COBOL 2

PROGRAMMATION

10/12/2020

SOMMAIRE

Éléments de base

HISTORIQUE

C O B O L :

C O m m o n
B u s i n e s s
O r i e n t e d
L a n g u a g e

Crée en 1960

Langage normalisé (ANSI, ISO)
Les jalons :

- 1968 : premier standard
- 1974 : deuxième standard (ANS 74)
- 1985 : troisième standard (ANSI X3.23-1985 et ISO 1989:1985)
  programmation structurée
- 1989 : ajout des fonctions intrinsèques (ANSI X3.23a-1989 et ISO 1989/Amendment 1)
- 1998..2000 : Cobol For Z/OS
  orientation objet, XML

COBOL ET IBM

Une succession de produits

- **OS/VS COBOL**
  - norme COBOL 1974
  - compatibilité avec les anciens COBOL IBM
- **COBOL II**
  - norme COBOL 1985
  - compatibilité avec la norme 1974
- **COBOL/370**
  - norme COBOL 1985 avec fonctions intrinsèques
  - compatibilité avec la norme 1974
  - LE/370 et CODE/370
- **COBOL for MVS**
  - nouveau nom de COBOL/370
  - mêmes caractéristiques
  - Language Environment for MVS et Debug Tool
  - support de l'orientation objet

EXEMPLE DE PROGRAMME

```
A    B                                                    72
78   12
 IDENTIFICATION DIVISION.
 PROGRAM-ID. EXEMPLE.
  zzz
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 DATE-JOUR PIC X(6).
 01 TEXTE     PIC X(10)
       VALUE   'BONJOUR XXXX'.

 PROCEDURE DIVISION.
 DEBUT.
     ACCEPT DATE-JOUR FROM DATE (.)
     DISPLAY DATE-JOUR

     DISPLAY TEXTE .
*---------------  commentaires ----
 FIN.
                  STOP RUN.
```

STRUCTURE D'UN PROGRAMME

Ensemble hiérarchisé d'éléments

- Division
  - Section
    - Paragraphe
      - Phrase
        - Chaîne de caractères

DEFINITIONS

PHRASE

- Suite de mots ou chaînes de caractères
- Se termine par un point

PARAGRAPHE

- Ensemble de phrases (instructions)
- Commence par un nom suivi d'un point
- Se termine avec le début d'un nouveau paragraphe

SECTION

- Ensemble de paragraphes (ou de phrases)
- Commence par un nom suivi du mot clé SECTION.
- Se termine avec le début d'une nouvelle section

DIVISION

- Ensemble de sections (ou de paragraphes)
- Au nombre de quatre et apparaissent dans un ordre défini

Noms de sections et paragraphes en PROCEDURE DIVISION

30 caractères maximum (lettres, chiffres et tiret)

LES QUATRE DIVISIONS

1. IDENTIFICATION DIVISION

- Identifie le programme
- **Obligatoire** et en tête de programme
- Comprend uniquement des paragraphes, pas de sections

2. ENVIRONMENT DIVISION

- Pour déclarer la configuration et les fichiers utiles au traitement
- Facultative, comprend 1 ou 2 sections

3. DATA DIVISION

- Comporte la description des données
- Facultative
- Comprend 1 à 4 sections.

4. PROCEDURE DIVISION

- Comporte les instructions
- Facultative
- Comprend des sections et/ou des paragraphes

FORMAT DE REFERENCE

- **Colonne 1 à 6**
  - Numéro de ligne
- **Colonne 7**
  - A blanc ou caractères spéciaux , exemples
    - * pour les commentaires
    - / saut de page sur la liste de compilation
    - - caractère de continuation pour les littéraux
    - D pour Debug mode
- **Colonne 8 à 11** : marge A

  Début des noms de divisions, sections, paragraphes ...
- **Colonne 12 à 72** : marge B

  Phrases, instructions ...
- **Colonne 73 à 80**

  Identification du programme

SCHEMA GENERAL D'UN PROGRAMME

```
  IDENTIFICATION DIVISION.
*--------------------
  PROGRAM-ID. nom-du-programme.
  paragraphes Optionnels
/
  ENVIRONMENT DIVISION.
*-----------------
  CONFIGURATION SECTION.
  SOURCE-COMPUTER. machine-de-compilation.
  OBJECT-COMPUTER. machine-d'exécution.
  SPECIAL-NAMES.
      description de certains paramètres

  INPUT-OUTPUT SECTION.
  FILE-CONTROL.
     déclaration des fichiers
  I-O-CONTROL.
      contrôle des entrées/sorties
/
  DATA DIVISION.
*----------
  FILE SECTION.
         description des enregistrements de fichiers

  WORKING-STORAGE SECTION.
     description des zones de travail (variables)

  LOCAL-STORAGE SECTION.
     description des zones allouées et libérées dynamiquement

  LINKAGE SECTION.
     description des zones de communication
/
  PROCEDURE DIVISION.
  nom-section SECTION.
  nom-paragraphe.
      instructions de traitement
```

SCHEMA GENERAL D'UN PROGRAMME

VARIABLES ET LITTERAUX

Variable

- identifiée par un nom
- définie en DATA DIVISION
- Variable : Peut être modifiée

  exemples : NOM, CPT, NB-CLIENT-LUS

Littéral (constante)

- chaîne de caractères qui représente directement une valeur (figée)
- **2 types**
  - **Non numériques** : 1 à 160 caractères entre guillemets (") ou apostrophes (’)
    Option de compilation QUOTE/APOST
- **Numériques** : 1 à 18 chiffres (et éventuellement signe et marque décimale)

- **exemples:** ’FORMATION COBOL’, ‘123,45’, 12345

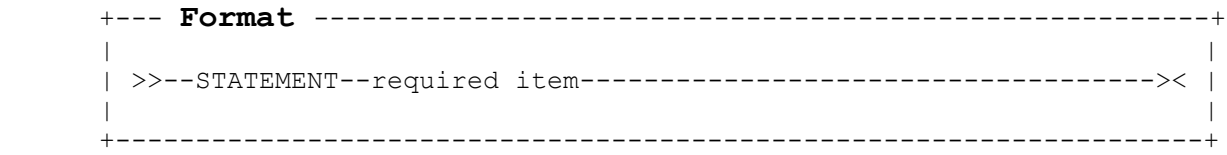
A un type de variable correspond un type de littéral

Notation

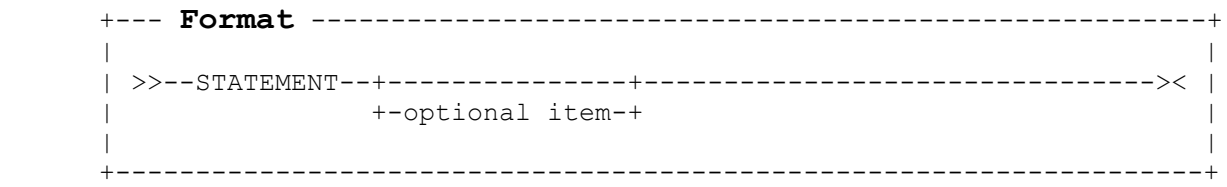
NOTATIONS

- début d'une instruction
- fin d'une instruction
- l'instruction se poursuit
- suite de l'instruction
- mots en majuscules = mots clés
- mots en minuscules = variables

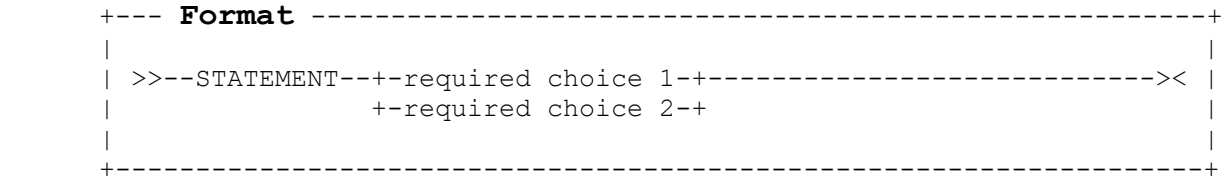
  créées par l'utilisateur
- extensions IBM entre accolades

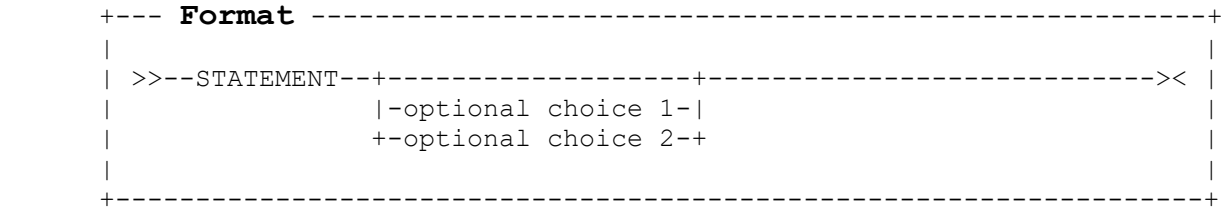

Format
>>--STATEMENT--required item----------><

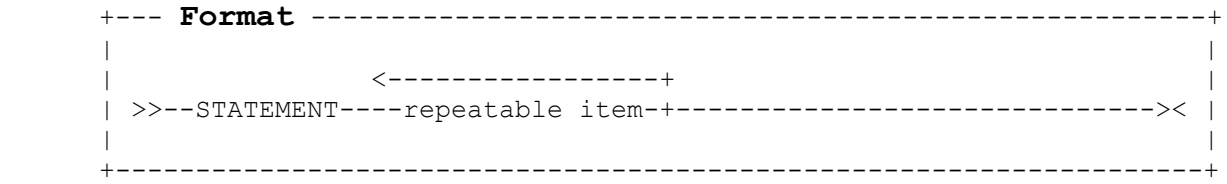
.

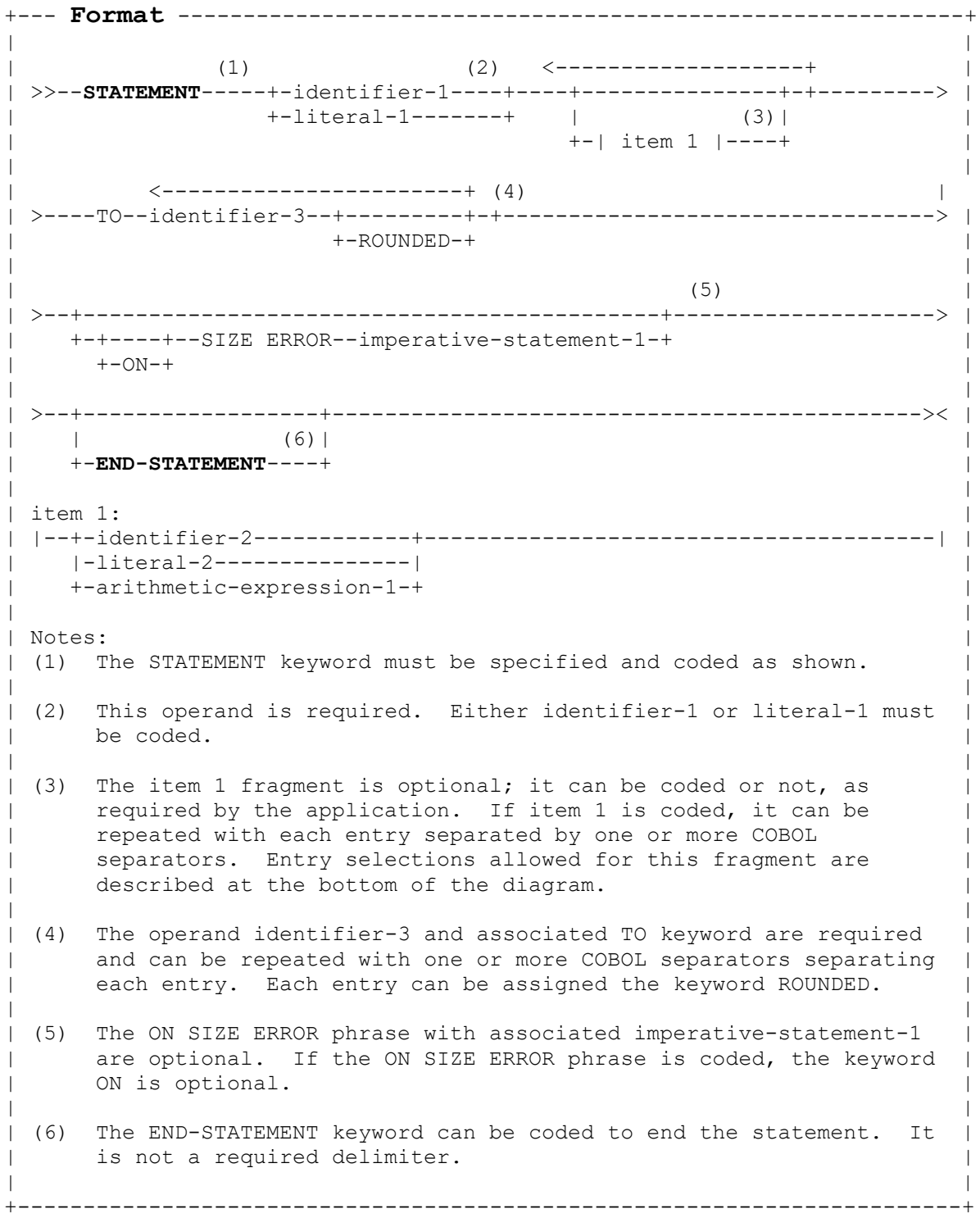

Format
>>--STATEMENT--+---------------+---------><
+-optional item-+

.


Format
>>--STATEMENT--+-required choice 1-+---------><
+-required choice 2-+

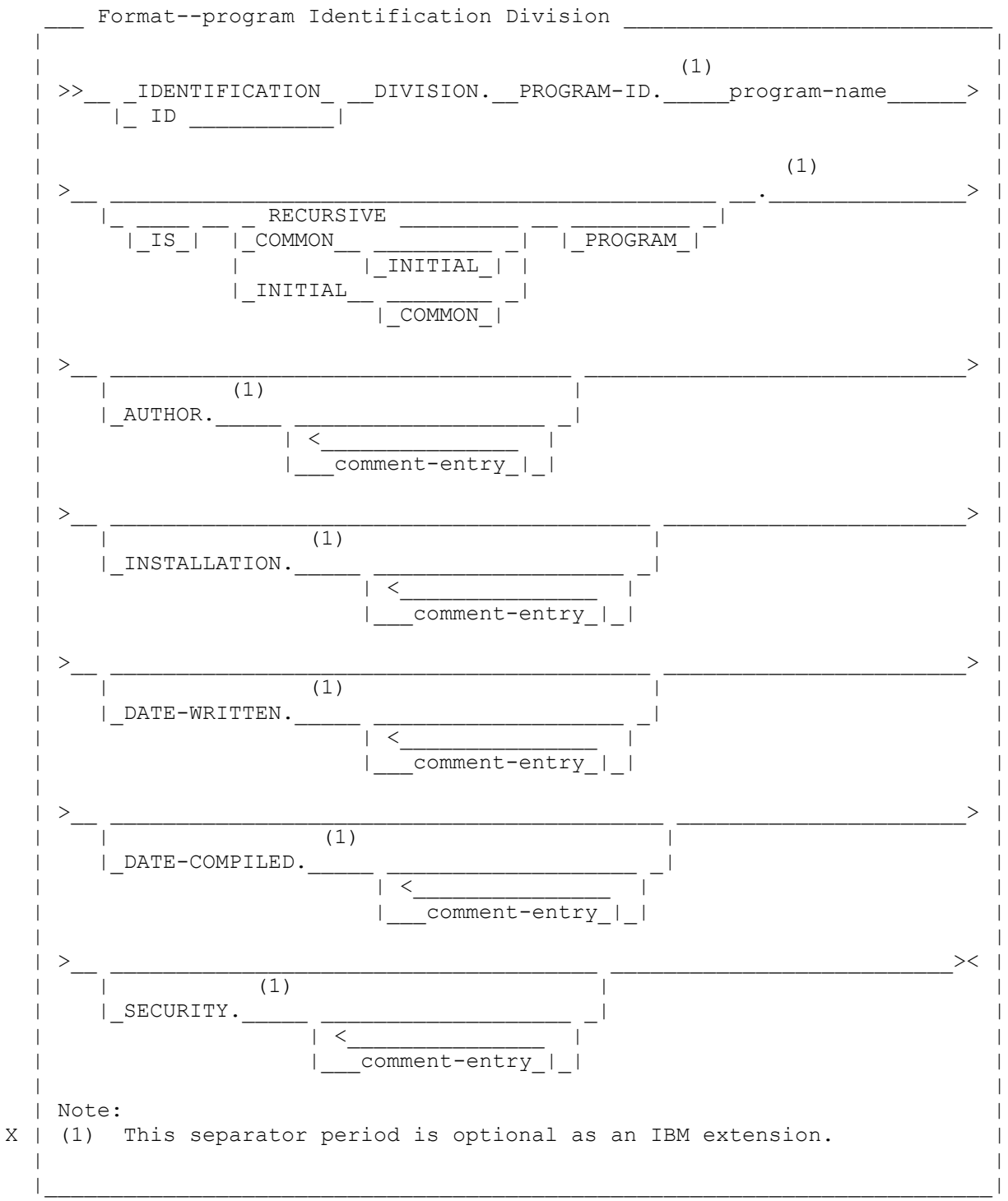

Format
>>--STATEMENT--+-------------------+---------><
|-optional choice 1-|
+-optional choice 2-+


Format
<-----------------+
>>--STATEMENT----repeatable item-+---------><


Format
(1)
(2)
STATEMENT
identifier-1
literal-1
(3)
item 1
(4)
TO
identifier-3
ROUNDED
(5)
SIZE ERROR
imperative-statement-1
ON
(6)
END-STATEMENT
item 1:
identifier-2
literal-2
arithmetic-expression-1
Notes:
(1) The STATEMENT keyword must be specified and coded as shown.
(2) This operand is required. Either identifier-1 or literal-1 must be coded.
(3) The item 1 fragment is optional; it can be coded or not, as required by the application. If item 1 is coded, it can be repeated with each entry separated by one or more COBOL separators. Entry selections allowed for this fragment are described at the bottom of the diagram.
(4) The operand identifier-3 and associated TO keyword are required and can be repeated with one or more COBOL separators separating each entry. Each entry can be assigned the keyword ROUNDED.
(5) The ON SIZE ERROR phrase with associated imperative-statement-1 are optional. If the ON SIZE ERROR phrase is coded, the keyword ON is optional.
(6) The END-STATEMENT keyword can be coded to end the statement. It is not a required delimiter.

IDENTIFICATION
et
ENVIRONMENT DIVISION

Format général simplifié



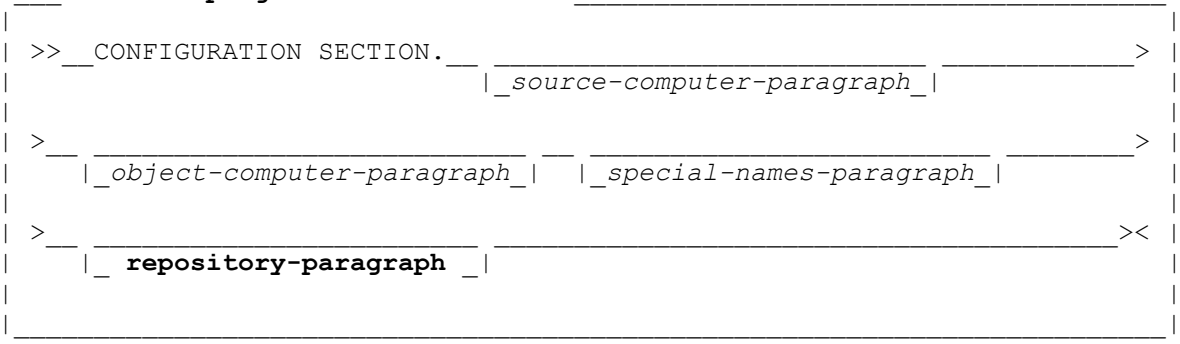
Nom-programme

- le premier caractère doit être alphabétique
- seuls les 8 premiers sont pris en compte
- si tiret : transformé en zéro

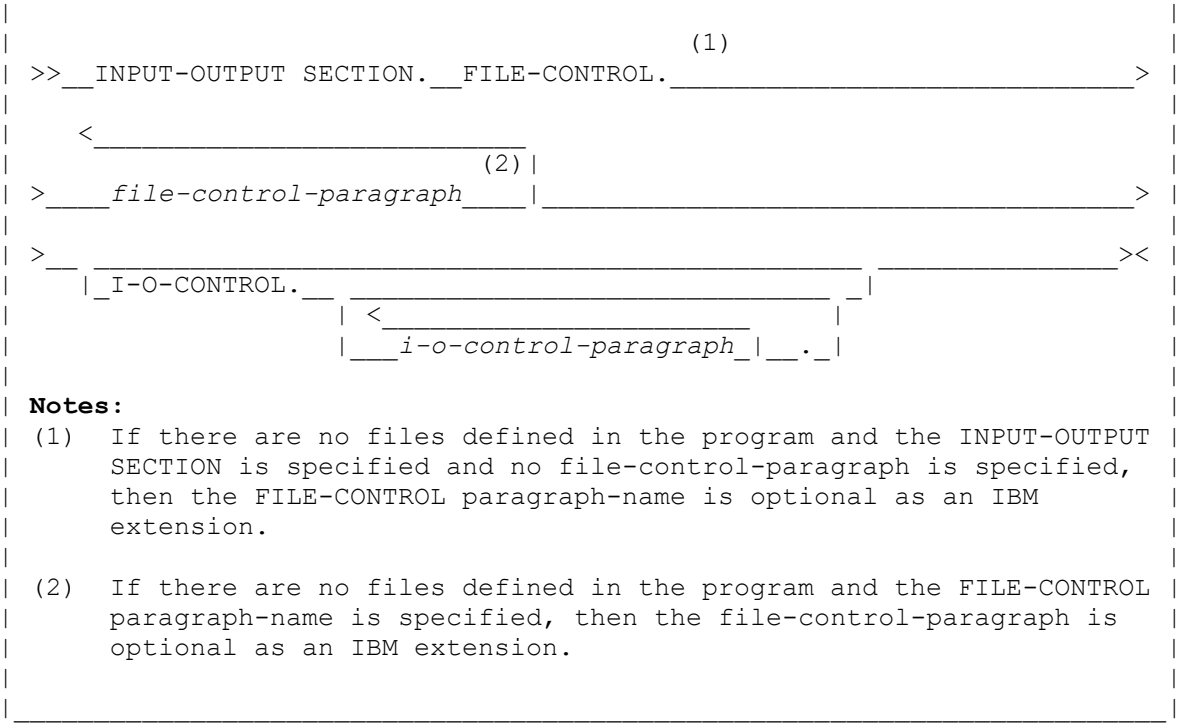
L'ENVIRONMENT DIVISION est optionnelle

ENVIRONMENT DIVISION. environment-division-content

Format--programs and classes



Programs and methods

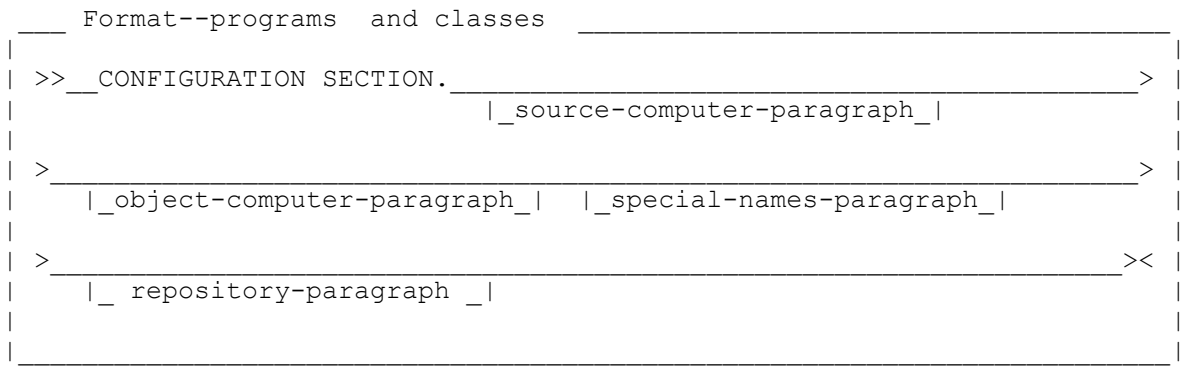


Notes:

(1) If there are no files defined in the program and the INPUT-OUTPUT SECTION is specified and no file-control-paragraph is specified, then the FILE-CONTROL paragraph-name is optional as an IBM extension.

(2) If there are no files defined in the program and the FILE-CONTROL paragraph-name is specified, then the file-control-paragraph is optional as an IBM extension.

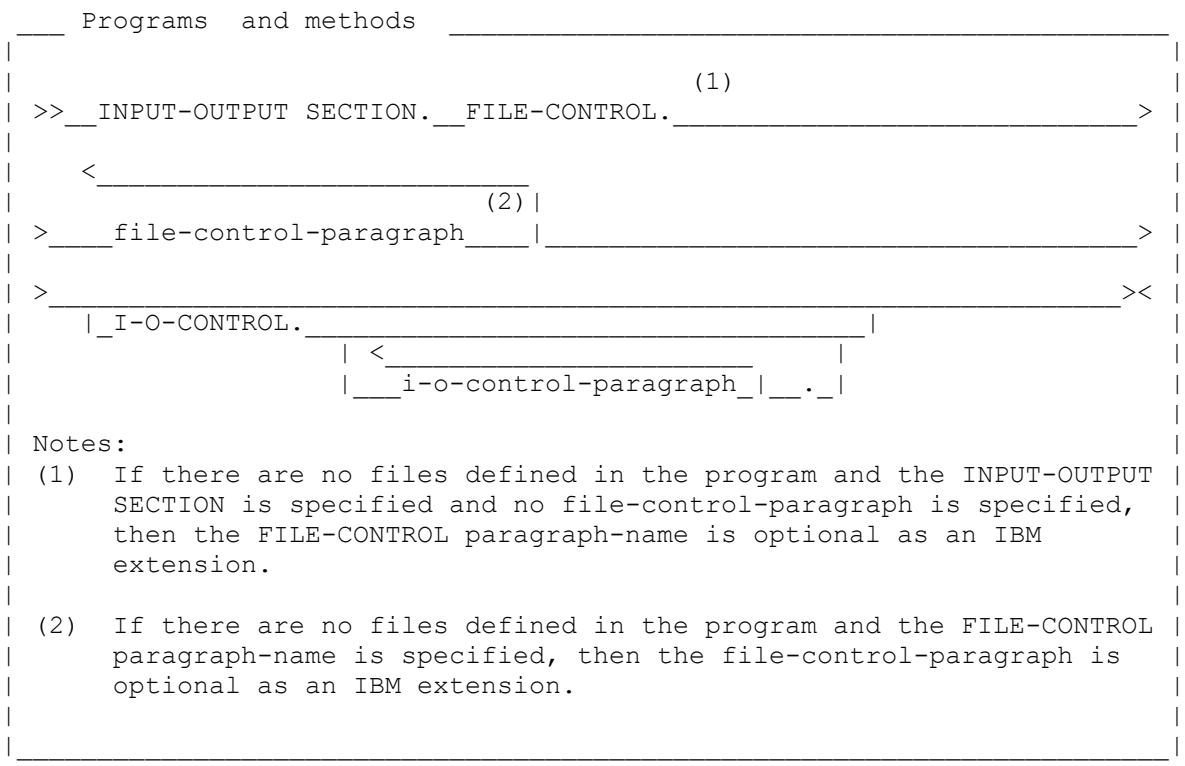
CONFIGURATION SECTION



Exemple :

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
     DECIMAL-POINT IS COMMA.
```

INPUT-OUTPUT SECTION



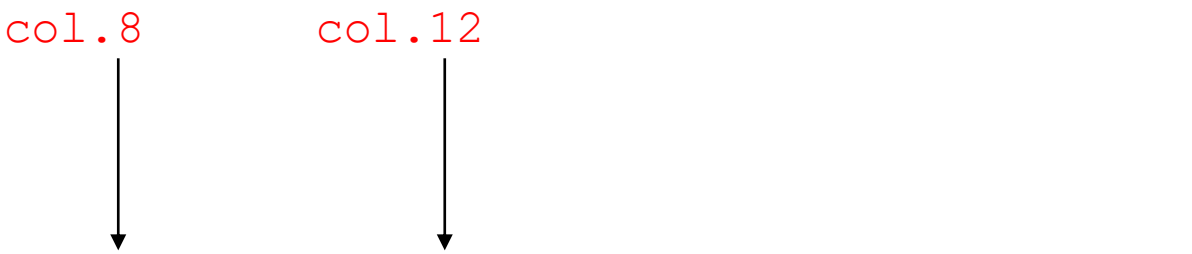
FILE-CONTROL : déclaration des fichiers utilisés par le programme

I-O-CONTROL : paramètres de gestion des entrées/sorties

DATA DIVISION

Description des données

DONNEES ELEMENTAIRES




```
77              Z1              PICTURE X.
01              Z2A             PIC XXX.
01              Z2B             PIC X(3).
01              Z2C             PIC A(10).
01              Z3A             PIC 99999.
1               Z3B             PIC 9(5).
01              Z4              PIC 9(18).
01              Z5              PIC X(16000000).
```

Nombre, niveau

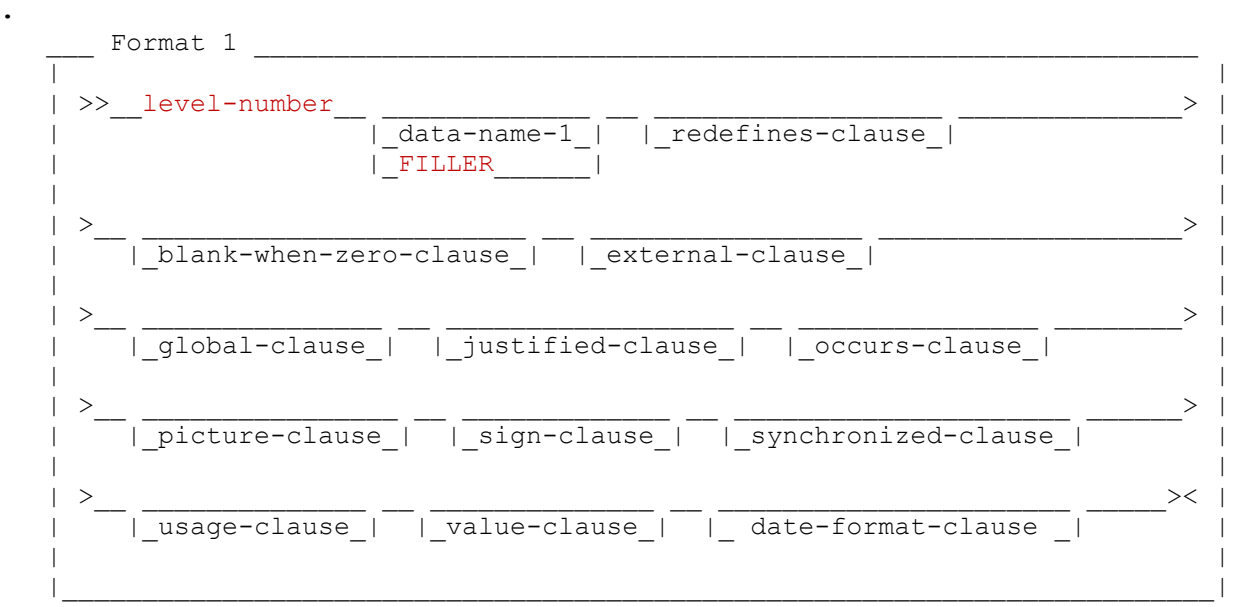
- 01 ou 77, en marge A

Nom de donnée

- 30 caractères maximum :
  - lettres majuscules et minuscules (au moins 1)
  - chiffres

tirets (pas en premier ni dernier caractère)

PICTURE : TYPE ET LONGUEUR



Principaux types :

- **X** : alphanumérique
- **A** : alphabétique
- **9** : numérique , combinable avec
  - **S** : signe opérationnel
  - **V** : marque décimale implicite
  - **P** : facteur d'échelle

Longueur :

- **Par répétition** : PIC 999
- **Par mise en facteur** : PIC X(10)

Exemples:

```
WORKING-STORAGE SECTION.
01 ZONE                    PIC X(20).
01 MONTANT                 PIC 9(5)V99.
01 MONTANT                 PIC 9(5)V9(2).
01 TOTAL                   PIC S9(6)V99.
```

Exemple : numéro de sécurité sociale

```
01    NUM-SS.
      05  SEXE                PIC 9.
      05  DAT-NAIS.
           10 AN               PIC 99.
           10 MOIS             PIC 99.
      05 LIEU-NAISS.
           10 DEPT             PIC 99.
           10 VILLE            PIC 9(3).
      05 COD                   PIC 9(3).
```

Nombre niveau

- Fait apparaître la hiérarchie des données
- 01 à 49
- 01 en marge A, autres en marge A ou B

Nom de donnée

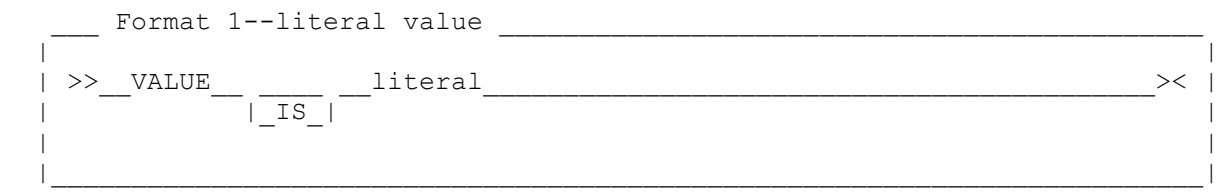
- Facultatif
- FILLER (ou rien) pour réserver une zone sans la nommer

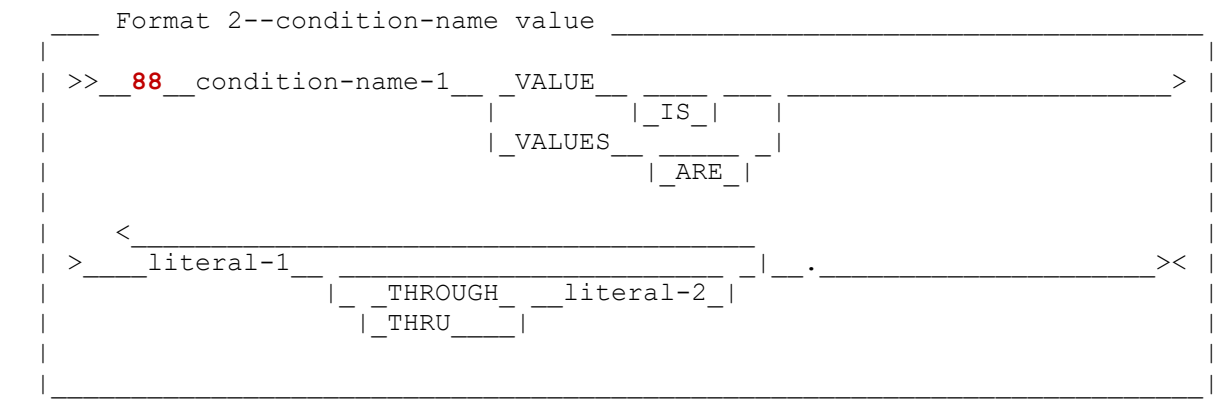
Picture

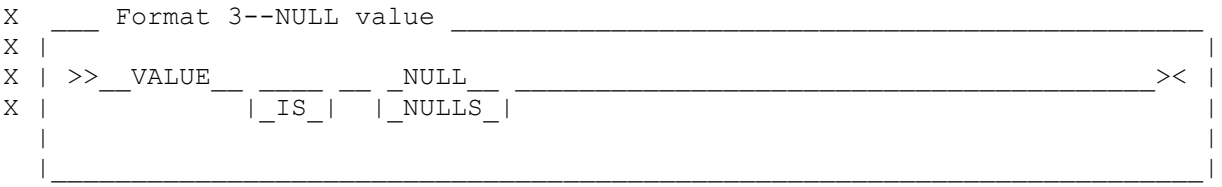
- Uniquement aux niveaux élémentaires

Comment déterminer type et longueur d'une zone groupe ?

VALUE







Donne une valeur initiale à une donnée de

WORKING-STORAGE SECTION

Exemple :

```
01 COMPTEUR      PIC S999  VALUE +10.
01 TITRE         PIC X(10) VALUE 'FORMATION'.
01 TITRE         PIC X(10) VALUE ZERO.
```

CONSTANTES FIGURATIVES

- ZERO, ZEROS, ZEROES
- SPACE, SPACES
- HIGH-VALUE, HIGH-VALUES
- LOW-VALUE, LOW-VALUES
- QUOTE, QUOTES
- ALL suivi d'un littéral non numérique d'un caractère ou plus
- NULL, NULLS (extension de la norme COBOL 85)

PROCEDURE DIVISION

Neuf ensembles :

- Arithmétiques
- Tri
- Mouvement de donnée
- Branchement
- Décision
- Gestion de table
- Entrée/sortie
- Gestion de sous-programme
- Fin de programme

ARITHMÉTIQUES

- ADD
- DIVIDE
- MULTIPLY
- SUBTRACT
- COMPUTE

TRI

- MERGE
- RELEASE
- SORT
- RETURN

MOUVEMENT DE DONNÉE

- ACCEPT (time, day, date, day-of-week)
- INITIALIZE
- MOVE
- INSPECT
- STRING
- UNSTRING

BRANCHEMENT

- ALTER
- EXIT
- GO TO
- **PERFORM**

FIN DE PROGRAMME

- STOP RUN
- EXIT PROGRAM
- GOBACK

DÉCISION

- EVALUATE
- IF

GESTION DE TABLE

- SEARCH
- SET

ENTREES/SORTIES

- ACCEPT identificateur
- DISPLAY
- OPEN
- READ
- WRITE
- REWRITE
- DELETE
- START
- CLOSE

GESTION DE SOUS-PROGRAMME

- CALL
- CANCEL
- ENTRY

LES INSTRUCTIONS : QUATRE CATEGORIES

IMPÉRATIVES

- Action inconditionnelle effectuée par le programme
- Une instruction conditionnelle avec délimiteur explicite est considérée comme une instruction impérative

CONDITIONNELLES

- Action déterminée par le résultat d'une condition
- Une instruction impérative peut devenir conditionnelle si une condition est incluse dans son format

  (ex : ADD ... ON SIZE ERROR) avec un délimiteur implicite

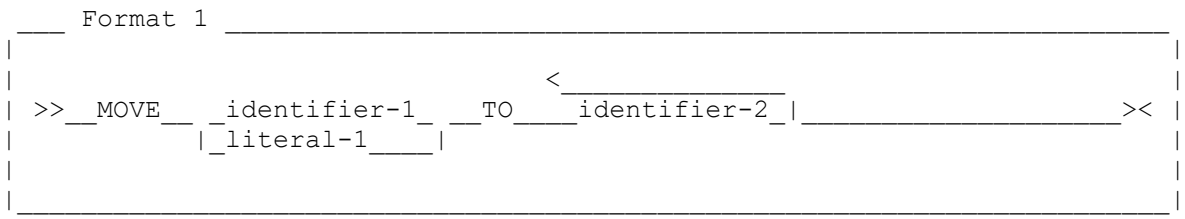
AVEC DÉLIMITEUR

- Le '.' est le délimiteur implicite
- Les **END-xxxx** sont les délimiteurs explicites des instructions pouvant l'utiliser

DIRECTIVES

Ordres donnés au compilateur codés dans le programme

Affectation



Transfert de données

EMETTEUR ----------> RECEPTEUR

MOVE V1 TO V2

MOVE 1 TO V2 V2 PIC 9)

MOVE 'A' TO V2 (si V2 PIC 9 Erreur compilation)

MOVE 'formation' TO V2 (PIC X)

MOVE V1 TO V2 V3 V4

Que se passe-t-il si émetteur et récepteur sont de type ou de longueur différents ?

Exemples :

```
01  ZONE1-X              PIC X(5).
01  ZONE1-9              PIC 9(5).
```

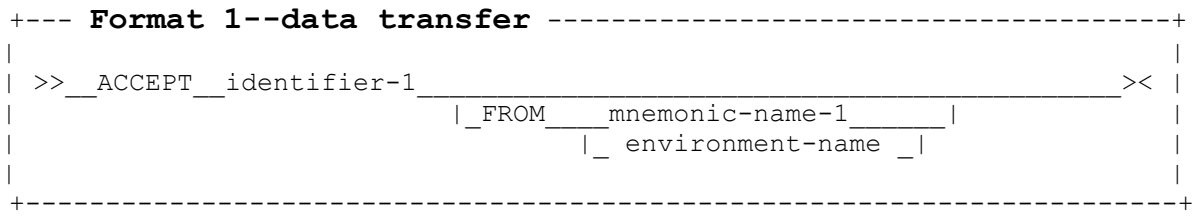
```
MOVE 'ABC'       TO   ZON1-X     ZON1-X  =  ABCbb   (b : espace)
MOVE 'ABCDEFG'   TO   ZON1-X     ZON1-X  =  ABCDE'
MOVE  123        TO   ZONE1-9    ZON1-9  =  00123
MOVE  123567     TO   ZONE1-9    ZON1-9  =1234567   (12 perdu)
```

```
01  ENREGISTREMENT1.
     02 MATRICULE          PIC 9(6).
     02 NOM-PRENOM         PIC X(20).
     02 ADRESSE            PIC 9(10).
     02 RUE-ADRESSE        PIC X(18).
     02 CODE-POSTAL        PIC 9(5).
     02 VILLE              PIC X(12).

 01  ENREGISTREMENT2.
     02 VILLE              PIC X(12).
     02 NOM-PRENOM         PIC X(20).
     02 RUE-ADRESSE        PIC X(18).
     02 ADRESSE            PIC 9(10).
     02 MATRICULE          PIC 9(6).
     02 CODE-POSTAL        PIC 9(5).
```

```
MOVE MATRICULE OF ENREGISTREMENT1 TO MATRICULE OF ENREGISTREMENT2
MOVE ..........  OF ENREGISTREMENT1 TO ..........  OF ENREGISTREMENT2
MOVE ..........  OF ENREGISTREMENT1 TO ..........  OF ENREGISTREMENT2
MOVE VILLE       OF ENREGISTREMENT1 TO VILLE       OF ENREGISTREMENT2
```

```
MOVE CORR ENREGISTREMENT1 TO ENREGISTREMENT2
```

Réception de paramètres



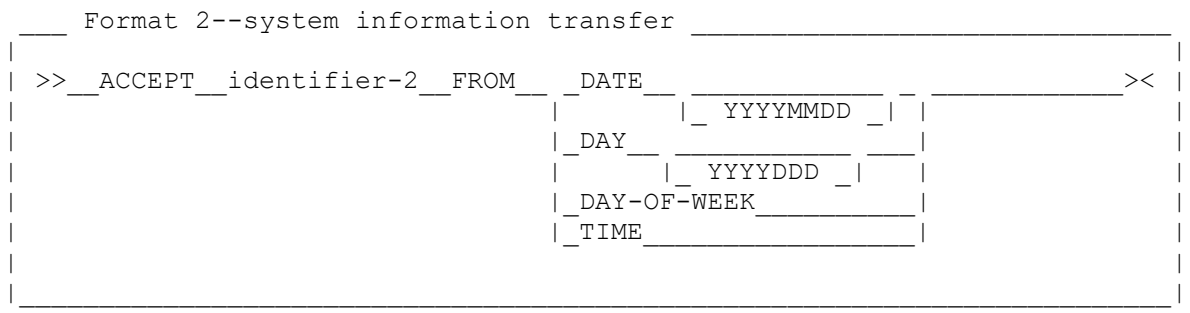
Nom-mnémonique-1

- défini en SPECIAL-NAMES

Nom-externe

- SYSIN (défaut)
- CONSOLE

Réception date et heure système



Identificateur-2 peut être un groupe, une donnée alphanumérique ou numérique

DATE

- Format implicite PIC 9(6) : AAMMJJ

DAY

- Format implicite PIC 9(5) : AAJJJ

DAY-OF-WEEK

- Format implicite PIC 9 : rang du jour de la semaine

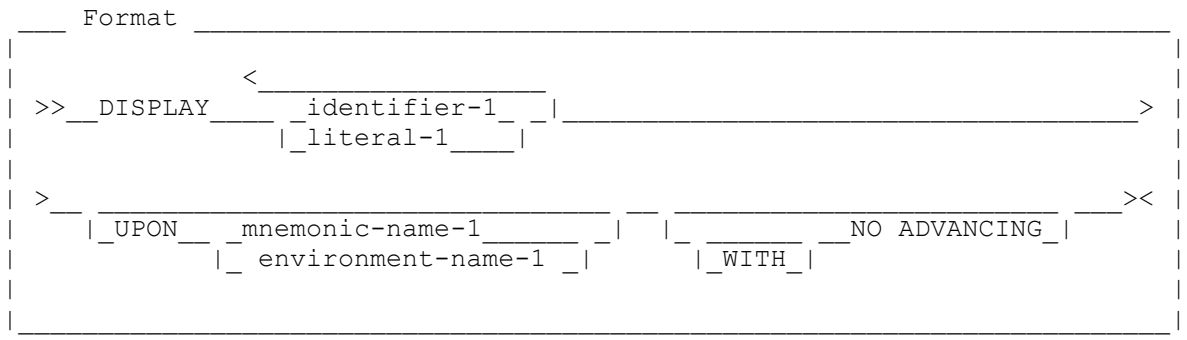
TIME

- Format implicite PIC 9(8) : HHMMSSCC

Exemples :

```
01 DATE1        PIC  9(6).
01 DATE2        PIC  9(8).
   ACCEPT  DATE1   FROM  DATE  ===> 200706
   ACCEPT  DATE8   FROM  DATE YYYYMMDD ===> 20200706
```

Emission de messages



Nom-mnémonique-1

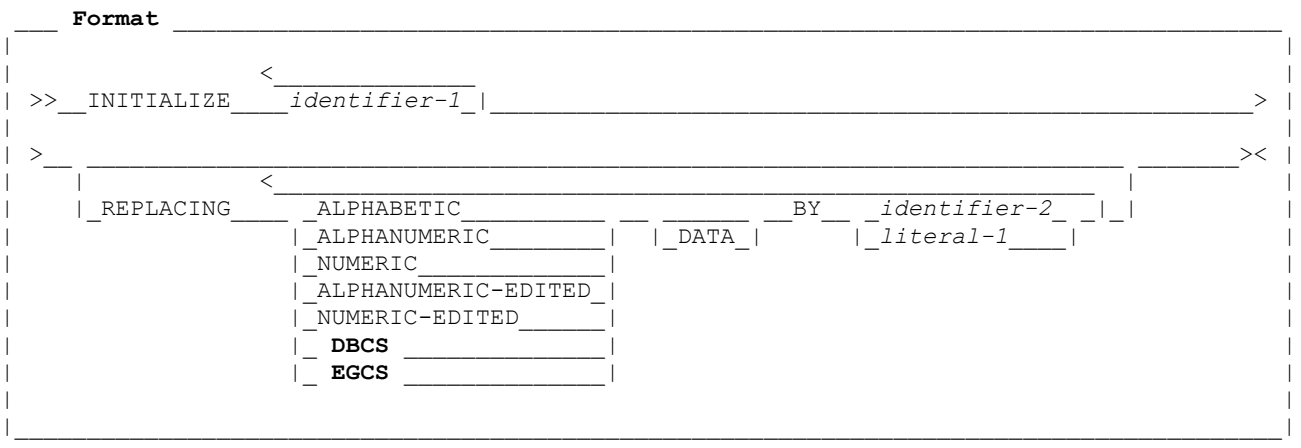
- défini en SPECIAL-NAMES

Nom externe-1

- SYSOUT (défaut)
- CONSOLE

WITH NO ADVANCING permet de ne pas déclencher de saut de lignes après l'instruction DISPLAY

INITIALIZE



Les données sont initialisées selon leur nature, par défaut

- alphabétiques et alphanumériques : SPACE
- numériques : ZERO

Initialisation non effectuée pour FILLER

Exemple:

```
01  LIGNE.
    05             PIC    X(20) VALUE SPACES.
    05             PIC    X(25) VALUE 'EXEMPLE'.
    05  NOM        PIC    X(25).
    05  DATEJ.
        10  JJ     PIC    99.
        10         PIC     X VALUE '-'.
        10  MM     PIC    99.
        10         PIC     X VALUE '-'.
        10  AA     PIC    99.
    05
```

INITIALIZE LIGNE

STOP

Fin de programme

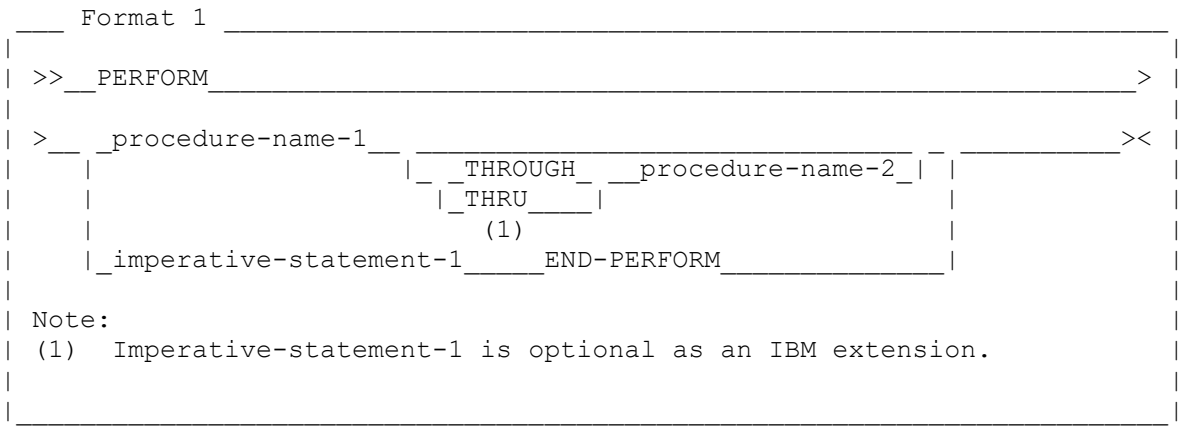
STOP RUN

- Arrêt définitif du programme (fin logique)

STOP littéral

Arrêt momentané, reprise sur intervention de l'opérateur

Débranchement à une séquence d'instructions puis retour

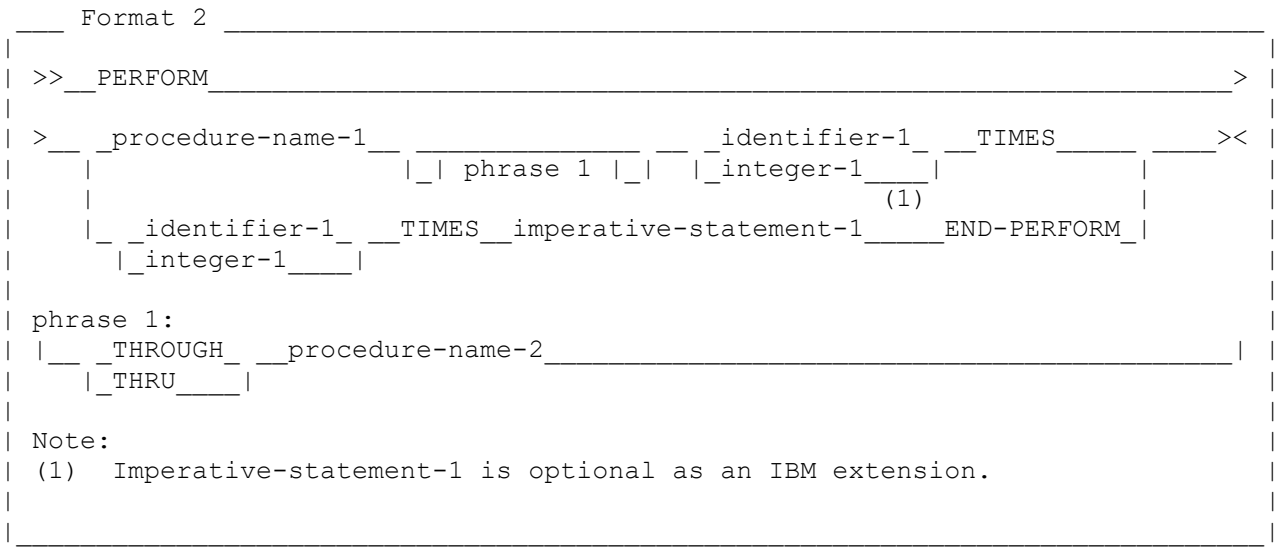


Procédure : nom de section ou de paragraphe de la PROCEDURE DIVISION

Exemple :

```
PROCEDURE DIVISION.
S1 SECTION.
P11.
    MOVE 0 TO CPT
    PERFORM P12
    PERFORM S2.

P12.
     PERFORM P22 THRU P23.
P13.
...
S2 SECTION.

P21.
P22...
    ADD 1 TO CPT.
P23.

...
S3 SECTION.
```

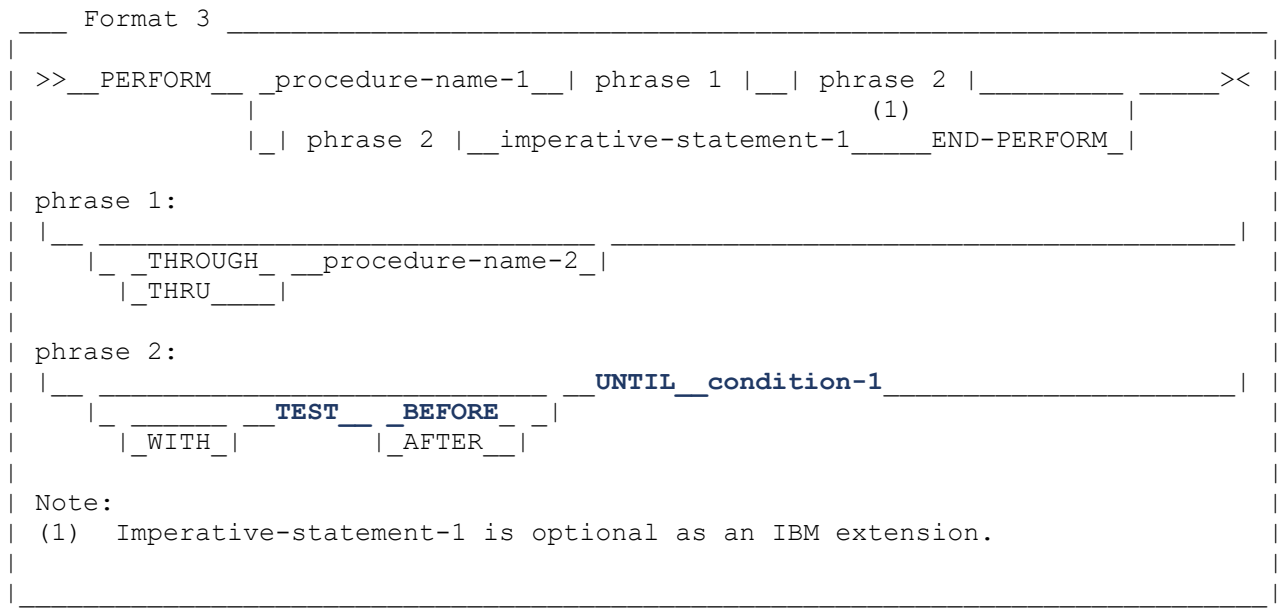
Que vaut CPT ?



Exemples

```
77 CPT      PIC 9  VALUE 5.
PERFORM CPT TIMES
    ADD  1  TO  CPT
    DISPLAY CPT
END-PERFORM
```

```
PERFORM P05 CPT TIMES.

P05.
    ADD  1  TO  CPT
    DISPLAY CPT
```

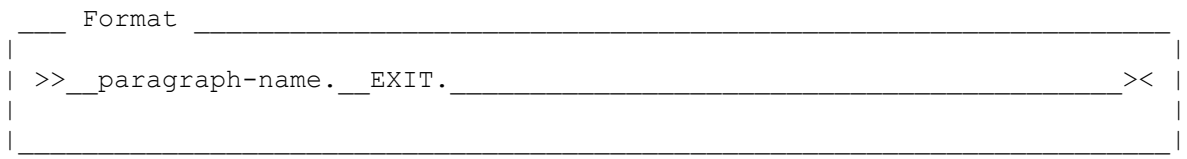


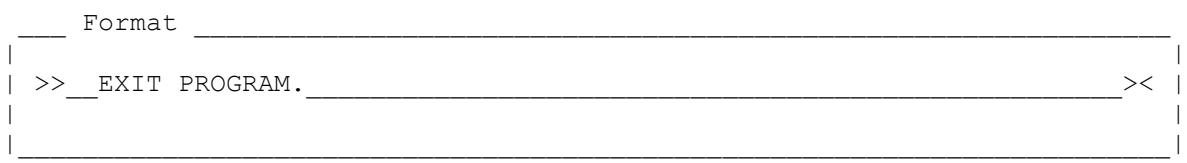
Format phrase 1

TEST BEFORE
0 à N fois

TEST AFTER
1 à N fois

EXIT





- Fin commune à une série de paragraphes
- Instruction non exécutable
- Doit être la seule instruction du paragraphe

Attention, EXIT ne provoque pas la sortie d'un PERFORM.
S'il existe des instructions après EXIT, elles sont exécutées


Format
EVALUATE
identifier-1
literal-1
expression-1
TRUE
FALSE
ALSO
identifier-2
literal-2
expression-2
TRUE
FALSE
WHEN
phrase 1
imperative-statement-1
ALSO
phrase 2
WHEN OTHER
imperative-statement-2
END-EVALUATE
phrase 1:
ANY
condition-1
TRUE
FALSE
identifier-3
NOT
literal-3
arithmetic-expression-1
THROUGH
THRU
identifier-4
literal-4
arithmetic-expression-2
phrase 2:
ANY
condition-2
TRUE
FALSE
identifier-5
NOT
literal-5
arithmetic-expression-3
THROUGH
THRU
identifier-6
literal-6
arithmetic-expression-4

EVALUATE

Les options précédant WHEN sont appelées les SUJETS
Les options suivant WHEN sont appelées les OBJETS

L'instruction procède en évaluant l'égalité d'un jeu de sujets avec un jeu d'objets

- Si l'égalité est trouvée, on exécute la ou les instructions impératives correspondantes
- Si l'égalité n'est pas trouvée, on évalue la condition suivante
- Si aucun WHEN n'a été sélectionné, et si WHEN OTHER est écrit, les instructions qui le suivent sont exécutées

- Si aucun WHEN n'est sélectionné, et WHEN OTHER n'est pas écrit, on se transfère à l'instruction suivant END-EVALUATE

- Les conditions sont effectuées dans l'ordre d'écriture
- Après une condition vérifiée, le programme continue à l'instruction suivant l'EVALUATE

Exemple 1

```
EVALUATE  A
     WHEN 1 PERFORM PAR1
     WHEN 2 PERFORM PAR2
     WHEN 5 CONTINUE
     WHEN OTHER PERFORM PAR3
END-EVALUATE
```

Exemple 2

```
EVALUATE A ALSO B ALSO C
    WHEN 1 ALSO 5 ALSO NOT 7
              PERFORM PAR1
    WHEN 2 ALSO 4 THRU 7 ALSO 3
              PERFORM PAR2
    WHEN 5 ALSO ANY ALSO ANY
             MOVE A TO B
    WHEN OTHER
          PERFORM PAR3
 END-EVALUATE
```

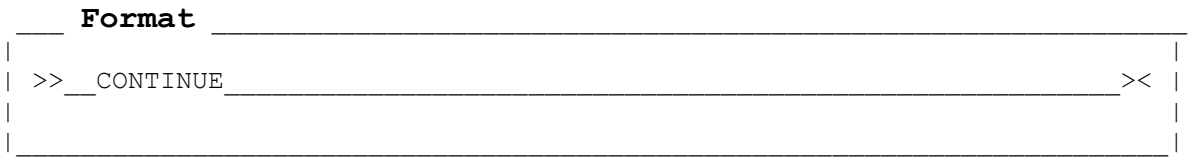
Exemple 3

```
EVALUATE TRUE ALSO FALSE
   WHEN A = 1 ALSO B = 3
              PERFORM PAR1
   WHEN A + B < C ALSO C = 4
              PERFORM PAR2
   WHEN D ALSO F
         PERFORM PAR2
   WHEN OTHER
         PERFORM PAR3
END-EVALUATE
```

Exemple 4

```
EVALUATE A = 1 ALSO B = 3
      WHEN TRUE ALSO TRUE
                PERFORM PAR1
OU  { WHEN TRUE ALSO FALSE
    { WHEN FALSE ALSO TRUE
                PERFORM PAR2
      WHEN OTHER
              PERFORM PAR3
END-EVALUATE
```

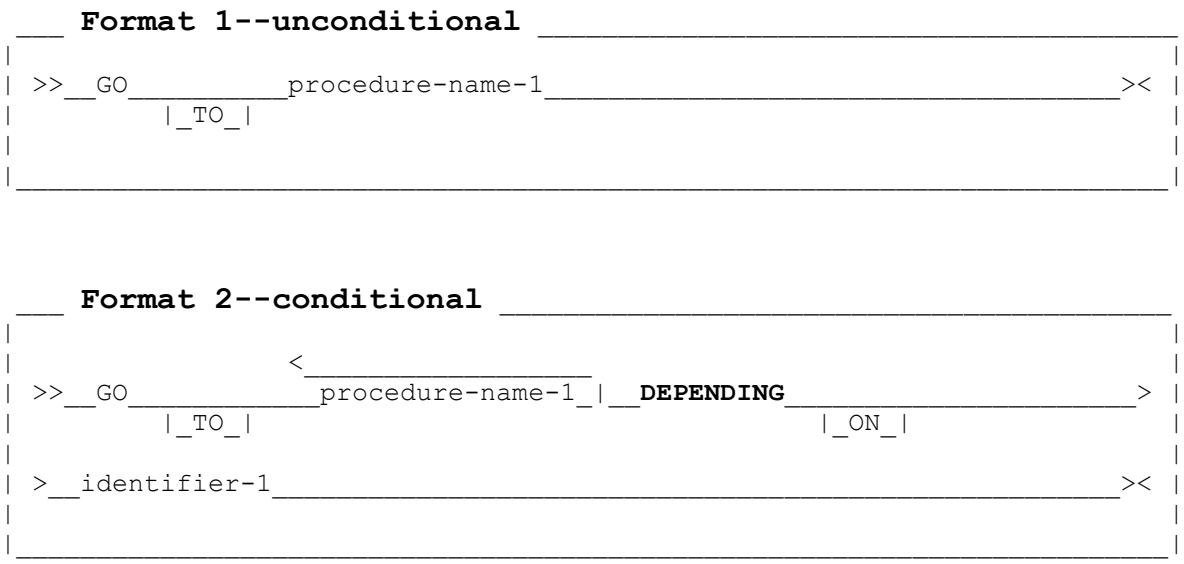
CONTINUE



- Non-instruction, "ne rien faire"

NEXT SENTENCE :

- pour remplacer une instruction conditionnelle ou impérative
- Débranchement à la fin de la phrase (après le prochain point.)



Débranchement à une section ou à un paragraphe de la PROCEDURE DIVISION (sans retour)

Option DEPENDING ON :

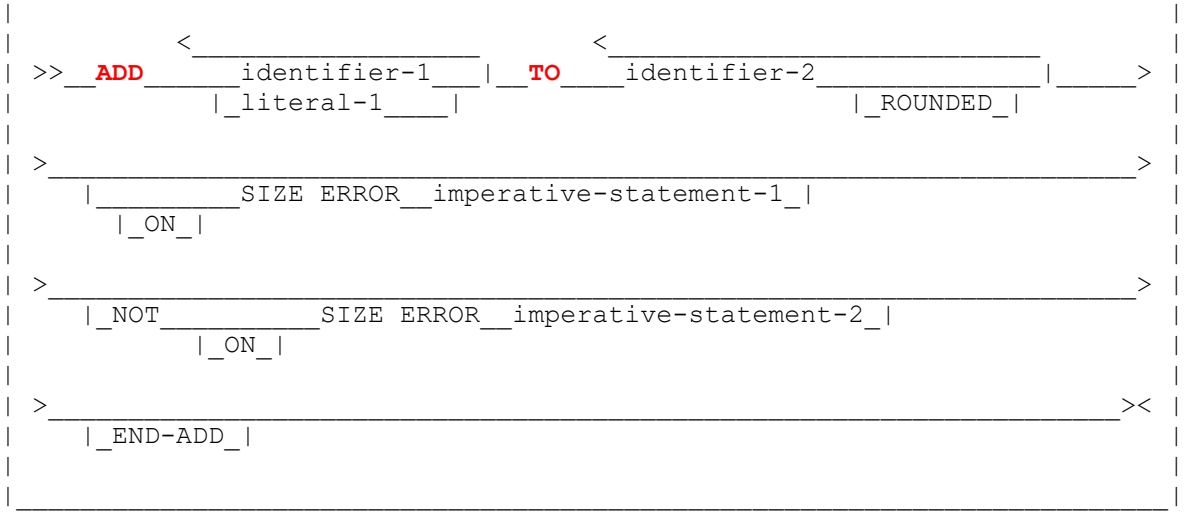
- identificateur-1 contient un nombre entier débranchement à la procédure de rang égal à la valeur d'identificateur

Opérations :

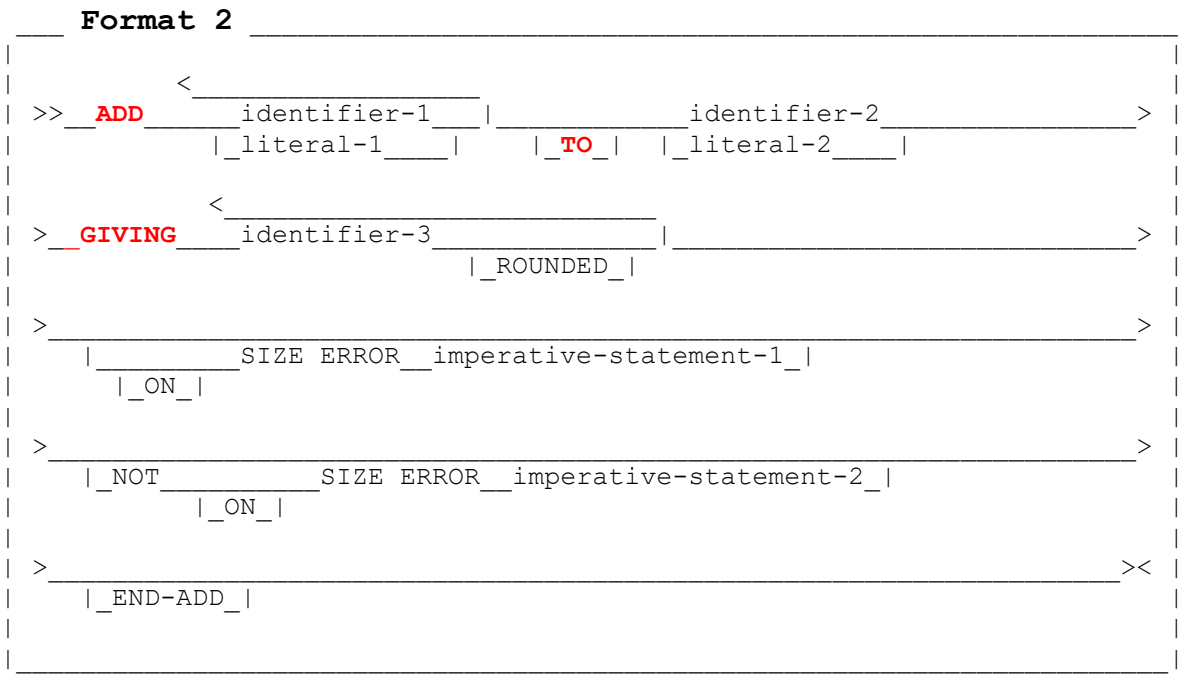
- ADD
- SUBTRACT
- MULTIPLY
- DIVIDE
- COMPUTE

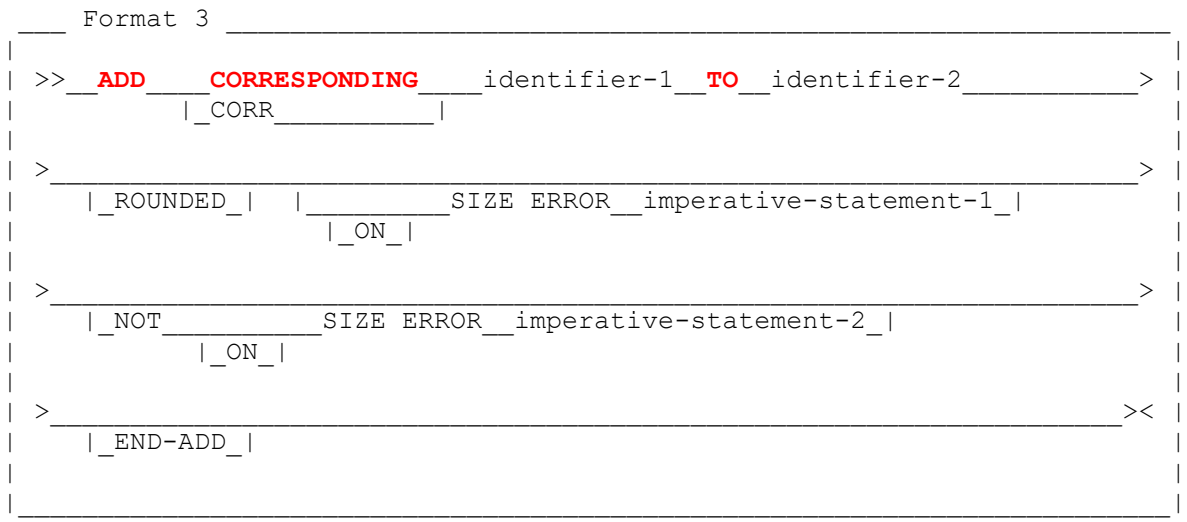
ROUNDED peut être associé à chaque zone réceptrice

Format 1



Format 2





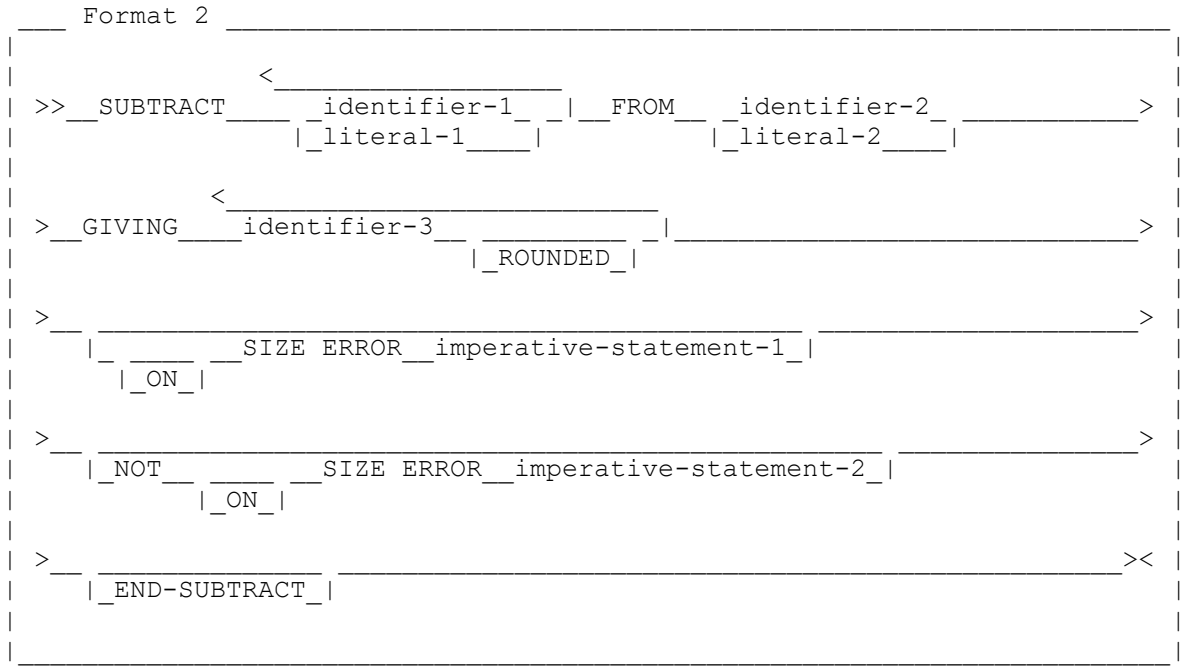
Exemples :

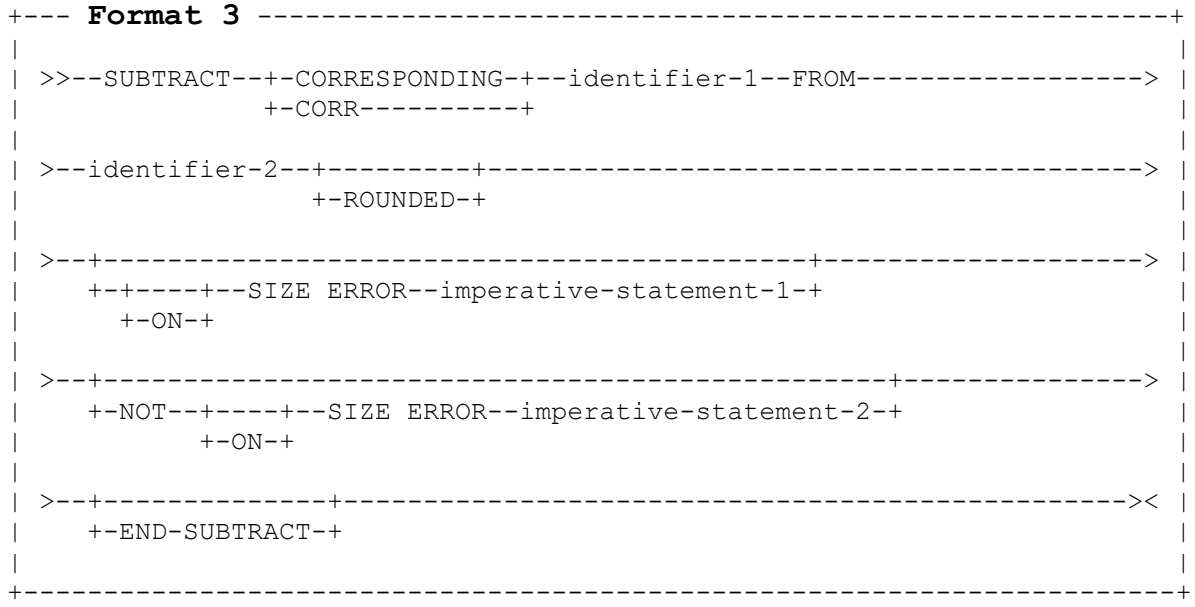
ADD A TO B --→ B = B + A
ADD A B GIVING C --→ C = A + B (A et B inchangés)
01 A PIC 999 VALUE 70.
01 B PIC 99 VALUE 40.
01 TOTAL PIC 99.

ADD A TO B =➔ B = 110
ADD A TO B GIVING TOTAL
ON SIZE ERROR
DISPLAY 'Pb B TROP PETITE, TRONQUée'
NOT ON SIZE ERROR
DISPLAY 'OK'
END-ADD
TOTAL = 110

SUBTRACT


Format 1
SUBTRACT
identifier-1
literal-1
FROM
identifier-2
ROUNDED
SIZE ERROR imperative-statement-1
ON
NOT
SIZE ERROR imperative-statement-2
ON
END-SUBTRACT

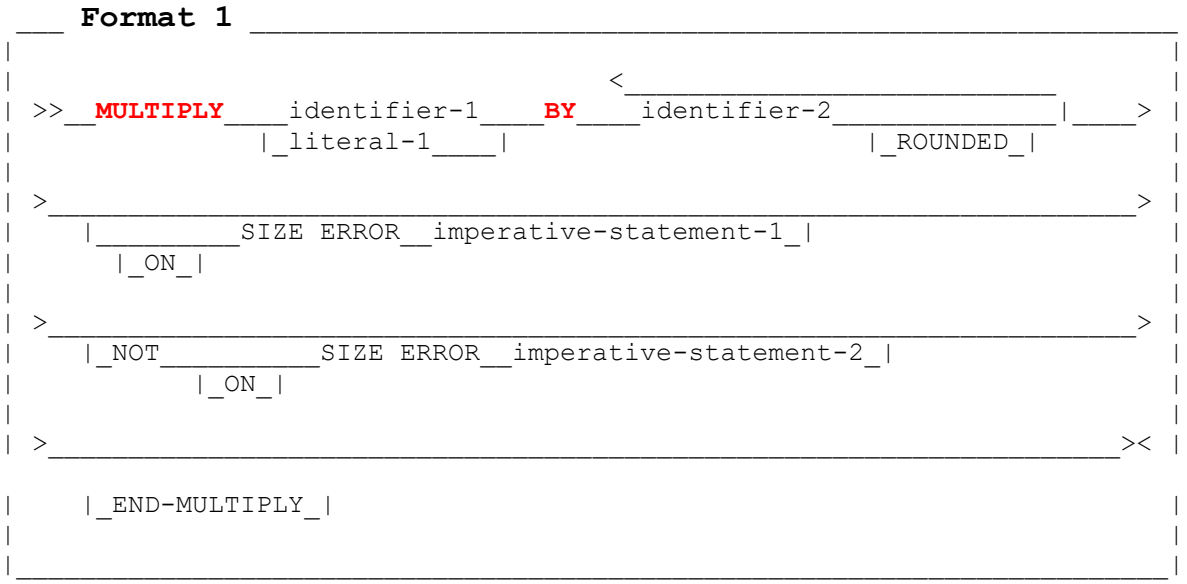

Format 2
SUBTRACT
identifier-1
literal-1
FROM
identifier-2
literal-2
GIVING
identifier-3
ROUNDED
SIZE ERROR imperative-statement-1
ON
NOT
SIZE ERROR imperative-statement-2
ON
END-SUBTRACT



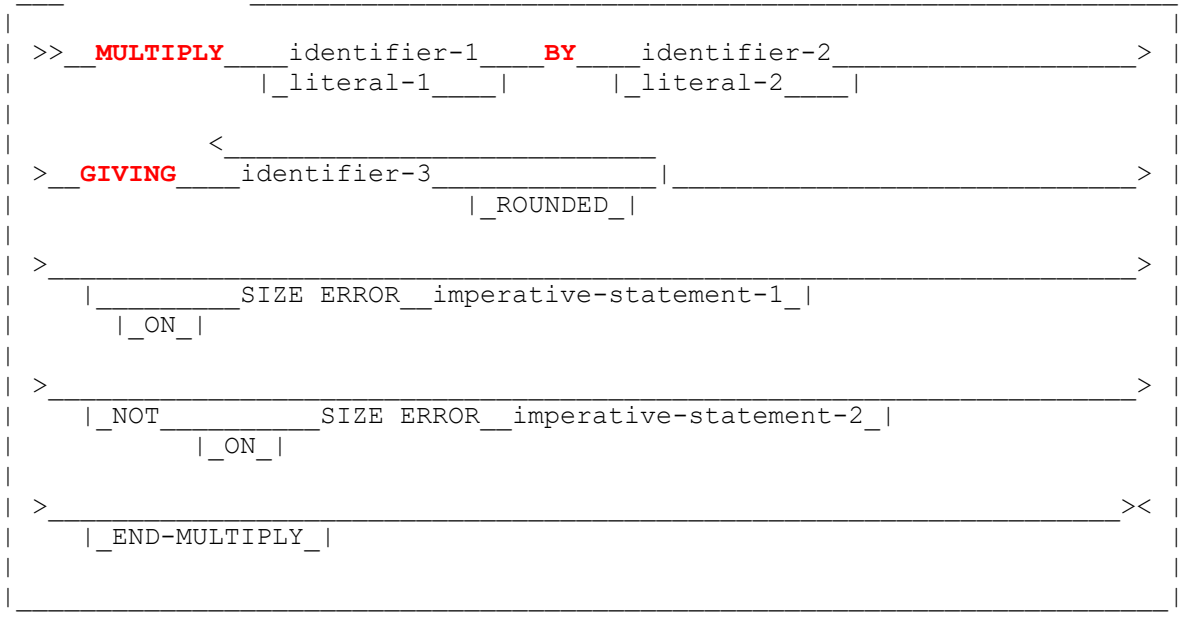
Exemples :

```
SUBTRACT A FROM B              -> B = B - A
SUBTRACT A FROM B GIVING C -> C = B - A
```

Format 1



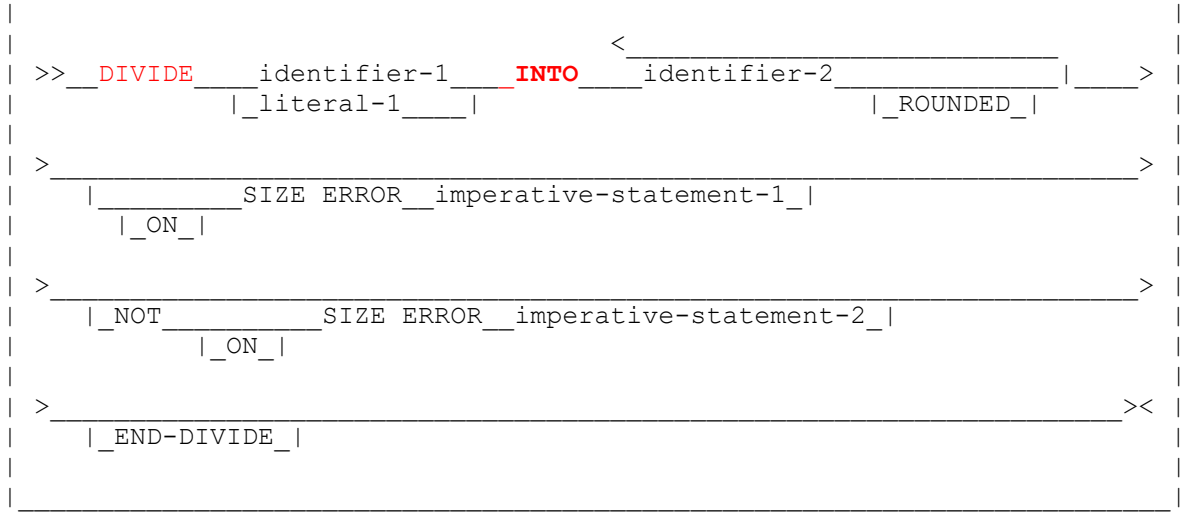
Format 2



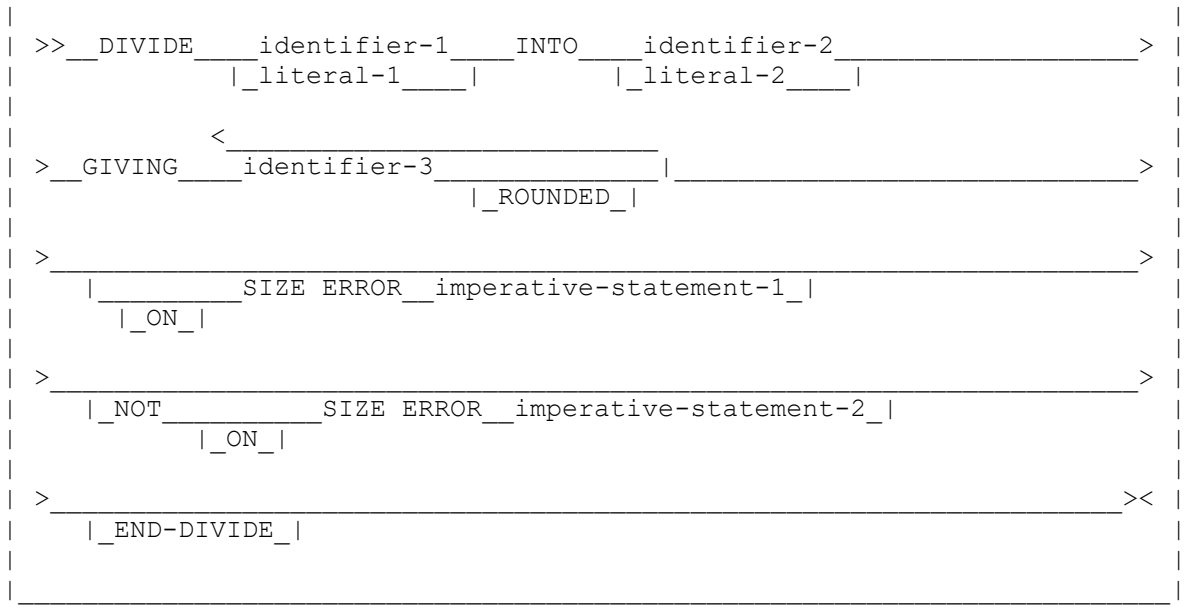
Exemples:

MULTIPLY A BY B ---> B = B x A
MULTIPLY A BY B GIVING C ---> C = A x B

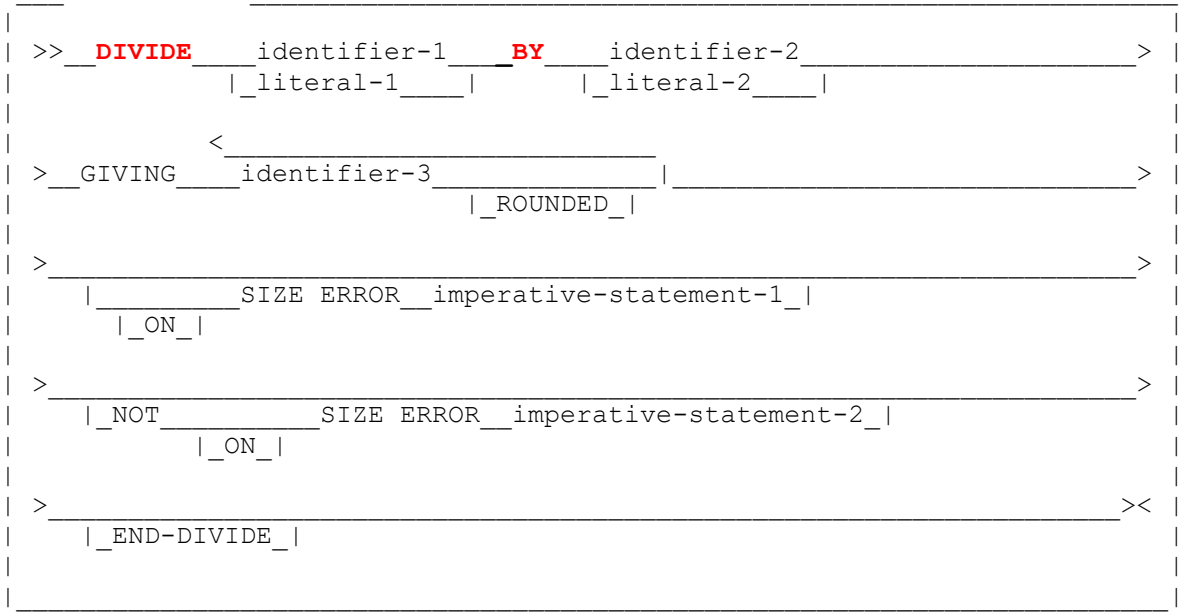
Format 1



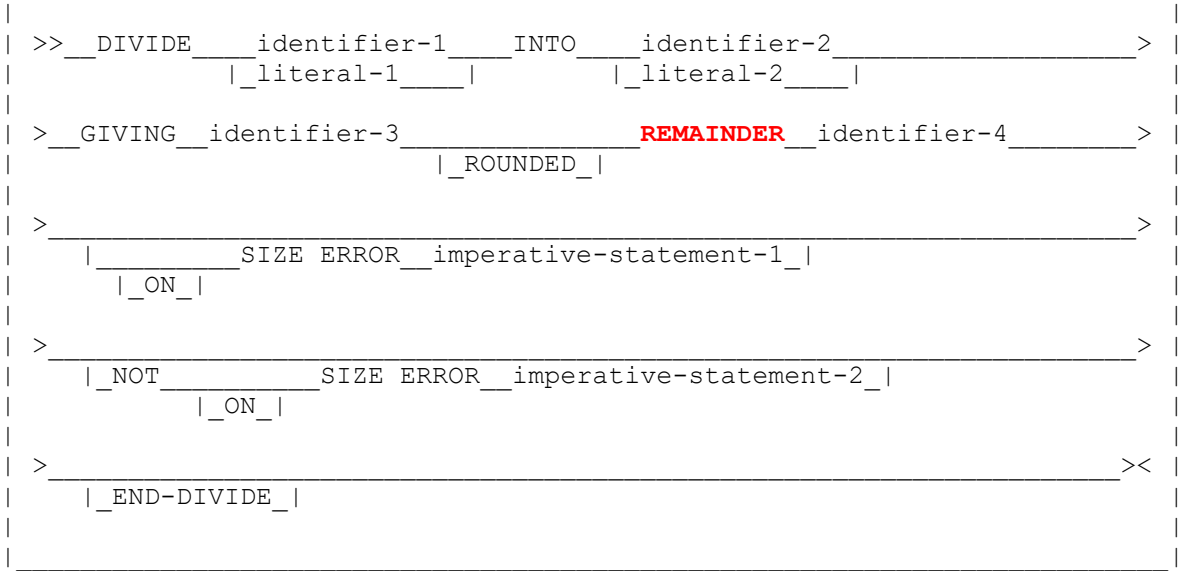
Format 2

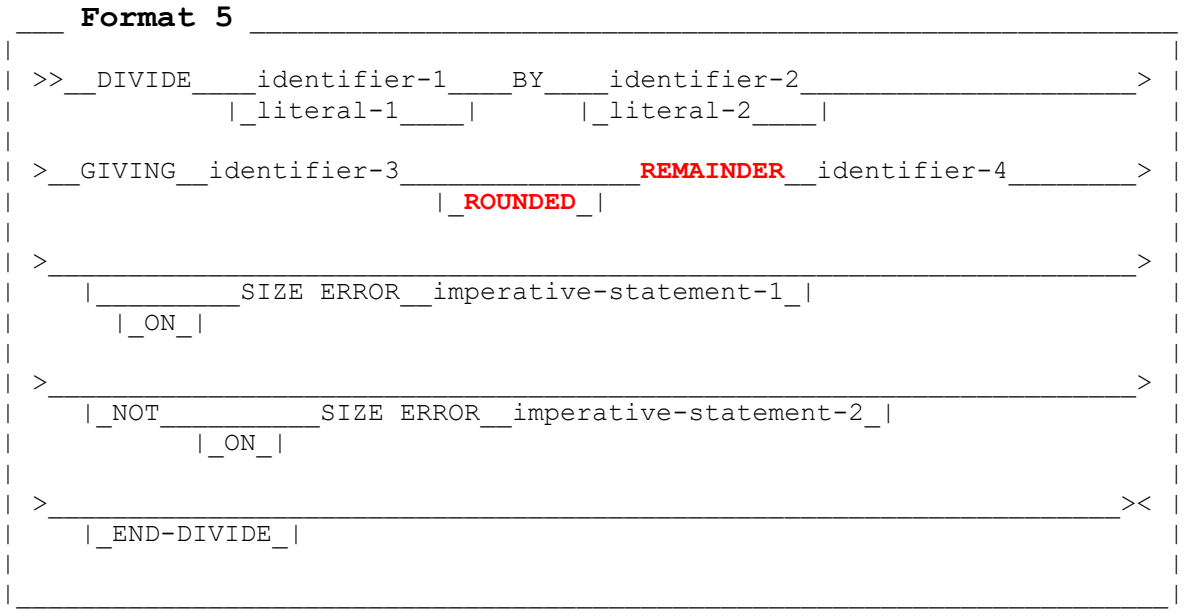


Format 3



Format 4

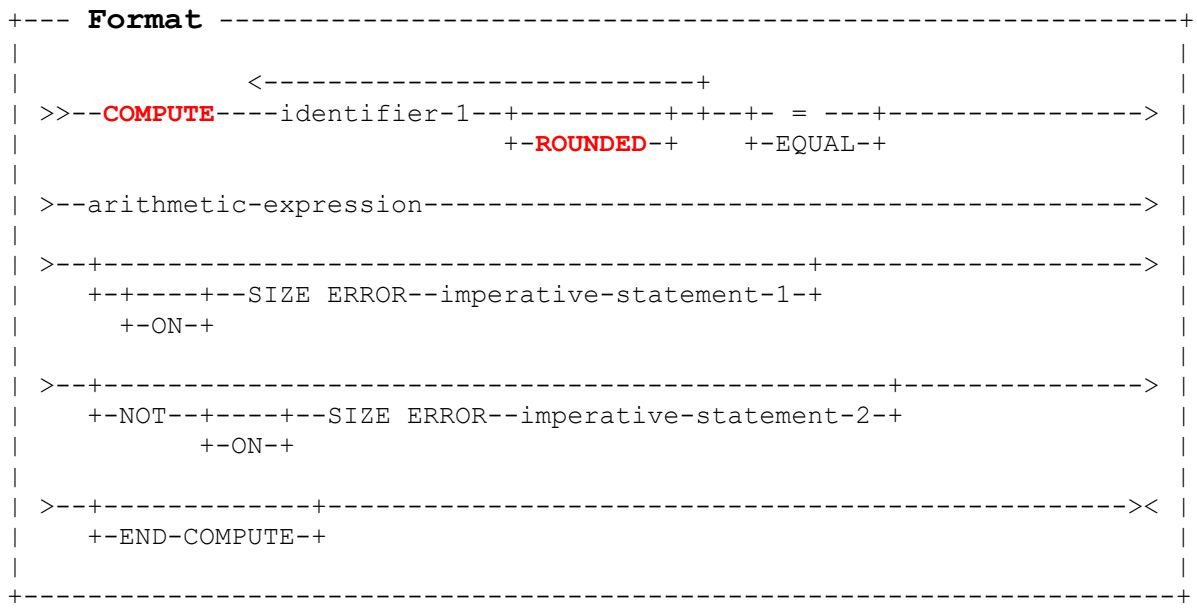




Exemples :

```
DIVIDE A INTO B ---> B = B / A
DIVIDE A BY B GIVING C ---> C = A / B
DIVIDE A INTO B GIVING C ---> C = B / A
          REMAINDER RESTE
```

COMPUTE



L'expression arithmétique peut utiliser les symboles:

\+ - * / **

Ordre de priorité des opérateurs :

1. parenthèses des plus internes aux plus externes
2. opérateurs unaires (+ et -)
3. puissance (**)
4. multiplication et division (* et /)
5. addition et soustraction (+ et -)

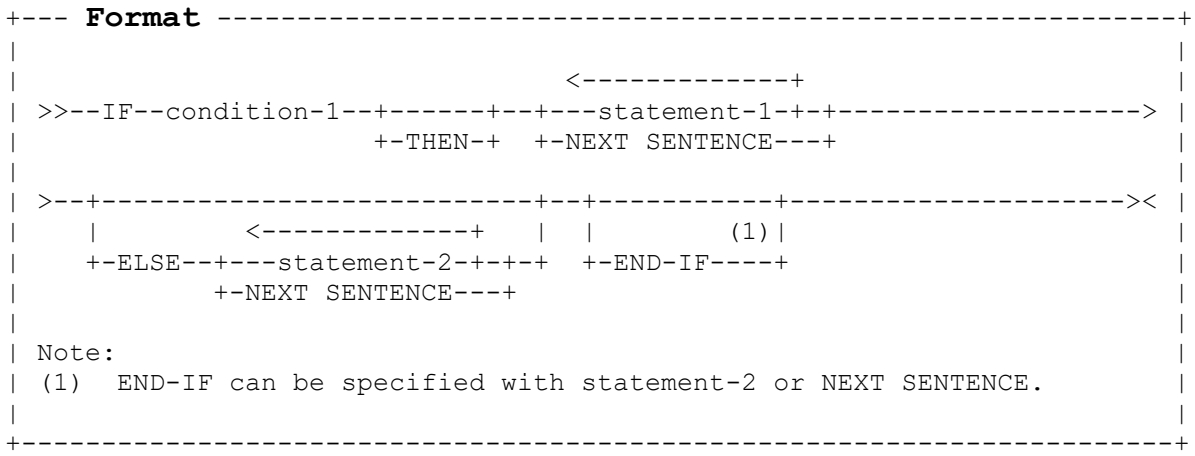
Exemple : COMPUTE A = ((B * C) + D) / (E - 10)

IF...THEN...ELSE...END-IF

PERFORM...UNTIL...END-PERFORM

EVALUATE...WHEN...END-EVALUATE

GO...TO...



CONTINUE ou NEXT SENTENCE ?

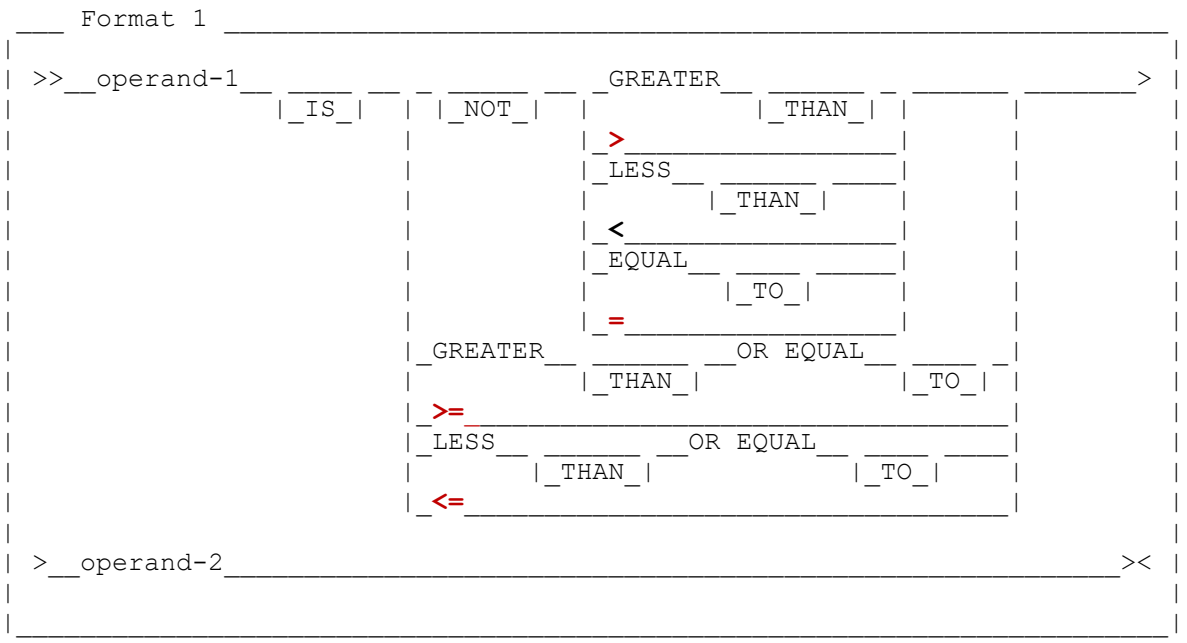
Exemple:

```
IF REPONSE = 'OUI'
   DISPLAY 'OK'
 ELSE
   DISPLAY 'NOT OK'
END-IF
```

Imbrication possible

```
IF condition-1
    IF condition-2
         instruction-1
         instruction-2
     ELSE
         instruction-3
    END-IF
         instruction-4
END-IF
```

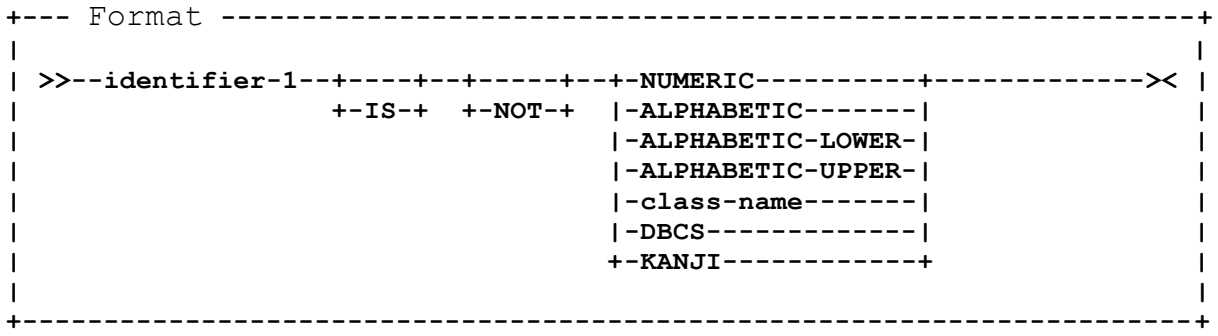
CONDITION DE COMPARAISON



- Si tous les opérandes numériques : comparaison algébrique (conversions automatiques)
- Si un opérande non numérique : comparaison bit à bit de gauche à droite (pas de conversion)

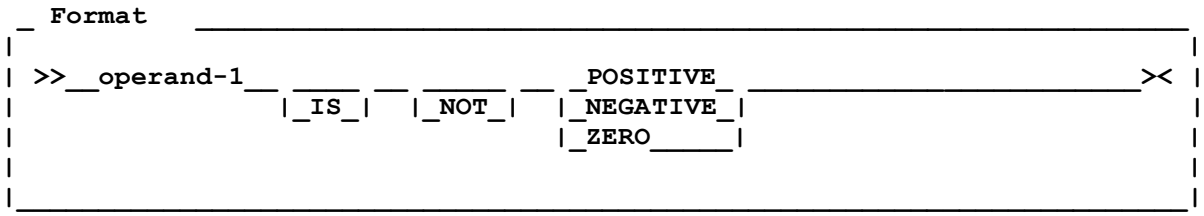
Exemple:

```
IF SEXE = 'M'
   MOVE 'MONSIEUR' TO LIBELLE
ELSE
   IF NB-ENFANTS > 0
      MOVE 'MADAME' TO LIBELLE
   ELSE
      MOVE 'MADEMOISELLE' TO LIBELLE
   END-IF
END-IF
```



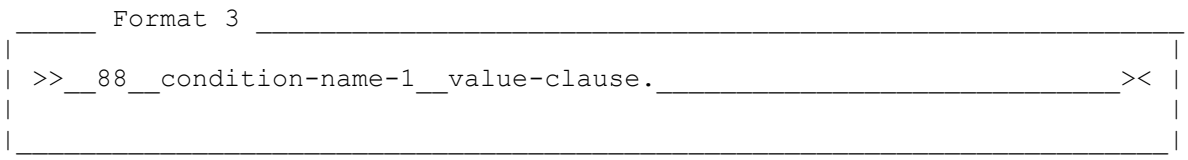
Exemple

```
IF ZONE NOT NUMERIC
   ------------------
   ------------------
   ------------------
END-IF
```



L'opérande doit être numérique

A définir en DATA DIVISION

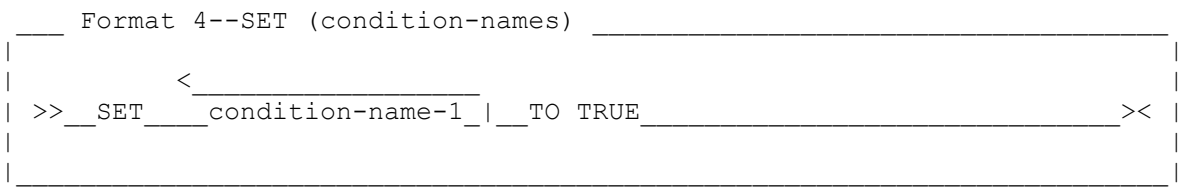


Niveau 88 obligatoire
Doit suivre immédiatement la donnée à conditionner
Plusieurs niveaux 88 autorisés

Exemple:

```
01  AGE-GROUP      PIC  99.
    88  INFANT              VALUE 0.
    88  BABY                VALUE 1, 2.
    88  CHILD               VALUE 3 THRU 12.
    88  TEEN-AGER           VALUE 13 THRU 19.


IF INFANT...            (Tests for value 0)
IF BABY...              (Tests for values 1, 2)
IF CHILD...             (Tests for values 3 through 12)
IF TEEN-AGER...         (Tests for values 13 through 19)
```

SET nom condition



CONDITIONS COMPLEXES

Les conditions peuvent être combinées au moyen de trois opérateurs logiques et de parenthèses

- **NOT** : négation logique ou inverse ou complément
- **AND** : intersection logique ou produit logique
- **OR** : réunion logique ou somme logique

Exemples :

```
IF A > B AND B > C AND C > A
 DISPLAY 'ETRANGE !'
END-IF

IF SEXE = 'M' AND NB-ENF > 0 OR NOM = 'NOEL'
   MOVE 'PAPA' TO LIBELLE
END-IF
```

Attention à l'ordre d'évaluation :

1. parenthèses en commençant par la plus intérieure
2. NOT
3. AND
4. OR

Opérateurs de même rang évalués de la gauche vers la droite

Représentation interne des données

DIFFERENTS MODES DE REPRESENTATION

Rappel

Physiquement

- **Bit**
- **Octet**
- **Code binaire**

Logiquement :

- **Hexadécimal**
- **Décimal**
- **EBCDIC**

Binaire	1111 0010
Hexadécimal	F2
Décimal	242
EBCDIC	2

DISPLAY

Défaut si aucun usage défini

La valeur de la donnée est stockée sous la forme caractère, c'est à dire que les 8 bits d'un octet constituent un caractère

Numérique : signe dans le dernier demi-octet gauche

USAGE IS DISPLAY est valide pour les types suivants

- Alphabétique
- Alphanumérique
- Alphanumérique édité PIC B (nn)
- Numérique édité
- Décimal externe

Exemple

```
01 ZONE          PIC X   VALUE 'A'.       ---> X'C1'
01 NUM           PIC S999 VALUE +123.     ---> X'F1F2C3' (12C)
```

BINARY ou COMPUTATIONAL ou COMP ou COMP-5 (de préférence)

Représentation binaire (données numériques)

Occupe 2, 4 ou 8 octets:

PIC 9(1) à 9(4) : 2 octets
PIC 9(5) à 9(9) : 4 octets
PIC 9(10) à 9(18) : 8 octets

Signe : bit de gauche (complément à 2 pour le négatif)

Exemple

```
01 NUM    PIC S9(3) BINARY VALUE +10.
          --> 0 0 0 A
```

PACKED-DECIMAL ou COMPUTATIONAL-3 ou COMP-3

Représentation décimale condensée (données numériques)

- 2 chiffres par octet
- Signe dans le dernier demi-octet de droite
- Nombre impair de chiffres et signe recommandés

Exemple

```
01 NUM PIC S9(3) PACKED-DECIMAL VALUE +10.
          ---> 0 1 0 C
```

Exemple

NUMERIQUE ETENDU

```
01  Z1        PIC  9(3)  VALUE 123.   F1F2F3
01  Z1        PIC S9(3)  VALUE 123.   F1F2F3
01  Z1        PIC S9(3)  VALUE +123.  F1F2C3     12C
01  Z1        PIC S9(3)  VALUE -123.  F1F2D3     12L
```

F : > 0 Par défaut
C : > 0 Spécifié
D : < 0 Spécifié

NUMERIQUE CONDENSE

```
01  Z1        PIC S9(3)  COMP-3                VALUE +123.
01  Z1        PIC S9(3)  PACKED-DECIMAL VALUE +123.
```

Long Z1packée = (Long Z1/2) + 1 = 2 octets

```
01  Z1        PIC 9(n).          =➔  n octets en mémoire
01  Z1        PIC S9(n) COMP-3.  =➔  (n/2) + 1 octets en mémoire
```

EXEMPLES :

```
    01 Z1     PIC S9(4).         VALEUR MAX    :+/- F9 F9 F9 C/D9
    01 Z2     PIC S9(7) COMP-3.  VALEUR MAX    :+/- 99 99 99 9C/D
```

NUMERIQUE BINAIRE

```
01  Z1        PIC S9(4)  COMP    VALUE +123.
01  Z1        PIC S9(4)  BINARY  VALUE +123.
```

COMPUTATIONAL-1 et COMPUTATIONAL-2

Représentation en virgule flottante (simple ou double précision)

- Occupe 4 ou 8 octets
- Pas de PICTURE
- Le premier bit est le signe de la donnée
- Les bits 2 à 8 réprésentent la caractéristique
- Les octets suivants forment la mantisse

CLAUSE USAGE : AUTRES FORMES

DISPLAY-1

Définit une donnée DBCS
Associé à une clause PICTURE contenant le caractère G

INDEX

Pour ranger un index de table
Pas de PICTURE, ni JUSTIFIED, ni BLANK ni VALUE
SYNCHRONIZED pour une meilleure efficacité
Occupe 4 octets

POINTER : 4 octets

Définit une donnée de type pointeur
Mêmes caractéristiques que INDEX

```
01   ptr1       usage is POINTER.
```

PROCEDURE POINTER : 8 octets

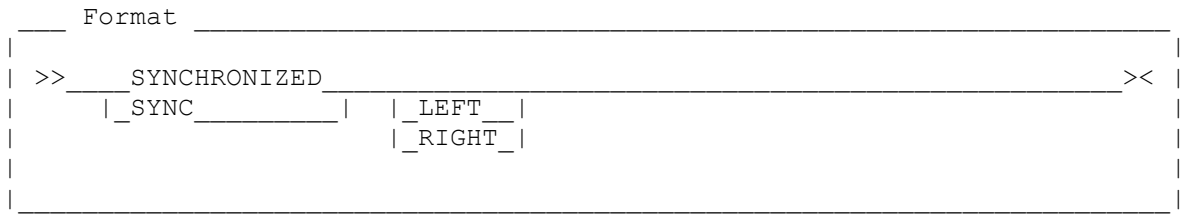
```
01   ptr1       usage is PROCEDURE POINTER.
```

utilisé seulement:

- avec SET
- dans une condition
- dans un CALL BY REFERENCE

Peut être initialisé par NULL

CLAUSE SYNCHRONIZED



Cadrage en mémoire d'une zone à la borne naturelle

Si spécifiée pour donnée groupe, provoque l'alignement de chaque donnée élémentaire du groupe

Recommandé pour les données binaires et pointeurs

TRAITEMENT DES FICHIERS

Types de fichiers utilisables en COBOL:

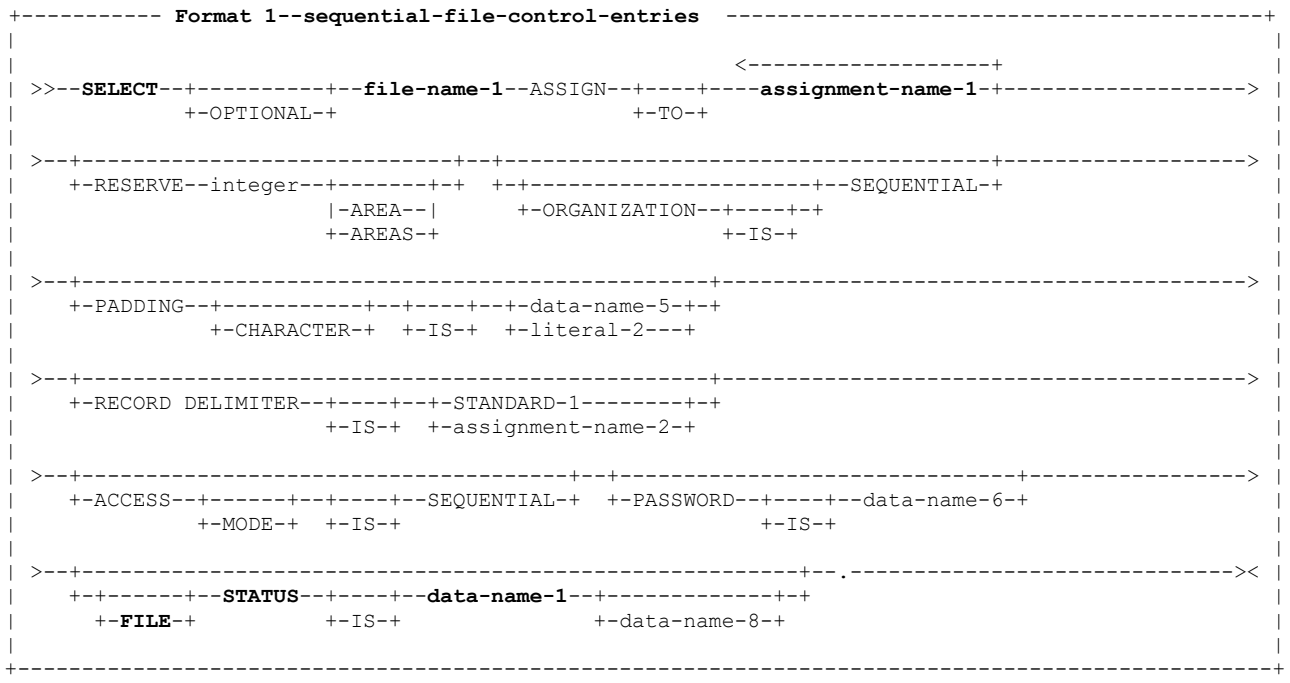
- QSAM
- VSAM

En COBOL : fin de fichier détectée après le dernier enregistrement

8 12
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.

Une clause SELECT par fichier logique du programme

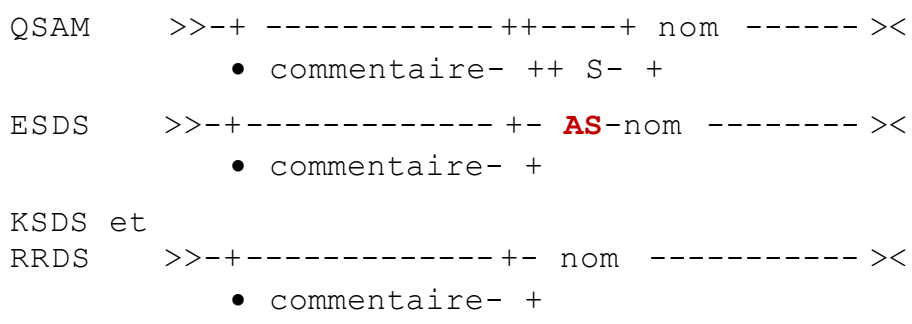
Clause SELECT simplifiée :



CLAUSE SELECT

Nom de fichier : nom COBOL, identifie le fichier dans le programme

Nom externe : fait le lien avec le JCL



Commentaire pour documentation

Pour les fichiers QSAM, le S- peut être omis

Pour les ESDS, AS- doit être spécifié

Le nom doit être de 8 caractères maximum et se retrouver sur une carte DD

CLAUSE FILE STATUS

Première donnée

Code retour valorisé après chaque entrée/sortie

2 caractères alphanumériques en DATA DIVISION

~ premier caractère

- 0 : opération correctement exécutée
- 1 : fin de fichier rencontrée (accès séquentiel)
- 2 : clé invalide (accès direct)
- 3 : erreur permanente (ex: accès à un fichier inexistant)
- 4 : erreur logique (ex: lire un fichier ouvert OUTPUT)
- 9 : erreurs diverses souvent liées à VSAM

~ second caractère

- de 0 à 9 , il apporte une information complémentaire et sa signification varie selon la valeur du 1er caractère

```
File status     pic 99.
```

Exemple :

```
File status = 00      opération d'E/S bien exécutée
File status = 10      Fin de fichier
```

Deuxième donnée

Information VSAM valorisée uniquement pour les fichiers VSAM et si le code retour est différent de zéro

6 octets en DATA DIVISION

- 2 octets binaires : code retour VSAM (0, 8 ou 12)
- 2 octets binaires : code fonction VSAM (0 à 5)
- 2 octets binaires : feedback code VSAM (0 à 255)

Exemple :

```
01 CODE-VSAM.
    05 RETOUR           PIC 99 COMP.
    05 FONCTION         PIC 9 COMP.
    05 FEEDBACK         PIC 999 COMP.
```

DESCRIPTION DE L'ENREGISTREMENT

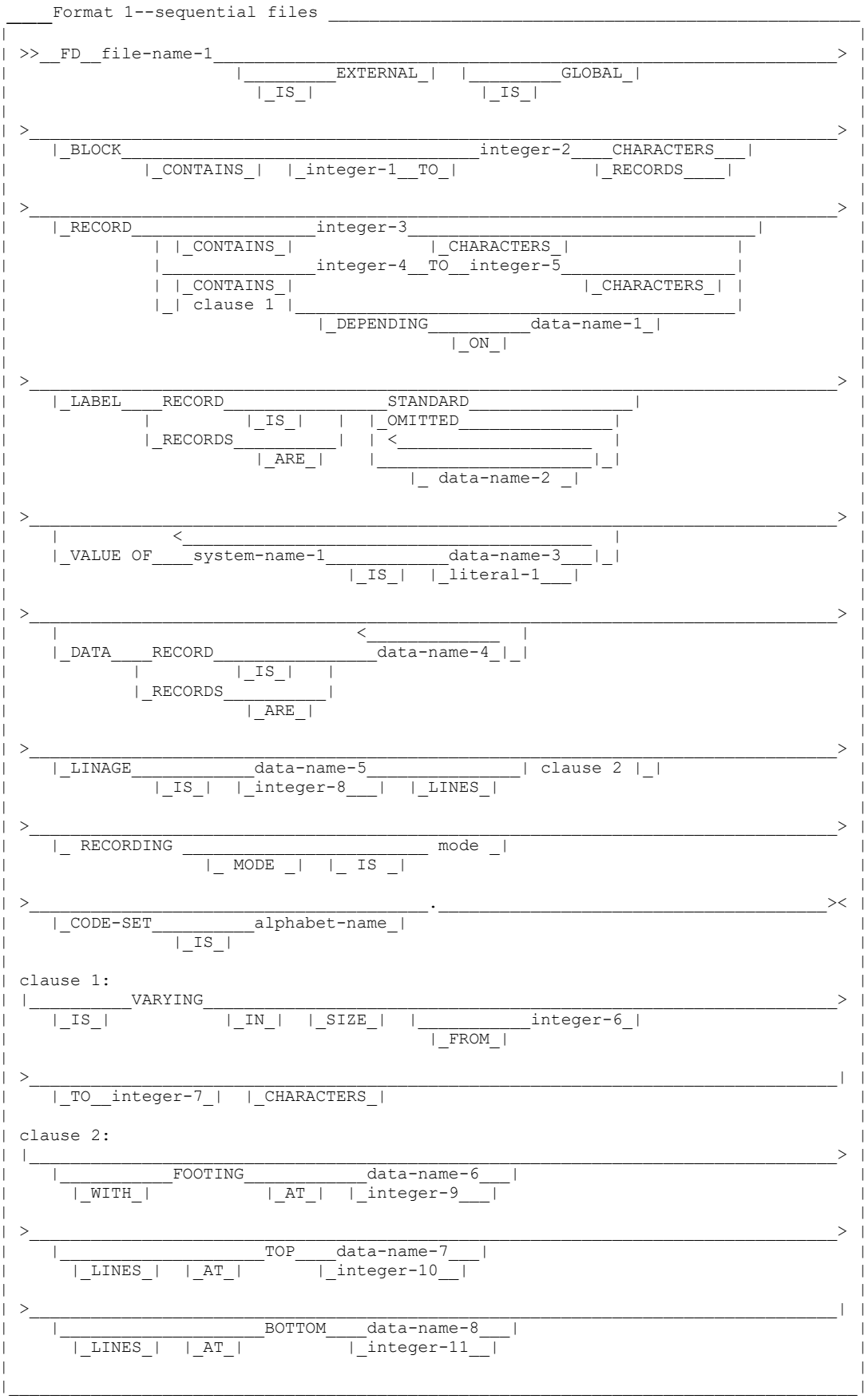
```
8      12
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
```

Une clause SELECT **FICHIER1** ASSIGN FICHJCL. //FICHJCL

```
DATA DIVISION.
FILE SECTION.
FD    FICHIER1 .....
01    E-FICHIER1.
```

Une clause FD par fichier déclaré en ENVIRONMENT DIVISION

Suivie de la description de l'enregistrement (structure de données)


Format 1--sequential files
>>__FD__file-name-1
|_IS_| EXTERNAL_| |_IS_| GLOBAL_|
|_BLOCK integer-2 CHARACTERS_|
|_CONTAINS_| |_integer-1__TO_| |_RECORDS_|
|_RECORD integer-3
|_CONTAINS_| |_CHARACTERS_|
integer-4__TO__integer-5
|_CONTAINS_| |_CHARACTERS_|
|_| clause 1 |
|_DEPENDING data-name-1_|
|_ON_|
|_LABEL___RECORD STANDARD
|_IS_| |_OMITTED_|
|_RECORDS_| | <
|_ARE_| |_|
|_ data-name-2 _|
|_VALUE OF___system-name-1 data-name-3___|_|
|_IS_| |_literal-1___|
|_DATA___RECORD data-name-4_|_|
|_IS_|
|_RECORDS_|
|_ARE_|
|_LINAGE data-name-5 | clause 2 |_|
|_IS_| |_integer-8___| |_LINES_|
|_ RECORDING mode _|
|_ MODE _| |_ IS _|
|_CODE-SET alphabet-name_|
|_IS_|
clause 1:
|_VARYING
|_IS_| |_IN_| |_SIZE_| |_integer-6_|
|_FROM_|
|_TO__integer-7_| |_CHARACTERS_|
clause 2:
|_FOOTING data-name-6___|
|_WITH_| |_AT_| |_integer-9___|
|_TOP___data-name-7___|
|_LINES_| |_AT_| |_integer-10__|
|_BOTTOM___data-name-8___|
|_LINES_| |_AT_| |_integer-11__|

CLAUSE FD

FD

En marge A (le reste en marge B)

Nom fichier

Défini en ENVIRONMENT DIVISION (SELECT)

Clause BLOCK

Facteur de blocage
BLOCK 0 pour les fichiers QSAM : le facteur de blocage est pris sur la carte DD.

Clause RECORD

Longueur d'un enregistrement
Documentation et contrôle seulement

Clause LABEL

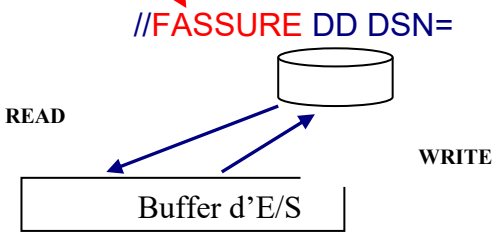
Prise en commentaire en VSAM

Clause DATA RECORD

Nom de l'enregistrement (Optionnel)

EXEMPLE RECAPITULATIF

```
78      12                                                          72
IDENTIFICATION DIVISION.
PROGRAM-ID.  EXEMPLE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
        DECIMAL-POINT IS COMMA.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
        SELECT   ASSURE          ASSIGN     FASSURE
                 FILE STATUS IS CR-ASSURE.
DATA DIVISION.
FILE SECTION.
FD      ASSURE
        BLOCK CONTAINS 0
        DATA RECORD IS E-ASSURE.
01      E-ASSURE                 PIC X(80).
WORKING-STORAGE SECTION.
01      CR-ASSURE                PIC 99.
01       W-ASSURE.
        02  MATRICULE            PIC 9(6).
        02  XXXX
01 DATE-JOUR6          PIC 9(6).
01 DATE-JOUR8          PIC 9(8).
01 TEXTE                PIC X(10) VALUE 'PREMIER ESSAI'.
PROCEDURE DIVISION.
DEBUT.
        ACCEPT DATE-JOUR6 FROM DATE
        ACCEPT DATE-JOUR8 FROM DATE  YYYYMMDD
        OPEN  INPUT ASSURE        (et non FASSURE)
       READ ASSURE
       MOVE E-ASSURE   TO   W-ASSURE
        READ ASSURE   INTO W-ASSURE
        DISPLAY   DATE-JOUR
        DISPLAY   TEXTE.
        CLOSE ASSURE
FIN.
        STOP  RUN.
```



EXEMPLE RECAPITULATIF

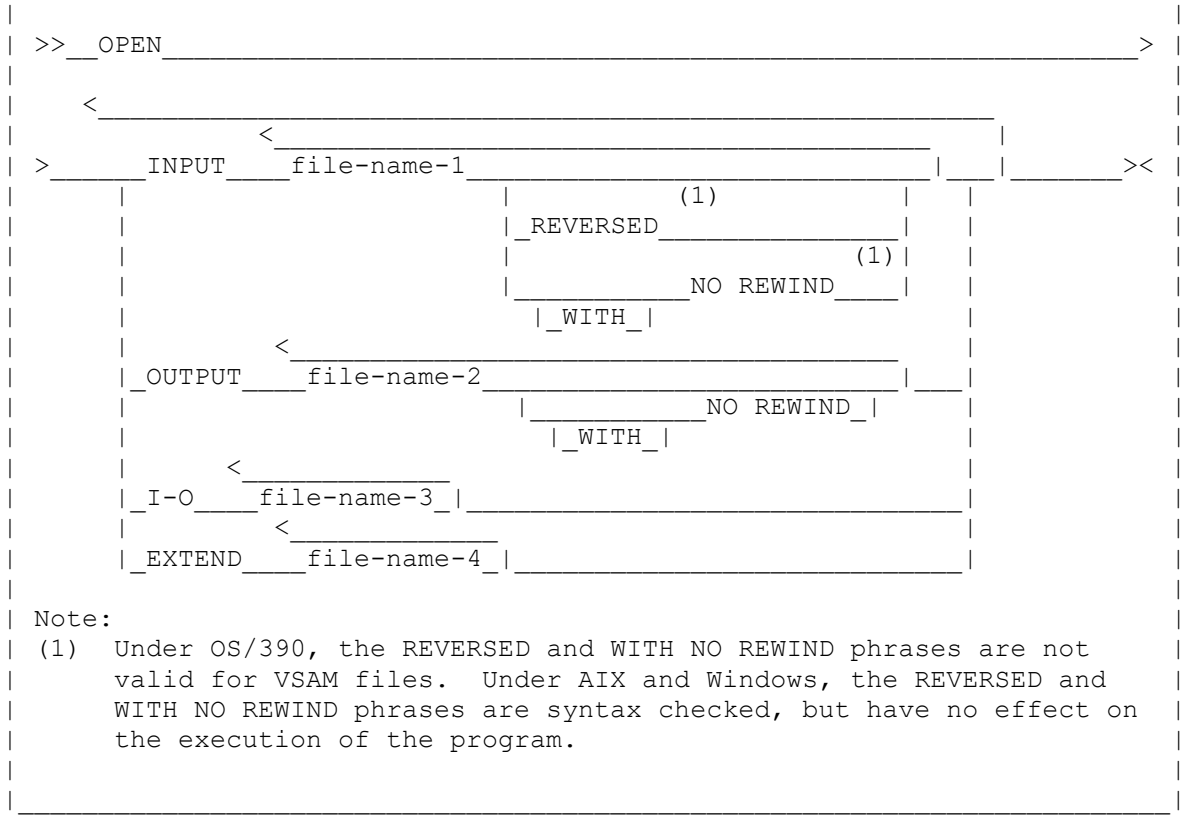
78 12 72

```
       SELECT  F1   ASSIGN        F1
                    FILE STATUS IS CR-F1.
       SELECT  F2   ASSIGN        F2
                    FILE-STATUS IS CR-F2.
       SELECT  F3   ASSIGN        F3
                    FILE-STATUS IS CR-F3.
       SELECT  F4   ASSIGN        F4
                    FILE-STATUS IS CR-F4.

DATA DIVISION.
FILE SECTION.
FD     F1
       BLOCK CONTAINS 0
       DATA RECORD IS E-F1.
01     E-F1                    PIC X(80).
FD     F2
       BLOCK CONTAINS 0
       DATA RECORD IS E-F2.
01     E-F2                    PIC X(230).
FD     F3
       BLOCK CONTAINS 0
       DATA RECORD IS E-F3.
01     E-F3                    PIC X(120).
FD     F4
       BLOCK CONTAINS 0
       DATA RECORD IS E-F4.
01     E-F4                    PIC X(80).


//F1 DD DSN=MPF02.FORM.F1,DISP=SHR
//F2 DD DSN=MPF02.FORM.F2,DISP=SHR
//F3 DD DSN=MPF02.FORM.F3,DISP=SHR
//F4 DD DSN=MPF02.FORM.F4,DISP=SHR

       OPEN I-O /EXTEND /INPUT      F1  F2
                        OUTPUT      F3  F4
```

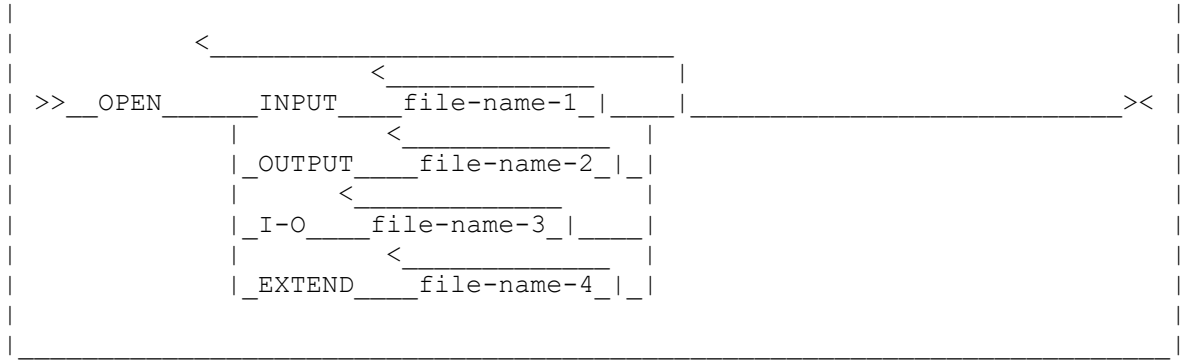
OPERATIONS SUR FICHIERS SEQUENTIELS

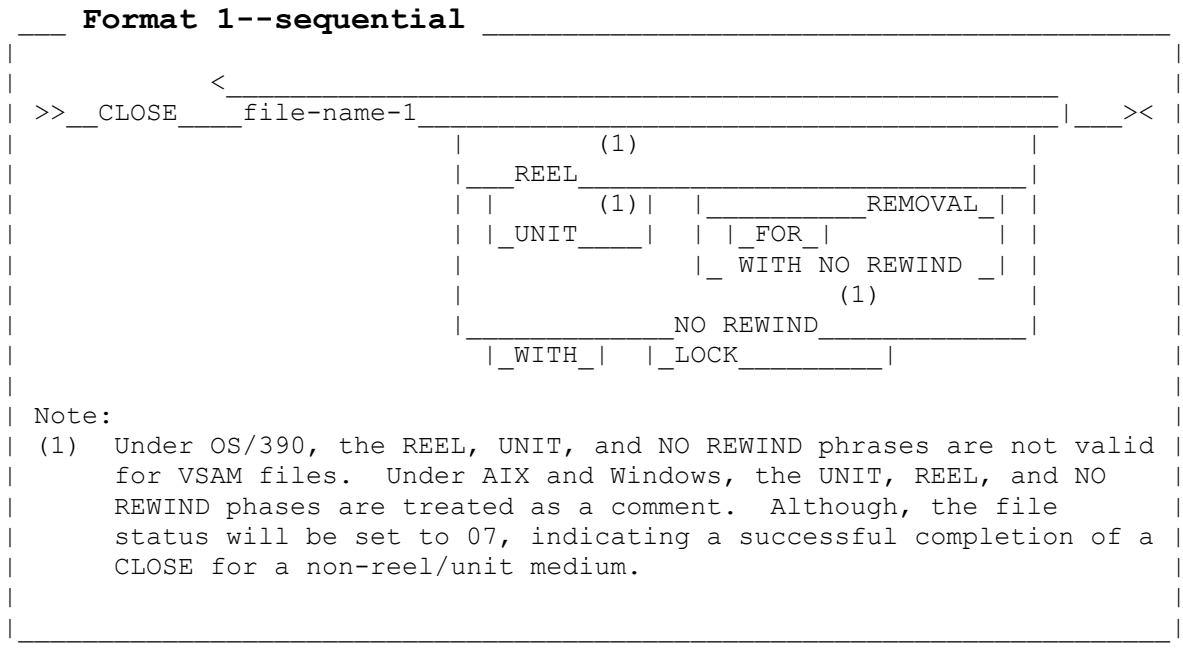
- Ouverture : **OPEN**
- Lecture : **READ**
- Ecriture : **WRITE**
- Fermeture : **CLOSE**

Format 1--sequential files



Format 2--indexed and relative files

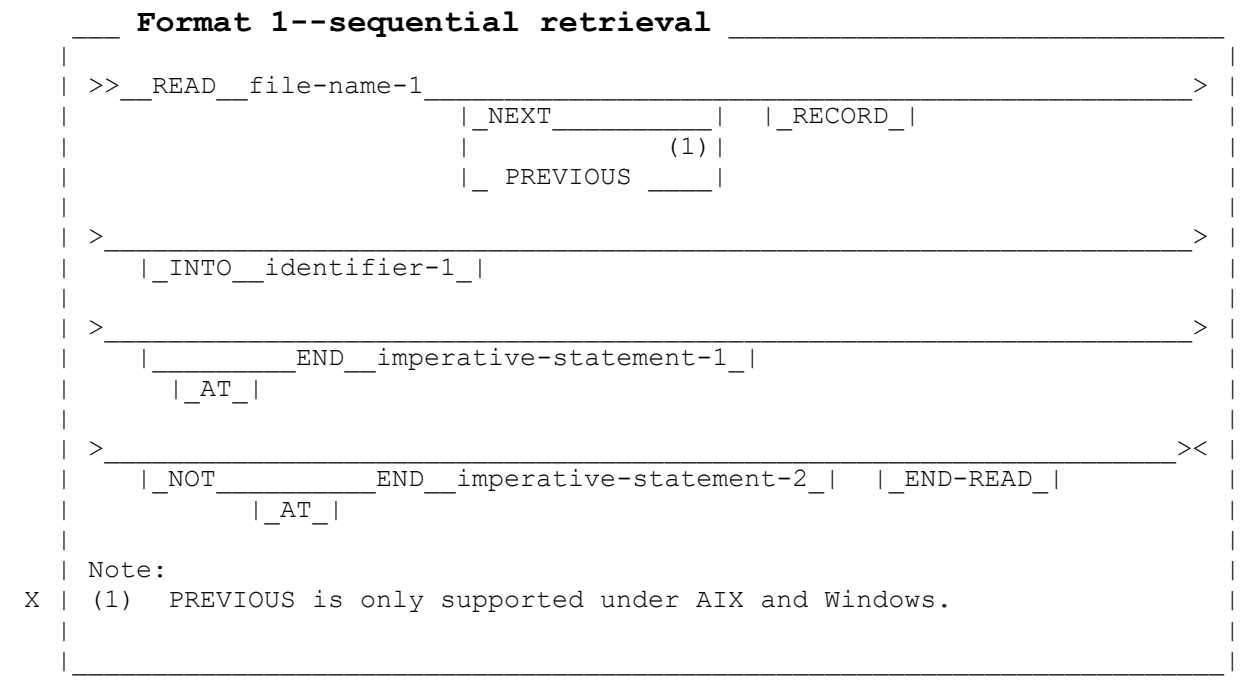




Exemple

CLOSE FIC-ENTREE

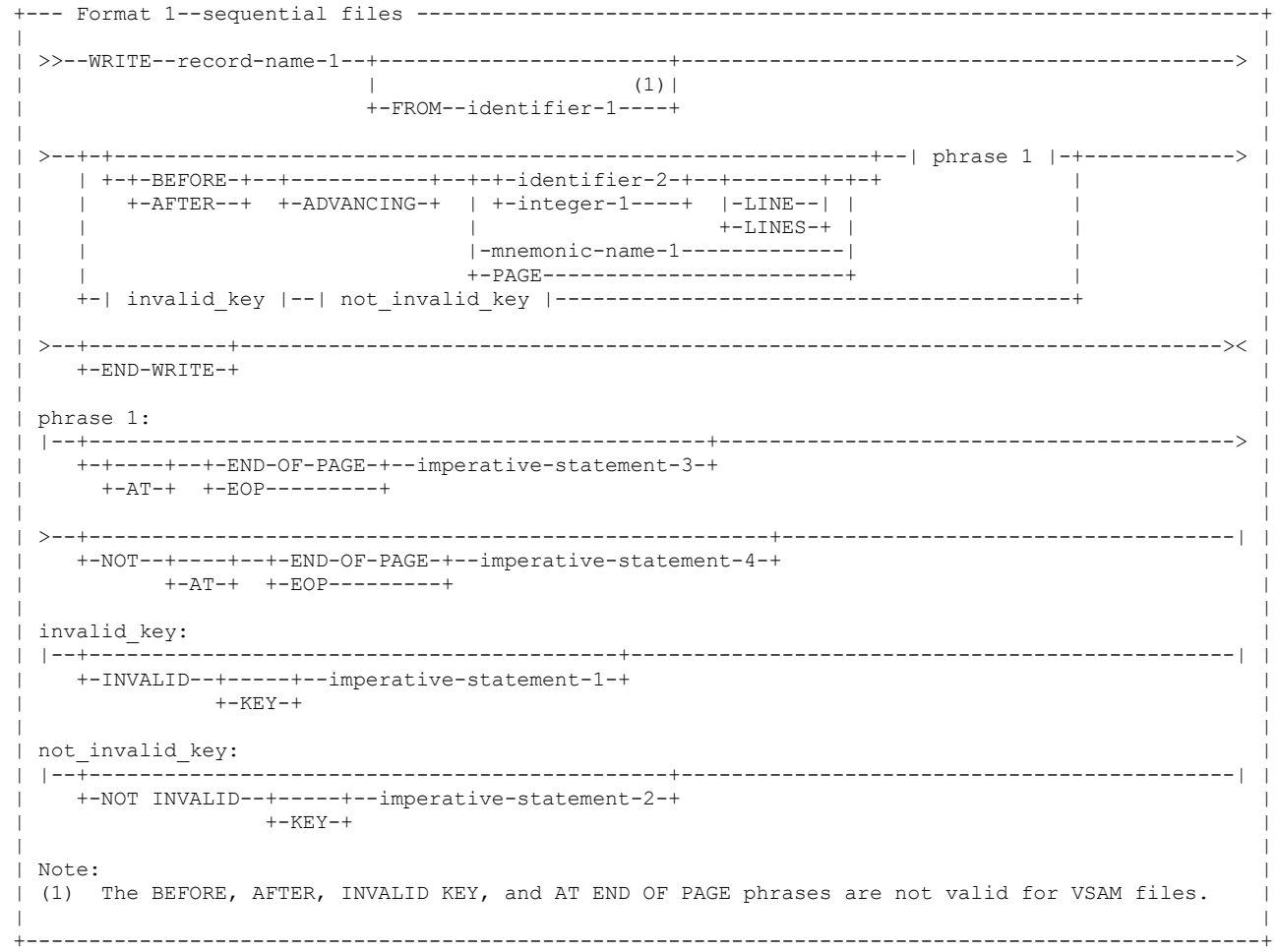
READ : LECTURE SEQUENTIELLE

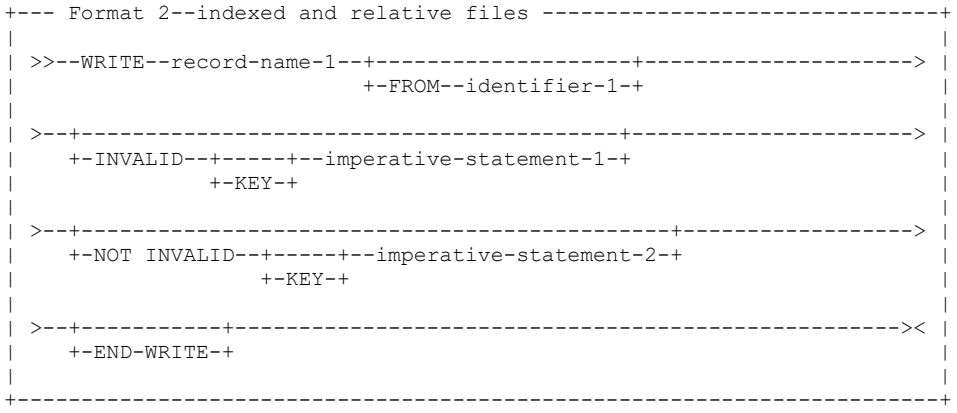


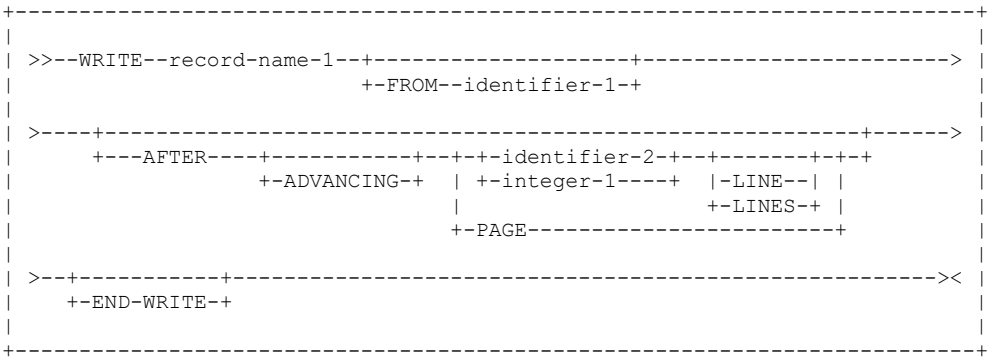
Exemple

```
01  CR                PIC 99.
Perform  until CR = 10
     READ FICHIER    into enregt
END-PERFORM

-----------------------------------------------
01  FIN-FICH         PIC 9.
     88 FIN      VALUE 1.
Move 0    to  fin-fich
Perform  until fin-fich = 1
     READ FICHIER
             AT END MOVE  1  TO  FIN-FICH
             NOT AT  END
                 DISPLAY 'ENGT : ' ENGT
     END-READ
END-PERFORM

-----------------------------------------------
Perform  until FIN
     READ FICHIER
             AT END
                 SET FIN TO TRUE =➔(fin-fich = 1)
             NOT AT  END
                 DISPLAY 'ENGT : ' ENGT
     END-READ
 END-PERFORM
```

WRITE : ECRITURE SEQUENTIELLE







Attention :

- après WRITE le contenu de l'article nom-article-1 disparaît
- par contre identificateur-1 garde son contenu

Exemple :

```
WRITE ENREG-SORTIE FROM ZONE-SORTIE
```

EXEMPLE COMPLET DE TRAITEMENT SEQUENTIEL

```
ID DIVISION.
PROGRAM-ID. TRAITSEQ.
*********************************************
* ECTURE D'UN FICHIER SEQUENTIEL ET RECOPIE  *
* SUR UN AUTRE FICHIER SEQUENTIEL            *
*********************************************

ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT FIC-ENTREE ASSIGN FIC1.
    SELECT FIC-SORTIE ASSIGN FIC2.
DATA DIVISION.
FILE SECTION.
FD FIC-ENTREE.
01 ENREG-ENTREE.
  05 FILLER PIC X(50).
FD FIC-SORTIE.
01 ENREG-SORTIE.
   05 FILLER     PIC X(50).

WORKING-STORAGE SECTION.
01 ETAT-BOUCLE      PIC 9.
    88 FIN-BOUCLE   VALUE 1.
    88 PAS-FIN-BOUCLE VALUE 0.

PROCEDURE DIVISION.
DEBUT.
      OPEN INPUT FIC-ENTREE
           OUTPUT FIC-SORTIE
     READ FIC-ENTREE
                  AT END DISPLAY 'FICHIER VIDE'
                              NOT AT END
                   SET PAS-FIN-BOUCLE TO TRUE
              PERFORM TRAIT-FIC
    END-READ

    CLOSE FIC-ENTREE FIC-SORTIE
    STOP RUN.

TRAIT-FIC.
    PERFORM UNTIL FIN-BOUCLE
         WRITE ENREG-SORTIE FROM ENREG-ENTREE
         READ FIC-ENTREE
              AT END SET FIN-BOUCLE TO TRUE
         END-READ
    END-PERFORM
   DISPLAY 'FIN DE TRAITEMENT'.
```

ÉDITIONS

COMPLEMENT SUR LA DESCRIPTION DES DONNEES

PICTURE d'édition :

- B blanc inséré
- / barre insérée
- 0 zéro inséré
- Z suppression des zéros à gauche
- * protection numérique
- , virgule insérée
- . point décimal
- + signe + ou -
- - signe -
- CR symbole CR si négatif
- DB symbole DB si négatif
- $ signe monétaire

Exemples :

```
01   DATEX          PIC 9(8).
01   DATE-FORM      PIC 9999/99/99.
01   MONTANT   PIC +BZZZBZZ9V,99$.
     ACCEPT    DATEX  FROM DATE YYYYMMDD
     MOVE      DATEX     TO   DATE-FORM
     Résultat : 2020/07/07
```

EXEMPLES

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
   DECIMAL-POINT IS COMMA.
```

05 PRIME-DE-BASE	**PIC 9(5)V99.**	**0374055**	**1235678**	**0001267**
05 PRIME	PIC ZZZZZZZ.	**3740**	**1235678**	**12**
05 PRIME	PIC 99.999,99.	**03.740,55**	**12.356,78**	**00.012,67**
05 PRIME	PIC **ZZ.ZZ9,99.**	**3.740,55**	**12.356,78**	**12,67**
05 PRIME	PIC **.**9,99.	***3.740,55**	**12.356,78**	****12,67**
05 PRIME	PIC **ZZBZZ9,99.**	**3 740,55**	**12 356,78**	**12,67**

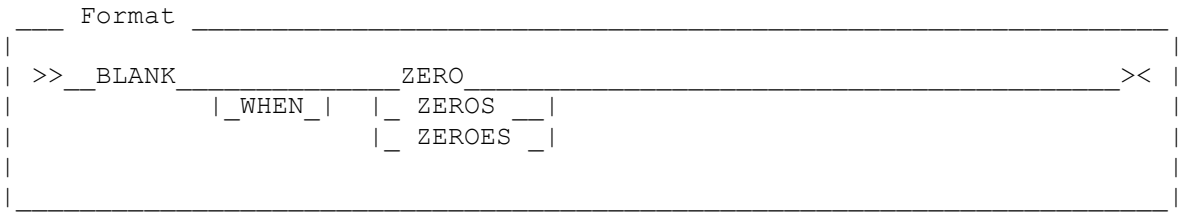
COMPLEMENT SUR LA DESCRIPTION DES DONNEES

Par défaut, marque décimale : point, symbole monétaire : $.

Possibilité de les modifier dans le paragraphe SPECIAL-NAMES.

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    CURRENCY SIGN IS 'F'
    DECIMAL-POINT IS COMMA.
```

BLANK WHEN ZERO



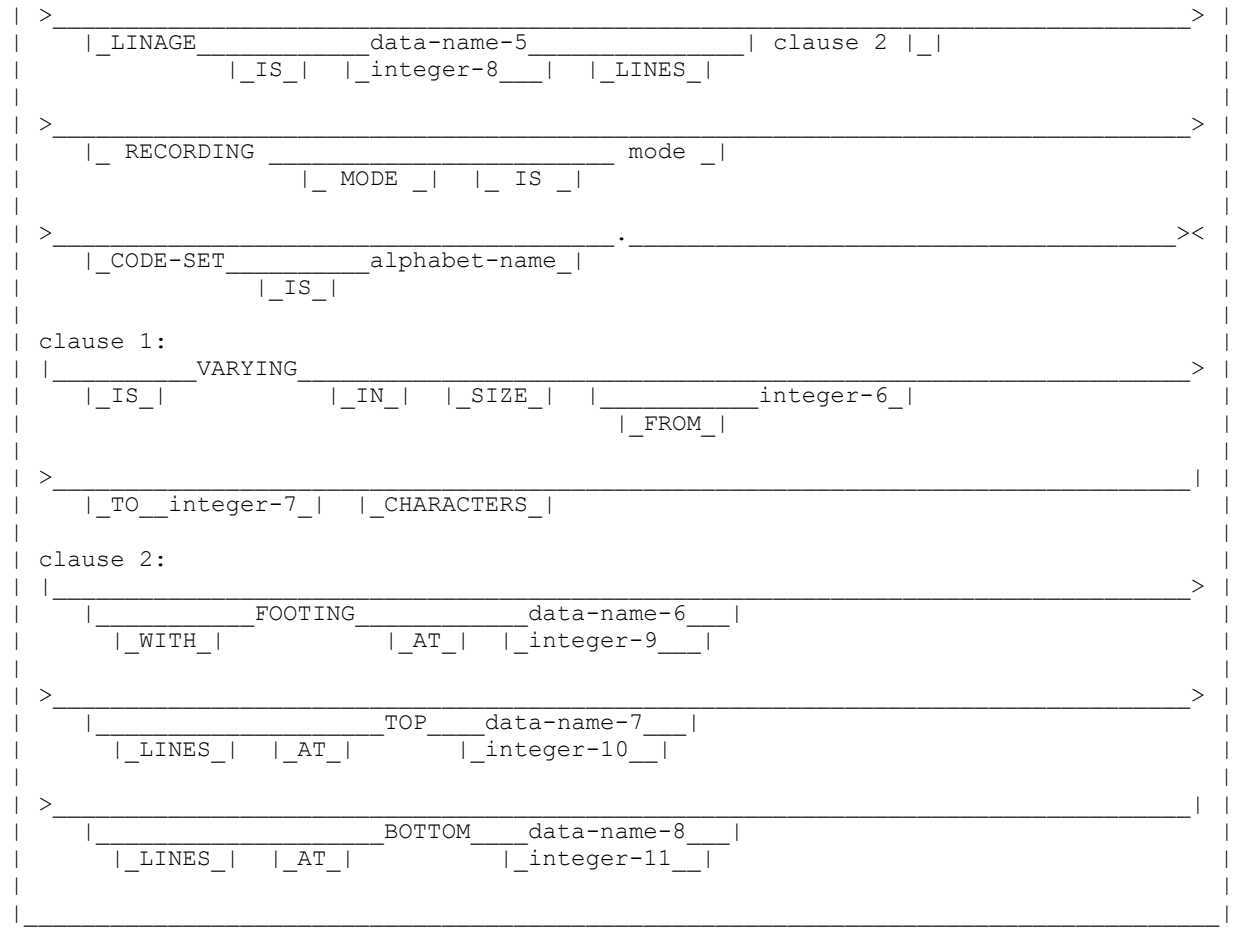
La zone est mise à blanc si elle est entièrement à 0

Seulement pour les données élémentaires

Clause SELECT : identique à celle d'un fichier séquentiel

```
SELECT fic-impr ASSIGN impr-ext.
```

Clause FD : paramètre LINAGE (facultatif)



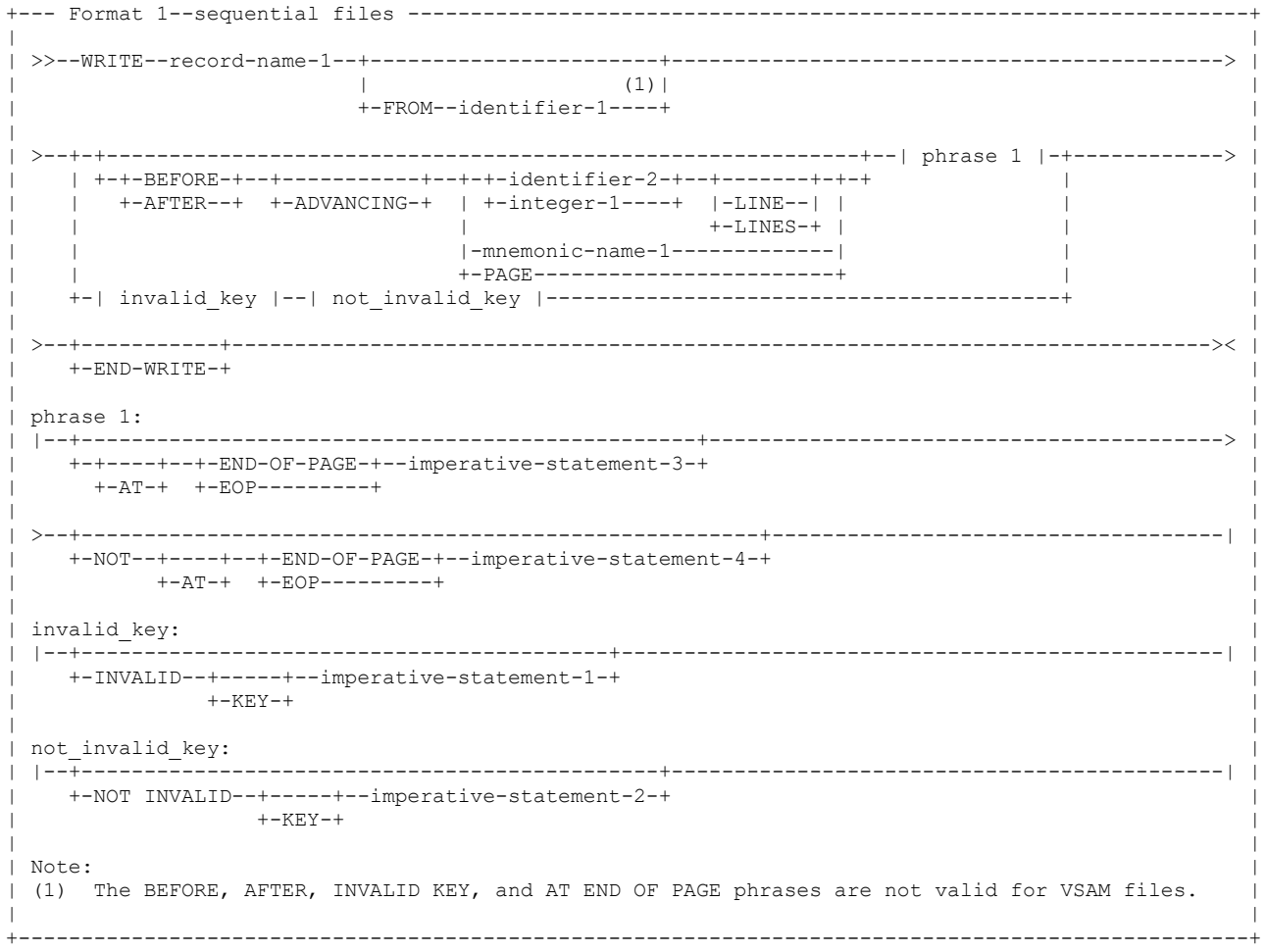

```
FD  fic-impr
    LINAGE nombre de lignes utilisables
    FOOTING numéro de ligne (de la partie LINAGE) activant
            l'indicateur END OF PAGE
    TOP nombre de lignes marge haute
    BOTTOM nombre de lignes marge basse.
01 ligne PIC X(132).
```

Hauteur page = TOP + LINAGE + BOTTOM

LINAGE-COUNTER contient le numéro de la dernière ligne écrite.

Caractère de saut au début de l'enregistrement-ligne (saut de page, ou de 0, 1, 2, 3 interlignes).

Option de compilation ADV/NOADV



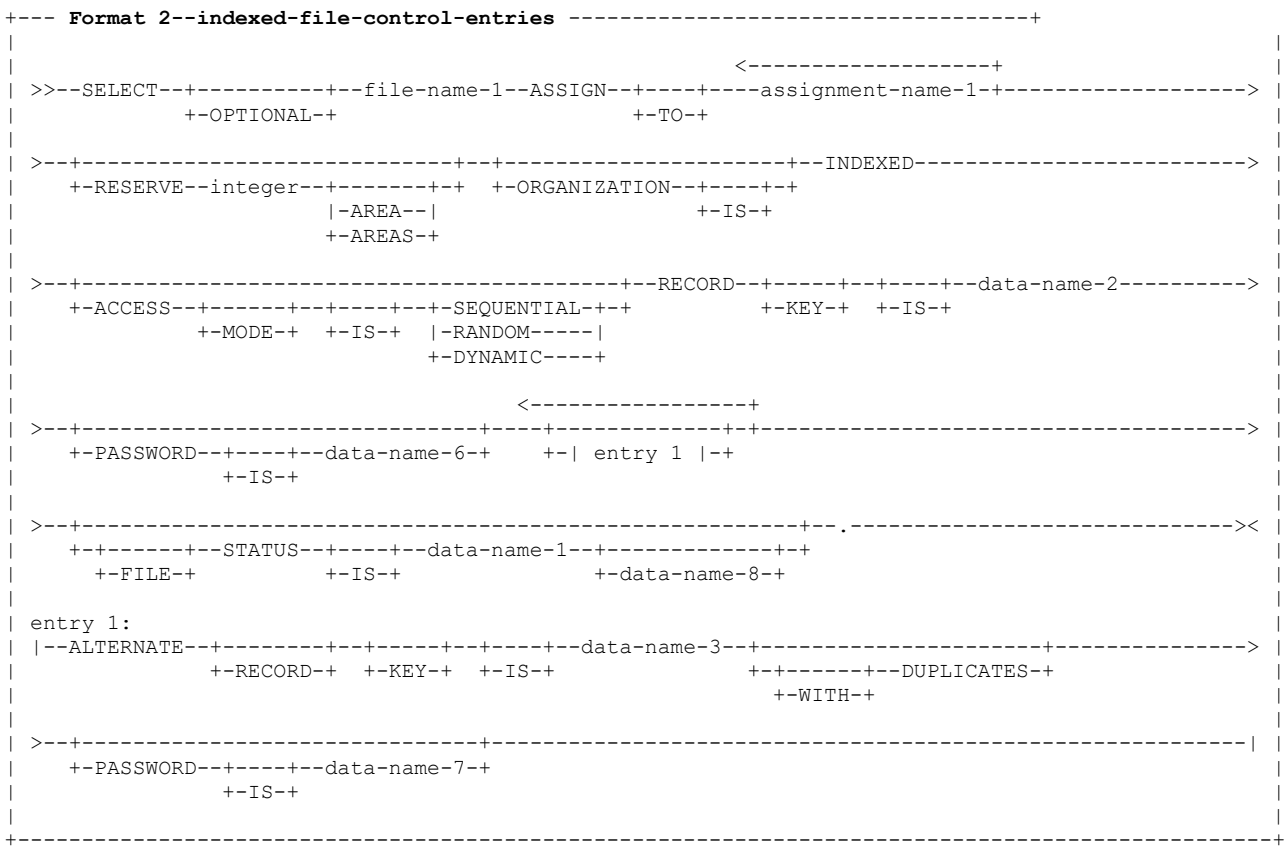
Clause ADVANCING : génère le caractère de saut

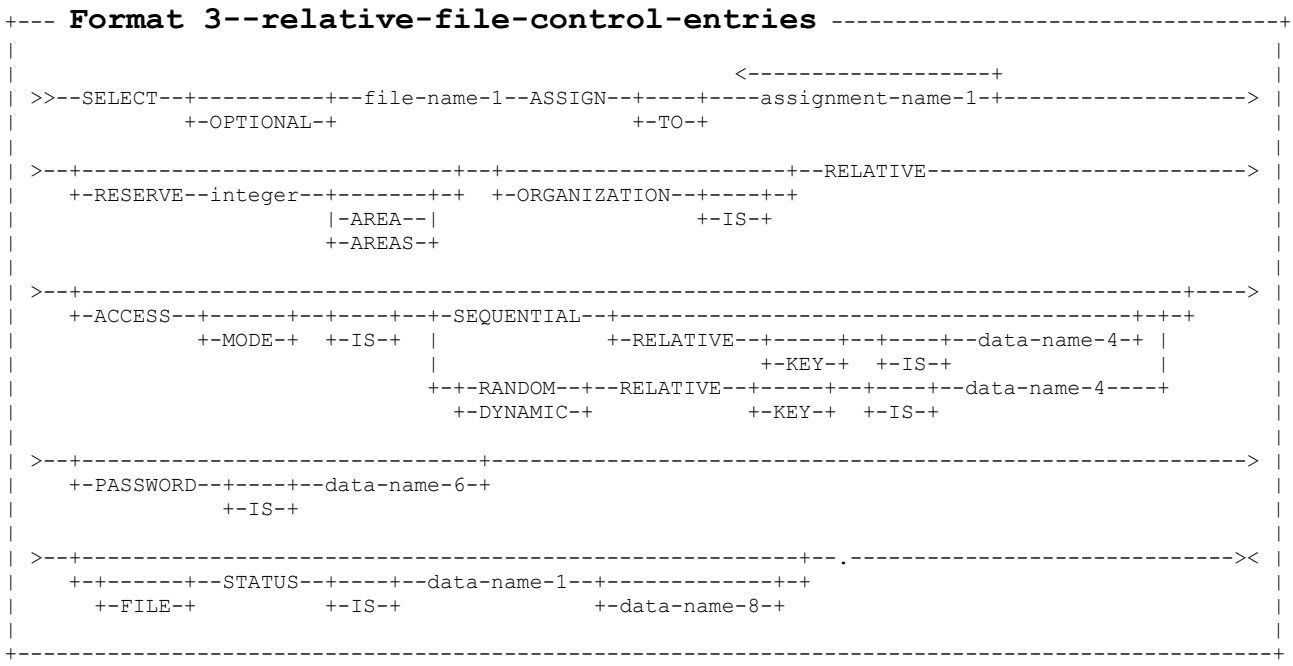
BEFORE : écriture puis saut de ligne (ou de page)

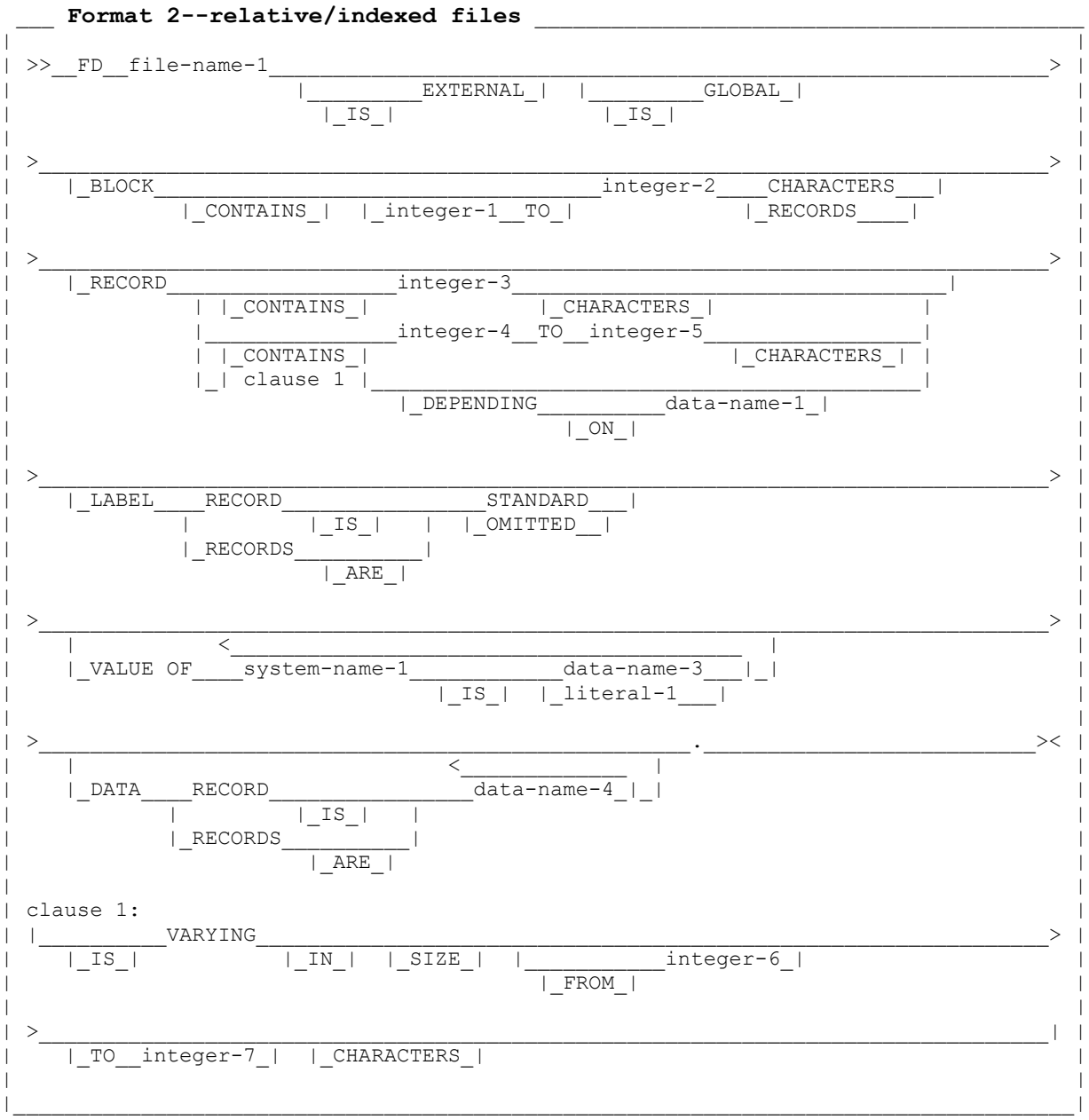
AFTER : écriture après saut de ligne (ou de page)

Condition **END-OF-PAGE** : fonctionne avec le paramètre FOOTING de LINAGE

FICHIERS A ACCES DIRECT

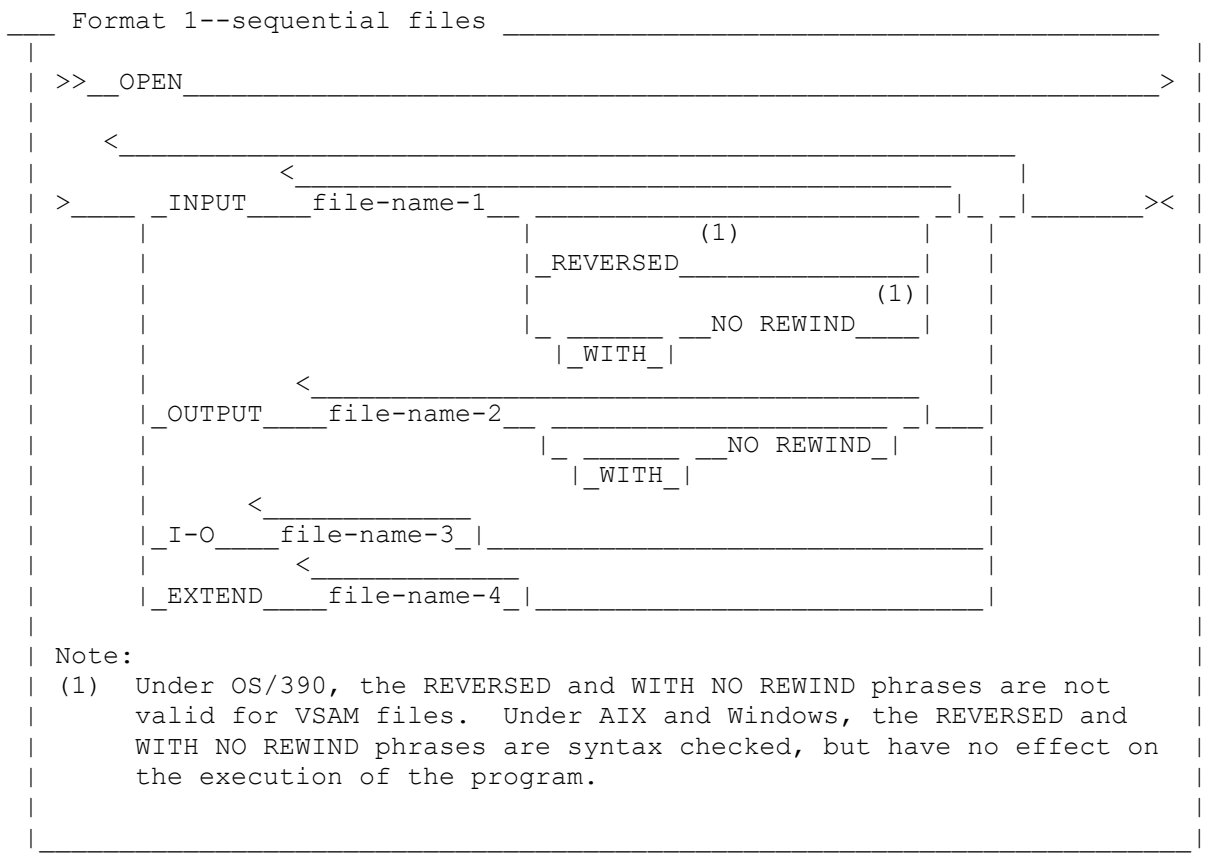

Format 2--indexed-file-control-entries
SELECT
OPTIONAL
file-name-1
ASSIGN
TO
assignment-name-1
RESERVE
integer
AREA
AREAS
ORGANIZATION
IS
INDEXED
ACCESS
MODE
IS
SEQUENTIAL
RANDOM
DYNAMIC
RECORD
KEY
IS
data-name-2
PASSWORD
data-name-6
IS
entry 1
STATUS
FILE
IS
data-name-1
data-name-8
entry 1:
ALTERNATE
RECORD
KEY
IS
data-name-3
DUPLICATES
WITH
PASSWORD
data-name-7
IS

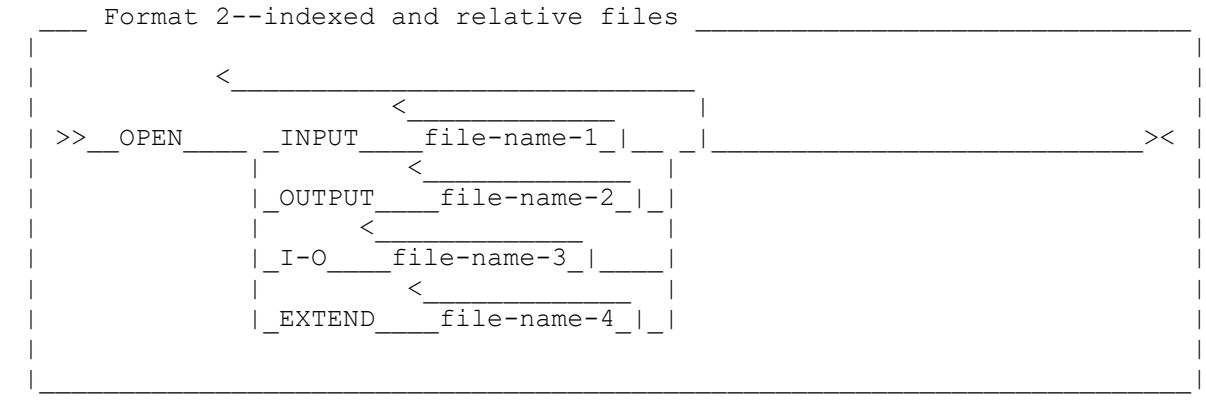

Format 3--relative-file-control-entries
>>--SELECT--+----------+--file-name-1--ASSIGN--+----+----assignment-name-1-+-------------------><
+-OPTIONAL-+ +-TO-+
<-------------------+
>--+-----------------------------+--+----------------------+--RELATIVE-------------------------><
+-RESERVE--integer--+--------+-+ +-ORGANIZATION--+----+-+
|-AREA--| +-IS-+
+-AREAS-+
>--+---+----><
+-ACCESS--+------+--+----+--+-SEQUENTIAL--+---+-+-+
+-MODE-+ +-IS-+ | +-RELATIVE--+-----+--+----+--data-name-4-+ |
| +-KEY-+ +-IS-+ |
+-+-RANDOM--+--RELATIVE--+-----+--+----+--data-name-4----+
+-DYNAMIC-+ +-KEY-+ +-IS-+
>--+-------------------------------+---><
+-PASSWORD--+----+--data-name-6-+
+-IS-+
>--+---+--.-------------------------------><
+-+------+--STATUS--+----+--data-name-1--+-------------+-+
+-FILE-+ +-IS-+ +-data-name-8-+

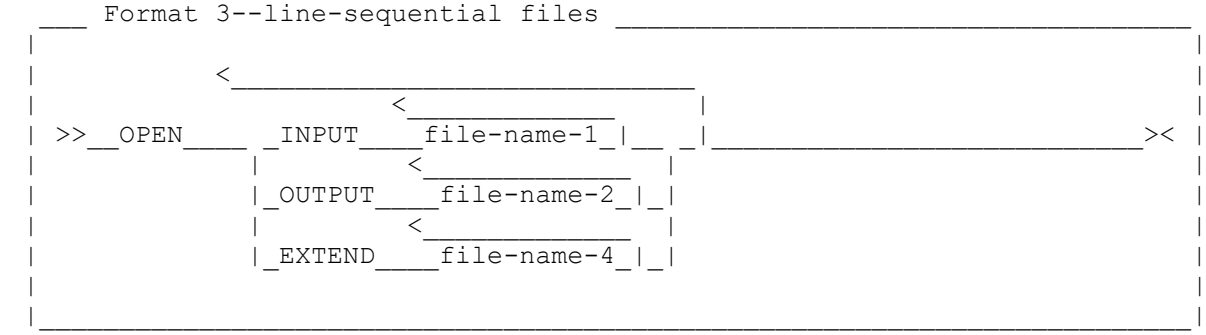

Format 2--relative/indexed files
>>__FD__file-name-1
|_IS_| EXTERNAL_|
|_IS_| GLOBAL_|
|_BLOCK
|_CONTAINS_| |_integer-1__TO_|
integer-2____CHARACTERS___|
|_RECORDS____|
|_RECORD
integer-3
| |_CONTAINS_|
|_CHARACTERS_|
integer-4__TO__integer-5
| |_CONTAINS_|
|_CHARACTERS_|
|_| clause 1 |
|_DEPENDING
data-name-1_|
|_ON_|
|_LABEL____RECORD
STANDARD___|
|_IS_|
|_OMITTED__|
|_RECORDS
|_ARE_|
|_VALUE OF____system-name-1
data-name-3___|_|
|_IS_|
|_literal-1___|
|_DATA____RECORD
data-name-4_|_|
clause 1:
|_VARYING
|_IS_|
|_IN_|
|_SIZE_|
|_integer-6_|
|_FROM_|
|_TO__integer-7_|
|_CHARACTERS_|

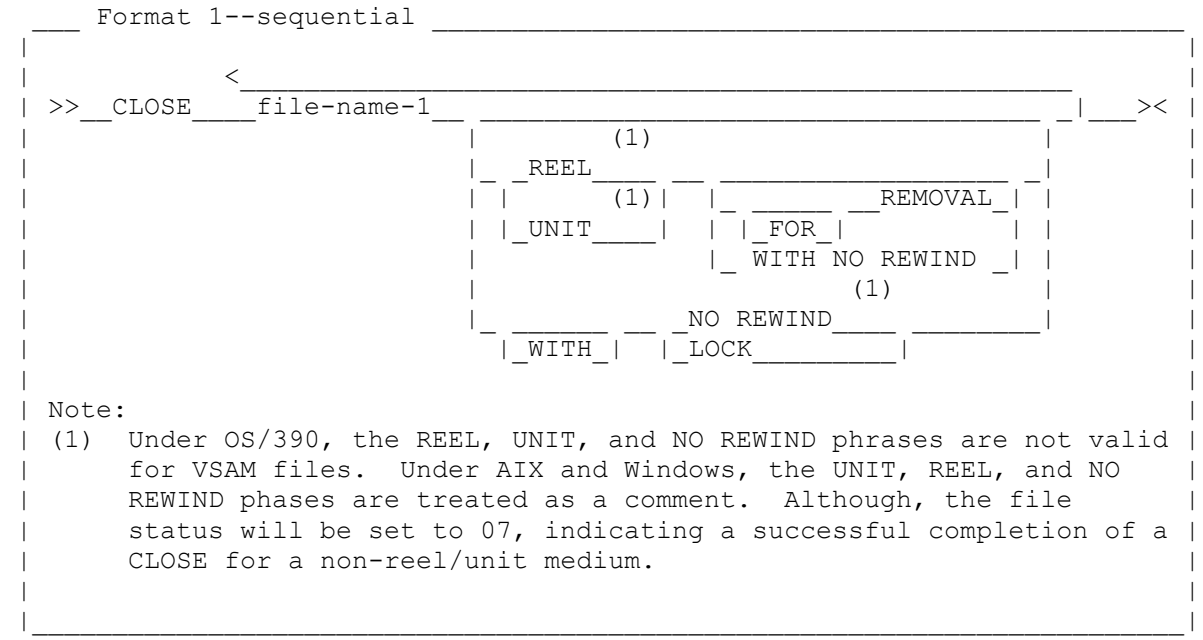
EXEMPLE

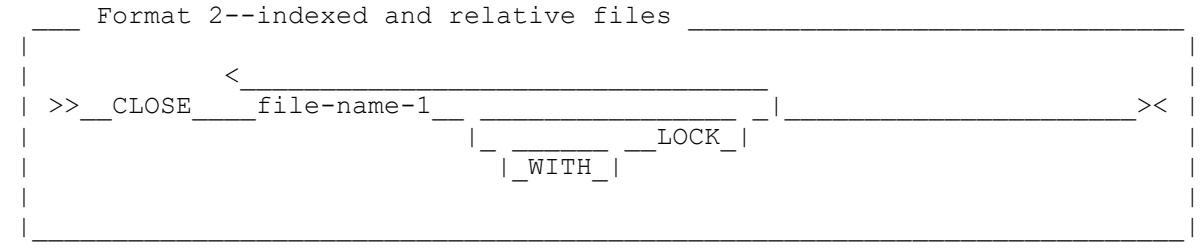
```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT FIC1 ASSIGN FIC1
           ORGANIZATION INDEXED
           ACCESS MODE RANDOM
           RECORD KEY FIC1-CLE
           FILE STATUS CODE-RETOUR CODE-VSAM.


FILE SECTION.
FD FIC1.
01 FIC1-ENREG.
    05 FIC1-CLE  PIC X(4).
    05 FIC1-SUITE    PIC ...


WORKING-STORAGE SECTION.
01 CODE-RETOUR            PIC XX.
01 CODE-VSAM.
    05 RET-VSAM           PIC 99 COMP.
    05 FONCT-VSAM         PIC 9 COMP.
    05 DETAIL-VSAM        PIC 999 COMP.
```


Format 1--sequential files
OPEN
INPUT
file-name-1
REVERSED
(1)
NO REWIND
WITH
OUTPUT
file-name-2
I-O
file-name-3
EXTEND
file-name-4
Note:
(1) Under OS/390, the REVERSED and WITH NO REWIND phrases are not valid for VSAM files. Under AIX and Windows, the REVERSED and WITH NO REWIND phrases are syntax checked, but have no effect on the execution of the program.


Format 2--indexed and relative files
OPEN
INPUT
file-name-1
OUTPUT
file-name-2
I-O
file-name-3
EXTEND
file-name-4


Format 3--line-sequential files
OPEN
INPUT
file-name-1
OUTPUT
file-name-2
EXTEND
file-name-4


Format 1--sequential
>>__CLOSE____file-name-1__
(1)
_REEL____
(1)
_UNIT____
__REMOVAL_
FOR
WITH NO REWIND
(1)
NO REWIND____
WITH
_LOCK_____
><
Note:
(1) Under OS/390, the REEL, UNIT, and NO REWIND phrases are not valid for VSAM files. Under AIX and Windows, the UNIT, REEL, and NO REWIND phases are treated as a comment. Although, the file status will be set to 07, indicating a successful completion of a CLOSE for a non-reel/unit medium.


Format 2--indexed and relative files
>>__CLOSE____file-name-1__
__LOCK_
WITH
><

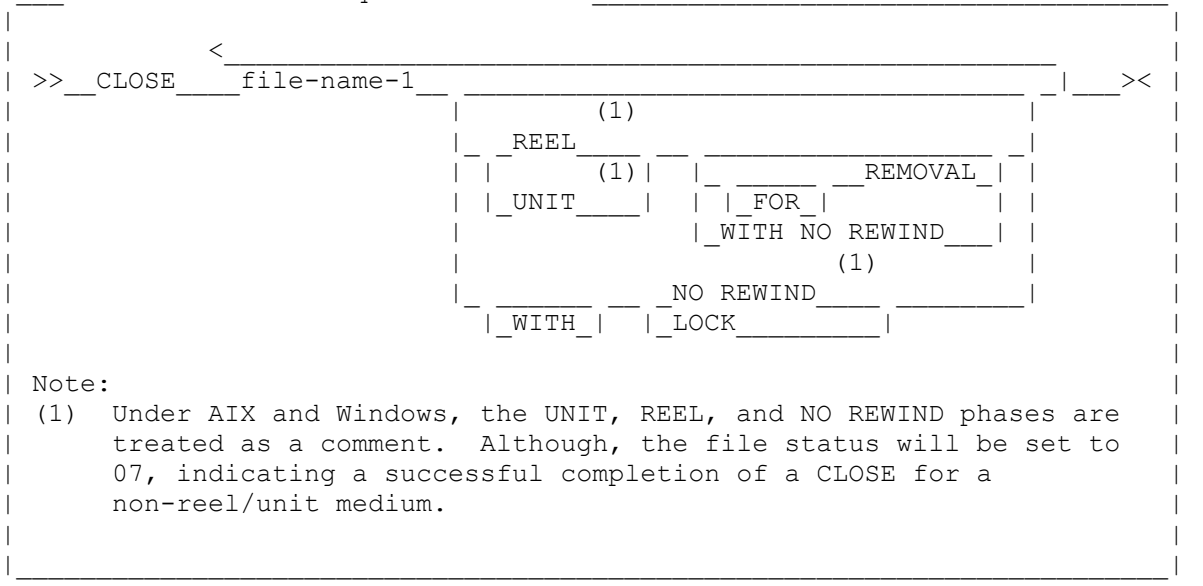
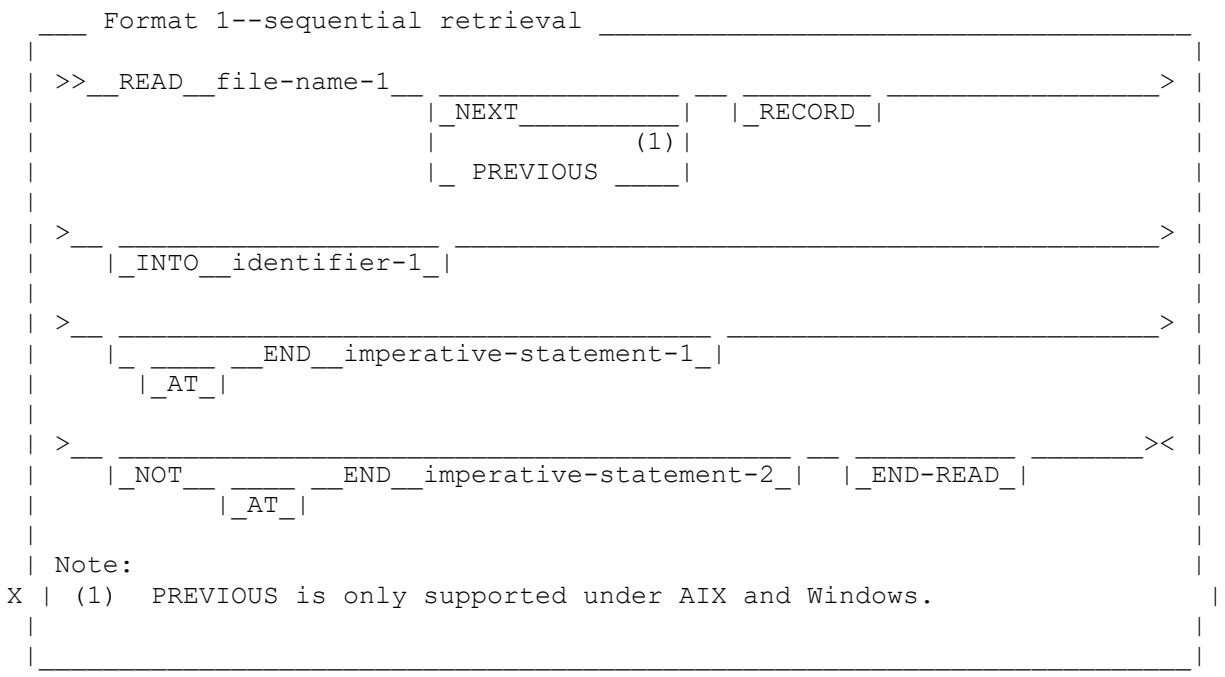

Format 3--line-sequential files
>>__CLOSE____file-name-1__
(1)
_REEL____
(1)
_UNIT____
__REMOVAL_
FOR
WITH NO REWIND
(1)
NO REWIND____
WITH
_LOCK_____
><
Note:
(1) Under AIX and Windows, the UNIT, REEL, and NO REWIND phases are treated as a comment. Although, the file status will be set to 07, indicating a successful completion of a CLOSE for a non-reel/unit medium.

Tableau récapitulatif des instructions autorisées selon le mode d'accès et le type d'ouverture.

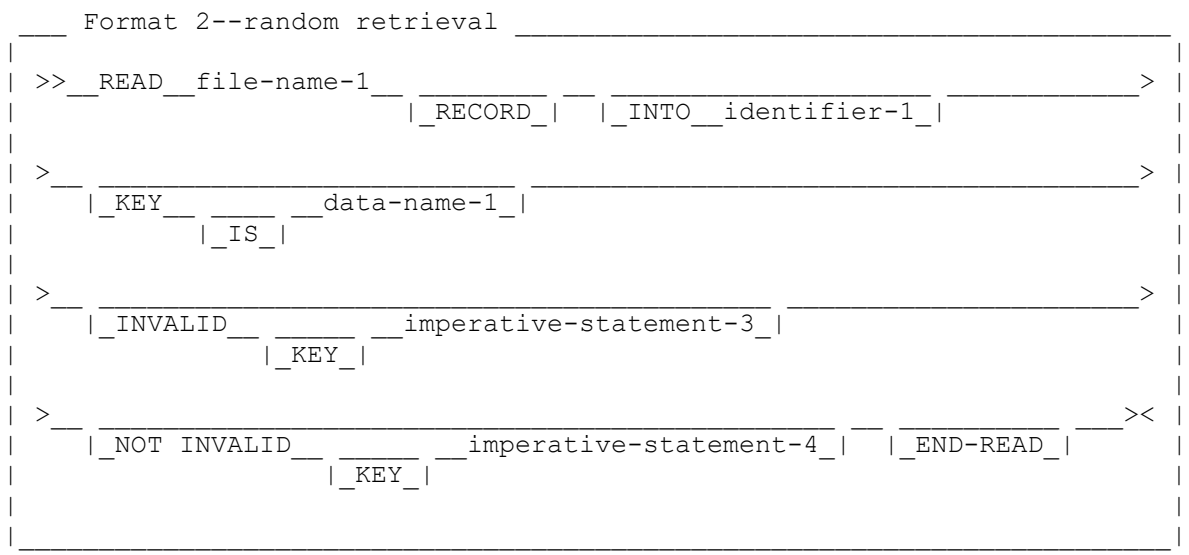
ACCESS / OPEN	RANDOM	SEQUENTIAL	DYNAMIC
INPUT	READ I.K.	READ A.E. START	READ I.K. READ NEXT START
OUTPUT	WRITE I.K. REWRITE DELETE	WRITE REWRITE DELETE	WRITE I.K. REWRITE DELETE
I-O	READ I.K. WRITE I.K. REWRITE DELETE	READ A.E. WRITE REWRITE DELETE START	READ I.K. READ NEXT WRITE REWRITE DELETE START
EXTEND	WRITE	WRITE	WRITE



NEXT : lecture séquentielle en accès DYNAMIC

Exemple

```
READ FIC1 NEXT
   AT END MOVE 1 TO FIN-FIC
END-READ
```

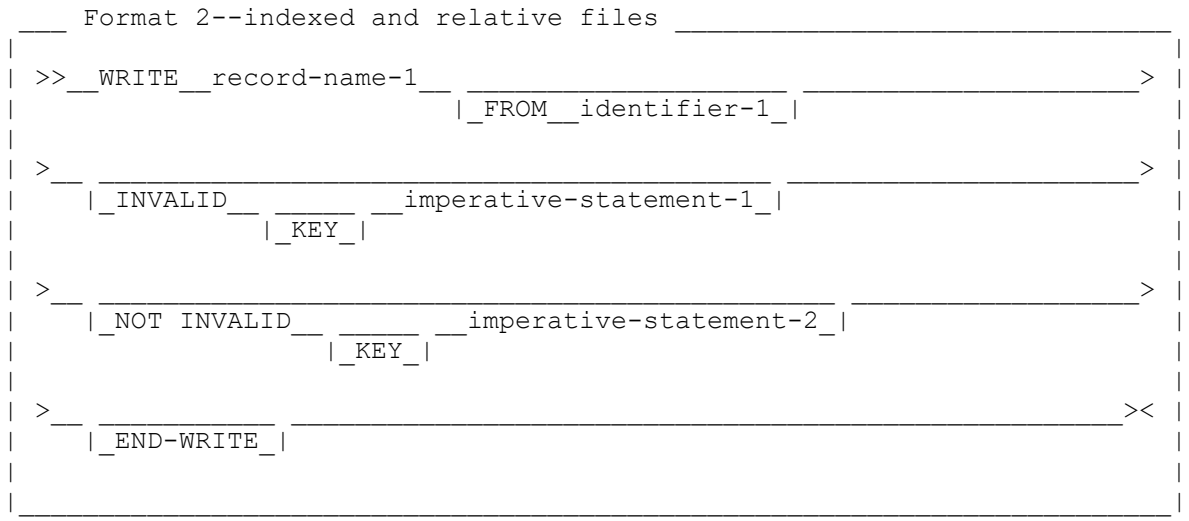


Fonctionnement

- recherche dans le fichier de l'identifiant identique à la valeur de la zone déclarée comme clé (ou comme clé alternative si KEY IS)

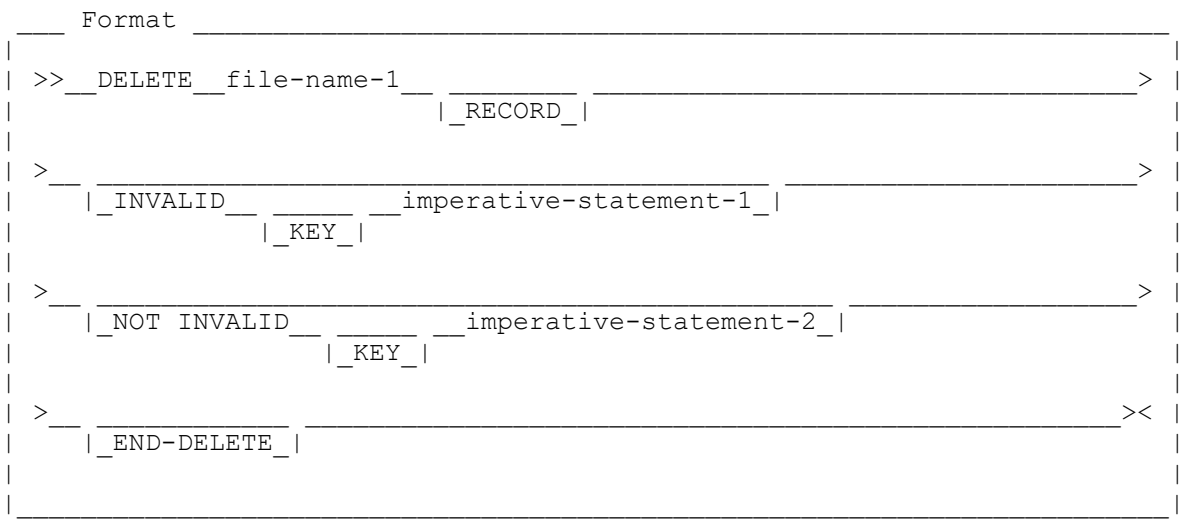
Exemple

```
MOVE CLE TO FIC1-CLE
READ FIC1
     INVALID-KEY DISPLAY 'KEY NOT FOUND
END-READ
```



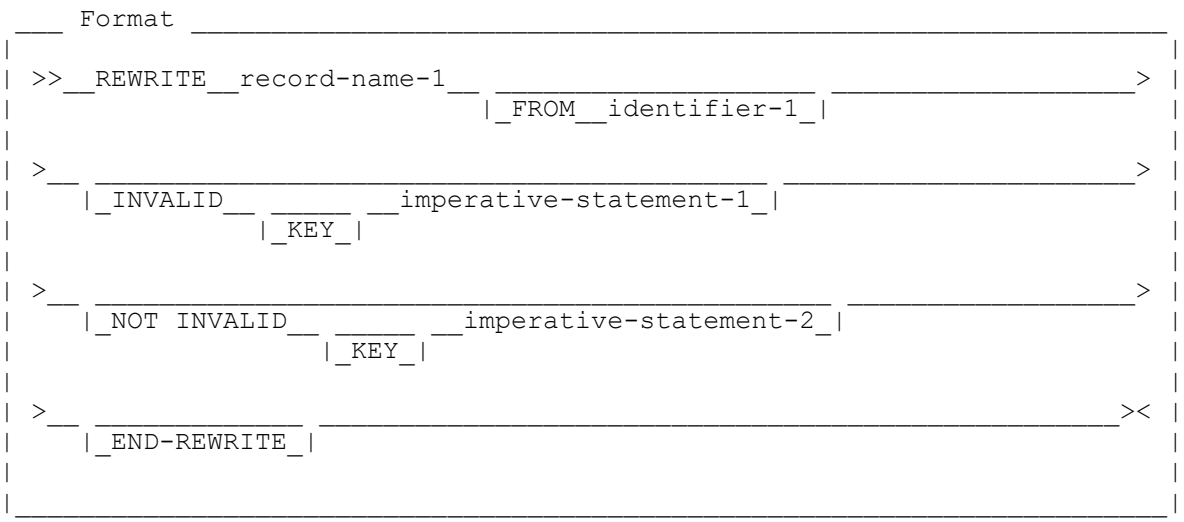
Exemple

```
* APRES AVOIR VALORISE L'ENREGISTREMENT (CLE AU MOINS)
    WRITE ENR-FIC1
        INVALID-KEY DISPLAY 'KEY ALREADY EXIST'
    END-WRITE
```



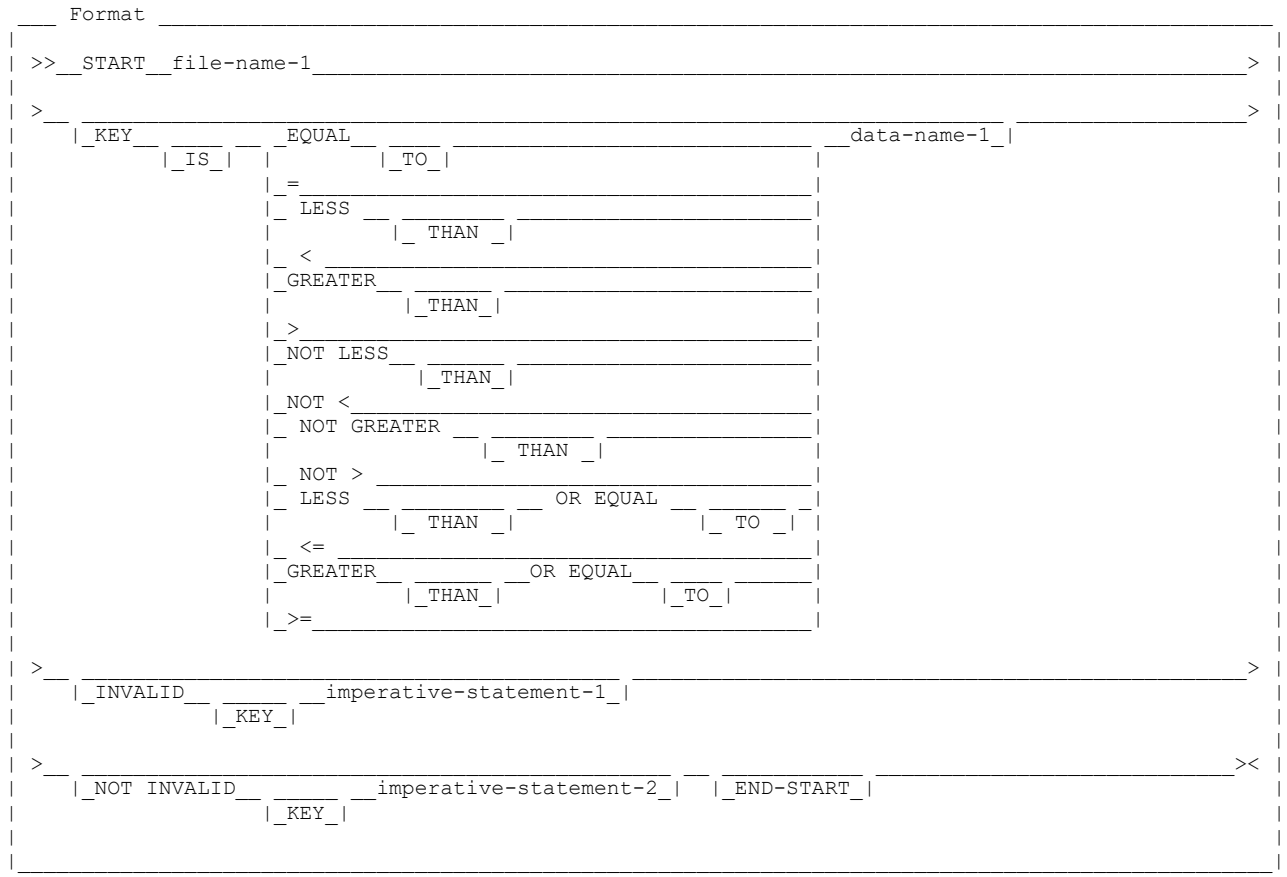
Exemple (*en accès RANDOM ou DYNAMIC)*

```
MOVE VAL-CLE TO FIC1-CLE
DELETE FIC1
    INVALID-KEY DISPLAY 'KEY NOT FOUND'
END-DELETE
```



Exemple (en accès SEQUENTIAL)

```
READ FIC1
     AT END ...
     NOT AT END
         MODIFICATION DE L'ENREGISTREMENT
         REWRITE FIC1-ENR
 END-READ
```



Exemple

```
 FD  FIC1.
01   FIC1-ENR.

     05  FIC1-CLE.
         10  FIC1-CLE1  PIC X.
         10  FILLER     PIC
     05  ...

    MOVE INITIALE TO FIC1-CLE1
    START FIC1 KEY IS >= FIC1-CLE1
          INVALID-KEY
               DISPLAY 'RECORD NOT FOUND FOR THIS KEY'
          NOT INVALID-KEY
               PERFORM LECTURE UNTIL FIN-FIC
     END-START
```

Le START n'effectue pas de lecture.

Mise au point des Programmes

Analyse
de la liste de compilation

LES INFORMATIONS TOUJOURS PRESENTES

Avec les options de compilation standards, la liste de compilation comprend toujours :

1. Liste des options de compilation actives
2. Source du programme mis en forme avec notamment
   - numérotation des lignes (générée à la compilation)
   - barre de colonnage en haut de chaque page
   - indication du niveau d'imbrication des instructions par un numéro à gauche
3. Liste des messages et erreurs de compilation
4. Informations statistiques (nombre de lignes, d'instructions,)

OPTIONS DE COMPILATION AYANT UN IMPACT SUR LA LISTE

Ces options permettent d'obtenir des informations complémentaires sur la liste de compilation.

Elles permettent de faciliter l'analyse du programme source et la recherche d'erreurs.

- **VBREF :** références croisées des verbes
- **XREF :** références croisées des données, procédures et programmes

- **OFFSET** : table de correspondance adresse/ligne-verbe
- **MAP** : définition assembleur et informations sur les données

- **LIST** : expansion du programme en assembleur

AUTRES OPTIONS DE COMPILATION UTILES

- **OPTIMIZE** : Optimisation du code généré (notamment mise en évidence des instructions jamais exécutées)
- **SSRANGE** : permet de contrôler les dépassements de limites de tables
- **FDUMP** : génère un code objet permettant DUMP formaté
- **TEST** : génère un code objet compatible avec COBTEST
- **FLAG** : conditionnement de l'apparition des messages de Compilation

Instructions d'aide à la mise au point

Activé par la clause WITH DEBUGGING MODE

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.
     IBM-370 WITH DEBUGGING MODE.
SPECIAL-NAMES.
      ..
```

Active les instructions :

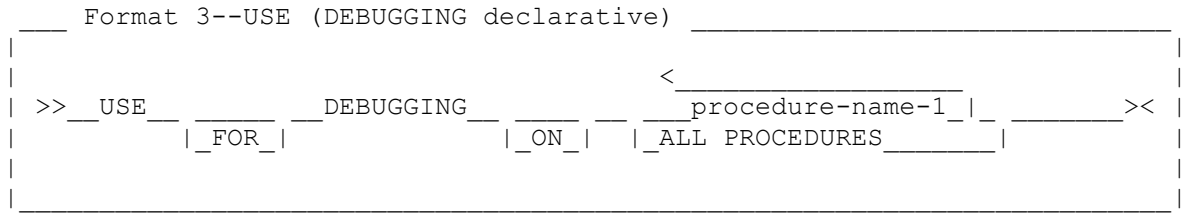
- marquées par un D en colonne 7
  présentes dans la section déclarative

USE FOR DEBUGGING

En tête de la PROCEDURE DIVISION

```
PROCEDURE DIVISION.
    DECLARATIVES.
    ERR1 section.
        USE FOR DEBUGGING
    PARAG1.
        instructions ...
    ERR2 SECTION.
    END DECLARATIVES.

    *       USE FOR DEBUGGING ON ALL PROCEDURES.
     D       USE FOR DEBUGGING ON 2000-
    TRAITEMENT.
     DDEBUG-DECLAR-PARAGRAPH.
     D      DISPLAY '+++++++++++++++  PASSAGE'
    DEBUG-ITEM.
     DDEBUG-DECLAR-END.
     D      EXIT.
     DEND DECLARATIVES.
     *-------PROGRAMME PRINCIPAL-------------*

      2000-TRAITEMENT.
     *--------
     D     IF A-MAT > F-MAT
     D      DISPLAY 'DEBUG-ITEM : '
              EVALUATE F-CODE
                 WHEN 'S'    MOVE 4 TO NUM
```

Format DEBUGGING



- Exécution d'intructions à chaque début de paragraphe/section ou seulement pour le paragraphe/section spécifié.
- Utilisé avec le registre DEBUG-ITEM

Format du registre DEBUG-ITEM

```
01  DEBUG-ITEM.
     02   DEBUG-LINE     PICTURE IS X(6).
     02   FILLER         PICTURE IS X  VALUE SPACE.
     02   DEBUG-NAME     PICTURE IS X(30).
     02   FILLER         PICTURE IS X  VALUE SPACE.
     02   DEBUG-SUB-1    PICTURE IS S9999  SIGN IS LEADING SEPARATE CHARACTER.
     02   FILLER         PICTURE IS X  VALUE SPACE.
     02   DEBUG-SUB-2    PICTURE IS S9999  SIGN IS LEADING SEPARATE CHARACTER.
     02   FILLER         PICTURE IS X  VALUE SPACE.
     02   DEBUG-SUB-3    PICTURE IS S9999  SIGN IS LEADING SEPARATE CHARACTER.
     02   FILLER         PICTURE IS X  VALUE SPACE.
     02   DEBUG-CONTENTS PICTURE IS X(n).
```

- **DEBUG-LINE** indique le numéro de ligne
- **DEBUG-NAME** indique le nom du paragraphe ou de la Section
  - **DEBUG-SUB-n** : indique les valeurs des indices si on utilise un PERFORM VARYING
- **DEBUG-CONTENTS** indique comment on est entré dans le paragraphe ou la section

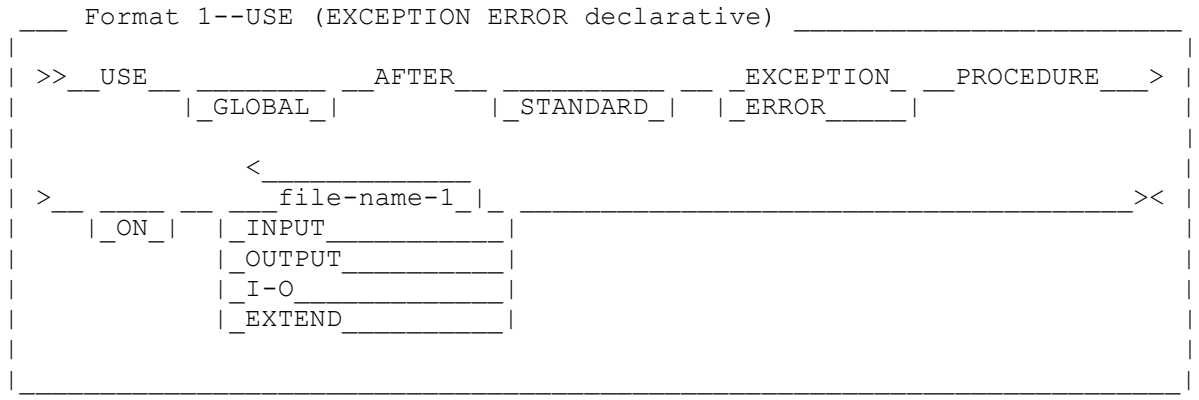
Exemple

```
DISPLAY DEBUG-ITEM
```

Résultat

```
000100 10-LECTURE
PERFORM LOOP
```

Format EXCEPTION/ERROR



Interception de certaines erreurs d'entrée sortie

PERFECTIONNEMENT COBOL

Compléments : Description des données

LITTERAUX HEXADECIMAUX ET DBCS

Depuis COBOL II, deux nouvelles formes de littéraux

Littéraux hexadécimaux

```
X'digits-hexa'
```

Exemple

```
X'C1F0F0F7'
```

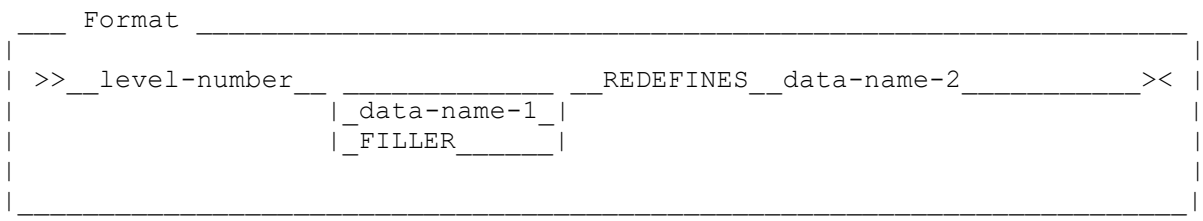
Littéraux DBCS (Double Byte Character Set)

```
G'so données-graphiques si'
```

so : SHIF-OUT (X'0E')
si : SHIF-IN (X'0F')

Utilisés principalement pour les caractères asiatiques

REDEFINES



Pour appliquer un autre masque sur une zone déjà définie

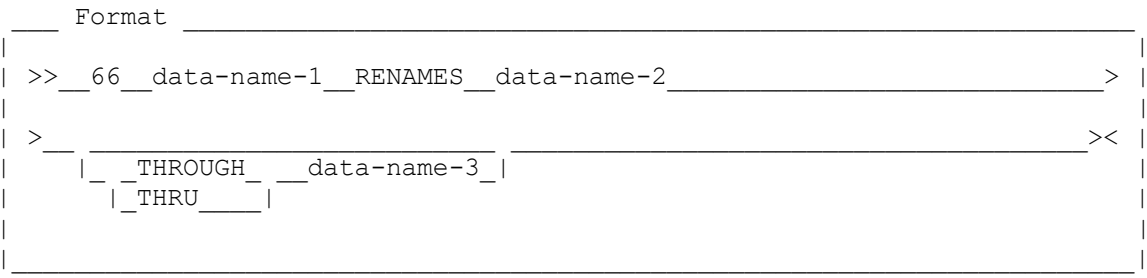
Mêmes numéros de niveau et mêmes longueurs

Pas de VALUE dans la redéfinition

Exemples :

```
05  REGULAR-EMPLOYEE.
  10  LOCATION                        PICTURE A(8).
  10  GRADE                           PICTURE X(4).
  10  SEMI-MONTHLY-PAY                PICTURE 9999V99.
  10  WEEKLY-PAY REDEFINES SEMI-MONTHLY-PAY
                                      PICTURE 999V999.

05  TEMPORARY-EMPLOYEE REDEFINES REGULAR-EMPLOYEE.
  10  LOCATION                        PICTURE A(8).
  10  FILLER                          PICTURE X(6).
  10  HOURLY-PAY                      PICTURE 99V99.

05  REGULAR-EMPLOYEE.
  10  LOCATION                      PICTURE A(8).
  10  GRADE                         PICTURE X(4).
  10  SEMI-MONTHLY-PAY              PICTURE 999V999.

05  TEMPORARY-EMPLOYEE REDEFINES REGULAR-EMPLOYEE.
  10  LOCATION                      PICTURE A(8).
  10  FILLER                        PICTURE X(6).
  10  HOURLY-PAY                    PICTURE 99V99.
  10  CODE-H REDEFINES HOURLY-PAY    PICTURE 9999.
```

RENAMES



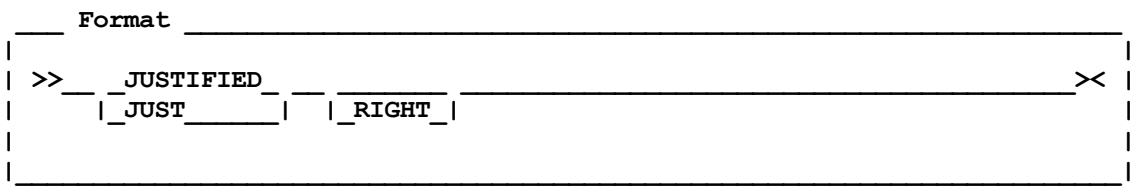
Pour renommer ou grouper différemment des données élémentaires d'un groupe

Interdit de renommer un niveau 01, 77 ou un autre niveau 66

Exemple

```
01 NUM-SS.
   05 SEXE         PIC X.
   05 DAT-NAISS.
      10 AN-N      PIC XX.
      10 MOIS-N    PIC XX.
   05 LIEU-NAISS.
      10 DEPT-N    PIC XX.
    10 VILLE-N     PIC X(3).
  05 COD           PIC X(3).

 66 MOIS-DEPT RENAMES MOIS-N THRU DEPT-N.
```

JUSTIFIED



Seulement pour les données élémentaires

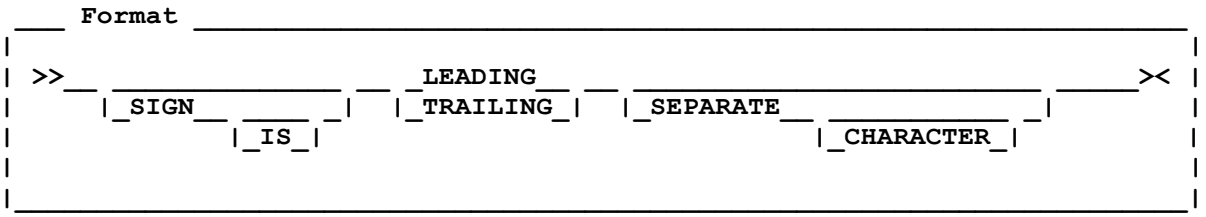
Interdit pour les zones numériques

Aucun effet avec un VALUE (seulement MOVE)

Exemple

```
01 ZONE PIC X(4) JUST VALUE 'AB'        AB

 MOVE 'CD' TO ZONE.                     CD
```

SIGN

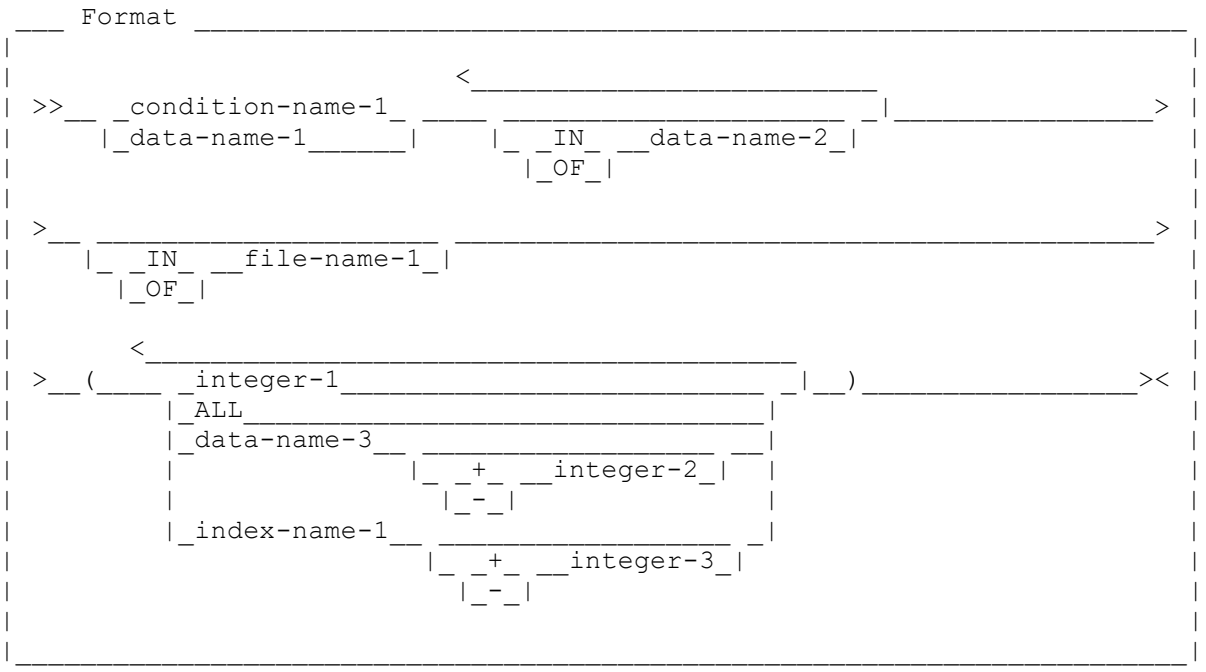


Applicable uniquement à une donnée numérique signée
(PIC S9(xx))

Une seule clause par donnée

Si l'option SEPARATE n'est pas utilisée, le signe est associé au premier ou au dernier caractère de la donnée
(LEADING/TRAILING)

Si l'option SEPARATE est utilisé, la lettre 'S' compte alors pour un caractère et le signe contient la valeur '+' ou '-'



Référence à une donnée dont le nom existe plusieurs fois

Référence à une sous-chaîne de caractères (uniquement avec du DISPLAY)

Exemple

```
MOVE    ZONE(1:15)    TO    AREA1
MOVE    ZONE(1:)      TO    AREA1
```

LES TABLES

NOTION DE TABLE

Suite d'éléments identiques contigus en mémoire auxquels on peut accéder en utilisant leur rang.

La donnée est définie une fois en indiquant le nombre de répétitions.

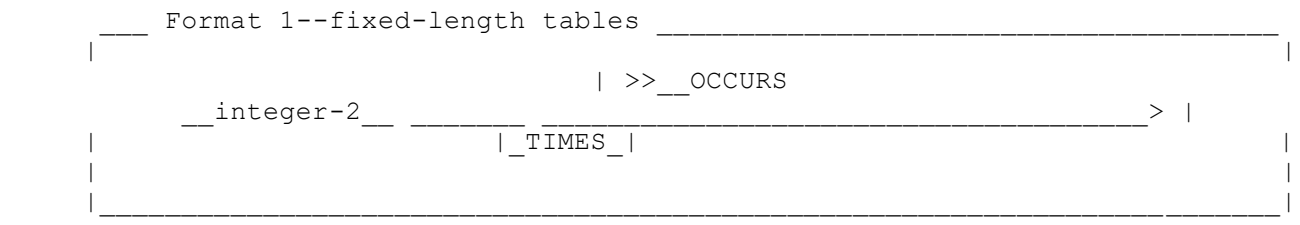
- Poste : chaque élément de la table
- Indice : référence la position relative d'un poste
- Occurrence : apparition d'un poste

Exemple

Table des mois avec nombre de jours

```
  JANVIER   31  FEVRIER   28  MARS    31  AVRIL  30
+ ------ +--+---------+--+-----+---+-----+--+
```

Définition de base




```
01  TABLE-THREE.
        05  ELEMENT-ONE OCCURS 3 TIMES.
            10  ELEMENT-TWO OCCURS 3 TIMES.
                15  ELEMENT-THREE OCCURS 2 TIMES    PIC X(8).
```

Pas au niveau 01 ou 77

Nombre d'occurrences : 1 à 16 777 215

Accès aux données d'une table :

- DONNEE (INDICE)
- Indice = variable ou constante numérique
- Possibilité d'indiçage relatif

Exemple :

```
01 TABLE-MOIS.
    05 POSTE-MOIS OCCURS 12.
        10 NOM         PIC X(10).
        10 NB-JOURS PIC 99.

......
    MOVE 'JANVIER 31' TO POSTE-MOIS (1)

...
    MOVE NOM (MM) TO LIBELLE-MOIS
```

Chaque poste peut lui-même être composé d'une table

Sept niveaux d'imbrication maximum

Eléments référencés en donnant la liste des indices (du plus haut au plus bas)

Exemple :

ZONE (I1 I2 I3)

Remarque :

```
01 ZONE.
   02  Z1       PIC X(5).
   02  Z2       PIC X(5).
   02  Z3       PIC X(5).
   02  Z4       PIC X(5).
   02  Z5       PIC X(5).

01 ZONE      PIC X(5)    OCCURS 5.
01 ZONE.
   02  Z     PIC X(5)    OCCURS 5.

  Z(1)    z(5)
```

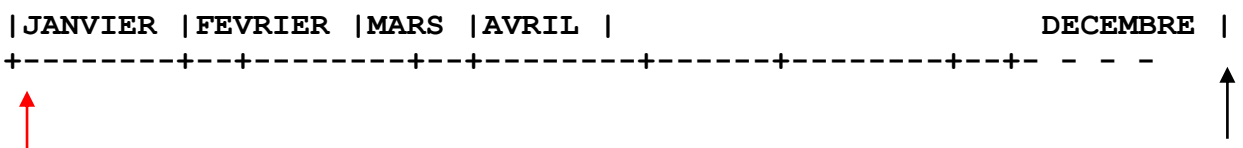
VALORISATION D'UNE TABLE

Par la clause VALUE :

Même valeur pour chacun des postes

Par un REDEFINES :

Exemple :




```
01  VALMOIS.
    05  PIC X(10) VALUE 'JANVIER'.
    05  PIC X(10) VALUE 'FEVRIER'.
    ..........................................................
    05  PIC X(10) VALUE 'DECEMBRE'.

01 TABLE-MOIS REDEFINES VALMOIS.
    05 POSTE-MOIS OCCURS 12        PIC X(10).
```

Par programme :

Avec une boucle de chargement

Attention aux dépassements de capacité !

Option de compilation et d'exécution :
SSRANGE/NOSSRANGE

PERFORM VARYING

On peut écrire jusqu'à six AFTER imbriqués pour initialiser une table à sept niveaux
(1 seul avec PERFORM en ligne)

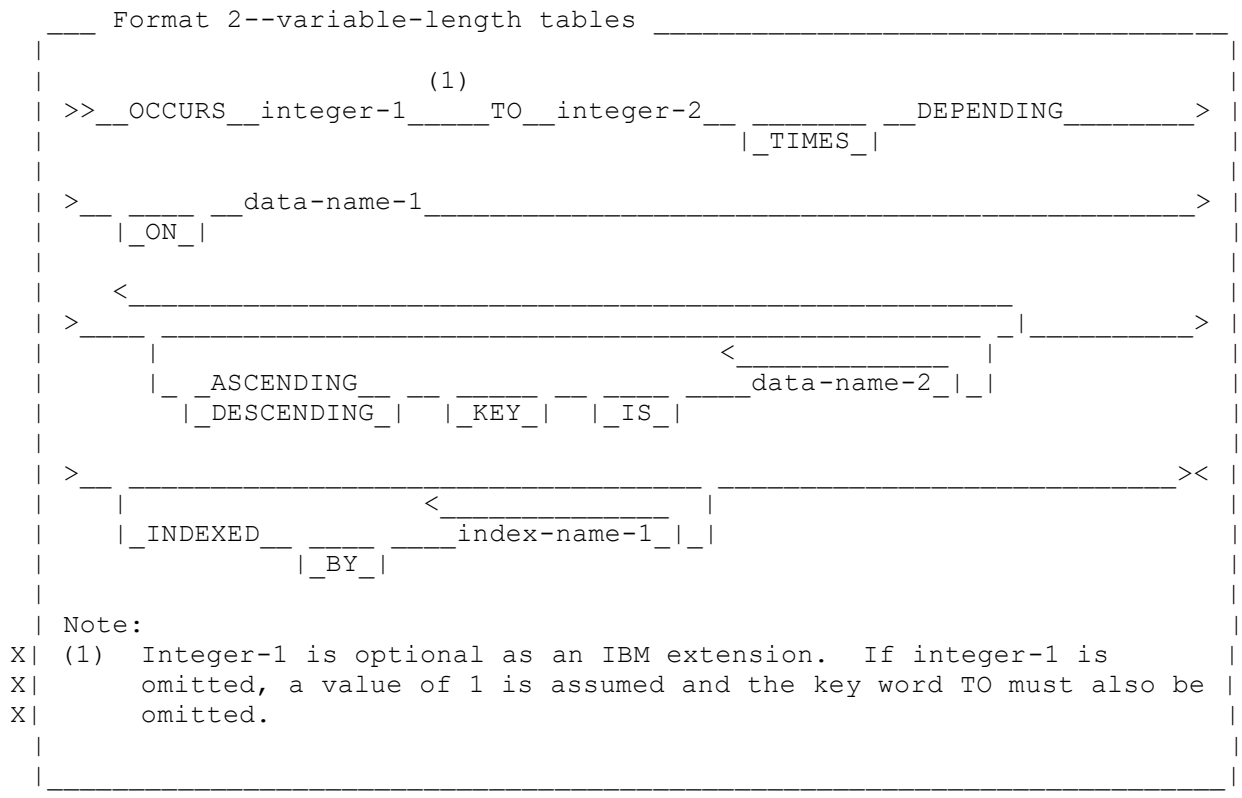
Avec TEST BEFORE : incrément uniquement si l'on reste dans la boucle.

Exemples

```
     PERFORM EXPLO-TABLE
                 VARYING I1 FROM 1 BY 1
                 UNTIL I1 = MAX-TABLE
      OR ZONE-RECH = POSTE (I1)

    PERFORM PROCEDURE-NAME-1 THROUGH PROCEDURE-NAME-2
            VARYING IDENTIFIER-2 FROM IDENTIFIER-3
                    BY IDENTIFIER-4 UNTIL CONDITION-1
                AFTER IDENTIFIER-5 FROM IDENTIFIER-6
                    BY IDENTIFIER-7 UNTIL CONDITION-2
                AFTER IDENTIFIER-8 FROM IDENTIFIER-9
                    BY IDENTIFIER-10 UNTIL CONDITION-3
```

```
01 TABLE-SALAIRE.
     05 POSTE-ANNEE  OCCURS 5.
          10 POSTE-MOIS  OCCURS 12.
               15 POSTE-JOUR  OCCURS 31.
                    20 NOM-EMP          PIC X(12).
                    20 SALAIRE          PIC 9(4)V99.
                    20 TAUX        PIC 99.

     MOVE POSTE-ANNEE (I)          TO  XXXXX.
     MOVE POSTE-MOIS(I,J)          TO  XXXXX
     MOVE POSTE-JOUR (I,J,K)       TO  XXXXX
     MOVE SALAIRE   (5,12,31)      TO  XXXXX

PERFORM VARYING I FROM 1 BY 1 UNTIL I > 5
          After J  FROM 1 BY 1 UNTIL  J > 12
          After K  FROM 1 BY 1 UNTIL  K > 31
               MOVE 'DUPOND'  TO NOM-EMP (I,J,K)
               MOVE      0          TO SALAIRE     (I,J,K)
               MOVE      0          TO TAUX    (I,J,K)
END-PERFORM
        PERFORM PX VARYING I FROM 1 BY 1 UNTIL I > 5

PX.
  MOVE 'DUPOND'  TO NOM-EMP (I,J,K)
  MOVE      0          TO SALAIRE     (I,J,K)
  MOVE      0          TO TAUX    (I,J,K)
```

TABLES DE LONGUEUR VARIABLE

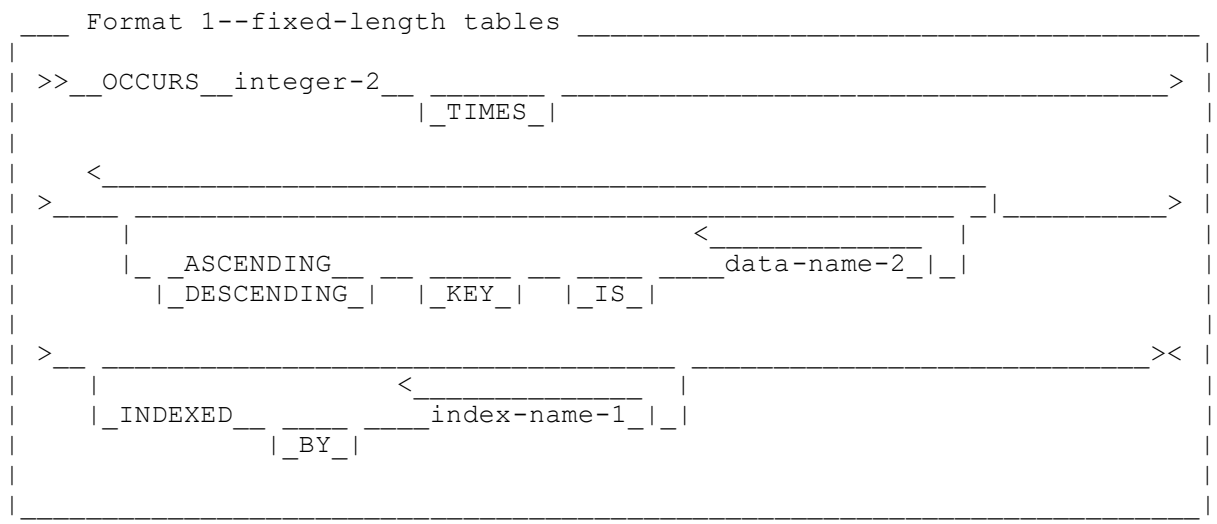


entier-1 peut avoir la valeur zéro ou être omis (1 par défaut)

entier-2 compris entre entier-1 et 16777215

nom-donnée-1 doit contenir une valeur entière positive, donnant le nombre instantané d'occurrences, valeur comprise entre entier-1 et entier-2 et ne causant pas de dépassement de capacité pour la table

```
01 TABLE1.
    05 POSTE OCCURS 1 TO 500 DEPENDING ON J
                                        PIC X(10).
```



12 index maxi pour une table

Pour tables de longueur fixe ou variable

```
WORKING-STORAGE SECTION.
   01  TABLE-RECORD.
     05  EMPLOYEE-TABLE OCCURS 100 TIMES
         ASCENDING KEY IS WAGE-RATE EMPLOYEE-NO
         INDEXED BY A,  B.
       10  EMPLOYEE-NAME                        PIC X(20).
       10  EMPLOYEE-NO                          PIC 9(6).
       10  WAGE-RATE                            PIC 9999V99.
       10  WEEK-RECORD OCCURS 52 TIMES
           ASCENDING KEY IS WEEK-NO INDEXED BY C.
         15  WEEK-NO                            PIC 99.
         15  AUTHORIZED-ABSENCES                PIC  9.
         15  UNAUTHORIZED-ABSENCES              PIC  9.
         15  LATE-ARRIVALS                      PIC  9.
```

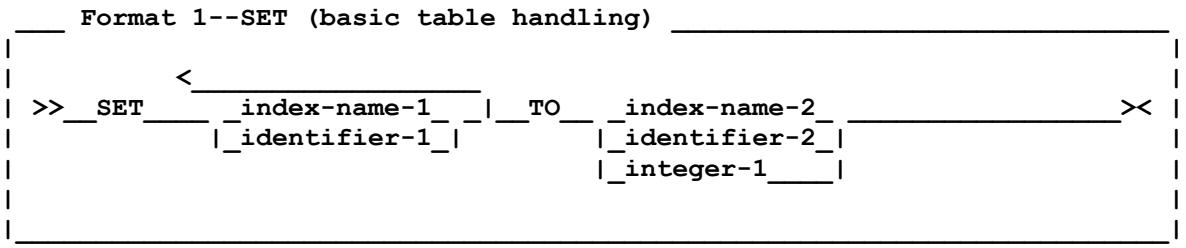
Index utilisables seulement avec les instructions : PERFORM,

SET et SEARCH

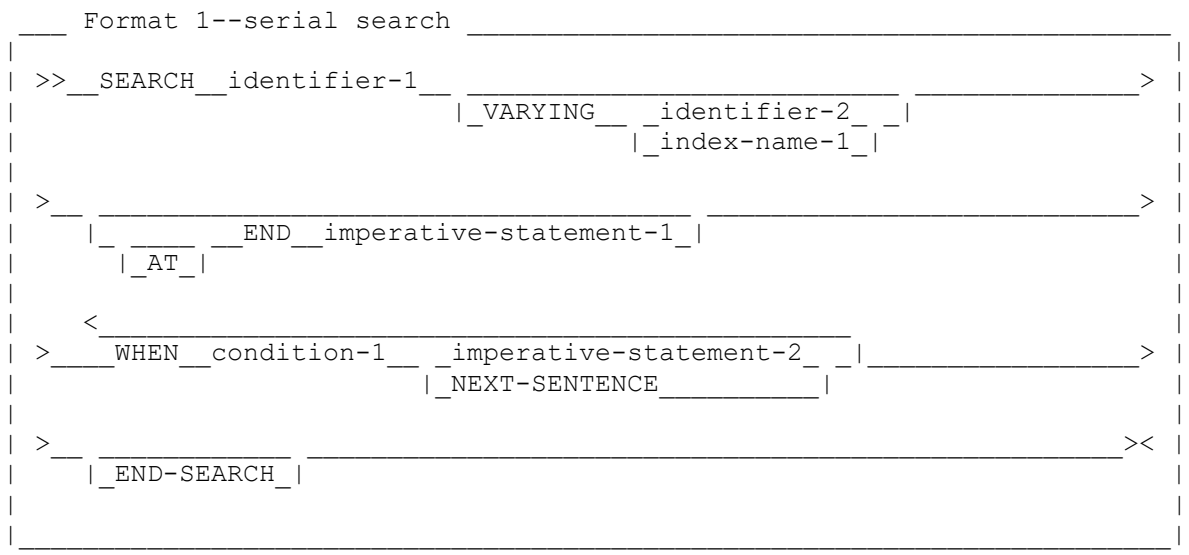
Définition supplémentaire d'index

USAGE IS INDEX

Instruction SET



SEARCH : RECHERCHE SEQUENTIELLE

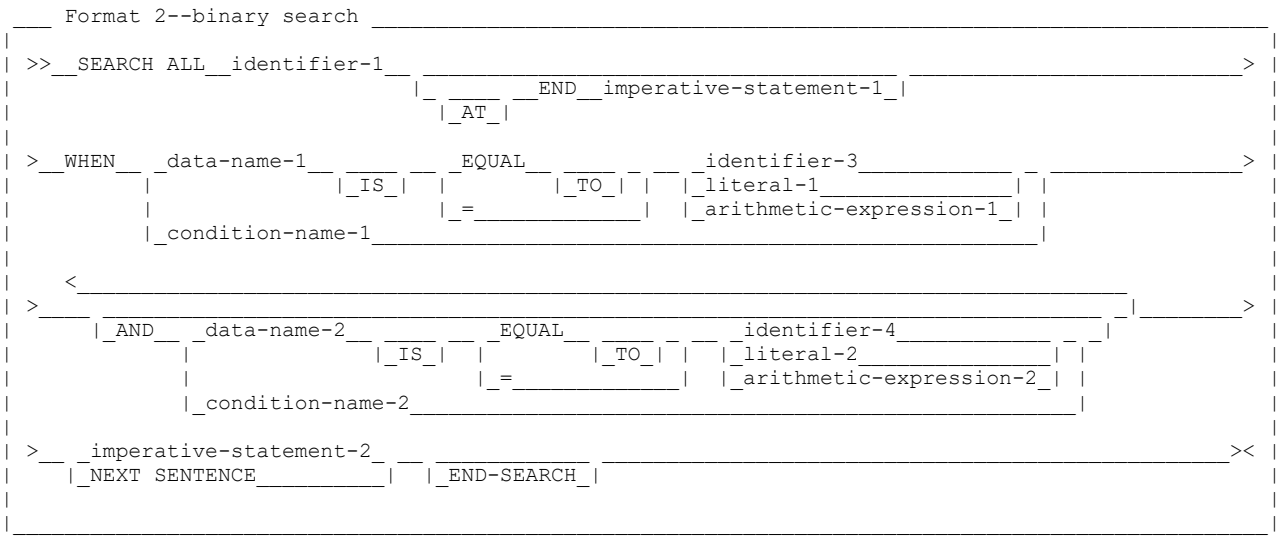


La recherche démarre à la position de l'index (à initialiser)

Elle s'arrête à la première condition WHEN vérifiée ou à la fin de la table si aucune condition n'est satisfaite

A la fin, l'index donne la position atteinte (éventuellement la finde la table)

SEARCH : RECHERCHE DICHOTOMIQUE



La table doit être classée

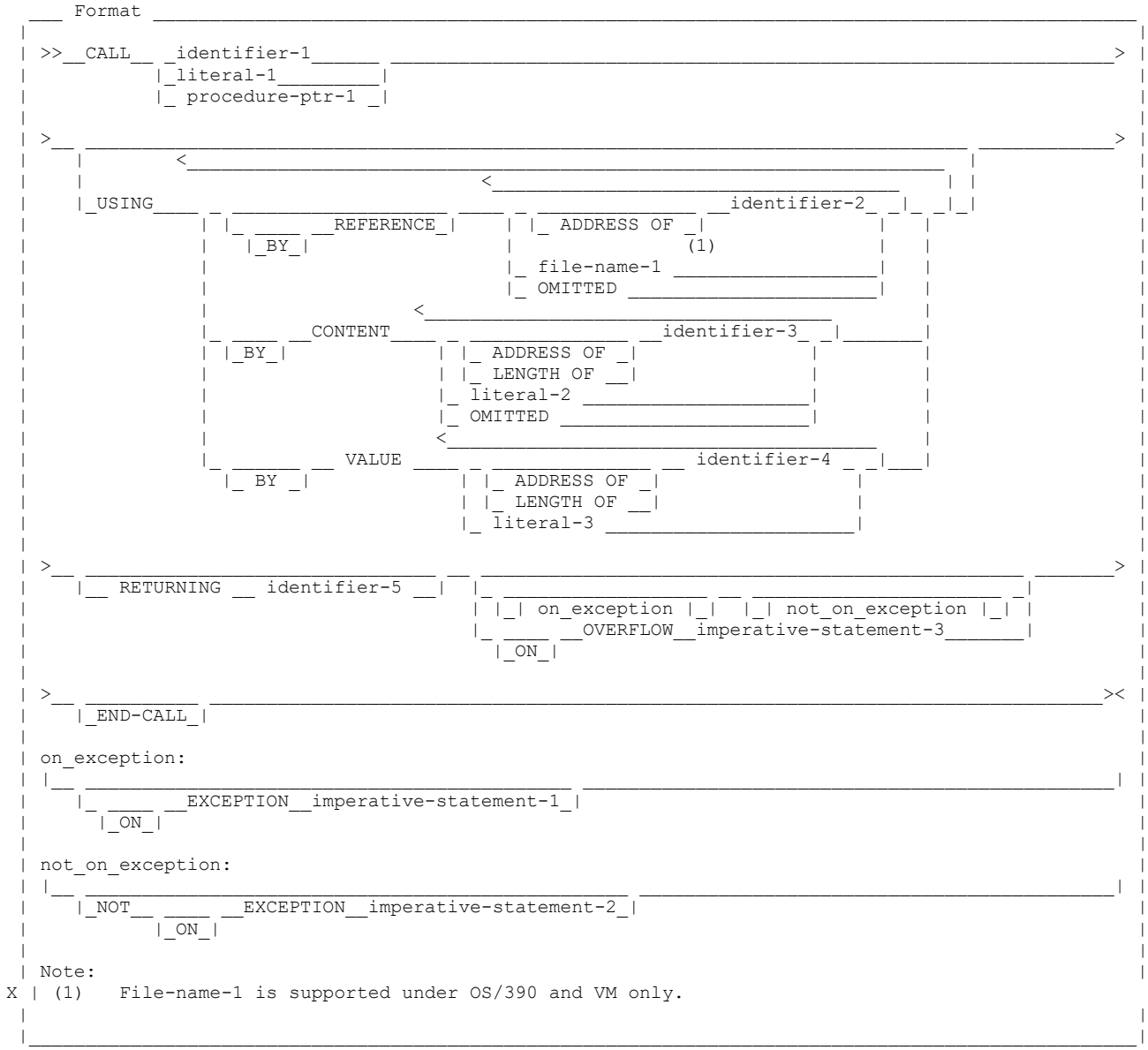
Le critère de classement doit avoir été précisé dans la définitionde la table (ASCENDING/DESCENDING KEY)

TABLES INDEXEES : EXEMPLE

```
 01  TABLE1.
     05 POSTE OCCURS 10 INDEXED BY IX.
        10 CODE1     PIC X.
        10 LIBELLE PIC X(20).
... ...
    SET IX TO 1

    SEARCH POSTE
        AT END
             DISPLAY 'CODE ' CODE-RECH ' INEXISTANT'
          WHEN CODE1 (IX) = CODE-RECH
             MOVE LIBELLE (IX) TO LIBELLE-RECH
    END-SEARCH
```

```
WORKING-STORAGE SECTION.
    01  TABLE-RECORD.
      05  EMPLOYEE-TABLE OCCURS 100 TIMES
          ASCENDING KEY IS WAGE-RATE EMPLOYEE-NO
          INDEXED BY A,  B.
        10  EMPLOYEE-NAME                        PIC X(20).
        10  EMPLOYEE-NO                          PIC 9(6).
        10  WAGE-RATE                            PIC 9999V99.
        10  WEEK-RECORD OCCURS 52 TIMES
            ASCENDING KEY IS WEEK-NO INDEXED BY C.
          15  WEEK-NO                            PIC 99.
          15  AUTHORIZED-ABSENCES                PIC  9.
          15  UNAUTHORIZED-ABSENCES              PIC  9.
          15  LATE-ARRIVALS                      PIC  9.
```

SOUS PROGRAMMES EXTERNES


Format
>>CALL identifier-1
literal-1
procedure-ptr-1
USING
BY
REFERENCE
ADDRESS OF
(1)
file-name-1
OMITTED
identifier-2
CONTENT
LENGTH OF
literal-2
identifier-3
VALUE
literal-3
identifier-4
RETURNING identifier-5
on_exception
NOT
EXCEPTION imperative-statement-1
not_on_exception
EXCEPTION imperative-statement-2
OVERFLOW imperative-statement-3
END-CALL
Note:
(1) File-name-1 is supported under OS/390 and VM only.

CALL BY REFERENCE

Les 2 programmes travaillent sur les mêmes zones mémoire

Option ADDRESS OF :

- Article doit être défini en niveau 01 ou 77 de LINKAGE
- Il existe un registre ADDRESS pour chacun de ces articles
- L'option permet ainsi de transmettre l'adresse à l'appelé

CALL BY CONTENT

Le sous-programme ne travaille pas dans les mêmes zones mémoire que le programme appelant (protection).

Option LENGTH OF :

- Mot binaire, définition implicite PIC 9(9) COMP
- Contient la longueur de la donnée identificateur-2
- Défini pour chaque donnée transmise BY CONTENT avec l'option LENGTH OF
- Si identificateur-2 est un élément de table, LENGTH contient la longueur de l'élément

FORMAT DU SOUS-PROGRAMME

```
ID DIVISION.
PROGRAM-ID. SUBPGM.
ENVIRONMENT DIVISION.
DATA DIVISION.
FILE SECTION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 ... définition des paramètres reçus
 • ..
PROCEDURE DIVISION USING paramètres.
 • ..
 • ..
GOBACK.
```

Chaque identifieur suivant USING doit être défini en niveau 01 ou 77 de la LINKAGE SECTION.

Permet aussi de recevoir des paramètres du JCL

Exemple

```
//       EXEC PGM=pgm1,PARM='SDJ'
//       ...
```

```
LINKAGE SECTION.
01 LIST.
    05 LONG       PIC S9(4) COMP.
    05 PARAM      PIC X(3).
PROCEDURE DIVISION USING LIST.
      IF PARAM = "'SDJ' ....
```

FIN DU SOUS-PROGRAMME

EXIT PROGRAM

- standard COBOL
- seul dans un paragraphe

GOBACK

- extension IBM

Retour au programme appelant, à l'instruction suivant le CALL

CALL STATIQUE OU DYNAMIQUE

Call statique

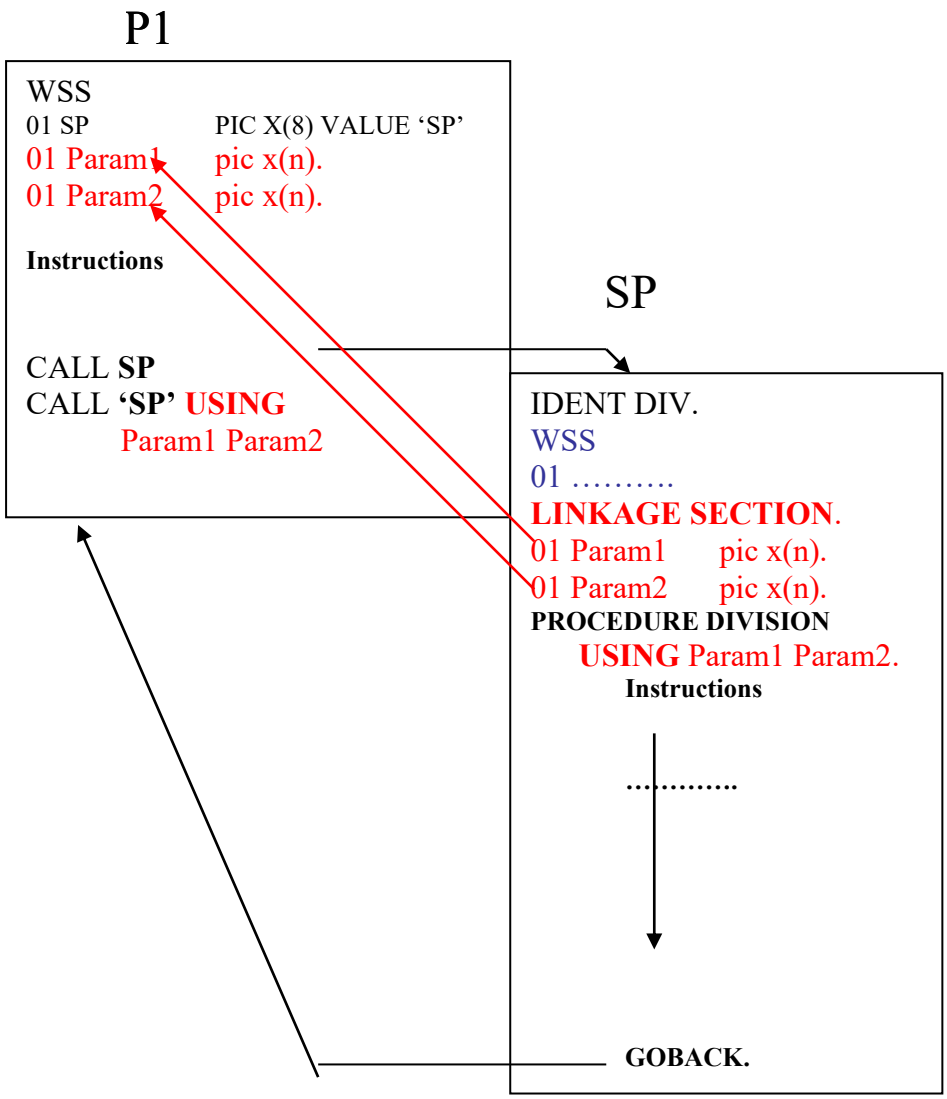
- CALL littéral avec l'option NODYNAM
- Programme et sous-programme(s) doivent appartenir au même load-module

Call dynamique

- CALL littéral avec l'option DYNAM
- CALL identificateur

CALL nom-sp USING BY CONTENT ZA ZB
BY REFERENCE ZC

DISPLAY LENGTH OF ASSURE



OPTION COMPIL CALL	**DYNAM**	**NODYNAM**
IDENTIFICATEUR **CALL SP**	APPEL DYNAMIQUE	APPEL DYNAMIQUE
LITTERAL **CALL 'SP'**	APPEL DYNAMIQUE	**APPEL STATIQUE**

CLAUSE EXTERNAL

Alternative au passage de paramètres entre programmes

Associée à la définition des données :

- niveau 01 en WORKING-STORAGE SECTION
- niveau FD en FILE SECTION

L'allocation mémoire pour la donnée est effectuée une seule fois dans le module exécutable au lieu d'être définie dans chacun des programmes

A l'exécution, tous les programmes décrivant les mêmes éléments avec l'attribut EXTERNAL accèdent à la même zone mémoire

Exemple

```
FILE SECTION.
FD FIC1
EXTERNAL. 01
FIC1-ENR.
05  ...
...
WORKING-STORAGE SECTION.
01 ZONE GROUPE EXTERNAL.
   05 Z1 PIC ...
```

ATTRIBUT INITIAL

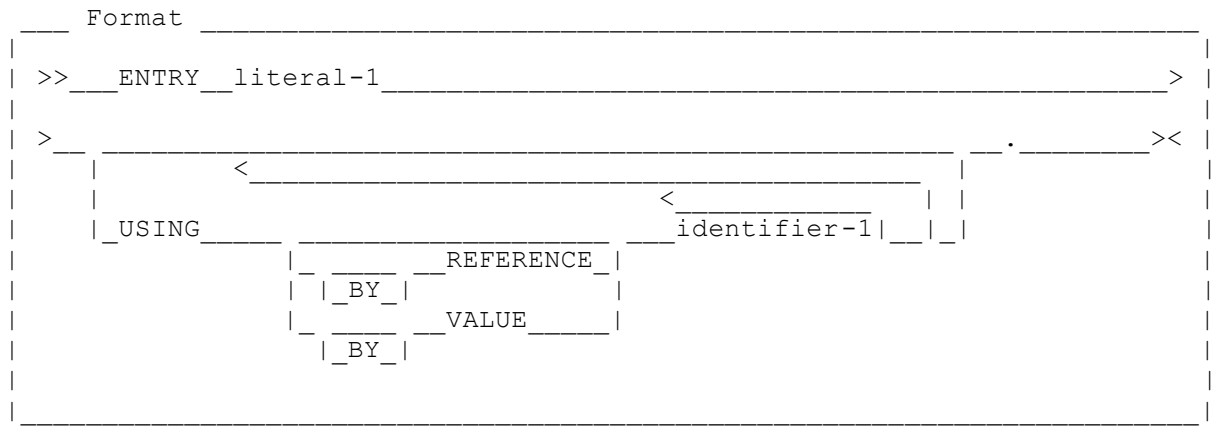
Associé au paragraphe PROGRAM-ID

Remise à l'état initial des zones mémoire du programme à chaque appel (VALUE)

Exemple

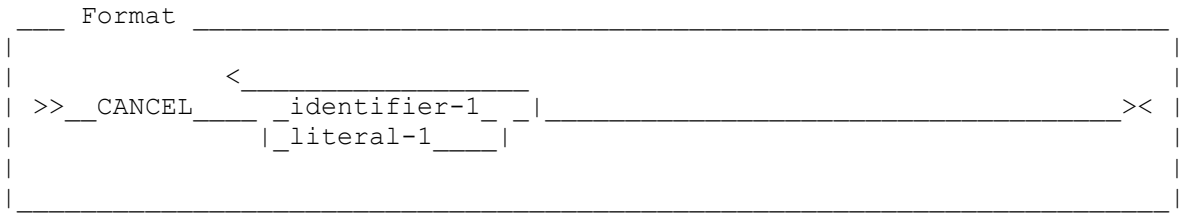
```
ID DIVISION.
PROGRAM-ID. PROG1 INITIAL.
```

Point d'entrée différent du début du programme

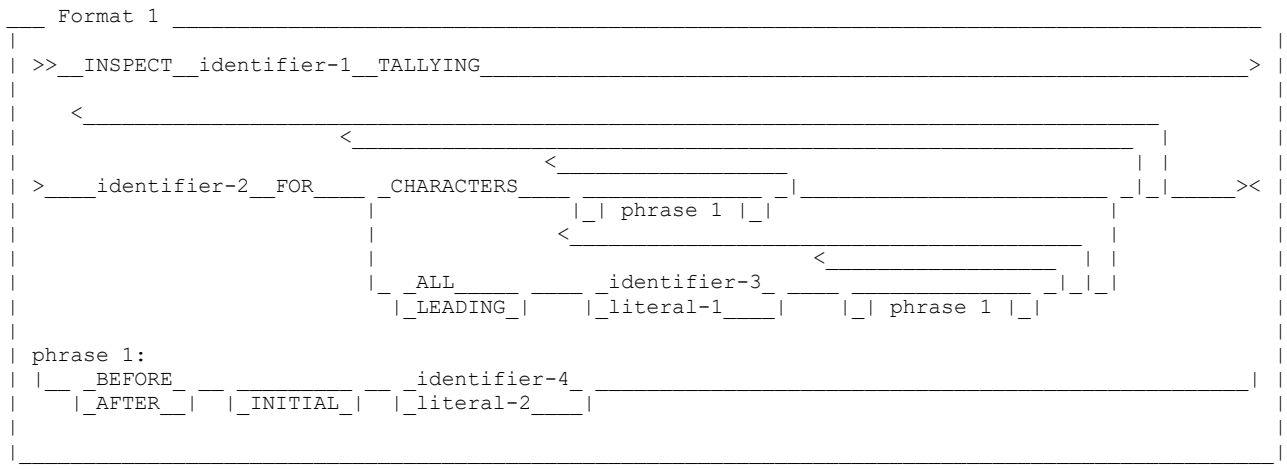


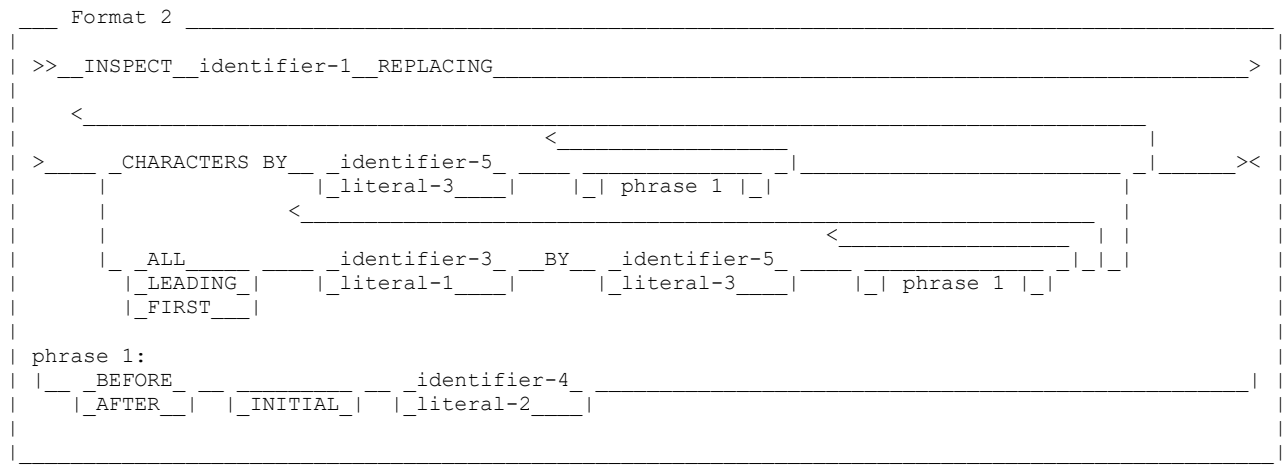
Établit un point d'entrée de nom égal au littéral (à référencer dans un CALL)

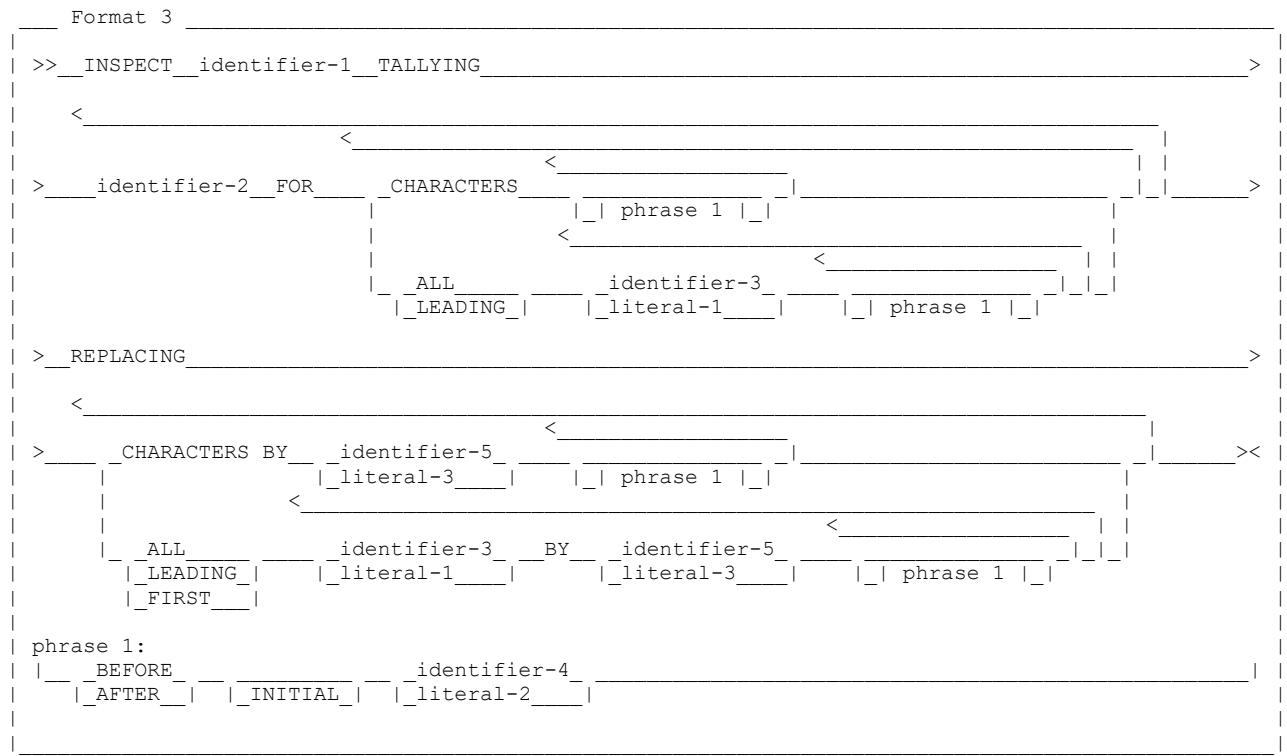
Libération de l'espace occupé par un sous-programme



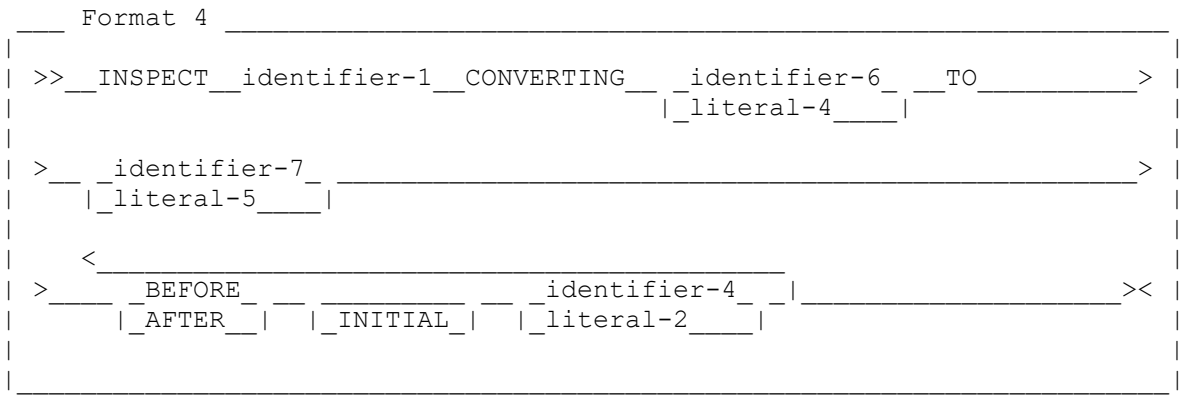
MANIPULATION DE CHAINES


Format 1
>>__INSPECT__identifier-1__TALLYING
>____identifier-2__FOR____ _CHARACTERS____
|_| phrase 1 |_|
|_ _ALL_____ _____ _identifier-3_ _____ _____ _|_|_|
|_LEADING_| |_literal-1____| |_| phrase 1 |_|
phrase 1:
|__ _BEFORE_ __ _________ __ _identifier-4_
|_AFTER__| |_INITIAL_| |_literal-2____|


Format 2
>>__INSPECT__identifier-1__REPLACING
>____ _CHARACTERS BY__ _identifier-5_ _____ _____ _|_____ _|_____><
|_literal-3____| |_| phrase 1 |_|
|_ _ALL_____ _____ _identifier-3_ __BY__ _identifier-5_ _____ _____ _|_|_|
|_LEADING_| |_literal-1____| |_literal-3____| |_| phrase 1 |_|
|_FIRST___|
phrase 1:
|__ _BEFORE_ __ _________ __ _identifier-4_
|_AFTER__| |_INITIAL_| |_literal-2____|


Format 3
>>__INSPECT__identifier-1__TALLYING
>____identifier-2__FOR____ _CHARACTERS____
|_| phrase 1 |_|
|_ _ALL_____ _____ _identifier-3_ _____ _____ _|_|_|
|_LEADING_| |_literal-1____| |_| phrase 1 |_|
>__REPLACING
>____ _CHARACTERS BY__ _identifier-5_ _____ _____ _|_____ _|_____><
|_literal-3____| |_| phrase 1 |_|
|_ _ALL_____ _____ _identifier-3_ __BY__ _identifier-5_ _____ _____ _|_|_|
|_LEADING_| |_literal-1____| |_literal-3____| |_| phrase 1 |_|
|_FIRST___|
phrase 1:
|__ _BEFORE_ __ _________ __ _identifier-4_
|_AFTER__| |_INITIAL_| |_literal-2____|

Format Option REPLACING



Utilisation des deux premiers formats séparés ou conjoints

Comptage et/ou remplacement d'un ou plusieurs caractères

Identificateur-1

- donnée sur laquelle porte l'opération
- donnée groupe ou élémentaire usage DISPLAY

Identificateur-2 (TALLYING)

- compteur, obligatoirement numérique ou TALLY, doit être initialisé avant l'exécution de l'ordre INSPECT

Identificateur-n, littéral-n

- données élémentaires usage DISPLAY

littéraux non numériques

INSPECT

L'option CONVERTING permet de définir une règle de conversion d'un caractère par un autre

- La taille d'identificateur-7 ou de littéral-5 doit être égale à la taille d'identificateur-6 ou de littéral-4
- Chaque caractère appartenant à identificateur-6 ou littéral-4 est remplacé par le caractère de rang correspondant dans identificateur-7 ou littéral-5

EXEMPLES

```
01 ZONE PIC X(6) VALUE 'AAABCA'.

01 CPT1 PIC 9 VALUE 0. 01 CPT2 PIC 9
VALUE 0.


    INSPECT ZONE TALLYING CPT1 FOR ALL 'A' CPT2 FOR LEADING 'A'


    INSPECT ZONE REPLACING LEADING 'A' BY 'X'


    INSPECT ZONE TALLYING TALLY FOR ALL 'A' AFTER 'B' REPLACING FIRST
                'AA' BY 'XX'


    INSPECT ZONE CONVERTING 'AB' TO '12'
```

Exemple :

```
01 Z1     PIC X(5) VALUE 'AB CD'.
01 Z2     PIC X(4) VALUE 'EF.G'.
01 Z3     PIC X(4).
```

```
STRING Z1 DELIMITED BY ' '
       Z2 DELIMITED BY '.'
       INTO Z3
```

STRING

Concaténation de plusieurs données pour former une seule chaîne

DELIMITED SIZE

- tous les caractères sont transmis

DELIMITED ident ou littéral

- caractères transmis jusqu'à ce que le délimiteur soit atteint

POINTER

- position à partir de laquelle doit s'effectuer le transfert dans la zone réceptrice
- doit être initialisé (minimum 1)
- doit être numérique

ON OVERFLOW

- si le pointeur est hors des limites de la zone réceptrice
- si zone réceptrice trop courte pour recevoir toutes les zones émettrices (tient compte du pointeur)

Exemple :

```
01 Z1     PIC X(11) VALUE 'BON APPETIT'.
01 Z2     PIC X(4).
01 Z3     PIC X(4).
01 CPT1   PIC 9.
01 CPT2   PIC 9.

     UNSTRING  Z1 DELIMITED BY ' '
          INTO Z2 COUNT IN CPT1
               Z2 COUNT IN CPT3
```

Dégroupage d'une chaîne de caractères

DELIMITER et COUNT

- ne sont permis qu'avec l'option DELIMITED BY

POINTER

- position à partir de laquelle doit s'effectuer le transfert dans la zone émettrice

ON OVERFLOW

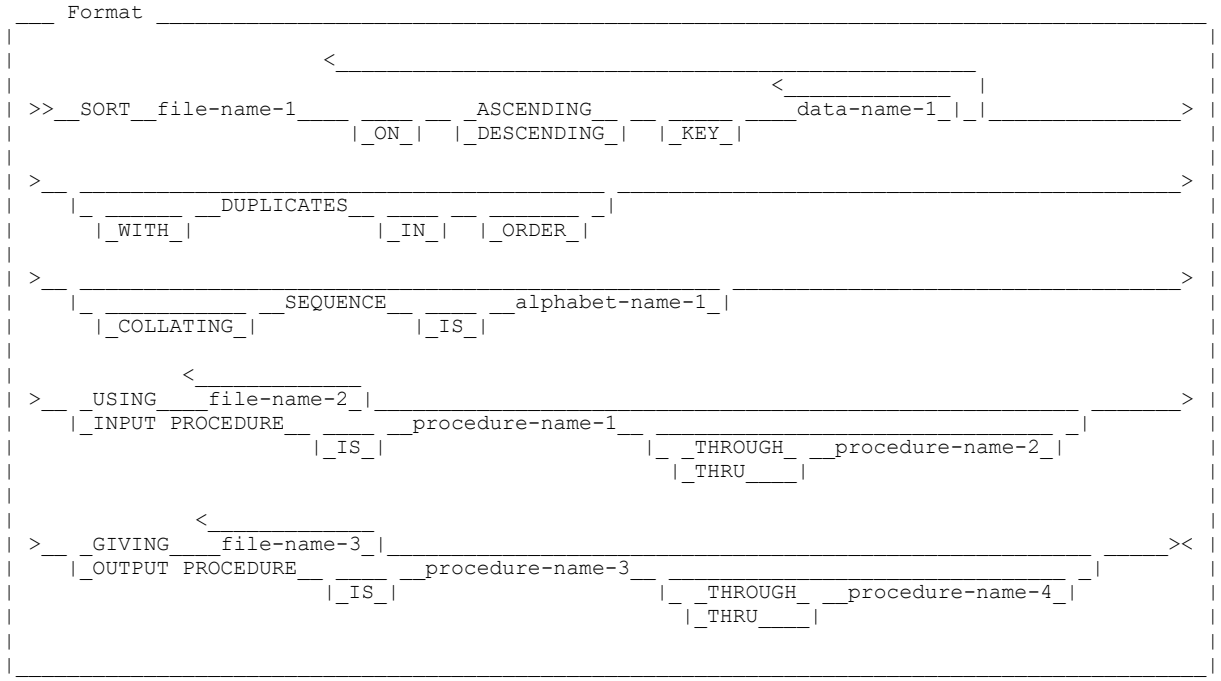
- si pas assez de zones réceptrices pour toute la zone émettrice (tient compte du pointeur)

TRI INTERNE

FICHIER DE TRI

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT FICTRI ASSIGN BIDON.
...

DATA DIVISION.
FILE SECTION.
SD FICTRI.
01 ENRTRI.
    05   ...
```

Fichier de travail, pas de carte DD, pas d'OPEN ni CLOSE


Format
SORT file-name-1
ON
DESCENDING
KEY
ASCENDING
data-name-1
DUPLICATES
WITH
IN
ORDER
SEQUENCE
COLLATING
IS
alphabet-name-1
USING file-name-2
INPUT PROCEDURE
procedure-name-1
THROUGH
THRU
procedure-name-2
GIVING file-name-3
OUTPUT PROCEDURE
procedure-name-3
procedure-name-4

SORT

- Possibilité d'intervention avant le tri : INPUT PROCEDURE
- Possibilité d'intervention après le tri : OUTPUT PROCEDURE
- Nom-de-donnée-n doit être défini dans l'article du fichier de tri
- Nom-de-fichier-1 par la clause SD
- Nom-de-fichier-n par la clause FD
- Pas d'OPEN ni CLOSE pour les fichiers USING et GIVING
- Registre spécial SORT-RETURN (2 octets binaires)

  2 valeurs :

  0 : tri bien terminé

  16 : erreur
- Option de compilation FASTSRT

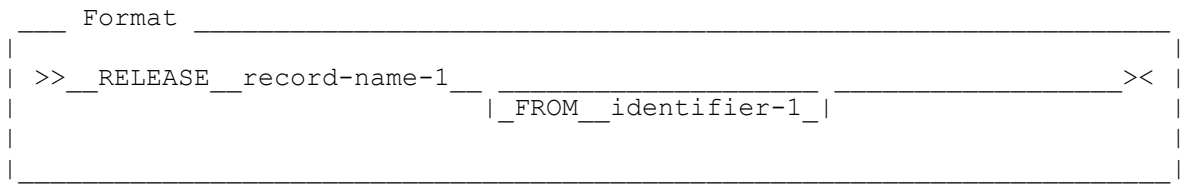
EXEMPLE DE SORT

```
 IDENTIFICATION DIVISION. *
 PROGRAM-ID. PGMTRI.
/
 ENVIRONMENT DIVISION. *
 CONFIGURATION SECTION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT  FENTREE  ASSIGN  TO  DDENT.  SELECT  FTRI  ASSIGN  TO
     DDTRI. SELECT FSORTIE ASSIGN TO DDSORTIE.
/
 DATA DIVISION. *
 FILE SECTION. FD FENTREE
      BLOCK CONTAINS 0 RECORDS
*     LABEL RECORD STANDARD.
  01  FENTREE-ENREG.
      05 ENT-TYPE     PIC 9.
      05 ENT-MAT      PIC 9(6).
      05 ENT-NOM      PIC
      05 ENT-ADR      PIC
      05 ENT-SAL      PIC
      05 ENT-DAT      PIC 9(6).
  FD  FSORTIE
*     LABEL RECORD OMITTED.
  01  FSORTIE-ENREG   PIC
                      X(132).
  SD  FTRI.
*
  01  FTRI-ENREG.
      05  TRI-NOM     PIC X(20).
      05  TRI-SAL      PIC 9(10)V99
      05  TRI-DAT      PIC 9(6).
```

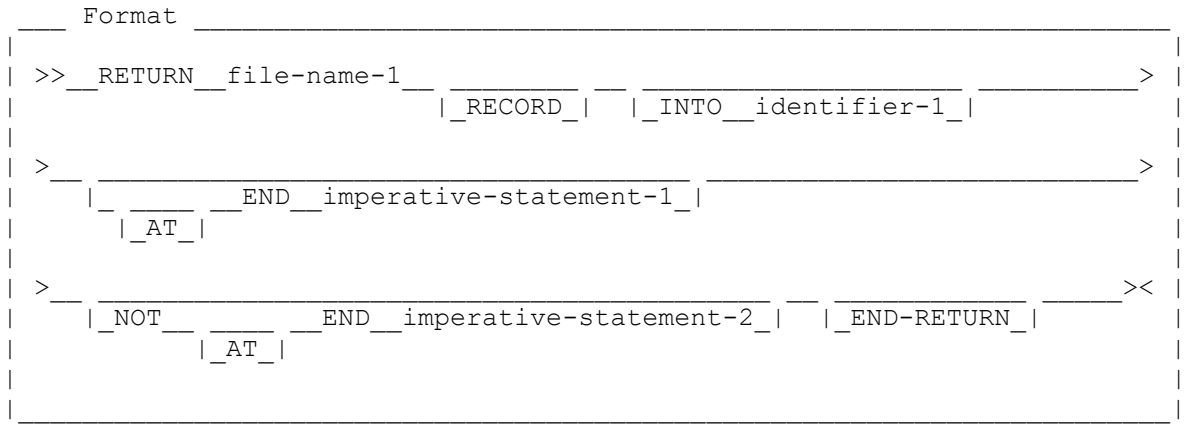
EXEMPLE DE SORT (SUITE)

```
WORKING-STORAGE SECTION.

01 DRAPEAU-ENTREE        PIC 9.
88 FIN-BOUCLE-ENT        VALUE 1.
01 DRAPEAU-TRI           PIC 9.
88 FIN-BOUCLE-TRI        VALUE 1.

   PROCEDURE DIVISION. *
   DEBUT-TRI SECTION.

       SORT FTRI ON ASCENDING KEY TRI-DAT TRI-SAL
            INPUT PROCEDURE EXTRACTION
           OUTPUT PROCEDURE EDITION STOP RUN.

   EXTRACTION SECTION.
       OPEN INPUT FENTREE
       READ FENTREE
            AT END SET FIN-BOUCLE-ENT TO TRUE
       END-READ
       PERFORM UNTIL FIN-BOUCLE-ENT
           IF ENT-TYPE = 2
              MOVE ENT-NOM TO TRI-NOM
              MOVE ENT-DAT TO TRI-DAT
              MOVE ENT-SAL TO TRI-SAL
              RELEASE FTRI-ENREG
           END-IF
           READ FENTREE
                AT END SET FIN-BOUCLE-ENT TO TRUE
           END-READ
       END-PERFORM
       CLOSE FENTREE.

   EDITION SECTION.
       OPEN OUTPUT FSORTIE.
       RETURN FTRI
            AT END SET FIN-BOUCLE-TRI TO TRUE
       END-RETURN
       PERFORM UNTIL FIN-BOUCLE-TRI
           WRITE FSORTIE-ENREG FROM FTRI-ENREG
           RETURN FTRI
                  AT END SET FIN-BOUCLE-TRI TO TRUE
           END-RETURN
       END-PERFORM

      CLOSE FSORTIE
```

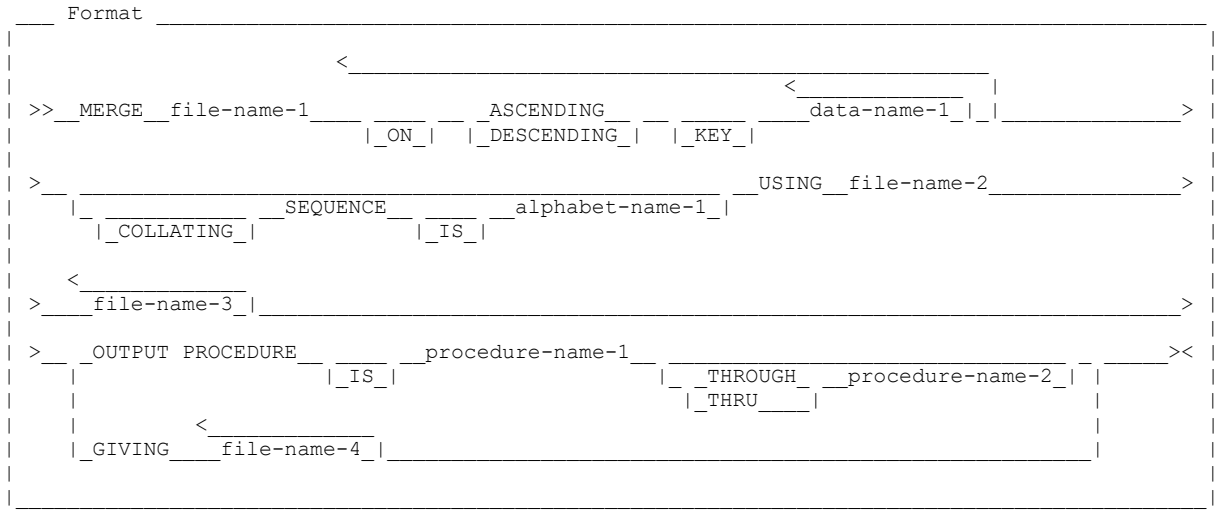
RELEASE : INPUT PROCEDURE



RETURN : OUTPUT PROCEDURE



MERGE



- Peu utilisé, SORT préconisé
- Fusion de plusieurs fichiers séquentiels (maxi 16)
- Les fichiers ne doivent pas être ouverts
- Nom-de-fichier-1 doit être défini par la clause SD
  Nom-de-fichier-n par la clause FD

DIRECTIVES

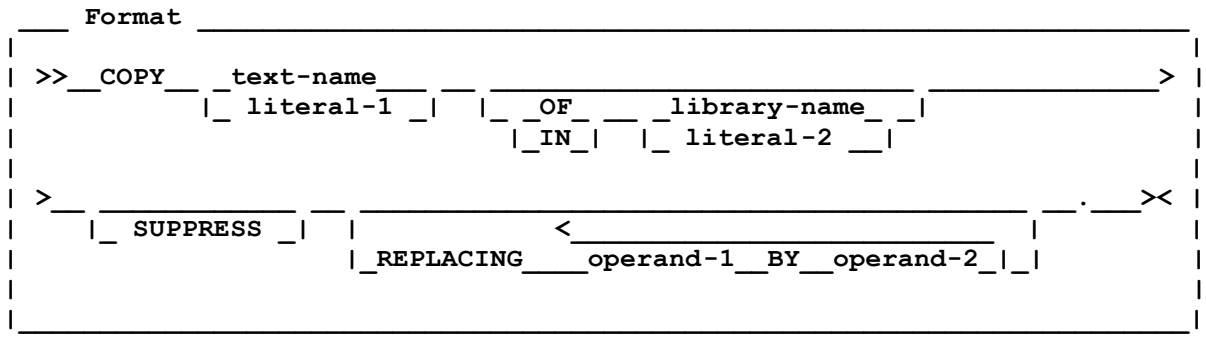
DIRECTIVES DE COMPILATION

Concernent la compilation et non l'exécution du programme
Ce ne sont pas des instructions

Syntaxe :

- en marge A ou B
- rien d'autre sur la même ligne
- point de fin facultatif

DIRECTIVE COPY



- Permet d'insérer des portions de description de données ou d'instructions
- COPY imbriqués possibles (incompatible avec REPLACING)
- SUPPRESS : pas d'impression sur la liste de compilation
- Opérande 1 et 2 : littéral, variable, pseudo-texte (==...==)
  Option de compilation LIB

*CONTROL (*CBL)

- Permet de contrôler (supprimer) la liste et la génération des instructions

EJECT

- Provoque un saut de page

SKIP1/2/3

- Provoque un saut de une, deux ou trois lignes

TITLE

- Suivi d'un littéral, provoque l'impression de ce littéral en haut de chaque page de la liste

PROGRAMMES IMBRIQUES

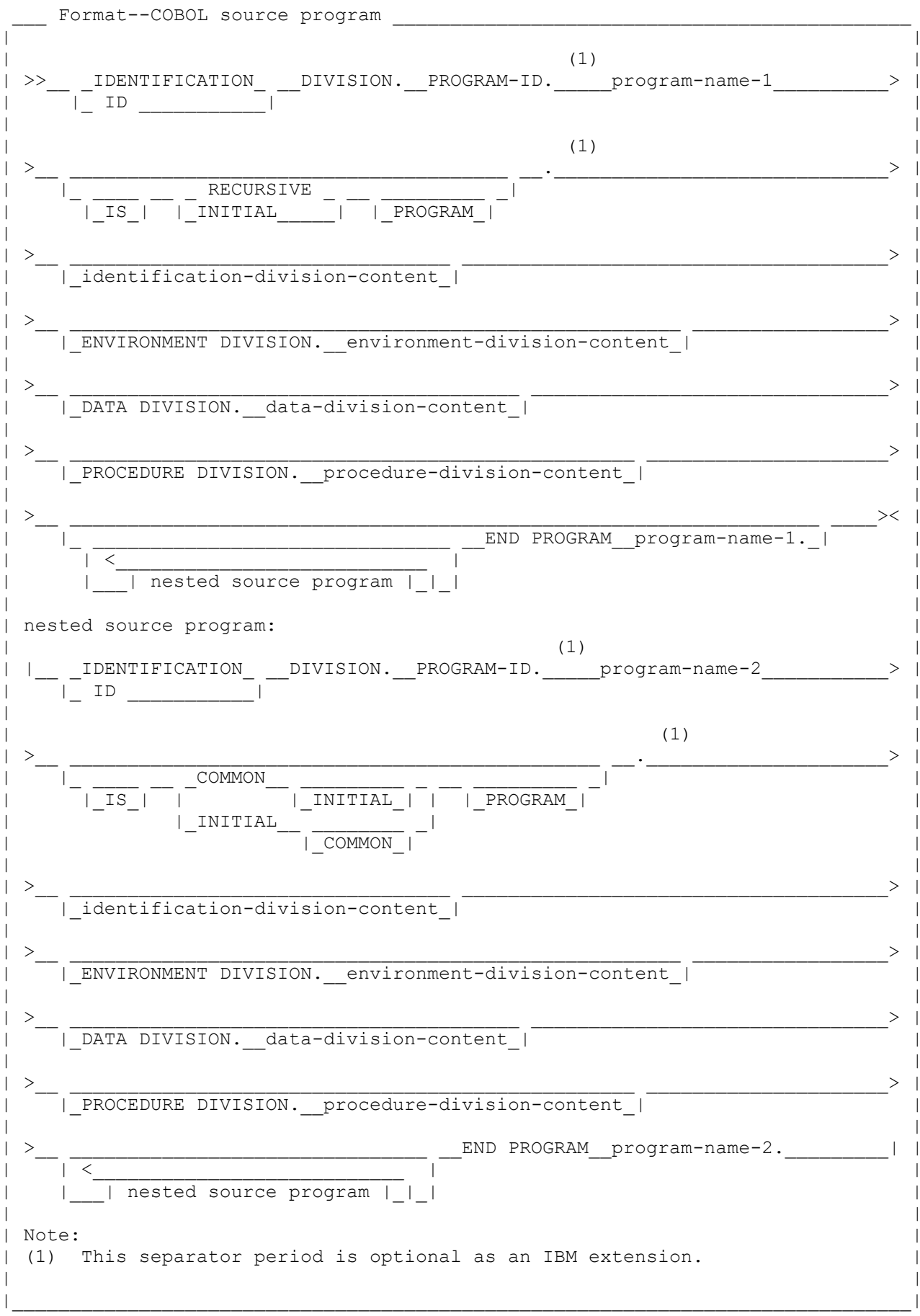
STRUCTURE D'UN PROGRAMME

Un programme source COBOL peut contenir d'autres programmes sources pouvant accéder aux ressources du programme dans lequel ils sont contenus.

L'IDENTIFICATION DIVISION est requise dans chaque programme, les autres divisions sont optionnelles.

Chaque programme doit se terminer par le délimiteur END PROGRAM immédiatement suivi du nom du programme.

STRUCTURE D'UN PROGRAMME IMBRIQUÉ



Exemple

```
IDENTIFICATION DIVISION.
PROGRAM-ID. PROG1.
..
ENVIRONMENT DIVISION.
..
DATA DIVISION.
..
PROCEDURE DIVISION.
..
IDENTIFICATION DIVISION.
PROGRAM-ID. PROG2.
..
ENVIRONMENT DIVISION.
..
DATA DIVISION.
..
PROCEDURE DIVISION.
..
END PROGRAM PROG2
END PROGRAM PROG1
```

ATTRIBUT COMMON

Par défaut un sous-programme ne peut être appelé que par celui qui le contient directement

Un programme ayant l'attribut COMMON peut être appelé par tous ceux qui appartiennent au même que lui

Pas de récursivité

CLAUSE GLOBAL

Associée à la définition des données :

- niveau 01 en WORKING-STORAGE SECTION
- niveau FD en FILE SECTION

Les noms de données peuvent être utilisés dans tous les programmes contenus lors de la même exécution

La portée d'un nom de donnée s'arrête dès que le programme contenu utilise ce nom pour définir une donnée

Exemple

```
IDENTIFICATION DIVISION.
PROGRAM-ID. PROG1.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ZONE-COMMUNE PIC 99 GLOBAL.
01 ZONE-P1   PIC 9.

PROCEDURE DIVISION.
 INITIALIZE ZONE-COMMUNE ZONE-P1
 CALL PROG2
DISPLAY ZONE-COMMUNE
DISPLAY ZONE-P1

IDENTIFICATION DIVISION.
PROGRAM-ID. PROG2.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ZONE-P1   PIC 9 VALUE 1.
01 ZONE-P2   PIC 9 VALUE 1.

PROCEDURE DIVISION.

...
ADD ZONE-P1 ZONE-P2 GIVING ZONE-COMMUNE

END PROGRAM PROG2.

END PROGRAM PROG1.
```

LES FONCTIONS INTRINSEQUES

Définition

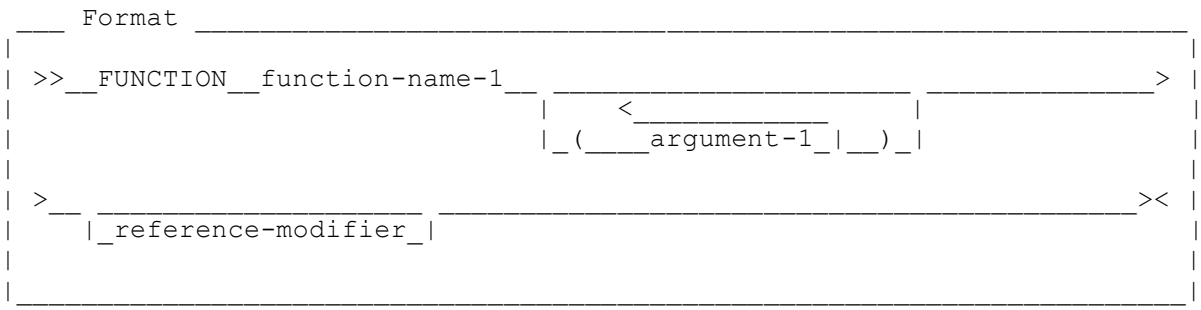
- Les fonctions intrinsèques sont un ensemble de possibilités rajoutées au COBOL normalisé 1985 par amendement de 1989, et sont intégrées dans COBOL/370.
- les fonctions sont introduites par le mot-clé FUNCTION
- Chaque fonction requiert des paramètres spécifiques.

Typologie

Il existe plusieurs types de fonctions intrinsèques

- date et heure
- alphanumériques
- mathématiques
- statistiques
- financières

Syntaxe



nom-de-fonction-1 doit être le nom d'une des fonctions intrinsèques

argument-1 peut être un identificateur, un littéral (mais pas une constante figurative) ou une expression arithmétique

modification-de-référence ne peut être spécifiée que pour les fonctions alphanumériques

Fonctionnement

- une fonction est considérée comme une donnée élémentaire
- selon la nature de la fonction, elle peut retourner une valeur numérique, un entier ou une valeur alphanumérique
- une fonction peut remplacer un opérande dans une instruction COBOL. Sa valeur dépend de son type et est calculée, en fonction des paramètres éventuels, lors de l'exécution.
- Le nombre et le type des paramètres dépendent de la fonction.
- Les fonctions intrinsèques permettent
  - l'imbrication (une fonction peut utiliser les résultats d'une autre fonction)
  - la récursivité (la même fonction peut être appelée plusieurs fois)

EXEMPLES

Mise en minuscule du contenu d'une zone, et transfert du résultat dans une autre zone.

```
MOVE FUNCTION LOWER-CASE(W-ZONE1) TO W-ZONE2.
```

Le résultat de w-zone2 est identique à celui contenu dans w-zone1.

Seule la taille des caractères est changée.

Mise en forme du nom/prénom dans une zone d'édition

```
MOVE FUNCTION UPPER-CASE (NOM)      TO NOM-ED
MOVE FUNCTION UPPER-CASE (PRENOM (1:1)) TO PRENOM-ED (1:1)
```

LE MOT-CLE ALL

Exemple : *calcul de la moyenne d'une zone appartenant à une table à une dimension*

```
COMPUTE W-MOY = FUNCTION MEAN(SALAIRE (ALL))
```

Le mot-clé ALL signifie que la fonction s'applique à toutes les occurrences de la table SALAIRE

Dans le cas d'une table à plusieurs dimensions,

```
TABLE (ALL, ALL)
```

signifie que la fonction s'applique à toutes les occurrences de la table

```
TABLE (ALL, 1)
```

est équivalent à

```
TABLE (1, 1) TABLE (2, 1) ... TABLE (n, 1)
```

Types des paramètres et des valeurs retournées

X : alphanumérique

A : alphabétique

N : numérique

E : entier

Q : quelconque

Table of functions (1/3)

	Function name	Arguments	Type	Value returned
	ACOS	N1	N	Arccosine of N1
	ANNUITY	N1, I2	N	Ratio of annuity paid for I2 periods at interest of N1 to initial investment of one
	ASIN	N1	N	Arcsine of N1
	ATAN	N1	N	Arctangent of N1
	CHAR	I1	X	Character in position I1 of program collating sequence
	COS	N1	N	Cosine of N1
	CURRENT-DATE	None	X	Current date and time and difference from Greenwich Mean Time
	DATE-OF-INTEGER (	I1	I	Standard date equivalent
X	DP)			(YYYYMMDD) of integer date
X	DATE-TO-YYYYMMDD	I1, I2	I	Standard date equivalent
X	(DP)			(YYYYMMDD) of I1 (standard date
X				with a windowed year, YYMMDD),
X				according to the 100-year
X				interval whose ending year is
X				specified by the sum of I2 and
X				the year at execution time
X	DATEVAL (DP)	I1 or	I	Date field equivalent of I1 or X1
X		X1	X	
	DAY-OF-INTEGER (	I1	I	Julian date equivalent (YYYYDDD)
X	DP)			of integer date
X	DAY-TO-YYYYDDD	I1, I2	I	Julian date equivalent (YYYYDDD)
X	(DP)			of I1 (Julian date with a
X				windowed year, YYDDD), according
X				to the 100-year interval whose
X				ending year is specified by the
X				sum of I2 and the year at
X				execution time
	FACTORIAL	I1	I	Factorial of I1
	INTEGER	N1	I	The greatest integer not greater than N1
	INTEGER-OF-DATE	I1	I	Integer date equivalent of standard date (YYYYMMDD)
	INTEGER-OF-DAY	I1	I	Integer date equivalent of Julian date (YYYYDDD)

Table of functions (2/3)

INTEGER-PART	N1	I	Integer part of N1
LENGTH	A1, N1, or X1	I	Length of argument
LOG	N1	N	Natural logarithm of N1
LOG10	N1	N	Logarithm to base 10 of N1
LOWER-CASE	A1 or X1	X	All letters in the argument are set to lowercase
MAX	A1... or I1... or N1... or X1...	X I N X	Value of maximum argument; note that the type of function depends on the arguments
MEAN	N1...	N	Arithmetic mean of arguments
MEDIAN	N1...	N	Median of arguments
MIDRANGE	N1...	N	Mean of minimum and maximum arguments
MIN	A1... or I1... or N1... or X1...	X I N X	Value of minimum argument; note that the type of function depends on the arguments
MOD	I1,I2	I	I1 modulo I2
NUMVAL	X1	N	Numeric value of simple numeric string
NUMVAL-C	X1 or X1,X2	N	Numeric value of numeric string with optional commas and currency sign
ORD	A1 or X1	I	Ordinal position of the argument in collating sequence
ORD-MAX	A1..., N1..., or X1...	I	Ordinal position of maximum argument
ORD-MIN	A1..., N1..., or X1...	I	Ordinal position of minimum argument
PRESENT-VALUE	N1 or N2...	N	Present value of a series of future period-end amounts, N2, at a discount rate of N1

Table of functions (3/3)

	RANDOM	I1, none	N	Random number
	RANGE	I1... or	I	Value of maximum argument minus value of minimum argument; note that the type of function depends on the arguments.
		N1...	N	
	REM	N1,N2	N	Remainder of N1/N2
	REVERSE	A1 or X1	X	Reverse order of the characters of the argument
	SIN	N1	N	Sine of N1
	SQRT	N1	N	Square root of N1
	STANDARD-DEVIATION	N1...	N	Standard deviation of arguments
	SUM	I1... or	I	Sum of arguments; note that the type of function depends on the arguments.
		N1...	N	
	TAN	N1	N	Tangent of N1
X	UNDATE (DP)	I1 or	I	Non-date equivalent of date field I1 or X1
X		X1	X	
	UPPER-CASE	A1 or X1	X	All letters in the argument are set to uppercase
	VARIANCE	N1...	N	Variance of arguments
	WHEN-COMPILED	None	X	Date and time when program was compiled
X	YEAR-TO-YYYY (DP)	I1, I2	I	Expanded year equivalent (YYYY) of I1 (windowed year, YY), according to the 100-year interval whose ending year is specified by the sum of I2 and the year at execution time
X	YEARWINDOW (DP)	None	I	If the DATEPROC compiler option is in effect, returns the starting year (in the format YYYY) of the century window specified by the YEARWINDOW compiler option; if NODATEPROC is in effect, returns 0

REPERTOIRE DES FONCTIONS

ACOS (N) Arc cosinus de N

ANNUITY (N1,E2) Ratio des annuités payées pendant E2 périodes avec un intérêt de N1 pour un investissement initial de 1

ASIN (N) Arcsinus de N

ATAN (N) Arctangente de N

CHAR (E) Eème caractère de la séquence de l'ordinateur

COS (N) Cosinus de N

CURRENT-DATE Date/heure système (et décalage heure Greenwich)
21 car : AAAAMMJJHHMMSSCC±HHMM

DATE-OF-INTEGER (E) Conversion nombre de jours depuis 1/1/1601 en date (AAAAMMJJ)

DAY-OF-INTEGER (E) Conversion nombre de jours depuis 1/1/1601 en date (AAAAQQQ)

FACTORIAL (E) Factorielle de E

INTEGER (N) Plus grand entier <= à N

REPERTOIRE DES FONCTIONS

INTEGER-OF-DATE (E)	Conversion date (AAAAMMJJ) en nombre de jours depuis 1/1/1601
INTEGER-OF-DAY (E)	Conversion date (AAAAQQQ) en nombre de jours depuis 1/1/1601
INTEGER-PART (N)	Partie entière de N
LENGTH (Q)	Longueur de l'argument
LOG (N)	Logarithme de N
LOG10 (N)	Logarithme à base 10
LOWER-CASE (X)	Conversion en minuscules
MAX (Q1,Q2,...)	Valeur maximum de l'argument (tous de même type)
MEAN (N1,N2,...)	Moyenne arithmétique des arguments
MEDIAN (N1,N2,...)	Valeur médiane des arguments
MIDRANGE (N1,N2,...)	Moyenne des 2 valeurs extrêmes
MIN (Q1,Q2,...)	Valeur minimum de l'argument (tous de même type)
MOD (E1,E2)	E1 modulo E2

NUMVAL (X) Conversion en numérique d'une chaîne de caractères numérique éditée

NUMVAL-C (X1,X2) Conversion en numérique d'une chaîne (X1) de caractères numériques, en ignorant point et symbole monétaire (X2)

ORD (X ou A) Position ordinale du caractère dans la séquence de l'ordinateur

ORD-MAX (Q1,Q2...) Position ordinale du caractère maximum de la liste dans la séquence de l'ordinateur

ORD-MIN (Q1,Q2...) Position ordinale du caractère minimum de la liste dans la séquence de l'ordinateur

PRESENT-VALUE (N1,N2...) Valeur actuelle d'une série de futurs versements N2 à un taux de N1

RANDOM (E) Nombre aléatoire - paramètre facultatif modifie l'algorithme -

RANGE (N1,N2...) Différence entre maximum et minimum

REM (N1,N2) Reste de N1/N2

REVERSE (X ou A) Inverse l'ordre des caractères dans la chaîne

SIN (N) Sinus de N

SQRT (N) Racine carrée de N

STANDARD-DEVIATION (N1,N2...) Ecart type

SUM (N1,N2...) Somme

TAN (N) Tangente

UPPER-CASE (X) Conversion en majuscules de la chaîne

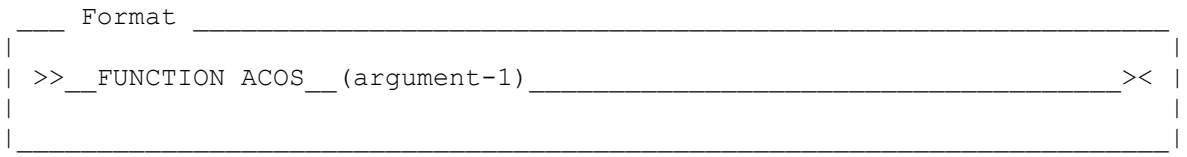
VARIANCE (N1,N2...) Variance

WHEN-COMPILED Date/heure de compilation du programme
AAAAMMJJHHMMSSCC±HHMM

Définition

Retourne une valeur numérique représentant la valeur en radians correspondant à l'arccosinus de l'argument

Syntaxe



Fonction et arguments

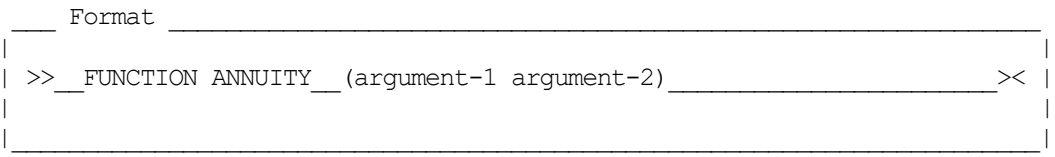
- le type de la fonction est numérique
- argument-1 doit être de type numérique, et compris entre –1 et +1
- la valeur retournée est comprise entre 0 et Pi

Définition

Retourne une valeur numérique représentant le montant payé à la fin de chaque période, pour un emprunt d'une unité à un intérêt argument-1 sur argument-2 périodes, selon la formule

(- **ARGUMENT-2))**

Syntaxe



Fonction et arguments

- le type de la fonction est numérique
- argument-1, intérêt, doit être numérique et supérieur ou égal à zéro

argument-2, nombre de périodes, doit être un entier positif

Exemple

```
       01   INTER-1 PIC 9V99         VALUE 0.07.
       01   PERIO-1 PIC 999       VALUE 10.
       01   SOMME-1 PIC 99999     VALUE 10000.
       01   RESUL-1 PIC +99.99999.
       01   REMBO-1 PIC +999999.9.

      * ANNUITY
           COMPUTE RESUL-1 = FUNCTION ANNUITY (INTER-1 PERIO-1)
           DISPLAY RESUL-1
           DISPLAY " Emprunt de " SOMME-1 " SUR " PERIO-1
            " PERIODES AU TAUX DE " INTER-1 " % "
           COMPUTE REMBO-1 = SOMME-1
                                    *
               FUNCTION ANNUITY (INTER-1 PERIO-1) DISPLAY
           REMBO-1
```

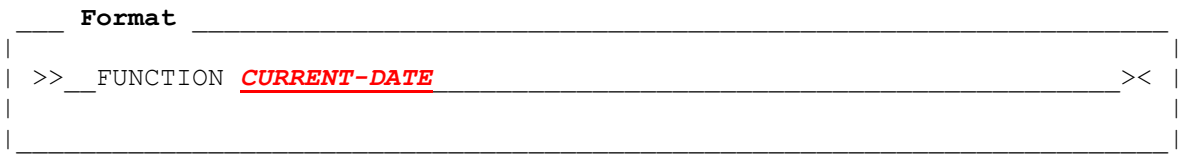
Résultat

```
+00.14238
 Emprunt de 10000 SUR 010 PERIODES AU TAUX DE 007
                                                 %
+001423.8
```

Définition

La fonction intrinsèque CURRENT-DATE (à ne pas confondre avec l'ancien mot réservé CURRENT-DATE) permet d'accéder à la date du jour en 21 caractères de format AAAAMMJJHHMNSSCCshhmn, où AAAA représente l'année, MM le mois, JJ le jour, HH l'heure, MN la minute, SS la seconde, CC le centième de seconde, shhmn le signe et le décalage horaire en heures et minutes par rapport à l'heure GMT.

Syntaxe



Fonction et arguments

le type de la fonction est alphanumérique

Exemple

```
01 DATE-C PIC X(21).
01 DATE-E PIC XXXXBXXBXXBXXBXXBXXBXXBXXXXX.
01 DATE-2 PIC 9(8).

* CURRENT DATE
    MOVE FUNCTION CURRENT-DATE TO DATE-C DATE-E
    DISPLAY " DATE COURANTE : " DATE-C
    DISPLAY "AAAA MM JJ HH MN SS CC SHHMN "
    DISPLAY DATE-E
    MOVE DATE-C(1:8)   TO DATE-2
    DISPLAY " DATE AAAAMMJJ"DATE-2
    MOVE FUNCTION CURRENT-DATE (1:8) TO DATE-2
    DISPLAY " DATE AAAAMMJJ"DATE-2
```

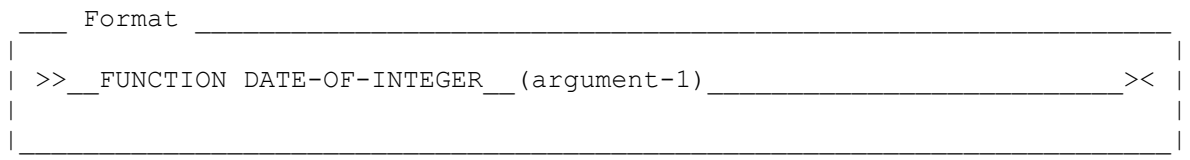
Résultat

```
DATE COURANTE : 1998011217305929+0000
AAAA MM JJ HH MN SS CC SHHMN
1998 01 12 17 30 59 29 +0000 DATE AAAAMMJJ
19980112
DATE AAAAMMJJ 19980112
```

Définition

La fonction DATE-OF-INTEGER transforme un entier exprimant un nombre de jours écoulés depuis le 31 décembre 1600 en une date sous forme AAAAMMJJ.

Syntaxe



Fonction et arguments

le type de la fonction est entier, et le résultat est exprimé en 8 chiffres

argument-1 doit être un entier positif, représentant le nombre de jours écoulés depuis le 31 décembre 1600, et doit avoir une valeur comprise entre 1 (premier janvier 1601) et 3.067.671 (31 décembre 9999)

Exemple

```
       01 DATE-1 PIC 9(8). 01 DATE-3
       PIC 9(8).

      * DATE-OF-INTEGER
           MOVE 1              TO DATE-1
           COMPUTE DATE-3 = FUNCTION DATE-OF-INTEGER (DATE-1)
           DISPLAY " DATE APRES 1 JOUR " DATE-3

           MOVE 3067671        TO DATE-1
           COMPUTE DATE-3 = FUNCTION DATE-OF-INTEGER (DATE-1)
           DISPLAY " DATE APRES " DATE-1 " JOURS " DATE-3
```

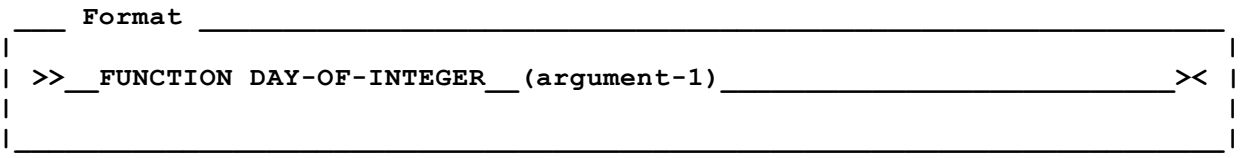
Résultat

```
DATE APRES 1 JOUR 16010101

DATE APRES 03067671 JOURS 99991231
```

Définition

La fonction DAY-OF-INTEGER transforme un entier exprimant un nombre de jours écoulés depuis le 31 décembre 1600 en une date sous forme AAAAQQQ.

Syntaxe



Fonction et arguments

- le type de la fonction est entier, et le résultat est exprimé en 7 chiffres
- argument-1 doit être un entier positif, représentant le nombre de jours écoulés depuis le 31 décembre 1600, et doit avoir une valeur comprise entre 1 (premier janvier 1601) et 3.067.671 (31 décembre 9999)

Exemple

```
       01 DATE-1 PIC 9(8).
       01 DATE-3 PIC 9(8).

      * DAY-OF-INTEGER
           MOVE 1             TO DATE-1
           COMPUTE DATE-3 = FUNCTION DAY-OF-INTEGER (DATE-1)
           DISPLAY " DAY APRES 1 JOUR " DATE-3

           MOVE 3067671       TO DATE-1
           COMPUTE DATE-3 = FUNCTION DAY-OF-INTEGER (DATE-1)

          DISPLAY " DAY APRES " DATE-1 " JOURS " DATE-3
```

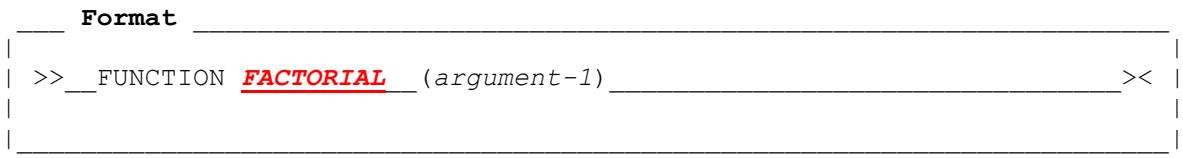
Résultat

```
DAY APRES 1 JOUR 01601001
DAY APRES 03067671 JOURS 09999365
```

Définition

Retourne une valeur entière représentant la factorielle de l'argument spécifié

Syntaxe



Fonction et arguments

- le type de la fonction est entier
- argument-1 doit être de type entier, compris entre 0 et 28 (bornes comprises)

Exemple

```
       01  VAL-1   PIC 99.
       01   RESUL-1 PIC
       01   RESUL-2 COMP-2.


      * FACTORIELLE
           MOVE 0 TO VAL-1
           COMPUTE RESUL-1 = FUNCTION FACTORIAL (VAL-1)
           DISPLAY " FACTORIELLE " VAL-1 " = " RESUL-1

           MOVE 10 TO VAL-1
           COMPUTE RESUL-1 = FUNCTION FACTORIAL (VAL-1)
           DISPLAY " FACTORIELLE " VAL-1 " = " RESUL-1

           MOVE 28 TO VAL-1
           COMPUTE RESUL-2 = FUNCTION FACTORIAL (VAL-1)
           DISPLAY " FACTORIELLE " VAL-1 " = " RESUL-2
```

Résultat

```
FACTORIELLE  00 = +000000001
FACTORIELLE  10 = +003628800
FACTORIELLE  28 =  .30488834461171385E 30
```

Définition

Retourne le plus grand entier dont la valeur est plus petite ou égale à l'argument spécifié

Syntaxe

```
 ___ Format ____________________________________________________________
|                                                                       |
| >>__FUNCTION INTEGER__(argument-1)_____________________________><     |
|                                                                       |
|_______________________________________________________________________|
```

Fonction et arguments

- le type de la fonction est entier
- argument-1 doit être de type numérique

```
01  VAL-2    PIC S99V99999.
01  VAL-2E   PIC +99.99999.
01  RESUL-3  PIC +99.99.
```

- ENTIER

```
MOVE 2.5 TO VAL-2 VAL-2E
COMPUTE RESUL-3 = FUNCTION INTEGER (VAL-2)
DISPLAY " ENTIER " VAL-2E " = " RESUL-3

MOVE -2.5 TO VAL-2 VAL-2E
COMPUTE RESUL-3 = FUNCTION INTEGER (VAL-2)
DISPLAY " ENTIER " VAL-2E " = " RESUL-3

MOVE 2.99 TO VAL-2 VAL-2E
COMPUTE RESUL-3 = FUNCTION INTEGER (VAL-2)
DISPLAY " ENTIER " VAL-2E " = " RESUL-3
```

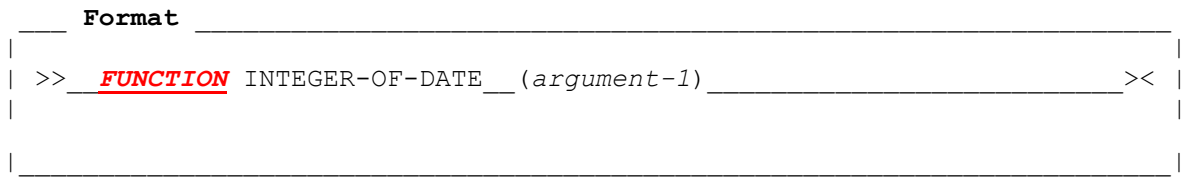
```
ENTIER   +02.50000 = +02.00
ENTIER   -02.50000 = -03.00
ENTIER   +02.99000 = +02.00
```

INTEGER-OF-DATE

Définition

La fonction INTEGER-OF-DATE transforme une date exprimée sous forme AAAAMMJJ en un entier exprimant un nombre de jours écoulés depuis le 31 décembre 1600.

Syntaxe



Fonction et arguments

le type de la fonction est entier, et le résultat est exprimé en 7 chiffres, de 1 à 3.067.671.

argument-1 doit être une date exprimée sous forme AAAAMMJJ (entre 16000101 et 99991231)

Exemple

```
       01 DATE-2 PIC 9(8). 01 DATE-3
       PIC 9(8).

      * INTEGER-OF-DATE
           MOVE 19961231 TO DATE-2
           COMPUTE DATE-3 = FUNCTION INTEGER-OF-DATE (DATE-2)
           DISPLAY " NOMBRE DE JOURS " DATE-2 " = " DATE-3

           MOVE FUNCTION CURRENT-DATE (1:8) TO DATE-2

           COMPUTE DATE-3 = FUNCTION INTEGER-OF-DATE (DATE-2)

           DISPLAY " NOMBRE DE JOURS COURANTS " DATE-
```

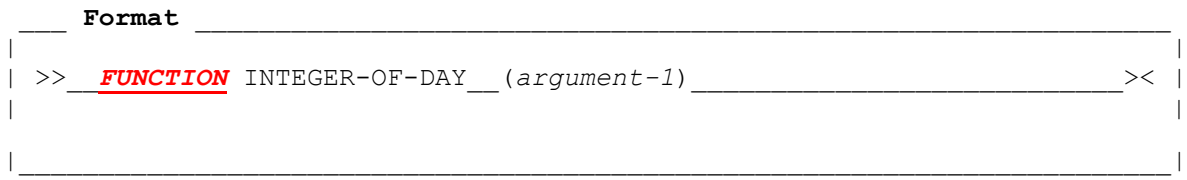
Résultat

```
 NOMBRE DE JOURS 19961231 = 00144636

NOMBRE DE JOURS COURANTS 00145013
```

Définition

La fonction INTEGER-OF-DAY transforme une date exprimée sous forme AAAAQQQ en un entier exprimant un nombre de jours écoulés depuis le 31 décembre 1600.

Syntaxe



Fonction et arguments

le type de la fonction est entier, et le résultat est exprimé en 7 chiffres, de 1 à 3.067.671.

argument-1 doit être une date exprimée sous forme AAAAQQQ (entre 1600001 et 9999365)

Exemple

```
 01  DATE-2 PIC 9(8).
 01  DATE-3 PIC 9(8).
 01  DATE-4 PIC 9(8).
 01  DATE-5 PIC 9(8).
*INTEGER-OF-DAY

    MOVE 1996366 TO DATE-2
    COMPUTE DATE-3 = FUNCTION INTEGER-OF-DAY    (DATE-2)
    DISPLAY " NOMBRE DE JOURS " DATE-2 " = "    DATE-3

    MOVE 2000001 TO DATE-2
    COMPUTE DATE-3 = FUNCTION INTEGER-OF-DAY    (DATE-2)
    DISPLAY " NOMBRE DE JOURS " DATE-2 " = "    DATE-3
    MOVE FUNCTION CURRENT-DATE   (1:8)   TO DATE-2

    COMPUTE DATE-3 =  FUNCTION INTEGER-OF-DATE (DATE-2)
    COMPUTE DATE-4 =  FUNCTION DAY-OF-INTEGER (DATE-3)
    COMPUTE DATE-5 =  FUNCTION INTEGER-OF-DAY  (DATE-4)
    DISPLAY " NOMBRE DE JOURS COURANTS " DATE-5
```

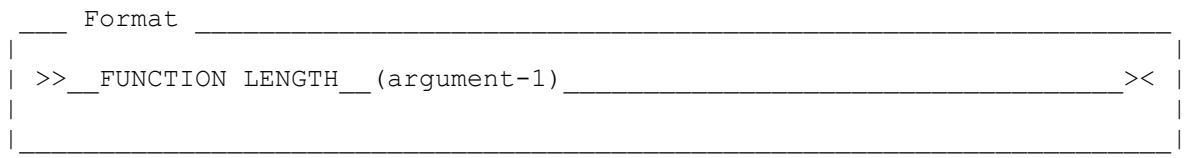
Résultat

```
NOMBRE DE JOURS 01996366 = 00144636
NOMBRE  DE JOURS 02000001 = 00145732
NOMBRE  DE JOURS COURANTS 00145013
```

Définition

Retourne un entier dont la valeur est égale à la longueur en octets de l'argument spécifié

Syntaxe



Fonction et arguments

le type de la fonction est entier, et le résultat est exprimé en 9 chiffres

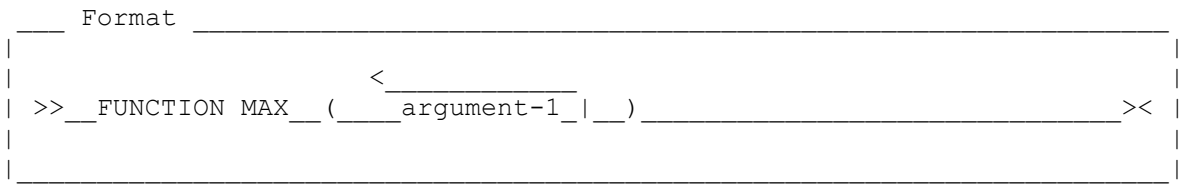
- argument-1 peut être de n'importe quelle nature

  s'il contient la clause OCCURS avec DEPENDING ON, c'est la longueur au moment de l'exécution de la fonction qui est affichée.

Définition

Retourne le contenu de l'argument ayant la plus grande valeur

Syntaxe



Fonction et arguments

- le type de la fonction dépend des arguments : il est
  - **alphanumérique** si les arguments sont alphabétiques ou alphanumériques
  - **entier** si tous les arguments sont entiers
  - numérique si tous les arguments sont numériques et que certains seulement sont entiers
- argument-1 peut être numérique, alphabétique ou alphanumérique

Exemple

```
01 VAL-1    PIC 99.
01 VAL-2    PIC 99.
01 VAL-3    PIC 99.
01 VAL-4    PIC 99.
01 VAL-5    PIC 99.
01 RESUL-1  PIC 99.
01 T-DATE.  PIC +9(9).
01
   05 PIC  X(08) VALUE     "DIMANCHE".
   05 PIC  X(08) VALUE     "LUNDI   ".
   05 PIC  X(08) VALUE     "MARDI   ".
   05 PIC  X(08) VALUE     "MERCREDI".
   05 PIC  X(08) VALUE     "JEUDI   ".
   05 PIC  X(08) VALUE     "VENDREDI".
   05 PIC  X(08) VALUE     "SAMEDI  ".

01  T-DATER REDEFINES T-DATE.
    05 JOURS PIC X(08)                    OCCURS 7.
01
    JOUR-MAX PIC X(08).

    MOVE     3     TO     VAL-1
    MOVE     8     TO     VAL-2
    MOVE    15     TO     VAL-3
    MOVE     4     TO     VAL-4
    MOVE     5     TO     VAL-5
    DISPLAY    " VALEURS :            "," VAL-2    "," VAL-3
                                       VAL-4        "," VAL-5
MAX
   COMPUTE RESUL-1 = FUNCTION
    MAX (VAL-1 VAL-2 VAL-3 VAL-4 VAL-5)
    DISPLAY " MAX = " RESUL-1

   MOVE FUNCTION MAX (JOURS(ALL)) TO JOUR-MAX

   DISPLAY " MAX = " JOUR-MAX
```

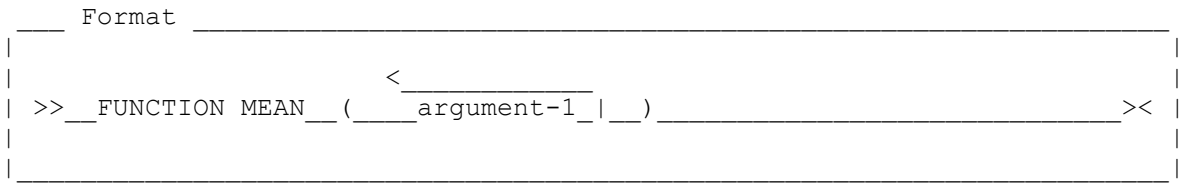
Résultat

```
VALEURS : 03,08,15,04,05
MAX = +000000015
MAX = VENDREDI
```

Définition

Retourne une valeur numérique correspondant à la moyenne arithmétique des arguments

Syntaxe



Fonction et arguments

- le type de la fonction est numérique
- argument-1 doit être de type numérique

Exemple

```
01   VAL-1    PIC    99.
01   VAL-2    PIC    99.
01   VAL-3    PIC    99.
01   VAL-4    PIC    99.
01   VAL-5    PIC    99.
01   RESUL-1 PIC     +9(9).
     MOVE 3    TO     VAL-1
     MOVE 8    TO     VAL-2
     MOVE 15   TO     VAL-3
     MOVE 4    TO     VAL-4
     MOVE 5    TO     VAL-5
     DISPLAY      "   VALEURS    : " VAL-1 "," VAL-2   "," VAL-3
                                      "," VAL-4 "," VAL-
* MEAN
     COMPUTE RESUL-1 = FUNCTION
          MEAN (VAL-1 VAL-2 VAL-3 VAL-4 VAL-5)
          DISPLAY " MEAN = " RESUL-1
```

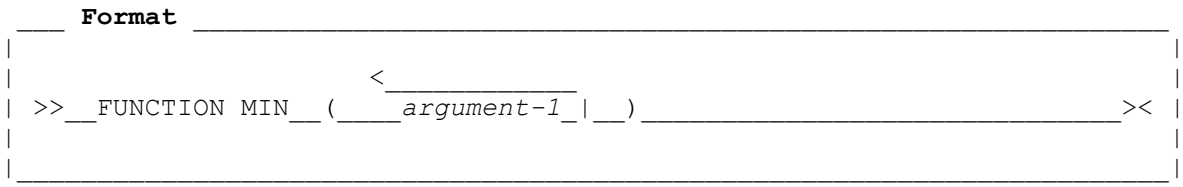
Résultat

```
VALEURS : 03,08,15,04,05

MEAN = +000000007
```

Définition

Retourne le contenu de l'argument ayant la plus petite valeur

Syntaxe



Fonction et arguments

- le type de la fonction dépend des arguments : il est
  - **alphanumérique** si les arguments sont alphabétiques ou alphanumériques
  - **entier** si tous les arguments sont entiers
  - **numérique** si tous les arguments sont numériques et que certains seulement sont entiers
- **argument-1** peut être numérique, alphabétique ou alphanumérique

Exemple

```
       01 VAL-1      PIC  99.
       01 VAL-2      PIC  99.
       01 VAL-3      PIC   99.
       01 VAL-4      PIC   99.
       01 VAL-5      PIC   99.
       01 RESUL-1    PIC +9(9).
       01 T-DATE.
          05 PIC  X(08) VALUE   "DIMANCHE".
          05 PIC  X(08) VALUE   "LUNDI   ".
          05 PIC  X(08) VALUE   "MARDI   ".
          05 PIC  X(08) VALUE   "MERCREDI".
          05 PIC  X(08) VALUE   "JEUDI   ".
          05 PIC  X(08) VALUE   "VENDREDI".
          05 PIC  X(08) VALUE    "SAMEDI ".

01  T-DATER REDEFINES T-DATE.
    05 JOURS PIC X(08)     OCCURS
01  JOUR-MAX PIC X(08).        7.

    MOVE  3 TO VAL-1

    MOVE  8 TO VAL-2
    MOVE 15 TO VAL-3
    MOVE  4 TO VAL-4
    MOVE  5 TO VAL-5
    DISPLAY  " VALEURS :    " VAL-1 "," VAL-2     "," VAL-3
                                "," VAL-4     "," VAL-5
```

```
MIN
   COMPUTE RESUL-1 = FUNCTION
        MIN (VAL-1 VAL-2 VAL-3 VAL-4 VAL-5)
        DISPLAY " MIN = " RESUL-1

  MOVE FUNCTION MIN (JOURS(ALL)) TO JOUR-MAX
  DISPLAY " MIN = " JOUR-MAX
```

Résultat

```
 VALEURS : 03,08,15,04,05 MIN = +000000003 MIN = DIMANCHE
```

Exemple

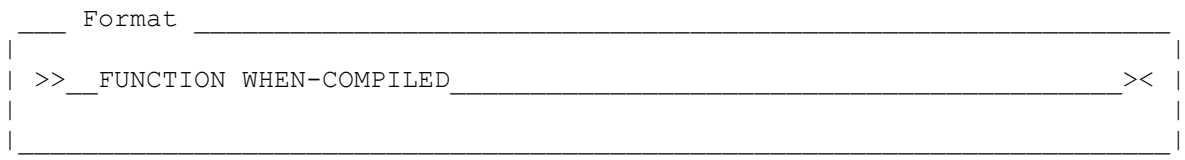
```
01   VAL-1    PIC     99.
01   VAL-2    PIC     99.
01   VAL-3    PIC     99.
01   VAL-4    PIC     99.
01   VAL-5    PIC     99.
01   RESUL-1  PIC       +9(9).
     MOVE 3     TO      VAL-1
     MOVE 8     TO      VAL-2
     MOVE 15    TO      VAL-3
     MOVE 4     TO      VAL-4
     MOVE 5     TO      VAL-5
     DISPLAY     "    VALEURS :   " VAL-1 "," VAL-2       "," VAL-3
                                    "," VAL-4       "," VAL-5

 * SUM
      COMPUTE RESUL-1 = FUNCTION SUM (VAL-1 VAL-2 VAL-3 VAL-4 VAL-5)
      DISPLAY " SUM = " RESUL-1
```

Résultat

```
VALEURS : 03,08,15,04,05
SUM = +000000035
```

Définition

Retourne la date et l'heure de compilation du programme en 21 caractères de format AAAAMMJJHHMNSSCCshhmn, où AAAA représente l'année, MM le mois, JJ le jour, HH l'heure, MN la minute, SS la seconde, CC le centième de seconde, shhmn le signe et le décalage horaire en heures et minutes par rapport à l'heure GMT.



le type de la fonction est alphanumérique

EXEMPLE

```
01 DATE-C PIC X(21).
01 DATE-E PIC XXXXBXXBXXBXXBXXBXXBXXBXXXXX.


* WHEN-COMPILED
    MOVE FUNCTION WHEN-COMPILED TO DATE-C DATE-E
    DISPLAY " DATE COMPILATION : " DATE-C
    DISPLAY "AAAA MM JJ HH MN SS CC SHHMN "
    DISPLAY DATE-E
```

RESULTAT

```
DATE COMPILATION : 1997051411363656+0000
AAAA MM JJ HH MN SS CC SHHMN
1997 05 14 11 36 36 56 +000
```

LANGUAGE ENVIRONMENT FOR MVS

LANGUAGE ENVIRONMENT FOR MVS

- Cet environnement comprend :
  - les programmes de service
  - les options d'exécution
  - les codes retour Language Environment for MVS.
- Language Environment for MVS permet la communication entre programmes écrits dans des langages différents (COBOL for MVS, C, C++, PL/I, FORTRAN).
- Tous ces programmes peuvent échanger des services par le biais de calls dynamiques ou statiques.
- Tous ces programmes peuvent s'interpeller, et ceci quelle que soit la plate-forme ou le système d'exploitation sur lesquels ils se trouvent.
- Un dump standard leur est commun.

Rôles

- simplifier l'écriture du code
- diminuer les coûts de réalisation

- Ils assurent des services, c'est à dire offrent aux réalisateursdes fonctions diverses souvent difficiles à programmer.

- Ils reprennent en partie les mêmes fonctionnalités que les fonctions intrinsèques mais sont exécutés différemment.

- Ils sont activés depuis la PROCEDURE DIVISION par des CALL dynamiques et/ou statiques.

- Ils peuvent être appelés depuis n'importe quelle plate-forme supportée par Language Environment for MVS (MVS, OS2, OS400, etc.),

- ces appels provenant de programmes écrits en COBOL, ASSEMBLEUR, ou en C), programmes batch ou CICS

- Les programmes de service destinés à l'ordinateur central commencent tous par les lettres CEE.
- L'appel se fait par

CALL CEExxx USING parm1, parm2, FC

- Les paramêtres des services sont parm1, ..., parmn.
- La zone "FC" sert à tester leur retour d'exécution et occupe 12 octets.

  - Elle doit être décrite en DATA DIVISION

```
01  FC.
    05  SEV      PIC S9(4)    BINARY.
    05  MSGNO    PIC S9(4)    BINARY.
    05  FLGS     PIC X(1).
    05  FACID    PIC X(3).
    05  ISI      PIC X(4).
```

Retour d'exécution

Quand tout s'est bien passé, la zone "FC" contient du "low-value".

Si un problème s'est produit, une "condition TOKEN" est retournée dans FC :
SEV

- Si SEV = **0** ====> **INFORMATION**
- Si SEV = **1** ====> **WARNING**
  (service réalisé, probablement correct)
- Si SEV = **2** ====> **ERROR**
  (service réalisé, probablement incorrect)
- Si SEV = **3** ====> **SEVERE ERROR**
  (service non réalisé)
- Si SEV = **4** ====> **CRITICAL ERROR**
  (service non réalisé)

Retour d'exécution (suite)

MSGNO

- La zone MSGNO contient le numéro du message d'erreur.

(Le programme CEEMGET permet de retrouver le message associé.)

FLGS

- Si FLGS = 0 ====> FACID assigné par IBM
- Si FLGS = 1 ====> FACID assigné par l'utilisateur

FACID

- FACID est formé de 3 caractères alphanumériques identifiant le produit qui a généré l'erreur.
- ex : FACID = "CEE"

ISI : Instance Specific Information

Voici une liste de correspondance entre les fonctions assurées à la fois par les programmes de service et par les fonctions intrinsèques.

PROGRAMMES DE SERVICE	FONCTIONS INTRINSEQUES
CEESDACS	ACOS
CEESDASN	ASIN
CEESDATN	ATAN
CEESDCOS	COS
CEEDATE	DATE-OF-INTEGER
CEEDATE	DAY-OF-INTEGER
CEEDAYS	INTEGER-OF-DATE
CEEDAYS	INTEGER-OF-DAY
CEESDLOG	LOG LOG10
CEERANO	RANDOM
CEESSMOD	REM
CEESDSIN	SIN
CEESDSQT	SQRT
CEESDTAN	TAN

NOTA

- Les dates du COBOL for MVS et celles données par les programmes de service ou les fonctions intrinsèques ne sont pas calculées à partir des mêmes données.

- Les fonctions intrinsèques de COBOL for MVS s'appuient sur la référence du 31/12/1600.

- Language Environment for MVS s'appuie sur la référence du 15/10/1582 (date d'application du calendrier GREGORIEN en occident).

- Le nombre de jours écoulés depuis le 15/10/1582 s'appelle format LILIAN.

LES PROGRAMMES DE SERVICE

Certains programmes de service sont très puissants.

Ils donnent des résultats bien supérieurs aux fonctions intrinsèques.

- C'est le cas entre autre des programmes de conversion de dates.

  Ils permettent d'obtenir très facilement de multiples formats de date et heure.

- Ils adaptent le format de la date aux normes du pays

  (Par exemple le rang du jour de la semaine)

Voici une liste des programmes de service les plus utilisés, classés par type de service.

Cette liste n'est pas exhaustive.

Pour obtenir une liste plus complète il faut consulter la brochure : IBM LANGUAGE ENVIRONMENT/370 - Programming Guide.

COBOL ET CICS

Les instructions suivantes ne sont pas autorisées

- **OPEN CLOSE**
- **READ WRITE**
- **START REWRITE**
- **DELETE ACCEPT DISPLAY**

A la place, utiliser les commandes CICS pour extraire, mettre à jour, insérer, détruire les données.

- Les options de compilations nécessaires :
  - **RES** (pris par défaut depuis LE)
  - **RENT**
  - **NODYNAM**
  - **LIB** (si le programme contient les instructions COPY ou BASIS)

Les instructions suivantes ne sont pas autorisées :

- **OPEN CLOSE**
- **READ WRITE**
- **REWRITE**

Les options de compilations recommandées :

- **RENT**
- **LIB (si le programme contient les instructions COPY)**
- **DATA(24) (si utilisation de CALL DL/I)**

OPTIONS DE COMPILATION

Attention à la compatibilité des options entre COBOL/VS , COBOL II et COBOL MVS

ADATA

```
>>--+-ADATA---+-----------------------------------------><
    +-NOADATA-+
```

- Défaut : NOADATA
- ADATA : Le système génère un fichier SYSADATA contenant des informations supplémentaires sur la compilation.

ADVANCING

```
>>--+-ADV---+-------------------------------------><
    +-NOADV-+
```

- Défaut : ADV
- ADV : Le système génère le caractère de contrôle de l'imprimante
- NOADV : Le programme place le caractère de contrôle dans le premier octet de l'enregistrement envoyé à l'imprimante

AWO

```
>>--+-AWO---+-----------------------------------------><
    +-NOAWO-+
```

- Défaut : NOAWO
- Avec AWO la clause APPLY WRITE-ONLY est effective pour tous les fichiers séquentiels variables du programme

BUFSIZE

```
>>--BUFSIZE(-+-nnnnn-+-)---------------------------------- ><
             +-nnnK--+
```

- Défaut : 4096
- Abréviation : BUF
- Taille des buffers pour la compilation
- nnnn octets ou nnn K (1024 = 1K)

CMPR2

```
>>--+-CMPR2---+>-------------------------------------------><
    +-NOCMPR2-+
```

- Défaut : NOCMPR2
- Avec CMPR2 le code généré sera compatible avec celui de COBOL II release 2

COMPILE

```
>>--+-COMPILE ----------+-----------------------------------><
    +-NOCOMPILE----------+
    +-NOCOMPILE(-+-W-+-)-+
                 +-E-+
                 +-S-+
```

- Défaut : NOCOMPILE (S)
- Abréviation : C / NOC
- COMPILE : résultat de compilation complet
- NOCOMPILE : résultat simplifié, avec erreurs de syntaxe

CURRENCY

```
>>--+-CURRENCY(littéral)-+   >< +-NOCURRENCY     +
```

- Défaut : NOCURRENCY
- CURRENCY(littéral) : permet l'utilisation d'un signe monétaire par défaut autre que $.
- NOCURRENCY : le signe $ est pris par défaut comme signe monétaire.

DATA

```
>>-- DATA(-+-24-+-)-------------------------------------><
           +-31-+
```

- Défaut : DATA(31)
- Chargement des données au dessus (31) ou en dessous (24) de la barre des 16 megaoctets

Effectif uniquement si option RENT spécifiée

DBCS

```
>>--+-DBCS---+ ----------------------------------------------- >< +-
    NODBCS-+
```

- Défaut : NODBCS
- DBCS permet de reconnaître X'0E' et X'0F' comme les caractères de début et de fin des chaînes DBCS

DECK

```
>>--+-DECK---+ ------------------------------------------------ ><
    +-NODECK-+
```

- Défaut : NODECK
- Abréviation : D / NOD
- DECK : code objet dans fichier SYSPUNCH du JCL

DUMP

```
>>--+-DUMP---+ ------------------------------------------------ ><
    +-NODUMP-+
```

- Défaut : NODUMP
- Abréviation : DU / NODU
- DUMP : Dump système si erreur dans la compilation
-

DYNAM

```
>>--+-DYNAM---+ ------------------------------------------------ ><
    +-NODYNAM-+
```

- Défaut : NODYNAM
- Abréviation : DYN / NODYN
- DYNAM : Appel dynamique aux sous-programmes
- NODYNAM : Appel statique pour les CALL "littéral" et dynamique pour les CALL nom-de-donnée

EXIT

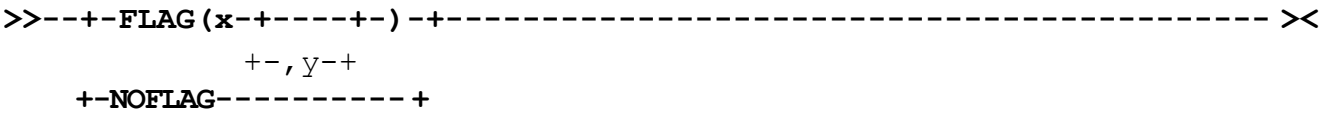
- Défaut NOEXIT
- EXIT(IN-id LIB-id PRT-id ADT-id)
- Permet de spécifier des exits pour le compilateur

FASTSRT

```
>>--+-FASTSRT---+ ------------------------------------------- ><
    +-NOFASTSRT-+
```

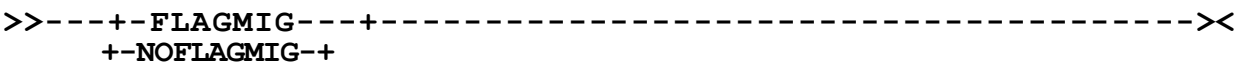
- Défaut : NOFASTSRT
- Abréviation : FSRT / NOFSRT
- FASTSRT : utilisation du tri IBM DFSORT
- inactif si INPUT ou OUTPUT procédure utilisée

FLAG



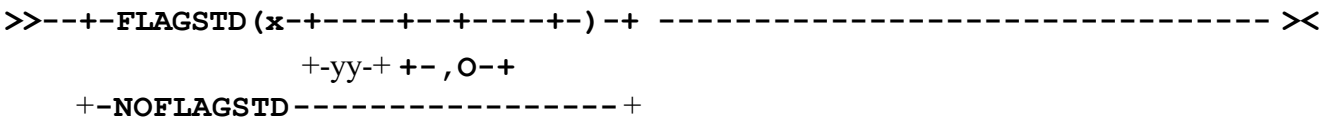
- Défaut : FLAG(I)
- Abréviation : F / NOF
- x,y : I, W, E, S ou U
- FLAG : Gestion des messages d'erreur en fin de programme (x) ou directement à la ligne en erreur (y)
- NOFLAG : Analyse quand même des erreurs

FLAGMIG



- Défaut : NOFLAGMIG
- FLAGMIG permet d'identificateur les éléments du langage différents entre COBOL II et COBOL for MVS

FLAGSTD



- Défaut : NOFLAGSTD
- x : M, I ou H
- y : D, N ou S
- O : O
- FLAGSTD : permet d'obtenir des messages d'information sur le niveau de standardisation des éléments COBOL du programme

IDLGEN

```
>>--+-IDLGEN ----+-----------------------------------------><
   +-NOIDLGEN---+
```

- Défaut NOIDLGEN
- Abréviation : IDL / NOIDL
- Génère de l'IDL (Interface Definition Langage) pour les classes COBOL

INTDATE

```
>>---INTDATE=--+- ANSI----+---------------------------------><
               +-LILIAN---+
```

Défaut INTDATE(ANSI)

- Détermine la date de départ pour les formats entiers de date dans les fonctions intrinsèques
- INTDATE(ANSI) donne un départ au 1 janvier 1601
- INTDATE(LILIAN) spécifie la date de départ lilienne de LE au 15 octobre 1582
- spécifié uniquement à l'installation

LANGUAGE

```
>>--LANGUAGE(XXxxxxxx) ----------------------------------------><
```

- Défaut : LANGUAGE (ENGLISH/UENGLISH/JAPANESE)
- Abréviation : LANG (EN/UE/JA/JP)
- permet de spécifier le langage des messages du compilateur

LIB

```
>>--+-LIB---+----------------------------------------------------><
    +-NOLIB-+
```

- Défaut : NOLIB
- LIB : Obligatoire si utilisation de clauses COPY
- SYSLIB DD = bibliothèques des copy

LINECOUNT

```
>>--LINECOUNT(nnn) ----------------------------------------------><
```

- Défaut : LINECOUNT(60)
- Abréviation : LC(nnn)
- Nombre de lignes de chaque page du listing de compilation

LIST

```
>>--+-LIST---+ ------------------------------------------------><
    +-NOLIST-+
```

- Défaut : NOLIST
- LIST : Expansion du source en langage assembleur
- NOLIST : Pas d'expansion

MAP

```
>>--+-MAP---+ ------------------------------------------------ ><
    +-NOMAP-+
```

- Défaut : NOMAP
- MAP : Informations sur les éléments de la DATA DIVISION

NAME

```
>>--+-NAME-+-------------------------------------------><
            +-(-+-ALIAS---+-)-+
                +-NOALIAS-+
    +-NONAME------------------+
```

- Défaut : NONAME ou NAME (ALIAS) si name spécifié
- NAME génère une carte link-edit NAME pour chaque load-module
- ALIAS permet de générer une carte link-edit ALIAS pour chaque ordre ENTRY

NUMBER

```
>>--+-NUMBER---+ ----------------------------------------------- ><
    +-NONUMBER-+
```

- Défaut : NONUMBER
- Abréviation : NUM / NONUM
- NUMBER : utilisation de la numérotation des colonnes 1 à 6 pour les messages d'erreur
- NONUMBER : pas de génération

NUMPROC

```
>>--NUMPROC(-+-PFD---+-) ---------------------------------------- ><
             +-NOPFD-+
             +-MIG---+
```

- Défaut : NUMPROC(NOPFD)
- NUMPROC(PFD) pas de contrôle de signe au niveau du code objet (plus performant)
- NUMPROC(NOPFD) contrôle de signe au niveau du code objet
- NUMPROC(MIG) aide à la migration de COBOL VS à COBOL for MVS

OBJECT

```
>>--+-OBJECT---+ ----------------------------------------------- ><
    +-NOOBJECT-+
```

- Défaut : OBJECT
- Abréviation : OBJ / NOOBJ
- OBJET : code objet dans fichier SYSLIN du JCL

OFFSET

```
>>--+-OFFSET---+ ------------------------------------------------ ><
    +-NOOFFSET-+
```

- Défaut : NOOFFSET
- Abréviation : OFF / NOOFF
- OFFSET : liste condensée de la PROCEDURE DIVISION
- NOOFFSET : pas de génération

RENT

```
>>--+-RENT---+ ------------------------------------------------ ><
    +-NORENT-+
```

- Défaut : NORENT
- RENT : Code réentrant
- NORENT : Code non réentrant
- Ne pas utiliser NORENT avec CICS

RMOD

```
>>--RMODE(-+-AUTO-+-) ---------------------------------------- ><
           24---+
           +-ANY--+
```

- Défaut : RMOD(AUTO)
- Permet aux programmes NORENT d'avoir RMODE(ANY)
- RMOD(24) possible

SEQUENCE

```
>>--+-SEQUENCE---+-------------------------------------------><
    NOSEQUENCE-+
```

- Défaut : SEQUENCE
- Abréviation : SEQ / NOSEQ
- SEQUENCE : Contrôle la séquence de lignes de source
- NOSEQUENCE : Pas de contrôle

SIZE

```
>>--SIZE(-+-nnnnn-+-)-----------------------------------------><
          +-nnnK--+
          +-MAX---+
```

- Défaut : SIZE(MAX)
- Abréviation : SZ
- Taille allouée pour le compilateur
- nnnn : Taille en octets (minimum 655340)
- nnnK : Taille en K octets (minimum 640k)
- MAX : Allocation de toute la mémoire possible
- L'option SIZE(MAX) ne doit pas être utilisée, le compilateur peut utiliser 16 mégaoctets....

SOURCE

```
>>--+-SOURCE---+ ------------------------------------------------ ><
    +-NOSOURCE-+
```

- Défaut : SOURCE
- Abréviation : S / NOS
- SOURCE : Liste du programme source à la compilation
- NOSOURCE : Pas de liste

SPACE

```
>>--SPACE(-+-1-+-) ------------------------------------------- ><
           +-2-+
           +-3-+
```

- Défaut : SPACE(1)
- Saut de 1, 2 ou 3 lignes dans la liste du programme source à la compilation

SSRANGE

```
>>--+-SSRANGE---+ ----------------------------------------- ><
    +-NOSSRANGE-+
```

- Défaut : NOSSRANGE
- Abréviation : SSR / NOSSR
- SSRANGE : génère le code pour contrôler les limites de table
- NOSSRANGE : pas de contrôle

TERMINAL

```
>>--+-TERMINAL---+ ------------------------------------------ ><
    +-NOTERMINAL-+
```

- Défaut : NOTERMINAL
- Abréviation : TERM / NOTERM
- TERMINAL : Envoi en cours de compilation des messages sur le fichier SYSTERM
- NOTERMINAL : Pas de message

TEST

```
>>--+-TEST---+ ----------------------------------------------- ><
    +-NOTEST-+
```

- Défaut : NOTEST
- Abréviation : TES / NOTES
- TEST : Active l'intéractif debug batch ou TSO
- NOTEST : pas d'activation

TRUNC

```
>>--TRUNC(-+-STD-+-) ---------------------------------------- ><
           +-OPT-+
           +-BIN-+
```

- Défaut : TRUNC(STD)
- STD : Valeur maximum d'une donnée conforme au nombre de digits de la clause PICTURE
- OPT : option de performance, troncature soit suivant le nombre de digits de la clause PICTURE soit suivant les valeurs maximums demi-mot, mot, double mot
- BIN : Assume les valeurs maximums demi-mot, mot ...

TYPCHECK

```
>>-+-TYPECHK---+ ----------------------------------------------- >< +-
   NOTYPECHK-+
```

- Défaut : NOTYPECHK.
- Abréviation : TC / NOTC
- Permet de contrôler les type en COBOL Orienté Objet

VBREF

```
>>--+-VBREF---+ ------------------------------------------------ ><
    +-NOVBREF-+
```

- Défaut : NOVBREF
- VBREF : Cross référence des verbes
- NOVBREF : Pas de génération

WORD

```
>>--+-WORD(xxxx)-+ ----------------------------------------><
    +-NOWORD  ---+
```

- Défaut : NOWORD
- Abréviation : WD / NOWD
- WORD(xxxx) : une autre table de mots réservés est utilisée pendant la compilation
- xxxx : suffixe du nom de cette table toujours préfixée par IGYC
- NOWORD : conforme à ANS

XREF

```
>>--+-XREF-+ ------------ +--+ ---------------------------------><
          +-(-+-SHORT-+-)-+
              +-FULL--+
    +-NOXREF----------------+
```

- Défaut : NOXREF
- XREF : Cross référence des noms de donnée et procédures
- NOXREF : Pas de génération

ZWB

```
>>--+-ZWB---+ ----------------------------------------------><
    <+-NOZWB-+
```

- Défaut : ZWB
- ZWB : Comparaison sans signe pour décimale externe
  NOZWB : Comparaison avec signe

LISTE DES MOTS RESERVES COBOL

LISTE DES MOTS RESERVES COBOL

ACCEPT
ACCESS
ADD
ADDRESS
ADVANCING
AFTER
ALL
ALLOWING
ALPHABET
ALPHABETIC
ALPHABETIC-LOWER
ALPHABETIC-UPPER
ALPHANUMERIC
ALPHANUMERIC-EDITED
ALSO
ALTER
ALTERNATE
AND
ANY
APPLY
ARE
AREA
AREAS
ARITHMETIC
ASCENDING
ASSIGN
AT
AUTHOR
AUTOMATIC
B-EXOR
B-LESS
B-NOT
B-OR
BASIS
BEFORE
BEGINNING
BINARY
BIT
BITS
BLANK
BLOCK
BOOLEAN
BOTTOM
BY
CALL
CANCEL
CBL
CD
CF
CH
CHARACTER
CHARACTER
S
CLASS
CLASS-ID

LISTE DES MOTS RESERVES COBOL

CONTAINS
COLLATING
COLUMN
COM-REG
COMMA
COMMIT
COMMON
COMMUNICATION
COMP
COMP-1
COMP-2
COMP-3
COMP-4
COMP-5
COMP-6
COMP-7
COMP-8
COMP-9
COMPUTATIONAL
COMPUTATIONAL-1
COMPUTATIONAL-2
COMPUTATIONAL-3
COMPUTATIONAL-4
COMPUTATIONAL-5
COMPUTATIONAL-6
COMPUTATIONAL-7
COMPUTATIONAL-8
COMPUTATIONAL-9
CONTENT
CONTINUE
CONTROL
CONTROLS
CONVERTING
COPY
CORR
CORRESPONDING
COUNT
CURRENCY
CURRENT
CYCLE
DATA
DATE
DATE-COMPILED
DATE-WRITTEN
DAY
DAY-OF-WEEK
DB
DB-ACCESS-CONTROL-KEY
DB-DATA-NAME
DB-EXCEPTION
DB-RECORD-NAME
DB-SET-NAME
DB-STATUS
DBCS
DEBUG-ITEM
DEBUG-LINE

LISTE DES MOTS RESERVES COBOL

DEBUG-NAME
DEBUG-SUB-1
DEBUG-SUB-2
DEBUG-SUB-3
DEBUGGING
DECIMAL-POINT
DECLARATIVES
DEFAULT
DELETE
DELIMITED
DELIMITER
DEPENDING
DESCENDING
DESTINATION
DETAIL
DISABLE
DISCONNECT
DISPLAY
DISPLAY-1
DISPLAY-2
DISPLAY-3
DISPLAY-4
DISPLAY-5
DISPLAY-6
DISPLAY-7
DISPLAY-8
DISPLAY-9
DIVIDE
DIVISION
DOWN
DUPLICATE
DUPLICATE
DUPLICATES
DYNAMIC
EGCS
EGI
EJECT
ELSE
EMI
EMPTY
ENABLE
END
END-ADD
END-CALL
END-COMPUTE
END-DELETE
END-DISABLE
END-DIVIDE
END-ENABLE
END-EVALUATE
END-IF
END-INVOKEEND
END-OF-PAGE
END-PERFORM
END-READ
END-RECEIVE
END-RETURN
END-REWRITE
END-SEARCH
END-SEND
END-START
END-STRING

LISTE DES MOTS RESERVES COBOL

END-SUBTRACT
END-TRANSCEIVE
END-UNSTRING
END-WRITE
ENDING
ENTER
ENTRY
ENVIRONMENT
EOP
EQUAL
EQUALS
ERASE
ERROR
ESI
EVALUATE
EVERY
EXACT
EXCEEDS
EXCEPTION
EXCLUSIVE
EXIT
EXTEND
EXTERNAL
FALSE
FD
FETCH
FILE
FILE-CONTROL
FILLER
FINAL
FIND
FINISH
FIRST
FOOTING
FOR
FORM
FORMAT
FREE
FROM
FUNCTION
GENERATE
GET
GIVING
GLOBAL
GO
GOBACK
GREATER
GROUP
HEADING
HIGH-VALUE
HIGH-VALUES
I-O
I-O-CONTROL
ID
IDENTIFICATION
IF
IN
INDEX
INDEX-1
INDEX-2

LISTE DES MOTS RESERVES COBOL

INDEX-3
INDEX-4
INDEX-5
INDEX-6
INDEX-7
INDEX-8
INDEX-9
INDEXED
INDICATE
INHERITS
INITIAL
INITIALIZE
INITIATE
INPUT
INPUT-OUTPUT
INSERT
INSPECT
INSTALLATION
INTO
INVALID
INVOKE
IS
JUST
JUSTIFIED
KANJI
KEEP
KEY
LABEL
LAST
LD
LESS
LIMIT
LIMITS
LINAGE
LINAGE-COUNTER
LINE
LINE-COUNTER
LINES
LINKAGE
LOCALLY
LOCAL-STORAGE
LOCK
LOW-VALUE
LOW-VALUES
MEMBER
MEMORY
MERGE
MESSAGE
METACLASS
METHOD
METHOD-ID
MODE
MODIFY
MODULES
MORE-LABELS
MOVE
MULTIPLE
MULTIPLY
NATIVE
NEXT

LISTE DES MOTS RESERVES COBOL

NO
NORMAL
PARAGRAPH
PASSWORD
PERFORM
PF
PH
PIC
PICTURE
PLUS
POINTER
POSITION
POSITIVE
PRESENT
PREVIOUS
PRINTING
PRIOR
PROCEDURE
PROCEDURE-POINTER
PROCEDURES
PROCEED
PROCESSING
PROGRAM
PROGRAM-ID
PROTECTED
PURGE
QUEUE
QUOTE
QUOTES
RANDOM
RANGE
RD
READ
READY
NOT
NULL
NULLS
NUMBER
NUMERIC
NUMERIC-EDITED
OBJECT
OBJECT-COMPUTER
OCCURS
OF
OFF
OMITTED
ON
ONLY
OPEN
OPTIONAL
OR
ORDER
ORGANIZATION
OTHER
OUTPUT
OVERFLOW
OVERRIDE
OWNER
PACKED-DECIMAL
PADDING
PAGE
PAGE-COUNTER

LISTE DES MOTS RESERVES COBOL

PARAGRAPH

READ

PASSWORD
PERFORM REALM

RECORDING
RECORDS
RECURSIVE
R
REDEFINES
REEL
REFERENCE
REFERENCES
ECORD-NAME
RELATION
RELATIVE
RELEASE
RELOAD
REMAINDER
REMOVAL
RENAMES
REPEATED
REPLACE
REPLACING
REPORT
REPORTING
REPORTS
REPOSITORY
RERUN
RESERVE
RESET
RETAINING
RETRIEVAL
RETURN

LISTE DES MOTS RESERVES COBOL

PF
PH
PIC
PICTURE
PLUS
POINTER
POSITION
POSITIVE
PRESENT
PREVIOUS
PRINTING
PRIOR
PROCEDURE
PROCEDURE-POINTER
PROCEDURES
PROCEED
PROCESSING
PROGRAM
PROGRAM-ID
PROTECTED
PURGE
QUEUE
QUOTE
QUOTES
RANDOM
RANGE
RD
RECEIVE
RECONNECT
RECORD

LISTE DES MOTS RESERVES COBOL

RETURN-CODE
RETURNING
REVERSED
REWIND
REWRITE
RF
RH
RIGHT
ROLLBACK
ROUNDED
RUN
SAME
SD
SEARCH
SECTION
SECURITY
SEGMENT
SEGMENT-LIMIT
SELECT
SELF
SEND
SENTENCE
SEPARATE
SEQUENCE
SEQUENTIAL
SERVICE
SESSION-ID
SET
SHARED
SHIFT-IN/OUT

LISTE DES MOTS RESERVES COBOL

SIGN
SIZE
SKIP1
SKIP2
SKIP3
SORT
SORT-CONTROL
SORT-CORE-SIZE
SORT-FILE-SIZE
SORT-MERGE
SORT-MESSAGE
SORT-MODE-SIZE
SORT-RETURN
SOURCE
SOURCE-COMPUTER
SPACE
SPACES
SPECIAL-NAMES
STANDARD
STANDARD-1
STANDARD-2
STANDARD-3
STANDARD-4
START
STATUS
STOP
STORE
STRING
SUB-QUEUE-1
SUB-QUEUE-2
SUB-SCHEMA
SUBTRACT
SUM
SUPER
SUPPRESS
SYMBOLIC
SYNC
SYNCHRONIZED
TABLE
TALLY
TALLYING
TAPE
TENANT
TERMINAL
TERMINATE
TEST
THAN
THEN
THROUGH
THRU
TIME
TIMEOUT
TIMES
TITLE
TO
TOP
TRACE
TRAILING
TRANSCEIVE
TRUE
TYPE

LISTE DES MOTS RESERVES COBOL

UNEQUAL	<
UNIT	<=
UNSTRING	+
UNTIL	*
UP	**
UPDATE	-
UPON	/
USAGE	>
USAGE-MODE	>=
USE	=
USING	
VALID	
VALIDATE	
VALUE	
VALUES	
VARYING	
WAIT	
WHEN	
WHEN-COMPILED	
WITH	
WITHIN	
WORDS	
WORKING-STORAGE	
WRITE	
WRITE-ONLY	
ZERO	
ZEROES	