Les données

1. Les verbes COBOL

1.1. Catégorie fonctionnelles de verbes

Verbes arithmétiques : `ADD,SUBTRACT, MULTIPLY, DIVIDE, COMPUTE`

Verbes d'entrée-sortie :`READ, WRITE, REWRITE, DELETE, ACCEPT, DISPLAY, START, OPEN, CLOSE, STOP, USE`

Verbes de rupture de séquence : `PERFORM, EXIT, STOP, GO TO, ALTER`

Verbes de communication inter-programmes : `CALL, CANCEL, EXEC`

Verbes de gestion de table : `SET, SEARCH`

Verbes de manipulation de données : `MOVE, INSPECT, STRING, UNSTRING, SORT, MERGE, RELEASE, RETURN`

Directives de compilation : `COPY, INSERT, REPLACE, USE`

Verbes de condition : `IF, EVALUATE,` les verbes d'entrée-sortie utilisés avec l'une des clauses `AT END` ou `INVALID KEY`, les verbes arithmétiques avec une phrase `SIZE ERROR`, `SEARCH avec WHEN`

1.2. Types d'instructions et de phrases

On appelle **instruction conditionnelle** une instruction identifiée par un verbe de condition ; les autres instructions sont dites **impératives**.

Une **phrase impérative** ne contient que des instructions impératives, sinon elle est de type **conditionnel**.

2. Représentation interne des données

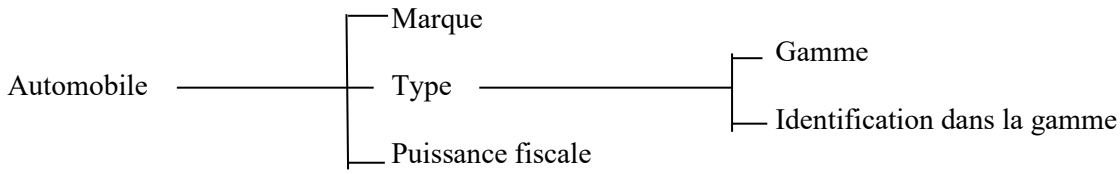
2.1. Caractéristiques d'une donnée

Une information ou **donnée** est la ***valeur*** d'un attribut qui caractérise un objet ou une entité.

Exemple : Si on considère l'objet "une automobile", on peut caractériser cet objet par sa marque, son type, sa puissance (entre autres) ; marque, type et puissance fiscale sont des attributs de "automobile", auxquels on peut donner des valeurs comme RENAULT, CLIO, 6CV.

Un attribut peut être un objet simple (p.ex. la puissance fiscale) ou non ; dans ce cas il peut être à son tour caractérisé par des attributs plus simples.

Exemple : Nous pouvons considérer que le type de la voiture est caractérisé par la gamme de fabrication et par une identification dans la gamme ; nous faisons apparaître deux attributs plus élémentaires distincts : la gamme CLIO et l'identification GL à l'intérieur de cette gamme.



Sur cet exemple, nous avons fait apparaître une structure hiérarchique entre les données. L'objet qui se trouve au sommet de la hiérarchie s'appelle une **entité de base**.

STRUCTURER des données consiste donc à regrouper ces données autour d'une entité de base en les classant suivant une hiérarchie.

Nous appelons **ARTICLE** (ou enregistrement logique) une telle structure de données.

Nous appelons **RUBRIQUE** la zone mémoire permettant de stocker la valeur d'un attribut.

En nous référant au schéma précédent, nous constatons que nous aurons deux sortes de rubriques :

- des **rubriques élémentaires** qui ne contiennent qu'une seule valeur non décomposée (ex. Marque, Gamme, etc..)

- des **rubriques structurées** : elles regroupent un ensemble quelconque de rubriques, élémentaires ou non (ex. Type)

Remarques :
a) La rubrique structurée la plus importante est celle qui est constituée de l'ensemble des rubriques relatives à un article.

b) La **décomposition** en rubriques d'une entité **dépend des fonctions à réaliser** et non des sous-attributs élémentaires qu'on peut mettre en évidence ; ainsi dans l'exemple précédent, il est inutile de décomposer l'attribut "type" en deux sous-attributs si aucun traitement n'utilise ces sous-attributs.

2.2. La hiérarchie des données en COBOL

La description de l'objet "automobile" se présente sous la forme d'une arborescence. Pour décrire cette arborescence de façon linéaire, on attribue à chaque niveau de la décomposition un **NUMERO** et un **ordre** d'écriture tels que pour tout attribut J dans un attribut I :

a) J est écrit après I

b) le numéro de J est strictement supérieur à celui de I

c) tout élément écrit entre I et J a un numéro strictement supérieur à celui de I.

```
01      AUTOMOBILE
            02      MARQUE
            02      TYPE
                    03      GAMME
                    03      IDENT-DANS-GAMME
            02      PUISSANCE
```

Les numéros de niveaux doivent être compris entre 01 (ou 1) et 49 ; ils ne sont **pas obligatoirement consécutifs** ; l'entité de base porte toujours le numéro 1.

2.3. Définition d'une rubrique

Chaque rubrique est identifiée par un nom qui doit être unique (cf. noms de données au chapitre 1).

Les rubriques élémentaires font l'objet d'une description plus détaillée à l'aide de clauses qui précisent leur nature, leur format, etc... et permettent d'avoir une **image théorique de l'attribut**. Ces clauses sont décrites au chapitre 3. Toutefois, notons dès à présent que l'on représente :

- par le symbole **X** une position de rubrique pouvant contenir un caractère quelconque
- par le symbole **9** une position de rubrique ne pouvant contenir qu'un chiffre.

Exemples : `MARQUE XXXXXXX`
qui s'écrit aussi `MARQUE X(7)`
(7 est un facteur de répétition)
La valeur Marque peut contenir une valeur sous forme d'une chaîne de 1 à 7 caractères.

`PUISSANCE 99`
La rubrique PUISSANCE peut contenir une valeur sous forme d'un nombre entier de 2 chiffres, comme 04.

2.4. Types et classes de rubriques

On distingue **cinq types** de rubriques :

- **alphabétique** : la rubrique ne contient que des lettres ou des espaces
- **numérique** : la rubrique ne contient que des chiffres
- **numérique édité** : la rubrique ne contient que des chiffres et des symboles d'édition
- (. , ~~b~~ , + , - $, etc...)
- **alphanumérique** : la rubrique contient des caractères quelconques
- **alphanumérique édité** : la rubrique contient des caractères quelconques et des symboles d'insertion (~~b~~ , /)

On distingue **trois classes** de rubriques :

- alphanumérique
- numérique
- alphanumérique (constitué des 3 derniers types)

Remarque très importante :
Une rubrique structurée est toujours de classe alphanumérique, quel que soit son type.

2.5. Formats internes des données

➲ ***Tous les exemples sont donnés en ASCII***

2.5.1. Formats fondamentaux

On distingue **deux représentations** de base :

◆ **le format binaire**, ne concerne que les **entiers**. La valeur est codée en complément à 2, sur 2, 4ou 8 octets. Les plages de valeurs sont :
-32768 à + 32767 pour la représentation sur 16 bits
$-2^{31}-1$ à $+2^{31}$ ($\sim 10^9$) pour la représentation sur 32 bits
$-2^{63}-1$ à $+2^{63}$

Désigné en COBOL par `BINARY, COMPUTATIONAL,` ou `COMPUTATIONAL-4`

◆ **la représentation codée**, peut être utilisée pour tous les types ; les valeurs sont codées en utilisant le code ASCII ou EBCDIC. (cf. chap. 1 par 4-1)

Une troisième représentation peut exister : la **virgule flottante**, simple (4 octets) ou double (8 octets).

Désigné par `COMPUTATIONAL-1, COMPUTATIONAL-2.`

2.5.2 Représentation codée

2.5.2.1. Rubrique de type non numérique

Chaque caractère de la rubrique occupe exactement un octet ; sa valeur est représentée par son code ASCII

Exemple : Soit la rubrique MARQUE d'image `X(7)` ; la valeur PEUGEOT est représentée (en hexadécimal)

50	45	55	47	45	4F	54

7 octets

Désigné par `DISPLAY`.

2.5.2.2. Rubriques de type numérique

On distingue deux représentations.

2.5.2.2.1 La représentation décimale étendue (`DISPLAY`)

Chaque chiffre occupe un octet et est représenté par son code ASCII ou EBCDIC

Exemple : Soit la rubrique `PUISSANCE` d'image `99` ; la valeur 06 est représentée (en hexadécimal, ASCII)

30	36

2 octets

Si le nombre est signé, il y a deux manières de préciser le signe :

a) Le signe occupe une position propre
Sa valeur est donnée par le code correspondant au caractère + (soit 2B) ou - (soit 2D) ; elle est placée dans une position séparée soit en tête de zone, soit en fin de zone ; ce choix est fixé par le programmeur.

Exemple :
Soit la rubrique `MONTANT` d'image `S9(5)`, (le symbole S indique une rubrique signée) avec **signe séparé en tête** ; la valeur + 953 est représentée :

2B	30	30	39	35	33

6 octets : 5 chiffres + le signe

En prenant la même rubrique, mais avec une déclaration de **signe séparé en fin**, nous aurions obtenu

30	30	39	35	33	2B

b) Le signe n'occupe pas de position séparée

Dans ce cas, la codification du signe est "superposée", à celle du chiffre de plus faible ou de plus fort poids (par analogie avec les cartes perforées, on parle de "sur-perforation").

Le tableau suivant donne les résultats de signe des chiffres perforés en hors texte

Chiffre de faible poids	Valeurs signées positives		Valeurs signées négatives	
	caractère	code ASCII	caractère	code ASCII
0	{	7B	}	7D
1	A	41	J	4A
2	B	42	K	4B
3	C	43	L	4C
4	D	44	M	4D
5	E	45	N	4E
6	F	46	O	4F
7	G	47	P	50
8	H	48	Q	51
9	I	49	R	52

Exemple : Soit la rubrique MONTANT d'image S9(5) (l'absence de spécification entraîne "signe superposé"); la valeur + 953 est représentée :

30	30	39	35	43

5 octets

On remarque que la valeur du dernier octet est identique au code ASCII de la lettre C.

Remarques : Utiliser de préférence la forme (a) car le système s'y ramène dans les opérations arithmétiques.

2.5.2.2.2 La représentation décimale condensée

(`PACKED-DECIMAL ou COMPUTATIONAL-3`**)**

Pour gagner de la place sur les supports et accélérer les traitements, il peut être utile de placer deux codes numériques par octet ; en effet, le quartet de gauche dit hors-texte, est identique pour tous les chiffres et vaut '3' en ASCII et 'F' en EBCDIC.

Exemple : Soit la rubrique MONTANT d'image `9(5)` avec représentation décimale condensée ; la valeur 953 est représentée

00	95	3F

"F" pour non signée

3 octets

Le premier chiffre est rajouté par le système pour compléter le premier octet ; il peut être quelconque et n'intervient pas dans les traitements ; seuls les 5 chiffres de droite, dûment déclarés, seront pris en compte (sauf cas particuliers).

Si la rubrique est signée, la codification du signe occupe le dernier demi-octet de la zone ; elle vaut 'C' pour + et 'D' pour -.

Exemple : Soit la rubrique MONTANT d'image S9(5) avec représentation décimale condensée.

La valeur + 953 est représentée :

00 95 3C

3 octets

A noter que sur cet exemple le nombre de positions est pair et donne un nombre entier d'octets.

Remarques :

- Une rubrique de type numérique, non signée, sera toujours considérée comme contenant un nombre positif, quelle que soit la représentation utilisée.

- Sur la plupart des ordinateurs on dispose d'instructions machine permettant d'effectuer des opérations arithmétiques sur des nombres représentés en format binaire ou en format décimal condensé ; les nombres en format décimal étendu doivent être d'abord être convertis dans l'un des deux formats précédents.

2.5.3 Longueur d'une rubrique :

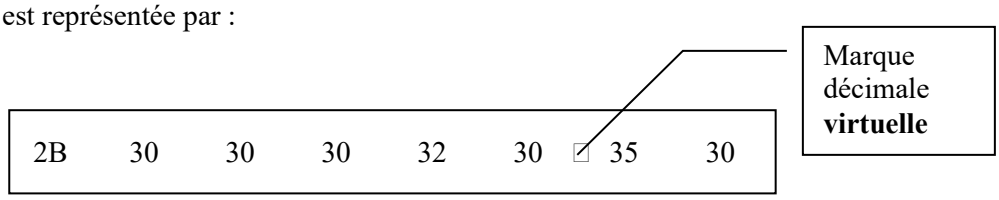
Elle dépend de la représentation interne choisie et de l'image

- Format binaire --> Longueur : 4 octets
- Rubrique de type non numérique, N symboles indiqués dans l'image --> longueur : N octets
- Représentation décimale étendue, sans signe ou avec signe superposé,
  N chiffres indiqués --> Longueur N octets
- Représentation décimale étendue, signe séparé, N chiffres indiqués --> Longueur : N + 1 octets
- Représentation décimale condensée, N symboles indiqués (y compris le symbole S si la rubrique est signée)
  Si N est pair --> Longueur : N/2 octets
  Si N est impair --> Longueur : (N + 1)/ 2 octets

2.5.4 Le séparateur décimal :

Pour séparer la partie entière de la partie fractionnaire d'un nombre on utilise le point (cf Littéraux numériques au chap. 1 § 4.2.1.). La représentation de ce point au niveau du format interne d'une donnée numérique n'est pas utile. Il suffit d'indiquer au système la **position virtuelle** de ce séparateur ; ceci se fait dans la description de l'image de la rubrique à l'aide du symbole `V`.

Exemple :
Soit la rubrique MONTANT d'image `S9(5)V99`, en représentation décimale étendue avec signe séparé en tête ; la valeur + 20.5 est représentée par :



8 octets (dont aucun pour la marque décimale)

Remarques :

- Le symbole V est **unique** au niveau d'une description d'une image ; s'il est omis, il est supposé être à droite et l'image correspond alors à celle d'un nombre entier.
- Si la rubrique doit être visualisée (sur écran ou sur papier), il est nécessaire de faire figurer le point au niveau de l'affichage ; pour cela on utilise des rubriques spéciales de type numérique édité étudiées ultérieurement.

3. Règles d'alignement

Ces règles sont appliquées chaque fois que l'on mémorise une valeur dans une rubrique. Le choix de l'une des deux règles dépend du type de la rubrique élémentaire qui "reçoit" la valeur.

3.1. Règle d'alignement numérique

Elle s'applique si la rubrique **réceptrice** est de type **numérique** ou **numérique éditée**. La valeur est alignée sur la marque décimale ; si la zone est trop grande, elle est complète vers ses extrémités par des zéros ; si elle est trop petite, la valeur est tronquée à l'une ou l'autre des extrémités.

Exemple :
Soit la rubrique R1 d'image `9(2)V9` ; indépendamment de la représentation interne, elle peut contenir un nombre de 3 chiffres dont 2 pour la partie entière.

Valeur transférée	Valeur obtenue

21	21.0
1.5	01.5
1.67	01.6
462.1	62.1
520.02	20.0

3.2. Règle d'alignement non numérique :

Elle s'applique si la rubrique réceptrice est de type **alphabétique** ou **alphanumérique** ou **alphanumérique édité**. La valeur est alignée sur la position d'extrême gauche de la rubrique réceptrice ; si la zone est trop grande, la valeur est complétée vers l'extrémité droite par des caractères "espaces" ; si elle est trop petite, la valeur est tronquée à droite.

Exemple :

Soit la rubrique R2 d'image `X(5)`

Valeur Transférée	Valeur obtenue
"VILLES"	"VILLE"
"BLANC"	"BLANC"
"BEAU"	"BEAU"

4. Unicité de référence

Un même mot-utilisateur pet être utilisé pour désigner des données différentes, sous réserve qu'ellea appartiennent à des groupes différents. Pour les référencer sans ambigüité, on **qualifiera** leur nom par celui du groupe, ou du fichier les contenant. La même disposition s'applique aux paragraphes de sections différentes et aux textes-source de bibliothèques différentes.

nd-1 { { IN / OF } nd-2 } ... [{ { IN / OF } fich-1]

Exemple

```
01 IDENTITE.
      02  NOM          PIC X(20).
      02  PRENOM       PIC X(15).

01 INDIV.
      02  NOM          PIC X(20).
      02  PRENOM       PIC X(15).
.
.
      MOVE NOM OF IDENTITE TO NOM OF INDIV.
```

5. Règles de ponctuation

4.1. Un ou plusieurs **espaces** consécutifs servent à séparer deux mots.

4.2. La **virgule**, le **point-virgule** et le **point**, utilisés comme séparateurs, doivent être suivis d'un espace et ne peuvent être utilisés que lorsque le format des instructions l'autorise.

4.3. Les **parenthèses** s'utilisent toujours par paires.

4.4. Les **guillemets** délimitent les littéraux non numériques :
- le guillemet "ouvrant" doit être précédé d'un espace
- le guillemet "fermant" doit être suivi d'un espace, d'une virgule, d'un point-virgule ou d'un point.

Les guillemets s'utilisent par paires (sauf lorsqu'un littéral s'écrit sur 2 lignes : cf § 5)

6. Feuille de saisie COBOL

Elle comprend cinq zones, respectivement de gauche à droite :
- zone de numérotation (col. 1 à 6) (inutilisée, ou de contenu quelconque)
- zone indicateur (col. 7)
- zone A (col. 8 à 11) ; la colonne 8 est dite marge A
- zone B (col. 12 à 72) ; la colonne 11 est dite marge B
- zone d'identification (col. 73 à 80) (inutilisée)

Les **en-têtes** de division, section, paragraphes et de description de fichier commencent en zone A. Les instructions et clauses s'écrivent en zone B. Une instruction peut être rédigée sur plusieurs lignes. Il est possible d'inclure des lignes totalement vierges à tout endroit du programme.

La zone indicateur (col. 7) assure trois fonctions selon le caractère qui y est écrit :

- **continuation** d'un littéral non numérique portant jusqu'à la col.72 de la ligne précédente

***** ligne **commentaire** : Le contenu de cette ligne ne sera pas pris en compte par le compilateur ; cette ligne ne sert qu'à améliorer la lisibilité et la compréhension du programme.

/ idem * ; de plus saut de page avant l'impression du commentaire

D ou **d** instrument de **mise au point** : Cette fonction sera étudiée plus tard.

Colonne	0 1 2 3 4 5 6 7 8 123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
Marge	A B
	* CECI EST UNE LIGNE DE COMMENTAIRES
	77 Exemple pic x(75) value "ce littéral non-numérique va jusqu'en colonne72
	- " et se prolonge par la suite"

Exemple de programme COBOL

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. PUBLI.

 AUTHOR. GR.
***------------------------------------------------

 ENVIRONMENT DIVISION.

 CONFIGURATION SECTION.
 SOURCE-COMPUTER. MWB.
 OBJECT-COMPUTER. MWB.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.

           SELECT BASE ASSIGN TO BASABON
               ORGANIZATION RECORD SEQUENTIAL
               ACCESS SEQUENTIAL
               FILE STATUS IS BS.

           SELECT MOUV ASSIGN TO MAJABON
               ORGANIZATION RECORD SEQUENTIAL
               ACCESS SEQUENTIAL
               FILE STATUS IS MS.

           SELECT NOUBASE ASSIGN TO NEWABON
               ORGANIZATION RECORD SEQUENTIAL
               ACCESS SEQUENTIAL
               FILE STATUS IS NS.

           SELECT ERREUR ASSIGN TO ERRABON
               ORGANIZATION LINE SEQUENTIAL
               ACCESS SEQUENTIAL
               FILE STATUS IS ES.
***------------------------------------------------

 DATA DIVISION.

 FILE SECTION.

*-----------------------------------------------------------------
 FD  BASE
            LABEL RECORD STANDARD.
 01  B-ENR.
* Code-adresse
      05 B-CD-ADR             PIC 9(6).
* Cle de controle
      05 B-CLE                PIC X.
* Nom-Prenom
      05 B-NM-PRNM            PIC X(25).
 * Adresse
      05 B-ADR                PIC X(40).
* Code Postal
```

```
      05 B-CD-PSTL            PIC 9(5).
* Ville
      05 B-VL                 PIC X(25).

*-----------------------------------------------------------------
 FD  MOUV
           LABEL RECORD STANDARD.
 01  M-ENR.
      05 M-CD-OPR             PIC X.
      05 M-DONNEES.
        10 M-CD-ADR           PIC 9(6).
        10 M-CD-ADRX          REDEFINES M-CD-ADR  PIC X(6).
        10 M-CLE              PIC X.
        10 M-NM-PRNM          PIC X(25).
        10 M-ADR              PIC X(40).
        10 M-CD-PSTL          PIC 9(5).
        10 M-VL               PIC X(25).
*-----------------------------------------------------------------
 FD  NOUBASE
      LABEL RECORD STANDARD.
 01  N-ENR                    PIC X(102).

*-----------------------------------------------------------------
 FD  ERREUR
      LABEL RECORD STANDARD.
 01  E-ENR                    PIC X(103).

*-----------------------------------------------------------------
 WORKING-STORAGE SECTION.
* Code etat du fichier BASE
 77  BS                       PIC XX.
* Code etat du fichier MOUV
 77  MS                       PIC XX.
* Code etat du fichier NOUBASE
 77  NS                       PIC XX.
* Code etat du fichier ERREUR
 77  ES                       PIC XX.
* Zone de travail
 77  QUOTIENT                 PIC 9(6) COMP.
* Rang de la lettre de controle
 77  RG-LTR                   PIC 99 COMP.
          88  CORRECTE  VALUE 1 THRU 23.
* Rang du message d'erreur
 77  RG-ERR                   PIC 9.

* Definition des cles de controle
 01  LETTRES      PIC X(23) VALUE "ABCDEFGHJKLMNPQRSTVWXY".
 01  TABCLE       REDEFINES LETTRES.
          05  CLE             PIC X OCCURS 23.

* Definition des messages d'erreur
 01  LST-MSG.
          05  FILLER PIC X(30) VALUE "(1) CLE ERRONEE".
          05  FILLER PIC X(30) VALUE SPACES.
          05  FILLER PIC X(30) VALUE SPACES.
```

```
          05  FILLER PIC X(30) VALUE "(4) CODE OPER. ERRONE".
          05  FILLER PIC X(30) VALUE "(5) DONNEE INCORRECTE".
          05  FILLER PIC X(30) VALUE SPACES.
          05  FILLER PIC X(30) VALUE SPACES.
          05  FILLER PIC X(30) VALUE SPACES.
 01  TABERR       REDEFINES LST-MSG.
          05  MSGERR          PIC X(30) OCCURS 8.

***-------------------------------------------------

 PROCEDURE DIVISION.
 DECLARATIVES.

 LECT SECTION. USE AFTER STANDARD ERROR PROCEDURE ON INPUT.
 L0.
     DISPLAY "Erreur d'entrée-sortie en lecture" UPON CONSOLE
            DISPLAY "Code d'etat des fichiers:" UPON CONSOLE
     DISPLAY "BASABON = " BS "      MAJABON = " MS UPON CONSOLE
            CLOSE MAJABON BASABON NOUBASE ERREUR.
            STOP RUN.

 ECR SECTION. USE AFTER STANDARD ERROR PROCEDURE ON OUTPUT.
 E0.
     DISPLAY "Erreur d'entrée-sortie en ecriture" UPON CONSOLE
            DISPLAY "Code d'etat des fichiers:" UPON CONSOLE
     DISPLAY "NOUBASE = " NS "      ERREUR = " ES UPON CONSOLE
            CLOSE MAJABON BASABON NOUBASE ERREUR.
            STOP RUN.

 END DECLARATIVES.

 PUBLI SECTION.

* Initialisation du traitement
 INIT.
           OPEN    INPUT  BASE MOUV
                   OUTPUT NOUBASE ERREUR.
           PERFORM LIRBAS.
           PERFORM LIRMAJ.

* Traitement majeur
 CORPS.
           PERFORM PRINCIP UNTIL BS NOT = ZERO OR MS NOT = ZERO.
           PERFORM FINBAS UNTIL BS NOT = ZERO.
           PERFORM FINMAJ UNTIL MS NOT = ZERO.

* Finalisation du traitement
 FIN.
           CLOSE BASE MOUV NOUBASE ERREUR.
           DISPLAY "FIN DE PUBLI" UPON CONSOLE.
           STOP RUN.

*-----------------------------------------------------------------
 SPECIALE SECTION.
```

```
* Repetitive principale
 PRINCIP.PERFORM CALCLE. IF  CORRECTE PERFORM TRAIT
                            ELSE PERFORM ERR PERFORM LIRMAJ.

* Traitement final de BASE
 FINBAS.  PERFORM COPIE   PERFORM LIRBAS.

* Traitement final de MOUV
 FINMAJ.  IF M-CD-OPR = "C" PERFORM AJOU
              ELSE MOVE 4 TO RG-ERR  PERFORM ERR.
           PERFORM LIRMAJ.

 LIRBAS.  READ BASE.

 LIRMAJ.  READ MOUV.

* Validation de la cle
 CALCLE.  IF M-CD-ADRX NOT NUMERIC
                           MOVE ZERO TO RG-LTR
                           MOVE 5 TO RG-ERR
             ELSE
               DIVIDE 23 INTO M-CD-ADR     GIVING QUOTIENT
                                           REMAINDER RG-LTR
               SUBTRACT RG-LTR FROM 23 GIVING RG-LTR
               IF CLE(RG-LTR) NOT = M-CLE
                           MOVE ZERO TO RG-LTR
                           MOVE 1 TO RG-ERR.

* Deroulement de BASE
 TRAIT.   IF M-CD-ADR > B-CD-ADR  PERFORM COPIE PERFORM LIRBAS
                ELSE PERFORM NONSUP.

* Ecriture des erreurs
 ERR.     WRITE E-ENR FROM M-ENR. WRITE E-ENR FROM MSGERR(RG-ERR).

* Ecriture de NOUBASE
 COPIE.   WRITE N-ENR FROM B-ENR.

* Ajout d'un mouvement en creation
 AJOU.
*------  Il faudrait controler la validite des rubriques!!
                WRITE N-ENR FROM M-DONNEES.

 NONSUP.  IF M-CD-ADR = B-CD-ADR
               PERFORM EGAL
             ELSE  IF M-CD-OPR = "C"
                   PERFORM AJOU
                        ELSE MOVE 4 TO RG-ERR PERFORM ERR.

          PERFORM LIRMAJ.

* Code-adresse existant dans BASE
 EGAL.    IF M-CD-OPR = "S"
                     NEXT SENTENCE
             ELSE IF M-CD-OPR = "M"
```

```
                        PERFORM MODIF
                        PERFORM COPIE
                            ELSE PERFORM COPIE
                               MOVE 4 TO RG-ERR PERFORM ERR.
           PERFORM LIRBAS.

* Chargement des champs a modifier
 MODIF.    IF M-NM-PRNM NOT = SPACES MOVE M-NM-PRNM TO B-NM-PRNM.
           IF M-ADR     NOT = SPACES MOVE M-ADR     TO B-ADR.
           IF M-CD-PSTL NOT = SPACES MOVE M-CD-PSTL TO B-CD-PSTL.
           IF M-VL      NOT = SPACES MOVE M-VL      TO B-VL.
```