INSTITUT de la gestion publique et du développement économique

Support stagiaire

COBOL
MODULE 1
Bases du langage

MINISTÈRE DE L'ÉCONOMIE
DES FINANCES ET DE L'INDUSTRIE

Sommaire

FICHE 01

Structure d'un programme Cobol

1. DIVISIONS ET SECTIONS.

Un programme COBOL est composé de **divisions** au nombre maximum de quatre. Seule la première (IDENTIFICATION DIVISION) est obligatoire. Les divisions sont présentées dans l'ordre où elles doivent être écrites.

Ces divisions sont elles-mêmes découpées en **sections** dont les noms sont imposés pour les trois premières divisions. Aucune section n'est obligatoire.

1.1. IDENTIFICATION DIVISION.

Comme son nom l'indique, elle est utilisée pour identifier le programme et, le cas échéant, pour lui affecter certains attributs. On y mentionnera obligatoirement le nom du programme par le titre PROGRAM-ID suivi d'un espace au moins.

Exemple :

- IDENTIFICATION DIVISION.
- PROGRAM-ID. monprog. (maximum 8 caractères)

Les autres clauses, facultatives en COBOL 74 sont devenues obsolètes en COBOL 85.

NOTES

ENVIRONMENT DIVISION.

Elle décrit l'environnement machine et peut contenir deux sections :

1.1.1. LA CONFIGURATION SECTION.

Ne contient que le seul paragraphe SPECIAL-NAMES, qui permet de modifier certaines options par défaut au moyen des clauses suivantes :

- CURRENCY SIGN permet de remplacer le signe monétaire $ par le signe de son choix.
- DECIMAL-POINT IS COMMA indique que la virgule devient marque décimale à la place du point.
- nom_de_système IS nom_mnémonique permet d'affecter un nom mnémonique à un mot système d'édition, par exemple au caractère de saut de page.

Exemple :

- CONFIGURATION SECTION.
- SPECIAL-NAMES.
- DECIMAL-POINT IS COMMA
- CURRENCY SIGN IS F *pas de point entre les clauses*
- C01 IS SAUT-PAGE. un point pour clore le paragraphe

NOTES

1.1.2. L'INPUT-OUTPUT SECTION.

Peut contenir deux paragraphes optionnels :

- FILE-CONTROL qui contient de 1 à n clauses SELECT.

Chaque clause SELECT associe le nom COBOL d'un fichier à son nom externe utilisé dans le JCL. Cette clause permet également de déclarer un fichier optionnel et de préciser son organisation, son mode d'accès et son file status.

- I-O-CONTROL :
  - ce paragraphe peut permettre de spécifier des points de reprise et de déclarer des buffers d'E/S communs à plusieurs fichiers. L'I-O-CONTROL est très peu utilisé.

Exemple

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
        SELECT FIC1 ASSIGN TO FIC1
        ORGANIZATION IS INDEXED
        ACCESS MODE IS DYNAMIC
        RECORD KEY IS CLE1
        FILE STATUS IS FIC1-FS.
        SELECT OPTIONAL TOTO ASSIGN TO FIC2.
I-O-CONTROL.
        SAME RECORD AREA FOR FIC1 TOTO.
```

NOTES

1.2. DATA DIVISION.

La DATA DIVISION contient la description de toutes les données, externes et internes, qui sont utilisées par le programme.

Elle est divisée en 3 sections optionnelles :

- la FILE SECTION où sont décrits les fichiers utilisés et leurs caractéristiques, y compris les fichiers de tris. Ce sont les données externes.
- la WORKING-STORAGE SECTION qui contient les données internes.
- la LINKAGE SECTION où sont décrites les zones utilisées par plusieurs programmes.

Pourquoi distingue-t-on données internes et externes ?

Une fois chargé en mémoire, un programme est composé de deux parties qui lui sont propres : les **données internes** et les instructions.

Les données externes résident en mémoire à l'extérieur du programme. Ce sont essentiellement les données fichier.

La description d'un fichier qui est faite en FILE SECTION est le découpage logique d'un enregistrement. On appelle ces zones des **zones de communication**.

NOTES

1.2.1. LA FILE SECTION.

La description d'un fichier se fait dans un paragraphe suivant le format :

- FD nom fichier (File Description) pour un fichier quelconque
- SD nom fichier (Sort Description) pour un fichier de tri. (les tris font l'objet d'un autre module de stage).

Dans ce paragraphe, on peut utiliser les clauses de description suivantes :

- BLOCK CONTAINS (FD uniquement) :
  - ◊ spécifie la taille des enregistrements physiques en caractères ou en nombre d'enregistrements logiques. Cette clause est facultative pour des enregistrements bloqués, inutile pour des fichiers sans blocage ou indexés.

- RECORD CONTAINS :
  - ◊ indique la taille d'un enregistrement. Cette clause est facultative, mais lorsqu'elle est spécifiée, le compilateur vérifie l'égalité entre la taille indiquée et la taille de la zone de niveau 01 qui suit.

Pour les fichiers de longueur variable, cette clause sera codée ainsi :

- RECORD VARYING FROM nb-minimum TO nb-maximum
  DEPENDING ON nom_de_donnée

Nota :

- la clause LABEL RECORD est obsolète en COBOL 85.
- la clause RECORDING MODE est également devenue obsolète. Cette clause précisait le format des enregistrements :
  - ◊ F Fixed
  - ◊ V Variable
  - ◊ U Indéfini
  - ◊ S Fixe ou variable

NOTES

1.2.2. LA WORKING-STORAGE SECTION.

Elle contient la description de toutes les données internes. Cette partie est étudiée dans le chapitre représentation des données (Cf. chapitre 2).

1.2.3. LA LINKAGE SECTION.

Déclarée dans un sous-programme, elle contient la description des données que l'on va recevoir du programme appelant. Elle peut contenir les mêmes types de description de données que la WORKING-STORAGE SECTION, à une exception près : il ne peut y avoir d'initialisation des données par "VALUE".

1.3. LA PROCEDURE DIVISION.

Cette division contient toutes les séquences d'instructions nécessaires au déroulement du programme. Ces instructions peuvent être hiérarchisées en :

sections
paragraphes
phrases
instructions

1.3.1. LA SECTION.

Elle est rarement utilisée en dehors des déclaratives et des tris, ceux-ci étant étudiés dans les modules suivants.

NOTES

1.3.2. LE PARAGRAPHE.

Il se compose d'un titre suivi d'un point et d'au moins un espace, puis d'une ou plusieurs phrases.

1.3.3. LA PHRASE.

Elle contient une ou plusieurs instructions et se termine par un point.

1.3.4. L'INSTRUCTION.

Elle commence par un verbe(au sens COBOL) et peut être suivie d'un ou plusieurs opérandes. Les mots utilisés dans le langage COBOL sont divisés en deux catégories, les **mots réservés** propres à COBOL et les mots définis par le développeur, souvent appelés **noms symboliques**.

Exemple :

```
parag.
   MOVE SPACE TO mazone
   IF f-cod = 5
         THEN PERFORM trait-piece
         ELSE  ADD 1 TO cpt
               MOVE f-zone TO mazone
   END-IF.
```

Les mots réservés, dont les verbes, sont notés en majuscules, les noms symboliques en minuscules.

NOTES

2. NOTION DE BLOC.

Avant l'arrivée de la version ANS 85, le COBOL ne permettait pas de traduire facilement un arbre programmatique. Autrement dit, il n'était pas aisé d'écrire en COBOL en respectant les normes de la programmation structurée.

Depuis cette version, on peut structurer l'écriture grâce à l'apparition des PERFORM en ligne et des délimiteurs explicites qui permettent de délimiter des blocs d'instructions. Ces notions seront vues en détail les deuxième et troisième jours de ce stage.

NOTES

Représentation des données

1. REGLES D'ECRITURE.

- Cinquante deux caractères sont reconnus par COBOL.
- 10 chiffres
- 26 lettres
- 16 autres caractères : + - * / = . , ; " ( ) < > : $ et l'espace.

Les noms symboliques sont des noms choisis par le développeur. Ils comportent au maximum 30 caractères, doivent commencer par une lettre et peuvent être composés de lettres, de chiffres et de traits d'union.

Parmi les noms symboliques, on trouve :

- des noms de données qui référencent une donnée en mémoire
- des noms de paragraphe
- des noms de condition
- des noms externes.

Tous les éléments d'un programme doivent être délimités par des séparateurs. Le plus courant est l'espace, mais on peut aussi utiliser les parenthèses, le point virgule, la virgule, le point et les deux points.

Conseil : éviter d'utiliser ";" et "," qui sont facultatifs et peuvent causer des erreurs de syntaxe puisqu'ils doivent être suivis d'un espace. Préférer l'espace.

Attention : un point doit toujours être suivi d'un espace au moins.

NOTES

2. DESCRIPTION DES DONNEES.

Une donnée est décrite en COBOL par :

- son nom symbolique
- sa PICTURE
- son USAGE
- son niveau
- des clauses de description

La description d'une donnée doit se terminer par un point.

2.1. LA CLAUSE PICTURE.

Cette clause permet de définir la classe et la longueur d'une donnée. La classe précise le contenu théorique de la donnée.

Il existe 3 classes :

- alphabétique (symbole A) : 26 lettres + l'espace
- numérique (symbole 9) : 10 chiffres
- alphanumérique (symbole X) : tous les caractères du code de l'ordinateur.

En pratique, on définira en numérique les zones sur lesquelles on veut effectuer des calculs, en alphanumérique toutes les autres. La longueur d'une donnée numérique ne peut dépasser 18 caractères, celle d'une donnée alphanumérique 160 caractères.

Exemple:

```
ZONA    PIC  99.
ZONB    PIC  X(20).
```

NOTES

2.1.1. DESCRIPTION DES ZONES NUMÉRIQUES (LES SYMBOLES VIRTUELS S, V, P)

Une zone numérique peut être signée et/ou décimale.

2.1.1.1 LE SYMBOLE S

Ce symbole indique à COBOL que la zone est signée. Il doit être unique et codé à gauche de la chaîne.

Comme les autres symboles virtuels, S ne compte pas dans la taille de la donnée.

Si S n'est pas spécifié, le nombre est en valeur absolue.

Exemple :

1 ZON1	PIC 9(4).	ZON1 peut contenir de 0 à 9999.
1 ZON2	PIC S9(4).	ZON2 peut contenir de -9999 à +9999

ZON1 et ZON2 auront une longueur de 4 octets.

Représentation interne en code EBCDIC :

Le signe est codé sur le quartet de gauche du dernier octet de droite.

En hexadécimal, F représente la valeur absolue, C le signe plus et D le signe moins.

Exemple :

ZON1 et ZON2 contiennent la valeur +1234

ZON1 : F1F2F3F4 ZON2 : F1F2F3C4

NOTES

2.1.1.2 LE SYMBOLE V.

Le symbole V est souvent nommé "virgule virtuelle"; il est préférable d'utiliser l'expression "marque décimale".

Ce symbole indique à COBOL l'emplacement sur lequel il devra aligner la virgule réelle si la zone est réceptrice; il doit être unique dans la description d'une donnée.

Si une donnée numérique est décrite sans V, la marque décimale se trouve implicitement à droite de la donnée.

Exemple :

1 ZON1 PIC 999. est équivalent à 1 ZON1 PIC 999V.

1 ZON2 PIC 9(4)V99. définit une zone de 6 caractères numériques dont 2 décimales.

2.1.1.3 LE SYMBOLE P.

Ce symbole, dénommé "facteur d'echelle", définit des caractères virtuels traités comme contenant zéro. Il est rarement utilisé.

Pour plus de renseignements, se reporter au manuel COBOL.

NOTES

2.1.2. LES PICTURES D'ÉDITION.

COBOL, langage de gestion, a été conçu pour faciliter l'écriture de programmes d'édition. Des symboles spéciaux ont été définis dans ce but et les données décrites en utilisant ces symboles sont dites numériques éditées ou alphanumériques éditées selon leur classe.

Contrairement aux symboles virtuels, les symboles d'édition sont comptés dans la taille de la donnée.

Symbole	Signification
Z	représente une position de caractère numérique; quand la position contient zéro, celui-ci est remplacé par un espace. Z doit figurer à gauche de la donnée.
B	représente une position de caractère dans laquelle un espace est inséré
0	représente une position de caractère dans laquelle un zéro est inséré.
/	représente une position de caractère dans laquelle un slash est inséré.
,	représente une position de caractère dans laquelle une virgule est insérée.
.	représente la marque décimale; ne doit pas être le dernier caractère de droite de la description. Il est possible d'échanger la fonction des symboles virgule et point en citant dans le paragraphe SPECIAL-NAMES la clause DECIMAL-POINT [IS] COMMA
*	symbole de protection pour les chèques; fonctionne comme Z mais en remplaçant zéro par *.
$	signe monétaire. peut être remplacé par un autre caractère dans le paragraphe SPECIAL-NAMES par la clause CURRENCY SIGN
+ **-** **CR** **DB**	représentent le signe d'une donnée numérique éditée, selon les conventions décrites ci-dessous.

NOTES

Symbole utilisé	Résultat pour une donnée >= 0	Résultat pour une donnée < 0
+	+	-
-	espace	-
CR	deux espaces	CR
DB	deux espaces	DB

Les symboles Z 0 / , + - * $ peuvent être utilisés plus d'une fois dans une description.

Les symboles . CR DB ne doivent être utilisés qu'une fois dans une description.

Evidemment les différents symboles peuvent être associés dans une description.

Exemples :

Description	Valeur de la donnée	Résultat édité
99,B999,B000	1234	01,b234,b000
Z(6)9.99	22,5	bbbbb22.50
*(9)9.99DB	-1594	******1594.00DB
+ZZZBZZZBZZ9	+25296	+bbbbb25b296
-9(3).99$	+123,456	b123.45$

où b représente un espace

NOTES

2.2. LA CLAUSE USAGE.

Par défaut, le format de la donnée est "USAGE DISPLAY", où un octet = un caractère.

Il existe d'autres usages :

- les usages dits "computationnal" pour les données numériques ; ils seront étudiés lors de la dernière journée de ce module.
- l'usage index étudié lors du module consacré aux tables.
- l'usage pointer qui définit une donnée adresse ; cet usage est très peu utilisé.

2.3. LES NIVEAUX.

La description d'une donnée doit comporter un numéro de niveau de 1 à 49 inclus. Le numéro de niveau spécifie la hiérarchie de la donnée dans un enregistrement ou un groupe de données.

Exemple : soit une zone décomposée en sous-zones

```
<- - - - - - - - - - - - - - - - - - -  ZON - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - >
                <- - - - - - - - - - - - - ZONG - - - - - - - - - - >

                <- - - - - ZONC - - - - > <- - - ZONF - - - - >
    ZONA       ZONB       FILLER       ZOND       ZONE          FILLER
I---------------I------------I--------------I----------------I-------I-------------------------------I
   5 x X          4 x 9           5            5 x 9       2 x X               9
```

```
       1 ZON.
          2 ZONA          PIC X(5).
          2 ZONG.
             3 ZONC.
                4 ZONB  PIC 9(4).
                4          PIC X(5).
             3 ZONF.
                4 ZOND  PIC 9(5).
                4 ZONE  PIC X(2).
          2 FILLER        PIC X(9).
```

NOTES

Cet exemple fait apparaître :

- différents niveaux hiérarchiques. Une zone de niveau n inclut toutes les zones de niveau supérieur qui suivent jusqu'à la rencontre d'un prochain niveau inférieur ou égal.
- des zones élémentaires, c'est-à-dire non divisées. Ce sont celles qui contiennent la clause PICTURE.
- des sous-groupes qui sont découpés en zones groupes et en zones élémentaires.
- le mot réservé FILLER.

Règles :

- une zone groupe est toujours de classe alphanumérique.
- la longueur d'une zone groupe est égale à la somme des longueurs des zones élémentaires qui la composent.
- le premier niveau d'une description de données est toujours 1 (ou 01). L'écriture de ce niveau doit se faire en colonne 8. Les niveaux 2 à 49 ne doivent pas commencer avant la colonne 12.
- FILLER est un mot réservé COBOL qui permet de décrire une donnée qui ne sera jamais référencée dans la PROCEDURE DIVISION. L'absence de nom de donnée est équivalente à FILLER (voir exemple ci-dessus).

Il existe des niveaux spéciaux :

- 77 : niveau qui peut être utilisé à la place de 01 pour des données élémentaires.
- 66 : niveau utilisé avec la clause RENAMES.
- 88 : niveau qui permet d'affecter un nom de condition à une zone. (cf. paragraphe 4.3 du chapitre 4).

NOTES

2.4. LES CLAUSES DE DESCRIPTION.

BLANK WHEN ZERO permet de remplacer les zéros non significatifs par des blancs pour une donnée numérique, ou numérique d'édition.

exemple :

```
01 NB-PAGES          PIC 9(5)  BLANK WHEN ZERO.
```

JUSTIFIED cadre le contenu d'une donnée alpha ou alphanumérique lors de sa réception. Cette clause est utilisée pour inverser le cadrage par défaut que nous étudions plus loin.

exemple :

```
01 TOTAL             PIC X(5) JUST RIGHT.
```

OCCURS est associé à un nombre ou à un nom de donnée numérique qui définit le nombre d'occurrences dans une table.

exemple :

```
01 TAB-MOIS.
     05 MOIS      PIC X(15)     OCCURS 12      [ INDEXED BY IND ].
```

Cette clause spécifie quelles sont les données qui devront être référencées à l'aide d'un indice [ou d'un index].

NOTES

DEPENDING ON est utilisé avec la clause OCCURS pour les tables de longueur variable.

exemple :

```
01  TAB-CLASST.
      05  CONCURRENT  PIC X(20)      OCCURS min  TO  max
                                      DEPENDING ON   nb-concurrents.
```

Min et max contenant des nombres entiers représentant les nombres minimum et maximum d'occurrences.

REDEFINES est utilisé pour donner une autre description à une donnée déjà décrite sous un autre nom.

Les deux données doivent être de même niveau. La deuxième donnée peut être plus grande que la première. La clause VALUE est interdite sur les données redéfinies.

La clause REDEFINES va affecter une autre nature ou un autre découpage à la donnée élémentaire ou groupe redéfini.

exemple :

```
1  ZADR                PIC X(30)  VALUE "10 PLACE DU CIRQUE".
1  ZADR2   REDEFINES  ZADR.
          5  NUMVOI   PIC 99.
          5  LIBVOI   PIC X(28).
```

NOTES

RENAMES, utilisé en niveau 66 seulement, permet de renommer une donnée ou un groupe de données dans un enregistrement. LE RENAMES doit venir juste derrière la dernière donnée élémentaire de la donnée de niveau 1.

Il ne peut renommer un niveau 01, 66, 77, OU 88.

Exemple :

```
01  ART-E.
    05  DONNEE-1   …….
    05  DONNEE-2   …….
        10  DONNEE-2A   ……
        10  DONNEE-2B   ……
    05  DONNEE-22  REDEFINES  DONNEE-2.
        10  DONNEE-3A   ……
        10  DONNEE-3B   ……
        10  DONNEE-3C   ……
    66  DONNEE-4  RENAMES  DONNEE-2A  THRU  DONNEE-3A.
```

A noter qu'un REDEFINES peut être inclus dans le groupe de données renommées. Par contre, il ne peut y avoir de clause OCCURS sur ces dernières.

VALUE permet d'initialiser le contenu d'une donnée ou de valoriser un nom de condition.

La clause VALUE est interdite en FILE SECTION et en LINKAGE SECTION.

Une donnée initialisée par value est appelée en COBOL une CONSTANTE. Le contenu de ces données est appelé LITTERAL.

NOTES

Instructions de base

1. MOVE.

MOVE est le verbe COBOL qui effectue le transfert de données d'une zone émettrice A vers une zone réceptrice B.

TRANSPARENT 01

Exemples de MOVE

Zone émettrice	Zone réceptrice AVANT	Zone réceptrice APRES
FACTEUR X(7)	bbbbbbbbbb X(10)	
MULTIPLICATION X(14)	00000 X(5)	
18041996 9(8)	000000000000 9(12)	
1996 9(4)	123 9(3)	
1996 9(4)	NEZ X(3)	
02345 9(5)	TOTO345 X(7)	
1456 9V999	0000 9(4)	
1456 9V999	0000 99V99	
25011996 X(8)	TOTOTOTOTO XX/XX/XXXX	
(-)001245 S9(4)V99	bbbbbbbbbb +ZBZZ9.99$	
123 9(3)	YYYYYYYY -*****9B000	
	DEEDITION	
bb1b456 ZZZBZZ9	666666 9(6)	
*****56.12 *(6)9.99	0000 999V9	

1.1. LES TROIS REGLES FONDAMENTALES DU MOVE.

Pour bien comprendre le fonctionnement du MOVE, il faut connaître les trois règles applicables systématiquement à ce verbe.

Règle n°1 :

- la zone émettrice n'est jamais modifiée.
- la zone réceptrice est toujours entièrement modifiée.

Cette règle est valable quelle que soit la longueur de chacune des deux zones.

Règle n°2 :

c'est la classe de la zone réceptrice qui détermine le cadrage.

- zone réceptrice numérique : cadrage sur la marque décimale, c'est-à-dire à droite pour les nombres entiers.
- zone réceptrice alphanumérique : cadrage à gauche (sauf si la zone a été définie avec la clause JUSTIFIED RIGHT).

Si la longueur de la zone réceptrice est supérieure à celle de la zone émettrice, il y a complémentation ; si c'est l'inverse, il y a troncature. Dans ces deux cas, la règle n°3 s'applique :

Règle n°3 :

- la complémentation s'effectue à droite avec des espaces si la zone réceptrice est alphanumérique, à gauche avec des zéros si la zone réceptrice est numérique.
- la troncature s'effectue à droite si la zone réceptrice est alphanumérique, à gauche si la zone réceptrice est numérique.

NOTES

1.2. FORMATS DU VERBE MOVE.

Premier format :

MOVE Nom_de_donnée1 TO Nom_de_donnée2 [Nom_de_donnée3 …]
Littéral
Constante figurative

Mouvements de zones élémentaires de classes différentes :

A	------>	9	est interdit
9	------>	X	est autorisé si la zone émettrice n'est pas décimale
X	------>	9	est autorisé aux risques et périls du développeur

Second format :

MOVE CORRESPONDING nom_de_donnée1 TO nom_de_donnée2

nom_de_donnée1 et nom_de_donnée2 sont des données groupes.

Cette forme de MOVE permet, en une seule instruction, de copier toutes les données élémentaires de la donnée groupe émettrice vers les données élémentaires de même nom de la donnée groupe réceptrice. Les autres zones ne sont pas modifiées.

NOTES

Exemple : retournement d'une date.

```
1  DAT1.
     5  AA     PIC XX.
     5  MM     PIC XX.
     5  JJ     PIC XX.

1  DAT2.
     5  JJ     PIC XX.
     5          PIC  X  VALUE '/'.
     5  MM     PIC XX.
     5          PIC X(3)  VALUE '/19'.
     5  AA     PIC XX.
```

MOVE CORRESPONDING DAT1 TO DAT2

Si DAT1 contient "950825", DAT2 contiendra "25/08/1995".

Remarque :

- l'option CORRESPONDING peut être utilisée avec les instructions arithmétiques ADD et SUBTRACT.

NOTES

1.3. MOUVEMENT DE LITTERAUX.

On a vu précédemment avec la clause VALUE que le contenu de constantes était appelé LITTERAL.

MOVE permet également d'initialiser les données en PROCEDURE DIVISION en utilisant comme émetteur un littéral.

Règles :

- un littéral alphanumérique doit être écrit entre guillemets, il est limité à 160 caractères.
- un littéral numérique doit être écrit sans guillemet, il est limité à 18 caractères. Il peut être signé ou non, décimal ou non.
- les mouvements de littéraux obéissent aux règles du MOVE.
- un littéral doit être de la classe de la donnée qui va le recevoir et sa taille doit être inférieure ou égale à la taille de cette donnée.

Continuation d'un littéral alphanumérique :

- écrire jusqu'à la colonne 72 sans fermer par un guillemet.
- ligne suivante : écrire un tiret en colonne 7 puis reprendre l'écriture du littéral par un guillemet à partir d'une colonne quelconque supérieure ou égale à 12.

NOTES

1.4. MOUVEMENT DE CONSTANTES FIGURATIVES.

Les constantes figuratives sont des mots réservés qui ont été créés en COBOL pour faciliter l'écriture. Elles remplacent avantageusement l'utilisation des constantes ou des littéraux.

ZERO représente un ou plusieurs zéros : MOVE ZERO TO ZONE permet de remplir ZONE de zéros quelle que soit la classe et la longueur de ZONE.

SPACE représente un ou plusieurs espaces : MOVE SPACE TO ZONE permet de remplir ZONE d'espaces. ZONE doit être de classe alphanumérique (ou alphabétique).

Remarque :

- MOVE " " TO ZONE est équivalent (application des trois règles du MOVE). Par contre, pour tester si une zone est remplie d'espaces, la constante figurative SPACE est pratique.

ALL permet de remplir la totalité d'une zone avec un caractère. MOVE ALL "Z" TO ZONE aura pour effet de remplir ZONE de Z. ZONE doit être alphanumérique.

HIGH-VALUE et LOW-VALUE représentent la plus forte et la plus faible valeur qu'il est possible de codifier sur un caractère, soit les valeurs hexadécimales FF et 00. La zone réceptrice doit être alphanumérique.

Ces deux constantes figuratives sont utiles, entre autres, pour les appariements de fichiers.

NOTES

2. INITIALIZE.

L'instruction INITIALIZE permet de sélectionner différentes catégories de données et de les initialiser à des valeurs prédéfinies.

Format :

```
INITIALIZE  nom_de_donnée1
[REPLACING      ALPHABETIC                DATA BY    nom_de_donnée2]
                ALPHANUMERIC                         litteral
                NUMERIC
                ALPHANUMERIC-EDITED
                NUMERIC-EDITED
```

INITIALIZE agit comme une série de MOVE.

- la zone à initialiser nom_de_donnée1 peut être une zone groupe ou une zone élémentaire.
- si l'option **REPLACING** n'est pas utilisée, cas le plus courant, les zones élémentaires alphanumériques sont initialisées à espace, les zones numériques à zéro.
- **ATTENTION**, ne sont pas affectées par **INITIALIZE** :
  - ◊ les zones décrites en **FILLER**
  - ◊ les zones redéfinies
  - ◊ les index

NOTES

Exemple :

```
1  ZONZON.
       5 ZON1       PIC X(4)       VALUE "5555".
       5 ZON2       PIC 9(2)       VALUE 32.
       5 ZON.
            10 ZON3      PIC X(5) VALUE "pilou".
            10           PIC X(6) VALUE "COUCOU".
       5 ZON4       PIC 9(4)       VALUE 1587.
```

INITIALIZE ZONZON a pour effet d'initialiser à espace ZON1 et ZON3, à zéro ZON2 et ZON4, le filler contenant "COUCOU" n'est pas touché.

INITIALIZE ZONZON REPLACING ALPHANUMERIC DATA BY "TT" a le même effet que MOVE "TT" TO ZON1 ZON3.

NOTES

TP1A

Exécuter le programme TP1FAU puis corriger la source pour obtenir un affichage correct des résultats

3. VERBES D'ENTREE/SORTIE SEQUENTIELS.

Nous ne verrons que les formats séquentiels des verbes d'E/S, il en existe d'autres pour les accès directs qui sont étudiés dans un autre module.

3.1. OUVERTURE DES FICHIERS.

OPEN INPUT Nom_de_fichier : ouverture en lecture

OPEN OUTPUT Nom_de_fichier : ouverture en écriture

OPEN EXTEND Nom_de_fichier : ouverture en écriture et positionnement derrière le dernier enregistrement du fichier.

L'OPEN réalise la lecture ou la création de labels et la réservation des ressources nécessaires (buffers, unités de disques, canaux).

3.2. FERMETURE DES FICHIERS.

CLOSE Nom_de_fichier

Le CLOSE effectue l'écriture des labels de fin et la libération des ressources allouées au fichier.

NOTES

3.3. LECTURE D'UN ARTICLE DE FICHIER.

Format :

```
READ Nom_de_fichier [INTO Nom_de_donnée]
     AT END instruction_impérative
     [NOT AT END instruction_impérative]
[END-READ]
```

READ a pour effet d'appliquer le (ou les) masque(s) logique(s) défini(s) en FILE SECTION sur l'article physique présent dans le buffer de lecture". Si l'option INTO est utilisée, l'enregistrement est copié dans Nom_de_donnée.

Remarque : le nom de la zone de communication n'est pas cité, c'est le nom du fichier qu'il faut écrire.

3.4. ECRITURE D'UN ARTICLE SUR UN FICHIER.

Format :

- WRITE Nom_de_zone_de_communication [FROM Nom_de_donnée]
  [END WRITE]

WRITE réalise l'écriture d'un article dans le buffer du fichier. Si l'option FROM est utilisée, il y a copie préalable de *nom_de_donnée* dans *nom_de_zone_de_communication* ; ce qui économise l'écriture d'un MOVE.

Remarque :

- contrairement à READ, il faut citer le nom de la zone de communication et pas le nom du fichier.

NOTES

3.5. DISPLAY ET ACCEPT.

Ces deux verbes permettent de lire ou d'écrire depuis ou sur des unités physiques (console, clavier, imprimante …) à partir de fichiers systèmes qui leur sont associés. Ces fichiers n'ont pas besoin d'être ouverts ou fermés.

Leur nom est en général SYSIN pour les entrées, SYSOUT pour les sorties.

3.5.1. DISPLAY :

Format :

```
DISPLAY     Nom_de_donnée1   [Nom_de_donnée2]
            Littéral1        [Littéral2]
      [UPON  Nom_mnémonique ou Nom_système]
      [WITH NO ADVANCING]
```

Si UPON n'est pas cité, l'écriture se fait sur un périphérique par défaut. Il y a saut de ligne à chaque nouveau DISPLAY sauf si l'option WITH NO ADVANCING est précisée.

Nota : DISPLAY convertit les données définies en USAGE autre que DISPLAY en USAGE DISPLAY.

NOTES

3.5.2. ACCEPT :

Premier format :

ACCEPT Nom_de_donnée [FROM Nom_mnémonique ou Nom_système]

Si FROM n'est pas cité, la lecture se fait à partir d'un périphérique par défaut.

Les données sont récupérées dans *Nom_de_donnée*, qui doit être déclaré en DATA DIVISION.

Second format :

ACCEPT Nom_de_donnée FROM DATE
DAY
TIME
DAY-OF-WEEK

Ce format permet de récupérer des informations du système.

DATE	format implicite 9(6) sous la forme AAMMJJ
DAY	format implicite 9(5) sous la forme AAQQQ
TIME	format implicite 9(8) sous la forme HHMNSSCC
DAY-OF-WEEK:	format implicite 9 donne le rang du jour dans la semaine (1 = Lundi)

NOTES

4. LES BRANCHEMENTS.

4.1. LE BRANCHEMENT SIMPLE.

GO TO Nom_de_paragraphe

Cette instruction est prohibée en programmation structurée.

4.2. LE BRANCHEMENT AVEC RETOUR.

Il s'agit de l'instruction PERFORM qui permet de se débrancher pour exécuter une séquence d'instructions et de se rebrancher à l'instruction suivant le PERFORM.

Cette instruction possède de nombreux formats :

♦ PERFORM Nom_de_procédure

Nom_de_procédure est un nom de paragraphe ou de section.

Si c'est un nom de paragraphe, le rebranchement s'effectue après exécution du paragraphe.

NOTES

Si c'est un nom de section, le rebranchement s'effectue après la dernière instruction du dernier paragraphe inclus dans la section.

♦ PERFORM Nom_de_procédure1 THRU Nom_de_procédure2

Exécution depuis procedure1 jusqu'à procédure2 incluse. Le rebranchement s'effectue dans les mêmes conditions que le format précédent, la dernière instruction étant incluse dans Nom_de_procédure2.

♦ PERFORM Nom_de_procédure1 [THRU Nom_de_procédure2]
n TIMES

Exécution d'une séquence d'instructions n fois ; n est un littéral ou un nom de donnée et représente une valeur numérique entière.

♦ PERFORM Nom_de_procédure1 [THRU Nom_de_procédure2]
[WITH TEST BEFORE] UNTIL Condition
TEST AFTER

Exécution d'une séquence d'instructions jusqu'à ce qu'une condition soit vraie.

Cette dernière peut être une expression conditionnelle ou un nom_de_condition.

L'option WITH TEST permet de demander le test de la condition avant ou après exécution de la séquence d'instructions. Pour des raisons de compatibilité ascendante avec COBOL 74, l'option par défaut est TEST BEFORE.

♦ PERFORM Nom_de_procédure1 [THRU Nom_de_procédure2]
[WITH TEST AFTER]
VARYING Nom_de_donnée
FROM Valeur initiale BY Incrément
UNTIL Condition

NOTES

Exécution d'une séquence d'instructions jusqu'à la réalisation d'une condition tout en incrémentant une donnée à partir d'une valeur initiale.

Nom_de_donnée représente une donnée numérique ou un index.

Valeur initiale peut être un littéral numérique ou un nom de donnée numérique de signe quelconque.

Même chose pour Incrément.

Ce format de PERFORM est très utilisé lors de la manipulation de tables.

♦ PERFORM [THRU …]
[WITH …]
VARYING … FROM … BY …
UNTIL …..
AFTER
FROM
UNTIL
AFTER
FROM
UNTIL
……

AFTER peut être utilisé six fois, c'est à dire qu'on peut faire varier jusqu'à 7 indices ou index, ce qui correspond au nombre maximum de niveaux pour une table.

NOTES

4.2.1. PERFORM EN LIGNE.

```
PERFORM
    Séquence d'instructions impératives
END-PERFORM
```

Le PERFORM dit "en ligne" est une variante du PERFORM que nous venons de détailler. Celle-ci a été introduite dans la version 85 de COBOL pour satisfaire aux exigences de la programmation structurée.

Il permet de mettre en évidence et d'exécuter un bloc d'instructions. Ce bloc est délimité par PERFORM et son délimiteur explicite END-PERFORM. Aucun point n'est autorisé entre ces deux verbes.

Le PERFORM "on line" fonctionne de la même façon que le PERFORM "out of line", ou PERFORM classique, à cette différence que les instructions sont énumérées immédiatement sans recourir à l'utilisation d'un nom de paragraphe.

Le PERFORM en ligne accepte les mêmes formats que les PERFORM classiques à l'exception de THRU.

NOTES

4.3. L'INSTRUCTION EXIT.

C'est une instruction exécutable, dite de débranchement.

Elle est utilisée pour donner une fin commune à une série de paragraphes appelée par PERFORM … THRU ….

En standard, EXIT doit être la seule instruction du paragraphe.

4.4. STOP RUN.

L'exécution de cette instruction provoque l'arrêt du programme et rend le contrôle au système d'exploitation.

STOP RUN ferme tous les fichiers du programme encore ouverts.

5. LES INSTRUCTIONS ARITHMETIQUES.

Elles ne sont utilisables que sur des zones numériques, donc des données élémentaires ou des littéraux numériques.

5.1. L'ADDITION.

```
ADD     Nom_de_donnée1    [Nom_de_donnée2 …]
  TO    Nom_de_donnée3    [Nom_de_donnée4 …]
```

Les contenus de *donnée1, donnée2*, ne sont pas modifiés et sont ajoutés au contenu de *donnée3, donnée4*.

NOTES

5.2. LA SOUSTRACTION.

```
SUBTRACT    Nom_de_donnée1   [Nom_de_donnée2  …]
    FROM    Nom_de_donnée3   [Nom_de_donnée4  …]
```

Les contenus de *donnée1, donnée2*, ne sont pas modifiés et sont soustraits du contenu des *donnée3, donnée4 …*

5.3. LA MULTIPLICATION.

```
MULTIPLY  Nom_de_donnée1  BY    Nom_de_donnée2
```

Le résultat de la multiplication est envoyé dans *donnée2*. Le contenu de *donnée1* reste inchangé.

5.4. LA DIVISION.

```
DIVIDE    Nom_de_donnée1  INTO  Nom_de_donnée2
```

donnée2 est divisée par *donnée1* et reçoit le résultat. *donnée1* reste inchangée.

NOTES

5.5. COMPUTE.

Ce verbe permet de demander en une instruction le calcul d'une expression arithmétique composée.

COMPUTE Nom_de_donnée1 [Nom_de_donnée2] = expression arithmétique

Opérateurs utilisables :

+	unaire	: multiplication par +1
-	unaire	: multiplication par -1
**		: exponentiation
*		: multiplication
/		: division
+		: addition
-		: soustraction

Sans parenthèse, l'évaluation des opérations se fait dans l'ordre où les opérateurs sont cités.

L'usage des parenthèses est vivement recommandé ; les expressions entre parenthèses sont évaluées en premier en commençant par le niveau de parenthèses le plus inclus.

NOTES

5.6. OPTIONS COMMUNES AUX INSTRUCTIONS ARITHMÉTIQUES.

5.6.1. GIVING

Format :

♦ instruction_arithmétique GIVING nom_de_résultat

Le résultat du calcul est envoyé dans *nom_de_résultat*, les opérandes sont inchangés. *Résultat* peut être une zone numérique d'édition.

Pour la division, l'utilisation de GIVING autorise celle de REMAINDER qui permet de récupérer le reste de la division ; elle autorise également l'usage de BY à la place de INTO, il faut alors inverser les opérandes.

Exemple :

```
5  diviseur      PIC 9(3)       VALUE 30.
5  dividende     PIC 9(3)       VALUE 80.
5  quotient      PIC 9(3)V9(2)  VALUE 0.
5  reste         PIC 9(3)V9(2)  VALUE ZERO.
```

DIVIDE dividende BY diviseur GIVING quotient REMAINDER reste

Contenu des données modifiées :

♦ quotient = 2.67
♦ reste = 0.2

NOTES

5.6.2. ON SIZE ERROR

Format :

```
instruction_arithmétique  ON SIZE ERROR          instruction_impérative1
                          [NOT ON SIZE ERROR     instruction_impérative2]
[délimiteur_de_l'instruction_arithmétique]
```

Cette option permet de prévoir le traitement spécifique d'une erreur de calcul et d'éviter ainsi un plantage machine ou, plus grave, un résultat aberrant.

5.6.3. ROUNDED

L'option ROUNDED s'écrit juste après le nom de la zone résultat.

Exemples :

```
MULTIPLY A BY B GIVING C ROUNDED
COMPUTE RES ROUNDED = A * (B - C ) / D
```

ROUNDED réalise l'arrondi en ajoutant 5 au premier chiffre à tronquer de droite.

5.6.4. CORRESPONDING

Cette option déjà vue avec MOVE est utilisable avec ADD et SUBTRACT.

NOTES

TP2A

A partir du fichier EDU1.DAT

qui recense, pour les villes de la banlieue parisienne, le nombre d'établissements d'éducation on vous demande

d'établir, en sortie, sur EDU2.DAT, la totalisation par département et la totalisation générale pour la banlieue,

avec le code 99.

Prévoir l'affichage des résultats pour contrôle.

EDU1.DAT, trié sur le codique du département, comprend :

- le n° codique du département 2 caractères
- le libellé du département 20 caractères
- le nom de la ville 10 caractères
- le nombre d'établissements 3 caractères

EDU2.DAT comprend :

- le n° codique du département ou 99 2 caractères
- le total 3 caractères

FICHE 04

Les instructions conditionnelles

1. DÉFINITION

COBOL possède deux catégories d'instructions :

- les instructions impératives
- les instructions conditionnelles

Une instruction impérative est une instruction dont l'exécution n'est pas soumise à la réalisation d'une condition.

Une instruction conditionnelle est une instruction dont l'exécution est soumise à la réalisation d'une condition.

Certains verbes COBOL donnent toujours des instructions impératives.

Par exemple :

- MOVE, PERFORM, OPEN, CLOSE …etc…

D'autres sont toujours conditionnels :

- IF, EVALUATE.

D'autres encore peuvent donner des instructions impératives ou conditionnelles selon le format utilisé :

Exemples :

- les opérations arithmétiques sans ou avec l'option ON SIZE ERROR
- le verbe READ sans ou avec la clause AT END

NOTES

2. TRANSFORMATIONS D'INSTRUCTIONS CONDITIONNELLES EN IMPÉRATIVES.

Pourquoi ? Il s'agit en fait de permettre l'imbrication d'instructions. Or, COBOL ne permet d'imbriquer que des instructions impératives (à l'exception du IF) ; pour pouvoir imbriquer n'importe quelle instruction, ce qui est nécessaire en programmation structurée, il faut donc pouvoir transformer une instruction conditionnelle en instruction impérative.

Depuis la version ANS85, chaque instruction COBOL conditionnelle ou pouvant l'être possède un délimiteur explicite, c'est à dire qui lui est propre, par opposition au point (".") qui est le délimiteur implicite. Une instruction conditionnelle terminée par son délimiteur explicite devient une instruction impérative et peut ainsi être imbriquée.

Un bloc d'instructions imbriquées ne doit pas comprendre de point car le point ferme toutes les instructions conditionnelles non délimitées qui le précèdent.

Conclusion : écrire en COBOL structuré implique d'utiliser systématiquement les délimiteurs explicites et d'écrire uniquement les points obligatoires ; rappel de ceux-ci :

- fin de titre
- fin de phrase
- fin de programme

Un paragraphe est donc composé d'un nom de paragraphe suivi d'un point et d'une seule phrase contenant n instructions.

NOTES

3. IF.

Format :

```
IF condition
      [THEN]      instruction1
      [ELSE       instruction2]
[END-IF]
```

- *condition* peut être une expression ou un nom_de_condition

- THEN est facultatif pour des raisons de compatibilité ascendante, mais il est conseillé de le coder

- *instruction1* et *instruction2* peuvent être n'importe quelle instruction ou séquence d'instructions.

- NEXT SENTENCE peut être utilisé à la place de *instruction1* ou *instruction2*; NEXT SENTENCE est une instruction COBOL de débranchement qui entraîne le rebranchement après la fin de la phrase, c'est à dire après le premier point rencontré. Si on écrit en suivant le conseil du §2, où une phrase est un paragraphe, NEXT SENTENCE entraîne le débranchement du paragraphe, proscrit en programmation structurée.

Conclusion : en COBOL structuré, NEXT SENTENCE est prohibé.

NOTES

COBOL ANS85 a introduit la nouvelle instruction CONTINUE qui signifie "ne rien faire". Cette instruction utilisée à la place de *instruction1* permet de continuer l'exécution après le END-IF.

Exemple :

```
IF NOT FIN-FICHIER AND A = B AND C  > D
            THEN CONTINUE
            ELSE MOVE  ZON TO SAUVE
END-IF
```

Remarque :

Pour des raisons de compatibilité ascendante avec les précédentes versions de COBOL, IF est la seule instruction conditionnelle qui peut être imbriquée sans utiliser son délimiteur. Le bloc de IF imbriqués est alors fermé par le premier point rencontré. Il est vivement conseillé d'utiliser systématiquement END-IF et surtout de ne pas mélanger les deux formes.

TRANSPARENT 02

Exemple :

Ecriture avec délimiteur (ANS 85)

```
IF IND = 3
   THEN
      IF DEP = "91"
         THEN  PERFORM TRAIT-91
         [ ELSE CONTINUE ]
      END-IF
   ELSE
      IF DEP-DOM
         THEN PERFORM TRAIT-DOM
         ELSE PERFORM TRAIT-AUTRE
      END-IF
END-IF
```

Ecriture sans délimiteur (ANS 74)

```
IF IND = 3
    IF DEP = "91"
    PERFORM TRAIT-91
    ELSE
        NEXT SENTENCE
ELSE
    IF DEP-DOM
        PERFORM TRAIT-DOM
    ELSE
        PERFORM TRAIT-AUTRE
.
```

NOTES

TP3A

A partir du fichier EDU3.DAT

qui recense, pour les villes de la banlieue parisienne, le nombre d'élèves par classe et la nature de chaque classe, classe de langue ou non,

EDU3.DAT, trié sur le n° codique du département, comprend :

- le n°codique du département 2 caractères
- le libellé du département 20 caractères
- le nom de la classe 12 caractères (nom de la ville+numéro séquentiel)
- le nombre d'élèves 2 caractères
- classe de langue ou non 1 caractère (blanc si aucune langue ou
- E si ANGLAIS ou
- D si ALLEMAND)

on vous demande de déterminer et d'afficher, pour chaque classe, si son effectif est NORMAL, TROP IMPORTANT ou TROP FAIBLE, en fonction des 4 critères suivants :

- nombre d'élèves < =20 et classe de langue ou non = FAIBLE
- nombre d'élèves > 20 et < = 30 et pas classe de langue = NORMAL
- nombre d'élèves > 20 et < = 30 et classe de langue = IMPORTANT
- nombre d'élèves > 30 et classe de langue ou non = IMPORTANT

4. LES EXPRESSIONS CONDITIONNELLES.

Elles sont utilisées avec les verbes de condition (IF, EVALUATE), avec PERFORM conditionné par UNTIL et avec SEARCH (verbe de recherche en table vu dans un autre module). Les expressions conditionnelles peuvent être simples ou complexes.

4.1. LES CONDITIONS DE CLASSE.

Format: Nom_de_donnée [IS] (NOT) NUMERIC
ALPHABETIC
ALPHABETIC-LOWER
ALPHABETIC-UPPER

C'est le contenu de la donnée qui est testé.

- NUMERIC : utilisable sur des zones définies en PIC 9 ou X
- ALPHABETIC : utilisable sur des zones définies en PIC A ou X
- ALPHABETIC-LOWER : idem ALPHABETIC + test si minuscules
- ALPHABETIC-UPPER : idem ALPHABETIC + test si majuscules

NOTES

4.2. LES CONDITIONS DE SIGNE.

Format :

nom_de_donnée [IS] (NOT) POSITIVE
NEGATIVE
ZERO

nom_de_donnée doit être une donnée numérique ou une expression arithmétique qui sera évaluée avant le test.

4.3. LES NOMS DE CONDITION.

Ce sont des noms mnémoniques associés à une condition portant sur le contenu d'une donnée. Ils sont définis par un niveau 88 sur la ligne de DATA DIVISION (FILE ou WORKING-STORAGE SECTION) qui suit immédiatement la ligne où est définie la donnée.

Les noms de condition présentent plusieurs avantages :

- simplification de l'écriture de la condition et donc meilleure lisibilité
- par un choix judicieux du nom, permet une compréhension plus rapide du code écrit ainsi qu'un auto-commentaire de la définition des données.
- permet de savoir rapidement quelles sont les valeurs que peut prendre la donnée sans avoir à consulter l'analyse ou le cahier des charges.

NOTES

Exemple :

```
FILE SECTION.
FD FIC.
1     F-ENR.
      5     F-IDENT             PIC X(20).
      5     F-DEP               PIC X(2).
      88    PARIS               VALUE "75".
      88    ILE-DE-FRANCE       VALUE "75" "77" "78" "91" THRU "95".
      88    CORSE               VALUE "2A" "2B".
      5                         PIC X(478).
```

Dans la PROCEDURE DIVISION, il sera plus simple et plus compréhensible d'écrire IF ILE-DE-FRANCE plutôt que IF F-DEP = "75" OR "77" OR "78" OR "91" OR "92" OR "93" OR "94" OR "95"

Remarques :

THRU permet de tester un intervalle croissant et continu (dans l'ordre du code EBCDIC ou ASCII) de valeurs.

NOTES

Un nom de condition peut être appliqué sur une zone groupe.

Un nom de condition peut être appliqué sur un FILLER.

Une condition peut être positionnée à vrai par le verbe SET ; Exemple : SET CORSE TO TRUE rend la condition CORSE vraie en mouvementant la **première valeur** du nom de condition dans la donnée. F-DEP contiendra alors "2A". Si on veut que F-DEP contienne "2B", il faut coder un MOVE "2B" TO F-DEP.

Autre exemple :

```
1     ZON.
      88   ZON-VRAIE          VALUE "ABC" "WXYZ".
      5    ZON2      PIC 9(3) VALUE 45.
      5              PIC X    VALUE "H"
                                        Contenu de ZON (PICTURE implicite X(4))
IF NOT ZON-VRAIE                        "045H"
    SET ZON-VRAIE TO TRUE               "ABC "
END-IF
```

Remarque :

- sur certains compilateurs, en particulier celui de GCOS7, une condition peut être positionnée à FALSE. Cette possibilité n'est pas au standard ANS 85.

Exemple :

```
1             PIC 9 VALUE 1.
88  FIN-FIC       VALUE 1 FALSE 0.
IF FIN-FIC
    SET FIN-FIC TO FALSE
END-IF
```

NOTES

4.4. LES CONDITIONS DE RELATION

Comparaison de deux opérandes au moyen d'un opérateur.

Liste des opérateurs relationnels :

>	(ou GT)	strictement supérieur à
> =	(ou GE)	supérieur ou égal à
<	(ou LT)	strictement inférieur à
< =	(ou LE)	inférieur ou égal à
=	(ou EQ)	égal à
NOT =	(ou NE)	différent de

Chaque opérateur relationnel doit être précédé et suivi d'au moins un blanc.

4.5. LES CONDITIONS COMPLEXES

Une condition complexe est formée de conditions simples combinées au moyen d'opérateurs logiques.

Liste des opérateurs logiques dans l'ordre d'évaluation hors parenthèse :

- NOT négation
- AND "et"
- OR "ou" inclusif (il n'existe pas de "ou" exclusif en COBOL)

Chaque opérateur logique doit être précédé et suivi d'au moins un blanc.

NOTES

5. EVALUATE.

Le verbe EVALUATE, introduit dans la version ANS 85, est ce qu'on appelle en français une instruction de choix multiple (dans d'autres langages, le choix multiple est appelé CASE). Il permet de remplacer avantageusement une succession de IF imbriqués. Il permet également de traduire simplement en COBOL une table de décision.

Format :

```
EVALUATE        sujet_1      [ALSO sujet_2 ….  ALSO sujet_n]
    WHEN        objet_11     [ALSO objet_21 …. ALSO objet_n1]
            instructions_impératives_1
    [WHEN       objet_12     [ALSO objet_22 …. ALSO objet_n2]
            instructions_impératives_2]
    …..
    [WHEN       objet_1m     [ALSO objet_2m …. ALSO objet_nm]
            instructions_impératives_m]
    [WHEN OTHER
            instructions_impératives_o]
 [END-EVALUATE]
```

NOTES

Un sujet et objet peut être un nom de donnée, une expression, une valeur, TRUE/FALSE ou ANY avec la correspondance suivante entre sujet et objet (ANY peut remplacer n'importe quel sujet ou objet) :

♦ SUJET :	nom_de_donnée	valeur	expression	TRUE/FALSE
♦ OBJET :	valeur	nom_de_donnée	TRUE/FALSE	expression

Un sujet possède autant d'objets qu'il y a de WHEN codés.

Dans une sélection multiple, les objets et sujets sont séparés par ALSO et se correspondent par leur position ordinale.

Pour chaque WHEN il doit y avoir autant d'objets qu'il y a de sujets.

Quand, pour un sujet, la valeur d'un objet peut être quelconque, il faut coder ANY.

ALSO est l'équivalent du "et" logique, WHEN est celui de "ou" logique.

Au premier WHEN vrai, les instructions impératives qui le suivent sont exécutées et il y a rebranchement après le END-EVALUATE (ou s'il n'y a pas de END-EVALUATE, après le premier point rencontré), les WHEN suivants sont ignorés.

Les instructions impératives qui suivent WHEN OTHER sont exécutées quand aucun des WHEN qui précèdent n'est vrai.

On peut tester un intervalle de valeurs par l'expression "m THRU n", considérée comme une valeur.

Les conditions complexes sont interdites.

L'usage de WHEN OTHER, optionnel, est vivement recommandé.

NOTES

Quelques exemples :

```
EVALUATE  SITUATION
        WHEN "C"   PERFORM trait-celib
        WHEN "D"   PERFORM trait-divorce
        WHEN "V"   PERFORM trait-veuf
        WHEN "M"
        WHEN "K"   PERFORM trait-couple
        WHEN OTHER PERFORM erreur
END-EVALUATE

EVALUATE   TRUE              ALSO        TRUE
        WHEN age < 13        ALSO      sexe = "F"   PERFORM fille
        WHEN age < 13        ALSO      sexe = "M"   PERFORM garcon
        WHEN age >= 13       ALSO      age < 18     PERFORM ado
        WHEN OTHER perform adulte
END-EVALUATE

EVALUATE   TRUE              ALSO        sexe
        WHEN age < 13        ALSO      "F"    PERFORM fille
        WHEN age < 13        ALSO      "M"    PERFORM garcon
        WHEN age >= 13       ALSO      ANY
              IF age < 18
                    THEN PERFORM ado
                    ELSE PERFORM adulte
              END-IF
END-EVALUATE
```

NOTES

TP4A

reprendre le TP3A mais utiliser l'instruction EVALUATE à la place de l'instruction IF et mettre un nom de condition pour la classe de langue.

FICHE 05

Traitement des chaînes de caractères

1. REFERENCE-MODIFICATION.

Depuis la version COBOL ANS 85, il est possible de travailler sur des portions de données, en citant la position de début dans la donnée et la longueur de la portion de donnée. Cette technique, nommée "référence-modification" en jargon COBOL, est souvent appelée "adressage par base/déplacement" dans d'autres langages.

Format : Nom_de_donnée(position_de_début:[longueur])

- *position_de_début* et *longueur* sont des entiers strictement positifs ou des expressions arithmétiques dont les valeurs sont des entiers strictement positifs.

- si *longueur* n'est pas citée, cela signifie : jusqu'à la fin de la zone.

- il n'est pas besoin de taper d'espace de chaque côté des deux points.

- *Nom_de_donnée* peut être un élément de table.

- *Nom_de_donnée* doit être définie en USAGE DISPLAY.

La portion de donnée adressée en référence-modification est **toujours alphanumérique**, quelle que soit la classe de nom_de_donnée.

Exemples :

```
1)      1  num-ss              PIC 9(13) VALUE 2590351454024.
        1  datnai               PIC 9(4).
        MOVE num-ss(2:4) TO datnai          datnai contient 5903
        MOVE num-ss(2:2) TO datnai          datnai contient ???
```

NOTES

2. INSPECT.

Cette instruction permet de remplacer et/ou de compter des caractères dans une donnée. Associée à la référence-modification, cette instruction est très pratique pour le contrôle et le formatage de zones de saisies.

Le format d'INSPECT est très riche car plusieurs possibilités sont offertes au développeur.

Format :

```
INSPECT nom_de_donnée1
      TALLYING compteur   FOR ALL        nom_de_donnée2 ou littéral2
                              LEADING nom_de_donnée2 ou littéral2
                              CHARACTERS
      REPLACING CHARACTERS BY nom_de_donnée3 ou littéral3
                    ALL       nom_de_donnée4 ou littéral4  BY nom_de_donnée5 ou littéral5
                    FIRST     nom_de_donnée4 ou littéral4  BY nom_de_donnée5 ou littéral5
                    LEADING   nom_de_donnée4 ou littéral4  BY nom_de_donnée5 ou littéral5
      CONVERTING      nom_de_donnée6 ou littéral6 TO nom_de_donnée7 ou littéral7
          [BEFORE     INITIAL  nom_de_donnée8 ou littéral8]
          [AFTER      INITIAL  nom_de_donnée8 ou littéral8]
```

NOTES

Remarques générales :

Les données sont en USAGE DISPLAY de classe quelconque ; *donnée1* peut être une zone groupe, les autres données doivent être des zones élémentaires.

Compteur est une zone numérique qui reçoit le résultat du comptage demandé ; ce compteur doit être réinitialisé avant chaque INSPECT.

Les littéraux doivent être alphanumériques ; on peut employer les contantes figuratives ne commençant pas par "ALL".

Le comptage et/ou le remplacement s'effectue toujours de la gauche vers la droite.

Si l'option BEFORE/AFTER n'est pas utilisée, l'examen de la zone commence par le premier caractère de gauche et continue caractère par caractère jusqu'au dernier caractère de droite.

Si l'option BEFORE est utilisée, l'examen de la zone commence par le premier caractère de gauche et s'interrompt à la rencontre du ou des caractère(s) spécifié(s) dans nom_de_donnée8 ou littéral8.

Si l'option AFTER est utilisée, l'examen de la zone commence à la rencontre du ou des caractère(s) spécifié(s) dans nom_de_donnée8 ou littéral8 et continue caractère par caractère jusqu'au dernier caractère de droite.

Les options BEFORE et AFTER peuvent être utilisées conjointement.

Les options de comptage (TALLYING) et de remplacement (REPLACING) peuvent être utilisées conjointement ; dans ce cas, le comptage s'effectue avant le remplacement.

NOTES

2.1. OPTION TALLYING.

- ALL :
  - ◊ toutes les occurrences sont comptées.
- LEADING :
  - ◊ les occurrences sont comptées à partir de la gauche jusqu'à la première différence.
- CHARACTERS :
  - ◊ on compte le nombre de caractères de la zone.

Exemples :

```
1  zon          PIC X(15)        VALUE "***18,82 €*****".
1    VALUE ZERO.
     5  cpt1          PIC 9(4).
     5  cpt2          PIC 9(4).
     5  cpt3          PIC 9(4).
INSPECT zon TALLYING cpt1 FOR ALL "*"                        cpt1 = 8
INSPECT zon TALLYING cpt2 FOR  LEADING "*"                   cpt2 = 3
INSPECT zon(cpt2 + 1:) TALLYING cpt3 FOR CHARACTERS
                                BEFORE INITIAL "*"           cpt3 = 7
```

NOTES

2.2. OPTION REPLACING.

ALL et LEADING : idem TALLYING en remplaçant au lieu de compter.

CHARACTERS : on remplace les caractères par **LE** caractère spécifié.

FIRST : seule la première occurrence rencontrée est remplacée.

TRANSPARENT 03

Exemples :

```
   1  zon          PIC X(15)          VALUE     "***18,82 €*****".
   1  cpt          PIC 9(4)  VALUE 0.
   INSPECT zon REPLACING CHARACTERS BY "F" AFTER INITIAL "F"
   contenu de zon après :              "***18,82 €***"
   INSPECT zon TALLYING cpt FOR LEADING "*"
                        REPLACING FIRST "." BY ","                    cpt = 3
   contenu de zon après :              "***18,82 €***"

1  zong.
   5  tata         PIC X(9)   VALUE "SAUTILLES".
   5               PIC X(5)   VALUE " AUX"
   5  titi         PIC X(9)   VALUE "SAUTILLES".
   1  cpt          PIC 9(4)   VALUE 0.
INSPECT tata     TALLYING cpt FOR ALL "S"
                          REPLACING ALL "LL" BY "SS"
Contenu de tata après :    "SAUTISSES"          valeur de cpt = 2 (comptage avant remplacement)
INSPECT tata     REPLACING FIRST "T" BY "C"
Contenu de tata après :    "SAUCISSES"
INSPECT titi     REPLACING LEADING "SAU" BY "LEN"
Contenu de titi après :    "LENTILLES"
Contenu de zong :          ???
```

NOTES

2.3. OPTION CONVERTING.

donnée6 (ou *littéral6*) et *donnée7* (ou *littéral7*) doivent être deux chaînes de caractères de même longueur ; chaque caractère à convertir de *donnée1* sera remplacé, s'il existe dans *donnée6*, par le caractère de même rang de *donnée7*.

Exemple :

```
1 zon      PIC X(60)     VALUE      "NOM:de ponthieu PRENOM:pierre-marie".
1 ch1      PIC X(6)      VALUE      "AEYUIO".
1 ch2      PIC X(6)      VALUE      "aeyuio".
INSPECT zon CONVERTING ch2 TO ch1 AFTER INITIAL ":" BEFORE INITIAL "PRENOM"
contenu de zon après :          "NOM:dE pOnthIEU PRENOM:pierre-marie"
```

NOTES

3. STRING/UNSTRING.

3.1. STRING.

L'instruction STRING permet de concaténer plusieurs données en une seule. Une instruction STRING peut remplacer une série d'instructions MOVE.

Format :

```
STRING      nom_de_donnée1              nom_de_donnée2              …
            littéral1                   litteral2
        DELIMITED BY                    nom_de_donnée3
                                        littéral3
                                        SIZE
            nom_de_donnée4              nom_de_donnée5              …
            littéral5                   litteral5
        DELIMITED BY                    nom_de_donnée6
                                        littéral6
                                        SIZE
            INTO nom_de_donnée7
                 [WITH POINTER          nom_de_donnée8]
                 [ON OVERFLOW           instruction_impérative]
[END-STRING]
```

NOTES

Option DELIMITED BY : détermine les limites de la chaîne de caractères à transférer

- *nom_de_donnée3* ou *littéral3* sont les délimiteurs, c'est à dire le ou les caractère(s) qui délimite(nt) la donnée.
- SIZE indique que toute la zone est à transférer.

La zone réceptrice *donnée7* doit être une donnée alphanumérique sans symbole d'édition et sans clause JUSTIFIED.

Option WITH POINTER : donnée8 doit être une zone numérique entière dont la taille lui permet de recevoir un nombre égal à la longueur de la zone réceptrice + 1 ; elle doit être initialisée à une valeur positive qui indique la position de début de la concaténation dans la zone réceptrice ; après l'exécution du STRING, cette zone est incrémentée du nombre de caractères concaténés ; autrement dit, ce compteur indique le premier caractère non écrasé de la zone réceptrice. Si la longueur cumulée des chaînes à concaténer égale ou excède la longueur de la zone réceptrice, la valeur du compteur est égale à la longueur de la zone réceptrice + 1 (Cf. exemples ci-après).

Option ON OVERFLOW : cette option ne peut être utilisée qu'avec l'option WITH POINTER.

instruction_impérative est exécutée quand la valeur de donnée8 est :

- inférieure à 1
- supérieure à la longueur de la zone réceptrice

NOTES

TRANSPARENT 04

Exemple :

```
1  zon1 PIC x(20)       VALUE       "j'*aime les filles".
1  zon2 PIC x(20)       VALUE       "ado*ado*adolescent".
1  zon3 PIC x(8)        VALUE       "re le CO".
1  zon4 PIC x(3)        VALUE       "BOL".
1  zon5 PIC x(25)       VALUE       ALL "$".
1  cpt          PIC 99   VALUE      5.
STRING          zon1 zon2           DELIMITED BY "*"
                zon3 zon4           DELIMITED BY SIZE
        INTO    zon5
                WITH cpt
                ON OVERFLOW DISPLAY "erreur string - troncature probable"
END-STRING
contenu de zon5 après  :            "$$$$$j'adore le COBOL$$$$"
                                          cpt = 21
```

Remarque :

- contrairement à MOVE, STRING n'implique pas la modification complète de la zone réceptrice.

NOTES

3.2. UNSTRING

Comme son nom l'indique UNSTRING est l'instruction inverse de STRING, c'est à dire qu'elle permet d'éclater le contenu d'une zone en plusieurs zones.

Format :

```
UNSTRING    nom_de_donnée1
      [DELIMITED BY              [ALL]        nom_de_donnée2
                                              littéral2
                         [OR     [ALL]        nom_de_donnée3]      ]
                                              littéral3
            INTO                             nom_de_donnée4
                         [DELIMITER IN        nom_de_donnée5]
                         [COUNT IN            nom_de_donnée6]
                   [nom_de_donnée7
                         [DELIMITER IN        nom_de_donnée8]
                         [COUNT IN            nom_de_donnée9]] ….

                   [WITH POINTER              nom_de_donnée10]
                   [TALLYING IN               nom_de_donnée11]
                   [ON OVERFLOW               instruction_impérative]
[END-UNSTRING]
```

NOTES

- *donnée1* doit être une donnée alphanumérique

- option DELIMITED BY : cette option spécifie les délimiteurs.

Les options DELIMITER IN et COUNT IN ne peuvent être utilisées que si l'option DELIMITED BY est spécifiée.

Les délimiteurs (*nom_de_donnée2* ou *littéral2*, *nom_de_donnée3* ou *littéral3*, … etc …) doivent être alphanumériques. Les constantes figuratives peuvent être employées à l'exception de ALL, elles sont alors considérées comme **un** caractère non numérique.

ALL signifie qu'une ou plusieurs occurrences consécutives du délimiteur sera considérée comme une seule occurrence. Quand DELIMITED BY ALL n'est pas spécifié et qu'il y a plusieurs occurrences consécutives du délimiteur, la zone réceptrice est alors remplie de zéros ou d'espaces selon sa description.

OR permet de définir plusieurs délimiteurs.

- si l'option DELIMITED BY n'est pas spécifiée, le nombre de caractères transférés de la zone émettrice dans la zone réceptrice courante est égal à la longueur de la zone réceptrice.
- option INTO : donnée4, donnée7, …, sont les zones réceptrices. Elles doivent être définies en USAGE DISPLAY et peuvent être de classe quelconque.

NOTES

DELIMITER IN spécifie une zone réceptrice du délimiteur.

COUNT IN spécifie un compteur recevant le nombre de caractères examinés (qui peut être différent du nombre de caractères transférés en cas de troncature) de la zone émettrice.

- option WITH POINTER : spécifie un compteur qui indique une position de départ dans la zone émettrice.
- option TALLYING : spécifie un compteur
- option ON OVERFLOW : il y a exécution de *instruction_impérative* quand :

1) la valeur de POINTER est inférieure à 1 ou supérieure à la longueur de la zone émettrice

2) toutes les données réceptrices ont été utilisées et il reste des caractères à examiner dans la donnée émettrice.

Remarque :

- UNSTRING suit les règles du MOVE.

Exemples :

```
1)      1 zon1          PIC X(12)       VALUE       "012345678901".
        1 zon2          PIC X(2).
        1 zon3          PIC X(6).
        1 zon4          PIC X(8) VALUE ALL "Z".
        UNSTRING zon1 INTO zon2 zon3 zon4
                ON OVERFLOW DISPLAY "OVERFLOW UNSTRING"
        END-UNSTRING
Résultats :             zon2= "01"    zon3= "234567"      zon4 = "8901    "

2)      même exemple avec une autre définition de zon4
        1 zon4          PIC 9(8) VALUE 99999999.
Résultats :             zon2= "01"    zon3= "234567"      zon4 = "00008901"
```

NOTES

TRANSPARENT 05

1)
```
1 zon1          PIC X(30)      VALUE        "JEAN DUPONTEAU**MARIE    1234".
1 zon2          PIC X(5).
1 zon3          PIC X(5).
1 zon4          PIC X(5).
1 zon5          PIC X(5).
1 zon6          PIC 9(10).
1       VALUE ZERO.
        5 cpt1          PIC 99.
        5 cpt2          PIC 99.
        5 cpt3          PIC 99.
        5 cpt4          PIC 99.
        5 cpt5          PIC 99.
        5 cpt6          PIC 99.
    MOVE 1 TO cpt1
    UNSTRING zon1
        DELIMITED BY "*" OR ALL SPACE
        INTO
            zon2 COUNT IN cpt2
            zon3 COUNT IN cpt3
            zon4 COUNT IN cpt4
            zon5 COUNT IN cpt5
            zon6 COUNT IN cpt6
        WITH POINTER cpt1
        ON OVERFLOW DISPLAY "OVERFLOW UNSTRING"
    END-UNSTRING
```

Résultats :

zon2 = "JEAN"	cpt2 = 04
zon3 = "DUPON"	cpt3 = 9
zon4 = " "	cpt4 = 00
zon5 = "MARIE"	cpt5 = 05
zon6 = 0000001234	cpt6 = 04
cpt1 = 31	

Quels seront les résultats si cpt1 est initialisé à 6, puis à 5 ???

NOTES

TP5A

1ere PHASE

A partir de l'instruction INSPECT et de la zone

1 ZXI1 PIC X(13) VALUE « DEVELOPPEMENT ».

on vous demande :

- de compter tous les «E »
- de compter les «O» avant le premier «P»
- de compter les caractères après le deuxième «P»
- de remplacer tous les «E» par «*» après «PP»
- de convertir «AEIOU» en «aeiou».

2eme PHASE

A partir du libellé :

« MONSIEUR PATELIN JOSEPH 40 RUE TURBIGO 75004 PARIS », charger les zones de la description ci-dessous :

```
1 z1.
5 zon-nom1.
      10 civ1          pic x(8).
      10               pic x.
      10 nom1          pic x(10).
      10               pic x.
      10 prnom1        pic x(10).
      10               pic x.
5 zon-rue1.
      10 num1          pic 999.
      10               pic x.
      10 rue1.
          15 voie          pic x(4).
          15 nom-rue       pic x(16).
      10               pic x.
5 zon-ville1.
10 cpost1              pic 9(5).
      10                  pic x.
      10 vil1             pic x(20).
```

En utilisant

- dans un premier temps les instructions INSPECT et l'adressage par référence-modification
- puis l'instruction UNSTRING

Prévoir l'affichage des résultats pour contrôle.

3eme PHASE

A partir des zones

- 1 ZINTIT PIC X(18) VALUE « NOUS SOMMES LE -A ».
- 1 ZDATE PIC X(6).
- 1 ZTIME PIC X(8).

afficher par l'utilisation d'ACCEPT et de STRING la date et l'heure système dans

- 1 ZAFFI PIC X(41) VALUE
« AAAAAAAAAAAAAAAAAAA**********AAAAA ».
la date, sous la forme JJMMAA, et l'heure.

Compléments

1. CLAUSE COPY

COPY permet d'insérer dans le source d'un programme n'importe quelle partie de texte COBOL préécrit et stocké en bibliothèque. COPY est principalement utilisé pour des descriptions de données.

Les parties de texte sont effectivement insérées au moment de la compilation avant la phase de vérification syntaxique.

Format :

```
COPY nom_de_texte    [OF ou IN nom_de_bibliothèque]
        [REPLACING opérande1        BY      opérande2]
```

- *nom_de_texte* est le nom du membre de la bibliothèque contenant le texte à copier.

- *nom_de_bibliothèque* doit être précisé si des membres de même nom se trouvent dans les bibliothèques en ligne à la compilation. Sur gros système, il est préférable de préciser un ordre de recherche dans le J.C.L.

- option REPLACING : permet de remplacer une partie du texte du membre au moment de la copie.

opérande1 et *opérande2* peuvent être du pseudo-texte, un nom de donnée, un littéral, ou un mot COBOL.

Le pseudo-texte est une suite de chaînes de caractères comprenant éventuellement des caractères de ponctuation et bornée par les délimiteurs de pseudo-texte (==). Les chaînes de caractères du pseudo-texte doivent être des mots COBOL entiers ; par exemple, un préfixe de donnée ne peut être remplacé par un pseudo-texte.

NOTES

2. LES FORMATS DE CALCUL.

Rappelons que c'est la clause COBOL USAGE qui spécifie le format d'une donnée à l'intérieur de la mémoire et que l'USAGE par défaut est l'USAGE DISPLAY où 1 caractère est représenté sur 1 octet. L'USAGE peut être spécifié pour une donnée de niveau groupe ; cet USAGE s'appliquera alors à toutes les données élémentaires du groupe et ne doit pas être contredit.

L'USAGE DISPLAY convient pour toutes les données qui ne font pas l'objet d'un calcul. En effet, un ordinateur n'effectue pas ses opérations en format caractère, mais dans un format qui lui est propre. En conséquence, si des données numériques sur lesquelles on veut effectuer des calculs sont définies en USAGE DISPLAY, il y aura, à l'exécution, des opérations de conversion avant et après le calcul, ce qui peut entraîner une perte sensible de performance.

Pour les données numériques, COBOL ANS 85 propose deux formats dits "computationnal", qu'on pourrait traduire par formats de calcul. Ce sont les USAGES BINARY et PACKED-DECIMAL.

Rappelons que la PICTURE d'une donnée indique le nombre de signes que l'on veut représenter et non le nombre d'octets à réserver ; cette taille, différente selon les formats, peut toujours se calculer.

La conversion d'un format à un autre se fait de façon transparente, c'est le compilateur qui génère les instructions de conversion. Il est évidemment souhaitable que les différents opérandes d'un calcul soient de même format.

NOTES

2.1. USAGE BINARY.

Le terme BINARY a été introduit dans la version ANS 85 dans un but de normalisation. En effet, souvent nommé COMP, le nom du format binaire pouvait être différent selon les compilateurs.

Une zone définie en BINARY contient du binaire pur ; c'est le format le plus proche de la machine, donc le plus performant pour les calculs. Son inconvénient est d'être difficilement lisible dans un fichier.

Format :

```
1 zonum          PIC 9(n)      [USAGE] BINARY.
```

Toutes les clauses vues précédemment pour les données numériques en format caractère sont possibles.

Nombre de chiffres à représenter			Taille de la zone
1	à	4	2 octets
5	à	9	4 octets
10	à	18	8 octets

Les nombres négatifs sont traités sous forme de complément à 2.

Le signe occupe le bit le plus à gauche.

NOTES

2.2. USAGE PACKED-DECIMAL.

Comme BINARY, le terme PACKED-DECIMAL, introduit dans la version ANS 85, était souvent appelé COMP-3 sur les compilateurs des gros systèmes (COMP et COMP-3 sont toujours reconnus pour la compatibilité, mais il est préférable d'utiliser les nouveaux termes pour des raisons de portabilité).

Il s'agit d'un format intermédiaire entre le format caractère et le binaire pur ; plus lisible, il nécessite des opérations de conversion pour les calculs et occupe le plus souvent une place supérieure à celle du format binaire. Le principe est de supprimer, par rapport à l'usage DISPLAY, le demi-octet inutile, en conservant le signe (F, C ou D) qui est codé en général sur le dernier demi-octet de droite.

En représentation hexadécimale du code EBCDIC, les octets représentant des chiffres commencent par "F".

Par exemple :

527 en format caractère est codé	:F5:F2:F7	sur 3 octets
527 en format packé sera codé	:52:7F	sur 2 octets
-6912 en format caractère est codé	:F6:F9:F1:D2	sur 4 octets
-6912 en format packé sera codé	:06:91:2D	sur 3 octets

Format :

```
1 zonum      PIC 9(n)      [USAGE] PACKED-DECIMAL.
```

Toutes les clauses vues précédemment pour les données numériques en format caractère sont possibles.

La taille de la zone réservée peut être calculée par la formule suivante :

- nombre d'octets réservés = (nombre de chiffres à représenter / 2) + 1

1. EXERCICE DE SYNTHÈSE.

TP6A

Un club de tennis possède un fichier de ses membres. Une fois par an, un traitement par lot a lieu pour actualiser le fichier et sortir une liste des compétiteurs par catégorie d'âge. Il vous est demandé d'écrire ce programme.

Fichiers en entrée :

1) le fichier des membres du club : c:\cobol\tpbase\anctenn.dat
2) le fichier des mouvements : c:\cobol\tpbase\mvttenn.dat

Les deux fichiers sont triés en ordre croissant sur le numéro de licence.

Fichier en sortie :

1) le fichier des membres du club mis à jour : c:\cobol\tpbase\noutenn.dat

Tous les fichiers ont la même structure ; la description COBOL est faite dans le membre c:\cobol\tpbase\fictenn.cpy.

Structure des fichiers :

Enregistrements de longueur fixe 160.

LONGUEUR	CLASSE	DESCRIPTION	VALEURS
7	9	numéro de licence	
30	X	nom	
20	X	prénom	
1	X	sexe	M ou F
8	9	date de naissance	JJMMAAAA
4	X	classement	
1	X	compétition	O ou N
1	X	catégorie d'âge	J, M, C, S, +, V
70	X	adresse (*)	
8	9	numéro de téléphone	
4	9	cotisation	0 ou montant acquitté
6	X	vide	

* voir la clause COPY pour la décomposition

2. TRAITEMENT À EFFECTUER.

Phase 1 : mise à jour du fichier des membres, affichage d'un bilan final.

1) Suppression : seul le numéro de licence de l'article du fichier mouvement est servi, le reste de l'article est saturé de zéros.

2) Création : tous les renseignements sont servis à l'exception de la catégorie d'âge, la cotisation est obligatoirement acquittée.

3) Mise à jour : le fichier mouvement contient 1 article par licencié, seules les zones à mettre à jour sont servies, les autres étant saturées d'espaces.

- liste des informations pouvant être mises à jour : nom, prénom, classement, compétition, adresse (l'adresse complète est alors servie), numéro de téléphone, cotisation.
- dans tous les cas, la cotisation doit être mise à jour.
- si la cotisation est à zéro, il est demandé d'afficher à l'écran le membre à relancer; les renseignements à afficher sont le nom, le prénom et l'adresse.

4) Bilan : afficher à l'écran les compteurs suivants avec les libellés correspondants

1 - nombre de licenciés au club la saison passée
2 - nombre de licenciés au club cette saison
3 - nombre de nouveaux licenciés
4 - nombre de départs
5 - nombre de compétiteurs
6 - nombre de relances

Phase 2 : traitement statistique

A partir du fichier mis à jour, il est demandé de compter et d'afficher le nombre de compétiteurs par sexe et par catégorie d'âge afin de déterminer le nombre d'équipes à engager dans les prochains championnats.