Langage orienté gestion

Le langage Cobol

Louis SWINNEN, 2010

TECHNIQUE HELMO Haute Ecole Libre Mosane Saint-Laurent

Références

- A. Clarinval, *Comprendre et connaître le Cobol 85*, Presses Universitaires de Namur, 1991

- N. Stern, R.A. Stern and J.P. Ley, *Cobol for the 21st Century*, John Wiley & Sons , 2005

Pourquoi **encore** étudier le Cobol ? (1)

Quelques réflexions et idées reçues :

- *« Le Cobol est un vieux langage qui est verbeux et rébarbatif ... »*
- *« On ne programme plus en Cobol aujourd'hui »*
- *« Le Java, le .Net, le C ou le C#, c'est vachement mieux ! »*
- *« C'est une perte de temps ! »*

Dès lors, pourquoi encore étudier le Cobol en 2010 ?

Pourquoi **encore** étudier le Cobol ? (2)

« **Did you know that …**

- Application managing about 85 percent of the world's business data are written in Cobol
- There are approximately 90,000 Cobol programmers in the U.S.
- The annual growth of the Cobol code over the next 4 years is projected to be 5 billions lines
- There are 200 billion lines of Cobol code running production applications worldwide
- It has been estimated that 35 percent of all new business application development is written in Cobol

« In summary, for those who suggested that Cobol is a dying language, these facts should to rest that belief. »

Extrait de : « *Cobol for the 21st Century* », © John Wiley & Sons

Pourquoi **encore** étudier le Cobol ? (3)

- De nombreux programmes existent
  - Surtout dans le domaine financier (banque, assurance), où le Cobol est particulièrement répandu.
  - Les applications existantes ne seront pas toutes migrées car c'est un processus dangereux (bug) et couteux (quelle est la plus-value ?)
  - **Il faut donc des programmeurs capables de maintenir ces logiciels !**
- Un langage parmi les autres
  - Paradigme *impératif* comme C ou Pascal, le Cobol est adapté aux applications de gestion

Pourquoi **encore** étudier le Cobol ? (4)

- Certains langages sont dérivés du Cobol
  - Le langage PL/1 est encore utilisé dans certaines entreprises et est directement dérivé du cobol
- Certains langages sont très proches du Cobol
  - Le langage ABAP utilisé dans SAP a de nombreuses similitudes avec le langage Cobol
- **Il y a encore des offres d'emplois pour du développement en Cobol !** Pourquoi s'en priver ?

Points forts du Cobol

- La gestion des fichiers
  - Cobol propose, en standard, des organisations de fichiers intéressantes:
    - séquentielle (dans tous les langages)
    - relative
    - indexé
- La génération de rapport
- Le tri

Comment apprendre le Cobol ?

- Suivant 2 axes:
  - Approche pratique
    - Comprendre un programme Cobol ;
    - Créer un nouveau programme Cobol (par *imitation* au début).
  - Approche par comparaison
    - Comparaison avec le langage C que vous connaissez.
- Pré-requis
  - Concepts de programmation (en C par exemple)

Informations pratiques (1)

- Répartition année - examen
  - **50 %** des points pour l'examen (écrit à cours ouvert)
  - **50 %** des points pour le travail de l'année
    - Principalement les laboratoires !
- Importance du cours
  - Nombre total d'heure: **60 h** - Pondération: **2**
- Enseignants impliqués:
  - Cours théorique (30 h): Louis SWINNEN (**SWILA**)
  - Laboratoires (30 h): Vincent REIP (**REIVI**)

Informations pratiques (2)

- Compilateur utilisé à l'institut:
  - *Microfocus COBOL Net Express Academic 5.1*
    - http://www.microfocus.com
- Compilateurs gratuits :
  - Deux pistes (à tester pour la compatibilité !)
    - Cobol-IT (http://www.cobol-it.com)
    - Open Cobol (http://www.opencobol.org)

Objectifs

- A la fin de ce cours, les étudiants pourront :
  - Intégrer, dans leur démarche d'analyse, les concepts techniques particuliers du langage Cobol
  - Développer une application Cobol répondant aux besoins de l'utilisateur
    - En intégrant les particularités et les avantages propres à ce langage
  - Maintenir une application existante en analysant du code pré-existant.

Chapitre 1
Survol introductif

- Origine
- Normes
- Premier programme
- Observations
- Exercices

Origines du Cobol

- Le Cobol est un langage *impératif* imaginé par CODASYL (**C**onference **O**n **DA**ta **SY**stem **L**anguage)
- Objectif
  - Créer (vers 1960) un langage:
    - Portable (car beaucoup de plateformes existantes et incompatibles)
    - Permettant d'élaborer des applications de gestion
    - Un langage lisible (en anglais, of course)

  COBOL est né (**CO**mmon **B**usiness **O**riented **L**anguage)

Cobol et les normes (1)

- Normes ?
  - COBOL a subi plusieurs évolutions importantes:
    - 60, 61, 65, 68, 74, 85, 2002
  - Mais **deux** normes sont très utilisées aujourd'hui
    - COBOL ANS 74 (ANS => Norme ANSI) – encore utilisée
    - COBOL ANS 85 (abordée dans ce cours)
  - Le cobol évolue toujours, propose une version orientée objet et **s'intègre aujourd'hui à la plateforme .Net !**

Premier programme (1)

```
#include <stdio.h>
#include <string.h>
#define MAX_CURRENCY 10

char ElCode[MAX_CURRENCY][2][4] = {"FB", "BEF", "DM ", "DEM", "PAT", "ESP", \
                                   "FF", "FRF", "£", "GBP", "LIT", "ITL", \
                                   "FL", "NLG", "$", "USD", "FS", "CHF", \
                                   "YEN", "JPY" };
int main(void)
{
        char code[4], code_tmp[4];
        int i=0;

        printf("Entrez le code:"); scanf("%s", code);
        while(strcmpi(code, "XXX") != 0)
        {
                for(i=0;i<MAX_CURRENCY;i++)
                {
                        if(strcmpi(code, ElCode[i][0]) == 0)
                        {
                                printf("Le nouveau code est %s.\n", ElCode[i][1]);
                                break;
                        }
                }
```

Premier programme (2)

```
            if(i >= MAX_CURRENCY)
                    printf("Recherche terminée, aucun élément trouvé !\n");

            printf("Entrez le code:"); scanf("%s", code);
    }

    return 0;

}
```

Que fait ce programme ?

Premier programme (3)

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. EXPERFT6.
* Programme proposé par J.M. Martin
* Mais que fait ce programme ?
*   ANCIEN -> NOUVEAU
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 77 Wcode PIC XXX.
 77 i PIC 99.
 77 maxv PIC 99 VALUE 10.
 01 CodesDevises.
   02 PIC X(12) VALUE "FB BEFDM DEM".
   02 PIC X(12) VALUE "PTAESPFF FRF".
   02 PIC X(12) VALUE "£  GBPLITITL".
   02 PIC X(12) VALUE "FL NLG$  USD".
   02 PIC X(12) VALUE "FS CHFYENJPY".
 01 TableCodes REDEFINES CodesDevises.
   02 ElCode OCCURS 10.
    03 Ancien PIC X(3).
    03 Nouveau PIC X(3).
```

```
PROCEDURE DIVISION.
pgm.
    DISPLAY "Code ? " WITH
        NO ADVANCING
    ACCEPT Wcode
    PERFORM UNTIL Wcode = "XXX"
        PERFORM VARYING i FROM 1 BY 1
        UNTIL i > maxv OR Wcode =
            Ancien(i)
        END-PERFORM
        IF i > maxv THEN
            DISPLAY "Le code " Wcode
                " est inconnu!"
        ELSE DISPLAY "Pour " Wcode
            "Le nouveau code est: "
            Nouveau (i)
        END-IF
        DISPLAY " "
        DISPLAY "Code ? " WITH NO
            ADVANCING
        ACCEPT Wcode
    END-PERFORM
    .
END PROGRAM EXPERFT6.
```

Observations (1)

- Structure

`IDENTIFICATION DIVISION.`
`PROGRAM-ID.` Nom-du-programme

Cette division regroupe les informations d'identification du programme.
Elle doit comprendre surtout le nom du programme.

`ENVIRONMENT DIVISION.`

Cette division comporte quelques informations concernant l'environnement.
Elle permet notamment de spécifier les fichiers à utiliser, mais aussi des paramètres propres au système d'exploitation, à des régions, ...

`CONFIGURATION SECTION.`

Cette section reprend les options régionales et de configuration de l'environnement.

Ex: le séparateur décimal, ...

`INPUT-OUTPUT SECTION.`

Dans cette section, on trouve essentiellement les définitions des fichiers.
On y indique le nom des fichiers utilisés ainsi que leurs organisations internes.

Observations (2)

- Structure (suite)

`DATA DIVISION.`

Cette division comporte toutes les déclarations de données. En Cobol, les données sont déclarées pour le « module » en cours.

`FILE SECTION.`

Cette section définit la structure des enregistrements à l'intérieur du fichier.

`WORKING-STORAGE SECTION.`

Cette section contient toutes les définitions de données pour le programme.

`LINKAGE SECTION.`

Permet de définir les données qui sont partagées entre un programme appelant et le programme appelé.

`REPORT SECTION.`

Section qui contient les informations pour la génération d'un rapport en utilisant le Report Writer de Cobol.

Observations (3)

- Structure (fin)

  `PROCEDURE DIVISION.`

  Cette division contient le code du module. On trouvera à l'intérieur de cette division toute la logique du programme.

  - Un programme COBOL ANS 85 commence à la colonne 8 et se termine à la colonne 72
    - Contraintes issues des cartes perforées, encore d'application sur de vieux systèmes et certains compilateurs.
  - Deux zones distinctes
    - Zone A: à partir de la colonne 8 ; B: colonne 12
  - Les commentaires: astérisque * en colonne 7

Observations (4)

- Sections, paragraphes et phrases
  - Une section est composée d'un ensemble (éventuellement vide) de *paragraphes*
  - Un paragraphe est composé d'un nom de paragraphe et d'un ensemble (éventuellement vide) de *phrases*
  - Une phrase est composée d'un ensemble d'instructions et se termine par un point
  - Intérêt de cette découpe ?
    - **Définition implicite de blocs !**

Observations (5)

- Conseils:
  - Garder une structure simple
    - Limiter le nombre de sections ;
    - Une seule phrase par paragraphe;
    - Limiter le nombre de paragraphe (créer un nouveau paragraphe lorsque cela en vaut la peine).
- Ponctuation: le point
  - *Il faut un point à la fin de toute phrase*
    - Si une seule phrase dans le paragraphe, un seul point est nécessaire !

Observations (6)

- Code

```
DATA DIVISION.
WORKING-STORAGE SECTION.
77 Wcode PIC XXX.
77 i PIC 99.
77 maxv PIC 99 VALUE 10.
```

Définition des données

```
01 TableCodes REDEFINES CodesDevises.
 02 ElCode OCCURS 10.
   03 Ancien PIC X(3).
   03 Nouveau PIC X(3).
```

Définition d'une structure

Format de la donnée:

Nom pour désigner cette donnée (facultatif)

Niveau (77: donnée non structurée)

PIC 9	Un chiffre
PIC X	Un caractère
PIC 99	Un nombre de 2 chiffres
PIC X(3)	Une chaîne de 3 caractères équivalent à PIC XXX
OCCURS	Répétition

Exercice 1

- Construisez le programme C qui:
  - lit une chaîne de caractère de 5 éléments
  - affiche cette chaîne dans l'ordre inverse
  - Exemple:
    - ABCDE -> EDCBA
- Construisez le programme COBOL équivalent

Correction 1 (1)

Correction 1 (2)

Exercice 2

Il faut créer le programme C et Cobol qui « chiffre » une chaîne de caractères (30 max) comme suit:

A -> I	N -> X
B -> M	O -> A
C -> D	P -> F
D -> K	Q -> S
E -> R	R -> O
F -> Z	S -> G
G -> L	T -> H
H -> T	U -> P
I -> N	V -> U
J -> V	W -> W
K -> Y	X -> Q
L -> C	Y -> B
M -> E	Z -> J

Correction 2 (1)

Correction 2 (2)

Correction 2 (3)