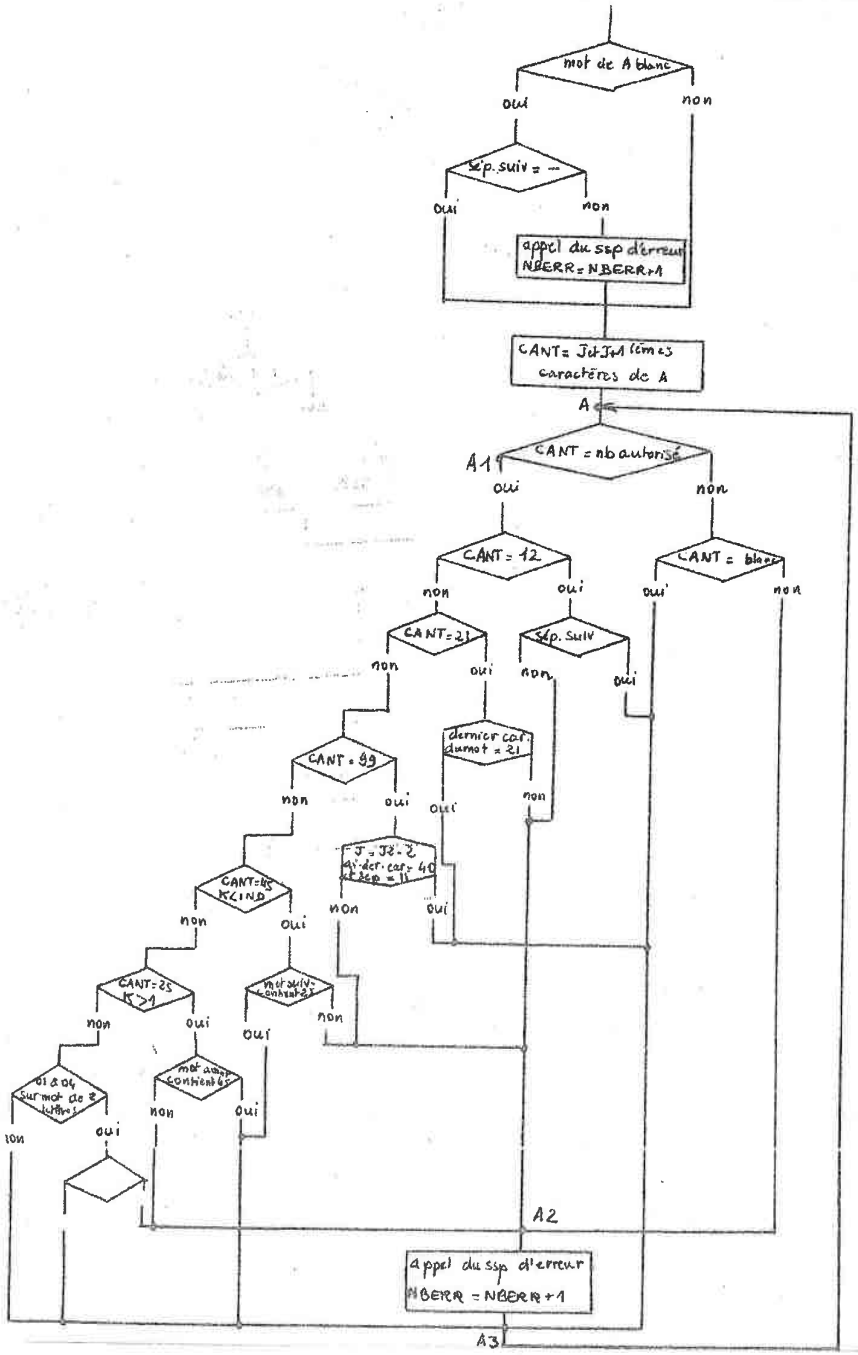


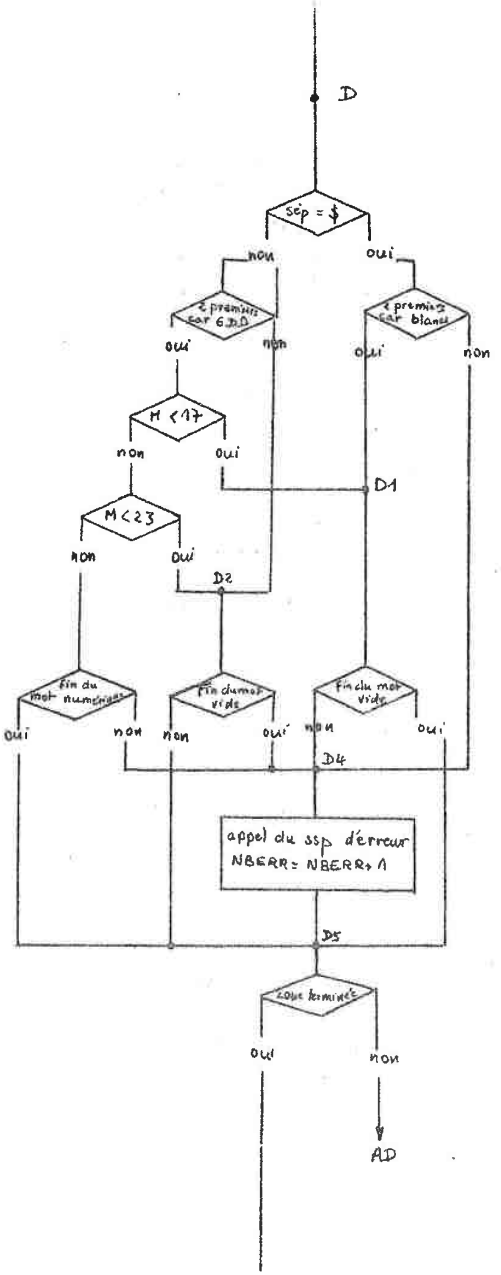

NV = nb de versets
VERSET
déclarations REFER
tableaux PLSEP - RANGSEP
Z0
variables LZ - NCART
NBCONS - NBERR - NBE
NBMOT
initialisation
Z0 = ' ' PONCT = '.'
NCART, NBCONS, NBE, NBMOT, LZ = 0
REF1 = initiale du livre
n° de chapitre
n° de verset
LECTURE
lecture des 5 enregistrements du texte T
NCART = NCART + 1
CR
référence de T exacte
oui
non
message référence inexacte zone non contrôlée
FIN
zone et séquence exactes
oui
non
message cartes non en séquence zone non contrôlée

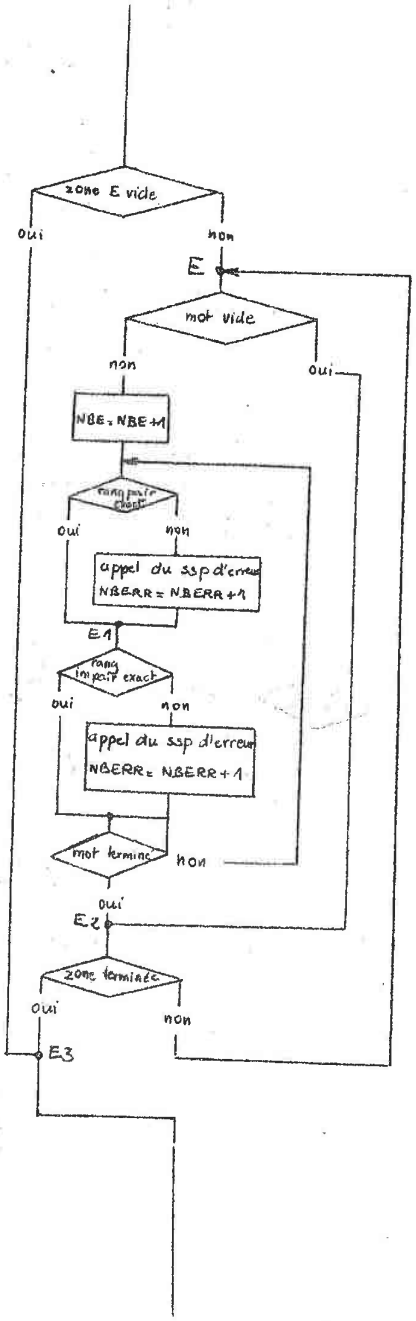

CR1
lecture et impression de l'enregistrement suivant
référence = REF1
oui
non
zone > B séquence = N+1
n° verset = N+1
oui
non
oui
changement de verset réinitialisation N = N+1 REF = référence + n° verset REF1 = liv. chap vers LZ, NCART, NBMOT, NBE, NBCONS, NBERR = 0
S1
CR2
ou $ en B si PONCT ≠ 1
oui
non
appel du ssp d'erreur NBERR = NBERR + 1
lecture chrg 4 enregistrements suivants
L = longueur de TEXTE (1)
NCART = NCART + 1
S2
CR
B(L) séparateur
oui
non
PONCT = séparateur
L = longueur texte (2)
PONCT, B(L) : C(L)
L pair
non
oui
L = L + 1
appel du ssp d'erreur

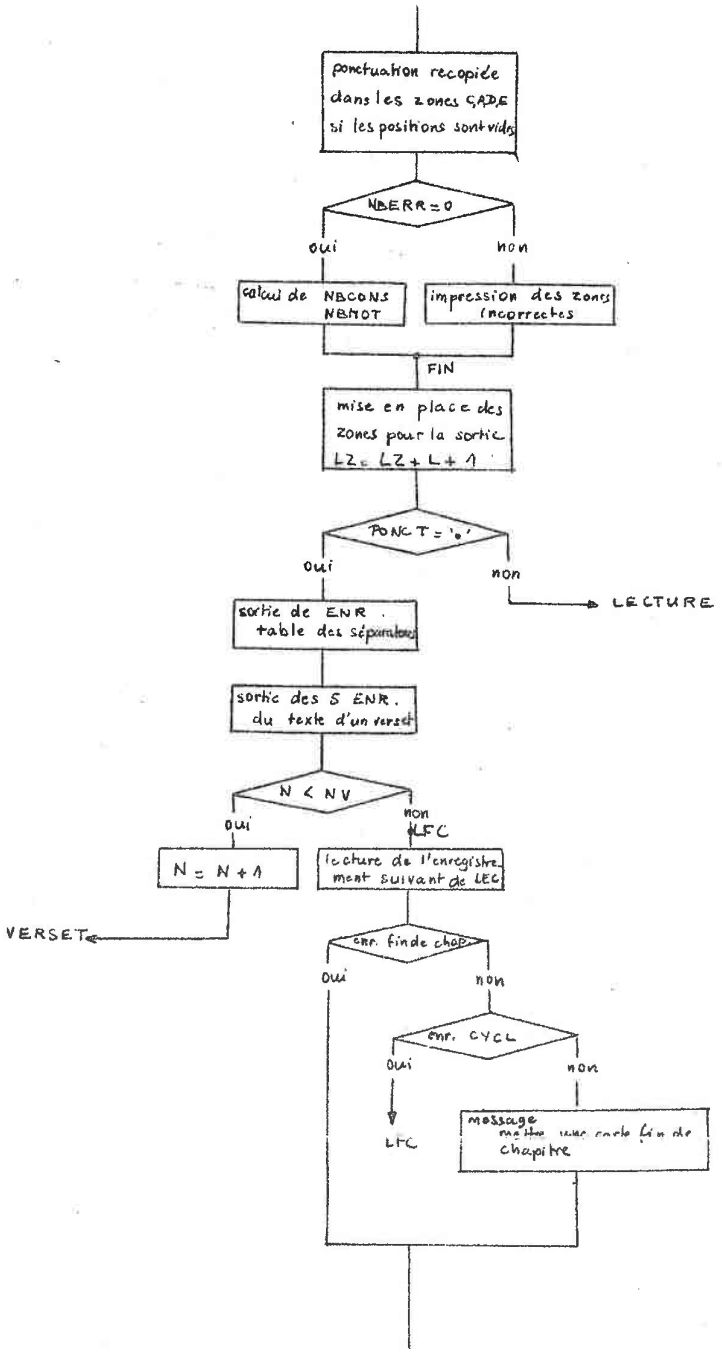

S3
L'avant dernier caractère de TEXTE(I) = PONCT ou $ ou 0
oui
non
appel du ssp d'erreur
déclaration CP, CI CAR DEB
initialisation IND = 0
PLSEP(0) = 0
J = 1
B
B1
CP ∈ CONS
oui
non
CP ∈ SEP
non
oui
appel du ssp d'erreur
CI = CP
B2
oui
non
CP = 0
CI = * ou blanc
oui
non
oui
non
IND = IND + 1
PLSEP(IND) = J
Rang suivant = SEP
oui
non
J = J + 2
B3
appel du ssp d'erreur
B4
J = J + 2
J < L - 1
oui
non

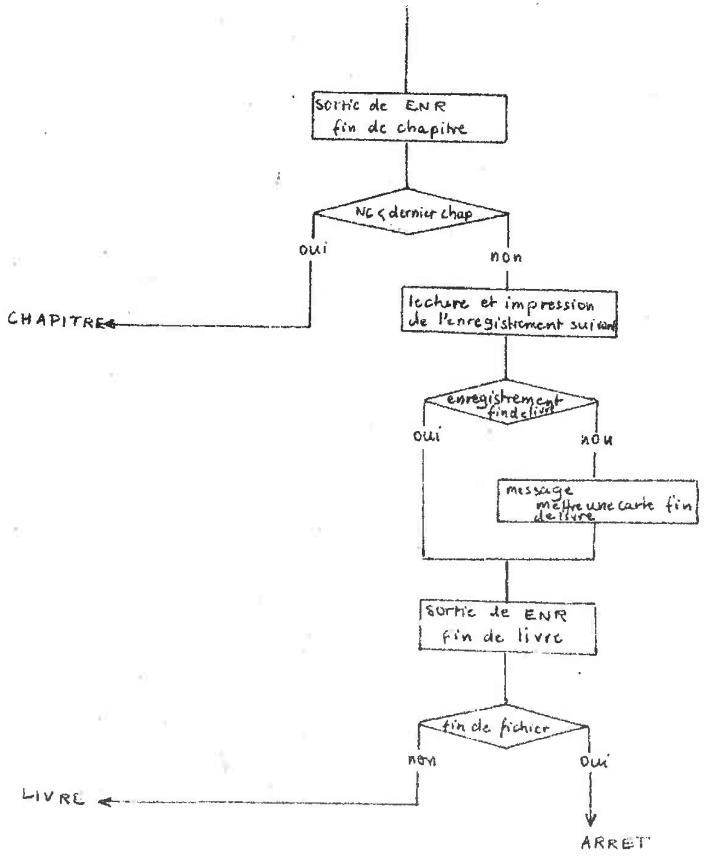

IND = IND + 1
PLSEP(IND) = L
RG = RG + 1
RANGSEP(RG) = L + 1
J = 10 à L - 2 pas 2
C
CP ∈ VOY
oui
non
CP = 0 mot
non
oui
appel du ssp d'erreur
NBERR = NBERR + 1
CI = CP
C1
non
oui
CI = 0 ou blanc
oui
non
00 dans ADE
non
C2
appel du ssp d'erreur
NBERR = NBERR + 1
C3
AD
J1 rang du séparateur avant
J2 rang du séparateur arrière


mot de A blanc
oui
non
Sép. suiv = --
oui
non
appel du ssp d'erreur
NBERR = NBERR+1
CANT = Jet Jet 15 premiers caractères de A
A
A1
CANT = nb autorisé
oui
non
CANT = 12
CANT = blanc
non
oui
oui
non
CANT = 21
Sép. suiv
non
oui
non
oui
CANT = 99
dernier car. du mot = 21
non
oui
oui
non
J = J1 - 2
si dern. car = 40 et Sép = 11
CANT = 45 K LIND
non
oui
non
oui
CANT = 25 K > 1
mot suiv. contient 2 ?
non
oui
non
oui
01 & 04 sur mot de 2 lettres
mot avec contenu 6?
non
oui
non
oui
A2
appel du ssp d'erreur
NBERR = NBERR+1
A3


D
Sép = $
non
oui
2 premiers car G ou D
2 premiers car blancs
oui
non
oui
non
M < 17
non
oui
D1
M < 23
non
oui
D2
fin du mot numérique
fin du mot vide
fin du mot vide
oui
non
non
oui
non
oui
D4
appel du ssp d'erreur
NBERR = NBERR+1
D5
zone terminée
oui
non
AD


zone E vide
oui
non
E
mot vide
non
oui
NBE = NBE+1
rang pair exact
oui
non
appel du ssp d'erreur
NBERR = NBERR+1
E1
rang impair exact
oui
non
appel du ssp d'erreur
NBERR = NBERR+1
mot terminé
non
oui
E2
zone terminée
oui
non
E3


ponctuation recopiée dans les zones CADE si les positions sont vides
NBERR = 0
oui
non
calcul de NBCONS NBMOT
impression des zones incorrectes
FIN
mise en place des zones pour la sortie LZ = LZ + L + 1
PONCT = '.'
oui
non
LECTURE
sortie de ENR. table des séparateurs
sortie des 5 ENR. du texte d'un verset
N < NV
oui
non
LFC
N = N + 1
lecture de l'enregistrement suivant de LEC
VERSET
enr. fin de chap.
oui
non
enr. CYCL
oui
non
LFC
message mettre une carte fin de chapitre



VERIF PROCEDURE OPTIONS(MAIN)

```
VERIF PROCEDURE OPTIONS(MAIN)
/*VERIFICATION DE L'ORTHOGRAPHE FORMELLE*/
    DECLARE (CONS (23), VOY (8), V (5), SEP (6)) CHAR (1),
            (DD (24), ABR_L (39)) CHAR (2),
     1 FL ,2 RANG PIC '99999', 2 FINL CHAR (6), 2 COM CHAR (75),
     1 FC, 2 RANG PIC '99999', 2 FINC CHAR (6), 2 COM CHAR (75),
           1 T(5), 2 RANG  PIC '99999' ,
                   2 REF,3 LIV CHAR(2),3 CH PIC '99',3 VE PIC '99
                   2 ZONE, 3 Z CHAR (1), 3 SUITE PIC '9',
                   2 P CHAR (1),
                   2 TEXTE CHAR (71) VARYING,
                   2 FE CHAR (1),
        1 REF1 LIKE T.REF,
      1 LV,2 RANG PIC '99999',
                2 DEB CHAR (8),
                2 LIV, 3 NO PIC '99', 3 INI CHAR (2),
                2 AST1 CHAR (1) INITIAL ('*'),
                2 CHAP PIC '999',
                2 AST2 CHAR (1) INITIAL ('*'),
                2 NB PIC '999',
                2 COMMENTAIRES CHAR (61),
        1 C LIKE LV, PCNCT CHAR (1), (N,NV,J) FIXED (2),
ENR CHAR (510), NC FIXED (3),
          LEC FILE INPUT, SOR FILE STREAM OUTPUT;
 GET LIST (CONS, VOY, DD, V, SEP);
    ON ENDFILE (LEC) GO TO FINISH;
    F1: FORMAT (F(5), A(8), F(2), A(2), A(1), F(3), A(1), F(3),
    A(61));
    F2: FORMAT (F(5), A(2), F(2), F(2), A(1), F(1), A(71), A(1));
    F3: FORMAT (F(5), A(6), A(75));
    GET FILE (LEC) EDIT (LV) (R (F1));
LIVRE: ENR=LV.NO||'0000000*DEBUTL*'||LV.NO||LV.INI||'*'||LV.CHAP||
LV.NB||LV.COMMENTAIRES;
    PUT FILE (SOR) EDIT (ENR) (A);
CHAPITRE:DO NC=LV.CHAP TO LV.NB;
    GET FILE (LEC) EDIT (C) (R (F1)) COPY;
IF  (C.DEB='*DEBUTC*' & C.NO=LV.NO & C.INI=LV.INI & C.CHAP=NC) THEN
     DO; DISPLAY('CARTEC INEXACTE  CHAPITRE' ||NC||'LIVRE'||LV.NO);
       C.DEB='*DEBUTC*';
        C.LIV=LV.LIV;
      C.CHAP=NC;
   END;
ENR=LV.NO||C.CHAP||'0000*DEBUTC*'||C.NO||C.INI||'*'||C.CHAP||'*'||
C.NB||C.COMMENTAIRES;
    PUT FILE (SOR) EDIT (ENR) (A);
    NV=SUBSTR (C.NB, 2, 2),N=01;
VERSET: DO WHILE (N<=NV);
```

```
VERIF PROCEDURE OPTIONS(MAIN)

        DCL PLSEP(0:30) FIXED(2),REFER CHAR(14),NCART FIXED(1), (NBE,
        LZ,NBCONS,NBMOT,NBERR) FIXED(3),ZC(5) CHAR(5CC) VAR,
                  (RG INITIAL (0) ,RANGSEP (6C)) FIXED (3),
        I FIXED(1),(L,K,J1,J2,J3,J4,M,IND) FIXED(2);
         NCART,NBMOT,NBE,LZ,NBCONS=0;
         ZC=' ';
         PONCT='.';
        REF1.LIV=LV.INI;REF1.CH=SUBSTR(NC,2,2);REF1.VE=N;
         ON TRANSMIT (LEC) BEGIN;
              DO I=1 TO 4; T(I)=T(I+1); END;
              GET FILE (LEC) EDIT (T(5)) (F(5), A(2), F(2), F(2), A(1),
        F(1), A(71), A(1));
         END;
      LECTURE: GET FILE (LEC) EDIT (T) (5 R (F2));
         NBERR=0;
         NCART=NCART+1;
      CR: /*COMPARAISON DES REFERENCES*/
         DO I=1 TO 5;
        IF  (T(I).LIV=REF1.LIV & T(I).CH=REF1.CH & T(I).VE=REF1.VE) THEN
        DO; DISPLAY('REFERENCE INEXACTE, ZONE NON CONTROLEE');
                GO TO FIN;
             END;
        IF  (T(I).Z=V(I) & T(I).SUITE=NCART) THEN
             DO; DISPLAY ('CARTES NON EN SEQUENCE,ZONE NON CONTROLEE');
         GO TO CR1;
         END;
      END CR; GO TO S1;
      CR1: GET FILE (LEC) EDIT (T(1)) (R (F2)) COPY;
           IF T(1).LIV=REF1.LIV & T(1).CH=REF1.CH & T(1).VE=REF1.VE
           THEN IF  (T(1).Z='8' & T(1).SUITE
        =NCART+1) THEN GO TO CR1;ELSE GO TO CR2;
        IF T(1).VE =N+1 THEN DO; FINC=T(1).LIV||T(1).CH||T(1).VE ;
                                 GO TO FCHAP;
                            END;
         /*CHANGEMENT DE VERSET,REINITIALISATION*/
         REF1.VE,N=N+1;
         NCART,NBERR,NBMOT,NBE,NBCONS,LZ=0 ;
      CR2: GET FILE (LEC) EDIT((T(I) DO I=2 TO 5))(4 R (F2)) COPY;
         NCART=NCART+1;
         GO TO CR;
      S1: REFER='*'|| C.NO || C.INI ||'*'|| C.CHAP||'*'|| N ||'*';
         /*CONTROLE DE LA PONCTUATION EN DEBUT ET FIN DE CARTE*/
         DO I=1 TO 3, 5 ; IF T(I).P =PONCT THEN
              DO; NBERR=NBERR+1;
                  CALL ERREUR (9, V(I), NCART, 'SEPARATEUR', LZ+1);
              END;
         END;
```

```
VERIF PROCEDURE OPTIONS(MAIN)

         IF  (T(4).P=PONCT T(4).P='$'& PONCT ='1') THEN
         DO; NBERR=NBERR+1;
           CALL ERREUR (9, V(4), NCART, 'SEPARATEUR', LZ+1);
         END;
         L=LENGTH (TEXTE(1));
      S2: DO K=2 TO 6;
         IF SUBSTR (TEXTE(1), L, 1)=SEP (K) THEN
         DO; PONCT=SEP(K);
             IF L =(L/2)*2 THEN
             DO; CALL ERREUR (L+9, V(1), NCART, 'SEPARATEUR', LZ+L+1);
               NBERR=NBERR+1; L=L+1;
             END; GO TO S3;
         END; END S2;
         L=LENGTH (TEXTE(2));
         SUBSTR (TEXTE(1), L, 1), PONCT=SUBSTR (TEXTE(2), L, 1);
         CALL ERREUR (L+9, V(1), NCART, 'SEPARATEUR', LZ+L+1);
         NBERR=NBERR+1;
      S3: DO I=2 TO 5; IF SUBSTR (TEXTE(I), L, 1)=PONCT THEN ;
         ELSE IF (SUBSTR (TEXTE(I), L, 1)='0'&PONCT='-'|
                  SUBSTR (TEXTE(4), L, 1)='$'&PONCT ='1') THEN ;
         ELSE DO; NBERR=NBERR+1;
                  CALL ERREUR (L+9, V(I), NCART, 'SEPARATEUR', LZ+L+1);
              END;
      END S3;
         DCL (CP, CI) CHAR (1), CANT CHAR(2)  , DEP CHAR (2);
         IND=0; PLSEP(0)=0; J=1;
      8: DO WHILE (J<L-1); CP=SUBSTR (TEXTE(1), J, 1);
                           CI=SUBSTR (TEXTE(1), J+1, 1);
         B1: DO K=1 TO 23; IF CP=CONS (K) THEN GO TO B2; END B1;
         DO K=1 TO 5; IF CP=SEP(K) THEN IF CI =CP THEN GO TO B3; ELSE
                                        DO; IF K=1 THEN GO TO B4;
                                            IND=IND+1;
                                            PLSEP (IND)=J;
                  RG=RG+1;
                  RANGSEP (RG)=J+LZ+1;
                                            DO K=2 TO 5;
                                            IF SUBSTR (TEXTE(1), J+2, 2)
                                            SEP(K)||SEP(K)) THEN DO; J=J
                                            GO TO B3; END; END; GO TO B4
                                        END;
        END;
          CALL ERREUR (J+9, V(1), NCART, 'CONSONNE', LZ+J+1);
          NBERR=NBERR+1;
        B2: IF CI='*'|CI=' ' THEN GO TO B4;
        B3: CALL ERREUR (J+9, V(1), NCART, 'CONSONNE', LZ+J+2);
          NBERR=NBERR+1;
        B4: J=J+2;
```

```
VERIF PROCEDURE OPTIONS(MAIN)

        END B;
            IND=IND+1; PLSEP (IND)=L;
                       RG=RG+1;
                       RANGSEP (RG)=L+LZ+1;
        C4: DO J=1 TO L-2 BY 2;  CP=SUBSTR (TEXTE (2), J, 1);
                                 CI=SUBSTR (TEXTE (2), J+1, 1);
            DO K=1 TO 8; IF CP=VOY (K) THEN GO TO C1; END;
            IF CP='0' THEN DO  IF CI =CP THEN GO TO C2;
                               IF SUBSTR (TEXTE (3), J, 2)='00'&
                                  SUBSTR (TEXTE (4), J, 2)='00'&
                                  SUBSTR (TEXTE (5), J, 2)='00'  THEN GO TO C3;
                               ELSE GO TO C2;
                            END;
            CALL ERREUR (J+9, V(2), NCART, 'VOYELLE', LZ+J+1);
            NBERR=NBERR+1;
          C1: IF CI=' '|CI='0' THEN GO TO C3;
          C2: CALL ERREUR (J+9, V(2), NCART, 'VOYELLE', LZ+J+2);
            NBERR=NBERR+1;
        C3:END C4;
        AC: DO K=1 TO IND; J1= PLSEP (K-1)+2; J2=PLSEP (K);
        /*A*/
             IF SUBSTR(TEXTE(3),J1,J2-J1)=' '
            THEN IF SUBSTR (TEXTE (1), J2, 1) ='-' THEN
                 CALL ERREUR (J2+9, V(1), NCART, 'SEPARATEUR', LZ+J+1);
         A: DO J=J1 TO J2-2 BY 2; CANT=SUBSTR (TEXTE (3), J, 2);
                                  DO K=00 TO 04, 12 TO 49, 90 TO 99;
                                  IF CANT=K THEN GO TO A1;
                                  END;
                 IF CANT ='  ' THEN GO TO A2;
              A1: IF CANT=12 THEN
                    IF SUBSTR (TEXTE (1), J2, 1)='.' THEN GO TO A3;
                    ELSE GO TO A2;
                    IF CANT=21 THEN
                    IF SUBSTR (TEXTE (3), J2-2, 2)='21' THEN GO TO A3;
                    ELSE GO TO A2;
                    IF CANT=99 THEN
                    IF SUBSTR (TEXTE (3), J2-4, 2)='40' & J=J2-2 &
                       SUBSTR (TEXTE (1), J2, 1)='1'  THEN GO TO A3;
                    ELSE GO TO A2;
                    IF CANT=45 & K<IND THEN
                    DO; J3=PLSEP (K+1)-2;
                    DO J4=J2+2 TO J3 BY 2;
                    IF SUBSTR (TEXTE (3), J4, 2)='25' THEN GO TO A3;
                    END; GO TO A2;
                    END;
                    IF CANT=25 & K>1 THEN DO; J3=PLSEP (K-2)+2;
                    DO J4=J3 TO J1-4;

VERIF PROCEDURE OPTIONS(MAIN)

                    IF SUBSTR (TEXTE (3), J4, 2)='45' THEN GO TO A3;
                    END; GO TO A2;
                    END;
                    IF J2-J1  = 6 &(CANT=01|CANT=02|CANT=03|CANT=04) THEN
                    IF SUBSTR (TEXTE(1), J2, 1)='-' THEN GO TO A3; ELSE GO TO
                    ELSE GO TO A3;
                 A2: CALL ERREUR (J+9, V(3), NCART, 'CANTILATION', LZ+J+1);
                    NBERR=NBERR+1;
        A3: END A;
        /* C */
            DEB=SUBSTR (TEXTE(4), J1, 2);
            IF SUBSTR (TEXTE(4), J1-2, 1)='$' THEN
            IF DEB='  ' THEN GO TO D1; ELSE GO TO D4;
            DO M=1 TO 24;
            IF DB=CD(M) THEN DO;
                                  IF M<17 THEN GO TO D1;
                                IF M<23 THEN GO TO D2;
                                DO J=J1+2 TO J2;
                                DO K1=0 TO 9;
                                IF SUBSTR (TEXTE (4), J, 1)=K1|
                                   SUBSTR (TEXTE (4), J, 1)=' ' THEN GO
                                D3; END;
                                D3: END;
                             END;
            END;
            D2: J=J1+2; DO WHILE (SUBSTR(TEXTE (4), J, 1)=' ');
                        IF J=J2 THEN GO TO D4; ELSE J=J+1;
                        END;
                        GO TO D5;
            D1: J=J1+2; DO WHILE (SUBSTR (TEXTE (4), J, 1)=' ');
                        IF J=J2 THEN GO TO D5; ELSE J=J+1;
                        END;
            D4: CALL ERREUR (J1+9, V(4), NCART, 'CODE', LZ+J+1);
                NBERR=NBERR+1;
        D5: END AD;
             IF SUBSTR (TEXTE (5), 1, L-1)=' ' THEN GO TO E3;
        E: DO M=1 TO IND; J1=PLSEP (M-1)+2; J2=PLSEP (M)-2;
            IF SUBSTR (TEXTE (5), J1, J2-J1+2)=' ' THEN GO TO E2;
            NBE=NBE+1;
            DO J=J1 TO J2 BY 2; CP=SUBSTR (TEXTE (5), J, 1);
                                CI=SUBSTR (TEXTE (5), J+1, 1);
            DO K=1 TO 23; IF CP=CONS (K) THEN GO TO E1;
            END;
            IF (CP=' '|CP='0') THEN
            DO; CALL ERREUR (J+9, V(5), NCART, 'CONSONNE', LZ+J+1);
                NBERR=NBERR+1;
            END;
```

```
/ERIF PRCCEDURE OPTIONS(MAIN)

          E1: IF  (CI='0'|CI='-'|CI=' ') THEN
          DO; CALL ERREUR (J+9, V(5), NCART, 'CONSONNE', LZ+J+2);
              NBERR=NBERR+1;
          END;
          END;
       E2: END E;
       E3: /* ECRITURE DE LA PONCTUATION DANS LES ZONES C,A,D,E */
          DC K=1 TC IND-1; J=PLSEP(K); CP=SUBSTR (TEXTE (1), J, 1);
          DO I=2 TO 5;
          IF SUBSTR (TEXTE (I), J, 2)='  ' THEN SUBSTR (TEXTE(I), J, 2)=(CP||
          CP);
          ELSE IF CP='-'&SUBSTR (TEXTE(I), J, 2)='CO'|SUBSTR (TEXTE(4), J, 2)
               ='$$'& CP='1' THEN ;
               ELSE DO; CALL ERREUR (J+9, V(I), NCART, 'SEPARATEUR', LZ+J+
                    NBERR=NBERR+1;
                END;
          END;
       END E3.
       IF NBERR=0 THEN DO;
          NBMOT=NBMOT+IND;
          NBCONS=NBCONS+L/2 -INC+2; END;
       ELSE DO;
       PUT EDIT (('****.****'||I DO I=1 TO 8))(SKIP (3), COLUMN (6),8 A(10));
       PUT SKIP(3); DO I=1 TC 5;PUT EDIT(T(I))(R(F2)) SKIP;END;
       END;
       FIN: /* MISE EN PLACE DES ZONES POUR LA SORTIE */
                    ZO (1)=ZO (1) || P(3) || TEXTE (3);
                    ZO (2)=ZO (2) || P(1) || TEXTE (1);
                    ZO (3)=ZO (3) || P(2) ||TEXTE (2);
                    ZO (4)=ZO (4) || P(4) || TEXTE (4);
                    ZO (5)= ZO (5)|| P(5) || TEXTE (5);
                   LZ=LZ + L + L;
          PUT DATA (NBERR);
          IF PONCT  ='.' THEN GO TC LECTURE;
          ENR=LV.NO||NC||'0'||N||'0000'||NBMOT||NBCONS||NBE;
          DC I=1 TC RG;
               ENR=ENR||RANGSEP(I);
          END;
          PUT FILE (SOR) EDIT (ENR) (A);
              DO I=1 TO 5;
          ENR=LV.NO||NC||'0'||N||I||ZO(I);
          PUT FILE (SOR) EDIT (ENR) (A);
              END; N=N+1;
       END VERSET;
       LFC: GET FILE (LEC) EDIT (FC) (R (F3));
       FCHAP: IF FINC  = '*FINC*' THEN DO;
                        IF FINC='*CYCL*' THEN GO TO LFC;
```

VERIF PRCCEDURE OPTIONS(MAIN)

```
                        DISPLAY ('METTRE UNE CARTE FINC');
                        PUT EDIT (FC) (R (F3)) SKIP;
                              END;
          ENR=LV.NC|| NC||'9990';
          PUT FILE (SOR) EDIT (ENR) (A);
       END CHAPITRE;
          GET FILE (LEC) EDIT (FL) (R (F3)) COPY;
          IF FINL  = '*FINL*' THEN DISPLAY ('METTRE UNE CARTE FINL');
          ENR=LV.NC||'9999990';
          PUT FILE (SOR) EDIT (ENR) (A);
          GET FILE (LEC) EDIT (LV) (R(F1)) COPY ;
          GO TO LIVRE;
          FINISH: CLOSE FILE (LEC),FILE(SCR);
       ERREUR: PROCEDURE (X,Y, Z, T, J);
          DCL (X, Z, J) FIXED, Y   CHAR (1), T CHAR (*);
          PUT EDIT ('ERREUR DE', T, ' ', X, 'IEME CARACTERE DE ', Y  ,
          J, 'IEME CARACTERE DE L''ENREGISTREMENT')
          (SKIP (2), A, A (12), A, F(3), A, A(1), F(2), X(5), F(2), A);
          RETURN;
       END ERREUR;
       END VERIF;
```

```
      VERIF PROCEDURE OPTIONS(MAIN)

COMPILER DIAGNOSTICS.


NINGS.

IEM0227I          NO FILE/STRING OPTION SPECIFIED IN ONE OR MORE GET/PUT STAT
                  ASSUMED IN EACH CASE.


OF DIAGNOSTICS.

ILE TIME          1.38 MINS
```

BIBLIOGRAPHIE

1. DEPARTMENT OF DEFENSE COBOL 65
   Government Printing Office Washington D.C. 20402

2. I.B.M. PL/1 Language spécifications
   Form C28-6571-4

3. I.B.M. PL/1 Référence Manual
   Form C28-8201-1

4. J. LAUTER et D. WICKMANN
   Méthodes d'analyse des différences stylistiques chez un ou plusieurs auteurs,
   Cahiers du C.R.A.L. 1967 n° 2.
   Faculté des Lettres et Sciences Humaines de NANCY.

5. H.W. LAWSON PL/1 List Processing
   Journal A.C.M. June 1967 p.358-367.

6. J. LEGRAS Une extension de l'Algol : l'Algol linguistique
   4ème Congrès de Calcul et de Traitement de l'Information (Versailles 1964),

7. G.M. WEINBERG
   PL/1 Programmer Primer
   Mac Graw Hill Book Company 1966,

NOM DE L'ETUDIANT : Mlle VILLARD

Nature de la thèse : Spécialité en Mathématiques Appliquées

Vu, Approuvé

et Permis d'Imprimer

NANCY, le 20 Juin 1969

Le DOYEN :

J. AUBRY