INSTITUT de la gestion publique et du développement économique

Support stagiaire

COBOL
MODULE 4
Synthèse

MINISTÈRE DE L'ÉCONOMIE
DES FINANCES ET DE L'INDUSTRIE

Sommaire

FICHE 01

Rendre modulaire par les sous-programmes

1. OBJECTIF.

Avec l'instruction PERFORM, des « séquences » répétitives de « code » (instructions) peuvent être écrites sous forme de « blocs » indépendants.

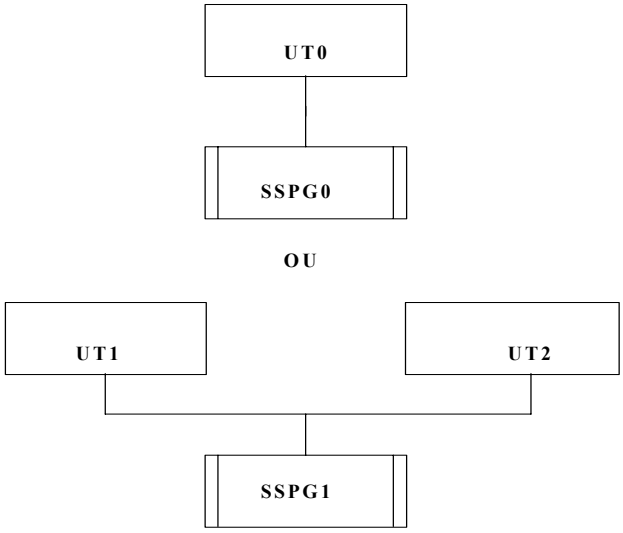
On peut écrire de telles séquences sous forme de programmes indépendants « appelés » par un programme principal, ou éventuellement plusieurs programmes principaux.

Le plus souvent les « sous-programmes » sont utilisés :

pour des parties de programmes particulièrement complexes, et/ou

par différentes « unités de traitement » (UT), et/ou

offrir des améliorations de performances, lorsqu'ils sont écrits en langage assembleur (par exemple).



NOTES

2. LES SOUS-PROGRAMMES EXTERNES.

2.1. PRINCIPES.

Les sous-programmes externes sont des programmes écrits en COBOL, ou tout autre langage, et compilés indépendamment du programme principal appelant.

Programme principal et sous-programmes sont ensuite reliés par le programme éditeur de liens.

Le sous-programme COBOL est bâti comme tout programme COBOL avec au minimum les divisions suivantes :

- IDENTIFICATION DIVISION.
- PROCEDURE DIVISION.

La différence avec le programme COBOL normal réside dans l'introduction :

- de clauses « **USING** » dans la PROCEDURE DIVISION, ou dans ENTRY ;
- d'une **LINKAGE SECTION** dans le sous-programme, pour établir une liaison avec les zones de données du programme principal désignées dans la clause « **USING** » de l'ordre « **CALL** ».

L'appel d'un sous-programme externe (branchement du traitement depuis le programme principal COBOL), vers le sous-programme est réalisé par l'ordre « **CALL** ».

NOTES

2.2. EXEMPLE 1 : APPELANT / APPELÉ ; TRANSFERT DE PARAMÈTRES.

PROGRAMME PRINCIPAL

```
8   12
IDENTIFICATION DIVISION.
PROGRAM-ID. PRINC001.
ENVIRONMENT DIVISION.
……
DATA DIVISION.
FILE SECTION.
FD ENTREE
01 E-ART..
    02 E-COD PIC X.
    02 E-IND  PIC 9(15).
    02              PIC X(64).
WORKING-STORAGE SECTION.
01 W-ART.
    02 W-COD PIC X.
    02 W-IND PIC 9(15).
    02 W-LIB PIC 9(16) OCCURS 4.
PROCEDURE DIVISION.

    ……
    CALL « SSPROG01 » USING E-ART W-
    ART.
    ……
    STOP RUN.
```

SOUS-PROGRAMME

```
8  12
IDENTIFICATION DIVISION.
PROGRAM-ID. SSPROG01.
ENVIRONMENT DIVISION.
……
DATA DIVISION.
……
LINKAGE SECTION.
01 K-ART1.
   02 K-COD1 PIC X.
   02 K-IND1 PIC 9(15).
   02              PIC X(64).
01 K-ART2.
   02 K-COD2 PIC X.
   02 K-IND2 PIC 9(15).
   02 K-LIB PIC X(16) OCCURS 4.

PROCEDURE DIVISION USING K-ART1 K-
ART2
   ……
   ……

FINSSPROG01.
   EXIT PROGRAM.
```

NOTES

2.3. DESCRIPTION DES DONNÉES COMMUNES.

Le programme principal et le sous-programme COBOL sont compilés séparément.

Il est donc nécessaire que la description des zones de données communes apparaisse dans les DATA DIVISION du programme principal et du sous-programme.

La **LINKAGE SECTION** permet de décrire des zones de données sans réserver de place en mémoire centrale, mais en leur attribuant une adresse virtuelle dans le sous-programme.

Il était exclu de réserver deux fois ces zones en mémoire centrale.

Une zone de données commune sera donc décrite normalement dans le programme principal en FILE SECTION ou WORKING-STORAGE SECTION et sera décrite dans le sous-programme dans la LINKAGE SECTION.

L'ordre d'appel de sous-programme « **CALL** » aura donc pour effet d'attribuer des adresses aux zones de données de la LINKAGE SECTION.

NOTES

2.4. CORRESPONDANCE DES DONNÉES.

2.4.1. *PROCEDURE DIVISION …. USING ….*

Si l'entrée du sous-programme est le premier ordre de la PROCEDURE DIVISION, la clause « **USING** » suit cet ordre :

PROCEDURE DIVISION [USING nom-donnée-A [nom-donnée-B]]

Dans la liaison effectuée entre le programme principal et le sous-programme, on fait correspondre :

- « nom-donnée-1 » du programme principal à « nom-donnée-A » du sous-programme;
- « nom-donnée-2 » du programme principal à « nom-donnée-B » du sous-programme;
- etc…

2.4.2. ENTRY …. USING ….

Si l'entrée du sous-programme n'est pas le premier ordre de la PROCEDURE DIVISION, le point d'entrée dans le sous-programme est désigné par une clause **ENTRY** incluant la clause **USING** avec le format :

ENTRY nomsymboliqueentrée [USING nomdonnéeA [nom-donnée-B]]

De la même façon, on fait correspondre ici les noms de zones de données deux à deux suivant leur rang :

- « nom-donnée-1 » du programme principal à « nom-donnée-A » du sous-programme;
- « nom-donnée-2 » du programme principal à « nom-donnée-B » du sous-programme;
- etc…

NOTES

2.5. APPEL D'UN SOUS-PROGRAMME ET TRANSFERT DE PARAMÈTRES : « CALL … USING … »

Le programme principal COBOL doit contenir dans le corps de traitement (en PROCEDURE DIVISION) le verbe « **CALL** » pour réaliser le traitement contenu dans un sous-programme externe.

Le format de cette instruction est le suivant :

```
CALL   { nom-donnée-1 }
       { littéral-1   }
       [[ USIN  { BY REFERENCE } nom-donnée-2 ]..]
       [  G     { BY CONTENT   }              ]
       [ ON OVERFLOW ordre-impératif-1 ]
       [ [ ON EXCEPTION ordre-impératif-2 ]                ]
       [ [ NOT ON EXCEPTION ordre-impératif-3]             ]
[ END-CALL ]
```

cf exemple 1 : CALL « SSPROG01 ».

Dans ce cas, l'adresse de branchement dans le sous-programme, encore appelée point-d'entrée, est le début même du sous-programme.

NOTES

2.5.1. *DÉSIGNATION DU SOUS PROGRAMME APPELÉ.*

« Nom-Donnée-1 ou Littoral » désigne le nom du sous-programme appelé, nom qui doit respecter la règle de formation des noms de programmes, c'est à dire au maximum 6 caractères pour le PC et 8 caractères autrement, dont le premier est obligatoirement alphabétique.

Avec « Nom-Donnée-1 », le nom du sous-programme est inclus dans la zone de données.

Avec « Littéral-1 », le nom du sous-programme est directement explicité dans l'instruction (cas général).

2.5.2. PARTAGE DE DONNÉES : CLAUSE USING.

```
USING { BY REFERENCE }   nom-donnée-2
      BY CONTENT
```

Auparavant le COBOL ne comportait que la clause BY REFERENCE, aujourd'hui option par défaut.

Cette clause signifie que le programme appelant et le sous-programme appelé ont en commun une zone mémoire « Nom-Donnée-2 », ou plusieurs zones.

cf exemple 1 :

```
CALL « SSPROG01 » USING E-ART W-ART.
```

Il est impératif que les descriptions de zones de données soient identiques dans les deux programmes.

NOTES

Et, puisque ces zones de données sont communes, le sous-programme appelé peut très bien modifier le contenu des zones. Par exemple, on peut appeler un sous-programme de conversion de chiffres, en écrivant :

```
CALL « CNVCHLET » USING E-CHIFFRE S-LETTRE.
```

On passe au sous-programme la valeur contenue dans « E-CHIFFRE » qui renvoie au programme la valeur correspondante « cent onze » dans « S-LETTRE ».

Les transmissions avec d'autres langages sont possible (voir les consignes de chaque constructeur).

Nous verrons plus loin qu'avec les données « EXTERNAL », existe un nouveau mode de partage des données.

« BY CONTENT » est sensiblement identique à « BY REFERENCE », si ce n'est que le sous-programme ne peut modifier le contenu de la (ou des) zone(s) transmise(s).

2.5.3. *GESTION DES DÉPASSEMENTS OU DES ANOMALIES.*

```
[ ON OVERFLOW ordre-impératif-1 ]
[ [ ON EXCEPTION ordre-impératif-2 ]
  [ NOT ON EXCEPTION ordre-impératif-3 ] ]
```

« OVERFLOW » et « EXCEPTION » ont sensiblement le même usage, si ce n'est que « OVERFLOW » agit dans le cas où il n'est pas possible d'introduire le sous-programme en mémoire alors que « EXCEPTION » est destiné à tous types d'anomalies détectées dans l'exploitation du sous-programme.

NOTES

2.5.4. TERMINAISON D'UN CALL : STRUCTURATION EN « BLOC ».

[END-CALL]

« END-CALL » est le délimiteur facultatif de l'instruction « CALL ».

2.5.5. REMARQUES.

La déclaration des paramètres peut se faire à des niveaux autres que 01 ou 77.

Le sous-programme ne comporte pas obligatoirement le verbe « **EXIT** ». Si la fin est atteinte « **EXIT** » est implicite.

Si le verbe « **EXIT** » est utilisé, cette instruction n'est pas obligatoirement la seule instruction du paragraphe.

On peut transférer :

- Les adresses des données « **BY REFERENCE** » valeur par défaut.
- Les contenus des données « **BY CONTENT** ». En ce cas on a la certitude de supprimer les effets de bord (non perturbation des données du programme appelant).

Pas de récursivité en COBOL 85.

NOTES

2.6. TERMINAISON D'UN SOUS-PROGRAMME.

Dans un sous-programme, la terminaison est réalisée par les clauses « **EXIT PROGRAM** » ou « **GOBACK** ».

```
Nom-Paragraphe.
        EXIT-PROGRAM.
```

Le paragraphe ne peut contenir d'autres instructions que « EXIT PROGRAM ».

```
        GOBACK.
```

L'effet de ces deux clauses est identique dans un sous-programme appelé par une instruction « CALL »; lorsque le traitement passe sur une telle clause, celui-ci est automatiquement rebranché juste après l'ordre « CALL » du programme principal appelant.

Si le programme n'a pas été appelé par un « CALL », l'instruction « EXIT PROGRAM » est simplement ignorée lorsque le traitement passe sur ce paragraphe. Par contre, dans le cas de « GOBACK », on a un effet identique à la clause « STOP RUN ».

NOTES

3. LES SOUS-PROGRAMMES INTERNES.

3.1. PRINCIPES.

Sous-programmes internes, contenus, imbriqués ou inclus.

L'évolution des techniques de programmation structurée a conduit les experts à introduire dans Cobol des programmes contenus, plus ou moins copiés sur les procédures internes Pascal ou PL/1.

Les différents programmes peuvent être compilés en une seule « passe », à l'aide d'un « fichier multiprogramme ». On appelle cela une « procédure batch ».

Comme pour les programmes externes (compilés séparément), les programmes contenus peuvent être appelés par un ordre « **CALL** » ou remis à l'état initial par un ordre « **CANCEL** ».

Dans un programme contenu, les divisions « ENVIRONMENT », « DATA » et même « PROCEDURE » sont facultatives.

Les données communes aux différents programmes peuvent être décrites une seule fois. Les zones de manoeuvre dans les sous-programmes sont connues uniquement de ceux-ci.

Un programme source peut contenir d'autres programmes sources pouvant utiliser des noms de données définies dans le programme contenant.

Le programme imbriqué se place à la fin du programme contenant mais avant la ligne contenant « **END PROGRAM** ».

Plusieurs niveaux d'imbrication sont possibles.

Pour simplifier l'exposé, nous partirons de l'exemple ci-après, le programme « PRINC1 » contient les programmes internes « P1PROG1 » et « P1PROG3 » ; le programme « P1PROG1 » contient le programme « P1PROG2 ».

NOTES

3.2. EXEMPLE 2 : IMBRICATION - TRANSFERT DE PARAMÈTRES.

```
8     12
IDENTIFICATION DIVISION.
PROGRAM-ID. PRINC1.
DATA DIVISION.
01 E-ART GLOBAL
       ……..
PROCEDURE DIVISION.
       CALL « P1PROG1 ».
       CALL « P1PROG3 ».
  IDENTIFICATION DIVISION.
  PROGRAM-ID. P1PROG1 INITIAL.
  DATA DIVISION.
  01 E-ART.
         ……..
  PROCEDURE DIVISION.
         CALL « P1PROG2 »
         MOVE E-ART TO P1P1-ART.
         CALL « P1PROG3 ».
    IDENTIFICATION DIVISION.
    PROGRAM-ID. P1PROG2.
    PROCEDURE DIVISION.
           CALL « P1PROG3 »
           MOVE E-ART TO P1P2-ART.
    END-PROGRAM P1PROG2.
  END-PROGRAM P1PROG1.
  IDENTIFICATION DIVISION.
  PROGRAM-ID. P1PROG3.
  PROCEDURE DIVISION.
         MOVE E-ART TO P1P3-ART.
  END-PROGRAM P1PROG3.
END-PROGRAM PRINC1.
```

NOTES

3.3. VUE COMMUNE D'UN SOUS-PROGRAMME « COMMON ».

Un programme contenu ne peut être appelé que par un programme qui le contient au niveau immédiatement supérieur.

L'attribut « **COMMON** » affecté au nom d'un programme contenu permet, d'appeler ce programme depuis n'importe quel autre programme de l'entité compilée.

Par exemple, le programme « P1PROG3 » de l'exemple 2 est connu de « PRINC1 », « P1PROG1 » et « P1PROG2 » alors que « P1PROG1 » n'est connu que de « PRINC1 » et que « P1PROG2 » n'est connu que de « P1PROG1 ».

3.3.1. PROGRAMMES IMBRIQUÉS.

```
PRINCIP1
CALL « P1PROG1 »
CALL « P1PROG2 »
DISPLAY « IMBRIC1 »
   P1PROG1
   CALL « P1PROG3 »
   DISPLAY « IMBRIC2 »
     P1PROG3.
     DISPLAY « IMBRIC3 »


   P1PROG2
   DISPLAY « IMBRIC2 »
```

Un sous-programme peut appeler un sous-programme contenu directement.

```
PRINC1.
CALL « P1PROG1 »
CALL « P1PROG2 »
DISPLAY « IMBRIC1 »
  P1PROG2
  CALL « P1PROG1 » *
  CALL « P1PROG3 »
  DISPLAY « IMBRIC2 »
    P1PROG3.
    CALL « P1PROG1 » *
    DISPLAY « IMBRIC3 »
  P1PROG1 IS COMMON
  DISPLAY « IMBRIC2 »
```

L'attribut **COMMON** rend l'appel possible par d'autres programmes imbriqués.

* Remarque : ces appels sont possibles parce que « P1PROG1 » est « COMMON ».

Attention : en « CALL » externe seul le programme de plus haut niveau peut être appelé (contenant).

NOTES

3.3.2. EXEMPLE 3 : PROGRAMME « COMMON ».

```
8     12
IDENTIFICATION DIVISION.
PROGRAM-ID. PRINC1.
PROCEDURE DIVISION.
       DISPLAY « DEBUT PRINC1 ».
       CALL « P1PROG1 ».
       CALL « P1PROG2 ».
       DISPLAY « RETOUR EN PRINC1 ».
       STOP RUN.
IDENTIFICATION DIVISION.
PROGRAM-ID. P1PROG1.
PROCEDURE DIVISION.
       DISPLAY « DEBUT-P1PROG1 ».
       CALL « P1PROG2 ».
       DISPLAY « RETOUR EN P1PROG1 ».
END-PROGRAM P1PROG1.
IDENTIFICATION DIVISION.
PROGRAM-ID. P1PROG2 IS COMMON.
PROCEDURE DIVISION.
DEBUT-P1PROG2.
       DISPLAY « DEBUT-P1PROG2 ».
END-PROGRAM P1PROG2.
END-PROGRAM PRINC1.
```

NOTES

Résultats :

```
DEBUT PRINC1.
DEBUT-P1PROG1.
DEBUT-P1PROG2.
RETOUR EN P1PROG1
DEBUT-P1PROG2.
RETOUR EN PRINC1.
```

« PRINC1 » peut faire appel à « P1PROG1 » et « P1PROG2 ».
« P1PROG1 » peut faire appel à « P1PROG2 » ainsi qu'à tous les sous-programmes qu'il pourrait contenir (parce que « **COMMON** »).
« P1PROG2 » ne peut pas faire appel à « P1PROG1 » (il n'est ni « **COMMON** » ni **inclus**).

3.4. VUE GLOBALE D'UNE DONNÉE : « GLOBAL ».

Un « nom-donnée » (niveau 01 ou 77) affecté de l'attribut « **GLOBAL** » est connu du programme dans lequel il est déclaré ainsi que de tous les programmes contenus dans ce programme (directement ou indirectement), à condition que le « nom-donnée » ne soit pas déclaré une seconde fois dans un programme contenu (la « portée » d'un « GLOBAL » se termine dès qu'un nom identique de zone est trouvé).

Les autres noms de données sont accessibles uniquement par les programmes qui les déclarent (homonymes possibles) : un « nom-donnée » déclaré sans attribut est par défaut « LOCAL », donc connu dans un seul programme.

Par exemple, « E-ART » décrit dans « PRINC1 » avec l'attribut « **GLOBAL** » est connu dans « PRINC1 », « P1PROG2 » et « P1PROG3 ».

NOTES

Par contre ZONA est « LOCAL » dans le programme « P1PROG1 » où il est déclaré une seconde fois.

PRINC1.
01 E-ART **GLOBAL**
Accès à E-ART
P1PROG1.
77 IND PIC 99.
Accès à E-ART et à IND de P1PROG1.

P1PROG2.
77 IND PIC 99.
Accès à E-ART et à IND de P1PROG2

L'attribut « **GLOBAL** » s'applique également aux noms de fichiers « **FD** » et aux noms d'état « **RD** » :

FD nom-de-fichier IS **GLOBAL.**
01 nom-de-donnée IS **GLOBAL**.

NOTES

3.5. L'ACCESSIBILITÉ DES DONNÉES : « EXTERNAL ».

L'attribut « **EXTERNAL** » s'applique essentiellement aux descriptions de fichiers dans la clause « **FD** » du programme principal sous la forme :

```
FD nom-de-fichier [IS EXTERNAL]
01 nom-de-donnée [IS EXTERNAL]
       description de fichier
```

Un fichier déclaré « **EXTERNAL** » signifie que ses enregistrements seront accessibles par tous les sous-programmes appelés par le programme principal, à condition de décrire ces enregistrements en WORKING-STORAGE SECTION des sous-programmes sous la même forme (niveau 01 ou 77).

Un objet COBOL (fichier, article ou donnée) peut être déclaré « **EXTERNAL** » par plusieurs programmes ou sous-programmes.

L'objet n'existe qu'une seule fois et est accessible par tous les programmes qui le déclarent.

```
PRINC1
01 ZON EXTERNAL.
```

```
PRINC2
01 ZON EXTERNAL.
```

Un seul objet « ZON » est accessible par « PRINC1 » et par « PRINC2 ».

Un objet peut être déclaré à la fois « **GLOBAL** » et « **EXTERNAL** ».

NOTES

3.6. RÉINITIALISATION DES DONNÉES : « INITIAL ».

3.6.1. DONNÉE STATIQUE / DONNÉE AUTOMATIQUE.

Les données déclarées en COBOL ont toujours été de type « statique », c'est-à-dire que l'état initial est affecté une seule fois à la compilation par la clause « **VALUE** » (sauf à utiliser l'instruction « **CANCEL** »).

L'attribut « **INITIAL** » affecté à un programme signifie que les données de ce programme seront de type « automatique ».

Dans ce cas les valeurs initiales sont affectées à chaque appel du programme, les fichiers sont remis à l'état initial (donc fermés à chaque appel), les instructions « **PERFORM** » sont repositionnées à l'état initial, etc…

```
PRINCIP1.
01 TAB GLOBAL
    02 … OCCURS …
    …..
    CALL « P1PROG1 ».
    …..
    CALL « P1PROG1 ».
 P1PROG1 IS INITIAL.
 77 P1 I PIC 99 VALUE 1.
 Après le premier CALL, P1-I varie de 1 à x.
 Au CALL suivant P1-I varie de 1 à y.
```

Un programme interne peut être déclaré simultanément « **COMMON** » et « **INITIAL** ».

L'attribut « **INITIAL** » permet d'éviter l'usage de l'instruction « **CANCEL** ».

Si un programme « **INITIAL** » a des programmes internes, ceux-ci seront également « **INITIAL** ».

NOTES

3.6.2. *EXEMPLE 4 : PROGRAMME « INITIAL ».*

```
IDENTIFICATION DIVISION.
PROGRAM-ID. PRINC1.
PROCEDURE DIVISION.
DEBUT PRINC1.
      DISPLAY « 1ER CALL DE P1PROG1 ».
      CALL « P1PROG1 ».
      DISPLAY « 2EME CALL DE P1PROG1 ».
      CALL « P1PROG1 ».
      DISPLAY « 1ER CALL DE P1PROG2 ».
      CALL « P1PROG2 ».
      DISPLAY « 2EME CALL DE P1PROG2 ».
      CALL « P1PROG2 ».
      STOP RUN.
IDENTIFICATION DIVISION.
PROGRAM-ID. P1PROG1.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 PA PIC 99 VALUE 10.
77 PB PIC 99 VALUE 5.
77 TOTAL PIC 9(4) VALUE 0.
PROCEDURE DIVISION.
DEBUT-P1PROG1.
      DISPLAY « TOTAL : » TOTAL.
      DISPLAY « PA : » PA.
      DISPLAY « PB : » PB.
      COMPUTE TOTAL = TOTAL + PA - PB.
      MOVE 1 TO PA PB.
      DISPLAY « TOTAL : » TOTAL.
      DISPLAY « PA : » PA.
      DISPLAY « PB : » PB.
END PROGRAM P1PROG1.
IDENTIFICATION DIVISION.
PROGRAM-ID. P1PROG2 IS INITIAL
DATA DIVISION.
WORKING-STORAGE SECTION.
77 PA PIC 99 VALUE 10.
77 PB PIC 99 VALUE 5.
77 TOTAL PIC 9(4) VALUE 0.
PROCEDURE DIVISION.
DEBUT-P1PROG2.
      DISPLAY « TOTAL : » TOTAL.
      DISPLAY « PA : » PA.
      DISPLAY « PB : » PB.
      COMPUTE TOTAL = TOTAL + PA - PB.
      MOVE 1 TO PA PB.
      DISPLAY « TOTAL : » TOTAL.
      DISPLAY « PA : » PA.
      DISPLAY « PB : » PB.
END PROGRAM P1PROG2.
END PROGRAM PRINC1.
```

Résultats :

```
1ER CALL DE P1PROG1
TOTAL:0000
PA:10
PB:05
TOTAL:0005
PA:01
PB:01
2EME CALL DE P1PROG1
TOTAL:0005
PA:01
PB:01
TOTAL:0005
PA:01
PB:01
1ER CALL DE P1PROG2
TOTAL:0000
PA:10
PB:05
TOTAL:0005
PA:01
PB:01
2EME CALL DE P1PROG2
TOTAL:0000
PA:10
PB:05
TOTAL:0005
PA:01
PB:01
```

NOTES

3.7. EXEMPLE 5 : TRANSFERT PAR ADRESSE OU PAR CONTENU.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. PRINC1.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 P1-ZONA.
     02 P-A PIC 99 VALUE 10.
     02 P-B PIC 99 VALUE 05.
77 TOTAL PIC 9(4).
PROCEDURE DIVISION.
DEBUT-PRINC1.
     CALL « P1PROG1 » USING BY CONTENT P-A
                            P-B BY REFERENCE
     TOTAL.
     DISPLAY « TOTAL  : » TOTAL.
     DISPLAY « P-A : » P-A.
     DISPLAY « P-B : » P-B.
     STOP RUN.
END PROGRAM PRINC1
IDENTIFICATION DIVISION.
PROGRAM-ID. P1PROG1.
DATA DIVISION.
LINKAGE SECTION.
77 P1-A PIC 99.
77 P1-B PIC 99.
77 TOTAL PIC 9(4).
PROCEDURE DIVISION USING P1-A P1-B TOTAL.

DEBUT-P1PROG1.
     DISPLAY « DEBUT P1PROG1 ».
     ADD P1-A P1-B GIVING TOTAL.
     MOVE 0 TO P1-A P1-B.
     DISPLAY « TOTAL  : » TOTAL.
     DISPLAY « P-A  : » P-A.
     DISPLAY ‘’P-B : ’’ P-B.
     DISPLAY « SORTIE P1PROG1 ».

END PROGRAM P1PROG1.
```

Résultats

en 2° (au retour du « CALL »)

```
TOTAL : 0015
P-A : 10
P-B : 05
```

en 1° au « CALL » du P1PROG1 :

```
DEBUT P1PROG1


TOTAL : 0015
P-A : 00
P-B : 00
SORTIE
P1PROG1
```

NOTES

3.8. TERMINAISON D'UN SOUS-PROGRAMME : « END PROGRAM ».

La sortie d'un sous-programme interne est réalisée par :

END PROGRAM nom-de-sous-programme

Chaque programme, appelant ou appelé(s), possède son « délimiteur » propre .

4. LIBÉRATION MÉMOIRE ET RÉINITIALISATION : « CANCEL ».

L'objet de l'instruction « CANCEL » est éventuellement de libérer l'espace de mémoire centrale occupé par le sous-programme, mais surtout de le remettre dans l'état initial pour le prochain appel .

En effet, lors de l'exploitation d'un programme, on modifie toujours quelques données, par exemple un indice de table : avec « CANCEL » on repart toujours aux valeurs « VALUE » initiales

CANCEL « nom-de-sous-programme »

NOTES

5. EXERCICES D'APPLICATION.

5.1. PROGRAMMES À ÉCRIRE ET À TESTER.

Exercice 1	Affichage et observation
Le programme PRINC01 appelle deux programmes contenus PROC-01 et PROC02, il faut afficher le déroulement du programme principal et des programmes contenus :	
a) Le programme PROC01 n'a pas l'attribut COMMON et appelle le programme PROC02 Le programme PROC02 n'a pas l'attribut COMMON et est au même niveau que PROC01	
b) Le programme PROC01 n'a pas l'attribut COMMON et appelle le programme PROC02 Le programme PROC02 n'a pas l'attribut COMMON et est interne à PROC01	
c) Le programme PROC01 n'a pas l'attribut COMMON et appelle le programme PROC02 Le programme PROC02 a l'attribut COMMON et est au même niveau que PROC01	
d) Le programme PROC01 n'a pas l'attribut COMMON et appelle le programme PROC02 Le programme PROC02 a l'attribut COMMON et est interne à PROC01	
e) Le programme PROC01 a l'attribut COMMON et appelle le programme PROC02 Le programme PROC02 a l'attribut COMMON et est interne à PROC01	
f) Le programme PROC01 appelle deux programmes contenus PROC02 et PROC03 Le programme PROC02 a l'attribut COMMON et est au même niveau que PROC01 Le programme PROC03 n'a pas l'attribut COMMON, est interne à PROC01 et appelle le programme PROC02	

NOTES

Exercice 2	**Affichage et observation**
Le programme PRINC01 contient une donnée W-FIN avec l'attribut GLOBAL initialisée à 'FFFF' et appelle le programme contenu PROC01 qui affiche W-FIN	
a) Le programme PROC01 n'a pas déclarer W-FIN	
b) Le programme PROC01 contient la donnée W-FIN initialisée à 'NNNN'	
c) Le programme PROC01 contient la donnée W-FIN avec l'attribut GLOBAL initialisée à 'NNNN'	

Exercice 3	**Affichage et observation**
Le programme PRINC01 ne contient pas la donnée W-FIN, appelle le programme contenu PROC01 et affiche W-FIN	
Le programme PROC01 contient la donnée W-FIN avec l'attribut GLOBAL initialisée à 'NNNN' et affiche W-FIN	

Exercice 4	**Affichage et observation**
Le programme PRINC01 contient la donnée W-FIN avec l'attribut EXTERNAL et appelle le programme contenu PROC01 et affiche W-FIN	
Le programme PROC01 ne contient pas la donnée W-FIN et affiche W-FIN	

NOTES

Exercice 5	**Affichage et observation**
Le programme PRINC01 appelle deux fois le programme contenu PROC01	
a) PROC01 contient une donnée numérique W-CPTEUR de 5 octets initialisée à zéro, dans sa procédure une incrémentation de 5 est effectuée dans W-CPTEUR ainsi que son affichage	
b) PROC01 avec l'attribut INITIAL contient une donnée numérique W-CPTEUR de 5 octets initialisée à zéro, dans sa procédure une incrémentation de 5 est effectuée dans W-CPTEUR ainsi que son affichage	
c) PROC01 avec l'attribut INITIAL appelle le programme contenu PROC02 PROC02 contient une donnée numérique W-CPTEUR de 5 octets initialisée à zéro, dans sa procédure une incrémentation de 5 est effectuée dans W-CPTEUR ainsi que son affichage	
d) PROC01 avec l'attribut INITIAL appelle deux fois le programme contenu PROC02 PROC02 contient une donnée numérique W-CPTEUR de 5 octets initialise à zéro, dans sa procédure une incrémentation de 5 est effectuée dans W-CPTEUR ainsi que son affichage	

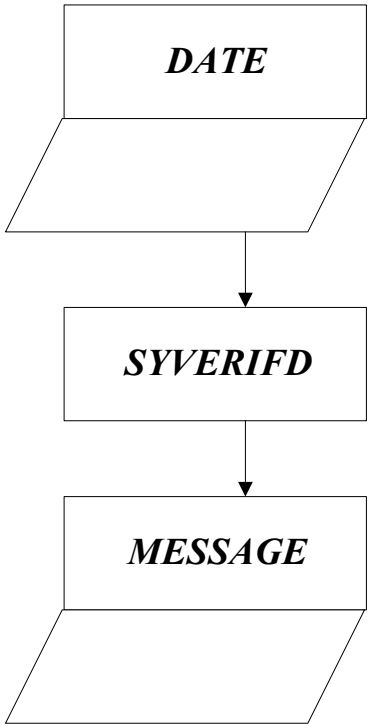
NOTES

TP1A

ÉCRITURE D'UN PROGRAMME DE VÉRIFICATION DE DATE ET AFFICHAGE ÉVENTUEL D'UN MESSAGE D'ERREUR DE DATE ERRONÉE

NOM DU PROGRAMME : ***SYVERIFD*** :

DIAGRAMME :



FICHIERS : sans objet pour ce programme.

TRAITEMENT :

- la date à saisir à l'écran sera de type « JJMMSSAA » (format français avec le siècle).
- en cas de date invalide, le message à afficher à l'écran sera composé du code erreur « 002 » accolé au libellé « DATE ERRONEE ».

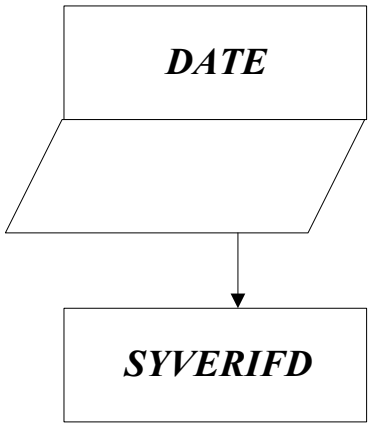
Remarque : le programme doit pouvoir vérifier autant de dates que l'utilisateur le désir jusqu'à l'obtention d'une date valide.

TP1A

ÉCRITURE D'UN PROGRAMME DE VÉRIFICATION DE DATE

NOM DU PROGRAMME : ***SYVERIFD*** :

DIAGRAMME :



FICHIERS : sans objet pour ce programme.

TRAITEMENT :

- la date à saisir à l'écran sera de type « JJMMSSAA » (format français avec le siècle).
- le sous-programme SYMSGERR sera appelé en cas de date invalide.
- le code erreur à passer au sous-programme SYMSGERR est « 002 » en cas de date erronée.

Remarque : le programme doit pouvoir vérifier autant de dates que l'utilisateur le désir jusqu'à l'obtention d'une date valide.

TP1A

ÉCRITURE D'UN SOUS-PROGRAMME DE CONTRÔLE DE VALIDITÉ DE DATE

NOM DU PROGRAMME : ***SYVRIF2D*** :

DIAGRAMME :

FICHIERS : sans objet pour ce programme.

TRAITEMENT :

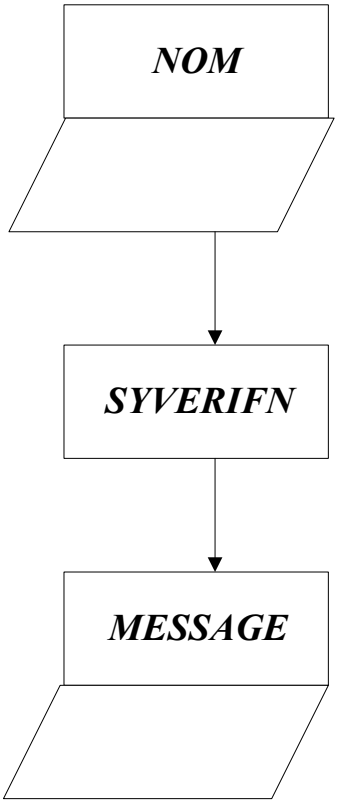
- la date à vérifier fournie par n'importe quel programme appelant sera de type « JJMMSSAA » (format français avec le siècle).
- le sous-programme SYMSGERR sera appelé en cas de date comportant jour, mois ou combinaison jour/mois impossible.
- le code erreur à passer au sous-programme SYMSGERR est « 002 » en cas de date erronée.
- ce programme doit pouvoir récupérer la date depuis n'importe quel programme appelant.

TP2A

ÉCRITURE D'UN PROGRAMME DE VÉRIFICATION DE VALIDITÉ DE NOM ET AFFICHAGE ÉVENTUEL D'UN MESSAGE D'ERREUR DE ZONE À BLANC

NOM DU PROGRAMME : ***SYVERIFN*** :

DIAGRAMME :



FICHIERS : sans objet pour ce programme.

TRAITEMENT :

- le nom (prénom) à saisir à l'écran sera de 100 caractères au plus.
- la longueur du nom saisi sera également à saisir pour procéder aux tests de vérification de validité et de longueur du nom.
- en cas de nom invalide le message à afficher à l'écran sera composé du code erreur « 003 » accolé au libellé « ZONE A BLANC ».

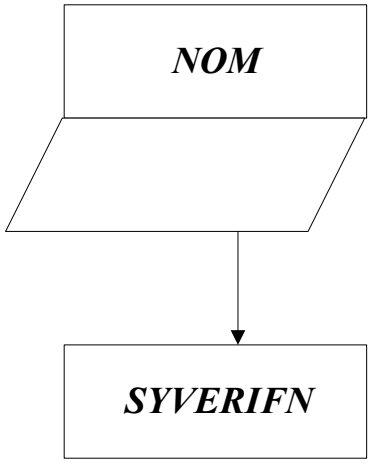
Remarque : le programme doit pouvoir vérifier autant de noms que l'utilisateur le désir jusqu'à l'obtention d'un nom valide.

TP2A

**ÉCRITURE D'UN PROGRAMME DE SAISIE
DE VÉRIFICATION DE VALIDITÉ DE NOM**

NOM DU PROGRAMME : ***SYVERIFN*** :

DIAGRAMME :



FICHIERS : sans objet pour ce programme.

TRAITEMENT :

- le nom (prénom) à saisir à l'écran sera de 100 caractères au plus.
- la longueur du nom saisi sera également à saisir pour procéder aux tests de vérification de validité.
- le sous-programme SYMSGERR sera appelé en cas de nom invalide.
- le code erreur à passer au sous-programme SYMSGERR est « 003 » en cas de nom invalide (zone à blanc).

Remarque : le programme doit pouvoir vérifier autant de noms que l'utilisateur le désir jusqu'à l'obtention d'un nom valide.

TP2A

ÉCRITURE D'UN SOUS-PROGRAMME DE VÉRIFICATION DE VALIDITÉ DE NOM

NOM DU PROGRAMME : ***SYVRIF2N*** :

DIAGRAMME :

FICHIERS : sans objet pour ce programme.

TRAITEMENT :

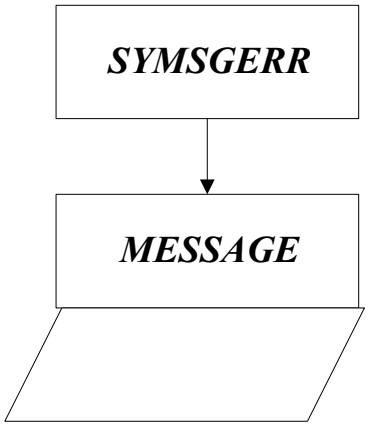
- le nom à vérifier fournie par n'importe quel programme appelant sera de 100 caractères au plus.
- la longueur du nom à vérifier sera également fournie par le programme appelant pour procéder aux tests de vérification de validité.
- le sous-programme SYMSGERR sera appelé en cas de nom invalide.
- le code erreur à passer au sous-programme SYMSGERR est « 003 » en cas de nom invalide (zone à blanc).
- ce programme doit pouvoir récupérer le nom et sa longueur depuis n'importe quel programme appelant.

TP3A

ÉCRITURE D'UN SOUS-PROGRAMME D'AFFICHAGE DE MESSAGE D'ERREUR

NOM DU PROGRAMME : ***SYMSGERR*** :

DIAGRAMME :



FICHIERS : sans objet pour ce programme.

TRAITEMENT :

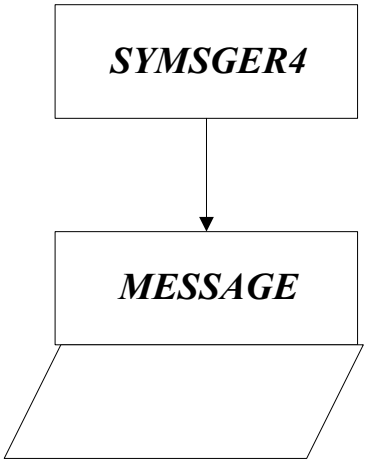
- prévoir la possibilité d'afficher à l'écran un message de 20 caractères alphanumériques, parmi 10 messages différents, précédé du code erreur associé, de valeur numérique 001 à 010 :
  - pour un code retourné de valeur autre que 001 à 010, il devra être affiché le message de code 001 ;
  - le code 001 sera associé au message de «ERREUR INCONNUE» ;
  - le code 002 «DATE ERRONEE» ;
  - le code 003 «ZONE A BLANC» ;
  - le code 004 «SIECLE ERRONEE» .
- ce programme doit pouvoir récupérer le code erreur depuis n'importe quel programme appelant :
  - il s'agit donc d'un programme modulaire (susceptible d'être à la disposition de tout utilisateur).

TP3A1

ÉCRITURE D'UN SOUS-PROGRAMME D'AFFICHAGE DE MESSAGE D'ERREUR AVEC TABLE DES MESSAGES ET CODE ERREUR

NOM DU PROGRAMME : ***SYMSGER4*** :

DIAGRAMME :



FICHIERS : sans objet pour ce programme.

TRAITEMENT :

- la table des messages et la zone de code erreur seront définis en « externe ».

- prévoir la possibilité d'afficher à l'écran un message de 20 caractères alphanumériques, parmi 10 messages différents, précédé du code erreur associé, de valeur numérique 001 à 010 :
  - pour un code retourné de valeur autre que 001 à 010, il devra être affiché le message de code 001 ;
  - le code 001 sera associé au message de «ERREUR INCONNUE» ;
  - le code 002 «DATE ERRONEE» ;
  - le code 003 «ZONE A BLANC» ;
  - le code 004 «SIECLE ERRONEE».

Ce programme doit pouvoir récupérer le code erreur depuis n'importe quel programme appelant.

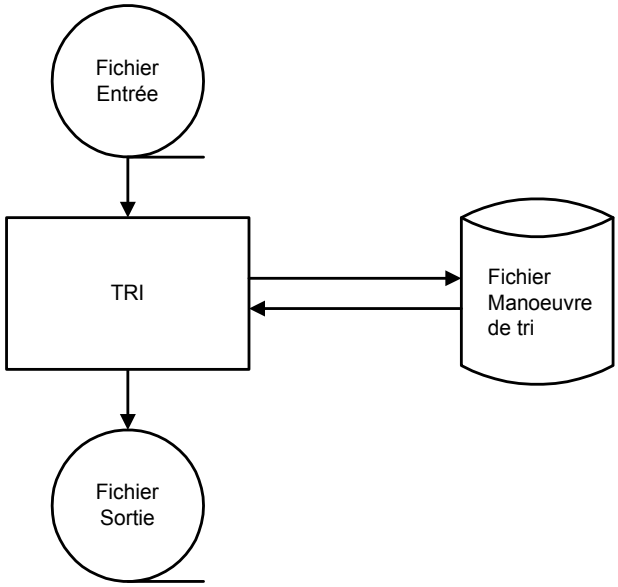
FICHE 02

Réorganiser articles et rubriques par le tri cobol

1. OBJECTIF.

L'utilitaire de tri COBOL permet, le plus souvent avec plus de souplesse et de possibilités que les outils « JCL » (ou « GCL ») des « gros » Systèmes, non seulement de trier les articles de(s) fichier(s) séquentiel(s) mais aussi de les sélectionner, les enrichir, manipuler leurs données, gérer les articles multiples, et ce sur un grand nombre de critères en se donnant une grande liberté de traitement, et en plus mettre le tout à l'abri d'erreurs de paramétrage en exploitation, en fixant les différentes règles par programme.

Diagramme :



- Cobol possède une routine permettant le tri de fichier(s).
  - ◊ l'appel de cette routine (SORT) peut se faire n'importe où en PROCEDURE DIVISION, sauf dans une section déclarative. Elle reçoit des enregistrements en provenance d'un ou de plusieurs fichiers;
  - ◊ elle va effectuer le tri selon des critères (arguments) spécifiés par le programmeur; elle permet optionnellement d'inclure des séquences de programme permettant la selection et la modification des enregistrements avant tri (**Input Procedure**), et/ou après tri (**Output Procedure**).
- L'emploi de la routine SORT nécessite une définition particulière des enregistrements à trier à travers la clause SD (**Sort Description**) dans la FILE SECTION. On parle de fichier de travail dédié au tri ou fichier de manoeuvre.

NOTES

2. PRINCIPES GENERAUX DU TRI.

2.1. TRI SANS MODIFICATION DES ENREGISTREMENTS EN ENTRÉE ET EN SORTIE.

SORT	**Nom-de-Fichier-Tri**	**USING Fichier(s)-en-entrée**
	(décrit en SD)	
		GIVING Fichier(s)-en-sortie

Remarque importante :

- Il n'est pas utile d'ouvrir, lire écrire ou fermer le(s) fichier(s) en entrée et en sortie car le tri gère l'ensemble des fichiers.

NOTES

2.2. TRI AVEC MODIFICATION DES ENREGISTREMENTS EN ENTRÉE.

SORT Nom-de-Fichier-Tri **INPUT PROCEDURE** Nom de paragraphe
(décrit en SD)
GIVING Fichier(s)-en-sortie

(8)Nom de paragraphe.

READ Fichier-entrée **AT END** Ordre-impératif

[**NOT AT END** Ordre-impératif]

END-READ

RELEASE Enregistrement du fichier-tri décrit en SD.

Remarques importantes :

- Il n'est pas utile d'ouvrir, écrire ou fermer le(s) fichier(s) en sortie car le tri les gère.
- S'il existe plusieurs fichiers en sortie, il faut les ouvrir, les écrire et les fermer.

NOTES

2.3. TRI AVEC MODIFICATION DES ENREGISTREMENTS EN SORTIE.

SORT	**Nom-de-Fichier-Tri**	**USING Fichier(s)-en-entrée**
	(décrit en SD)	**OUTPUT PROCEDURE nom de paragraphe**

(8)Nom de paragraphe.

RETURN Nom-de-Fichier-Tri AT END Ordre-impératif

[NOT AT END Ordre-impératif]

END-RETURN

WRITE Enregistrement du fichier-en sortie.

Remarques importantes :

- Il n'est pas utile d'ouvrir, lire ou fermer le(s) fichier(s) en entrée car le tri les gère.
- S'il existe plusieurs fichiers en sortie, il faut les ouvrir, les écrire et les fermer.

NOTES

2.4. TRI AVEC MODIFICATION DES ENREGISTREMENTS EN ENTRÉE ET EN SORTIE.

```
SORT    Nom-de-Fichier-Tri      INPUT PROCEDURE Nom de paragraphe-1
        (décrit en SD)
                                OUTPUT PROCEDURE Nom de paragraphe-2
```

```
(8)Nom de paragraphe-1.

READ Fichier-entrée             AT END Ordre-impératif

                                [ NOT AT END Ordre-          ]
                                  impératif
END-READ
RELEASE Enregistrement du fichier-tri décrit en SD.
(8)Nom de paragraphe-2.

RETURN Nom-de-Fichier-Tri       AT END Ordre-impératif

                                [ NOT AT END Ordre-          ]
                                  impératif
END-RETURN
WRITE Enregistrement du fichier-en sortie .
```

Remarques importantes :

- S'il existe plusieurs fichiers en entrée, il faut les ouvrir, les lire et les fermer.
- S'il existe plusieurs fichiers en sortie, il faut les ouvrir, les écrire et les fermer.

NOTES

3. LES INSTRUCTIONS CONCERNANT LE TRI.

3.1. SELECT.

3.1.1. FORMAT DE SELECT :

```
SELECT Nom-fichier-de-tri ASSIGN TO Nom-fichier-
système
```

3.1.2. EXEMPLE D'INPUT-OUTPUT SECTION.

Correspondant au diagramme décrit en « **1** » de la fiche « TRI » :

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT Nom-fichier-1-en-entrée ASSIGN TO Nom-fichier1-externe
    SELECT Nom-fichier-2-en-sortie ASSIGN TO Nom-fichier2-externe
    SELECT Nom-fichier-de-tri ASSIGN TO Nom-fichier-système
```

NOTES

3.2. S(ORT) D(ESCRIPTION).

3.2.1. FORMAT DE S(ORT) D(ESCRIPTION).

Remarque : une SD est identique à une FD.

```
SD  Nom-fichier de tri
    [DATA RECORD Nom-enreg1 [Nom-Enreg2]…]
    Description de(s) enregistrement(s) de tri.
01  Nom-Enreg1.
    Etc……
```

3.2.2. EXEMPLE DE FILE SECTION.

Correspondant au diagramme décrit en « 1 » de la fiche « TRI » :

```
FD  Nom-fichier1 en entrée.
    [DATA RECORD Nom-enreg1 [Nom-Enreg2]…]

FD  Nom-fichier2 en sortie.
    [DATA RECORD Nom-enreg1 [Nom-Enreg2]…]

SD  Nom-fichier3 de tri.
    [DATA RECORD Nom-enreg1 [Nom-Enreg2]…]
    Description de(s) enregistrement(s) de tri.
01  Nom-Enreg1.
    Etc……
```

NOTES

3.3. SORT.

3.3.1. FORMAT DE SORT :

SORT	**Nom-fichier de tri** **ON [DESCENDING]** **[ASCENDING] KEY [Argument de tri 1]…** **{ON [DESCENDING]** **[ASCENDING] KEY [Argument de tri n]…}** **WITH DUPLICATES IN ORDER**
	INPUT PROCEDURE Nom-Parag1[THRU Nom-Parag1n] ***OU*** **USING Nom-Fichier-n en entrée**
	OUTPUT PROCEDURE Nom-Parag2[THRU Nom-Parag2n] ***OU*** **GIVING Nom-Fichier-n en sortie**

NOTES

3.3.2. EXEMPLE DE PROCEDURE DIVISION.

Correspondant au diagramme décrit en « 1 » de la fiche « TRI » :

```
PROCEDURE DIVISION.
    SORT   Nom-fichier3 de tri
           ON ASCENDING KEY Argument de tri 1
           ON DESCENDING KEY Argument de tri2
           INPUT PROCEDURE Nom-Parag1
           OUTPUT PROCEDURE Nom-Parag2.
Nom-Parag1.
    READ Nom-Fichier1 AT END Ordre Impératif
                     NOT AT END Ordre Impératif
    END-READ
    RELEASE Nom-Enreg1-Fichier3 [FROM Nom-Enreg1-Fichier1]
Nom-Parag2.
    RETURN Nom-Fichier3 [INTO Nom-Enreg1-Fic2]
                              AT END Ordre Impératif
                     NOT AT END Ordre Impératif
    END-RETURN
    WRITE Nom-Enreg1-Fichier2 [FROM Nom-Enreg1-Fichier3]
```

NOTES

3.3.3. REMARQUES IMPORTANTES :

- Au maximum, il est possible de constituer 12 arguments de tri, la totalité ne dépassant pas 255 caractères.
- Le nombre d'ordre de lecture est équivalent à celui du nombre de fichier(s) en entrée.
- Le nombre d'ordre d'écriture est équivalent à celui du nombre de fichier(s) en sortie.
- **ASCENDING KEY** et **DESCENDING KEY** spécifient des arguments de tri, se réfèrent aux critères de tri spécifiés dans la description faîte en SD. On peut spécifier plusieurs critères à la suite (majeur, médian, mineur).
- **USING** spécifie le nom d'un fichier « entrée » fourni intégralement au tri sans intervention du programmeur (donc pas d'INPUT PROCEDURE). Le tri ira chercher les enregistrements à trier. Même l'ouverture et la fermeture sont pris en compte (pas d'OPEN, ni de CLOSE et encore moins de READ à réaliser).
- **INPUT PROCEDURE** spécifie le nom d'une section indépendante (ensemble de paragraphe) ou d'un paragraphe écrit par le programmeur, et qui va prendre en charge la gestion des enregistrements à trier pour effectuer éventuellement les sélections et/ou les modifications. Les enregistrements à trier devront être fournis au module de tri par l'ordre RELEASE (identique à un WRITE séquentiel).
- **GIVING** spécifie le nom d'un fichier en sortie de tri. Ce fichier sera géré intégralement par le module de tri qui y rangera ses enregistrements triés sans intervention du programmeur (donc pas d'OUTPUT PROCEDURE). Le tri ira y ranger les enregistrements triés. L'ouverture et la fermeture sont pris en compte (pas d'OPEN, ni de CLOSE et encore moins de WRITE à réaliser).
- **OUTPUT PROCEDURE** spécifie le nom d'une section indépendante (ensemble de paragraphe) ou d'un paragraphe écrit par le programmeur, qui va prendre en charge la gestion des enregistrements triés (sortie de tri) pour effectuer éventuellement les sélections et/ou les modifications, avant par exemple une écriture sur un fichier ou un stockage en table. Les enregistrements triés devront être extraits du module de tri par l'ordre RETURN (identique à un READ séquentiel).
- **WITH DUPLICATES** : si cet ordre est spécifié, et que plusieurs éléments ont des arguments de tri égaux, alors ces éléments restent dans l'ordre dans lequel ils ont été trouvés dans le fichier en entrée, et si une INPUT PROCEDURE a été spécifiée, ils restent dans l'ordre dans lequel cette procédure les a fournis au tri.

NOTES

3.4. RELEASE.

Format de RELEASE :

RELEASE Nom-Enreg-Fichier-de-tri [FROM Nom-Enreg-Fichier-en-entrée]

Nom-Enreg-Fichier-de-tri se réfère à un enregistrement de SD.

Nom-Enreg-Fichier-en-entrée se réfère à un enregistrement de FD.

3.5. RETURN.

Format de RETURN :

RETURN Nom-Enreg-Fichier-de-tri [INTO Nom-Enreg-Fichier-en-sortie]
AT END Ordre Impératif
NOT AT END Ordre Impératif
END-RETURN

Nom-Enreg-Fichier-de-tri se réfère à un enregistrement de SD.

Nom-Enreg-Fichier-en-sortie se réfère à un enregistrement de FD.

Le « RETURN » « lit » des enregistrements (triés) provenant du module de tri. Quand tous les enregistrements ont été lus, on est « AT END » comme lors d'une lecture séquentielle d'un fichier séquentiel « normal ».

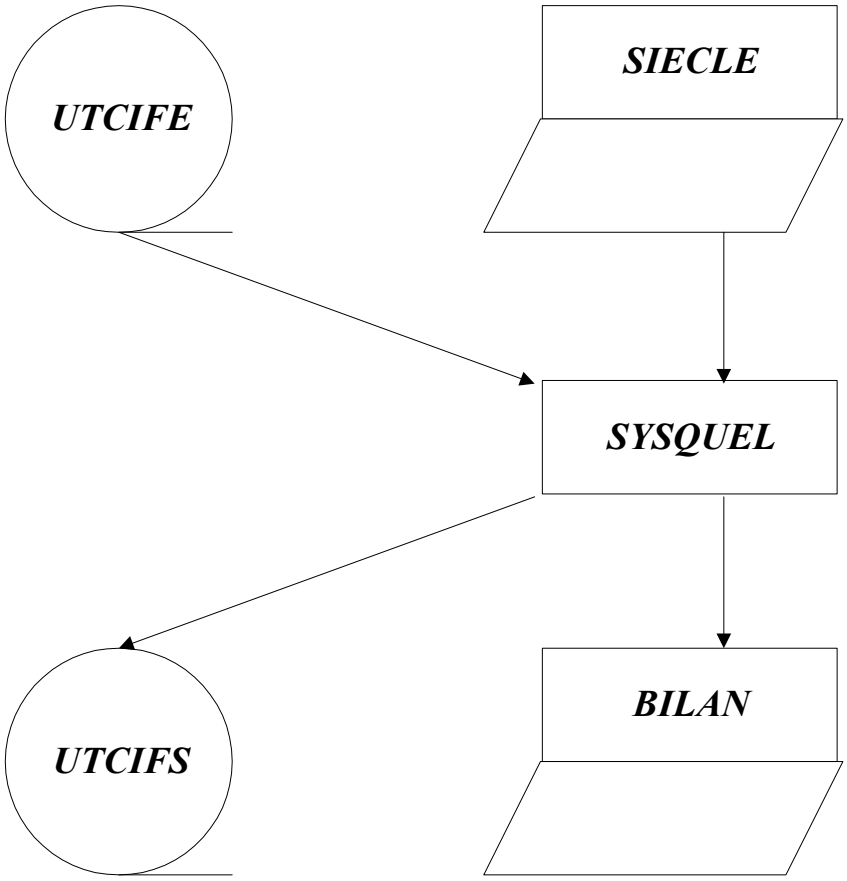
NOTES

TP4A

écriture d'un programme de vérification de fichier contrôle de validité du siècle et traitement d'un fichier de prénoms regroupés par date de fête avec appels au sous-programme de vérification de date appels au sous-programme de validité de nom appels éventuels au sous programme d'affichage de message d'erreur pour le siècle invalide, les dates ou les prénoms invalides, et comptage des articles lus et des articles invalides éliminés

NOM DU PROGRAMME : ***SYSQUEL*** :

DIAGRAMME :



FICHIERS : un fichier en entrée, objet du traitement, un fichier en sortie non utilisé dans ce module bien que présent pour utilisation dans un module ultérieur (sa prise en compte est cependant obligatoire pour l'homogénéité et la simplification de l'utilisation des JCL qui vous sont fournis) :

Fichier en entrée : ifn UTC1FE, efn UTC1FE, fd UTC1FE			
nom d'article	ART-ENT		
zone de codification d'article	CODART		1 caractère numérique
zone date de fête	DATENT	4	caractères numériques
zone inutilisée		3	caractères alphanum.
zone des prénoms et sexes	ZONE	6	postes
zone d'un prénom-sexe	ZPRENOM		indexé par I
zone d'un prénom	PRENOM	11	caractères alphanum.
zone du sexe	SEXE	1	caractère alphanum.

Fichier en sortie : ifn UTC1FS, efn UTC1FS, fd UTC1FS			
nom d'article	ART-SORTIE		
zone d'un prénom	LIBPRENOM	11	caractères alphanum.
zone nombre de dates de fête	ZONENB	2	caractères numériques
zone des dates de fête	SDATE	5	postes
zone d'une date de fête	DATESORT		
zone du mois	MMSORT	2	caractères numériques
zone du jour	JJSORT	2	caractères numériques
zone inutilisée		7	caractères alphanum.

TRAITEMENT :

- le siècle à saisir à l'écran sera de type « SS » sur 2 caractères numériques.
- le sous-programme SYMSGERR sera appelé en cas de siècle non numérique.
- le code erreur à passer au sous-programme SYMSGERR est « 004 » en cas de siècle erronée.
- chaque article du fichier entrée sera lu et compté dans le compteur CPTLU ; la date contenue sera vérifiée : le sous-programme SYVRIF2D sera appelé, et en cas d'invalidité (l'article éliminé), (le sous-programme SYMSGERR sera appelé).
- chaque prénom sera traité ; la zone des prénoms sera seulement vérifiée dans ce module : en cas d'invalidité le sous-programme SYVRIF2N sera appelé, et en cas d'invalidité (l'article éliminé), (le sous-programme SYMSGERR sera appelé).
- tout article éliminé sera compté dans le compteur CPTELIM et les zones DATENT et ZONE affichés à l'écran précédé du libellé « ART-ENT ».

En fin de traitement le message suivant sera affiché à l'écran :

```
''      «                          «
        «      BILAN DE PASSAGE    «
        «      _______________     «
        «                          «
        «      PROG. : SYSQUEL     «
        «                          «
        « ARTICLES LUS SUR UTC1FE  «     CPTLU
        « ARTICLES ELIMINES        «     CPTELIM
        « ARTICLE CREES SUR UTC1FS «     CPTECRIT      ''
```

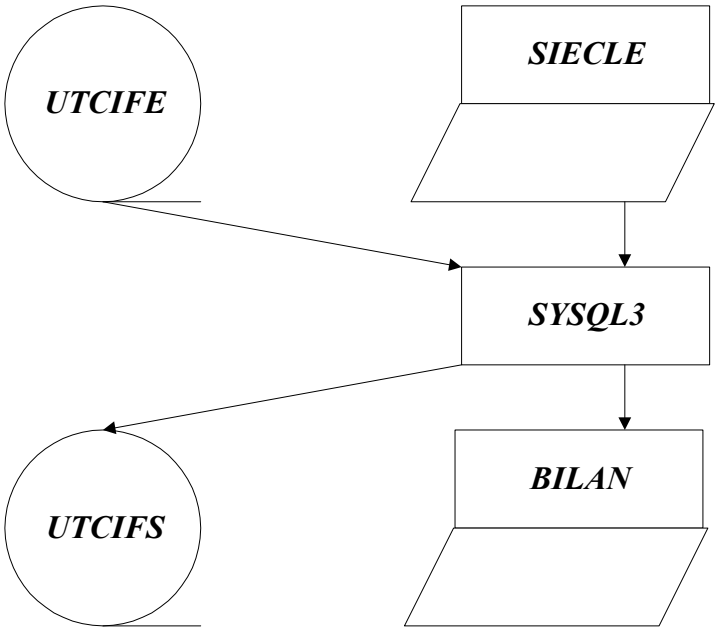
Remarque : le CPTECRIT inutilisé dans ce module sera donc à zéro.

TP4A1

écriture d'un programme de vérification de fichier
contrôle de validité du siècle et traitement d'un fichier de prénoms regroupés par date de fête avec appels au sous-programme de vérification de date (jour/mois) appels au sous-programme de validité de nom appels éventuels au sous programme d'affichage de message d'erreur pour le siècle, les dates ou les prénoms, invalides et comptage des articles lus et des articles invalides éliminés. Intégration des sous-programmes appelés au programme principal

NOM DU PROGRAMME : ***SYSQL3*** :

DIAGRAMME :



FICHIERS : un fichier en entrée, objet du traitement, un fichier en sortie non utilisé dans ce module bien que présent pour utilisation dans un module ultérieur (sa prise en compte est cependant obligatoire pour l'homogénéité et la simplification de l'utilisation des JCL qui vous sont fournis) :

Fichier en entrée : ifn UTC1FE, efn UTC1FE, fd UTC1FE			
nom d'article	ART-ENT		
zone de codification d'article	CODART	1	caractère numérique
zone date de fête	DATENT	4	caractères numériques
zone inutilisée		3	caractères alphanum.
zone des prénoms et sexes	ZONE	6	postes
zone d'un prénom-sexe	ZPRENOM	indexé par I	
zone d'un prénom	PRENOM	11	caractères alphanum.
zone du sexe	SEXE	1	caractère alphanum.

Fichier en sortie : ifn UTC1FS, efn UTC1FS, fd UTC1FS			
nom d'article	ART-SORTIE		
zone d'un prénom	LIBPRENOM	11	caractères alphanum.
zone nombre de dates de fête	ZONENB	2	caractères numériques
zone des dates de fête	SDATE	5	postes
zone d'une date de fête	DATESORT		
zone du mois	MMSORT	2	caractères numériques
zone du jour	JJSORT	2	caractères numériques
zone inutilisée		7	caractères alphanum.

TRAITEMENT :

Remarque : l'intégration de certains sous-programmes devra permettre la définition globale de certaines zones.

- le siècle à saisir à l'écran sera de type « SS » sur 2 caractères numériques.
- le sous-programme SYMSGERR sera intégré au programme principal sous le nom SYMSGER3.
- le sous-programme SYMSGER3 sera appelé en cas de siècle non numérique .
- le code erreur à passer au sous-programme SYMSGER3 est « 004 » en cas de siècle erronée.

Chaque article du fichier entrée sera lu et compté dans le compteur CPTLU ; la date contenue sera vérifiée :

- le sous-programme SYVRIF2D sera intégré au programme principal sous le nom SYVRIF3D où sera fait appel au sous-programme SYMSGER3 ;
- en cas d'invalidité de la date le sous-programme SYVRIF3D sera appelé et l'article éliminé.

Chaque prénom sera traité ; la zone des prénoms sera seulement vérifiée dans ce module :

- le sous-programme SYVRIF2N sera intégré au programme principal sous le nom SYVRIF3N où sera fait appel au sous-programme SYMSGER3 ;
- en cas d'invalidité du prénom le sous-programme SYVRIF3N sera appelé et l'article éliminé.

Tout article éliminé sera compté dans le compteur CPTELIM et les zones DATENT et ZONE affichés à l'écran précédé du libellé « ART-ENT ».

En fin de traitement le message suivant sera affiché à l'écran :

```
''      «                              «
        «          BILAN DE PASSAGE    «
        «          _______________     «
        «                              «
        «          PROG. : SYSQL3      «
        «                              «
        « ARTICLES LUS SUR UTC1FE :    «     CPTLU
        « ARTICLES ELIMINES            :     «     CPTELIM
        « ARTICLE CREES SUR UTC1FS     :     «     CPTECRIT      ''
```

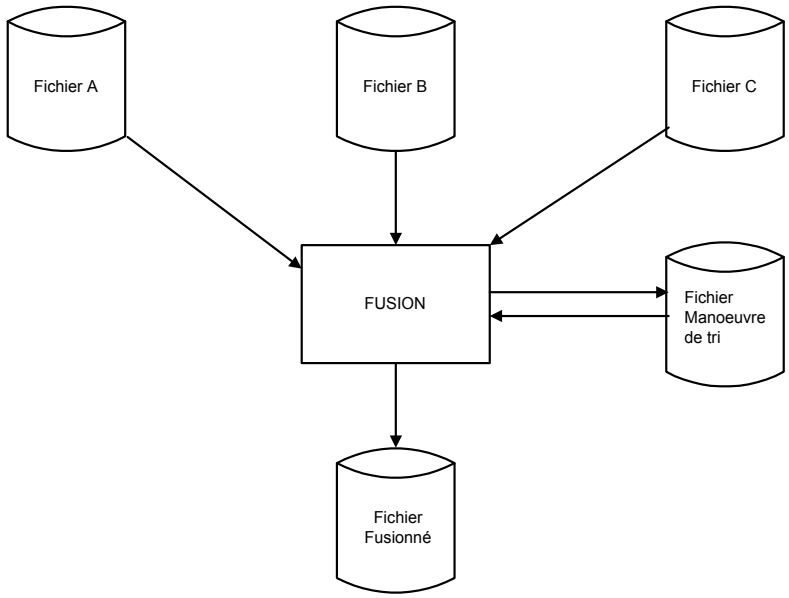
Remarque : le CPTECRIT inutilisé dans ce module sera donc à zéro.

Fusionner des fichiers triés par la fusion cobol

1. OBJECTIF.

L'utilitaire de fusion COBOL permet, le plus souvent de façon plus économique que les outils « JCL » (ou « GCL ») des « gros » Systèmes, non seulement de fusionner les articles de fichiers séquentiels déjà triés mais aussi de les sélectionner, les enrichir, manipuler leurs données, gérer les articles multiples, et ce sur un grand nombre de critères en se donnant une grande liberté de traitement, et en plus mettre le tout à l'abri d'erreurs de paramétrage d'exploitation, en fixant les différentes règles par programme.

Diagramme :



- Cobol possède une routine permettant la fusion de fichiers triés sur les mêmes critères :
  - L'appel de cette routine (MERGE) peut se faire n'importe où en PROCEDURE DIVISION, sauf dans une section déclarative. Elle reçoit des enregistrements en provenance d'au moins deux fichiers; elle va effectuer leur interclassement selon des critères (arguments) spécifiés par le programmeur; elle permet optionnellement d'inclure des séquences de programme permettant la sélection et la modification des enregistrements seulement après fusion (**Output Procedure**).
  - L'emploi de la routine MERGE nécessite une définition particulière des enregistrements à fusionner à travers la clause SD (**Sort Description**) dans la FILE SECTION. On parle de fichier de travail dédié à la fusion ou fichier de manoeuvre.

NOTES

2. PRINCIPES GENERAUX DE LA FUSION.

2.1. FUSION SANS MODIFICATION DES ENREGISTREMENTS EN SORTIE.

MERGE	**Nom-de-Fichier-Fusion**	**USING Fichiers-en-entrée**
	(décrit en SD)	
		GIVING Fichier(s)-en-sortie

Remarque importante :

- Il n'est pas utile d'ouvrir, lire écrire ou fermer les fichiers en entrée et le(s) fichier(s) en sortie car le tri gère l'ensemble des fichiers.

NOTES

2.2. FUSION AVEC MODIFICATION DES ENREGISTREMENTS EN SORTIE.

MERGE	**Nom-de-Fichier-Fusion**	**USING Fichiers-en-entrée**
	(décrit en SD)	
		OUTPUT PROCEDURE
		Nom de paragraphe

(8)Nom de paragraphe.

RETURN Nom-de-Fichier-fusion **AT END Ordre-impératif**

[NOT AT END Ordre-impératif]

END-RETURN

WRITE Enregistrement du fichier-en sortie.

Remarques importantes :

- Il n'est pas utile d'ouvrir, lire ou fermer les fichiers en entrée car la fusion les gère.
- S'il existe plusieurs fichiers en sortie, il faut les ouvrir, les écrire et les fermer.

NOTES

3. LES INSTRUCTIONS CONCERNANT LA FUSION.

3.1. SELECT.

3.1.1. FORMAT DE SELECT :

```
SELECT Nom-fichier-de-Fusion ASSIGN TO Nom-fichier-système
```

3.1.2. EXEMPLE D'INPUT-OUTPUT SECTION.

Correspondant au diagramme décrit en « **1** » de la fiche « FUSION » :

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT Nom-fichier-A-en-entrée ASSIGN TO Nom-fichier1-externe
    SELECT Nom-fichier-B-en-entrée ASSIGN TO Nom-fichier2-externe
    SELECT Nom-fichier-C-en-entrée ASSIGN TO Nom-fichier3-externe
    SELECT Nom-fichfusionné-sortie ASSIGN TO Nom-fichier4-externe
    SELECT Nom-fichier-de-Fusion ASSIGN TO Nom-fichier-système
```

3.2. S(ORT) D(ESCRIPTION).

3.2.1. FORMAT DE S(ORT) D(ESCRIPTION) :

remarque : une SD est identique à une FD.

```
SD   Nom-fichier de Fusion
     [DATA RECORD Nom-enreg1 [Nom-Enreg2]…]
     Description de(s) enregistrement(s) de fusion.
01   Nom-Enreg1.
     Etc……
```

NOTES

3.2.2. EXEMPLE DE FILE SECTION.

Correspondant au diagramme décrit en « 1 » de la fiche « FUSION » :

```
FD   Nom-fichier-A en entrée.
     [DATA RECORD Nom-enreg1 [Nom-Enreg2]…]

FD   Nom-fichier-B en entrée.
     [DATA RECORD Nom-enreg3 [Nom-Enreg4]…]

FD   Nom-fichier-C en entrée.
     [DATA RECORD Nom-enreg5 [Nom-Enreg6…]

FD   Nom-fichier-fusionné en sortie.
     [DATA RECORD Nom-enreg7 [Nom-Enreg8]…]

SD   Nom-fichier-de-fusion.
     [DATA RECORD Nom-enreg9 [Nom-Enreg10]…]
     Description de(s) enregistrement(s) de tri.
01   Nom-Enreg9.
     Etc……
```

NOTES

3.3. MERGE.

3.3.1. FORMAT DE MERGE :

MERG E **Nom-fichier de fusion**

ON [DESCENDING]
[ASCENDING] KEY [Argument de fusion 1]…
{ON [DESCENDING]
[ASCENDING] KEY [Argument de fusion n]…}
COLLATING SEQUENCE IS Nom-Alphabétique
USING Nom-Fichier-1-en entrée,
Nom-Fichier-n-en entrée *(minimum 2***}**

OUTPUT PROCEDURE Nom-Parag2 [THRU Nom-Parag2n]
OU
GIVING Nom-Fichier-n en sortie.

3.3.2. EXEMPLE DE PROCEDURE DIVISION.

Correspondant au diagramme décrit en « 1 » de la fiche « FUSION » :

```
PROCEDURE DIVISION.
   MERGE Nom-fichier de fusion
                ON ASCENDING KEY Argument de fusion1
                ON DESCENDING KEY Argument de fusion2
                USING Nom-fichier-A Nom-fichier-B Nom-fichier-C
                OUTPUT PROCEDURE Nom-Parag1
Nom-Parag1.
   RETURN Nom-Fichier de fusion [INTO Nom-Enreg7-Fichier-
   fusionné-en-sortie]      AT END Ordre Impératif
                NOT AT END Ordre Impératif
   END-RETURN
   WRITE Nom-Enreg7-Fichier-fusionné-en-sortie
                [FROM Nom-Enreg9-Fichier-de-fusion]
```

NOTES

3.3.3. REMARQUES IMPORTANTES :

- Au maximum, il est possible de constituer 12 arguments de fusion, la totalité ne dépassant pas 255 caractères.
- Le nombre d'ordre d'écriture est équivalent à celui du nombre de fichier(s) en sortie.
- **ASCENDING KEY** et **DESCENDING KEY** spécifient des arguments de fusion, se réfèrent aux critères de fusion spécifiés dans la description faîte en SD. On peut spécifier plusieurs critères à la suite (majeur, médian, mineur).
- **USING** spécifie le nom des fichiers « entrée » fourni intégralement à la fusion sans intervention du programmeur (donc pas d'INPUT PROCEDURE). La fusion ira chercher les enregistrements à fusionner. Même l'ouverture et la fermeture sont pris en compte (pas d'OPEN, ni de CLOSE et encore moins de READ à réaliser).
- **GIVING** spécifie le nom d'un fichier en sortie de fusion. Ce fichier sera géré intégralement par le module de fusion qui y rangera ses enregistrements triés et interclassés sans intervention du programmeur (donc pas d'OUTPUT PROCEDURE). La fusion ira y ranger les enregistrements triés et interclassés. L'ouverture et la fermeture sont pris en compte (pas d'OPEN, ni de CLOSE et encore moins de WRITE à réaliser).
- **OUTPUT PROCEDURE** spécifie le nom d'une section indépendante (ensemble de paragraphe) ou d'un paragraphe écrit par le programmeur, qui va prendre en charge la gestion des enregistrements triés et interclassés (sortie de fusion) pour effectuer éventuellement les sélections et/ou les modifications, avant par exemple une écriture sur un fichier ou un stockage en table. Les enregistrements triés et interclassés devront être extraits du module de fusion par l'ordre RETURN (identique à un READ séquentiel).
- **COLLATING SEQUENCE** : à tester pendant les TP.

NOTES

3.4. RETURN.

Format de RETURN :

```
RETURN Nom-Enreg-Fichier-de-FUSION
            [INTO Nom-Enreg-Fichier-fusionné-en-sortie]
             AT END Ordre Impératif
            NOT AT END Ordre Impératif
END-RETURN
```

Nom-Enreg-Fichier-de-fusion se réfère à un enregistrement de SD.

Nom-Enreg-Fichier-fusionné-en-sortie se réfère à un enregistrement de FD (il peut également désigner une zone quelconque définie en mémoire centrale ayant une description ad-hoc à la réception des données de la fusion).

Le « RETURN » « lit » des enregistrements (triés et interclassés) provenant du module de fusion. Quand tous les enregistrements ont été lus, on est « AT END » comme lors d'une lecture séquentiel d'un fichier séquentiel « normal ».

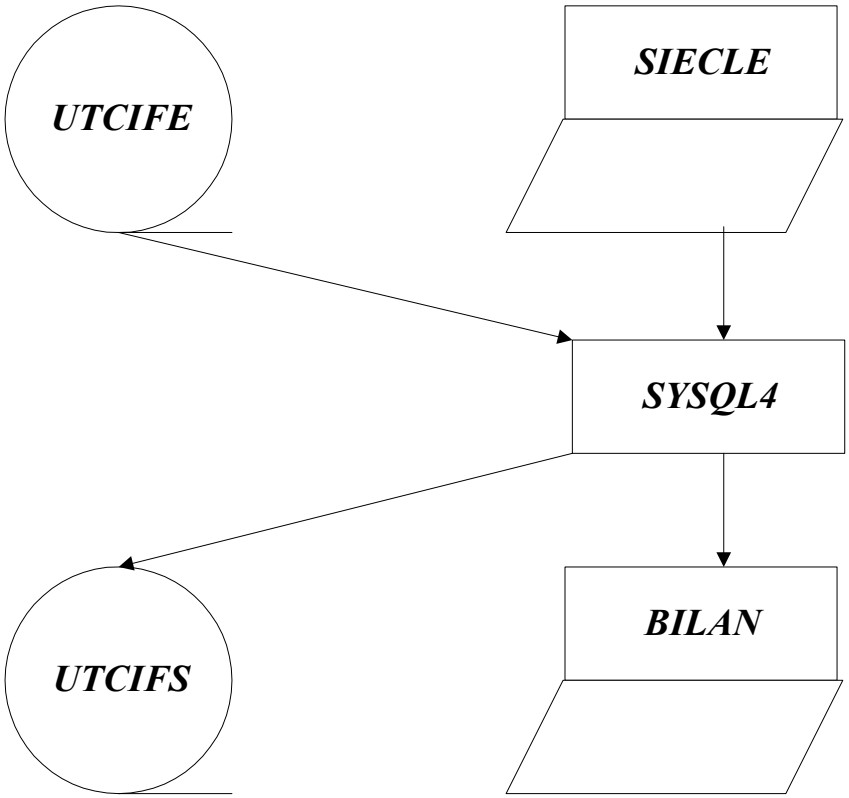
NOTES

TP4A2

écriture d'un programme de vérification de fichier
contrôle de validité du siècle et traitement d'un fichier de prénoms regroupés par date de fête avec appels au sous-programme de vérification de date appels au sous-programme de validité de nom appels éventuels au sous programme d'affichage de message d'erreur pour le siècle, les dates ou les prénoms, invalides et comptage des articles lus et des articles invalides éliminés .

NOM DU PROGRAMME : ***SYSQL4*** :

DIAGRAMME :



FICHIERS : un fichier en entrée, objet du traitement, un fichier en sortie non utilisé dans ce module bien que présent pour utilisation dans un module ultérieur (sa prise en compte est cependant obligatoire pour l'homogénéité et la simplification de l'utilisation des JCL qui vous sont fournis) :

Fichier en entrée : ifn UTC1FE, efn UTC1FE, fd UTC1FE			
nom d'article	ART-ENT		
zone de codification d'article	CODART	1	caractère numérique
zone date de fête	DATENT	4	caractères numériques
zone inutilisée		3	caractères alphanum.
zone des prénoms et sexes	ZONE	6	postes
zone d'un prénom-sexe	ZPRENOM	indexé par I	
zone d'un prénom	PRENOM	11	caractères alphanum.
zone du sexe	SEXE	1	caractère alphanum.

Fichier en sortie : ifn UTC1FS, efn UTC1FS, fd UTC1FS			
nom d'article	ART-SORTIE		
zone d'un prénom	LIBPRENOM	11	caractères alphanum.
zone nombre de dates de fête	ZONENB	2	caractères numériques
zone des dates de fête	SDATE	5	postes
zone d'une date de fête	DATESORT		
zone du mois	MMSORT	2	caractères numériques
zone du jour	JJSORT	2	caractères numériques
zone inutilisée		7	caractères alphanum.

TRAITEMENT :

Remarque : l'intégration de certains sous-programmes devra permettre la définition globale de certaines zones

- le siècle à saisir à l'écran sera de type « SS » sur 2 caractères numériques.
- le sous-programme SYMSGER4 sera appelé en cas de siècle non numérique.
- le code erreur à passer au sous-programme SYMSGER4 est « 004 » en cas de siècle erronée.

Chaque article du fichier entrée sera lu et compté dans le compteur CPTLU ; la date contenue sera vérifiée :

- le sous-programme SYVRIF2D sera intégré au programme principal sous le nom SYVRIF3D où sera fait appel au sous-programme SYMSGER4 ;
    - ◊ en cas d'invalidité de la date le sous-programme SYVRIF3D sera appelé et l'article éliminé.

Chaque prénom sera traité ; la zone des prénoms sera seulement vérifiée dans ce module :

- le sous-programme SYVRIF2N sera intégré au programme principal sous le nom SYVRIF3N où sera fait appel au sous-programme SYMSGER4 ;
    - ◊ en cas d'invalidité du prénom le sous-programme SYVRIF3N sera appelé et l'article éliminé.

Tout article éliminé sera compté dans le compteur CPTELIM et les zones DATENT et ZONE affichés à l'écran précédé du libellé « ART-ENT ».

En fin de traitement le message suivant sera affiché à l'écran :

```
‘’      «                                       «
        « BILAN DE PASSAGE                      «
        « ________________                      «
        «                                       «
        « PROG. : SYSQL4                        «
        «                                       «
        « ARTICLES LUS SUR UTC1FE :     «       CPTLU
        « ARTICLES ELIMINES             :       «       CPTELIM
        « ARTICLE CREES SUR UTC1FS      :       «       CPTECRIT        ‘’
```

Remarque : le CPTECRIT inutilisé dans ce module sera donc à zéro.

1. CODE RETOUR.

- **ABSTACK** (Valeur de clé d'état COBOL 30)
  - ◊ Opération quelconque : Arrêt prématuré produite par appel de programme (et non par une anomalie)
- **CALLVIOL**
  - ◊ SORT : Une tentative d'exécuter SORT deux fois au cours d'une même activité produit une anomalie; l'activité est suspendue.
  - ◊ RELEASE : L'accès à la procédure d'entrée au moyen de RELEASE est refusée.
  - ◊ RETURN : L'accès à la procédure de sortie au moyen de RETURN est refusée.
- **CMWSOV** (Valeur de clé d'état COBOL 30)
  - ◊ SORT : Fichier non ouvert par manque de place pour les tampons. Réduire le nombre de blocs par tampon BPB (dans DEFINE) ou augmenter la taille (SIZE) de la mémoire de contrôle.
- **DATAGAIN**
  - ◊ Lecture des articles triés (RETURN) : La fin des données n'a pas encore été atteinte, mais le nombre des articles triés et renvoyés est déjà égal au nombre des articles envoyés au tri. Les articles suivant seront lus si c'est spécifié dans le programme.
- **DATALOSS**
  - ◊ Lecture des articles triés (RETURN) : La fin des données est atteinte, mais certaines données ont été perdues (pendant le processus de tri), car le nombre d'articles triés est inférieur au nombre d'articles envoyés pour le tri. La lecture d'un article suivant entraînera le code retour **DATALIM.**

NOTES

- **EXHAUST** (Valeur de clé d'état COBOL 30)
  - ◊ Lecture des articles triés (RETURN) : Fin de fichier déjà détectée. Ceci implique qu'il y a moins d'articles triés que prévu. L'activité est suspendue (L'instruction RETURN a été exécutée quand la fin de fichier a été détectée).
- **FILENASG**
  - ◊ Déclenchement du processus SORT : Aucun fichier de travail n'a été affecté; ou, pour un tri sur bande, le nombre de fichiers de travail affecté est inférieur à trois.
- **FILEWSOV**
  - ◊ Transfert d'articles au fichier de tri (RELEASE) : Taille limite du fichier de travail atteinte. Article courant non traité. Créer un fichier de travail plus grand.
  - ◊ Achèvement du processus de tri : Taille limite du fichier de travail atteinte. Les autres articles ne seront pas traités. Créer un fichier de travail plus grand.
- **IOFAIL** (Valeur de clé d'état COBOL 30)
  - ◊ Lancement du processus SORT: Incident d'E/S logiciel. les fichiers de travail n'ont pas été ouvert correctement.
- **LNERR** (Valeur de clé d'état COBOL 92)
  - ◊ Transfert d'articles dans le fichier de tri (RELEASE) : La longueur de l'article de longueur variable dépasse la longueur maximale spécifiée.
- **SORTERR**
  - ◊ Déclenchement ou achèvement d'un process SORT : Une erreur de traitement système s'est produite pendant le tri. L'activité est arrêtée prématurément.
  - ◊ Transfert de l'article dans le fichier de tri (RELEASE) : Une erreur de traitement de tri s'est produite. Si l'erreur n'est pas imputable à l'utilisateur, il faut prévenir le centre de service constructeur. L'activité est arrêtée prématurément.

NOTES

- **SPACENAV**
  - ◊ SORT : Place insuffisante pour le fichier de travail temporaire. Créer un fichier plus grand.
- **EXCEPTION 06-02:UNAVAIBLE (SEGMENT 0.0) IN TASK MAIN.0 AT ADRESS 8.XX.XXXX**
  - ◊ Le verbe SORT n'a pas été écrit dans le programme. Il existe juste une instruction COBOL du SORT (Ex : RELEASE, RETURN…).
- **UNEXPECTEDE RETURN CODE (RC=F0001889---> USER 0,SEQERR) GOT IN TASK MAIN.0 AT ADRESS 8.XX.XXXX**
  - ◊ L'instruction STOP RUN a été oubliée.

2. CONSIGNES PARTICULIERES.

INPUT PROCEDURE ou OUTPUT PROCEDURE

- Dans ces procédures, un seul paragraphe est possible. Il est interdit d'utiliser les instructions GO TO et PERFORM. Pour pallier à cet inconvénient, il faut sectionner le programme, une section pour le corps du programme et autant de section que de procédures.

SORT ou MERGE

- Dans un programme, seules 10 instructions maximum de ce type peuvent exister. Une instruction SORT ou MERGE ne peut être appelée dans une INPUT ou OUTPUT PROCEDURE.

DONNEES

- Les arguments de tri ne doivent pas être dans une clause OCCURS.
- Tous les noms de données KEY doivent être décrits dans la première description d'article associée au nom de fichier, et uniquement dans celle-là.
- Toutes les zones d'arguments doivent se trouver dans les 4092 premières positions de l'article du fichier de tri.

NOTES

3. RAPPEL D'INSTRUCTIONS COBOL.

3.1. PERFORM EN LIGNE.

Afin de satisfaire aux règles de programmation structurée qui préconisent un enchaînement de blocs d'instructions sans débranchement vers des sections ou des paragraphes. Il faut remonter les instructions sans citer le nom de la procédure en l'encadrant par le verbe PERFORM …… et son délimiteur END-PERFORM.

Exemple :

```
PERFORM n TIMES
   ……
   ……
   ……
END-PERFORM
```

Ce PERFORM en ligne s'applique pour le PERFORM UNTIL et pour le PERFORM VARYING (limitée à un niveau).

Aucun point(.) ne doit se trouver entre le perform et le end-perform

Le PERFORM a 2 options (Perform en ligne ou en paragraphe)

- L'option WITH TEST BEFORE : C'est l'option par défaut (Compatibilité avec COBOL 74) qui évalue d'abord la condition puis exécute les instructions (0 à n fois).
- L'option WITH TEST AFTER : Elle exécute d'abord les instructions puis évalue la condition (1 à n fois).

NOTES

3.2. NOTION DE BLOC.

La notion de bloc d'instruction existe pour tous les verbes conditionnels. Il est interdit de trouver un point (.) dans ce bloc, il est possible de traiter les conditions inverses et la condition vraie doit toujours être en tête.

Exemples :

```
IF condition                READ fichier
   THEN instruction            AT END instruction
   ELSE instruction            NOT AT END instruction
END-IF                      END-READ
```

Les délimiteurs de fin de blocs sont les suivants :

END-ALL	END-IF	END-SEARCH
END-CALL	END-MULTIPLY	END-START
END-COMPUTE	END-PERFORM	END-STRING
END-DELETE	END-READ	END-SUBTRACT
END-DIVIDE	END-RETURN	END-UNSTRING
END-EVALUATE	END-REWRITE	END-WRITE

NOTES

3.3. TRAITEMENT DE CARACTÈRES.

Il est possible de modifier par référence.
Il est permis d'atteindre des caractères contigus à l'intérieur d'une donnée.

Exemples :

A(X:Y + Z)

Les Y + Z caractères de A à partir du Xième.

T(I + 1 J) (M + N:)

Les caractères à partir du (M + N)ième de l'élément de la table à 2 dimensions T de rangs I + 1 et J.

M(10:)

Du dixième caractère à la fin.

```
DATA DIVISION.
01 A PIC X(6) VALUE « ABCDEF ».
01 B PIC XXX.
PROCEDURE DIVISION.
MOVE A TO B  ========>  B=ABC
MOVE A(3:2) TO B  =====>  B=CD
```

NOTES

3.4. EVALUATE.

Elle permet d'évaluer des conditions multiples et de réaliser les actions correspondantes, elle peut traduire une table de vérité complexe.

Son format :

```
EVALUATE    Nom donnée 1                [ALSO   Nom donnée 2                ].
            Littéral 1                          Littéral 2
            Expression 1                        Expression 2
            TRUE                                TRUE
            FALSE                               FALSE
  WHEN      ANY                         [ALSO   ANY                        ]
            Condition 1                         Condition 2
            TRUE                                TRUE
            FALSE                               FALSE
            [NOT] Nomdonnée3                    [NOT] Nomdonnée5
            [THRU Nomdonnée4]                   [THRU Nomdonnée6]
      Ordre-impératif 1
  WHEN OTHER Ordre impératif 2
END-EVALUATE
```

Soit Nomdonnée3, Nomdonnée4, Nomdonnée5, Nomdonnée6, soit Littéral3, Littéral4, Littéral5, Littéral6,
soit Expression3, Expression4, Expression5, Expression6.

Exemple :

A midi je veux déjeuner :
Si la cantine est bonne, j'y vais.
Si elle n'est pas bonne et qu'il y a un restaurant proche, je vais au restaurant.
Si ce restaurant est éloigné, je vais à la cantine.
S'il n'y a pas de cantine et qu'il y a un restaurant proche, je vais au restaurant.
S'il y a ni cantine, ni restaurant, je ne déjeune pas.

NOTES

1ére forme possible :

```
EVALUATE TRUE ALSO TRUE ALSO TRUE
    WHEN CANT = « O » ALSO BON = « O » ALSO ANY
            DISPLAY « Il y a une bonne cantine, j'y vais »
    WHEN CANT = « O » ALSO BON = « N » ALSO RES = « O »
            DISPLAY « La cantine n'est pas bonne, je vais au restaurant »
    WHEN CANT = « O » ALSO BON = « N » ALSO RES = « N »
            DISPLAY « Mauvaise cantine, pas de resto, va pour la cantine »
    WHEN CANT = « N » ALSO ANY ALSO RES = « O »
            DISPLAY « Pas de cantine, heureusement il y a un resto »
    WHEN CANT = « N » ALSO ANY ALSO RES = « N »
            DISPLAY « Je ne déjeune pas »
END-EVALUATE.
```

2éme forme possible :

```
EVALUATE CANT ALSO BON ALSO RES
    WHEN « O » ALSO « O » ALSO ANY
            DISPLAY « Il y a une bonne cantine, j'y vais »
    WHEN « O » ALSO « N » ALSO « O »
            DISPLAY « La cantine n'est pas bonne, je vais au restaurant »
    WHEN « O » ALSO « N » ALSO « N »
            DISPLAY « Mauvaise cantine, pas de resto, va pour la cantine »
    WHEN « N » ALSO ANY ALSO « O »
            DISPLAY « Pas de cantine, heureusement il y a un resto »
    WHEN « N » ALSO ANY ALSO « N »
            DISPLAY « Je ne déjeune pas »
END-EVALUATE.
```

3éme forme possible :

```
EVALUATE « O » ALSO « O » ALSO « O »
    WHEN CANT ALSO BON  ALSO ANY
            DISPLAY « Il y a une bonne cantine, j'y vais »
    WHEN CANT ALSO NOT BON ALSO RES
            DISPLAY « La cantine n'est pas bonne, je vais au restaurant »
    WHEN CANT ALSO NOT BON ALSO NOT RES
            DISPLAY « Mauvaise cantine, pas de resto, va pour la cantine »
    WHEN NOT CANT ALSO RES ALSO ANY
            DISPLAY « Pas de cantine, heureusement il y a un resto »
    WHEN NOT CANT ALSO ANY ALSO NOT RES
            DISPLAY « Je ne déjeune pas »
END-EVALUATE.
```

NOTES

4. JEU D'ESSAI.

4.1. PRINCIPE GÉNÉRAL.

4.1.1. POURQUOI ?

Les mises en exploitation de programmes mettent en évidence un manque de test. Certaines parties de ces programmes n'ont pas été validées du fait d'un jeu d'essai mal évalué. Il apparaît donc nécessaire de pouvoir identifier les blocs non validés par le jeu d'essai afin de l'améliorer pour éviter tout problème lors de l'exploitation du programme.

4.1.2. COMMENT ?

Le COBOL 85 donne la possibilité de mettre en oeuvre une option permettant à la compilation d'indiquer que l'on veut effectuer un test de couverture du jeu d'essai pour un programme donné. Il existe plusieurs types de résultats produits par cet utilitaire.

4.2. MISE EN OEUVRE.

4.2.1. DANS LE JCL DE COMPILATION.

Pour intégrer au programme cobol compilé le code nécessaire au contrôle de couverture du jeu d'essai, il faut ajouter l'option **COVLIB** à la commande de compilation CBL. Celle-ci a pour but d'indiquer la bibliothèque où seront stockés les résultats intermédiaires du traitement.

Syntaxe du JCL :

```
CBL SOURCE=Nomprogr,INLIB=.COBLIB,MAP,XREF,
      LEVEL=NSTD,COVLIB=.PRTLIB;
```

NOTES

4.2.2. DANS LE JCL D'EXÉCUTION.

Pour exécuter le programme compilé avec **'COVLIB'**, il faut rajouter l'option **OPTIONS** à la commande **STEP** du JCL d'exécution.

Syntaxe du JCL

$**STEP** Nomprogr,.LMLIB,**OPTIONS**=« **COVLIB** NomProgr **SULV** »;

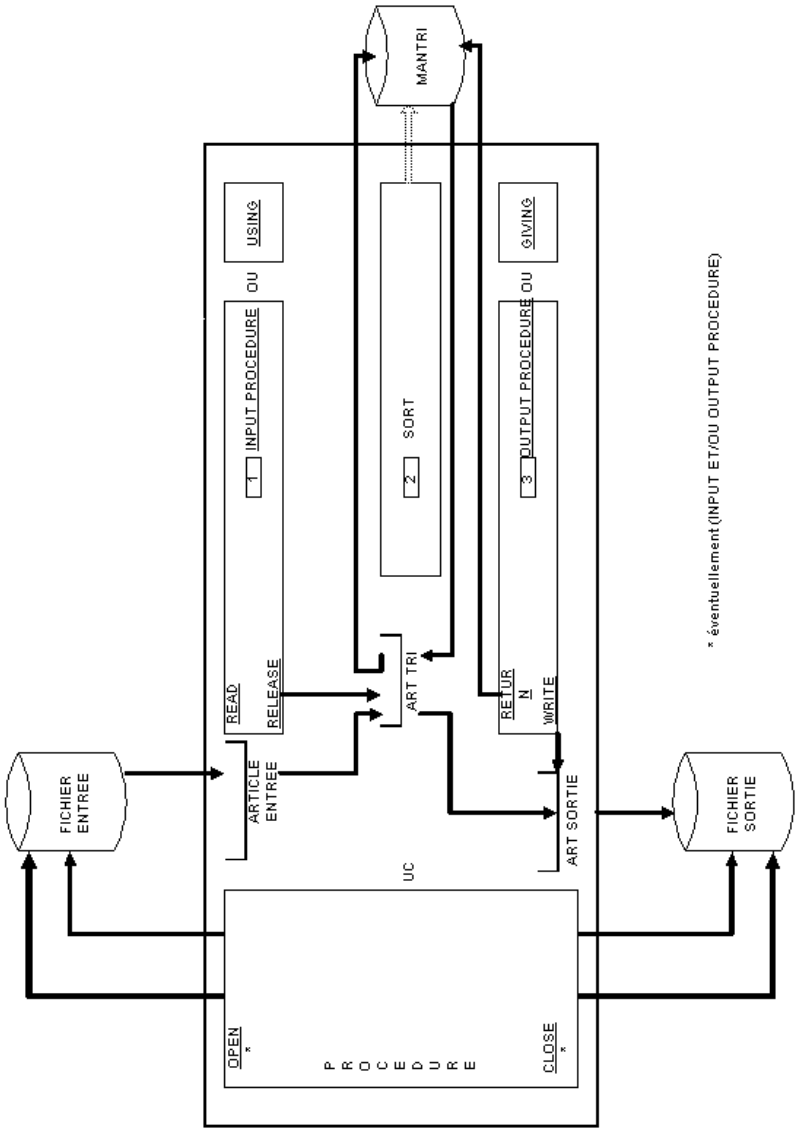
Les options **S,U,L** ou **V** indiquent les résultats que l'on désire obtenir :

- **S**: « **S**hort Listing » produit un histogramme de l'exécution de toutes les lignes du programme.
- **U** : « **U**nused Path » donne la liste non exécutées.
- **L** : « **L**ong Listing » donne pour chaque ligne le type de verbe, le pourcentage d'exécution et pour chaque test la non-condition par un mot unique **OTHERWISE**.
- **V** : « **V**erb Usage » donne le pourcentage d'exécution pour chaque verbe cobol différent.

NOTES

LE TRI.

PRINCIPES GENERAUX DU TRI.



NOTES

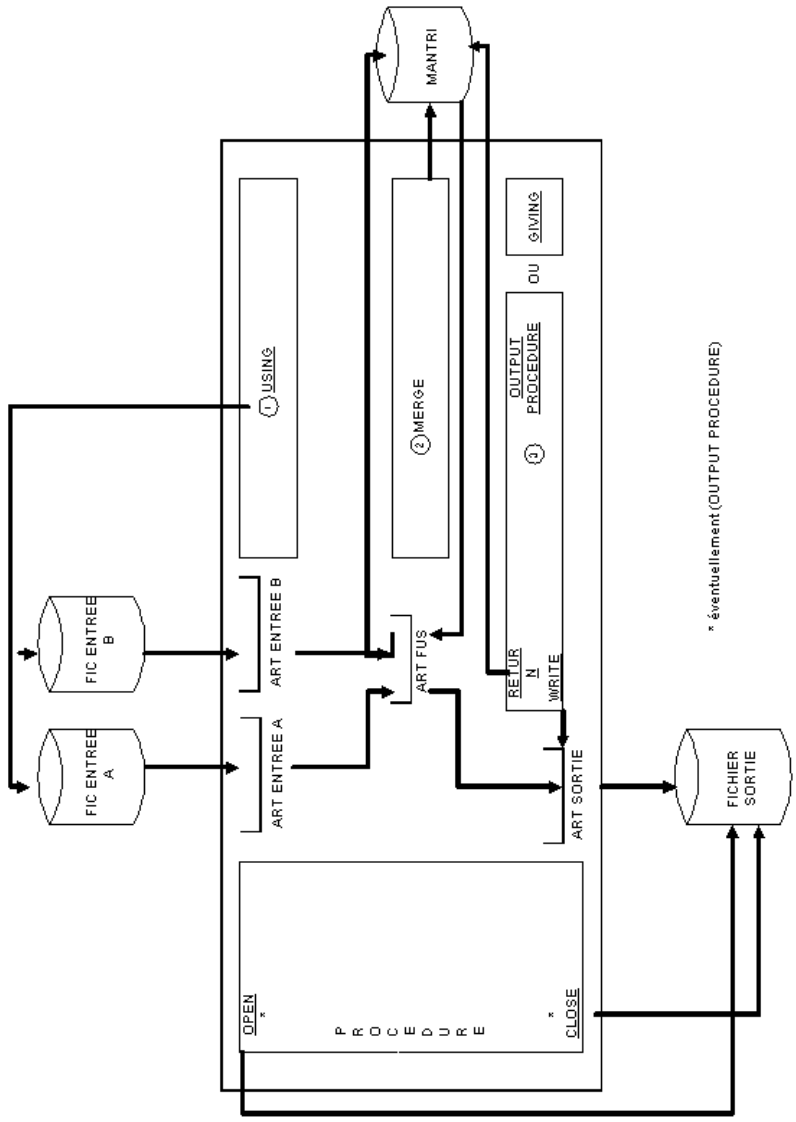
LA FUSION.

PRINCIPES GENERAUX DE LA FUSION.

Remarque :

Il n'est pas utile d'ouvrir, lire ou fermer les fichiers en entrée car la fusion les gère.

S'il existe plusieurs fichiers en sortie, il faut les ouvrir, les écrire et les fermer.



NOTES

TP5A

ECRITURE D'UN PROGRAMME COMPLEXE

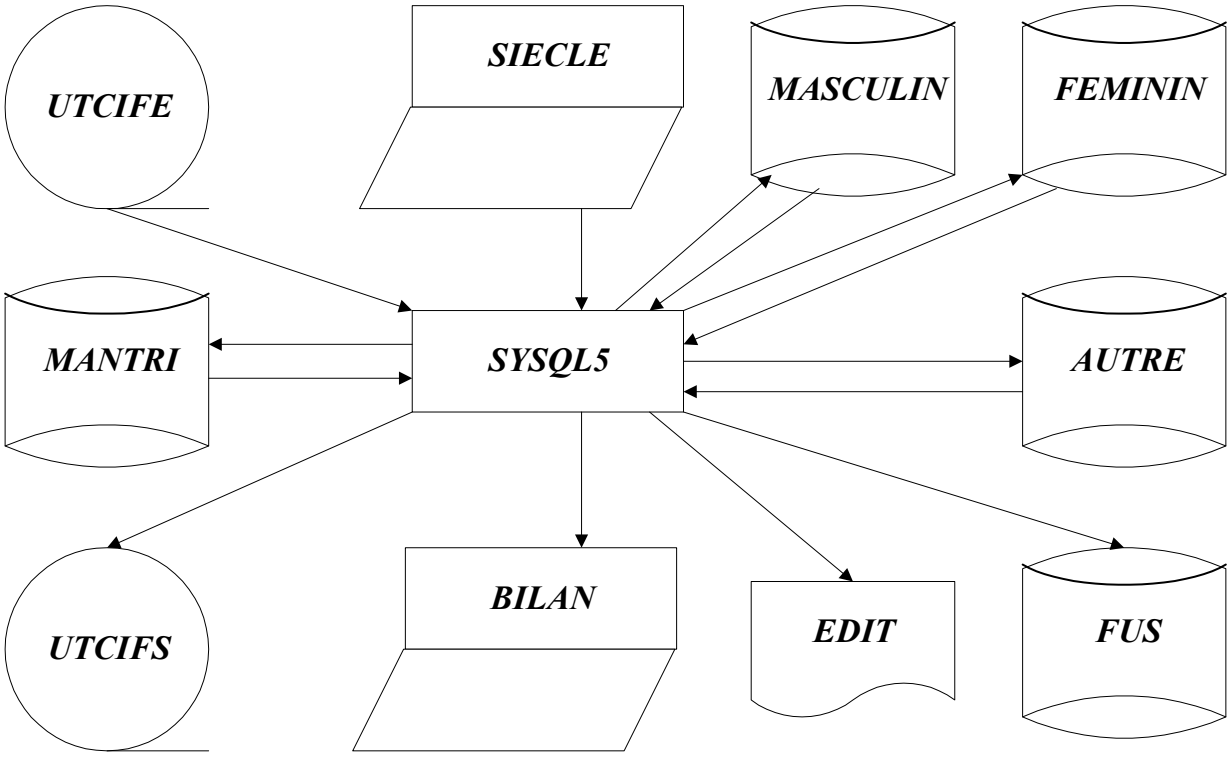
- reprenant les traitements proposés dans le module 4ter de saisie validation de siècle, de traitement d'un fichier de prénoms regroupés par date de fête (table et index), avec vérification de date, de validité de nom, affichage de messages d'erreur,

- en les incluants dans un TRI cobol, en développant maintenant le traitement de réorganisation des informations valides dans un fichier de sortie (avec table), leur édition, puis par éclatement de ces informations selon le sexe en trois fichiers supplémentaires,

- en fin en fusionnant ces trois derniers fichiers dans une FUSION cobol.

NOM DU PROGRAMME : ***SYSQL5*** :

DIAGRAMME :



FICHIERS : un fichier en entrée, trois fichiers en sortie (dont une édition), trois fichiers en sortie puis en entrée, un fichier de manoeuvre en entrée/sortie

Fichier séquentiel en entrée : ifn UTC1FE, efn UTC1FE, fd UTC1FE

nom d'article	ART-ENT		
zone de codification d'article	CODART	1	caractère numérique
zone date de fête	DATENT	4	caractères numériques
zone inutilisée		3	caractères alphanum.
zone des prénoms et sexes	ZONE	6	postes
zone d'un prénom-sexe	ZPRENOM	indexé par I	
zone d'un prénom	PRENOM	11	caractères alphanum.
zone du sexe	SEXE	1	caractère alphanum.

Fichier séquentiel en sortie : ifn UTC1FS, efn UTC1FS, fd UTC1FS

nom d'article	ART-SORTIE		
zone d'un prénom	LIBPRENOM	11	caractères alphanum.
zone nombre de dates de fête	ZONENB	2	caractères numériques
zone des dates de fête	SDATE	5	postes
zone d'une date de fête	DATESORT		
zone du mois	MMSORT	2	caractères numériques
zone du jour	JJSORT	2	caractères numériques
zone inutilisée		7	caractères alphanum.

Fichier séquentiel en sortie puis en entrée : ifn MASCULIN, efn MASCULIN, fd MASCULIN

nom d'article	ART-MASCUL		
zone d'un prénom/sexe	PRENMASC	12	caractères alphanum.

Fichier séquentiel en sortie puis en entrée : ifn FEMININ, efn FEMININ, fd FEMININ

nom d'article	ART-FEMINI		
zone d'un prénom/sexe	PRENFEMI	12	caractères alphanum.

Fichier séquentiel en sortie puis en entrée : ifn AUTRE, efn AUTRE, fd AUTRE

nom d'article	ART-AUTRE		
zone d'un prénom/sexe	PRENAUTR	12	caractères alphanum.

Fichier séquentiel en sortie : ifn FUS, efn FUS, fd FUS			
nom d'article	ART-FUS		
zone d'un prénom/sexe	PRENFUS	12	caractères alphanum.
zone date	DATEFUS	4	caractères numérique

Fichier séquentiel en sortie : ifn EDIT, efn PR1-SYSOUT, fd EDIT			
nom d'article	ART-EDIT		
ligne d'édition	LIGN1	132	caractères alphanum.

Fichier manoeuvre en entrée/sortie : ifn MANTRI, efn H_SORT, sd MANTRI			
nom d'article	ART-TRI		
zone d'un prénom/sexe	PRENTRI	12	caractères alphanum.
zone date	DATETRI	4	caractères numérique

TRAITEMENT :

Remarque : l'intégration de certains sous-programmes devra permettre la définition globale de certaines zones.

- le siècle à saisir à l'écran sera de type « SS » sur 2 caractères numériques.
- le sous-programme SYMSGER4 sera appelé en cas de siècle non numérique.
- le code erreur à passer au sous-programme SYMSGER4 est « 004 » en cas de siècle erronée.

Chaque article du fichier entrée sera lu et compté dans le compteur CPTLU ; la date contenue sera vérifiée :

- le sous-programme SYVRIF2D sera intégré au programme principal sous le nom SYVRIF3D où sera fait appel au sous-programme SYMSGER4 ;
- en cas d'invalidité de la date le sous-programme SYVRIF3D sera appelé et l'article éliminé.

Chaque prénom sera traité ; la zone des prénoms sera vérifiée :

- le sous-programme SYVRIF2N sera intégré au programme principal sous le nom SYVRIF3N où sera fait appel au sous-programme SYMSGER4 ;
- en cas d'invalidité du prénom le sous-programme SYVRIF3N sera appelé et l'article éliminé.

Tout article éliminé sera compté dans le compteur CPTELIM et les zones DATENT et ZONE affichés à l'écran précédé du libellé « ART-ENT ».

Tout prénom lu sera compté dans CPTPRENLU.

A partir du fichier entrée UTC1FE, créer les fichiers sorties

UTC1FS trié sur	zone libellé prénom en séquence ascendante zone date en séquence descendante ;
FUS fusionné sur	zone libellé prénom en séquence ascendante.

Les articles crées et triés du fichier UTC1FS seront listés.

L'édition comportera : 5 articles par ligne,
des positions 8 à 30, 31 à 53, 54 à 76, 77 à 99, 100 à 122,
même les doublons, sans la notion de sexe ;
50 lignes par page.

Une zone article est composée :

- d'un numéro de séquence croissant de 1 en 1, sur 3 caractères numériques,
- d'un blanc,
- du libellé prénom sur 12 caractères alphanumériques,
- du jour de fête sur 4 caractères numériques.

Un sous-programme des écritures des articles des fichiers d'éclatement sera selon intégré au programme principal sous le nom SYECRFIC : la valeur du code sexe sera évaluée pour réaliser les ordres d'écriture :

- M pour MASCULIN comptés dans CPTMASC
- F pour FEMININ comptés dans CPTFEM
- A pour AUTRE comptés dans CPTAUT.

En fin de traitement le message suivant sera affiché à l'écran :

```
''    «                                          «
      «            BILAN DE PASSAGE              «
      «            _______________               «
      «                                          «
      «            PROG. : SYSQL5                «
      «                                          «
      « ARTICLES LUS SUR UTC1FE  :               «   CPTLU
      « ARTICLES ELIMINES        :               «   CPTELIM
      « ARTICLE CREES SUR UTC1FS      :          «   CPTECRIT
      « NOMBRE DE PRENOMS LUS    :               «   CPTPRENLU
      « ARTICLES CREES MASCULINS      :          «   CPTMASC
      « ARTICLES CREES FEMININS  :               «   CPTFEM
      « ARTICLES CREES AUTRES    :               «   CPTAUT      ''
```