Instructions cobol et expressions

1. La division IDENTIFICATION

Elle identifie le programme

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  nom-de-programme.
[AUTHOR.  [rubrique-commentaire]]
[INSTALLATION. [rubrique-commentaire]]
[DATE-WRITTEN. [rubrique-commentaire]]
[DATE-COMPILED. [rubrique-commentaire]]
[SECURITY. [rubrique-commentaire]]
[REMARKS. [rubrique-commentaire]]
```

Tous les paragraphes autres que `PROGRAM-ID` sont obsolètes.

Généralement, seuls les 8 premiers caractères du nom du programme sont évalués.
ANS85 introduit un second format pour `PROGRAM-ID`

```
PROGRAM-ID.  nom-de-programme [IS [COMMON] [INITIAL] PROGRAM].
```

`COMMON`: uniquement pour les programmes contenus
`INITIAL`: programme remis à son état initial à chaque appel.
Cf Communication inter-programme

2. La division ENVIRONMENT

Elle contient la description du **système** utilisé et des informations relatives à la gestion des entrées-sorties.

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. ordinateur-origine [WITH DEBUGGING MODE].
OBJECT-COMPUTER. ordinateur-objet.
[SPECIAL-NAMES. [rubrique-des-noms-spéciaux]]
[INPUT-OUTPUT SECTION.
FILE-CONTROL. rubrique-de-gestion-de-fichier ....
[I-O-CONTROL. [rubrique-de-gestion-d'entrée-sortie]]]
```

En ANS85, cette division est facultative.

Le paragraphe `SPECIAL-NAMES` a le format suivant (partiel).

```
SPECIAL-NAMES.
```

[définition des mnémoniques du réalisateur]
[définition de l'alphabet]
[CURRENCY SIGN IS lit-1]
[DECIMAL-POINT IS COMMA]
[NUMERIC SIGN IS TRAILING SEPARATE]
[CONSOLE IS CRT]

- Le symbole monétaire (CURRENCY SIGN), par défaut $, doit être un caractère **autre** que :

un chiffre, une minuscule, un symbole arithmétique ou de ponctuation, les lettres A, B, C, D, P, R, S, V, X, Z.

Exemple : `CURRENCY SIGN IS "F".`

- La clause DECIMAL-POINT IS COMMA permet de permuter le rôle du point et de la virgule dans une clause PICTURE et dans les littéraux numériques.

Les clauses [NUMERIC SIGN IS TRAILING SEPARATE] et [CONSOLE IS CRT] relèvent du dialecte Microfocus.

L'INPUT-OUTPUT SECTION est détaillée dans le chapitre des fichiers.

3. La division des données

Elle contient la description de toutes les informations manipulées par le programme.

3.1. Organisation générale :

```
DATA DIVISION.

FILE SECTION.

rubrique-de-description-de-fichier
rubrique-de-description-d'article          ...


WORKING-STORAGE SECTION.

rubrique-de-description-de-niveau-77
rubrique-de-description-d'article          ...


LINKAGE SECTION.

rubrique-de-description-de-niveau-77
rubrique-de-description-d'article          ...
```

Trois sections :

- `FILE SECTION` : description des fichiers (voir chap. 6)

- `WORKING-STORAGE SECTION` : description des données internes au programme.
- `LINKAGE SECTION` : description des paramètres lorsque le programme est un sous-programme (détaillée au : Communications inter-programmes).

Signalons toutefois qu'à quelques exceptions près, la description d'une rubrique se fait de façon identique dans les trois sections. Dans les paragraphes qui suivent, nous nous placerons en priorité au niveau de la `WORKING-STORAGE SECTION`.

Ces sections peuvent être suivies de la `COMMUNICATION SECTION` et de la `REPORT SECTION` si l'on utilise respectivement les modules de communication et d'édition , ainsi que de sections spécifiques aux dialectes (`SCREEN, LOCAL-STORAGE`, ...).

3.2. Description générale d'une rubrique

Elle est de la forme :

numéro-de-niveau { nd | FILLER } [[clause] ...] .

- le numéro de niveau est compris entre 1 à 49 ou vaut 77, 66 ou 88.
- nd : nom de donnée, identifie la rubrique.
- le mot clé `FILLER` remplace le nom d'une rubrique qui n'est jamais référencée exeplicitement par les instructions du programme ; il peut être utilisé dans plusieurs descriptions de rubriques du même programme.

La description d'une rubrique structurée se réduit au couple (n° de niveau, nom) ; les rubriques élémentaires doivent être détaillées par des clauses (sauf les index).

Exemple :

```
02 DATE-NAISSANCE.
  03 JOUR ....        .
  03 MOIS ....        .
  03 ANNEE ...        .
```

3.3. La clause PICTURE

Elle sert à définir le format externe ou image d'une rubrique élémentaire (nombre de chiffres, position de la marque décimale, etc...)

{ PICTURE | PIC } `IS` Format

Le format est constitué d'une suite de symboles décrits ci- dessous (liste partielle) :

- `9` : représente un chiffre
- `V` : indique la position **virtuelle** du séparateur décimal ; unique dans un format
- `S` : indique une donnée signée ; unique dans un format, il doit être le premier symbole du format
- `X` : représente un caractère quelconque
- `.` : matérialise le séparateur décimal : le format interne de la rubrique doit être le format décimal étendu ; son utilisation exclut celle de V et de S.

♦ Les 3 premiers symboles permettent de décrire des données de type numérique : le symbole X permet de décrire des données de type alphanumérique (chaînes de caractères) ; la présence du symbole **.** dans un format indique une donnée de type numérique édité.

♦ La description du format peut comporter jusqu'à **30** symboles ;

♦ L'emploi de facteurs de répétition est conseillé : `X(5)` au lieu de `XXXXX`

♦ Cette clause est obligatoire pour toutes les rubriques élémentaires. Elle est interdite pour les index.

Exemple :

1) `PIC X(5)`	chaînes de 5 caractères	
2) `PIC 9(3)`	entier de 3 chiffres non signé	
3) `PIC 99V9`	nombre décimal non signé ;	2 chiffres pour la partie entière, 1 chiffre pour la partie décimale
4) `PIC S9(5)`	entier de 5 chiffres, signé	
5) `PIC 99.9`	nombre décimal en format numérique édité ; le séparateur décimal est matérialisé par un point.	

Remarquons que la rubrique de l'exemple 5 est de classe alphanumérique (du fait de la présence du point) alors que les rubriques des exemples 2 à 4 sont de type numérique.

Il existe d'autres symboles pour décrire un format numérique édité (*Cf Module Edition*)

Capacité maximum d'une rubrique :

- 18 chiffres, signe exclu, pour une rubrique numérique.
- 31 caractères pour une rubrique numérique éditée
- 32768 caractères pour une rubrique alphanumérique

Modification de Référence

En ANS85, il est possible de référence une sous-chaîne d'une donnée alphanumérique:

nd(*début*:[*longueur*]) désigne la sous-chaîne de nd débutant à la position *début* et comportant *longueur* caractères.

Le rang du premier caractère est 1.

début et *longueur* est une expression numérique quelconque
Si *longueur* n'est pas précisée, elle porte jusqu'au dernier caractère de nd.

Si nd est de type autre qu'alphanumérique, on le considère comme redéfini dans une donnée de ce type de la même taille que nd.

Exemple

```
77  DONNEE PIC X(10)  VALUE "abcdefghij"
```

`DONNEE(5:2)` renvoie "ef" tout comme `DONNEE(I:I-3)` si I vaut 5
`DONNEE(5:)` renvoie "efghij"

3.4. La clause USAGE

Elle permet de préciser le format interne d'une rubrique.

```
USAGE  IS  { DISPLAY
             COMPUTATIONAL
             BINARY
             COMPUTATIONAL-3
             PACKED-DECIMAL }   ou autres, suivant dialectes
```

- `DISPLAY` : représentation codée de type non numérique ou décimale étendue (cf. §2.5.2.2.) ; valeur prise par défaut

- `COMPUTATIONAL-3` ou `COMP-3` ou `PACKED-DECIMAL`: représentations décimales condensées signées.

- `COMPUTATIONAL-1` `(COMPUTATIONAL-2)` = virgule flottante simple (double) précision

- `COMPUTATIONAL` ou `COMP` ou `BINARY` : format binaire sur 32 bits (moins de 10 chiffres dans la PICTURE)

  Cette clause peut s'appliquer à une rubrique élémentaire ou structurée ; dans ce cas, elle est valable pour toutes les rubriques élémentaires qui composent la rubrique structurée.

3.5. La clause REDEFINES

Elle permet de redéfinir la **structure** d'une rubrique (et non sa ***valeur***).

n°-de-niveau nd1 REDEFINES nd2

Règles du langage :

- cette clause doit suivre immédiatement nd1.
- nd1 et nd2 doivent être de même niveau
- nd2 ne peut contenir la clause REDEFINES mais nd2 peut être subordonné à une rubrique qui contient cette clause.
- cette clause est interdite pour les rubriques de niveau 01 de la FILE SECTION.
- nd1 peut être remplacé par FILLER

Principe :

La redéfinition de la rubrique nd2 commence à nd1 et se termine lorsqu'un numéro de niveau inférieur ou égal à celui de nd1 est rencontré.

```
02    D1.
      04 NOM      PIC X(20).
      04 PRENOM   PIC X(20).
02    D2 REDEFINES   D1.
      03 ADRESSE    PIC X(20).
      03 C-POSTAL   PIC X(5).
      03 LOCALITE   PIC X(15).
```

En cas de redéfinitions multiples, c'est toujours la rubrique d'origine qui est citée en nd2.

```
03    DATE-1.
        04  JOUR-1    PIC 99.
        04  MOIS-1    PIC 99.
        04  AN-1      PIC 99.
03    DATE-2 REDEFINES   DATE-1 PIC 9(6).
03    DATE-3 REDEFINES   DATE-1.
        04  FILLER  PIC X.
        04  JOUR-3  PIC 9(3).
        04  AN-3    PIC 99.
```

Sauf si les rubriques nd1 et nd2 sont de niveau 01, les longueurs de la rubrique de redéfinition et de la rubrique d'origine doivent être égales.

3.6. La clause VALUE

Elle permet de préciser la valeur initiale d'une rubrique ; une rubrique déclarée sans valeur initiale contient une valeur quelconque non définie.

```
VALUE IS  littéral
```

Le littéral doit être du même type que la rubrique.

Cette clause est interdite :

- dans la `FILE SECTION`
- à l'intérieur d'une rubrique de redéfinition.

Exemple

```
02  D1  PIC  X(5)  VALUE "DEBUT".
03  N1  PIC  99V9 VALUE 5.2.
```

Une seconde utilisation de la clause `VALUE` permet la définition de noms-conditions.

Un **nom-condition** est une référence symbolique assignée à une ou plusieurs **valeurs** spécifiques d'une variable donnée, ou à un inverseur.

Il sera défini soit dans le paragraphe `SPECIAL-NAMES`, soit en `DATA DIVISION`. Dans ce dernier cas, il est repéré par un numéro de niveau `88`.

Exemple :
```
01  MOUVEMENT.
    02  INDIC      PIC 9(6).
    02  TYP        PIC 9.
                  88  LECTURE  VALUE 1.
                  88  MODIF    VALUE 2.
                  88  SUPPRES  VALUE 4.

IF LECTURE.................
       ELSE IF MODIF...........
```

L'intérêt des noms-conditions est donc d'améliorer la lisibilité du pg ; ils permettent aussi de simplifier l'énoncé de conditions, car un seul nom-condition peut être assigné à plusieurs valeurs en particulier grâce à la locution `THRU` :

Format général :

```
                          ┌ VALUE IS  ┐ ┌                ┌ THROUGH ┐                 ┐ ┐
88 nom-condition           │           │ │ littéral-1     │         │ littéral-2      │ │ …
                          └ VALUES ARE┘ └                └ THRU    ┘                 ┘ ┘
```

Exemples :
```
VALUES ARE 1  2   16.
VALUES ARE 4  THRU 12.
VALUES ARE 5  THRU 9  13  17.
```

Un nom-condition doit suivre la description de la rubrique à laquelle il est associé ; cette dernière peut être quelconque à l'exception de :

- un autre nom-condition
- une rubrique de niveau 66 (clause RENAMES)
- un index
- un groupe contenant des données avec JUSTIFIED, SYNCHRONIZED, et USAGE (autre que DISPLAY).

Si la référence à une variable conditionnelle nécessite l'indexation ou l'indiçage, la référence à l'un quelconque de ses noms-conditions la nécessite de la même façon.

Les règles d'alignement appliquées sont celles décrites au chapitre 2 § 3.

Utilisation des constantes figuratives

Ce sont des constantes identifiées par des mots clés ; elles ont des valeurs particulières.

ZERO ZERO ZEROES	représente soit "0", soit un ou plusieurs caractères "0" selon le contexte
SPACE SPACES	représente un ou plusieurs caractères espace
HIGH-VALUE HIGH-VALUES	représente un ou plusieurs caractères de rang le plus élevé dans l'ordre de classement ASCII. La valeur la plus élevée est FF hexadécimale.
LOW-VALUE LOW-VALUES	représente un ou plusieurs caractères de rang le plus bas dans l'ordre de classement ASCII. La valeur la plus basse est 00 hexadécimal.
QUOTE QUOTES	représente un ou plusieurs caractères guillemets (") ; le mot QUOTE ou QUOTES ne peut être utilisé à la place d'un guillemet dans un programme origine pour borner un littéral non- numérique. Ainsi QUOTE ABD QUOTE est une façon incorrecte d'écrire le littéral non numérique "ABD".
ALL *littéral*	représente une ou plusieurs fois le *littéral* non- numérique.

- La première (ZERO) peut s'appliquer à des rubriques de type numérique ou non numérique ; les autres sont de type non numérique.

- Elles s'écrivent sans guillemet.
- Contrairement aux autres constantes, leur longueur dépend de la longueur des rubriques qu'elles servent à initialiser.

Exemples :
```
D1   PIC   X(5)   VALUE   "0".        0bbbb
D1   PIC   X(5)   VALUE ZERO.         00000
D1   PIC   X(5)   VALUE ALL"-"        -----
```

3.7. La clause BLANK

Elle permet de remplacer la valeur nulle d'une rubrique de type numérique ou numérique édité par des espaces.
Le format interne de la rubrique doit être DISPLAY.

```
BLANK   WHEN   { ZERO  }
               { ZEROS }
               { ZEROES}
```

La rubrique sera considérée comme étant du type numérique édité.

Exemple : `D1  PIC  9(5)  BLANK ZERO.`

Si D1 est affecté d'une valeur nulle, la rubrique sera remplie par 5 espaces.

3.8. La clause SYNCHRONIZED

Elle permet de préciser l'**alignement** d'une rubrique élémentaire sur les bornes d'adressage spécifiques de la mémoire de l'ordinateur à savoir :

- une adresse multiple de 2 pour un mot

- une adresse multiple de 4 pour un double mot.

```
{SYNCHRONIZED}   { LEFT  }
{SYNC        }   { RIGHT }
```

Elle ne peut s'utiliser que pour des rubriques élémentaires de format interne binaire.

Les options LEFT et RIGHT sont équivalentes.

Les rubriques de niveau 1 et 77 commencent à des limites de mot.

3.9. La clause JUSTIFIED

Elle permet, associée à une zone de réception alphabétique ou alphanumérique, de force l'alignement à droite, plutôt qu'à gauche (alignement standard).

```
{ JUSTIFIED } RIGHT
{ JUST      }
```

3.10. La clause SIGN

Elle permet de préciser la représentation interne du signe pour une rubrique numérique de format décimal étendu. (cf.§ 2.5.2.2-2).

```
[SIGN IS]  { LEADING  } [SEPARATE CHARACTER]
           { TRAILING }
```

- Elle peut être spécifiée pour un groupe.
- le format externe (PICTURE) doit comporter le symbole S
- la clause indique que le signe occupe dans la représentation interne une position séparée à gauche (LEADING) ou à droite (TRAILING).
- l'absence de cette clause pour une rubrique numérique signée, de format interne décimal étendu, entraîne une représentation du signe superposée avec celle du chiffre de droite.

Exemple : `D1 PIC S999V9 SIGN TRAILING SEPARATE`

3.11. La clause RENAMES

Elle permet un regroupement alternatif des rubriques.

```
66 nd-1 RENAMES nd-2 { THROUGH } nd-3
                     { THRU    }
```

- Interdite aux niveaux 01, 66, 77, 88.

Exemple:

```
01  EMPLOYE.
    02  NOM          PIC X(20).
    02  PRENOM       PIC X(15).
    02  RUE          PIC X(30).
    02  CP           PIC 9(5).
    02  VILLE        PIC X(25).
    66 IDENTITE  RENAMES NOM THRU PRENOM.
```

```
66  ADRESSE   RENAMES RUE THRU VILLE.
```

3.12. Exemple de synthèse

```
77   N  USAGE COMP.
77   TOTAL-TTC  PIC 9(5).99 BLANK ZERO.
01   DONNEE-1.
    05  CODE-ART-1PIC X(5)    VALUE SPACE.
    05  PRIX-UNIT-1     PIC 9(4)V99 COMP-6.
    05  QUANTITE-1      PIC S9(3)   COMP-3 VALUE +0.
    05  QUANTITE-2REDEFINES   QUANTITE-1 PIC S9V99   COMP-3.
    05  COEF-1          PIC S99V99  SIGN TRAILING  SEPARATE.
```

4. La division PROCEDURE

Elle contient la description des traitements à effectuer sur les informations spécifiées dans la division des données.

4.1. Expressions arithmétiques

Une expression arithmétique est une combinaison syntaxiquement correcte de symboles pris dans l'ensemble suivant :

- les noms de données identifiant des rubriques élémentaires de type numérique
- les constantes de type numérique
- les opérateurs arithmétiques
- les parenthèses

Les opérateurs arithmétiques sont, dans l'ordre décroissant des priorités :

- les opérateurs unaires + et -
- l'exponentiation **
- la multiplication * et la division /
- l'addition + et la soustraction -

Les principales règles de syntaxe à respecter sont au nombre de deux :

1. deux opérandes (variable ou constante) sont toujours reliés par un opérateur arithmétique ; toutes les opérations doivent être spécifiées explicitement.

2. deux opérateurs arithmétiques ne se suivent jamais sauf si le second est un opérateur unaire.
   Exemples : `(A + B  * C ) / 3.5`
   `- A / - (B + 1)`

Remarques :

1) Le même symbole / sert à spécifier la division réelle et la division entière ; le choix est déterminé par le type des objets manipulés : / indique une division entière si et seulement si tous les opérandes (y compris celui qui sera affecté de la valeur finale) sont de type entier.

2) Pour évaluer la valeur d'une expression arithmétique, le processeur conserve les valeurs intermédiaires avec une précision maximale ; seul le résultat final sera, tronqué ou arrondi en fonction du format externe de la rubrique affectée.

3) Les opérandes n'ont pas besoin d'être du même format interne ; les conversions de format sont faites automatiquement.

4.2. Valeurs arrondies

Le résultat d'un calcul peut comporter davantage de chiffres que de positions prévues au niveau de la rubrique affectée.

Rappelons que l'affectation d'une valeur numérique se fait conformément à la règle d'alignement numérique énoncée au chap. 2 paragraphe 3 ; s'il y a débordement à gauche la valeur sera tronquée ; s'il y a débordement à droite, la valeur sera tronquée ou arrondie selon que le programmeur aura précisé ou non la locution `ROUNDED`. Cette locution est à spécifier pour chaque instruction dont le résultat doit être arrondi.

L'arrondi est déterminé de la manière suivante :

- si le premier chiffre tronqué vaut de 0 à 4, les chiffres restants ne sont pas modifiés.
- si ce chiffre vaut de 5 à 9, on rajoute une unité au dernier chiffre restant.

Exemple : Soit `A PIC 99V99`

Le résultat 5.1772 donne	sans arrondi	5.17
	avec arrondi	5.18
Le résultat 5.1729 donne	avec ou sans arrondi	5.17
Le résultat 5.195 donne	sans arrondi	5.19
	avec arrondi	5.20

Remarque : Dans l'évaluation d'une expression arithmétique, l'arrondi ne porte que sur le **résultat final**.

4.3. Expressions conditionnelles

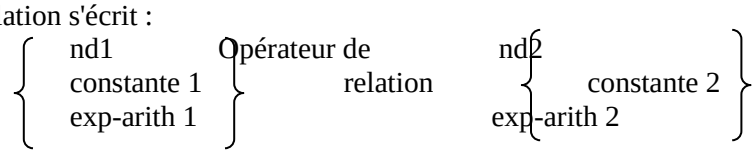
Une expression conditionnelle est une expression dont le résultat est VRAI ou FAUX.

4.3.1. Expressions conditionnelles simples

4.3.1.1. Relation

Une relation permet d'exprimer la comparaison de deux objets ; ces objets sont des données constantes ou variables ; l'un des deux doit être obligatoirement une donnée variable.

Une relation s'écrit :



Opérateurs de relations	Signification	Négation
IS [NOT] GREATER THAN IS [NOT] >	strictement supérieur	inférieur ou égal
IS [NOT] LESS THAN IS [NOT] <	strictement inférieur	supérieur ou égal
IS [NOT] EQUAL TO IS [NOT] =	égal	différent

```
IS GREATER THAN OR EQUAL TO
IS >=

IS LESS THAN OR EQUAL TO
IS <=
```

Exemples

```
A  <  10
NOM    =  NOM-REF
NOM    =  "DUPONT"
A - B   =  RESUL + 15
```

On distingue deux algorithmes de comparaison :

1) Comparaison numérique :

Elle compare les valeurs algébriques de deux données de type numérique ; la longueur de ces données n'est pas significative. Une donnée non signée est considérée comme positive.

2) Comparaison non numérique :

Elle opère sur deux données de type non numérique ; par extension, l'une des deux peut être numérique à condition qu'elle soit entière et déclarée en mode DISPLAY. L'algorithme compare les deux données caractère par caractère, de gauche à droite, jusqu'à trouver une différence ou la fin des rubriques. Le résultat de la comparaison est déterminé comme suit :

- en cas de différence, c'est la donnée contenant le caractère de plus fort poids (cf. table du code ASCII) qui est considérée comme étant la plus grande

- sinon, les deux données ont des valeurs égales.

Si les deux données sont de longueurs différentes, la plus courte est considérée comme étant complétée à droite d'espaces jusqu'à concurrence de la plus longue.

Exemples

1) Soit
```
A     PIC    X(5).
B     PIC    X(5).
```

Valeur de A	Valeur de B	Résultat
TOTO␣	TOTO␣	Egalité
TOTO␣	TOT␣␣	A > B
TOTO␣	TO␣v␣	A > B

2) Soit
```
A     PIC    X(5).
C     PIC    X(3).
```

Valeur de A	Valeur de C	Résultat
TOT␣␣	TOT	Egalité
TOTO␣	TOT	A > C
TOTO␣	XY␣	A < C

4.3.1.2. Test de classe

Un test de classe permet d'exprimer le fait que la valeur d'une donnée est de classe numérique ou alphanumérique.

- Un test de classe numérique ne peut s'appliquer qu'à une rubrique élémentaire déclarée en USAGE DISPLAY.
- Un test de classe alphabétique ne peut pas s'appliquer à une rubrique de type numérique.

Il s'écrit :

```
nd  IS [NOT]  { NUMERIC    }
              { ALPHABETIC }
```

Remarque :

Pour effectuer un test de classe numérique, le processeur tient compte de la déclaration du signe au niveau du format de la rubrique ; ainsi une valeur non signée dans une rubrique signée donne un résultat faux ; il en est de même pour une valeur signée dans une rubrique non signée.

Exemples : Soit `A  PIC  X(5).`

- Le test `A ALPHABETIC` donne un résultat vrai si et seulement si les 5caractères sont des lettres ou des espaces.

- Le test `A NUMERIC` donne un résultat vrai si et seulement si les 5 caractères sont des chiffres.

4.3.1.3. Condition de signe

Elle détermine si la valeur algébrique d'une expression arithmétique est inférieure, supérieure ou égale à zéro.

Expression arithmétique `IS [NOT]` { NEGATIVE | POSITIVE | ZERO }

4.3.1.4. Condition de nom-condition

Elle s'exprime par le simple énoncé du nom-condition :

nom-condition

Elle est vraie si la valeur courante de la variable conditionnelle associée appartient (ou est égale) à celle attribuée au nom-condition.

4.3.2. Expressions conditionnelles complexes

Une condition simple `NEGATIVE` est une condition simple précédée de l'opérateur logique NOT. Elle a pour effet d'attribuer la valeur booléenne opposée à celle de la condition simple.

Ex. : `IF NOT A = B ....`

Une condition COMPOSEE résulte de la conjonction de conditions par les opérateurs AND ou OR :

condition { AND | OR } condition

où condition peut être :

- une condition simple
- une condition simple négative
- une condition composée
- une condition composée négative (i.e. une condition composée entre parenthèses et précédée de NOT)
- toute combinaison de ce qui précède

Ex : `A > B AND NOT (A = C) OR A NOT<D AND B = C`

Il est possible d'agréger une condition composée dans le cas où des conditions simples successives contiennent un même sujet et/ou un même opérateur logique, par l'omission du sujet ET de l'opérateur.

Ex. : `A > B AND NOT < C OR D` signifie

```
((A > B) AND (A NOT < C)) OR (A NOT < D)
```

4.4. Organisation de la division PROCEDURE

La structure de cette division est de l'une des deux formes ci-dessous :

a) `PROCEDURE DIVISION.`

nom de section `SECTION.`

nom de paragraphe. phrase ... ...

b) `PROCEDURE DIVISION.`

nom de paragraphe. phrase ... ...

Ceci signifie que :

- le découpage de la division en paragraphes ou en sections et paragraphes est **obligatoire**.
- le découpage doit être **complet** :
  - toute phrase appartient à un paragraphe explicitement déclaré.
  - si la forme (a) est choisie, tout paragraphe appartient à une section explicitement déclarée.

Exemple

```
PROCEDURE DIVISION.
S0 SECTION.
P1.
P2.
S1 SECTION.
P3
END PROGRAM.
```

4.5. ACCEPT

Permet d'initialiser une rubrique à l'aide d'une valeur particulière (date ou heure) ou d'une valeur lue dans le fichier d'entrée de l'utilisateur.

Format : `ACCEPT` nd [`FROM` { `DATE` | `DAY` | `TIME` | `DAY-OF-WEEK` }]

nd est une rubrique en mode DISPLAY d'au plus 71 caractères ; seules les positions correspondant à des caractères effectifs sont initialisées ; les autres restent inchangées ; il est donc vivement conseillé d'initialiser correctement la rubrique nd avant l'exécution de l'ordre ACCEPT.

De nombreuses variantes existent, suivant les implantations, permettant une gestion avancée de messages à l'écran.

Exemple : `01 IDENTITE.`

```
02  NOM PIC X(10) VALUE SPACE.
02  PRENOM PIC X(10) VALUE SPACE.
.
.
.
ACCEPT IDENTITE
```

Si la valeur lue est DUPONT~~bbbb~~JULES, la rubrique IDENTITE contient après exécution de l'ordre

DUPONT~~bbbb~~JULES~~bbbbb~~b
espaces provenant de la clause VALUE

Les options `DATE` et `TIME` permettent de lire respectivement la date et l'heure enregistrées au niveau du système de l'ordinateur.

- Pour l'option `DATE`, nd doit être une rubrique élémentaire de 6 chiffres, non signée, `PIC 9(6) DISPLAY` ; la valeur obtenue est du format
  AAMMJJ
  Année Mois Jour

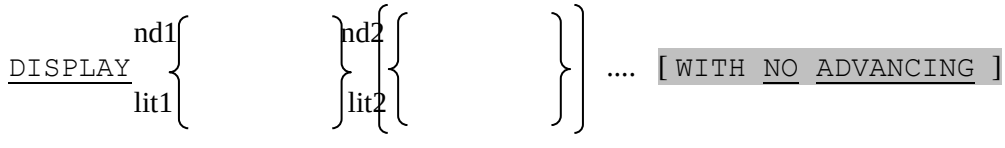
- `DAY` fournit la date au format AAJJJ au format PIC 9(5)

- Pour l'option `TIME`, nd doit être une rubrique élémentaire de 8 chiffres, non signée
  `PIC 9(8) DISPLAY` ; la valeur obtenue est du format hhmmsscc
  heure minutes secondes 1/100è seconde

- `DAY-OF-WEEK` est du format `PIC 9` et renvoie le rang du jour (1=lundi, 7=dimanche)

4.6. DISPLAY

Permet d'envoyer sur un fichier de sortie un faible volume d'informations construit par juxtaposition des valeurs des objets cités :



L'impression n'incluant aucune conversion de format, les valeurs numériques doivent être codées en ASCII.

Ex. : `DISPLAY "NOM : " NM " PRENOM : " PRNM.`

Même remarque que pour `ACCEPT`

4.7. Verbes arithmétiques

4.7.1. Généralités

- Dans toutes les instructions de ce paragraphe, les opérandes identifiés par nd désignent des rubriques numériques ; cependant, les zones de réception des calculs spécifiées après GIVING, n'intervenant pas dans les calculs, peuvent être de type numérique édité.
- Tous les littéraux "lit" sont de type numérique
- La spécification d'un arrondi se fait par la locution ROUNDED.
- La phrase ON SIZE ERROR permet de spécifier un traitement à accomplir en cas de dépassement de capacité lors des calculs ou de vision par 0. Les rubriques de réception associées à cette phrase ne seront pas modifiées en cas d'erreur. La phrase NOT ON SIZE ERROR en est le pendant.
- La locution CORRESPONDING (ou CORR) peut être utilisée dans le cas où les opérandes nd1 et nd2 désignent des rubriques de GROUPES. Une rubrique de nd1 et une rubrique de nd2 correspondent si :

  elles sont désignées par le même nom (autre que FILLER) et ont les mêmes qualificateurs jusqu'à nd1 et nd2 non inclus
  ce sont des données élémentaires
  nd1 et nd2 ne doivent pas contenir de niveau 66, 77, 88
  une donnée subordonnée à nd1 et nd2 et contenant une clause REDEFINES, RENAMES, OCCURS ou INDEX est ignorée de même que les données qui lui sont subordonnées.

Sous ces conditions, l'opération portera individuellement sur chaque couple de données correspondantes.

4.7.2. Addition

Format 1 :
```
ADD { nd1  nd2 } TO nd-m [ROUNDED] [nd-n [ROUNDED]]…
    { lit1 lit2}
        [ON SIZE ERROR ph-impér.]
        [NOT ON SIZE ERROR ph-impér.]
    [END-ADD]
```

Format 2 :
```
ADD { nd1  } … TO lit2 { nd2  } GIVING nd-m [ROUNDED] …
    { lit1 }           {      }
        [ON SIZE ERROR ph-impér.]
        [NOT ON SIZE ERROR ph-impér.]

    [END-ADD]
```

Format 3 :
```
ADD { CORRESPONDING } nd1 TO nd2 [ROUNDED]
    { CORR          }
        [ON SIZE ERROR ph-impér.]
        [NOT ON SIZE ERROR ph-impér.]
    [END-ADD]
```

Exemple : `ADD N M 3 TO P` P=N + M + 3 + P
`ADD N M 3 GIVING P` P=N + M + 3

4.7.3. Soustraction

Format 1 :

```
SUBTRACT { nd1  } ... FROM {nd-m [ROUNDED]} …
         { lit1 }
      [ON SIZE ERROR ph-impér.]
      [NOT ON SIZE ERROR ph-impér.]
   [END-SUBTRACT]
```

Format 2 :

```
SUBTRACT { nd1  } ... FROM { nd2  }
         { lit1 }          { lit-m }
   GIVING nd-n [ROUNDED]

      [ON SIZE ERROR ph-impér.]
      [NOT ON SIZE ERROR ph-impér.]
   [END-SUBTRACT]
```

Format 3 :

```
SUBTRACT {CORRESPONDING} nd-1 FROM nd-2 [ROUNDED]
         {CORR         }

      [ON SIZE ERROR ph-impér.]
      [NOT ON SIZE ERROR ph-impér.]
   [END-SUBTRACT]
```

4.7.4. Multiplication

Format 1 :

```
MULTIPLY { nd1  } BY {nd-2 [ROUNDED]} ...
         { lit1 }

      [ON SIZE ERROR ph-impér.]
      [NOT ON SIZE ERROR ph-impér.]
   [END-MULTIPLY]
```

Format 2 :

```
MULTIPLY { nd1  } BY { nd2  } GIVING {nd-3 [ROUNDED]} …
         { lit1 }    { lit2 }

      [ON SIZE ERROR ph-impér.]
      [NOT ON SIZE ERROR ph-impér.]
   [END-MULTIPLY]
```

4.7.5. Division

Format 1 :

```
DIVIDE { nd1  } INTO {nd-2 [ROUNDED] } ...
       { lit1 }

       [ON SIZE ERROR ph-impér.]
       [NOT ON SIZE ERROR ph-impér.]
[END-DIVIDE]
```

Format 2 :

```
DIVIDE { nd1  } INTO { nd2  } GIVING {nd-3 [ROUNDED]} …
       { lit1 }      { lit2 }

       [ON SIZE ERROR ph-impér.]
       [NOT ON SIZE ERROR ph-impér.]
[END-DIVIDE]
```

Format 3 :

```
DIVIDE { nd1  } BY { nd2  } GIVING {nd-3 [ROUNDED]} …
       { lit1 }    { lit2 }

       [ON SIZE ERROR ph-impér.]
       [NOT ON SIZE ERROR ph-impér.]
[END-DIVIDE]
```

Format 4 :

```
DIVIDE { nd1  } INTO { nd2  } GIVING nd-3 [ROUNDED]
       { lit1 }      { lit2 }
       REMAINDER nd-4

       [ON SIZE ERROR ph-impér.]
       [NOT ON SIZE ERROR ph-impér.]
[END-DIVIDE]
```

Format 5 :

```
DIVIDE { nd1  } BY { nd2  } GIVING nd-3 [ROUNDED]
       { lit1 }    { lit2 }
       REMAINDER nd-4

       [ON SIZE ERROR ph-impér.]
       [NOT ON SIZE ERROR ph-impér.]
[END-DIVIDE]
```

Remarques :

1. L'option `BY` implique `GIVING`
2. L'option `REMAINDER` permet de recueillir le reste de la division dans nd-4 ; elle interdit l'usage de plusieurs zones de réception.

4.7.6. Calcul

La valeur d'une expression arithmétique complexe peut être obtenue par l'instruction :

```
COMPUTE {nd-1 [ROUNDED]} ... = expression-arithmétique
                [ON SIZE ERROR ph-impér.]
                [NOT ON SIZE ERROR ph-impér.]
          [END-COMPUTE].
```

Ex. : `COMPUTE TOTAL ROUNDED = PUHT * QTE + TRSP * (1 - TX)`

4.8. MOVE

L'instruction MOVE permet de recopier dans une (ou plusieurs)rubrique(s) dites réceptrices, le contenu d'une (ou plusieurs) rubrique(s) dites émettrices.

Format 1 : `MOVE { nd1 | lit1 } TO {nd-2} ...`

Format 2 : `MOVE { CORRESPONDING | CORR } nd1 TO nd2`

- L'opération inclut si nécessaire les conversions de format interne.

- Dans le format 2, au moins une des données du couple de données concerné par l'opération doit être élémentaire.

Ex. :

```
        05 DT-AMER.
              10 AA PIC 99.
              10 MM PIC 99.
              10 JJ PIC 99.

        05 DT-JOUR.
              10 JJ PIC 99.
              10 FILLER PIC X VALUE "/".
              10 MM PIC 99.
              10 FILLER PIC X VALUE "/".
              10 AA PIC 99.

   MOVE CORR DT-AMER TO DT-JOUR.
```

- Il y a deux types de transferts :
  - transfert **numérique**
  - transfert **non numérique**

  qui respectent les règles d'alignement décrites au chapitre 2 § 3 ; il ne peut y avoir de conversion de format interne dans un transfert non numérique.

- Le type du transfert est déterminé par le type de l'émetteur et du récepteur ; le tableau ci-dessous résume les transferts possibles

	Zone de Réception					
Zone d'Emission	Num. entier	Num.fract.	Alphabét.	Alphanum.	Alph. Edité	Num. Edité
Num. entier	N	N	I	X	X	N
Num. fractionnaire	N	N	I	I	I	N
Alphabétique	I	I	X	X	X	I
Alphanumérique	X*	X*	X*	X	X	X*
Alphanum. Edité	I	I	X	X	X	I
Num. Edité	I	I	I	I	X	X

X : transfert non numérique
X* : transfertautorisé seulement si tous les caractères émis sont autorisés dans la zone de réception
N : transfert numérique
I : transfert interdit

Une rubrique structurée se comporte comme une rubrique élémentaire alphanumérique.

Transfert du signe :

- Emetteur signé/récepteur non signé : le récepteur contient la valeur absolue de l'émetteur
- Emetteur non signé/récepteur signé : la valeur de l'émetteur est considérée comme positive
- Emetteur numérique entier/récepteur alphanumérique : le signe n'est pas transféré.

4.9. IF

Elle permet d'exprimer une alternative entre deux traitements.

Format : `IF` expression conditionnelle `THEN` { phrase-1 | `NEXT SENTENCE` }

{ `ELSE` | `OTHERWISE` } { phrase-2 | `NEXT SENTENCE` }

[`END-IF`]

Exemple
```
IF A NOT NUMERIC OR A NOT > 0
                    DISPLAY "ERREUR***"
            ELSE ADD A TO SOMME.
```

- Si l'expression conditionnelle donne un résultat vrai, on effectue le traitement décrit dans "phrase-1" sinon on effectue celui qui est décrit dans "phrase-2".

Remarques :

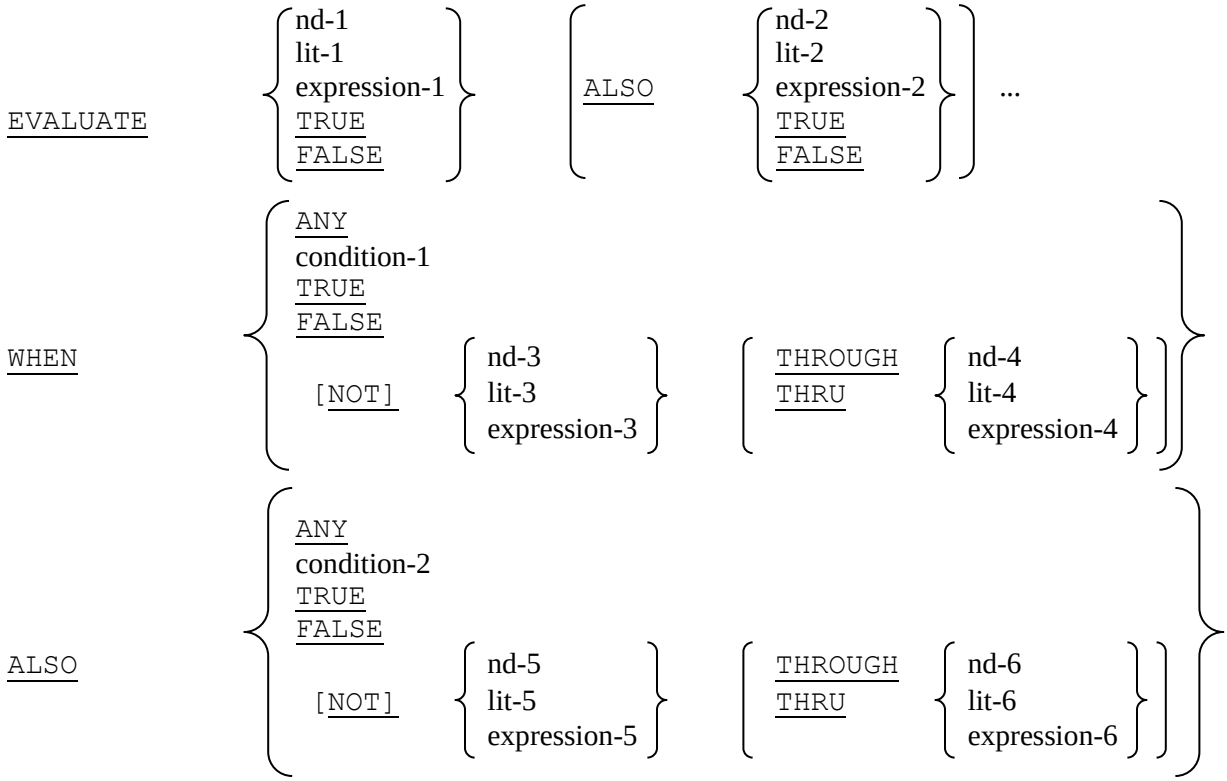
- ne pas oublier le point à la fin de "phrase-2" ou `END-IF`. Le mot clé `ELSE` indique la fin de "phrase-1".
- phrase-1 et/ou phrase-2 se composent d'instructions impératives et/ou d'une instruction IF.
- IF et ELSE s'emploient par paires ; tout mot ELSE est à associer avec la première condition qui le précède et qui n'est pas déjà reliée à un mot ELSE.

Exemple :

```
IF ....
    IF ....
    ELSE ....
        ELSE   IF ....
ELSE .... .
```

4.10. EVALUATE

En COBOL 85, l'instruction EALUATE permet de généraliser la notion de structure alternative, et représente une implémentation de la liste de cas.



phrase-impérative-1 …

[WHEN OTHER phrase-impérative-2]

[END-EVALUATE]

Les opérandes ou les mots TRUE ou FALSE précédant le premier WHEN sont appelés les **sujets** de la sélection, alors que les opérandes suivant WHEN sont appelés les objets de la sélection. Le nombre de sujets doit être égal au nombre d'objets, la **correspondance s'opérant par rang** (position relative).

L'instruction s'exécute en évaluant les conditions, expressions ou valeurs logiques des sujets et des objets. Chaque objet de la sélection suivant le premier WHEN est comparé au sujet de même rang. Si la comparaison est satisfaite pour chacun des couples sujet-objet (ANY satisfaisant toute condition), alors l'instruction impérative qui suit ce WHEN est exécutée et l'on quitte EVALUATE.

Le processus est répété en cas d'inégalité pour trouver le premier WHEN satifaisant l'ensemble des conditions. Si aucune phrase suivant WHEN n'a été sélectionnée, et si WHEN OTHER a été spécifié, la

phrase-impérative-2 est exécutée, et l'instruction EVALUATE est terminée.
Il est possible de préciser CONTINUE en tant que phrase-impérative-1 : aucune action n'est effectuée et l'on passe à l'instruction suivant EVALUATE.

Exemple 1
```
EVALUATE A
        WHEN   1   PERFORM TRAITEMENT-1
        WHEN   2   PERFORM TRAITEMENT-2
        WHEN   5   PERFORM CONTINUE
        WHEN OTHER PERFORM TRAITEMENT-3
END-EVALUATE.
```

Le paragraphe TRAITEMENT-1 est exécuté si A a la valeur 1, TRAITEMENT-2 si A a la valeur 2, et TRAITEMENT-3 si A a toute valeur autre que l, 2 ou 5.

Exemple 2
```
EVALUATE A ALSO B ALSO C
     WHEN 1 ALSO 5 ALSO NOT 7    PERFORM TRAITEMENT-1
     WHEN 2 ALSO 4 THRU 7 ALSO 3 PERFORM TRAITEMENT-2
     WHEN 5 ALSO ANY ALSO ANY    PERFORM TRAITEMENT-2
        WHEN OTHER               PERFORM TRAITEMENT-3
END-EVALUATE .
```

Les instructions de TRAITEMENT-1 sont exécutées pour A=1, B=5 et C différent de 7, celles de TRAITEMENT-2 pour A=2, B compris entre 4 et 7, et C=3 ainsi que pour A=5 quelles que soient les valeurs de B et C, et enfn celles de TRAITEMENT- 3 sont exécutées dans tous les autres cas.

Exemple 3
```
EVALUATE A = 1 ALSO B = 1
  WHEN TRUE  ALSO TRUE      PERFORM TRAITEMENT-1
  WHEN TRUE  ALSO FALSE     PERFORM TRAITEMENT-2
  WHEN FALSE ALSO TRUE      PERFORM TRAITEMENT-2
  WHEN OTHER                PERFORM TRAITEMENT-3
END-EVALUATE.
```

A	B
1	1
1	0
0	1
0	0

Table de vérité XOR.

Exemple 4
```
EVALUATE TRUE ALSO TRUE
     WHEN A = 1 ALSO B = 1 PERFORM TRAITEMENT-1
     WHEN MY ALSO B = 2    PERFORM TRAITEMENT-2
     WHEN OTHER            PERFORM TRAITEMENT-3
END-EVALUATE.
```

Les instructions de TRAITEMENT-1 sont exécutées pour A=1 et B=1, celles de TRAITEMENT-2 si B=2 et quel que soit A, et celles de TRAITEMENT-3 dans les autres cas.

4.11. PERFORM

Elle traduit l'action de "faire" un traitement 0, 1 ou plusieurs fois

Format 1 : PERFORM nt1 { THROUGH | THRU } nt2
[ph-impér. END-PERFORM]

Format 2 : PERFORM nt1 { THROUGH } nt2 { nd } TIMES

```
                                        THRU              entier-1
        [ph-impér. END-PERFORM]
Format 3 : PERFORM  nt1  { THRU     }  nt2 [ WITH TEST { BEFORE } ]
                         { THROUGH  }                  { AFTER  }
        UNTIL condition-1
        [ph-impér. END-PERFORM]
```

- Terminologie :
  - nt désigne un paragraphe ou une section
  - nd doit référencer un entier non signé
- L'instruction PERFORM permet d'exécuter un bloc de traitement décrit sous forme :
  - d'un paragraphe
  - d'une section
  - d'un ensemble de paragraphes et/ou de sections contigus dont le premier s'appelle nt1 et le dernier nt2.
- L'exécution du bloc de traitement commence à la première instruction du bloc et se termine obligatoirement à la dernière instruction du bloc.
- L'instruction PERFORM peut être écrite avant ou après le traitement qu'elle fait exécuter.
- Dans les formats 3 et 4 (cf ci-après), **par défaut** ou si l'on précise TEST BEFORE la condition est évaluée **avant** d'exécuter le bloc ; avec TEST AFTER, après exécution du bloc.

Exemple

```
S0  SECTION.
P0.
P1.

S1 SECTION.
P2.
P3.

S2 SECTION.


P4. PERFORM S0                  (1)
   PERFORM P0 THRU P1           (2)
   PERFORM P1 THRU P2           (3)
   PERFORM S0 THRU S1           (4)
   PERFORM P0 THRU P3           (5)
```

Les lignes (1) et (2) sont équivalentes ; il en est de même des lignes (4) et (5).

- Le format 2 permet de traduire un traitement itératif, le nombre de répétitions étant connu par avance.

Exemples :
```
PERFORM S0   3   TIMES
PERFORM P1  N1   TIMES
```
Où `N1 PIC 99.`

A noter que seule la valeur de `N1` définie **au moment** de l'exécution de l'ordre `PERFORM` est prise en compte ; toute modification ultérieure de `N1`, au cours des itérations, est sans effet sur le nombre de répétitions effectuées.

- Le format 3 permet de traduire un traitement itératif à répéter jusqu'à ce qu'une condition soit satisfaite.

Exemple : `PERFORM SI UNTIL J > 50`

Le cycle des opérations standard est le suivant :

1) Evaluation de l'expression conditionnelle (sauf si `TEST AFTER`)
2) Si le résultat de l'expression conditionnelle est vrai
Arrêt des itérations sinon exécuter S1 reprendre au point 1)

Remarque:

Si au départ J contient une valeur supérieure à 50, il n'y a pas d'exécution de S1.

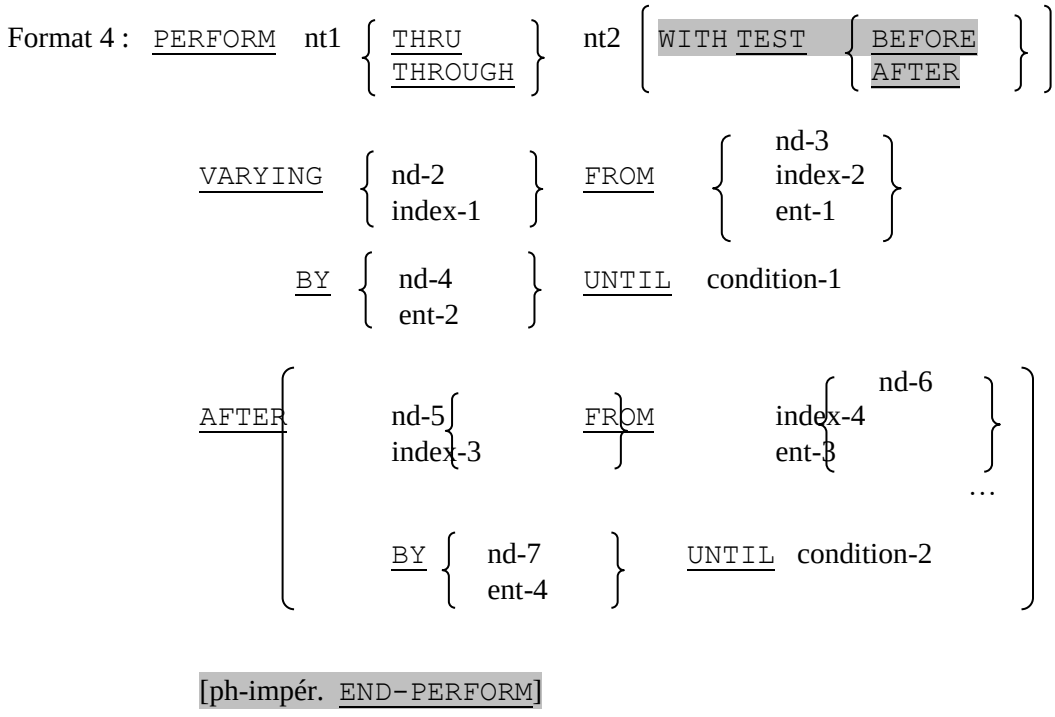
- Instructions PERFORM imbriquées :

Le bloc de traitement exécuté à partir d'un ordre PERFORM peut contenir des ordres PERFORM à condition que ceux-ci réfèrent des blocs de traitement **totalement inclus ou disjoints** du bloc d'origine.

Exemple :
```
PERFORM  P0 THRU P4

        P0.

        P1.

        P2.

        P3.    PERFORM P1 THRU P2 2 TIMES        (1)

        P4.    PERFORM P5 UNTIL NOM = SPACE

    P5.
```

Le bloc P1-P2 est inclus dans le bloc P0-24
Le bloc P5 est disjoint du bloc P0-P4.
Il est interdit de remplacer par exemple la ligne (1) par PERFORM P4 THRU P5

Instruction PERFORM avec variable de comptage



Jusqu'à 6 phrases AFTER

Il inclut la gestion des indices ou des index jusqu'à 3 (7) niveaux d'imbrication.

Exemple : Soit la procédure CUMUL-NOTE permettant de cumuler une note (appartenant à une table de dimension 2) à une variable de cumul partiel (SOMME). E et N sont deux indices indiquant respectivement le rang de l'élève et le rang de la note.

```
       MOVE 0 TO SOMME
       PERFORM CUMUL-NOTE VARYING E FROM 1 BY 1
                          UNTIL E > 25
                          AFTER N FROM 1 BY 1
                              UNTIL N > 10.

CUMUL-NOTES.
       ADD NOTE-ELEVE (E,N) TO SOMME.
```

4.12. EXIT

Format 1 : EXIT .

Format 2 : EXIT PROGRAM.

Cette instruction doit être la seule instruction de la seule phrase du paragraphe où elle est écrite.

Son rôle : - EXIT : définir un paragraphe vide
- EXIT PROGRAM : définir un paragraphe vide et déclencher le retour au programme appelant; cette option ne peut être utilisée que dans un sous-programme.

Exemple :

Dans un tableau INDIVIDUS de dimension 1 comportant 25 noms de personnes, déterminer le rang de l'individu DUPONT.

```
        PERFORM RECHERCHE VARYING E FROM 1 BY 1
                          UNTIL NOM (E) = "DUPONT"
                                       OR E > 25.
RECHERCHE. EXIT.
```

A la fin de l'itération, E contient le rang de DUPONT ou la valeur 26 si cette personne ne figure pas dans la table.

4.13. STOP

Permet d'arrêter temporairement ou définitivement l'exécution d'un programme.

Format : STOP { RUN / lit }

- STOP RUN provoque l'arrêt définitif du programme ;
- Dans le format 2, le littéral est affiché sur le terminal de l'opérateur et le programme est suspendu. A ce point, l'utilisateur peut relancer le programme par Enter, ou l'abandonner.

4.14 GO

Permet d'effectuer dans un programme une rupture de séquence inconditionnelle. L'usage de cette instruction doit être réservé exclusivement à la traduction de certaines structures algorithmiques bien précises.

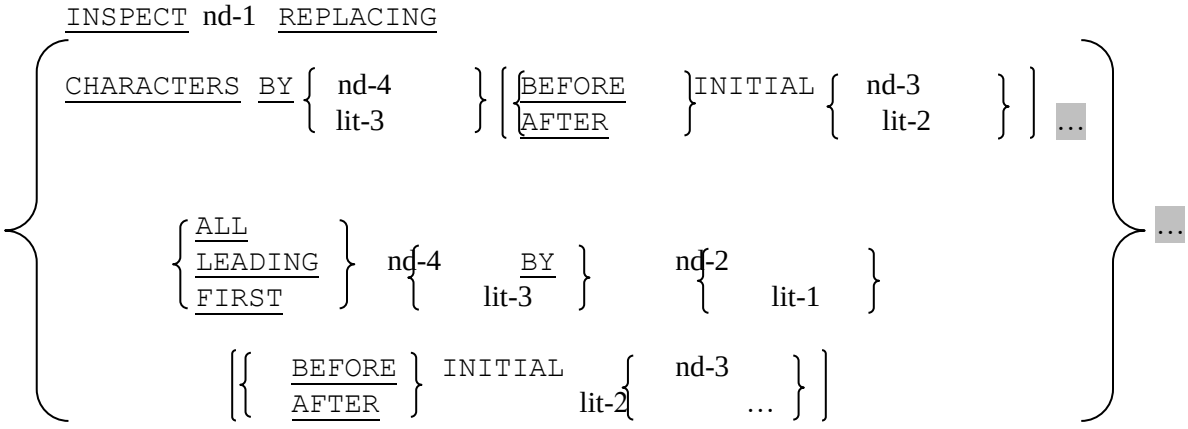
Format 1 : GO TO nt
Format 2 : GO TO nt1 [nt2] ..., ntn DEPENDING ON nd

Le deuxième format (aiguillage) permet de traduire un choix entre n traitements en fonction de la valeur de nd; nd réfère une rubrique élémentaire à valeur entière comprise entre 1 et n ; si nd vaut k, il y a branchement au $k^{ième}$ traitement de la liste ; si k # [1,n], l'instruction est sans effet.

4.15 INSPECT

Permet, à l'intérieur d'une rubrique de type non numérique, de remplacer (Format 1) les occurrences d'un ou plusieurs caractères, ou de les compter (Format2).

Format 1



- nd1 réfère la rubrique étudiée élémentaire ou de groupe , en mode `DISPLAY` obligatoirement.
- nd2 à nd4 doivent être des rubriques élémentaires.
- Algorithme de remplacement : L'algorithme examine le contenu de la rubrique de gauche à droite ; il y a deux possibilités :

a) Option `CHARACTERS` : on remplace chaque caractère rencontré par la valeur précisée nd2 ou lit1. Cette clause ne peut apparaître qu'une fois.

b) Autres options : on remplace chaque sous-chaîne de valeur nd4 ou lit3 par la valeur précisée par nd2 ou lit1 ; la sous-chaîne de remplacement doit avoir la même longueur que la sous-chaîne d'origine. Il est possible de préciser plusieurs options de ce type permettant de procéder au remplacement des occurrences de plusieurs sous-chaînes distinctes.

Exemples
```
INSPECT ZONE REPLACING CHARACTERS BY "*"
```
Tous les caractères de ZONE sont remplacés par des *
```
INSPECT ZONE REPLACING ALL "ABC" BY "XXX"
                       ALL "CDE" BY "YYY"
```
Valeur d'origine de ZONE : B C D E Z A B C D E A B C C D E
Valeur finale de ZONE : B Y Y Y Z X X X D E X X X Y Y Y
Les opérations sont traitées suivant l'ordre d'écriture ; ainsi la 2e occurrence de CDE n'est pas prise en compte, la lettre C ayant déjà été traitée par le remplacement de la 1ère occurrence de ABC.

Les mots clés `ALL`, `LEADING` et `FIRST` permettent de sélectionner ou non certaines occurrences de la sous-chaîne à remplacer :

- `ALL` : toutes les sous-chaînes ayant la valeur nd4 ou lit3 sont traitées

- `LEADING` : **toutes** les occurrences **contiguës** de la sous-chaîne de valeur nd4 ou lit3 sont traitées à condition que la première se trouve au début de la zone à traiter.

Exemple : `INSPECT ZONE REPLACING LEADING "AB" BY "ZZ"`

Valeur d'origine de ZONE : XYABABXAB
Valeur finale : identique
si on rajoute `AFTER "Y"`, valeur finale : XYZZZZXAB

♦ `FIRST` : seule **la première** sous-chaîne de valeur nd4 ou lit3 est traitée.

Délimitation de la zone à étudier :

L'option `AFTER/BEFORE`, valable pour les deux algorithmes de remplacement, permet de faire débuter (respect. d'arrêter) le traitement à la rencontre de la première occurrence de la valeur spécifiée par nd3 ou lit2 (obligatoirement de longueur 1 caractère pour l'algorithme (a)).

Format 2

```
INSPECT   nd-1 TALLYING

nd-2 FOR  { ALL        }   { nd-3 }
          { LEADING    }   { lit-1 }
          { CHARACTERS }

[ { BEFORE } INITIAL { nd-4 }   ]
[ { AFTER  }         { lit-2 }  ]  ...
```

Avec `TALLYING`, nd-2 doit référencer une rubrique numérique, précédemment initialisée. Cette rubrique est incrémentée de un pour chaque occurence de sous-chaîne (ou chaque caractère) suivant des modalités analogues à celles de `REPLACING`.

Format 3

Résulte de la juxtaposition du format 2 (`TALLYING`) suivi du format 1 (`REPLACING`).

Exemple :
```
INSPECT RUB TALLYING COMPTEUR FOR ALL "L" REPLACING
        LEADING  "A" BY "E"  AFTER INITIAL "L".
```

Format 4

Il permet de remplacer, dans nd1, chaque caractère de nd2 par le caractère de même rang de nd3 (équivaut à une série de INSPECT REPLACING pour chaque caractère).

```
INSPECT   nd-1 CONVERTING

{ nd-2 } TO { nd-3 }
{ lit-1 }   { lit-2 }

[ { BEFORE } INITIAL { nd-4 }   ]
[ { AFTER  }         { lit-3 }  ]  ...
```

Exemple :

```
77 ZON PIC X(10)  VALUE "ABCAEFACA".
INSPECT ZON CONVERTING "ABC" TO "XYZ" BEFORE "F".
```

ZON contiendra "XYZXEFACA"

4.16 STRING et UNSTRING

4.16.1. Concaténation

L'instruction STRING permet la juxtaposition des contenus partiels ou complets de 2 ou plusieurs rubriques en une seule.

```
STRING { émetteur-1 ... DELIMITED BY { délimiteur-1 } } ...
       {                             { SIZE        }
       INTO récepteur [WITH POINTER pointeur]
       [ON OVERFLOW phrase-impérative 1]
            [NOT ON OVERFLOW phrase-impérative 2]
            [END-STRING]
```

- Chaque rubrique d'émission doit être d'usage DISPLAY ; si elle est numérique, elle doit être entière. Ce peut être aussi un littéral non numérique (ou une constante figurative, considérée comme un seul caractère).
- La rubrique de réception doit être **alphanumérique** élémentaire.
- La clause DELIMITED est **obligatoire** ; les délimiteurs (1 ou plusieurs caractères) suivent les règles de syntaxe des émetteurs ; un délimiteur n'est PAS TRANSMIS dans la zone de réception. Si SIZE est spécifié, le contenu complet de l'émetteur (ou des émetteurs) précédant DELIMITED est transféré dans le récepteur.
- Le pointeur est une rubrique entière de longueur suffisante pour contenir la taille + 1 du récepteur; il contient à chaque instant le rang du prochain caractère à recevoir ; s'il est spécifié, il doit être initialisé à une valeur supérieure ou égale à 1(défaut = 1).

Aucun des caractères du récepteur, non concernés par le transfert (en début ou en fin de zone) n'est modifié par l'instruction.

- La locution OVERFLOW sera validée si au cours des transferts, le nombre de caractères émis excède la longueur du récepteur (à défaut, le pg se poursuit en séquence).

Remarque : On peut considérer que les transferts se font caractère par caractère, en commençant, pour chaque émetteur pris en séquence, par le caractère le plus à gauche et jusqu'à atteindre :

- soit la fin de l'émetteur,
- soit le(s) caractère(s) du délimiteur associé (jamais transmis).

Exemples :
```
STRING JJ "/" MM "/" AA DELIMITED BY SIZE INTO DATE-COM.
STRING NOM DELIMITED BY "   " SPACE DELIMITED SIZE PRENOM
DELIMITED SPACE INTO IDENTITE.
```

4.16.2. Séparation

L'instruction UNSTRING provoque la séparation de données contigües d'une rubrique émettrice et leur transfert dans des rubriques réceptrices multiples.

```
UNSTRING émetteur   [DELIMITED BY [ALL] délimiteur-1
                [OR [ALL] délimiteur-2] ...]

INTO     {récepteur-1 [DELIMITER IN délimiteur-3] [COUNT IN compteur-1]}
  [WITH POINTER pointeur] [TALLYING IN compteur-3]
   [ON OVERFLOW phrase-impérative 1]
      [NOT ON OVERFLOW phrase-impérative 2]
      [END-UNSTRING]
```

- L'émetteur et les délimiteurs doivent être des rubriques alphanumériques ; les délimiteurs peuvent àtre des littéraux non numériques.
- Les récepteurs peuvent être alphabétiques, alphanumériques ou numériques (d'usage DISPLAY).
- Les compteurs et pointeurs doivent être numériques entiers.
- Les locutions DELIMITER et COUNT ne peuvent figurer que si DELIMITED BY est spécifié.

Règles d'application

- délimiteur-3 et -4 sont les rubriques de réception des délimiteurs, et accueillent ces derniers selon les règles de transfert non-numérique. Si la condition de délimitation est la fin de l'émetteur (i.e. délimiteur non rencontré) ces rubriques sont à blanc.
- Chaque compteur contient le nombre de caractères transmis dans le récepteur associé, délimiteur exclu.
- Le pointeur indique le rang du premier caractère de l'émetteur participant à l'opération (défaut = 1), puis est incrémenté au fur et à mesure.
- Le compteur-3 enregistre le nombre de rubriques de réception mises en jeu par l'opération (initialisation à charge du programmeur).
- Lorsque ALL est spécifié, plusieurs apparitions contigües du délimiteur associé sont traitées comme une seule occurrence ; sinon, deux apparitions contigües entraînent une mise à blanc ou à

zéro du récepteur associé, selon sa définition.

- Lorsqu'un délimiteur contient plusieurs caractères, ils doivent être présents en totalité, et dans l'ordre, dans l'émetteur pour être reconnus comme délimiteur.

- Si plusieurs délimiteurs sont spécifiés dans la locution DELIMITED BY, reliés par OR, l'émetteur est analysé pour chacun d'eux dans l'ordre donné, jusqu'à correspondance.

- Si DELIMITED BY n'est pas spécifié, le transfert s'effectue jusqu'a remplir le récepteur courant.

- Après chaque extraction d'une sous-chaîne de l'émetteur, l'analyse reprend au caractère suivant le délimiteur.

- L'analyse se poursuit jusqu'à la fin de l'émetteur ou épuisement de récepteurs.

- A la fin de l'opération, le pointeur est égal à sa valeur initiale plus le nombre de caractères examinés dans l'émetteur.

La locution ON OVERFLOW sera validée :

- si le pointeur est initialisé à une valeur inférieure à 1 ou supérieure à la taille de l'émetteur
- si toutes les zones de réception ont été utilisées et il reste des caractères à examiner dans l'émetteur.

- Si une condition de débordement apparaît et OVERFLOW n'est pas spécifié, l'exécution se poursuit en séquence.

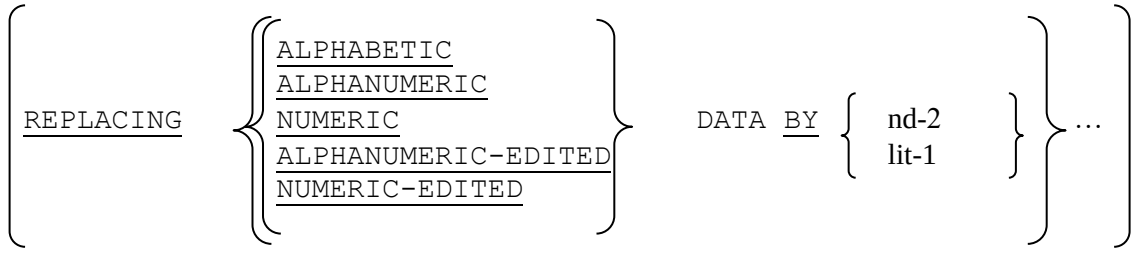
- Si un délimiteur ou un récepteur est indicé ou indexé, l'indice ou l'index est évalué immédiatement avant le transfert de donnée dans la rubrique correspondante.

Exemples :
```
UNSTRING DATE-COM DELIMITED BY "/" INTO JJ MM AA

UNSTRING IDENTITE DELIMITED BY SPACE INTO NOM
COUNT IN CN PRENOM COUNT IN CP POINTER CT
```

4.17. INITIALIZE

En ANS85, cette instruction permet d'initialiser, certains **types** de données.

INITIALIZE { nd-1} …



- nd-1 est le récepteur, à initialiser ; nd-2 ou lit-1 sont les émetteurs servant à l'initialisation.
- nd-1 ne peut être un index, ni contenir de clause RENAMES ou DEPENDING ON
- en l'absence de clause REPLACING, les données élémentaires des récepteurs sont initialisée à **b** pour les types non-numériques et à **0** pour les types numériques.
- Si REPLACING est spécifié, l'instruction opère comme un ensemble de MOVE entre l'émetteur et le(s) récepteur(s), au niveau élémentaire s'il s'agit de groupes. L'opération n'est effectuée que pour les données correspondant au type cité. Un type ne peut apparaître qu'une seule fois dans un INITIALIZE.

Exemple
```
01  COMPTE.
    02  NUM     PIC 9(5).
    02  LIBEL   PIC X(20).
    02  SOLDE   PIC S9(6)V99  PACKED-DECIMAL.
    …

INITIALIZE COMPTE.

INITIALIZE COMPTE  REPLACING ALPHANUMERIC BY ALL "X".
```