Module A206

(version V2)

Programmation fichiers – COBOL

Chapitre 5

COBOL et le génie logiciel

Alain Vailly

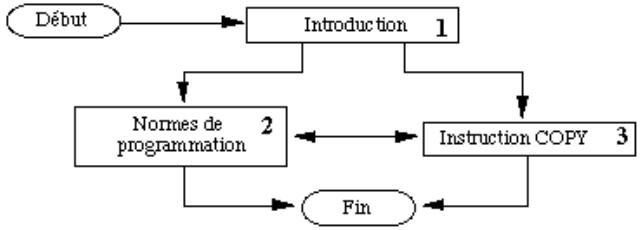
Alain.Vailly@univ-nantes.fr

COBOL et le génie logiciel

1. Introduction

Après avoir, dans cette introduction, justifié une telle partie, situé les enjeux et illustré notre propos de deux exemples, un bon et un mauvais, nous présentons quelques normes de programmation qu'il nous parait judicieux de respecter. L'instruction COPY est ensuite étudiée.

La structure de ce chapitre est la suivante :



1.1 Essai de définition

Selon Ian SOMMERVILLE (auteur d'un ouvrage fameux Le génie logiciel, paru dans sa version française en octobre 1992, aux éditions ADDISON-WESLEY France -ISBN n° 2-87908-033-9), le terme de génie logiciel a été introduit à la fin des années soixante... Le génie logiciel (nous citons toujours SOMMERVILLE) ne concerne pas seulement la réalisation de produits, mais surtout la façon la plus efficiente de les produire. Nous faisons notre cette définition et considérons que le génie logiciel se préoccupe d'industrialisation du processus de fabrication des programmes et, encore plus précisément, de production de programmes fiables, économiques... toute propriété que l'on associe généralement à la notion de qualité.

1.2 Essai de justification

Lors de la conception de ce module, nous avons absolument tenu à glisser ce chapitre dans le plan. "Apprendre à programmer en COBOL" n'est pas un objectif suffisant pour nous. Nous voulons y ajouter le mot "bien", pour le transformer en "Apprendre à bien programmer en COBOL". Bien programmer, en effet, doit être l'objectif de tout informaticien et ce quel que soit le langage utilisé. Dans le cadre qui est le notre actuellement, COBOL, cet objectif est encore plus prioritaire. Ce langage a, rappelons-le, été conçu pour être standard et appliqué à la gestion.

La création d'une application est une activité coûteuse et l'entreprise doit planifier cette activité en termes d'investissements et de retour sur investissements. Plus l'application sert, plus le retour sera important. Les informaticiens, d'un autre côté, ne se consacrent pas qu'à une seule série de programmes. Durant leur vie professionnelle, ils vont créer, reprendre, modifier, adapter... des programmes.

Si l'on combine tous ces éléments, on s'aperçoit que la principale qualité attendue de la production des informaticiens est la lisibilité (pour qu'une tierce personne puisse intervenir dessus, seule, sans assistance). Celle-ci sera atteinte si quelques critères sont respectés. Nous les présentons. Il est également nécessaire de produire vite, ce qui sous-entend un outil de production efficace. Nous avons déjà garni la trousse à outil du développeur

d'un ensemble de règles de travail pour mieux appréhender le problème à traiter. Celles-ci forment ce que nous avons appelé la cinématique de fichiers. Cela ne suffit pas. Savoir analyser un problème, savoir choisir le bon outil algorithmique, savoir l'appliquer rapidement, tout ceci contribue à accroître la vitesse de développement. Nous complétons cet arsenal par une pratique qui permet de définir des modules (ie. des morceaux de programmes) et de les inclure dans un programme, via l'instruction COPY.

1.3 À propos de normes

Par tradition, les français n'aiment pas trop les normes trop strictes. Par habitude, les étudiants programmeurs relèguent ces normes au second plan (cela est sans doute dû au fait que leur premier objectif est d'abord de faire tourner ce f... programme ; alors, pensez, bien programmer...). Vouloir initier les étudiants programmeurs français aux normes de programmation est donc une vraie gageure. Pour autant est-ce nécessaire. Nous sommes convaincus du fait que, par delà la définition précise des normes, l'existence même de celles-ci prime sur le reste.

Lorsqu'on est dans une société de services, on applique les normes en vigueur dans celle-ci. Si l'on change de société, il y a de grandes chances qu'elles changent également. Malgré cela, il faudra appliquer ces nouvelles normes. Nous considérons donc qu'il y a une norme, qu'elle est discutable mais que, comme toute autre, elle ne doit pas être discutée mais appliquée. Nous allons, sur ce point, jusqu'à considérer comme nul (et donc lui attribuer un zéro) un programme qui tourne mais qui ne respecte pas les normes. Radical, mais efficace.

1.4 Exemple et contre-exemple

La meilleure illustration que nous puissions fournir de la justesse de nos propos précédents réside dans un programme rédigé de deux façons, sans norme et avec. Une simple lecture de ces deux versions (elles sont présentées ci-après) suffit à convaincre les plus réticents.

15/12/05

Exemples de programmes COBOL

Avertissement

Les exemples qui sont présentés dans cette partie de cours sont, dans la quasi-totalité des cas, des "oeuvres" d'étudiants. Ils correspondent à des programmes qui "tournent". Ils ont été rédigé dans un dialecte de COBOL, celui produit par la société Legacy et appellé Percobol. Les programmes ont été compilé avec succès. Ils ont également fait l'objet d'un test par un enseignant, dans un environnement Linux/Dec. Ils peuvent donc être considérés comme un exemple fiable, parfaitement transposable (à des variations près de compilateurs) dans un autre environnement.

===> Pour éviter une perte de place, nous avons éliminé de la présentation du code les six premiers caractères. Le bord gauche de l'écran correspond donc à la colonne 7. <===

Tous les programmes, qu'il s'agisse de simples extraits ou de programmes complets, sont numérotés (selon une numérotation séquentielle continue). Ces numéros serviront de référence pour identifier clairement l'exemple en cas de dialogue apprenant-professeur. Plutôt que de parler du programme qui édite un état des ventes, il faudra, par exemple, parler du programme P37.

(Contre) Exemple P06

Ce programme est une des réponses à un problème posé dans une des fiches de travaux dirigés de cet enseignement. Il est "mal bâti", dans la mesure où il ne respecte presque aucune des normes de programmations que nous préconisons. Nous en proposons une correction.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. P7.
 AUTHOR. JOALEX.
 DATE-WRITTEN. 11/01/04.
***********************
 ENVIRONMENT DIVISION.
***********************
 CONFIGURATION SECTION.
***********************
 SPECIAL-NAMES.
 CONSOLE IS CRT.
 INPUT-OUTPUT SECTION.
***********************
 FILE-CONTROL.
* accès dynamique
 SELECT FIND ASSIGN TO "fetudind.dat"
 ORGANIZATION INDEXED ACCESS MODE DYNAMIC
 RECORD KEY IS FNUD
 ALTERNATE RECORD KEY IS FNOM WITH DUPLICATES
 ALTERNATE RECORD KEY IS FANAISS WITH DUPLICATES.
* accès séquentiel
 SELECT FINDSEQ ASSIGN TO "fetudind.dat"
 ORGANIZATION INDEXED ACCESS MODE SEQUENTIAL
 RECORD KEY IS FNUD
 ALTERNATE RECORD KEY IS FNOM WITH DUPLICATES
 ALTERNATE RECORD KEY IS FANAISS WITH DUPLICATES.
***********************
```

```
 DATA DIVISION.
**********************
 FILE SECTION.
**********************
 FD FIND.
 01 FETUDIANT.
 03 FNUD PIC 999.
 03 FNOM PIC X(20).
 03 FPRENOM PIC X(20).
 03 FADR PIC X(25).
 03 FANAISS PIC 9(4).
 FD FINDSEQ.
 01 FETUDIANTSEQ.
 03 FNUDSEQ PIC 999.
 03 FNOMSEQ PIC X(20).
 03 FPRENOMSEQ PIC X(20).
 03 FADRSEQ PIC X(25).
 03 FANAISSSEQ PIC 9(4).
 WORKING-STORAGE SECTION.
**********************
* variables générales
 77 WFIN_LECT PIC 9.
 77 WPAUSE PIC 9.
 77 WCHOIXMENU PIC X.
 77 WMENUQUIT PIC X.
* A3
 77 WRECHNOM PIC X(20).
 77 WNBETUDTROUVE PIC 99.
* A4
 77 WRECHNUM PIC 999.
 PROCEDURE DIVISION.
**********************
* Programme principal
 PRINCIPAL.
 MOVE "Z" TO WCHOIXMENU.
 MOVE "Z" TO WMENUQUIT.
 PERFORM AFF UNTIL WCHOIXMENU = "Q" OR WMENUQUIT = "Q".
 STOP RUN.
***** Menu
 AFF.
 MOVE "X" TO WCHOIXMENU.
 DISPLAY "Liste des étudiants (un par page) - 1" AT 1002
 WITH BLANK SCREEN.
 DISPLAY "Recherche par nom - 2" AT 1102.
 DISPLAY "Recherche par numéro - 3" AT 1202.
 DISPLAY "Quitter - Q" AT 1302.
 DISPLAY "Votre Choix :" AT 1402.
* saisie du choix de l'utilisateur
 PERFORM CHOIXMENU UNTIL WCHOIXMENU = "Q" OR WCHOIXMENU = "1"
 OR WCHOIXMENU = "2" OR WCHOIXMENU = "3".
* lancement de la procédure choisie
 EVALUATE WCHOIXMENU
 WHEN "1" PERFORM P6
 WHEN "2" PERFORM A3
 WHEN "3" PERFORM A4
 WHEN "Q" PERFORM QUITTE_PRINCIPAL
* inutile mais on ne sait jamais...
 WHEN OTHER PERFORM ERREURFCT.
 CLOSE FIND.
 CLOSE FINDSEQ.
***** Fin du menu
 QUITTE_PRINCIPAL.
 STOP RUN.
```



```
 CHOIXMENU.
 ACCEPT WCHOIXMENU AT 1416.
 MOVE FUNCTION UPPER-CASE(WCHOIXMENU) TO WCHOIXMENU.
* en cas d'erreur de saisie de l'utilisateur
 ERREURFCT.
 DISPLAY "ATTENTION, la procédure que vous avez demandé
- " n'est pas encore disponible" WITH BLANK SCREEN AT 1002.
 STOP RUN.
*****************************************************************
* PROCEDURE P6 - Affichage des étudiants un par un *
*****************************************************************
 P6.
 OPEN INPUT FINDSEQ.
 MOVE "X" TO WMENUQUIT.
 MOVE 0 TO WFIN_LECT.
 READ FINDSEQ AT END MOVE 1 TO WFIN_LECT.
 PERFORM RECUPETUD UNTIL WMENUQUIT = "Q" OR WMENUQUIT = "M".
 CLOSE FINDSEQ.
 RECUPETUD.
 MOVE "X" TO WMENUQUIT.
 DISPLAY "NUMERO :" LINE 5 COL 2
 WITH BLANK SCREEN.
 DISPLAY "NOM :" LINE 8 COL 2.
 DISPLAY "PRENOM :" LINE 10 COL 2.
 DISPLAY "ADRESSE :" LINE 12 COL 2.
 DISPLAY "ANNEE NAISSANCE :" LINE 14 COL 2.
 DISPLAY FNUDSEQ LINE 5 COL 11.
 DISPLAY FNOMSEQ LINE 8 COL 8.
 DISPLAY FPRENOMSEQ LINE 10 COL 11.
 DISPLAY FADRSEQ LINE 12 COL 12.
 DISPLAY FANAISSSEQ LINE 14 COL 20.
 READ FINDSEQ NEXT AT END MOVE 1 TO WFIN_LECT.
 IF WFIN_LECT = 1 THEN
 PERFORM QUESTMENU UNTIL WMENUQUIT = "Q" OR WMENUQUIT = "M"
 ELSE PERFORM QUESTMENUSUIV UNTIL WMENUQUIT = "S" OR WMENUQUIT = "Q"
 OR WMENUQUIT = "M".
*****************************************************************
* FIN PROCEDURE P6 - Affichage des étudiants un par un *
*****************************************************************
*****************************************************************
* PROCEDURE A3 - Recherche par nom *
*****************************************************************
 A3.
* Initialisation des variables
 MOVE 0 TO WFIN_LECT.
 MOVE 0 TO WNBETUDTROUVE.
 MOVE "012aAxB" TO WRECHNOM.
 MOVE "X" TO WMENUQUIT.
* Lecture dans le fichier
 OPEN INPUT FIND.
* Demande du nom de l'étudiant et le met en majusucle
 PERFORM RECUPNOMETUDDERECH.
 MOVE WRECHNOM TO FNOM.
 READ FIND KEY FNOM
 INVALID KEY PERFORM FETUDIANT_INEXISTANT
 NOT INVALID KEY PERFORM FETUDIANT_EXISTANT UNTIL WFIN_LECT = 1
 OR WMENUQUIT = "M" OR WMENUQUIT = "Q"
 END-READ.
 CLOSE FIND.
*******************************************************
* Récupère le nom de l'étudiant à rechercher *
*******************************************************
 RECUPNOMETUDDERECH.
```

```
 DISPLAY "Donnez un nom d'etudiant :" AT 1502
 WITH BLANK SCREEN.
 ACCEPT WRECHNOM AT 1530.
 MOVE FUNCTION UPPER-CASE(WRECHNOM) TO WRECHNOM.
 DISPLAY "Vous avez selectionne le nom : " AT 1702.
 DISPLAY WRECHNOM AT 1733.
 ACCEPT WPAUSE.
 FETUDIANT_INEXISTANT.
 DISPLAY "Le nom "
 AT 1002 WITH BLANK SCREEN.
 DISPLAY "n'existe pas." AT 1102.
 DISPLAY WRECHNOM AT 1009.
 PERFORM QUESTMENU UNTIL WMENUQUIT = "Q"
 OR WMENUQUIT = "M".
 FETUDIANT_EXISTANT.
* Affichage de l'etudiant trouvé
 DISPLAY FNUD LINE 15 COL 2
 WITH BLANK SCREEN.
 DISPLAY FNOM LINE 15 COL 6.
 DISPLAY FPRENOM LINE 15 COL 27.
 DISPLAY FANAISS LINE 15 COL 48.
 DISPLAY FADR LINE 16 COL 4.
* incrémentation du compteur d'etudiants trouvés
 ADD 1 TO WNBETUDTROUVE.
* Lecture d'un enregistrement dans le fichier
 READ FIND NEXT.
 IF FNOM <> WRECHNOM THEN MOVE 1 TO WFIN_LECT.
 MOVE "X" TO WMENUQUIT
 PERFORM QUESTMENUSUIV UNTIL WMENUQUIT = "S"
 OR WMENUQUIT = "Q"
 OR WMENUQUIT = "M"
 IF WFIN_LECT = 1 THEN
 PERFORM STATS
 END-IF.
*****************************************************************
* FIN PROCEDURE A3 - Recherche par nom *
*****************************************************************
*****************************************************************
* PROCEDURE A4 - Recherche par numéro *
*****************************************************************
 A4.
* Initialisation des variables
 MOVE 0 TO WRECHNUM.
 MOVE 0 TO WFIN_LECT.
 MOVE "X" TO WMENUQUIT.
* Lecture dans le fichier
 OPEN INPUT FIND.
* Demande le numéro de l'étudiant
 PERFORM RECUPNUMETUDDERECH.
 MOVE WRECHNUM TO FNUD.
 READ FIND
 INVALID KEY PERFORM NUMERO_INEXISTANT
 NOT INVALID KEY PERFORM NUMERO_EXISTANT UNTIL
 WFIN_LECT = 1
 OR WMENUQUIT = "M"
 OR WMENUQUIT = "Q"
 END-READ.
 CLOSE FIND.
 NUMERO_INEXISTANT.
 DISPLAY "Le numéro n'existe pas."
 AT 1002 WITH BLANK SCREEN.
 DISPLAY WRECHNUM AT 1012.
 PERFORM QUESTMENU UNTIL WMENUQUIT = "M"
```



```
 OR WMENUQUIT = "Q".
 NUMERO_EXISTANT.
* Affichage de l'etudiant trouvé
 DISPLAY FNUD LINE 15 COL 2
 WITH BLANK SCREEN.
 DISPLAY FNOM LINE 15 COL 6.
 DISPLAY FPRENOM LINE 15 COL 27.
 DISPLAY FANAISS LINE 15 COL 48.
 DISPLAY FADR LINE 16 COL 4.
* on quitte la procédure
 PERFORM QUESTMENU UNTIL WMENUQUIT = "M"
 OR WMENUQUIT = "Q".
 RECUPNUMETUDDERECH.
 DISPLAY "Donnez un numéro d'etudiant :" AT 1502
 WITH BLANK SCREEN.
 ACCEPT WRECHNUM AT 1533.
 DISPLAY "Vous avez selectionne le numéro : " AT 1702.
 DISPLAY WRECHNUM AT 1736.
 ACCEPT WPAUSE.
*****************************************************************
* FIN PROCEDURE A4 - Recherche par numéro *
*****************************************************************
***************** Procédures diverses ***************************
* Affichage de la Question [Menu/Quitter] ***********************
 QUESTMENU.
 DISPLAY "M)enu Q)uitter" LINE 22 COL 3.
 DISPLAY "Votre choix (réponse M,m,Q,q) :" LINE 23 COL 3.
 ACCEPT WMENUQUIT LINE 23 COL 35.
 MOVE FUNCTION UPPER-CASE(WMENUQUIT) TO WMENUQUIT.
 IF WMENUQUIT = "Q" THEN
 CLOSE FIND
 STOP RUN.
* Affichage de la Question [Suivant/Menu/Quitter] ******************
 QUESTMENUSUIV.
 DISPLAY "S)uivant M)enu Q)uitter" LINE 22 COL 3.
 DISPLAY "Votre choix (réponse S,s,M,m,Q,q) :" LINE 23 COL 3.
 ACCEPT WMENUQUIT LINE 23 COL 39.
 MOVE FUNCTION UPPER-CASE(WMENUQUIT) TO WMENUQUIT.
 IF WMENUQUIT = "Q" THEN
 CLOSE FIND
 STOP RUN.
* Affichage de la Question [Quitter] ****************************
 QUITTE.
 PERFORM QUESTMENU UNTIL WMENUQUIT = "M"
 OR WMENUQUIT = "Q".
 STATS.
 DISPLAY "Il n'y a plus d'etudiant nommé : " LINE 10 COL 2
 WITH BLANK SCREEN.
 DISPLAY WRECHNOM LINE 10 COL 35.
 DISPLAY "Vous en avez trouvé en tout !" LINE 11 COL 2.
 DISPLAY WNBETUDTROUVE LINE 11 COL 22.
* Affichage de la question
 PERFORM QUESTMENU UNTIL WMENUQUIT = "M"
 OR WMENUQUIT = "Q".
 END-PROGRAM P7.
```

(Bon) Exemple P07

Ce programme est une des réponses à un problème posé dans une des fiches de travaux dirigés de cet enseignement. Il respecte les normes de programmations que nous préconisons. Nous en proposons une version mal "fichue" pour comparaison.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. P7.
 AUTHOR. JOALEX.
 DATE-WRITTEN. 11/01/04.
**********************
 ENVIRONMENT DIVISION.
**********************
 CONFIGURATION SECTION.
**********************
 SPECIAL-NAMES.
     CONSOLE IS CRT.
 INPUT-OUTPUT SECTION.
**********************
 FILE-CONTROL.
* accès dynamique
     SELECT FIND ASSIGN TO "fetudind.dat"
         ORGANIZATION INDEXED ACCESS MODE DYNAMIC
         RECORD KEY IS FNUD
         ALTERNATE RECORD KEY IS FNOM WITH DUPLICATES
         ALTERNATE RECORD KEY IS FANAISS WITH DUPLICATES.
* accès séquentiel
     SELECT FINDSEQ ASSIGN TO "fetudind.dat"
         ORGANIZATION INDEXED ACCESS MODE SEQUENTIAL
         RECORD KEY IS FNUD
         ALTERNATE RECORD KEY IS FNOM WITH DUPLICATES
         ALTERNATE RECORD KEY IS FANAISS WITH DUPLICATES.
**********************
 DATA DIVISION.
**********************
 FILE SECTION.
**********************
 FD FIND.
 01  FETUDIANT.
     03  FNUD PIC 999.
     03  FNOM PIC X(20).
     03  FPRENOM PIC X(20).
     03  FADR PIC X(25).
     03  FANAISS PIC 9(4).
 FD  FINDSEQ.
 01  FETUDIANTSEQ.
     03  FNUDSEQ PIC 999.
     03  FNOMSEQ PIC X(20).
     03  FPRENOMSEQ PIC X(20).
     03  FADRSEQ PIC X(25).
     03  FANAISSSEQ PIC 9(4).
 WORKING-STORAGE SECTION.
**********************
* variables générales
 77  WFIN_LECT PIC 9.
 77  WPAUSE PIC 9.
 77  WCHOIXMENU PIC X.
 77  WMENUQUIT PIC X.
* A3
 77  WRECHNOM PIC X(20).
 77  WNBETUDTROUVE PIC 99.
* A4
 77  WRECHNUM PIC 999.
 PROCEDURE DIVISION.
**********************
* Programme principal
 PRINCIPAL.
     MOVE "Z" TO WCHOIXMENU.
     MOVE "Z" TO WMENUQUIT.
```



```
     PERFORM AFF UNTIL WCHOIXMENU = "Q" OR WMENUQUIT = "Q".
     STOP RUN.
***** Menu
 AFF.
     MOVE "X" TO WCHOIXMENU.
     DISPLAY "Liste des étudiants (un par page) - 1" AT 1002
         WITH BLANK SCREEN.
     DISPLAY "Recherche par nom - 2" AT 1102.
     DISPLAY "Recherche par numéro - 3" AT 1202.
     DISPLAY "Quitter - Q" AT 1302.
     DISPLAY "Votre Choix :" AT 1402.
* saisie du choix de l'utilisateur
     PERFORM CHOIXMENU UNTIL WCHOIXMENU = "Q" OR WCHOIXMENU = "1"
         OR WCHOIXMENU = "2" OR WCHOIXMENU = "3".
* lancement de la procédure choisie
     EVALUATE WCHOIXMENU
         WHEN "1" PERFORM P6
         WHEN "2" PERFORM A3
         WHEN "3" PERFORM A4
         WHEN "Q" PERFORM QUITTE_PRINCIPAL
* inutile mais on ne sait jamais...
         WHEN OTHER PERFORM ERREURFCT.
     CLOSE FIND.
     CLOSE FINDSEQ.
***** Fin du menu
 QUITTE_PRINCIPAL.
     STOP RUN.
 CHOIXMENU.
     ACCEPT WCHOIXMENU AT 1416.
     MOVE FUNCTION UPPER-CASE(WCHOIXMENU) TO WCHOIXMENU.
* en cas d'erreur de saisie de l'utilisateur
 ERREURFCT.
     DISPLAY "ATTENTION, la procédure que vous avez demandée"
-        " n'est pas encore disponible" WITH BLANK SCREEN AT 1002.
     STOP RUN.
**********************************************************************
* PROCEDURE P6 - Affichage des étudiants un par un *
**********************************************************************
 P6.
     OPEN INPUT FINDSEQ.
     MOVE "X" TO WMENUQUIT.
     MOVE 0 TO WFIN_LECT.
     READ FINDSEQ AT END MOVE 1 TO WFIN_LECT.
     PERFORM RECUPETUD UNTIL WMENUQUIT = "Q" OR WMENUQUIT = "M".
     CLOSE FINDSEQ._ RECUPETUD.
     MOVE "X" TO WMENUQUIT.
     DISPLAY "NUMERO :" LINE 5 COL 2 WITH BLANK SCREEN.
     DISPLAY "NOM :" LINE 8 COL 2.
     DISPLAY "PRENOM :" LINE 10 COL 2.
     DISPLAY "ADRESSE :" LINE 12 COL 2.
     DISPLAY "ANNEE NAISSANCE :" LINE 14 COL 2.
     DISPLAY FNUDSEQ LINE 5 COL 11.
     DISPLAY FNOMSEQ LINE 8 COL 8.
     DISPLAY FPRENOMSEQ LINE 10 COL 11.
     DISPLAY FADRSEQ LINE 12 COL 12.
     DISPLAY FANAISSSEQ LINE 14 COL 20.
     READ FINDSEQ NEXT AT END MOVE 1 TO WFIN_LECT.
     IF WFIN_LECT = 1 THEN
         PERFORM QUESTMENU UNTIL WMENUQUIT = "Q"
             OR WMENUQUIT = "M"
     ELSE PERFORM QUESTMENUSUIV UNTIL WMENUQUIT = "S"
             OR WMENUQUIT = "Q"
             OR WMENUQUIT = "M".
```

```
*****************************************************************
* FIN PROCEDURE P6 - Affichage des étudiants un par un *
*****************************************************************
*****************************************************************
* PROCEDURE A3 - Recherche par nom *
*****************************************************************
 A3.
* Initialisation des variables
     MOVE 0 TO WFIN_LECT.
     MOVE 0 TO WNBETUDTROUVE.
     MOVE "012aAxB" TO WRECHNOM.
     MOVE "X" TO WMENUQUIT.
* Lecture dans le fichier
     OPEN INPUT FIND.
* Demande du nom de l'étudiant et mise en majusucle
     PERFORM RECUPNOMETUDDERECH.
     MOVE WRECHNOM TO FNOM.
     READ FIND KEY FNOM
         INVALID KEY PERFORM FETUDIANT_INEXISTANT
         NOT INVALID KEY PERFORM FETUDIANT_EXISTANT UNTIL
             WFIN_LECT = 1
             OR WMENUQUIT = "M"
             OR WMENUQUIT = "Q"
     END-READ.
     CLOSE FIND.
*****************************************************
* Récupère le nom de l'étudiant à rechercher *
*****************************************************
 RECUPNOMETUDDERECH.
     DISPLAY "Donnez un nom d'etudiant :" AT 1502
         WITH BLANK SCREEN.
     ACCEPT WRECHNOM AT 1530.
     MOVE FUNCTION UPPER-CASE(WRECHNOM) TO WRECHNOM.
     DISPLAY "Vous avez selectionne le nom : " AT 1702.
     DISPLAY WRECHNOM AT 1733.
     ACCEPT WPAUSE.
 FETUDIANT_INEXISTANT.
     DISPLAY "Le nom " AT 1002 WITH BLANK SCREEN.
     DISPLAY "n'existe pas." AT 1102.
     DISPLAY WRECHNOM AT 1009.
     PERFORM QUESTMENU UNTIL WMENUQUIT = "Q" OR WMENUQUIT = "M".
 FETUDIANT_EXISTANT.
* Affichage de l'etudiant trouvé
     DISPLAY FNUD LINE 15 COL 2 WITH BLANK SCREEN.
     DISPLAY FNOM LINE 15 COL 6.
     DISPLAY FPRENOM LINE 15 COL 27.
     DISPLAY FANAISS LINE 15 COL 48.
     DISPLAY FADR LINE 16 COL 4.
* Incrémentation du compteur d'etudiants trouvés
     ADD 1 TO WNBETUDTROUVE.
* Lecture d'un enregistrement dans le fichier
     READ FIND NEXT.
     IF FNOM <> WRECHNOM THEN MOVE 1 TO WFIN_LECT.
     MOVE "X" TO WMENUQUIT
     PERFORM QUESTMENUSUIV UNTIL WMENUQUIT = "S"
         OR WMENUQUIT = "Q"
         OR WMENUQUIT = "M"
     IF WFIN_LECT = 1 THEN
         PERFORM STATS
     END-IF.
*****************************************************************
* FIN PROCEDURE A3 - Recherche par nom *
*****************************************************************
```



```
*****************************************************************
* PROCEDURE A4 - Recherche par numéro *
*****************************************************************
 A4.
* Initialisation des variables
     MOVE 0 TO WRECHNUM.
     MOVE 0 TO WFIN_LECT.
     MOVE "X" TO WMENUQUIT.
* Lecture dans le fichier
     OPEN INPUT FIND.
* Demande le numéro de l'étudiant
     PERFORM RECUPNUMETUDDERECH.
     MOVE WRECHNUM TO FNUD.
     READ FIND
         INVALID KEY PERFORM NUMERO_INEXISTANT
         NOT INVALID KEY PERFORM NUMERO_EXISTANT UNTIL
             WFIN_LECT = 1 OR WMENUQUIT = "M" OR WMENUQUIT = "Q"
     END-READ.
     CLOSE FIND.
 NUMERO_INEXISTANT.
     DISPLAY "Le numéro n'existe pas." AT 1002 WITH BLANK SCREEN.
     DISPLAY WRECHNUM AT 1012.
     PERFORM QUESTMENU UNTIL WMENUQUIT = "M" OR WMENUQUIT = "Q".
 NUMERO_EXISTANT.
* Affichage de l'etudiant trouvé
     DISPLAY FNUD LINE 15 COL 2 WITH BLANK SCREEN.
     DISPLAY FNOM LINE 15 COL 6.
     DISPLAY FPRENOM LINE 15 COL 27.
     DISPLAY FANAISS LINE 15 COL 48.
     DISPLAY FADR LINE 16 COL 4.
* On quitte la procédure
     PERFORM QUESTMENU UNTIL WMENUQUIT = "M" OR WMENUQUIT = "Q".
 RECUPNUMETUDDERECH.
     DISPLAY "Donnez un numéro d'etudiant :" AT 1502
         WITH BLANK SCREEN.
     ACCEPT WRECHNUM AT 1533.
     DISPLAY "Vous avez selectionne le numéro : " AT 1702.
     DISPLAY WRECHNUM AT 1736.
     ACCEPT WPAUSE.
*****************************************************************
* FIN PROCEDURE A4 - Recherche par numéro *
*****************************************************************
***************** Procédures diverses ***************************
* Affichage de la Question [Menu/Quitter] ***********************
 QUESTMENU.
     DISPLAY "M)enu Q)uitter" LINE 22 COL 3.
     DISPLAY "Votre choix (réponse M,m,Q,q) :" LINE 23 COL 3.
     ACCEPT WMENUQUIT LINE 23 COL 35.
     MOVE FUNCTION UPPER-CASE(WMENUQUIT) TO WMENUQUIT.
     IF WMENUQUIT = "Q" THEN
         CLOSE FIND
         STOP RUN.
* Affichage de la Question [Suivant/Menu/Quitter] *****************
 QUESTMENUSUIV.
     DISPLAY "S)uivant M)enu Q)uitter" LINE 22 COL 3.
     DISPLAY "Votre choix (réponse S,s,M,m,Q,q) :" LINE 23 COL 3.
     ACCEPT WMENUQUIT LINE 23 COL 39.
     MOVE FUNCTION UPPER-CASE(WMENUQUIT) TO WMENUQUIT.
     IF WMENUQUIT = "Q" THEN
         CLOSE FIND
         STOP RUN.
* Affichage de la Question [Quitter] ******************************
 QUITTE.
```

```
    PERFORM QUESTMENU UNTIL WMENUQUIT = "M" OR WMENUQUIT = "Q".
 STATS.
    DISPLAY "Il n'y a plus d'etudiant nommé : " LINE 10 COL 2
        WITH BLANK SCREEN.
    DISPLAY WRECHNOM LINE 10 COL 35.
    DISPLAY "Vous en avez trouvé en tout !" LINE 11 COL 2.
    DISPLAY WNBETUDTROUVE LINE 11 COL 22.
* Affichage de la question
    PERFORM QUESTMENU UNTIL WMENUQUIT = "M" OR WMENUQUIT = "Q".
 END-PROGRAM P7.
```

15/12/05

COBOL et le génie logiciel

2. Normes de programmation

Rappelons, avant d'entrer dans le vif du sujet, notre credo :

- il n'y a pas de bonne programmation sans norme ;
- quelles que soient les normes en vigueur, il faut les appliquer à la lettre.

Ceci étant fait, que préconisons-nous ? Notre ordonnance comporte quatre prescriptions, agissant sur des éléments différents :

- noms des variables : significatifs et indicateurs,
- commentaires : en importance,
- procédures : bien présentées, de taille raisonnable,
- entrées-sorties : séparées

2.1 Normes relatives aux noms de variables

Nous recommandons de recourir systématiquement à des noms de variables ayant du sens. Le langage COBOL repose sur un usage intensif de structures textuelles. Il n'est donc pas obligatoire d'avoir des noms réduits. Il est certes peu recommandable d'avoir à manipuler des variables ayant des noms à rallonge, mais entre NC et NOM_CLIENT, nous n'hésitons pas une seconde. Nous jetons le premier et retenons le second.

Nous recommandons également d'associer systématiquement aux noms des variables un code permettant de savoir dans quelle section elles sont définies. Un programme COBOL, a priori, est un programme volumineux. Il n'est pas rare d'avoir deux cents ou trois cents lignes de code, ce qui, bon gré, mal gré, correspond à six ou sept pages de format A4 de listing. Savoir rien qu'en lisant une instruction que toutes les variables sont des paramètres permet d'aller plus rapidement à l'endroit de leur définition.

Nous proposons le code suivant :

F	variable décrite en FILE SECTION
P	paramètre décrit en LINKAGE SECTION
R	variable définie en REPORT SECTION
S	élément d'une SCREEN SECTION
W	variable de travail, en WORKING-STORAGE SECTION

Pour des raisons similaires, **nous suggérons d'ajouter au code F au moins trois lettres rappelant le nom du fichier**. L'exemple que nous avons fourni dans la partie relative à la définition des variables de travail est plus clair en le reprenant et en appliquant les règles que nous venons dénoncer :

```
IDENTIFICATION DIVISION.
PROGRAM-ID. prog_P7.
AUTHOR. Bastien et Johan.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT FETUD ASSIGN TO "fetud.dat".
```

```
    SELECT FETUD-IND-CRE ASSIGN TO "fetudindcre.dat"
        ORGANIZATION INDEXED
        ACCESS MODE IS RANDOM
        RECORD KEY F-ETU-NUM-IND-CRE
        ALTERNATE RECORD KEY F-ETU-NOM-IND-CRE
            WITH DUPLICATES
        ALTERNATE RECORD KEY F-ETU-ANAISS-IND-CRE
            WITH DUPLICATES.
    SELECT FETUD-IND ASSIGN TO "fetutind.dat"
        ORGANIZATION INDEXED
        ACCESS MODE IS DYNAMIC
        RECORD KEY F-ETU-NUM-IND
        ALTERNATE RECORD KEY F-ETU-NOM-IND
            WITH DUPLICATES
        ALTERNATE RECORD KEY F-ETU-ANAISS-IND
            WITH DUPLICATES.
DATA DIVISION.
FILE SECTION.
FD  FETUD.
01  F-ETUDIANT.
    02  F-ETU-NUM PIC 999.
    02  F-ETU-NOM PIC X(20).
    02  F-ETU-PRENOM  PIC X(20).
    02  F-ETU-ADRESSE PIC X(40).
    02  F-ETU-ANAISS PIC 9(4).
FD  FETUD-IND-CRE.
01  F-ETUDIANT-IND-CRE.
    02  F-ETU-NUM-IND-CRE PIC 999.
    02  F-ETU-NOM-IND-CRE PIC X(20).
    02  F-ETU-PRENOM-IND-CRE PIC X(20).
    02  F-ETU-ADRESSE-IND-CRE PIC X(40).
    02  F-ETU-ANAISS-IND-CRE PIC 9(4).
FD  FETUD-IND.
01  F-ETUDIANT-IND.
    02  F-ETU-NUM-IND PIC 999.
    02  F-ETU-NOM-IND PIC X(20).
    02  F-ETU-PRENOM-IND PIC X(20).
    02  F-ETU-ADRESSE-IND PIC X(40).
    02  F-ETU-ANAISS-IND PIC 9(4).
WORKING-STORAGE SECTION.
01  W-ETUDIANT.
    02   W-NUM PIC 999.
    02   W-NOM PIC X(20).
    02   W-PRENOM PIC X(20).
    02   W-ADR PIC X(40).
    02   W-ANAISS PIC 9 !4).
77  W-NOM-CLE PIC X(20).
77  W-NUM-CLE PIC 999.
77  W-CHOIXMENU PIC X.
    88  QUITTE VALUE "Q" "q".
    88  CHOIX-OK VALUE 1 2 3 "Q" "q".
77  W-FINFIC PIC 9.
    88  FINFIC VALUE "0".
    88  PASFINFIC VALUE "N".
77  W-SUIVANT PIC A.
    88  SUIVANT VALUE "S" "s".
    88  QUITTER VALUE "Q" "q".
77  W-AUTRECLE PIC A.
    88  AUTRECLE VALUE "O" "o".
    88  PASAUTRECLE VALUE "N" "n".
77  W-EXISTE PIC A.
    88  EXISTE VALUE "O".
    88  NONEXISTE VALUE "N".
```


2.2 Normes relatives aux commentaires

En principe, il n'y a pas de programme sans algorithme. Nous recommandons d'insérer dans le code, à des points "stratégiques", une partie de cet algorithme. Dans ce domaine, il faut des commentaires en quantité suffisante : ni trop, ni trop peu.

Ces commentaires (ils sont, rappelons-le, annoncés, matérialisés, dans le programme par des étoiles en colonne 7) peuvent être insérés en début de procédure et servir à annoncer le contenu de celle-ci :

a) procédure ne respectant pas la norme :

```
PROCEDURE DIVISION.
    PERFORM DEB.
    MOVE "F" TO WOK, WARRET.
    PERFORM ETAPE1 UNTIL WOK = "V" OR WARRET = "V".
    IF WOK = V THEN
        MOVE F TO WOK, WARRET
        PERFORM ETAPE2 UNTIL WOK = "V" OR WARRET = "V"
        IF WOK = V THEN
            MOVE F TO WOK, WARRET
            PERFORM ETAPE3 UNTIL WOK = V OR WARRET = V
            IF WOK = V THEN
                MOVE F TO WOK, WWARRET
                PERFORM ETAPE4 UNTIL WOK = V OR WARRET = V
            END-IF
        END-IF
    END-IF.
    PERFORM FIN.
```

b) la même, respectant la norme :

```
1       7A   B
        PROCEDURE DIVISION.
*
*Procédure de saisie des programmes d'entraînement des
*coureurs. Pour être acceptés, les programmes doivent
*vérifier les contraintes suivantes :
*- le coureur existe dans le fichier des coureurs ;
*- la semaine existe dans le fichier des semaines ;
*- le programme n'existe pas encore dans le fichier des
*   programmes.
*
*Cette procédure est découpée en six parties :
*
*- DEB, qui se charge des initialisations,
*- ETAPE1, qui vérifie l'existence du coureur,
*- ETAPE2, qui vérifie l'existence de la semaine,
*- ETAPE3, qui vérifie que le programme n'existe pas,
*- ETAPE4, qui enregistre le programme,
*- FIN, qui ferme les fichiers et demande à l'utilisateur
*   s'il veut revenir au menu ou arrêter.
*
*Une étape N n'est abordée que si l'étape N-1 est réussie
* (W-OK à vrai).
```

```
*
    PERFORM DEB.
    MOVE "F" TO WOK, WARRET.
    PERFORM ETAPE1 UNTIL WOK = "V" OR WARRET = "V".
    IF WOK = V THEN
        MOVE F TO WOK, WARRET
        PERFORM ETAPE2 UNTIL WOK = "V" OR WARRET = "V"
        IF WOK = V THEN
            MOVE F TO WOK, WARRET
            PERFORM ETAPE3 UNTIL WOK = V OR WARRET = V
            IF WOK = V THEN
                MOVE F TO WOK, WWARRET
                PERFORM ETAPE4 UNTIL WOK = V OR WARRET = V
            END-IF
        END-IF
    END-IF.
    PERFORM FIN.
```

Ils peuvent aussi être insérés en milieu de procédure, aux points de contrôle :

```
1       7 A   B
      PROCEDURE DIVISION.
      *
      * Procédure de saisie des programmes d'entraînement des
      * coureurs. Pour être acceptés, les programmes doivent
      * vérifier les contraintes suivantes :
      * - le coureur existe dans le fichier des coureurs ;
      * - la semaine existe dans le fichier des semaines ;
      * - le programme n'existe pas encore dans le fichier des
      *   programmes.
      *
      * Cette procédure est découpée en six parties :
      *
      * - DEB, qui se charge des initialisations,
      * - ETAPE1, qui vérifie l'existence du coureur,
      * - ETAPE2, qui vérifie l'existence de la semaine,
      * - ETAPE3, qui vérifie que le programme n'existe pas,
      * - ETAPE4, qui enregistre le programme,
      * - FIN, qui ferme les fichiers et demande à l'utilisateur
      * s'il veut revenir au menu ou arrêter.
      *
      * Une étape N n'est abordée que si l'étape N-1 est réussie
      *  (W-OK à vrai).
      *
           PERFORM DEB.
           MOVE "F" TO WOK, WARRET.
           PERFORM ETAPE1 UNTIL WOK = "V" OR WARRET = "V".
           IF WOK = V THEN
      *
      * Le coureur existe, on vérifie la semaine
      *
```



```
               MOVE F TO WOK, WARRET
               PERFORM ETAPE2 UNTIL WOK = "V" OR WARRET = "V"
               IF WOK = V THEN
      *
      * Coureur et semaine existent, on vérifie le programme
      *
                   MOVE F TO WOK, WARRET
                   PERFORM ETAPE3 UNTIL WOK = V OR WARRET = V
                   IF WOK = V THEN
      *
      * Coureur et semaine existent, le programme n'existe pas,
      * on peut le créer
      *
                       MOVE F TO WOK, WWARRET
                       PERFORM ETAPE4 UNTIL WOK = V OR WARRET = V
                   END-IF
               END-IF
           END-IF.
      *
      * Saisie et enregistrement terminés (éventuellement mal) ;
      * On demande à l'utilisateur s'il veut continuer.
      *
           PERFORM FIN.
```

Il peut également y avoir un vrai dictionnaire de données dans la DATA DIVISION, chaque variable étant définie en COBOL et en français :

```
 WORKING-STORAGE SECTION.
*
* W-ETUDIANT : zone de stockage temporaire des infos sur l'étudiant
*
 01  W-ETUDIANT.
     02  W-NUM PIC 999.
     02  W-NOM PIC X(20).
     02  W-PRENOM PIC X(20).
     02  W-ADR PIC X(40).
     02  W-ANAISS PIC 9(4).
*
* W-NUM-CLE : numéro de l'étudiant recherché
* W-NOM-CLE : nom de l'étudiant recherché
* W-CHOIXMENU : fonction demandée par l'utilisateur (1, 2, 3 ou Q)
* W-FINFIC : fin de fichier détectée (1 ou 2)
* W-SUIVANT : étudiant suivant demandé ou non (S ou Q)
* W-AUTRECLE : autre recherche sur la clé souhaitée (O ou N)
* W-EXISTE : résultat de la recherche (O ou N)
*
 77  W-NOM-CLE PIC X(20).
 77  W-NUM-CLE PIC 999.
 77  W-CHOIXMENU PIC X.
     88  QUITTE VALUE "Q" "q".
     88  CHOIX-OK VALUE 1 2 3 "Q" "q".
 77  W-FINFIC PIC 9.
     88  FINFIC VALUE "O".
     88  PASFINFIC VALUE "N".
 77  W-SUIVANT PIC A.
     88  SUIVANT VALUE "S" "s".
     88  QUITTER VALUE "Q" "q".
```

```
 77  W-AUTRECLE PIC A.
     88  AUTRECLE VALUE "O" "o".
     88  PASAUTRECLE VALUE "N" "n".
 77  W-EXISTE PIC A.
     88  EXISTE VALUE "O".
     88  NONEXISTE VALUE "N".
```

NB : cette description correspond à l'exemple que nous avons présenté plus haut dans ce paragraphe et aussi dans un chapitre consacré à la description des variables de travail.

2.3 Normes relatives aux procédures

Une procédure est constituée d'un ensemble, parfois fort long, de lignes de codes. Ces lignes de code ne sont pas exécutées sans contrôle. Il y a des structures qui pilotent cet ensemble (alternatives, boucles...). Il est vital pour le programmeur qui met au point le programme, mais aussi pour celui qui devra, plus tard, le reprendre pour le modifier, que ces structures de contrôle soivent visibles. **Nous recommandons de mettre en évidence les structures de contrôle en indentant le code (en créant des décalages).** L'exemple ci-après permet de juger de l'aide que représente une telle présentation :

a) procédure ne respectant pas cette norme :

```
 SAISIR-CHOIX.
     IF WGENRE = 0 THEN
     PERFORM NOMBRE-SAISIE
     IF WCP-SAISIE >= 3 THEN
     DISPLAY "M)enu Q)uitter" LINE 21
     DISPLAY "Votre choix : " LINE 22 COL 10
     ACCEPTE WCHOIX LINE 22 COL 25
     IF WCHOIX = "M" OR WCHOIX = "m" OR WCHOIX = "Q"
     OR WCHOIX = "q" THEN
     MOVE 1 TO WSAISIE-VALIDE
     ELSE DISPLAY "Vous avez effectué un mauvais choix : "
     LINE 23.
```

NB : nous avons effectivement trouvé cette procédure dans un travail d'étudiant. Nous lui avons bien entendu demandé de corriger.

b) la même, avec indentation :

```
SAISIR-CHOIX.
     IF WGENRE = 0 THEN
         PERFORM NOMBRE-SAISIE
         IF WCP-SAISIE >= 3 THEN
             DISPLAY "M)enu Q)uitter" LINE 21
             DISPLAY "Votre choix : " LINE 22 COL 10
           ACCEPTE WCHOIX LINE 22 COL 25
           IF WCHOIX = "M" OR WCHOIX = "m" OR WCHOIX = "Q"
           OR WCHOIX = "q" THEN
              MOVE 1 TO WSAISIE-VALIDE
           ELSE
              DISPLAY "Vous avez effectué un mauvais choix : "
              LINE 23
             END-IF
         END-IF
     END-IF.
```

Une procédure est une unité de lecture. Elle a un rôle précis dans le programme. Celui-ci est, en général, soit affiché sur un écran, soit imprimé sur un papier de format A4 ou avoisinant le A4.

Nous recommandons d'avoir des procédures de taille suffisante, ni trop, ni trop peu. La borne supérieure nous paraît être clairement la page ou l'écran, ce qui correspond à peu près à 20-25 lignes de code. La borne inférieure est située aux environs de la dizaine de lignes. Il n'y a rien de pire qu'un programme ayant beaucoup de toutes petites procédures. Un programme ainsi fragmenté impose à celui qui essaye d'en comprendre la structure et le rôle, des allers et retours incessants sur le papier et lui occasionnera sans aucun doute des maux de tête. L'exemple ci-après est assez parlant, pensons-nous. Nous préférons, et de loin, le second.

Si, malgré tout, la procédure "persiste" à faire plus de 20-25 lignes, rappelons qu'il est toujours possible de forcer le compilateur à sauter à la page suivante en utilisant un / en colonne 7.

(Contre) Exemple P04

Ce programme est une réponse à un des sujets évoqués dans les fiches de travaux dirigés. Il est constitué de plusieurs procédures que l'on appelle par des PERFORM. Il s'agit, selon nous, d'un programme construit avec de bonnes intentions, mais conduisant à un mauvais résultat. Il est, en effet, beaucoup trop morcelé. Nous en donnons une "correction", bâtie selon d'autres principes. Nous laissons les apprenants juger.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. P3.
**********************
 ENVIRONMENT DIVISION.
**********************
 CONFIGURATION SECTION.
**********************
 SPECIAL-NAMES.
    CONSOLE IS CRT.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
    SELECT FETUD ASSIGN TO "fetud.dat".
**************
 DATA DIVISION.
**************
 FILE SECTION.
*************
 FD FETUD.
 01 FETUDIANT.
    02 FNUM PIC 999.
    02 FNOM PIC X(20).
    02 FPRENOM PIC X(20).
    02 FADR PIC X(25).
    02 FANAISS PIC 9(4).
 WORKING-STORAGE SECTION.
************************
    77 WVALIDE PIC X VALUE "O".
    77 WCHOIX PIC X.
    77 WFIN PIC X VALUE "N".
    77 WNBETUD PIC 9 VALUE 0.
    77 WI PIC 9 VALUE 0.
    77 WJ PIC 99 VALUE 5.
 PROCEDURE DIVISION.
*******************
*Vérifie la validité de wnbetud
    PERFORM SAISIENB UNTIL WNBETUD>=1 AND WNBETUD<=3.
    OPEN INPUT FETUD.
    PERFORM OBTENIR.
    PERFORM AFFPAGE UNTIL (WCHOIX="Q" OR WCHOIX="q") OR WFIN="O".
    CLOSE FETUD.
    STOP RUN.
*Affichage d'une page
**********************
```

```
 AFFPAGE.
    MOVE 0 TO WI.
    MOVE 5 TO WJ.
    DISPLAY " " LINE 5 WITH BLANK SCREEN.
    PERFORM AFFICH UNTIL WI>=WNBETUD OR WFIN="O".
    MOVE " " TO WCHOIX.
*Vérifie la validité de wchoix
    PERFORM SAISIECHOIX
       UNTIL WCHOIX="Q" OR WCHOIX="q" OR WCHOIX="S" OR WCHOIX="s".
*Procédure d'affichage d'un étudiant
*********************************
 AFFICH.
    DISPLAY "NUMERO : " FNUM LINE WJ COL 1.
    DISPLAY "NOM : " FNOM LINE WJ COL 20.
    DISPLAY "PRENOM : " FPRENOM LINE WJ COL 40.
    ADD 1 TO WJ.
    DISPLAY "ADRESSE : " FADR LINE WJ COL 1.
    DISPLAY "ANNEE DE NAISSANCE : " FANAISS LINE WJ COL 40.
    ADD 1 TO WI.
    ADD 3 TO WJ.
    PERFORM OBTENIR.
*Procédure de lecture d'un étudiant
*********************************
 OBTENIR.
    READ FETUD AT END MOVE "O" TO WFIN.
*Procédure qui demande le nb d'étudiants à afficher par page
************************************************************
 SAISIENB.
    DISPLAY "Combien d'étudiant souhaitez-vous afficher par paquet ?"
      LINE 5 COL 5.
    DISPLAY "(le nb doit être compris entre 1 et 3)" LINE 6 COL 5.
    ACCEPT WNBETUD LINE 7 COL 5.
*Procédure qui propose de continuer ou de quitter
************************************************************
 SAISIECHOIX.
    DISPLAY "Q)uitter" LINE 22 COL 1.
    IF WFIN="N" DISPLAY "S)uivant" LINE 22 COL 10
        DISPLAY "Votre choix (réponse Q,q,S,s)" LINE 23 COL 1
    ELSE
        DISPLAY "Votre choix (rèponse Q,q)" LINE 23 COL 1.
    ACCEPT WCHOIX LINE 23 COL 40.
```

(Bon) Exemple P05

Ce programme est une réponse à un des sujets évoqués dans les fiches de travaux dirigés. Nous sommes partis d'une réponse fournie par un de nos étudiants et nous l'avons modifiée. L'original est constitué de plusieurs procédures que l'on appelle par des PERFORM. Il s'agit, selon nous, d'un programme construit avec de bonnes intentions, mais conduisant à un mauvais résultat. Il est, en effet, beaucoup trop morcelé. La solution que nous présentons ici s'est "affranchie " de cette structuration en procédures. Nous laissons les apprenants juger.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. P3.
*********************
 ENVIRONMENT DIVISION.
*********************
 CONFIGURATION SECTION.
*********************
 SPECIAL-NAMES.
    CONSOLE IS CRT.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
```



```
    SELECT FETUD ASSIGN TO "fetud.dat".
**************
 DATA DIVISION.
**************
 FILE SECTION.
*************
 FD FETUD.
 01 FETUDIANT.
    02 F-ETU-NUM PIC 999.
    02 F-ETU-NOM PIC X(20).
    02 F-ETU-PRENOM PIC X(20).
    02 F-ETU-ADR PIC X(25).
    02 F-ETU-ANAISS PIC 9(4).
 WORKING-STORAGE SECTION.
***********************
    77 W-VALIDE PIC X VALUE "O".
    77 W-CHOIX PIC X VALUE SPACE.
        88 W-CHOIXQ VALUE "Q" "q".
        88 W-CHOIXS VALUE "S" "s".
    77 W-FIN PIC X VALUE "N".
    77 W-NBETUD PIC 9 VALUE 0.
    77 W-I PIC 9 VALUE 0.
    77 W-J PIC 99 VALUE 5.
 PROCEDURE DIVISION.
*******************
    PERFORM UNTIL W-NBETUD >= 1 AND W-NBETUD <= 3
*
*       Saisie du nb d'étudiants par page
*
        DISPLAY "Nb d'étudiants à afficher par paquet ?" LINE 5 COL 5
        DISPLAY "(le nb doit être entre 1 et 3)" LINE 6 COL 5
        ACCEPT W-NBETUD LINE 7 COL 5
    END-PERFORM.
    OPEN INPUT FETUD.
    READ FETUD AT END MOVE "O" TO W-FIN.
    PERFORM UNTIL W-CHOIXQ OR W-FIN="O"
*
*       Affichage d'une page
*
        MOVE 0 TO W-I.
        MOVE 5 TO W-J.
        DISPLAY " " LINE 5 WITH BLANK SCREEN.
        PERFORM UNTIL W-I >= W-NBETUD OR W-FIN = "O"
*
*           Affichage d'un étudiant
*
            DISPLAY "NUMERO : " F-ETU-NUM LINE W-J COL 1
            DISPLAY "NOM : " F-ETU-NOM LINE W-J COL 20
            DISPLAY "PRENOM : " F-ETU-PRENOM LINE W-J COL 40
            ADD 1 TO W-J
            DISPLAY "ADRESSE : " F-ETU-ADR LINE W-J COL 1
            DISPLAY "ANNEE DE NAISSANCE : " F-ETU-ANAISS LINE W-J COL 40
            ADD 1 TO W-I
            ADD 3 TO W-J
            READ FETUD AT END MOVE "O" TO W-FIN END-READ
        END-PERFORM.
        MOVE " " TO W-CHOIX.
        PERFORM UNTIL W-CHOIXQ OR W-CHOIXS
*
*       Saisie choix utilisateur (continuer ou quitter)
*
            DISPLAY "Q)uitter" LINE 22 COL 1.
            IF W-FIN="N" THEN
```

```
            DISPLAY "S)uivant" LINE 22 COL 10
            DISPLAY "Votre choix (réponse Q, q, S, s)"
                LINE 23 COL 1
        ELSE
            DISPLAY "Votre choix (réponse Q, q)"
                LINE 23 COL 1
        END-IF
        ACCEPT W-CHOIX LINE 23 COL 40
    END-PERFORM
END-PERFORM.
CLOSE FETUD.
STOP RUN.
```

2.4 Normes relatives aux entrées-sorties

Dans la mesure où nous promouvons les techniques présentées dans le chapitre 3 (cinématique de fichiers), nous recommandons de bien isoler les procédures d'entrées-sorties du reste du programme, surtout l'ordre de lecture.

```
1       7A   B
        *************************************************
        **
        *Procédure OBTENIR-FETUD de lecture d'un étudiant
        **
        *************************************************
        **
        OBTENIR-FETUD.
            READ FETUD AT END MOVE "O" TO W-FIN.
```

Le recours systématique à une procédure appelée OBTENIR-XXX qui se charge de récupérer un enregistrement du fichier XXX (voir sa définition dans le chapitre consacré à la cinématique de fichiers) garantit lisibilité (les instructions READ, avec leurs clauses AT END..., ne viennent pas "polluer" le programme) et efficacité (si, pour une raison quelconque, le fichier d'origine -celui que nous avons appelé XXX ci-dessus- change de support, d'organisation et/ou de nature, seule la procédure OBTENIR-XXX devra être modifiée. Le corps du programme, l'essentiel, reste inchangé).

2.5 Conclusion

Les normes de programmation que nous souhaitons promouvoir sont au nombre de cinq :

- N1 : recours systématique à des noms de variables ayant un sens,
- N2 : noms de variables permettant une localisation rapide de leur définition,
- N3 : procédure indentée (les structures de contrôle mises en évidence), de taille suffisante, comprise entre 10 et 25 lignes par page,
- N4 : commentaires en nombre significatif dans le programme,
- N5 : procédures de lectures séparées nettement du reste du programme.

Leur application est indispensable.

27/01/06

COBOL et le génie logiciel

3. Instruction COPY

Le but d'une instruction COPY est de permettre au programmeur d'inclure dans un programme COBOL des lignes qui ont été enregistrées au préalable dans un fichier. Ces lignes de code seront insérées à la place de l'instruction COPY, le programme ainsi obtenu étant ensuite compilé. L'intérêt d'un tel dispositif réside dans le fait qu'une fois stockés dans une bibliothèque, ces modules peuvent être introduits dans plusieurs programmes en une seule instruction. Une sorte de couper-coller en quelque sorte.

Il y a plusieurs syntaxes pour cette instruction. La plus simple est la suivante :

```
COPY Nom-module.
```

Le module doit être enregistré dans un fichier ayant pour nom Nom-module. Il doit être dans le même répertoire que le programme dans lequel il sera inséré. Il ne peut excéder 16Ko. Il ne doit pas contenir lui-même d'instruction COPY.

Il est possible de fournir un nom de répertoire différent :

```
COPY Nom-module IN Nom-répertoire.
```

Il est également possible de définir des modules génériques, dans lesquels des paramètres ont été placés et de les instancier lors de l'exécution du COPY (clause REPLACING).

Consulter la documentation de l'éditeur pour plus de renseignements sur ces deux dernières options.

09/02/04
