A V E R T I S S E M E N T

Le présent document ne doit pas être considéré comme un manuel de référence du COBOL mais comme un support du cours COBOL.

Certaines parties dans le codage des instructions ont été volontairement omises :

- Par souci de clarté de l'exposé
- Pour éviter les outils incitant à la rédaction de programmes mal structurés
- Parce qu'elles sont obsolètes
- Parce qu'elles ne sont pas (ou peu) utilisées dans la pratique
-

Présentation

1. Historique

En 1959, des représentants des secteurs public et privé se réunissent au Pentagone pour étudier les possibilités d'élaborer un langage de programmation adapté à la résolution des problèmes de gestion de l'Armée; ce langage fut appelé COBOL (Common Business Oriented Language : Langage commun orienté vers la gestion). Sa première version fut opérationnelle en décembre 1959.

Issue de ce groupe de travail, la CODASYL (Conférence On Data System Language) produisit une seconde version en 1961.

Depuis, d'autres organisations, nationales ou internationales, ont réalisé de nouvelles versions, en particulier l'ANSI (American National Standard Institute) et l'ISO (International Standard Institute). Plusieurs versions de langage ont finalement été normalisées: COBOL-68, COBOL-74 (la plus utilisée), COBOL-85 (revue en 89). Malgré cela, il subsiste des différences entre les compilateurs proposés par les constructeurs (syntaxe, extensions propres, intégration dans le système), qui définissent des **dialectes** spécifiques.
COBOL continue son évolution et les dernières versions intègrent les concepts de la programmation orientée objet.

2. Intérêt du COBOL

2.1. Indépendance de la machine utilisée

Les programmes peuvent être facilement transposés d'une machine à une autre.

La majorité des ordinateurs disposent d'un compilateur COBOL, quels que soient leur puissance et le système d'exploitation qu'ils utilisent.

Ceci facilite les changements de configuration; toutefois l'indépendance n'est réelle que si l'on respecte la norme, et donc si l'on s'abstient d'utiliser les extensions propres au constructeur.

2.2. Un langage adapté à la résolution des problèmes de gestion

Ces problèmes se caractérisent par des **volumes importants** de données à traiter; le COBOL intègre la majorité des fonctions de manipulation des fichiers traditionnels

2.3. Apprentissage facile

Les phrases de COBOL sont proches du langage courant anglais ce qui facilite la lecture du programme mais entraîne en contrepartie certaines "longueurs".

2.4 Interfaçage avec d'autres outils

3. Eléments du langage

Tous les éléments surlignés sont spécifiques à la norme ANS85

4.1. Jeu des caractères autorisés:

Il comporte les **caractères** suivants:
- 10 chiffres: 0 1 2 9
- 26 lettres: A B C ..... Z a b c ……. z
- le symbole monétaire: $
- 4 signes arithmétiques: + - * /
- 3 symboles de relation: > < =
- 7 séparateurs utilisés pour la ponctuation: , ; . ~~b~~ ( ) " :
  (Nous notons ~~b~~ le caractère "espace").

Tous ces caractères sont représentés en mémoire par leur code (ASCII ou EBCDIC); le code ASCII natif utilise 7 chiffres binaires (bits) et est complété par un 8ème bit égal à zéro afin de stocker à l'octet et de pouvoir utiliser une représentation hexadécimale.
Sauf dans des littéraux non-numériques, les lettres minuscules sont équivalentes aux majuscules.

4.2. Les mots

Un mot COBOL est une chaîne continue d'au plus **30** caractères pris dans le sous-ensemble { lettres, chiffres, - }. Un tiret (-) ne peut être ni le premier, ni le dernier caractère d'un mot.

On distingue :
- les mots-utilisateurs
- les mots-réservés
- les noms-système

4.2.1. Les mots-utilisateurs

Créés par le programmeur, ce sont des noms servant à référencer des structures de données ou de traitements. On distingue :

- les **noms de données** : ils servent à désigner une zone de mémoire contenant une information :
- les noms de fichiers : permettent de référencer une structure de fichier
- le nom de programme : identifie le programme
- les **noms de paragraphes et de sections** : délimitent des sous-ensembles du programme
- les **noms d'article** : identifient une structure d'enregistrement logique dans un fichier.
- les **noms d'index** : désignent des index associés à une table
- les **numéros de niveaux** : indiquent la hiérarchie des rubriques à l'intérieur d'une structure complexe.

Tous ces mots propres au programmeur ont une **portée limitée au programme** qui les utilisent (et les programmes contenus en cas de globalisation).

Les numéros de niveaux sont uniquement composés de chiffres et appartiennent à l'ensemble {**01** à **49**, **66**, **77**, **88**}.

A l'exception des noms de paragraphes et de sections et des numéros de niveaux, les mots-utilisateurs doivent comporter **au moins une lettre**.

Exemples

Noms de données : `A1  TOTAL-FACTURE  COEFF3`

Nom de paragraphes ou de sections : `S1  CALCUL  71  7-1`

Remarques :

- Les mots cités en exemples de noms de données pourraient aussi être des noms de fichiers ou d'articles ou d'index.

- L'écriture `TOTAL-FACTURE` représente un seul mot à cause du tiret ; TOTAL FACTURE est considéré comme une suite de deux mots.

- Le nom de section `1` est différent du nom `01`

A cette liste, il convient d'ajouter les **constantes (ou littéraux)**

- numériques

- non numérique ou chaînes de caractères

- Les **littéraux numériques** sont limités à 18 chiffres et peuvent contenir un signe (à l'extrême gauche) et une marque décimale (.).

Exemple : `125  -42.1  +21`

Certains dialectes acceptant les données en virgule flottante, un littéral numérique de ce type s'exprime par
[±] mantisse E [±] exposant , mantisse d'au plus 16 chiffres, et exposant sur 2 chiffres (domaine de valeurs de 0.54E-78 à 0.72E+76).

Exemples : 15.2E18 -458E-42

- Les **littéraux non numériques** sont des chaînes d'**au plus 160 caractères** quelconques encadrées de **guillemets** (ou apostrophes suivant dialecte).

Exemple: `"*** Liste des Fournisseurs ***"`

4.2.2. Les mots-réservés

Ces mots ont une signification précise pour le compilateur COBOL que le programmeur ne peut pas modifier. Il est donc prudent lorsqu'on choisit un mot utilisateur de s'assurer qu'il ne s'agit pas d'un mot réservé ; à cet effet consulter la liste en Annexe.

Ces mots ont par ailleurs une orthographe fixée qu'il faut impérativement respecter.

On distingue :

- **les mots-clés** : Ils identifient les clauses et les instructions ; la présence du mot clé identifiant une instruction est indispensable chaque fois que cette instruction est utilisée.

  Exemple : `ADD` identifie le verbe arithmétique de l'addition

- **les mots facultatifs** :Ils ne servent qu'à se rapprocher de la syntaxe de la langue anglaise ; leur absence ne modifie pas le sens de la clause ou de l'instruction.

  Exemple : `DATA RECORD IS ARTICLE1`
  peut s'écrire `RECORD ARTICLE1,` `ARTICLE1` étant un mot utilisateur

- **les conjonctions** : pour relier des opérandes dans les clauses ou instructions : virgule ou point-virgule (facultatifs) ; pour former des expressions conditionnelles composées : AND et OR

- **les opérateurs arithmétiques et de relation** : `+ - * / = > < **`

- **les constantes figuratives** : ces mots sont utilisés pour désigner des constantes particulières (`ZERO, SPACE, HIGH-VALUE, LOW-VALUE, QUOTE, ALL` lit) ; ils seront abordés ultérieurement.

- **les registres spéciaux** : zones de mémoire accueillant des valeurs particulières ; ils sont spécifiques des dialectes ; par exemple en OSVS :
  `CURRENT-DATE` date du jour (MM/DD/YY)
  `RETURN-CODE` code-retour émis par le pg
  `TALLY` compteur associé à `EXAMINE`
  `TIME-OF-DAY` heure-système
  ...

4.2.3. Les noms-système

Ils servent à communiquer avec le système d'exploitation pour désigner une machine, un langage ou un dispositif (`PRINT`, `SWITCH-1, CONSOLE, TERMINAL, SYSIN, SYSOUT,` ...)

4.3. Instructions et clauses

4.3.1. Les instructions

Une instruction est une combinaison syntaxiquement correcte de mots et de symboles; elle figure dans la **division des traitements** et spécifie une action particulière. Elle commence toujours par un **verbe**, mot réservé qui l'identifie.

Exemple :

```
ADD 1 TO C                          Addition
MOVE RESULTAT TO EDITION            Transfert
```

4.3.2. Les clauses

Une clause est une combinaison syntaxiquement correcte de mots et de symboles ; elle permet de spécifier un **attribut** ou une **disposition** particulière. Contrairement aux instructions, les clauses ont un rôle descriptif statique.

Exemple : `USAGE IS DISPLAY` précise le mode de codification utilisé indépendamment des valeurs et des opérations effectuées.

`LABEL RECORD IS STANDARD` indique que pour identifier ce fichier, on utilise la procédure des étiquettes standards.

4.4. Les phrases

Une phrase est une **suite d'instructions ou de clauses terminée par un point**.

4.5. Les paragraphes

Un paragraphe est une suite d'une ou plusieurs phrases précédée par un **en-tête de paragraphe** qui l'identifie.
L'en-tête de paragraphe est un nom de paragraphe suivi d'un point.
Un paragraphe se termine à la rencontre d'un des éléments suivants :

- en-tête d'un nouveau paragraphe
- en-tête de section
- en-tête de division
- fin du programme

Exemple :

```
PAR0. ADD A TO B
      MOVE A TO I.
PAR1. MOVE B TO C.
S2 SECTION.
```

4.6. Les sections

Une section est un ensemble de zéro, un ou plusieurs paragraphes précédé par un **en-tête de section** au format suivant :

nom de section `SECTION`.

Une section se termine à la rencontre d'un des éléments suivants :
- en-tête d'une nouvelle section
- en-tête de division
- fin de programme

4.7. Les divisions

Une division est un ensemble de zéro, une ou plusieurs sections précédé par un **en-tête de division** au format suivant :

nom de division `DIVISION`.

Le nom de division est un mot clé.

Une division se termine à la rencontre d'un en-tête de division ou de la fin du programme.

5. Structure d'un programme

Un programme COBOL comporte **4 divisions obligatoires** (sauf ANS85), présentées dans l'ordre suivant :

5.1. `IDENTIFICATION DIVISION`

Elle permet de préciser des renseignements généraux sur le programme tels que : nom du programme, date d'écriture, nom de l'auteur ...

5.2. `ENVIRONMENT DIVISION`

Elle contient la description de l'ordinateur utilisé (`CONFIGURATION SECTION`) et des informations relatives à la gestion des entrées-sorties (`INPUT-OUTPUT SECTION`).

5.3. `DATA DIVISION`

Elle contient la description des informations ou données que le programme reçoit, traite et produit ; ces données se répartissent en trois sections :

- La `FILE SECTION` contient toutes les informations relatives aux structures de fichiers (**données externes**).

- La WORKING-STORAGE SECTION rassemble toutes les **données internes** au programme (zones de travail, de calcul, tables, etc ...)
- La LINKAGE SECTION contient les **données communes** à plusieurs programmes.
- La COMMUNICATION SECTION décrit l'interface entre le programme et le sous-système de gestion des messages (terminaux locaux et distants).
- La REPORT SECTION permet la spécification et la description des états.

On trouve aussi, dans les dialectes, les SCREEN SECTION et LOCAL-STORAGE SECTION.

5.4. PROCEDURE DIVISION

Elle contient **toutes** les instructions permettant d'effectuer les traitements portant sur les informations spécifiées dans la DATA DIVISION.

Le programme-source se termine à la fin de la dernière ligne, ou à la rencontre de l'en-tête

END PROGRAM nom-programme .

6. Conventions

Pour définir le format des instructions et des clauses de COBOL nous appliquerons les conventions suivantes :

6.1. Tous les **mots** écrits en **majuscules** sont des **mots-réservés** ; les mots écrits en minuscules sont des mots-utilisateurs.

6.2. Les **mots**-réservés **soulignés** sont des **mots-clés** ; les mots-réservés non soulignés sont facultatifs.

6.3. Les **accolades** { } rassemblent les **options** de format, écrites les unes en dessous des autres, parmi lesquelles il faut en choisir une.

Exemple :

```
{ PICTURE }
{         } IS chaîne
{ PIC     }
```

6.4. Les **crochets** [ ] délimitent une **partie** du format **facultative**

Exemple :

```
ACCEPT nd [ FROM { DATE } ]
                 { TIME }
```

Ce format autorise les trois instructions

```
ACCEPT V1
ACCEPT V1 FROM DATE
ACCEPT V1 FROM TIME
```

V1 étant un mot utilisateur

6.5. Les **points de suspension** … indiquent que le mot précédent ou la partie précédente entre accolades ou crochets, peut être répétée plusieurs fois.

Exemple : MOVE { nd1 / lit1 } TO nd2 [,nd3] ...

Ce format autorise en particulier les instructions

```
MOVE 1 TO A, B
MOVE B TO C
```

A, B, C étant des mots utilisateurs

6.6. Les caractères de **ponctuation** **,** et ***;*** apparaissant dans les formats sont facultatifs.

6.7. **Abréviations** utilisées dans ce support

nd : nom de donnée
nt : nom de traitement (paragraphe ou section)
ch : chaîne de caractère
ent : entier
lit : littéral (numérique ou chaîne de caractères)
fich : nom de fichier
enreg : nom d'enregistrement logique