

***www.Mcours.com***

Site N°1 des Cours et Exercices Email: [contact@mcours.com](mailto:contact@mcours.com)

# **INTRODUCTION AU LANGAGE COBOL**

1	Introduction .....	2
1.1	Objectif .....	2
1.2	Dans le département .....	2
1.3	Exemples .....	2
2	Structure d'un programme Cobol.....	2
2.1	La division d'identification .....	2
2.2	La division environnement .....	2
2.3	La division de données .....	2
2.4	La division programme.....	2
2.5	Structure de la ligne .....	3
3	Un premier exemple .....	3
3.1	Déclarations .....	3
3.2	Formats .....	3
3.2.1	Données numériques .....	3
3.2.2	Données d'édition .....	3
3.2.3	Données caractères .....	3
3.3	Les instructions.....	3
3.4	Les fichiers séquentiels.....	4
3.4.1	Déclaration .....	4
3.4.2	Organisation séquentielle, par défaut .....	4
3.4.3	ouverture.....	4
3.4.4	Lecture.....	4
3.4.5	Ecriture .....	4
3.4.6	Fermeture .....	5
4	Deuxième exemple .....	5
4.1	Tableaux .....	5
4.1.1	La clause OCCURS.....	5
4.1.2	La clause Redefines.....	5
4.2	Les fichiers séquentiels indexés .....	5
4.2.1	Principe.....	5
4.2.2	En Cobol.....	5
4.2.2.1	Déclaration .....	5
4.2.2.2	Les différents modes d'accès .....	5
4.2.2.3	Les primitives.....	6

# 1 Introduction

## 1.1 OBJECTIF

Le but de ce chapitre est de savoir lire ou modifier du code Cobol ; car 47% des programmes exécutables utilisés de nos jours sont issus de code cobol.

Le langage Cobol (Comon Business Langage, créé en 1957 par l'armée américaine, dernière mise à jour : 1985 : ANSI) est adapté à la gestion des fichiers relatifs et des fichiers séquentiels indexés. Il sert donc à tout ce qui est manipulations de fichiers.

## 1.2 DANS LE DEPARTEMENT

Sous linux :

- compilateur : Acucobol-GT v5.1 : pour compiler : `$>ccbl fich.cbl`  
Génère un fichier intermédiaire `fich.acu`
- Pour exécuter ce fichier :  
`$>RUNCBL fich.acu`

## 1.3 EXEMPLES

`$>ccbl -o fich -e fich.err fich.cbl`

L'option `-e fich.err` permet de récupérer les erreurs de compilations dans un fichier.

`$>ccbl -Zd fich.cbl` : utilisation du débogueur

# 2 Structure d'un programme Cobol

Un programme Cobol a une structure très hiérarchisée :

Chaque programme est partagé en 4 divisions. Ces divisions sont elles-même partagées en sections, divisées en paragraphes, divisées en phrases.

## 2.1 LA DIVISION D'IDENTIFICATION

IDENTIFICATION DIVISION.

PROGRAM-ID. Monpremier. {Identification nominale : elle est obligatoire}

AUTHOR. Moi. {Identification auteur : facultative}

## 2.2 LA DIVISION ENVIRONNEMENT

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

INPUT-OUTPUT SECTION. {Pour définir les fichiers qui seront utilisés}

## 2.3 LA DIVISION DE DONNEES

DATA DIVISION.

FILE SECTION. {Pour la déclaration des enregistrements de fichiers.}

WORKING-STORAGE SECTION. {déclarations}

## 2.4 LA DIVISION PROGRAMME

PROCEDURE DIVISION

PRINCIPAL. {Correspond à la fonction principale. Il peut y avoir d'autres fonctions}  
.....{Programme}.....  
STOP RUN.

## 2.5 STRUCTURE DE LA LIGNE

Une ligne comporte 80 caractères :

- Les caractères 1 à 6 servent à définir le numéro de ligne
- Le caractère 7 : '\*' pour définir une ligne de caractère
- Les caractères 8 à 12 servent à nommer la division, la section ou le paragraphe
- Les caractères 13 à 80 servent à inscrire les instructions programme.

# 3 Un premier exemple

Voir le polycopié ci joint.

## 3.1 DECLARATIONS

Les données élémentaires comme les suites de caractères sont définies au niveau 77.

Le format de déclaration est donc le suivant :

77 <identificateur> pic <format>

## 3.2 FORMATS

### 3.2.1 Données numériques

9 : signifie qu'il s'agit d'un chiffre

Donc 9(4) : nombre d'au plus 4 chiffres

S9(3) : nombre d'au plus 3 chiffres et signé (positif ou négatif)

9(4)V99 : décimal, d'au plus quatre chiffres pour la partie entière et deux chiffres après.

### 3.2.2 Données d'édition

zzz9.99 : les chiffres notés z ne seront édités que s'ils sont significatifs (c'est à dire pour l'affichage de 28.13 on obtiendra 28.13 plutôt que 0028.13)

zzBzz9.99 : l'affichage par exemple de 10750.26 sera 10\_750.26

### 3.2.3 Données caractères

Le caractère se définit par X : ainsi, X(12) est une ligne de caractères d'au plus 12 caractères.

Données structurées

01 ETUDIANT.

05 NOM PIC X(30).

05 PRENOM PIC X(30).

05 DATE\_NAISS.

10 JOUR PIC 99.

10 MOIS PIC 99.

10 ANNEE PIC 9999.

Attention à l'ordre croissant des numéros et au fait que les lignes de même champs ont les mêmes numéros.

## 3.3 LES INSTRUCTIONS

Quelques exemples d'instructions :

Display : écrire

Accept : lire

Add A to B : B<= A+B

Add A to B giving C : C<=A+B

Compute A=2\*(B-C)+6\*D

Move : déplacement

Perform <nomparamètre> : on va exécuter la ligne portant le nom nomparamètre

Perform <nom1> thru <nom2> : on exécute les paramètres consécutifs de nom1 à nom2

Perform <par> [thru <par2>] until <condition> : on exécute par1 (ou de par1 à par2) jusqu'à condition.

Attention : dans ce cas, la condition est testée avant l'exécution de par1. Cette structure correspond donc à un « tant que non-condition faire par ».

## 3.4 LES FICHIERS SEQUENTIELS

### 3.4.1 Déclaration

ENVIR. DIVISION.

I-O SECTION.

File – control

Select fic-etd assign to fic1.dat.

Fic-etd est le nom logique et fic1.dat est le nom physique.

### 3.4.2 Organisation séquentielle, par défaut

DATA DIVISION

File section.

FD fic-etd.

01 enr-etd.

02 nom pic X(20)

02 note pic 99V99

### 3.4.3 ouverture

OPEN OUTPUT fic-etd (création)

INPUT (lecture)

EXTEND (rajout)

### 3.4.4 Lecture

READ fic-etd

AT END move "V" to weof

END READ

Remarques :

La lecture de l'enregistrement est placée dans enr-etd.

Si on lit « FdF », la clause AT END est exécutée.

⇒ Squelette standard de l'écriture d'un fichier :

MOVE « F » TO WEOF

PERFORM LIT.ENR

PERFORM TRAITEMENT UNTIL WEOF = "F"

LIT.ENR

READ FIC

AT END MOVE "V" to WEOF

END READ.

La déclaration de la variable weof est particulière :

77 weof PIC X

88 eof value « V ».

Remarques : 88 eof est le nom de la condition. Eof signifie Weof= « V »

### 3.4.5 Ecriture

WRITE enr-etd

Attention : il faut mettre le nom logique et pas le nom physique.

### 3.4.6 Fermeture

CLOSE fic-etd.

## 4 Deuxième exemple

Voir le polycopié exemple 2.

### 4.1 TABLEAUX

#### 4.1.1 La clause OCCURS

La clause OCCURS permet de répéter un niveau.

Par exemple :

01 TAB.

    02 ETD OCCURS 100.

        03 Nom pic X(30)

        03 Notes pic 99V99 OCCURS 20.

On suppose qu'il y a 100 étudiants et que chacun d'entre eux a 20 notes.

#### 4.1.2 La clause Redefines

Cette clause permet de redéfinir, c'est à dire définir à nouveau, dans une zone mémoire (voir cours polycopié page 14).

### 4.2 LES FICHIERS SEQUENTIELS INDEXES

#### 4.2.1 Principe

L'idée est la suivante : on a un fichier qui est une collection d'enregistrements. Il est indexé dans la mesure où chaque enregistrement est identifié (de façon unique) par une clé (exemple : NIP, pour les étudiants). La clé fait partie de l'enregistrement (c'est l'un des champs). L'intérêt de cette clé c'est qu'elle permet un accès direct en lecture.

Fonctionnement : le système crée deux fichiers :

- Le fichier INDEX (.ndx) : il contient les clés + une organisation permettant une recherche rapide.
- Le fichier DATA (.dat) : il contient le reste des informations.

#### 4.2.2 En Cobol

##### 4.2.2.1 Déclaration

Select fic-etd assign to « etd-a1 »

Organization indexed {fichier indexé}

Acces random {mode d'accès}

Record key NIP. {clé}

    FD fic-etd

    01 enr-etd

        02 NIP pic 9(7)

        02 Nom ...

    ...

##### 4.2.2.2 Les différents modes d'accès

Il existe trois modes d'accès :

- Accès direct : RANDOM
  - Accès séquentiel : SEQUENTIAL
  - Accès dynamique : DYNAMIC
- La lecture est séquentielle par clé croissante

L'accès dynamique est une sorte de mix des 2 précédents modes.

#### 4.2.2.3 Les primitives

- **Ouverture**

OPEN INPUT fic-etd {ouverture en mode lecture}  
 OUTPUT {ouverture en mode création}  
 I-O {ouverture en mode lecture ou écriture (mise à jour, rajout)}

- **Lecture**

o **Directe :**

Move ? to NIP.  
 READ fic-etd  
     INVALID KEY.../\*instructions\*/ {lecture échouée, clé inexistante}  
     NOT INVALID KEY.../\*instructions\*/ {lecture réussie}

o **Séquentielle**

READ fic-etd NEXT {lecture fenêtre}  
     AT END...  
     NOT AT END...  
 END-READ.

- **Ecriture**

WRITE : enr-etd  
     INVALID KEY... {clé déjà existante}  
     NOT INVALID KEY....  
 END-WRITE

- **Réécriture**

Ce mode ne fonctionne qu'en mode d'ouverture I-O  
 REWRITE enr-etd  
     INVALID KEY. ...  
     NOT INVALID KEY. ...  
 END- REWRITE

Il réécrit le dernier lu mais il est interdit de modifier la clé.

- **Effacement**

Ce mode ne fonctionne qu'en mode d'ouverture I-O  
 Move ? to NIP  
 DELETE fic-etd {Attention : on efface sur le fichier et pas sur l'enregistrement}  
     INVALID KEY  
     NOT INVALID KEY  
 ...