1 Programmation structurée

1.1 Concepts

La programmation structurée est un style de programmation dans lequel la structure du programme est construite de la façon la plus claire que possible.

Cette clarté est due à la restriction des séquences logiques de contrôle aux trois structures suivantes :

la séquence :

En absence du contraire, les instructions sont exécutées selon leurs ordre d'écriture.

L'alternative :

Avec cette structure une décision est faite. Si la réponse à la question est vrai, alors un chemin est suivi; quand la réponse est égale à Faux, le chemin correspondant au sinon est choisi.

La répétitive :

Cette structure est un mécanisme pour la création et le contrôle d'une boucle.

Objectifs

Les principaux objectif**s de la programmation structurée sont les suivants :**

- le développement modulaire du programme;
- la facilité du codage;
- la rapidité du codage;
- la facilité du déboguage;
- la facilité de la maintenance.

En programmation structurée, toute sorte de logique est exprimée par les trois structure : *la séquence*, l'*alternative* et la *répétitive*.

2 Introduction au langage COBOL

Le langage COBOL est né du double souci de:

1. Créer un langage adapté aux problèmes de gestion et de traitement de données.
2. Rendre ce langage aussi indépendant que possible du matériel sur lequel doivent être exécutés les programmes.

- En 1960, le comité CODASYL (**CO**nference on **DA**ta **SY**stems **L**anguages) réuni à l'initiative de la défense nationale américaine, a défini les spécifications d'un langage commun de traitement de problèmes de gestion: **CO**mmon **B**usiness **O**riented Language; d'où **COBOL-60**.

Caractéristiques

commun: à tous les constructeurs d'ordinateurs

modulaire:modules de fonctionnement (division,section)

3. Paragraphes

proche du langage courant: les mots et les phrases, en langue anglaise, pouvant être compris par un non informaticien.

Exemple

```
MULTIPLY prix-unitaire BY quantite GIVING total
```

Standardisation

L'un des objectifs des efforts de standardisation vise la création d'un langage portable sur la plupart des systèmes existants (1974, 1983, 1985) par l'American National Standards Institute (ANSI)

3 STRUCTURE D'UN PROGRAMME COBOL

Un programme COBOL peut être considéré comme un ensemble hiérarchisé d'éléments. En décomposant cet ensemble, en partant du niveau le plus élevé, on retrouve:

1. Quatre (4) divisions:
   - IDENTIFICATION DIVISION
   - ENVIRONMENT DIVISION
   - DATA DIVISION
   - PROCEDURE DIVISION
2. Des sections

1. IDENTIFICATION DIVISION

C'est la première division d'un programme COBOL. Elle permet d'identifier:

- le nom de programme
- l'auteur
- la date d'écriture et de compilation

2. ENVIRONMENT DIVISION

Cette division définit l'environnement matériel dans lequel le programme tournera.

- sur quelle machine le programme sera compilé et exécuté.
- définit les fichiers utilisés dans le programme.
  . noms des fichiers
  . organisations

3. DATA DIVISION

Description détaillée des fichiers utilisés:

. le nom du fichier;

. enregistrements, champs, etc.

ainsi que les variables (simples et composées) manipulées dans le programme.

4. PROCEDURE DIVISION

Dans cette division on retrouve les instructions du programme.

4 Syntaxe d'un programme COBOL

4.1 Alphabet lexicographique

L'alphabet lexicographique de COBOL est constitué des éléments suivants:

- Les caractères alphanumériques:

. les chiffres: 0,1,2,....,9

. les lettres: A, B, C, D,..., Z et l'espace (blanc)

- Les caractères de ponctuation :

. virgule (,), pt. virgule (;), point (.), cote (')

. double cote ("), parenthèse dte et gche) (.

- Les caractères de comparaison :

. supérieur (>), inférieur (<), égal (=)

- Les caractères arithmétiques

. +, -, /, *, $

4.2 Les mots COBOL

Un mot COBOL est une suite de 30 caractères au plus.

Chaque caractère est choisi dans l'ensemble des chiffres (0..9), des lettres (A..Z) et du tiret (-)

(le tiret ne peut être ni le premier ni le dernier caractère d'un mot).

Les mots usuels: définis par le programmeur

Exemple :

CHAINE-DE-CARACTERES
000-debut
1000-traitement
9999-fin
Var-1
Var-2
1000 (nom d'un paragraphe)
mais-ceci-n-est-pas-un-mot-valide-pourquoi

Les mots réservés: mot COBOL appartenant à une liste précise, pouvant être utilisé dans le programme, mais n'étant pas défini par le programmeur lui même.

Exemple :

Les nom des divisions et des sections (**PROCEDURE, SECTION, IDENTIFICATION, DATA**, etc..) .

Les mots clés (**SELECT, FD, EQUAL, MOVE, PERFORM**, etc..).

5 Règles d'écriture d'un programme COBOL

Pour pouvoir être soumis à un ordinateur, un programme COBOL doit être décrit en respectant un certain format. Ce format, matérialisé en général par un bordereau de codification, servira de guide pour la rédaction du programme.

La plupart des compilateurs COBOL utilisent des lignes de **80 caractères**, une tradition qui date depuis l'ère des cartes perforées.

La colonne 7 est appelée indicateur. Elle indique par un "*" ou "-" que la ligne a une interprétation spéciale.

- ***** : à la colonne 7 indique que c'est une ligne de commentaire.
- \- : Si la ligne précédente s'arrête au milieu d'un mot, le caractère - à la colonne 7 de la ligne suivante, indique la suite du mot dans la zone B.

Les ordres COBOL commencent à la colonne 8 et se terminent à la colonne 72.

Les colonnes 8 à 11 constituent la marge A.

En règle générale, tous les entêtes de divisions, de sections, et de paragraphes, ainsi que les indicateurs de niveau devront débuter dans la zone A.

Les colonne 12 à 72 constituent la marge B. Les noms de données et les phrases commencent dans la marge B.

COLONNES				
123456	7	Marge A 8→11	Marge B : 12 → 72	73 → 80
		IDENTIFICATION DIVISION.		
	*	Ceci est un commentaire : une étoile à la colonne 7		
		PROGRAM-ID. Multiplication. AUTHOR. Imed Jarras. DATA DIVISION. WORKING-STORAGE SECTION. 01 LesEntrees 02 Num1 PIC 9 VALUE 0. 02 Num2 PIC 9 VALUE ZEROS. 77 Produit PIC 99 VALUE ZERO. PROCEDURE DIVISION. 000-debut.		
			DISPLAY "Entez la premiere valeur : " WITH NO ADVANCING. ACCEPT Num1. DISPLAY "Entez la deuxième valeur : " WITH NO ADVANCING. ACCEPT Num2. MULTIPLY Num1 BY Num2 GIVING produit. DISPLAY "Produit = ", PRODUIT. STOP RUN.	

6 IDENTIFICATION DIVISION

C'est la première division d'un programme COBOL.

Format général

IDENTIFICATION DIVISION.
PROGRAM-ID. nom-du-programme.
[AUTHOR. commentaire**]**
[INSTALLATION. commentaire**]**
[DATE-WRITTEN. date-d'écriture**]**
[DATE-COMPILED. date-de-compilation**]**
[SECURITY. commentaire de sécurité**]**

DATE-COMPILED: à la compilation la date du jour est insérée par le système.

[ ] : paragraphe optionnel.

ENVIRONMENT DIVISION

Environnement matériel du programme (Ordinateur et fichiers utilisés).

Format général

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. ordinateur.
OBJECT-COMPUTER. ordinateur.
[INPUT-OUTPUT SECTION.
FILE-CONTROL.
.....]
I-O-CONTROL. .....]

Exemple

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-370.
OBJECT-COMPUTER. IBM-370.

7 DATA DIVISION

Déclaration et description détaillée des structures utilisées (variables, constantes, vecteurs, fichiers).

Format général

DATA DIVISION.
[FILE SECTION.
description des fichiers]
[WORKING-STORAGE SECTION.
déclaration et description des variables,
constantes, vecteurs, etc..]

7.1 LES VARIABLES ÉLÉMENTAIRES

Les variables élémentaires ne sont pas décomposables en constituants.

Elles peuvent exister soit isolément, soit comme constituants d'une variable structurée.

Quatre éléments sont nécessaires à la description d'une variable **élémentaire**:

1. le nom de la variable
2. le type de la variable
3. le nombre maximum de caractères
4. le mode de représentation en mémoire

Exemple de variables élémentaires

```
77   NOM      PIC      X(20)  USAGE IS DISPLAY.
77   PRENOM   PICTURE  X(15).
```

Exemple de variable structurée

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

SYNTAXE DE LA DESCRIPTION D'UNE VARIABLE

Une donnée élémentaire sera décrite dans la WORKING-STORAGE SECTION de la DATA DIVISION.

1 Numéro de niveau

Le numéro de niveau doit commencer en zone A.

Il doit être le premier élément de la description d'une donnée.

Il peut avoir la valeur **77** qui indique une donnée élémentaire dans la WORKING-STORAGE SECTION.

2. Nom de donnee

Le nom de la donnée doit être un mot COBOL: de 1 à 30 caractères pris parmi les lettres [A..Z], les chiffres et le tiret, non entièrement numérique, et ne commençant ni finissant par un tiret.

Il ne doit pas être un mot réservé.

Il doit commencer en zone B.

Exemple

```
DATA DIVISION.
WORKING-STORAGE SECTION.
77        NOM-DE-DONNEE .....
77        NOM-1 .....
```

3. PICTURE

La clause **PICTURE** (ou **PIC**) permet de décrire les caractéristiques générales de la donnée: le ***type*** et la ***longueur***. Ces éléments sont fournis sous la forme d'une chaîne de caractères.

3.1 les types de données

Les données pourront être :

alphabétiques: le symbole permettant de définir des donnée alphabétique (de [A..Z] + espace) sera la lettre **A**.

Chaque occurrence de la lettre A de la chaîne de caractères de picture représentera une position de caractère de la donnée pouvant contenir une lettre ou un espace.

alphanumériques: Elles pourront contenir une combinaison quelconque du jeu de caractères de l'ordinateur.

Le symbole **X** caractérisera une donnée alphanumérique.

numériques: Elles pourront alors contenir une combinaison de chiffres (0..9) et éventuellement un signe algébrique.

Le symbole permettant de définir une donnée numérique sera le chiffre **9**.

Une donnée numérique comprend un maximum de 18 chiffres significatifs.

3.2 la longueur des données

C'est la répétition des symboles caractérisant le type de données (A, X, 9) qui fournira la longueur de la donnée élémentaire.

Deux notations pourront être employées:

répétition: Par exemple, PIC XXX indiquera une donnée alphanumérique de 3 caractères.

mise en facteur: PIC X(3) ou PIC X(03), décrivent de la même manière une donnée élémentaire alphanumérique de 3 caractères.

Exemples de PICTURE

```
77      NOM     PIC     A(20).
77      NOM     PICTURE A(20).
```

Définit une donnée alphabétique occupant 20 caractères.

```
77      ADRESSE PIC     X(40).
```

Définit une donnée alphanumérique occupant 40 caractères.

```
77      AGE     PIC     9(2).
77      AGE     PIC     99.
```

Définit une donnée numérique occupant 2 positions.

PICTURES	SIGNIF.	VALIDE	N.VALIDE
XXX	3 caractères	A$%	
X(3)	3 caractères	B52	
99999	5 chiffres	16754	AH767
9(5)	5 chiffres	00000	ttttt
AAAA	4 caractères alphabétiques	AB D	A-C1
AA99	4 caractères	AC01	01AC
99XA	4 caractères	15-X	CD11

4. USAGE

La clause **USAGE** indique le mode de représentation des données à l'intérieur de l'ordinateur.

USAGE IS DISPLAY

C'est le mode de représentation utilisé par défaut.

Il indique en pratique que chaque caractère occupe un octet.

L'usage ne peut être DISPLAY que si l'élément de donnée est:

alphanumérique: PIC X(n)

alphabétique: PIC A(n)

numérique: PIC 9(n), utilisé à but d'identification et non pas de calcul.

Exemple:

```
77      NOM     PIC     A(20)   USAGE IS DISPLAY.
```

Est équivalent à:

```
77      NOM     PIC     A(20).
```

USAGE IS COMPUTATIONAL (COMP)

Pour les données numériques qui seront utilisées pour des calculs.

COMP: seulement si la donnée est numérique et utilisée dans des calculs.

Exemple:

```
77      PRIX-UNIT       PIC     99      COMP.
77      QTE-STOCK       PIC 9(3)  USAGE IS COMPUTATIONAL.
```

5. LES NUMERIQUES

Le symbole 9 décrit une donnée numérique.

Trois autres symboles sont nécessaires.

Le symbole V

- Indique la présence d'un point décimal fictif (n'est pas réellement représenté).
- Utilisé en conjonction avec le symbole 9.
- Représente une marque décimale implicite qui ne compte pas dans la longueur de la donnée.
- Figure une seule fois dans la description de la donnée.
- Utilisé avec le USAGE DISPLAY et le USAGE COMP.

Exemple

```
77      TAUX-HORAIRE        PIC     99V99.
```

TAUX-HORAIRE sur 4 positions dont deux après la virgule. (08,25)

```
77      PRIX-UNIT           PIC     9(3)V99 COMP.
```

PRIX-UNIT sur 5 positions dont deux après la virgule. (002,28).

Le symbole S

Il indique la présence d'un signe.

Il doit être écrit le plus à gauche de la chaîne de caractères de picture.

Il ne compte pas dans la taille de la donnée.

Sans le symbole S, l'ordinateur génère toujours le signe + pour le nombre: sa valeur absolue.

Exemple

```
77      SOLDE           PIC             S9(3)V9(2).
77      BALANCE             PIC             S9999V99.
```

6. LES CARACTERES D'EDITION

Les caractères d'édition (B Z . , $ * ...) de la clause PICTURE sont inutilisables dans des calculs.

Toutefois, ils peuvent recevoir des résultats de calcul.

Ne jamais utiliser le USAGE COMP ou COMP-3 dans une clause PICTURE de caractères d'édition.

N.B: Dans la suite, le caractère **b** désigne un espace.

6.1 Le caractère B (blanc ou espace)

Représente la position d'un caractère espace.

Reformate les données lorsqu'elles sont affectées à une variable qui comporte ce picture d'édition.

Élément	Picture	Élément édité
325	9B9B9	3b2b5
PERSONNE	ABBABA(6)	PbbEbRSONNE
SOLDE-2	XBXBXBXBX(3)	SbObLbDbE-2

Le caractère d'édition B sera insérer dans la chaîne de caractères.

Le caractère Z

Représente la position d'un caractère poids fort, caractère remplacé par un espace lorsqu'il s'agit d'un zéro non significatif.

Le caractère Z ne doit jamais être précédé par des caractères 9, espace ou zéro.

Élément	Picture	Élément édité
120	9(4)	0120
0	9(3)	000
120	Z9(3)	b120
0	ZZ9	bb0
0	ZZZ	bbb
13	ZZ9(2)	bb13

Le point décimal (.)

Représente la position d'un point décimal réel dans un champ.

- Remplace le point décimal fictif (V).
- Il ne peut jamais être le dernier caractère dans une clause picture.
- Le caractère point décimal (.) entouré de deux caractères Z ne s'imprime pas si les deux Z représentent des zéros en tête.

Élément	Picture	Élément édité
234	**999.99**	**234.00**
13,25	**ZZ.99**	**13.25**
0	**ZZ.ZZ**	**bbbbb**
0.01	**ZZ.99**	**bb.01**

La virgule

Représente le caractère virgule, qui sépare les chiffres d'un nombre par tranches de trois chiffres, de la droite vers la gauche.

- Une virgule placée entre deux Z est aussi remplacée par un blanc si les deux Z représentent des zéros de tête.
- Une virgule est remplacée par un blanc s'il n'y a que des blancs à sa gauche.

Élément	Picture	Élément édité
36425	**99,999**	**36,425**
128	**ZZ,999**	**bbb128**
265,32	**ZZ,ZZZ.99**	**bbb265,32**
0	**ZZ,ZZZ.ZZ**	**bbbbbbbbb**
0.02	**ZZ,ZZZ.99**	**bbbbbb.02**

Le caractère $

Représente à l'édition de la donnée, lorsqu'il est unique et placé dans la position d'extrême gauche, un caractère signe $ dans cette position.

Élément	Picture	Élément édité
123	**$999**	**$123**
013	**$999**	**$013**
30	**$ZZ9**	**$b30**
12.45	**$Z,ZZZ.99**	**$bbb12.45**
3265.12	**$Z,ZZZ.99**	**$3,265.12**

Lorsque plusieurs signes $ consécutifs sont placés à l'extrême gauche, $ flottant, un seul signe est édité, celui qui est placé à gauche du caractère le plus significatif du nombre.

Élément	Picture	Élément édité
12	**$$9**	**$12**
4.12	**$$,$$$.99**	**bbbb$4.12**
0.01	**$$,$$$.99**	**bbbbb$.01**

Le caractère Astérisque (*)

Représente un caractère de protection à gauche, c'est-à-dire la suppression des zéros non significatifs et remplacement de chaque zéro par un astérisque.

Élément	Picture	Élément édité
12	***9	**12
3612.48	$*,***,***.99	$****3,612.48
0.00	$*,***.**	$*****.**

Le caractère zéro (0)

Représente un caractère zéro à insérer dans la position indiquer par le picture.

Élément	Picture	Élément édité
138	ZZ9000	138000
1280	999900	128000
3225	$ZZ,ZZZ.00	$b3,225.00
1	$ZZ,ZZZ.00	$bbbbb1.00
1	$$$,$$$.00	bbbbb$1.00

Le caractère (/)

Réserve un octet (le caractère /) dans le résultat édité.

Représenter la date.

Élément	Picture	Élément édité
270194	99/99/99	27/01/94

Caractères de signe (+ -)

Le signe + à la fin de la chaîne picture entraîne toujours l'impression du signe du nombre qu'il soit positif ou négatif.

- Le signe - n'est imprimé que lorsque la valeur éditée est négative.
- Les signes + et - peuvent être utilisés de manière flottante comme le signe $.

Élément	Picture	Élément édité
-215.8	ZZZ.9+	215.8-
-215.8	+ZZZ.9	-215.8
2.1	++,+++.9	bbbb+2.1
0	++,++9	bbbb+0
2.1	--,---.9	bbbbb2.1
32.4	-ZZ.9	b32.4
32.4	ZZ.9-	32.4b-
-32.4	-ZZ.9	-32.4

-32.4	ZZ.9-	32.4-

Les caractères de signe (DB CR)

Les symboles CR et DB (Crédit et Débit) peuvent être utilisés à droite d'une valeur numérique pour indiquer une valeur négative.

Élément	Picture	Élément édité
12.48	$Z,ZZZ.99CR	$bbb12.48bb
13.28	$Z,ZZZ.99DB	$bbb13.28bb
-12.48	$Z,ZZZ.99CR	$bbb12.48CR
-13.28	$Z,ZZZ.99DB	$bbb13.28DB

8 LES VARIABLES STRUCTUREES

Les variables étudiées jusqu'à présent ont une caractéristique importante: l'indivisibilité de leurs valeurs.

- Chaque valeur est manipulée comme un tout.
- Lorsque les éléments de l'ensemble sont des variables élémentaires et que l'ensemble porte un nom collectif, on a une variable structurée sur laquelle on peut opérer soit obligatoirement soit au niveau de ses composants.

Les composants d'une variable structurée peuvent être des variables de type structurée et il est possible d'y accéder directement, quelle que soit leurs places dans la structure.

Pour définir une variable structurée, il faut préciser:

1. Son **mode** de structuration
2. Le ou **les types** de variables intervenant dans la structure.

EXEMPLE

01 LIGNE-EDITION.

1. numéro-de-niveau

Le numéro de niveau montre la hiérarchie des données.

- Il pourra prendre une valeur allant de 01 à 49.
- Le numéro de niveau 01 servira à identifier la première rubrique de chaque description d'une structure. Il devra **commencer** en **zone A**.
- Les numéros de niveau des données dépendantes prendront des valeurs comprises entre 02 et 49, en permettant ainsi de refléter la hiérarchie des données.
- Ils devront commencer en zone B.

2. nom-de-donnée

Le nom de donnée indiquera le nom attribué à la donnée décrite.

Ce sera le seul élément de description d'une donnée structurée.

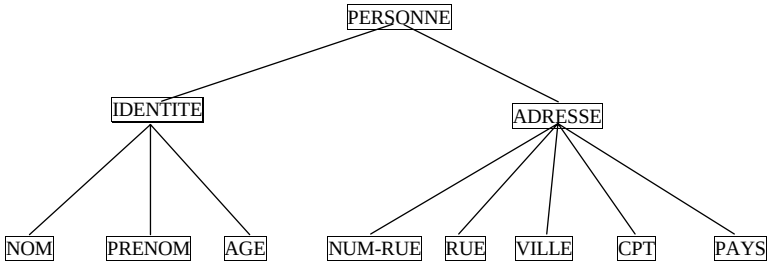
Une donnée structurée a toujours une définition implicite:

1. Elle est alphanumérique.

2. Elle a pour longueur la somme des longueurs des données qui en dépendent.

3. Les données élémentaires d'une donnée structurée ne devront être décrites que dans l'ordre de leur apparition.

Exemple



DATA DIVISION.

WORKING-STORAGE SECTION.

77	NOM	PIC	X(20).
77	PRENOM	PIC	X(15).
77	AGE	PIC	99.
77	NUM-RUE	PIC	9(3).
77	RUE	PIC	X(20).
77	VILLE	PIC	X(15).
77	CODE-POSTAL	PIC	X(6).
77	PAYS	PIC	X(15).

DATA DIVISION.

WORKING-STORAGE SECTION.

01	PERSONNE.				
	05	IDENTITE.			
		10	NOM	PIC	X(20).
		10	PRENOM	PIC	X(15).
		10	AGE	PIC	99.
	05	ADRESSE.			
		10	NUM-RUE	PIC	999.
		10	RUE	PIC	X(20).
		10	VILLE	PIC	X(15).
		10	CODE-POSTAL	PIC	X(6).
		10	PAYS	PIC	X(15).

3.3 INITIALISATION DES DONNEES

3.3.1 VALUE

Le mot réservé VALUE suivi d'une valeur (littéral numérique ou alphanumérique, constante figurative) permet de définir une constante.

Exemple1

DATA DIVISION.

WORKING-STORAGE SECTION.

77	PI	PIC	9V9(5)	VALUE	3.14159.
77	TITRE	PIC	X(21)	VALUE	"BULLETIN".

Le littéral ou la constante figurative spécifiés après l'ordre VALUE devraient être de même type que la variable déclarée.

Exemple2

AGE <---- 05.

PRENOM <---- "JEANbbbbbbbb"

DATA DIVISION.

WORKING-STORAGE SECTION.

01	DETAILS.					
	05	AGE	PIC	99	VALUE	5.
	05	PRENOM	PIC	X(12)	VALUE	"JEAN".

mots réservés = { IS ZEROS, IS SPACES, ZERO, SPACE, ALL caractère}

Exemple1

	VALUE	**IS ZEROS.**
	VALUE	**SPACES.**

```
            VALUE       ZERO.
            VALUE       ALL "*".
            VALUE       ALL "*-".
```

3.3.2 FILLER

Le mot réservé FILLER permet de définir, dans une variable structurée, une variable élémentaire à laquelle il est prévu de ne pas avoir accès.

Exemple1

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01    ENTETE.
      05   FILLER    PIC   X(5)   VALUE    "NOM: ".
      05   FILLER    PIC   X(10)  VALUE    SPACES.
      05             PIC   X(8)   VALUE    "ADRESSE:".
      05   FILLER    PIC   X(10)  VALUE    ALL "*".
      05             PIC   X(5)   VALUE    "TEL: ".
      05   FILLER    PIC   X(10)  VALUE    SPACES.
```

9 PROCEDURE DIVISION

C'est la division de traitement. C'est dans cette division que l'algorithme sera traduit en instructions exécutables. Elle est constituée de sections et/ou de paragraphes.

Format général

```
PROCEDURE DIVISION.
[NOM-DU-PARAGRAPHE.]
        INSTRUCTION-1.
        INSTRUCTION-2.
                .....
                .....
        [APPEL  PARAGRAPHE-2.]
                .....
                .....
        INSTRUCTION-n.
        STOP RUN.
[PARAGRAPHE-2.]
```



```
        INSTRUCTION-1.
        INSTRUCTION-2.
                .....
        INSTRUCTION-n.
```

PROCEDURE DIVION

- L'indicatif de cette division, PROCEDURE, doit être frappé dans la zone A.

NOM-DU-PARAGRAPHE

- Nom du paragraphe principal (programme principal).
- Optionnel.

LES INSTRUCTIONS

- Les instructions du programme devraient commencer à la colonne 12 (zone B).
- Chaque ligne comprendra une seule instruction.

STOP RUN

Indique la fin du programme principal.

- Elle doit être tapée à partir de la colonne 12 (zone B).
- Aucune autre instruction du programme principal ne doit être écrite après l'ordre STOP RUN.

PARAGRAPHE-2

- Indique un bloc d'instructions.
- Correspond à une procédure non paramètrée.

Exemple

```
        PROCEDURE DIVISION.
        DEBUT.
                ADD 1 TO NOMBRE.
                PERFORM P-CUMUL VARYING CPT FROM
                        1 BY 1 UNTIL CPT = NOMBRE.
                DISPLAY SOMME.
                STOP RUN.
        P-CUMUL.
                ADD CPT TO SOMME.
```

REMARQUES:

- Les noms de paragraphes devraient commencer dans la zone A.
- Une seule instruction par ligne.
- Décaler les instructions qui prennent plus qu'une ligne.
- Ne pas oublier l'ordre STOP RUN à la fin du programme principal (dans la zone B).

10 LES INSTRUCTIONS D'ENTREE/SORTIE

Leur exécution provoque un transfert de données entre la mémoire centrale d'une part et les unités d'entrée/sortie d'autre part.

ACCEPT

C'est l'ordre COBOL qui permet d'entrer des données dans la mémoire centrale à partir d'un terminal (CONSOLE) ou d'un fichier spécial (SYSIN).

Format1

ACCEPT nom-variable

```
ACCEPT NOMBRE FROM TERMINAL.
ACCEPT NOMBRE FROM CONSOLE.
```

Format2

ACCEPT nom-variable [FROM { DATE, DAY, TIME, }]

Cette instruction transfère dans la variable désignée la date, le jour ou l'heure.

Les données transférées sont de type numérique 9(n).

Elles sont représentées sur six (6) octets pour la date, cinq (5) octets pour les jours et huit (8) octets pour l'heure.

Exemple1

```
DATA DIVISION.
WORKING-STORAGE SECTION.
77 DATE-DU-JOUR PIC 9(6).
PROCEDURE DIVISION.
    ACCEPT DATE-DU-JOUR FROM DATE.
```

Par exemple, la date 25 Janvier 1993 sera représentée sous la forme: 930125.

Exemple 2

```
DATA DIVISION.
WORKING-STORAGE SECTION.
77 JOUR PIC 9(5).
PROCEDURE DIVISION.
    ACCEPT JOUR FROM DAY.
```

Le système d'exploitation place la date julienne dans la variable JOUR. Elle sera représentée sous la forme AAJJJ (JJJ >= 1 et JJJ <= 365).

25 janvier 1993 --> 93025

2 Février 1993 --> 93033

Exemple3

```
DATA DIVISION.
WORKING-STORAGE SECTION.
77 HEURE PIC 9(8).
PROCEDURE DIVISION.
    ACCEPT HEURE FROM TIME.
```

L'heure du jour est un numérique sur 8 positions sous la forme **HH**mm**ss**cc.

5H25mn3s25c (p.m) --> 17250325

DISPLAY

C'est l'ordre COBOL qui permet de transférer des données de la mémoire centrale au terminal (CONSOLE) ou à un fichier SYSOUT.

Format général

DISPLAY {constante, littéral, variable} [{constante, littéral, variable } …]

Exemple1

```
PROCEDURE DIVISION.
DISPLAY  "EXEMPLE DE TEST" .
DISPLAY   TOTAL.
DISPLAY "TOTAL = "  TOTAL .
```


11 L'INSTRUCTION D'AFFECTATION MOVE

L'instruction d'affectation permet de transférer une donnée d'une zone de la mémoire centrale dans une autre zone.

- L'instruction MOVE permet de copier le contenu d'une variable (élémentaire ou structurée) dans une ou plusieurs variables.
- Le contenu de la variable émettrice reste inchangé par l'exécution de l'instruction MOVE.
- Le contenu des variables réceptrices sera remplacé par celui des variables émettrices.
- L'exécution de l'instruction MOVE dépend du type (PICTURE) de la variable émettrice et réceptrice, du mode de représentation en mémoire (USAGE) et de la taille des variables.

Format général

MOVE {variable} {littéral} {constante} TO variable1 [variable2 ...].

4.2.1 Transfert des données élémentaires

Exemple1: Alphanumérique/Alphanumérique

```
77  A  PIC  X(5)  VALUE  "CHAMP".
77  B  PIC  X(5)  VALUE  "DANS".
77  C  PIC  X(6)  VALUE  "NOMBRE".
77  D  PIC  X(3)  VALUE  "STR".
```

-Les deux données ont la même longueur:

A= "CHAMP", B= "DANS"
MOVE A TO B.
A= "CHAMP", B= "CHAMP"

-Les deux données sont de longueurs différentes:

Longueur de la donnée émettrice est supérieur à celle de la donnée réceptrice.

C= "NOMBRE", D= "STR"
MOVE C TO D
C= "NOMBRE", D="NOM".

Longueur de la donnée émettrice est inférieur à celle de la donnée réceptrice.

D= "STR", C= "NOMBRE"
MOVE D TO C.
D= "STR" C= "STRbbb".

Exemple numérique/numérique édité

```
77  A  PIC  S9(4)V99  VALUE  +2451.33.
```

77 B PIC S9(6)V99 VALUE +000431.75.
77 C PIC $$$,$$$.99-.
77 D PIC $*,***.99.**

1. MOVE A TO C.

C= b$2,451.33b.

2. MOVE B TO D.

D= $****431.75

TRANSFERT DES DONNEES STRUCTURÉES

Exemple

```
01  ANCIENNE-BALANCE.
    05  A-BAL-PRINC PIC  9(5)  VALUE  11111.
    05  A-BAL-INTER PIC  9(5)  VALUE  22222.
    05  A-BAL-TOTAL PIC  9(5)  VALUE  33333.
01  NOUVELLE-BALANCE.
    05  N-BAL-PRINC PIC  9(7).
    05  N-BAL-INTER PIC  9(7).
    05  N-BAL-TOTAL      PIC  9(7).
```

- La variable structurée ANCIENNE-BALANCE est de type ALPHANUMÉRIQUE de 15 caractères.
- La variable structurée NOUVELLE-BALANCE est de type ALPHANUMERIQUE de 21 caractères.

MOVE ANCIENNE-BALANCE TO NOUVELLE-BALANCE.

NOUVELLE-BALANCE ="111112222233333bbbbbb"

N-BAL-PRINC= "1111122"

N-BAL-INTER= "2223333"

N-BAL-TOTAL= "3bbbbbb"

N.B: IL EST CONSEILLER D'UTILISER LE **MOVE** ELEMENTAIRE MÊME AVEC LES VARIABLES STRUCTURÉES.

12 LES OPERATIONS ARITHMETIQUES

Il existe quatre instructions arithmétiques: addition, soustraction, multiplication et division.

Les opérandes (variables ou constantes) qui interviennent dans les opérations doivent être du type numérique non édité.

Instructions d'addition

1.Addition de plusieurs opérandes à une ou plusieurs variables.

Syntaxe

ADD {litt1, var1} [{litt2, var2} …] TO varn [varm…].

Exemple

```
ADD    1    TO          COMPTEUR.
                COMPTEUR <-- COMPTEUR + 1
ADD    TOT-PARTIEL          TO      TOTAL.
                TOTAL <-- TOTAL + TOT-PARTIEL
ADD    NOMBRE1  NOMBRE2  TO  SOMME.
       SOMME <-- SOMME + NOMBRE1 + NOMBRE2

ADD            S1      S2      TO      T3      T4.
               T3 <-- T3 + S1 + S2
               T4 <-- T4 + S1 + S2
```


- L'option TO requiert au moins un élément à sa gauche et au moins un à sa droite.
- Les littéraux numériques ne peuvent figurer qu'à gauche de TO.
- Les variables doivent être numériques et non éditées.
- Le système se charge de la conversion d'un usage à un autre.
- Le résultat de l'addition doit être contenu sans perte de chiffres dans la zone mémoire qui lui est réservée.

Addition de plusieurs opérandes et affectation du résultat à une nouvelle variable

Syntaxe

ADD {litt1, var1} {litt2,var2} … [{littj, varj}] GIVING var_n [var_m...]

Exemple

```
ADD    NOMBRE1  NOMBRE2  GIVING   SOMME.
               SOMME <-- NOMBRE1 + NOMBRE2.
ADD    TOT-1    TOT-2  TOT-3   GIVING  TOTAL.
               TOTAL <-- TOT-1 + TOT-2 + TOT-3
ADD            S1   S2   S3      GIVING       T1   T2.
               T1 <-- S1 + S2 + S3
               T2 <-- S1 + S2 + S3
```

- L'option GIVING requiert au moins deux arguments à sa gauche et au moins un argument à sa droite.
- Les éléments à gauche de GIVING doivent être numériques et non édités.
- Ceux qui sont à droite doivent être numériques et peuvent être édités.
- Cette deuxième expression syntaxique de l'addition est vivement recommandée en raison de:
  - Pas de risque d'écrasement des données.
  - Pas de risque de dépassement de capacité si les zones mémoires sont correctement dimensionnées.

Instructions de soustraction

1. Soustraction de plusieurs opérandes d'une ou plusieurs variables

Syntaxe

SUBTRACT {litt1, var1} [{litt2, var2} …]FROM var_n [var_m…]

Exemple

SUBTRACT 1 FROM COMPTEUR.
COMPTEUR <-- COMPTEUR -1
SUBTRACT NOMBRE FROM SOMME.
SOMME <-- SOMME - NOMBRE
SUBTRACT NOMBRE1 NOMBRE2 FROM TOTAL.
TOTAL <-- TOTAL - NOMBRE1 - NOMBRE2
Remarque : SUBTRACT ... FROM ... même principe que ADD .... TO ....

Soustraction de plusieurs opérandes et affectation du résultat à une nouvelle variable.
Syntaxe

SUBTRACT {litt1, var1} [{litt2, var2}] .... FROM {$litt_n$, var_n} GIVING var_m.
Exemple
SUBTRACT NOMBRE FROM TOTAL GIVING N-TOTAL.
N-TOTAL <-- TOTAL - NOMBRE.
SUBTRACT 2 FROM CPT GIVING CPT2.
CPT2 <-- CPT - 2
SUBTRACT S1 S2 FROM T1 GIVING T2.
T2 <-- T1 - S1 - S2
SUBTRACT S1 S2 S3 FROM T1 GIVING T2 T3.
T2 <-- T1 - S1 - S2 - S3
T3 <-- T1 - S1 - S2 - S3.

Instructions de multiplication
1. Multiplication d'un opérande et d'une variable
Syntaxe
MULTIPLY {litt1, var1} BY var2.
Exemple
MULTIPLY S1 BY T1.
T1 <-- T1 * S1
MULTIPLY 20 BY PRIX-UNIT.
PRIX-UNIT <-- PRIX-UNIT * 20

2. Multiplication de deux opérandes et affectation du résultat à une nouvelle variable.
Syntaxe
MULTIPLY {litt1,var1} BY {litt2, var2} GIVING var3.
Exemple
MULTIPLY P-UNIT BY QTE GIVING P-TOTAL.
P-TOTAL <-- P-UNIT * QTE
MULTIPLY A BY B GIVING C D.

C <-- A * B
D <-- A * B

Instructions de division
1. Division d'une variable par un opérande et affectation du résultat à la variable
Syntaxe
DIVIDE {litt1, var1} INTO var2.
Exemple
DIVIDE DIVISEUR INTO DIVIDENDE.
DIVIDENDE <-- DIVIDENDE/DIVISEUR.
DIVIDE 2 INTO NOMBRE.
NOMBRE <-- NOMBRE/2
DIVIDE A INTO B.
B <-- B/A
Division de deux opérandes et affectation du résultat à une nouvelle variable
Syntaxe

DIVIDE {litt1, var1} {BY, INTO} {litt2, var2} GIVING var3.

Exemple
DIVIDE DIVISEUR INTO DIVIDENDE GIVING QUOTIENT.
QUOTIENT <-- DIVIDENDE/DIVISEUR.
DIVIDE DIVIDENDE BY DIVISEUR GIVING QUOTIENT.
QUOTIENT <-- DIVIDENDE/DIVISEUR.
DIVIDE B BY A GIVING C.
C <-- B/A.

Calcul du reste
En ajoutant aux expressions syntaxiques du paragraphe 1 et 2 l'option REMAINDER var4, la valeur du reste de la division sera affectée à la var4.
Exemple
DIVIDE B BY A GIVING C REMAINDER D.
C <-- B/A
D <-- B - (A * C)

DIVIDE DIVISEUR INTO DIVIDENDE GIVING QUOTIENT REMAINDER RESTE.
QUOTIENT <-- DIVIDENDE/DIVISEUR
RESTE <-- DIVIDENDE - (QUOTIENT * DIVISEUR)

DIVIDE DIVIDENDE BY DIVISEUR GIVING QUOTIENT REMAINDER RESTE.
QUOTIENT <-- DIVIDENDE/DIVISEUR
RESTE <-- DIVIDENDE - (QUOTIENT * DIVISEUR)

L'option ROUNDED

En faisant suivre le nom d'une variable résultat de ROUNDED, le résultat du calcul est arrondi à l'unité superieur du chiffre le moins significatif (le plus à droite de la partie non tronquée), si le chiffre le plus significatif de la partie tronquée (le plus à gauche) est supérieur ou égal à 5.

Exemple

```
77    TOTAL      PIC   S99V9.
ADD   A    B     GIVING      TOTAL.
      TOTAL <-- A + B
```

Si la somme de A et B est +21.3687, les trois dernières décimales seront tronquées et TOTAL <-- +21.3

```
ADD   A    B  GIVING  TOTAL ROUNDED.
      TOTAL <-- +21.4
```

Le chiffre le plus à gauche de la partie à tronquer est un 6 (>5). Donc, le chiffre le plus à droite de la partie gardée 3 sera 3 + 1 = 4 d'où TOTAL <-- +21.4.

Si la somme de A et B était +21.3487.

```
ADD A  B  GIVING  TOTAL ROUNDED.
      TOTAL <-- 21.3
```

Le chiffre le plus à gauche de la partie à tronquer est 4 (<5).

DEPASSEMENT DE CAPACITE

Un dépassement de capacité lors de l'exécution des instructions arithmétiques provoque l'interruption du traitement, à moins d'utiliser l'option ON SIZE ERROR.

La clause ON SIZE ERROR permet de spécifier un jeu d'instructions qui ne seront exécutées que dans le cas d'un débordement.

Exemple

```
77    TOTAL      PIC   S9(3)V9     VALUE   +121.2.
77    NB1        PIC   S9(3)V9     VALUE     +933.8.
77    NB2        PIC   S9(2)V9     VALUE     +93.1.
ADD   NB1 NB2 GIVING TOTAL ROUNDED
                   ON SIZE ERROR
                   DISPLAY "DEBORDEMENT: TOTAL".
```

Dans le cas de dépassement de capacité, la variable TOTAL gardera sa valeur juste avant l'opération (+121.2) et le message impératif de l'ordre ON SIZE ERROR s'imprime.

```
SUBTRACT   NB1 FROM TOTAL GIVING NB2
                   ON SIZE ERROR
"APPEL D'UNE PROCEDURE DE RECOUVREMENT".

DIVIDE     A     INTO B GIVING C
                        ON SIZE ERROR
                        DISPLAY "ERREUR DANS C".
```


Format Général

```
      {var1} {var2}
ADD   {litt1} [      {litt2} ] ....  TO  var_n [ROUNDED] ... var_m [ROUNDED]]
                        [ON SIZE ERROR ordre-impératif].
      {var1} {var2}      {varj}
ADD  {litt1}          {litt2} ... [ {littj} ]  ...  GIVING  var_n [ROUNDED] ...      [var_m
[ROUNDED]]  [ON SIZE ERROR ordre_impératif].

            {var1} {var2}
SUBTRACT    {litt1} [    {litt2}  ] .... FROM  varn [ROUNDED]  [varm [ROUNDED]] ...
            [ON SIZE ERROR ordre_impératif].

            {var1} {var2}                 {varj}
SUBTRACT    {litt1} [    {litt2}  ] ....  FROM   {littj}
                     GIVING  varn [ROUNDED]   [varm [ROUNDED]]...
            [ON SIZE ERROR ordre_impératif].

            {var1}
MULTIPLY  {litt1}  BY  var2 [ROUNDED]   [varm [ROUNDED]]...
            [ON SIZE ERROR ordre_impératif].
            {var1}      {var2}
MULTIPLY  {litt1} BY  {litt2}  GIVING  var3 [ROUNDED]
                  [varn [ROUNDED]]...   [ON SIZE ERROR ordre_impératif].

            {var1}
DIVIDE      {litt1}    INTO   var2 [ROUNDED]   [varm [ROUNDED]]...
            [ON SIZE ERROR ordre_impératif].

          {var1}  {INTO}    {var2}
DIVIDE    {litt1}  {BY}    {litt2}  GIVING   var3 [ROUNDED] [varm [ROUNDED]]...
            [ON SIZE ERROR ordre_impératif].
          {var1}  {INTO}    {var2}
DIVIDE    {litt1}  {BY}    {litt2}  GIVING  var3 [ROUNDED]   REMAINDER  var4
            [ON SIZE ERROR ordre_impératif].
```

L'ORDRE COMPUTE

Permet de spécifier plusieurs opérations arithmétiques à la fois.

Syntaxe

```
COMPUTE  var1 [ROUNDED]  [var2  [ROUNDED]] ...
            = expression arithmétique      [ON SIZE ERROR ordre_impératif].
```

L'expression arithmétique est formée d'éléments numériques et/ou de littéraux numériques, combinés avec un ou plusieurs opérateurs arithmétiques (**, *, /, +, -).

var1, var2 sont de type numérique et peuvent être éditées.

Chaque opérateur arithmétique devait être suivi et précédé d'un espace.

Exemple

```
COMPUTE  COMPTEUR = COMPTEUR  +  1.
                ADD 1 TO COMPTEUR
COMPUTE SOMME = SOMME  +  NB1  +  NB2.
                ADD NB1  NB2  TO SOMME
COMPUTE TOTAL ROUNDED = T1  +  T2  +  T3.
                ADD T1 T2 T3 GIVING TOTAL ROUNDED
COMPUTE   A =  B  -  C.
                SUBTRACT  C FROM  B  GIVING A


COMPUTE  T3 = T1   -  S1  -  S2.
                SUBTRACT  S1  S2  FROM  T1  GIVING  T3.
COMPUTE  T1 ROUNDED =  T2  /  T3
                DIVIDE T2  BY  T3  GIVING  T1 ROUNDED.


COMPUTE A =  - B.
                SUBTRACT B FROM ZERO GIVING A.
```

COMPUTE A = B ** C. ?

5.2.1 EVALUATION D'UNE EXPRESSION ARITHMETIQUE

- L'expression arithmétique est évaluée selon les règles de précédence des opérateurs: **; * /; + -.
- Les opérateurs de même précédence sont pris en compte de la gauche vers la droite.
- Si les parenthèses sont utilisées, les expressions entre parenthèses sont évaluées en premier.
- A l'interieur de plusieurs parenthèses, l'évaluation se fait en allant des parenthèses les plus internes vers les plus externes.
- Le résultat de l'expression est affecté à la variable dont le nom suit compute, cette variable est obligatoirement numérique et peut être éditée.

Exemple

COMPUTE A ROUNDED B = A - B * T / F ON SIZE ERROR DISPLAY "ERREURS: A B".

évaluation (B * T)
évaluation ((B * T)/ F)
évaluation (A - ((B * T) / F))

L'ORDRE PERFORM

- C'est l'instruction de contrôle cobol qui permet de réaliser un branchement avec retour.
- Elle transfère le contrôle à la première instruction d'un paragraphe et après exécution de la dernière instruction du paragraphe, le contrôle est ensuite rendu à l'instruction qui suit PERFORM.

Syntaxe

PERFORM nom-du-paragraphe.

Exemple

```
PROCEDURE DIVISION.
DEBUT.
        MOVE ZERO TO TOTAL-QTE.
*   LECTURE DES DONNEES
        DISPLAY 'ARTICLE1, QTE:' .
        ACCEPT  QTE-1  FROM CONSOLE.
        DISPLAY 'ARTICLE1, P-UNIT-1:' .
        ACCEPT  P-UNIT-1  FROM CONSOLE.
        DISPLAY 'ARTICLE2, QTE:' .
        ACCEPT  QTE-2  FROM CONSOLE.
        DISPLAY 'ARTICLE2, P-UNIT-2:' .
        ACCEPT  P-UNIT-2  FROM CONSOLE.

*       CALCUL DE LA QUANTITE TOTALE
        ADD   QTE-1  QTE-2  TO  QTE-TOTALE
         ON SIZE ERROR DISPLAY 'ERREUR QTE-TOTALE'.
        DISPLAY 'QTE-TOTALE:'  QTE-TOTALE
                                                    .
*       CALCUL PRIX-TOTAL
        COMPUTE  PRIX-TOTAL ROUNDED
          = (QTE-1  *  P-UNIT-1) + (QTE-2  *  P-UNIT-2)
              ON SIZE ERROR
                        DISPLAY 'ERREUR PRIX-TOTAL'.
        DISPLAY PRIX-TOTAL .
        STOP RUN.
```

UTILISATION DE PERFORM

```
PROCEDURE  DIVISION.
DEBUT.
        MOVE  ZERO  TO  TOTAL-QTE.
        PERFORM  SAISIE-DONNEES.
        PERFORM  CALCUL-QTE.
```

```
        PERFORM  CALCUL-PRIX.
        STOP  RUN.
SAISIE-DONNEES.
        DISPLAY  'ARTICLE1,  QTE:' .
        ACCEPT  QTE-1  FROM  CONSOLE.
        DISPLAY  'ARTICLE1, P-UNIT-1:'  .
        ACCEPT  P-UNIT-1  FROM  CONSOLE.
        DISPLAY  'ARTICLE2,  QTE:' .
        ACCEPT  QTE-2  FROM  CONSOLE.
        DISPLAY  'ARTICLE2, P-UNIT-2:' .
        ACCEPT  P-UNIT-2  FROM  CONSOLE.

CALCUL-QTE.
        ADD  QTE-1  QTE-2  TO  QTE-TOTALE
          ON  SIZE  ERROR  PERFORM  MESSAGE-ERREUR-1.
        DISPLAY  'QTE TOTALE:'  QTE-TOTALE
                                                        .
CALCUL-PRIX.
        COMPUTE  PRIX-TOTAL  ROUNDED
          = (QTE-1  *  P-UNIT-1) + (QTE-2  *  P-UNIT-2)
            ON  SIZE  ERROR
                  PERFORM  MESSAGE-ERREUR-2.
        DISPLAY  PRIX-TOTAL  .
MESSAGE-ERREUR-1.
        DISPLAY  'ERREUR QTE-TOTALE'  .
MESSAGE-ERREUR-2.
        DISPLAY  'ERREUR PRIX-TOTAL'  .
```


LA STRUCTURE ALTERNATIVE

Quatre types de conditions:

* Condition de Relation
* Condition de Classe
* Condition de Signe
* Condition de Nom-Condition

C'est l'instruction Cobol **IF ELSE** qui permet de traduire l'alternative.

Exemples

Conditions de Relation

```
    IF  CODE  =  "R"
        PERFORM  D-RETRAIT
    ELSE
        DISPLAY "CODE INVALIDE" .
      IF  AGE  NOT GREATER THAN  25 ...
      IF  TOTAL  LESS THAN  TOTAL-2
            NEXT SENTENCE
      ELSE
            PERFORM IMPRESSION.
```

Syntaxe

	{ LESS THAN }	
	{ GREATER THAN}	
	{ EQUAL TO }	
Opérande-1 [IS] [NOT]	**{ > }**	**Opérande-2.**
	{ < }	
	{ = }	

Conditions de Classe

```
    IF CODE-VILLE  IS  NOT  ALPHABETIC
          PERFORM  CONVERTIR-CODE.
    IF  MATRICULE  IS  NOT  NUMERIC ...
```

Conditions de Signe

```
    IF      BALANCE     IS          POSITIVE ...
    IF      SOLDE       IS  NOT     NEGATIVE ...
    IF      SOLDE       IS          ZERO
```

Conditions de Nom-Condition

```
    IF  END-OF-PAGE  PERFORM  CHANGE-PAGE.
    IF  DEPOT  PERFORM  OPER-DEP.
```

Exemples

1.1

```
77    ETAT-CIVIL   PIC    9.
.....
IF    ETAT-CIVIL = 1
      PERFORM  TRAIT-CELIB.
IF    ETAT-CIVIL = 2
      PERFORM  TRAIT-MARIE.
IF    ETAT-CIVIL = 3
      PERFORM  TRAIT-DIVORCE.
IF    ETAT-CIVIL = 4
      PERFORM  TRAIT-VEUF.
```

1.2

```
77    ETAT-CIVIL  PIC  9.
      88    CELIBATAIRE        VALUE       1.
      88    MARIE        VALUE       2.
      88    DIVORCE            VALUE       3.
      88    VEUF         VALUE       4.
.....
IF    CELIBATAIRE
      PERFORM     TRAIT-CELIB.
IF    MARIE
      PERFORM     TRAIT-MARIE.
IF    DIVIRCE
      PERFORM     TRAIT-DIVIRCE.
IF    VEUF
      PERFORM     TRAIT-VEUF.
```

Exemples

1.1

```
IF  TOTAL  IS  LESS  THAN  TOTAL-P
    COMPUTE  TOTAL = TOTAL + TAUX * TOTAL-P
    MOVE "A"   TO  COD-OPER
    DIVIDE  TOTAL  BY  N   GIVING   TOT-N
ELSE
    NEXT  SENTENCE.
DISPLAY  TOT-N .
```

1.2

```
IF   TOTAL  IS  LESS  THAN  TOTAL-P
```



```
    PERFORM  TRAIT-TOT-P
ELSE
    NEXT  SENTENCE.
DISPLAY   TOT-P .
```

4.

```
05    COTES        PIC    99.
      88    A      VALUE       86      THRU  100.
      88    B      VALUE       71      THRU  85.
      88    C      VALUE       60      THRU  70.
      88    D      VALUE       50      THRU  59.
      88    E      VALUE       40      THRU  49.
      88    F      VALUE       0       THRU  39.
```

5.

```
      88    NOUV-COMPTE      VALUE  IS  1.
      88    CODE             VALUE          0.
      88    PERIODE          VALUES ARE 16 THRU 20.
      88    VACANCES         VALUE 'MARS' 'AVRIL' '.
      88    FLAG             VALUE  'AA'  THRU  'AD'.
      88    MAT              VALUE  200 THRU 250
                                    282   285.
```

OPERATEURS LOGIQUES

Opérateur unaire: NOT

Opérateurs binaires: AND OR

LES CONDITIONS IMBRIQUEES

Exemple 1.

```
IF      c1
        …       (pas de point à ce niveau)
ELSE IF C2
        …       (pas de point à ce niveau)
ELSE IF C3
        …       (pas de point à ce niveau)
ELSE
        …       .(point )
```

Exemple 2.

```
IF      c1
        …       (pas de point à ce niveau)
```

```
ELSE IF C2
        …       (pas de point à ce niveau)
        END-IF          (de c2)
        …       (suite de else de c1)
        IF C3
                …
        ELSE
                …
        END-IF (fin de if de c3)
END-IF (ou point)
```


LES ÉNONCÉS RÉPÉTITIFS

1: PERFORM ..... UNTIL
2: PERFORM ..... TIMES
3: PERFORM ..... VARYING

1: PERFORM ..... UNTIL

Syntaxe

PERFORM nom-paragraphe UNTIL condition.

Le fonctionnement de cette instruction correspond au même organigramme que l'instruction WHILE de Pascal avec le complémentaire de la condition.

```
PERFORM B-CALCUL UNTIL STOP .

MOVE ZERO TO FIN.
PERFORM BLOC-INSTRUCTIONS UNTIL FIN EQUAL 1.


PERFORM MODULE-INTERNE
        UNTILFIN-OPERATION
                OR      FLAG GREATER THAN FLAG-IN
                OR      TOTAL EQUAL 25 000.
```

2: PERFORM ..... TIMES

Syntaxe

PERFORM nom-paragraphe {Identifiant} {Entier} TIMES.

Identifiant est un entier (sans point décimal). Il indique le nombre de fois que le module nom-paragraphe sera exécuté. La valeur zéro peut être affectée à Identifiant. Dans ce cas, PERFORM ne provoquera pas l'exécution du module.

```
PERFORM CALCUL-SALAIRE 20 TIMES.

PERFORM SAISIE-DONNEES NRE-DE-FOIS TIMES.
```

3: PERFORM ..... VARYING

Syntaxe

```
PERFORM   nom-paragraphe
      VARYING     Identifiant1
            FROM        {Littéral1/Identifiant2}
            BY          {Littéral2/Identifiant3}
            UNTIL Condition-1

[     AFTER Identifiant4
            FROM        {Littéral3/Identifiant5}
            BY          {Littéral4/Identifiant6}
            UNTIL Condition-2           ]

[     AFTER Identifiant7
            FROM        {Littéral5/Identifiant8}
            BY          {Littéral6/Identifiant9}
            UNTIL Condition-3           ].
```

```
PERFORM  TEST-ARTICLES
      VARYING NRE-ARTICLE FROM 1 BY 1
      UNTIL NBRE-ARTICLE > 200.
```

```
PERFORM  PARCOURS-TABLE
      VARYING INDICE FROM DEBUT BY STEP
      UNTIL INDICE GREATER THAN FIN.
```

```
PERFORM PARCOURS-DIM-2
            VARYING IND-1 FROM DEB-1 BY STEP-1
            UNTIL IND-1 GREATER THAN FIN-1
AFTER
            VARYING IND-2 FROM DEB-2 BY STEP-2
            UNTIL IND-2 GREATER THAN FIN-2.
```

```
PERFORM  MODULE-LISTER
      VARYING INDICE-1 FROM 5 BY 2
            UNTIL INDICE-1 GREATER THAN 120
      AFTER
            INDICE-2 FROM 25 BY 5
```



```
            UNTIL INDICE-2 GREATER THAN 75.
```

```
indice-1 <-- 5
Tant que ( indice-1 <= 120 )
Faire
      indice-2 <-- 25
      Tant que ( indice-2 <= 75 )
      Faire
            MODULE-LISTER
            indice-2 <-- indice-2 + 5
      FinFaire
      indice-1 <-- indice-1 + 2
FinFaire
```

LES TABLEAUX

Syntaxe

niveau **NOM-TABLE OCCURS** entier **TIMES.**

Exemple1

```
01 TABLE-DES -NOTES.
   05 NOTE PIC 9(3)V99 OCCURS 7 TIMES.
```

La même table peut être déclarée comme suit:

```
01 TABLE-DES-NOTES.
   05 ELEMENT OCCURS 7 TIMES.
      10 NOTE PIC 9(3)V99 .
```

Exemple2

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01 TABLES-ETUDIANTS.
   05 ETUDIANT OCCURS 100 TIMES.
      10 NOM PIC X(20).
      10 PRENOM PIC X(15).
      10 ADRESSE PIC X(40).
      10 TEL PIC X(10).
```

Accès aux différents éléments d'une table

L'accès aux composantes d'une table se fait à l'aide d'un compteur ou d'un indice.

Etudiant(1), Etudiant(i), Etudiant(65),
Nom(1), Nom(j), …
Adresse(1), Adresse(j), … Adresse(100).

La Clause REDEFINES

On pourra donner une valeur initiale à une table en procédant en deux temps:

- Définir une ou plusieurs zones de mémoire et leur attribuer une valeur initiale.

- Redéfinir la même zone de mémoire, au moyen de la clause **REDEFINES**, en la décrivant comme étant une table.

La clause **REDEFINES** permet d'utiliser une même zone de mémoire pour recevoir des données de définition différentes.

Format général

niveau nom-donnée-1 **REDEFINES** nom-donnée-2

Exemple1


```
DATA DIVISION.
WORKING-STORAGE SECTION.
01 TABLE-CALENDRIER.
   05 VALEURS-INITIALES.
      10 FILLER PIC X(15) VALUE 'JANVIER'.
      10 FILLER PIC X(15) VALUE 'FEVRIER'.
      .....
      10 FILLER PIC X(15) VALUE 'DECEMBRE'.
   05 MOIS REDEFINES VALEURS-INITIALES PIC X(15) OCCURS 12 TIMES.
```

Exemple2

```
01 TABLE-DES-NOTES.
   05 LISTES-DES-NOTES.
      10 FILLER PIC X(30) VALUE 'SYSTEME EXPLOITATION'.
      10 FILLER PIC X VALUE 'A'.
      10 FILLER PIC X(30) VALUE 'TELEINFORMATIQUE'.
      10 FILLER PIC X VALUE 'C'.
      10 FILLER PIC X(30) VALUE 'I.A. 1'.
      10 FILLER PIC X VALUE 'E'.
   05 NOTES REDEFINES LISTES-DES-NOTES OCCURS 3 TIMES.
      10 COURS PIC X(30).
      10 COTE PIC X.
```

Les deux zones mémoires **nom-donnée-1** et **nom-donnée-2** doivent être de même taille.

2. Procédure d'initialisation

```
DATA DIVISION.
WORKING-STORAGE SECTION.
77 INDICE PIC 9(3) COMP.
77 V-ART PIC X(20).
77 V-QTE PIC 9(6)V99 COMP.
01 TABLE-DES-ARTICLES.
   05 ARTICLES OCCURS 4 TIMES.
      10 LIBELLE PIC X(20).
      10 QTE-STOCK PIC 9(6)V99 COMP.
PROCEDURE DIVISION.
   ......
   .....
   PERFORM INIT-TABLE
      VARYING INDICE FROM 1 BY 1
         UNTIL INDICE GREATER THAN 4.
   ........
   ........
```

```
    STOP  RUN.
INIT-TABLE.
    DISPLAY  'NOM ARTICLE: ' .
    ACCEPT   V-ART    FROM   CONSOLE.
    DISPLAY   'QTE:'   UPON   CONSOLE.
    ACCEPT   V-QTE    FROM  CONSOLE.
    MOVE   V-ART   TO    LIBELLE ( INDICE ).
    MOVE   V-QTE   TO   QTE-STOCK ( INDICE ).
```

Parcours d'une Table

1. Parcours Partiel : Recherche d'un élément dans une table

```
DATA DIVISION.
WORKING-STORAGE SECTION.
77  I     PIC      9(5)      COMP.
01  VL-TROUVE   PIC 9.
    88  TROUVE   VALUE     1.
01  TABLE-PERSONNE.
    05  IDENTITE OCCURS   100 TIMES.
        10  NOM    PIC X(20).
        10  PRENOM    PIC     X(15).
        10  SEXE         PIC     X.
        10  AGE          PIC     9(3) COMP.
PROCEDURE DIVISION.
    PERFORM RECHERCHE-ELEMENT
        VARYING I     FROM    1   BY   1
            UNTIL    I    GREATER    THAN   100
                OR   TROUVE.
    IF  TROUVE
        DISPLAY   NOM ( I )   .
    STOP  RUN.
RECHERCHE-ELEMENT.
    IF    SEXE  ( I )   EQUAL   'F'
        AND    AGE  LESS THAN 40
            MOVE    1    TO  VL-TROUVE.
```

2. Parcours Total

```
DATA DIVISION.
WORKING-STORAGE SECTION.
77  TOTAL-SALAIRE  PIC     9(12)V99.
77  I     PIC      9(5)      COMP.
01  TABLE-EMPLOYES.
```



```
    05  EMPLOYES     OCCURS 100 TIMES.
        10  NAS     PIC X(9).
        10  SALAIRE   PIC     9(6)V99      COMP.
PROCEDURE DIVISION.
    MOVE   0   TO  TOTAL-SALAIRE.
    PERFORM CUMUL-SALAIRE
        VARYING I  FROM   1  BY  1
            UNTIL   I   GREATER   THAN   100.
CUMUL-SALAIRE.
    ADD    SALAIRE  ( I )   TO   TOTAL-SALAIRE.
```

ALGORITHME TRI_ECHANGE (T[1..N] ---> T[1..N])

```
DEBUT
/*Cet algorithme trie en ordre croissant un tableau T de N éléments */
    Pour   I = 1 à N-1  Faire
    Debut
        Indice_min = I
        Pour  J = I + 1  à  N   Faire
        Debut
            Si  T[J] < T[Indice_Min]
                Alors
                    Indice_min = J
            Finsi
        Fin
        Si Indice_Min  I
            Alors
                Tampon = T [ I ]
                T [ I ] = T [Indice_Min]
                T [Indice_Min] = Tampon
        Finsi
    FIN
```

TABLEAUX À DEUX DIMENSIONS

Exemple 1 (Tableau à éléments simples)

Déclaration d'une matrice d'entiers de 3x4 = 12 éléments.

```
01  MATRICE-ENTIERS.
    05  LIGNE OCCURS  3   TIMES.
        10  COLNNE    OCCURS  4   TIMES.
            15   ELEMENT PIC      99  COMP.
```

ELEMENT (1 1)	**ELEMENT (1 2)**	**ELEMENT (1 3)**	**ELEMENT (1 4)**
ELEMENT (2 1)	**ELEMENT (2 2)**	**ELEMENT (2 3)**	**ELEMENT (2 4)**

ELEMENT (3 1)	ELEMENT (3 2)	ELEMENT (3 3)	ELEMENT (3 4)

Accès

L'accès aux différents éléments d'une table à deux dimensions se fait à l'aide de deux indices qui gèrent les lignes et les colonnes de la matrice.

Les indices doivent être numériques, sur 5 positions au maximum et d'usage COMP.

Exemple

```
DATA DIVISION.
WORKING-STORAGE SECTION.
77 I   PIC    9(5) COMP.
77 J   PIC 9(5)    COMP.

01 TABLE-COURS.
   05 COURS    OCCURS 3   TIMES.
      10 ID     PIC   X(5).
      10 NOM      PIC    X(20).
      10 NOTES PIC    9(3)V99     COMP OCCURS 4 TIMES.
```

ID (1)	NOM (1)	**NOTES (1 1)**	**NOTES (1 2)**	**NOTES (1 3)**	**NOTES (1 4)**
ID (2)	NOM (2)	**NOTES (2 1)**	**NOTES (2 2)**	**NOTES (2 3)**	**NOTES (2 4)**
ID (3)	NOM (3)	**NOTES (3 1)**	**NOTES (3 2)**	**NOTES (3 3)**	**NOTES (3 4)**

ID (I) : accéder à l'identifiant du I^{ème} cours dans la table.

NOM (I) : accéder au nom du I^{ème} cours dans la table.

NOTES (I J) : accéder à la J^{ème} note du I^{ème} cours dans la table.

Initialisation

1. La clause REDEFINES

Dans le cas où les différentes valeurs que peut prendre une table sont connues à l'avance, l'utilisation de la redéfinition d'une zone mémoire est recommandée.

Exemple

```
01 TABLE-DES-ENTIERS.
   05 LES-VALEURS-INITIALES.
      10 FILLER.
         15 FILLER    PIC   9(3)V99    COMP VALUE   10.2.
         15 FILLER    PIC   9(3)V99    COMP VALUE   77.1.
         15 FILLER    PIC   9(3)V99    COMP   VALUE  11.4.
```



```
         15 FILLER    PIC   9(3)V99    COMP VALUE   15.61.
      10 FILLER.
         15 FILLER    PIC   9(3)V99    COMP VALUE   23.2.
         15 FILLER PIC 9(3)V99    COMP   VALUE   49.67.
         15 FILLER    PIC   9(3)V99    COMP VALUE   58.4.
         15 FILLER    PIC   9(3)V99    COMP VALUE   94.8.
   05    MATRICE-ENTIERS REDEFINES
         LES-VALEURS-INITIALES.
      10 LIGNE OCCURS 2   TIMES.
         15    COLONNE    OCCURS 4   TIMES.
               20    ELEMENT PIC 9(3)V99    COMP.
```

Le contenu de la table sera comme suit:

010.20	077.10	011.40	015.61
023.20	049.67	058.40	094.80

ELEMENT (2 3) aura donc pour valeur 058.40.

ELEMENT (1 4) aura pour valeur 015.61.

2. Initialisation avec PERFORM .. VARYING .. UNTIL.

```
DATA DIVISION.
WORKING-STORAGE SECTION.
77 I   PIC 9(5)    COMP.
77 J   PIC 9(5)    COMP.
77 A   PIC 9(3)V99.
01 TABLE.
   05 LIGNE OCCURS 2   TIMES.
      10 COLNNE  OCCURS 4   TIMES.
         15 VALEUR  PIC 9(3)V99    COMP.
PROCEDURE DIVISION.
   ....
   ...
   PERFORM INIT-TABLE
      VARYING I   FROM 1   BY 1
         UNTIL I   GREATER THAN 2
   AFTER
      J   FROM 1   BY1
         UNTIL J   GREATER THAN 4.
   ....
   STOP RUN.
INIT-TABLE.
   ACCEPT   A   FROM CONSOLE.
```

```
        MOVE A   TO  VALEUR ( I J ).
```

Parcours d'une table à deux dimensions.

1. Parcours Partiel: Recherche d'un élément dans une table

```
    DATA DIVISION.
    WORKING-STORAGE SECTION.
    77  I    PIC 9(5)     COMP.
    77  J    PIC 9(5)     COMP.
    77  A    PIC 9(3)V99.
    01  MATRICE.
        05  LIGNE OCCURS   5    TIMES.
            10  COLNNE   OCCURS   6    TIMES.
                15  VALEUR   PIC 9(3)V99      COMP.
    01  VL-TROUVE   PIC     9.
        88  TROUVE    VALUE     1.
    PROCEDURE DIVISION.
        ....
        PERFORM CHERCHER-VALEUR
            VARYING  I    FROM  1    BY  1

                UNTIL  I    GREATER THAN 5  OR TROUVE
        AFTER
            J    FROM  1    BY1
                UNTIL  J    GREATER THAN 6 OR TROUVE.
        ....
        STOP RUN.
    CHERCHER-VALEUR.
        IF   VALEUR (I J)  EQUAL  A
            MOVE 1     TO VL-TROUVE.
```

2. Parcours Total: Afficher le contenu de la table

```
    DATA DIVISION.
    WORKING-STORAGE SECTION.
    77  I    PIC      9(5)     COMP.
    77  J    PIC      9(5)     COMP.
    01  TABLE-COURS.
        05  COURS     OCCURS   7    TIMES.
            10  ID        PIC X(5).
            10  NOM          PIC     X(20).
            10  NOTESPIC 9(3)V99     COMP
```



```
PROCEDURE DIVISION.
    ....
    ...
    PERFORM TRAITER-COURS
            VARYING  I    FROM  1    BY  1
                UNTIL  I    GREATER   THAN   7.
    STOP RUN.
TRAITER-COURS.
    DISPLAY 'IDENTIFIANT:'  ID (I) .
    DISLPLAY 'NOM DU COURS:' NOM (I) .
    PERFORM AFFICHER-NOTE
            VARYING  J    FROM  1    BY  1
                UNTIL  J    GREATER  YHAN 4.
AFFICHER-NOTE.
   DISPLAY 'NOTE' J 'EST: ' NOTES (I J) .
```

Les Index

Cobol offre la possibilité de remplacer les indices utilisés pour accéder aux éléments d'une table par des index.

Un index joue pratiquement le même rôle qu'un indice néanmoins qu'il est implicitement réservé par Cobol lors de la déclaration de la table.

Cette Partie ne sera pas traité durant cette formation.

Syntaxe

.... OCCURS entier TIMES
[INDEXED BY nom-index [nom-index2]...]

Exemple 1

```
77  I PIC  9(5)  COMP.
01  TABLEAU.
    05 ELEMENT PIC X(2) OCCURS 5 TIMES.
```

L'indice I déclaré au niveau 77 permet de parcourir la table TABLEAU et d'accéder à ses différents éléments.

La même table et l'indice I peuvent être déclarés comme suit à l'aide des index:

```
01  TABLEAU.
    05 ELEMENT PIC X(2) OCCURS 5 TIMES
              INDEXED BY I.
```


LES SOUS-PROGRAMMES

Avec l'instruction PERFORM, nous avons vu comment programmer des parties répétitives sous forme de séquences indépendantes.

On pourrait aussi écrire de telles séquences sous forme de programmes indépendants appelés par un programme principal.

Les sous-programmes externes sont donc des programmes écrits en langage Cobol, ou tout autre langage, et compilés à part du programme principal.

APPEL D'UN SOUS-PROGRAMME

Le branchement du traitement depuis le programme principal Cobol vers le sous-programme est réalisé par l'instruction CALL.

Syntaxe

CALL nom-donnée [USING {BY REFERENCE / BY CONTENT} nom-donnée1...nom-donnéen]

nom-donnée: désigne le nom du sous-programme appelé, nom qui doit respecter la règle de formation des noms des programmes, c-à-d 8 caractères dont le premier est obligatoirement alphabétique.

Exemple

1. Programme appelant

```
IDENTIFICATION DIVISION.
PROGRAM-ID. PRGA.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  ZON1.
    05      COD1  PIC X(2).
    05      VAL1  PIC 9(2)  COMP  OCCURS  22 TIMES.
01  ZON2.
    05      COD2  PIC  X(3).
    05      VAL2  PIC 9(3)V99  COMP.
PROCEDURE DIVISION.
    ........
    CALL  'SOUSPROG'  USING  ZON1  ZON2.
    ....
    ...
    STOP RUN.
```

2. Sous-programme appelé

```
IDENTIFICATION DIVISION.
```

```
PROGRAM-ID. SOUSPROG.
ENVIRONMENT DIVISION.
DATA  DIVISION.
WORKING-STORAGE SECTION.
.........
.......
LINKAGE SECTION.
01  ZON1.
    05      COD1  PICX(2).
    05      VAL1   PIC  9(2)   OCCURS  22 TIMES.
01  ZON2.
    05      COD2  PIC     X(3).
    05      VAL2  PIC     9(3)V99 .
PROCEDURE DIVISION USING  ZON1   ZON2.
    ........
    ...
    ....
    EXIT  PROGRAM.
```

* USING {BY REFERENCE/ BY CONTENT }

1. CALL 'SOUSPROG' USING A B C D.

La clause BY REFERENCE est par défaut.

Cette clause signifie que le programme appelant et le sous-programme appelé ont en commun une zone de données ou plusieurs zones.

CALL 'SOUSPRG' USING BY REFERENCE A B C D.

est équivalent à

CALL 'SOUSPROG' USING A B C D.

Il est impératif que la description des paramètres soit identique dans les deux programmes.

Et puisque ces zones de données sont communes (BY REFERENCE), le sous-programme appelé peut très bien modifier le contenu des zones.

Par exemple, on peut appeler un sous-programme de conversion de date du format AADDD en AAMMJJ, en écrivant CALL 'CONVDAT' USING DAT1 DAT2.

On passe au sous-programme la valeur contenue dans DAT1 qui renvoie au programme appelant la valeur correspondante AAMMJJ dans DAT2.

CALL 'SOUSPROG' USING BY CONTENT A B C D.

BY CONTENT est sensiblement identique à BY REFERENCE, si ce n'est que le sous-programme ne peut modifier le contenu des variables transmises comme paramètres.

LE SOUS-PROGRAMMES

Un sous-programme Cobol est bâti comme tout autre programme Cobol avec les quatres divisions usuelles:

IDENTIFICATION DIVISION.

ENVIRONMENT DIVISION.

DATA DIVISION.

PROCEDURE DIVISION.

La différence avec le programme Cobol normal réside dans l'introduction de la clause USING liste- des-paramètres avec la PROCEDURE DIVISION et la LINKAGE SECTION après la WORKING-STORAGE SECTION dans la DATA DIVISION.

1. LINKAGE SECTION

....

DATA DIVISION.

WORKING-STORAGE SECTION.

....

LINKAGE SECTION.

Description des paramètres déclarés dans la working-storage section du programme appelant.

.....

PROCEDURE DIVISION USING $param_1$ $param_2$... $param_n$.

.....

EXIT PROGRAM.

Le programme appelant et le sous-programme étant compilé séparément, il est nécessaire que la description des paramètres communs apparaisse dans les DATA DIVISION respectives des programmes.

LES FICHIERS

1. DESCRIPTION D'UN FICHIER

L'organisation générale de la DATA DIVISION est la suivante:

```
        DATA DIVISION.
-->     FILE SECTION.
        FD    Nom-du-fichier
              Description des blocs, records, label, linage.
        01    Nom-de-l'enregistrement.
              Description détaillée de l'enregistrement.
        WORKING-STORAGE SECTION.
        ....
```

C'est dans la FILE SECTION de la DATA DIVISION que doivent être décrites toutes les informations concernant les fichiers utilisés dans le programme.

Syntaxe

```
DATA DIVSION.
FILE SECTION.
FD Nom-du-fichier
   [ BLOCK CONTAINS Entier-1 {CHARACTERS / RECORDS} ]
   [ RECORD CONTAINS Entier-2 [ TO Entier-3 ] CHARACTERS ]
   [ LABEL RECORD IS {OMITTED / STANDARD}]
   [ DATA RECORD IS Nom-enrg]
   [ LINAGE IS {Nom-donnée / Entier-4} LINES
   [ WITH FOOTING AT {Nom-donnée-2 / Entier-5} ]
   [ LINES AT TOP {Nom-donnée-3 / Entier-6 } ]
   [ LINES AT BOTTOM { Nom-donnée-4 / Entier-7}]].
```

*** FD** Nom-du-fichier

FD est l'abréviation de FILE DESCRIPTION (Description de fichiers) et Nom-du-fichier est un nom symbolique que le programmeur affecte au fichier considéré.

Exemple:

```
FD STOCK ou FD CLIENT.
```

*** BLOCK CONTAINS** Entier-1 **{CHARACTERS / RECORDS}**

Cette clause permet de définir que le fichier a des enregistrements bloqués afin d'utiliser au mieux l'espace disque ou bande et d'accélerer le traitement.

Les enregistrements d'un fichier sont bloqués selon un nombre d'enregistrements logiques par bloc.

Ce nombre est appelé Facteur de blocage.

Le facteur de blocage est généralement défini par le système.

Dans ce qui vient, nous verrons comment définir le facteur de blocage afin de bloquer les enregistrements logiques d'un fichier en enregistrements physiques.

*** RECORD CONTAINS** Entier-2 [TO Entier-3] **CHARACTERS**

Cette clause sert à indiquer la longueur des enregistrements logiques. Lorsqu'on indique Entier-2, par exemple: RECORD CONTAINS 100 CHARACTERS, c'est que le fichier contient des enregistrements de longueur fixe (100 octets).

Indiquer Entier-2 plus Entier-3 par exemple: RECORD CONTAINS 100 TO 256 CHARACTERS signifie que le fichier est de structure variable avec des enregistrements de longueur comprise entre 100 et 256 octets.

Le compilateur compile la dimension sur les structures, mais si la clause est indiquée, la priorité est donnée à celle-ci en cas de divergence.

*** LABEL RECORD IS {OMITTED / STANDARD}**

Les labels sont des enregistrements spéciaux destinés à identifier avec précision les fihiers.

OMITTED: signifie que le fichier n'a pas de label. Les fichiers d'impression (fichier états) n'aurons jamais de labels.

STANDARD: spécifie que les labels sont standards, ce qui signifie que le programmeur laisse au système d'exploitation le soin de bâtir et contrôler les labels.

*** DATA RECORD IS** Nom-enrg

Cette clause non obligatoire désigne la structure d'enregistrement associé au fichier.

Exemple 1

```
DATA DIVISION.
FILE SECTION.
FD CLIENT
   BLOCK CONTAINS 5 RECORDS
   RECORD CONTAINS 86 CHARACTERS
   LABEL RECORD IS STANDARD
   DATA RECORD IS ENRG-CL.

01 ENRG-CL.
   05 ID-CL PIC X(5).
   05 NOM-CL PIC X(20).
   05 PNOM-CL PIC X(15).
   05 DAT-NAIS.
      10 JJ-CL PIC X(2).
      10 MM-CL PIC X(2).
      10 AA-CL PIC X(2).
   05 ADR-CL PIC X(40).
```

Le fichier CLIENT est composé d'enregistrements de taille fixe (**RECORD CONTAINS** 86 **CHARACTERS**): enregistrements logiques.

- Ces enregistrements sont regroupés par 5 (**BLOCK CONTAINS** 5 **RECORDS**)

- Un enregistrement physique (bloc) contient 5 enregistrements logiques d'où le facteur de blocage = 5.
- Le label du fichier est géré par le système d'exploitation (**LABEL RECORD IS STANDARD**).
- L'enregistrement du fichier CLIENT est ENRG-CL (**DATA RECORD IS** ENRG-CL).

Exemple 2

```
DATA DIVISION.
FILE SECTION.
FD COURS
    BLOCK CONTAINS 5 RECORDS
    RECORD CONTAINS 32 TO 50 CHARACTERS
    LABEL RECORD IS STANDARD
    DATA RECORD IS ENRG-CRS.
01  ENRG-CRS.
    05 ID-CRS      PIC     X(5).
    05 NOM-CRS PIC     X(20).
    05 NOTES-CRS.
       10      NB-NOTE  PIC    9(5).
       10      NOTES  PIC 9(2) OCCURS  1 TO 10
               DEPENDING ON NB-NOTE.
```

Le fichier COURS est composé d'enregistrements de taille allant de 32 à 50 octets.

La clause **BLOCK CONTAINS** 5 **RECORDS** peut être remplacé par **BLOCK CONTAINS** 160 **TO** 250 **CHARACTERS** où 160 = 5 Records/Bloc * 32 caractères/Record et 250 = 5 Records/Bloc * 50 caractères/Record.

Exemple 3

```
DATA DIVISION.
FILE SECTION.
FD      ETAT
        BLOCK CONTAINS 1 RECORD
        RECORD CONTAINS 132
        LABEL RECORD IS OMITTED
        DATA RECORD IS LIGNE-ET.
01      LIGNE-ET.
        05      LIGNE PIC      X(132).
```

Le fichier ETAT est un fichier d'impression (**LABEL RECORD IS OMITTED**).

Chaque enregistrement est une ligne de 132 caractères.

Avec les fichiers d'impression, on peut définir la clause LINAGE.

LINAGE spécifie une page logique qui comprend un haut de page, un bas de page et un corps de page à l'interieur duquel se fait l'impression elle même.

[**LINAGE IS** {Nom-donnée / Entier-4} **LINES**

[**WITH FOOTING AT** {Nom-donnée-2 / Entier-5}]
[**LINES AT TOP** {Nom-donnée-3 / Entier-6 }]
[**LINES AT BOTTOM** {Nom-donnée-4 / Entier-7}]].

Entier-4: Nombre maximum de lignes écrites pour chaque page de l'état.

Entier-5: Désigne le numéro de ligne à partir de laquelle seront éditées les lignes de bas de page (FOOTING).

On doit respecter 0<Entier-5<Entier-4.

Entier-6: Désigne le nombre de lignes non écrites en tête de page (TOP).

Entier-7: Désigne le nombre de lignes non écrites en bas de page (BOTTOM).

Exemple

```
DATA DIVISION.
FILE SECTION.
FD      ETAT
        BLOCK CONTAINS 1 RECORD
        RECORD CONTAINS 132
        LABEL RECORD IS OMITTED
        DATA RECORD IS LIGNE-ET
        LINAGE IS 54 LINES
                WITH FOOTING AT 44
                LINES AT TOP 6
                LINES AT BOTTOM 6.
01      LIGNE-ET.
        05      LIGNE PIC      X(132).
```

Nous verrons plus loin l'utilisation de l'ordre WRITE en concordance avec la clause LINAGE.