Normierte Programmierung (nach DIN 66220)

Rahmen für eine Programmablaufsteuerung

Hauptprogramm mit Steuerlogik und Ansteuerung der Standard-Unterprogramme:
Lesen, Satzfreigabe, Gruppenkontrolle/-wechsel, Einzelverarbeitung
Vorprogramm und Schlussverarbeitung
und mit allen Datendefinitionen für die Ablaufsteuerung

Autor: Wikipedia-Benutzer 'VÖRBY' Stand: 28. Oktober 2011

Inhaltsverzeichnis:

Vorbemerkungen:

```
* - Dieser Programmrahmen wurde, basierend auf einer anderen Lösung,
*   weitgehend an die Logik und die Namenskonventionen in
*   [http://de.wikipedia.org/wiki/Normierte_Programmierung] angepasst.
* - Unterschiede bestehen in folgenden Punkten:
*   -Die dateispezifischen Begriffe heißen nicht L1, L2, .. sondern werden
*    über eine Tabelle mit 'Lx(Datei-Index)' angesprochen; Codeoptimierung
*   -Die Zusammenfassung der Grp-Begriffe wurden nicht L4, L3 ... genannt,
*    sondern L1-4, L1-3 usw.; Grund: bessere Lesbarkeit.
*    Sie umfassen nur die fachlichen Datenfelder - ohne Dateistatus & Prio.
* - Das Schema behandelt 4 steuernde Eingabedateien und 4 Gruppenstufen.
*   Aufgabenspezifisch sind also ggf. Anpassungen erforderlich.
* - Alle Stellen mit Anpassungsbedarf sind in roter Schrift gehalten;
*   Begriffe wie 'Bestand_A' können ggf. auch unverändert bleiben.
* - Der Rahmen ist in einer Cobol-orientierten Syntax erstellt.
*   Anpassungen an eine konkrete Compilerversion sind noch erforderlich.
*   Im Besonderen gilt dies für Schleifenkonstrukte, Move-Befehle etc.
* - Zum Teil sind Anweisungen wie exit, endIf usw. eingebaut - zum besseren
*   Verständnis von Codesequenzen mit bedingter Verarbeitung.
*   Eine 'Section' ist immer eine Subroutine - mit Rücksprung am Ende.
* - mit '*' beginnende Zeilen oder Textteile sind Kommentare.
* - Erläuterungen in 'multiplen' Abschnitten (wie Datei_A bis x) gelten
*   sinngemäß für alle n-Ausprägungen.
* - Achtung, wird 'steuernder Code' verändert, entsteht das Risiko, dass
*   das Rahmenprogramm nicht mehr korrekt arbeitet.
* - Lizenz: Das Template darf kopiert und frei benutzt werden.
*   Jegliche Haftung des Autors ist ausgeschlossen.
* - Das für die Entwicklungsmethodik zuständige Team eines Unternehmens
*   könnte z.B. ein solches Template auf eigene Gegebenheiten (Compiler,
*   Namens- und andere Regeln) anpassen, sprachspezifische Angaben ergänzen
*   und diesen 'leeren Programmrahmen' für die Entwickler bereitstellen.
*   Nach deren Anpassungen würde ein lauffähiges Programm vorliegen, das
*   alle seine Dateien liest, leer 'verarbeitet' und regulär endet.
```

Datendefinitionen für das Programm

```
** Datenbereiche für steuernde Eingabedateien
* Bestandsbezeichnungen entsprechend anpassen!
  01 BESTAND_A.
*    (Bestandsname)
     02 SATZ_A.
*       (Name des Einzelobjekts in diesem Bestand - optional)
>>      Include xxxx bzw. individuelle Definitionen,
*       So sind ggf. auch Redefinitionen möglich.

  01 BESTAND_B.
     02 SATZ_B.
>>      Include xxxx

  01 BESTAND_C
     02 SATZ_C.
>>      Include xxxx

  01 BESTAND_D.
     02 SATZ_D.
>>      Include ???

** DATEIBEZOGENE GRUPPENBEGRIFFSFELDER ***********************************
  01 DATEI_GRUPPEN_BEGRIFFE.
*    Datei-Gruppenfelder je steuernder Datei
*    Achtung, auch bei LA und LN identische Struktur schaffen!
*    Die Einzelfelder müssen identisches Format aufweisen, ggf.immer PIC
  02 Lx     occurs 4.          *Anzahl steuernder Eingabedateien
*           identisch zu XDAT_MAX, Belegung siehe Routine A_VOR_INIT
     03 LX_STAT               PIC X(01).
*       Datei-Status:
*        0 = Satz lesen
         1 = Satz ist gelesen
         2 = Datei abgeschlossen
         3 = Datei nicht vorhanden
     03 LX_SORT.
*       Sortierbegriff der jeweiligen Datei - inkl. Prio
       04 LX_BGRF.
*         Fachlicher Key, nachfolgend nur Beispiel
          05 LX_1              PIC  X(08).
          05 LX_2              PIC  X(02).
          usw.
       04 LX_PRIO              PIC  X(01).
*         Priorität - steuert die Verarbeitungsfolge
*                     bei gleichem fachlichem Sortierbegriff
*         Init in der Vorroutine
```

```
** DATEINEUTRALE GRUPPENBEGRIFFSFELDER ****************************
  01 NEUTRALE_GRUPPEN_BEGRIFFE.
     02 LA.                    Alter Gruppenbegriff
     03 LA_STAT                PIC  X(01).
     03 LA_SORT.
       04 LA_BGRF.
         05 LA_1.
>>        06 LA_STAMM           PIC  X(08).   *Beispiel
*            alternativ zu LA_1 als fachliche Feldbezeichnung ansprechbar
         05 LA_2.
>>        06 LA_VNR             PIC  X(02).
>>       05 LA_3                usw.
>>       05 LA_4                usw.
*      04 Gruppierungen_Gruppenbegriffe redefines LA_BGRF.
         05 LA1-1   redefines   LA_BGRF   PIC  X(08).  *identisch zu LA_1
         05 LA1-2   redefines   LA_BGRF   PIC  X(10).
         05 LA1-3   redefines   LA_BGRF   PIC  X(>>).
         05 LA1-4   redefines   LA_BGRF   PIC  X(>>).
       04 LA_PRIO               PIC  X(01).

     02 LN.                    Neuer Gruppenbegriff
     03 LN_STAT                PIC  X(01).
     03 LN_SORT.
       04 LN_BGRF.
*>>       nachfolgend Feldnamen der Gruppenbegriffe eintragen.
         05 LN_1.
>>        06 LN_STAMM           PIC  X(08).
         05 LN_2.
>>        06 LN_VNR             PIC  X(02).
>>       05 LN_3.               usw.
>>       05 LN_4.               usw.
*      04 Gruppierungen_Gruppenbegriffe redefines LN_BGRF.
         05 LN1-1   redefines   LN_BGRF   PIC  X(08).  *identisch mit LN_1
         05 LN1-2   redefines   LN_BGRF   PIC  X(10).
         05 LN1-3   redefines   LN_BGRF   PIC  X(>>).
         05 LN1-4   redefines   LN_BGRF   PIC  X(>>).
       04 LN_PRIO               PIC  X(01).

** SCHALTER ********************************************
01 Q_SCHALTER.
*       derzeit keine Schalter definiert;
*       Durchlauf 1 wird über LA-STAT = X'00' gesteuert

** Indexe ***********************************************************
01 X_BEREICH.
*     Indexfelder
     02 XDAT                   PIC s9(4) comp.
*       ausgewählte Datei in DATEI_TAB, Einstellung in 'C_Satzfreigabe'
     02 XDAT_MAX               PIC s9(4) comp.
*>>     Anzahl zu mischender Dateien
     03 XWORK                  PIC s9(4).
*       kurzfristige Verwendung
```

Haupt- / Steuerprogramm

STEUERPROGRAMM Section

```
    Perform A_VORPROGRAMM

    While LN_STAT < "2"    * hier Schleifen-Ende / nicht Ende
     Perform B_EINGABE

     Perform C_SATZFREIGABE
*    Ergebnis: GrpBegriff Alt und Neu sind aktualisiert
*              XDAT adressiert die ausgewählte Datei

     Perform D_GRUPPENKONTROLLE
*            Feststellen der eingetretenen Gruppenwechselstufe
*            dort ggf. Aufrufen der NACHLAUF- und Vorlaufroutinen

     IF LN_STAT = "1"    * Satz liegt vor; kein EOF
        Perform E_EINZELVERARBEITUNG      (ein (!) Datensatz)
        LX_STAT (XDAT) = "0"              (nachlesen)
    EndWhile

    PERFORM F_SCHLUSSPROGRAMM
```

Block A: VORPROGRAMM

A_VORPROGRAMM Section

```
* Evtl. Abbruchbedingungen wären noch zu programmieren; Direktabbruch?
* Die Unterteilung in 3 Teile ist ein Vorschlag, der veränderbar ist
     Perform A_VOR_INIT
*    Initialisieren von Programm-Datenbereichen

     Perform A_VOR1
*    Funktionaler Code - z.B. dateibezogen
*    z.B. Open Files und Lesen Parameter etc.

     Perform A_VOR2
*            weitere individuelle Codeteile nach Open etc.
```

A_VOR_INIT Section

```
** Initialisieren von Datenbereichen zur Steuerung:
*  alternativ die Daten ggf. mit INIT entsprechend definieren
      XDAT_MAX = 4              Anzahl steuernder Dateien

**   Prio-Kennzeichen statisch setzen, d.h. keine Änderung im Pgm-Ablauf!
*    Prio-Festlegung nach fachlichen ablauftechnischen Gegebenheiten.
*    Hinweis: Hier im Beispiel sind Index und Prio nicht gleich;
*    die Priorität wäre HIER zwar leicht änderbar, ist jedoch im Code ggf.
*    auch zu berücksichtigen (für Zugriff auf 'fremde' Satzinhalte)!
*     Prio für Bestand_A                          Index = 1
>>    Lx_PRIO(1) = 4
*     Prio für Bestand_B                          Index = 2
>>    Lx_PRIO(2) = 2
*     Prio für Bestand_C                          Index = 3
>>    Lx_PRIO(3) = 3
*     Prio für Bestand_D                          Index = 4
>>    Lx_PRIO(4) = 1

*    Status für alle Dateien auf 'nachlesen'
      LX_STAT(1) = "0"
      LX_STAT(2) = "0"
      LX_STAT(3) = "0"
      LX_STAT(4) = "0"
      LA = X'00'            (oder per Init; Anfangswert)
      LN = X'00'            (oder per Init)
```

A_VOR1 Section

```
* 'Evtl. dateibezogener Code; z.B. open aller Dateien
    >> hier Code
*  Lesen von Parametern, nach denen steuernde Dateien entfallen;
*     in einem solchen Fall:
      LX_STAT(Index dieser Datei) = "3"
```

A_VOR2 Section

```
** weitere Verarbeitung der Vorroutine:

************************************************************
```

Block B: Eingabe

B_EINGABE Section

```
     While XDAT from 1 to XDAT_MAX
       CASE = XDAT:
       = 1:  Perform LESEN_BESTAND_A
* zur Optimierung evtl. jeweils bedingt aufrufen (Lx_STAT (XDAT) = "0")
       = 2:  Perform LESEN_BESTAND_B
       = 3   Perform LESEN_BESTAND_C
       = 4   Perform LESEN_BESTAND_D
      CASE END
     While End
Exit
```

B_LESEN_BESTAND_A Section

```
** Steuern des Lesens für Bestand_A, ggf. Filteraufruf:
** Diesen Code für alle relevanten Dateien kopieren
*    while Lx_STAT (XDAT) = "0"
*     Eingabe ist zu lesen, ggf. nachlesen nach Filterung
      READ BESTAND_A       *Lesebefehl oder eigene Subroutine
        wenn EOF:
           Lx_STAT (XDAT) = "2
           LX_BGRF = X'FF'
*          Achtung: Hierdurch soll nach dem letzten Satz ein Wechsel auf
*          jeden Fall erkannt werden; ggf. auf andere Werte setzen
        Else
*          evtl. hier Sortierfolgeprüfung vornehmen (oder in Filter)
*          Lx_STAT (XDAT) = "1"
*  ggf:    PERFORM B_FILTER_BESTAND_A
*           dort wird Lx_STAT (XDAT) ggf. auf "0" zurückgesetzt
           IF Lx_STAT (XDAT) = "1"
>>            Move <Sortierfelder> to Lx_1, Lx_2, Lx_3, Lx_4 (XDAT)
           endIf
*    End While
```

B_LESEN_BESTAND_B usw. Section

```
* wie BESTAND_A
```

B_LESEN_BESTAND_C usw. Section

```
* wie BESTAND_A
```

B_LESEN_BESTAND_D usw. Section

```
* wie BESTAND_A
```

B_FILTER_BESTAND_A Section

```
* (aus der Leseroutine aufgerufen)
* ggf. überlesen von Datensätzen. Diese gehen nicht in die Steuerung ein,
* werden also behandelt, als ob sie nicht da wären.
* Lx_STAT (XDAT) steuert ob überlesen werden soll; Wert bei Aufruf = "1"
*     IF >> Bedingung für Überlesen <<<
         Lx_STAT (XDAT) = "0"  *führt in B_LESEN  zum Nachlesen
```

B_FILTER_BESTAND_B usw. Section

```
* wie für Bestand_A - falls gefiltert werden muss.
```

B_FILTER_BESTAND_C usw. Section

```
* wie für Bestand_A - falls gefiltert werden muss.
```

B_FILTER_BESTAND_D usw. Section

```
* wie für Bestand_A - falls gefiltert werden muss.
```

Block C: SATZFREIGABE

C_SATZFREIGABE Section

```
    XDAT = 1
*      potenziell niedrigster Satz
    While XWORK from 2    to XDAT_MAX
       IF LX-SORT(XWORK) < LX-SORT(XDAT) then
*         bei absteigender Sortierfolge ist die Logik anzupassen!
          XDAT = XWORK
       END IF
       (Add 1              to XWORK)      * das tut das While-Konstrukt
    End While

*   XDAT zeigt jetzt auf die Datei mit dem niedrigsten Gruppenbegriff
    LA = LN
*    bisheriger (bzw. Init-) Gruppenbegriff NEU nach Grp-Begriff ALT
    LN = LX(XDAT)
*    gesamten Gruppenbegriff inkl. Status und Prio als NEU-GrpBegriff.
*    XDAT bleibt erhalten!
```

Block D: GRUPPENKONTROLLE

D_GRUPPENKONTROLLE Section

```
**   Aufruf der NACHLAUF-Routinen:
     IF LA_Stat > X'00'       *erster Aufruf; keine NACHLÄUFE
        IF LN_1-4 <> LA_1-4    *(= Optimierung)
           Perform DN_GRUPPENNACHLAUF
        endIF
     endIF

** Aufruf der VORLAUF-Routinen:
    IF LN_STAT < "2"      *d.h. es steht ein Datensatz zur Verarbeitung an
*       (>= 2 = nach letztem Satz; keine Vorläufe mehr ausführen)
        IF LN_1-4 <> LA_1-4     *(= Optimierung)
           Perform DV_GRUPPENVORLÄUFE
        END IF
    endIF
Exit D_GRUPPENKONTROLLE
```

DV_GRUPPENVORLÄUFE Section

```
* Durchlauf von höchstem bis niedrigstem Vorlauf, höhere Stufen bedingt
*   Anzahl Gruppen ggf. anpassen
    IF LN_1-1 <> LA_1-1
       PERFORM GV1_VORLAUF_<Feldname>
    IF LN_1-2 <> LA_1-2        *(Das ist auch bei LN1<>LA1 der Fall, ff.)
       PERFORM GV2_VORLAUF_<Feldname>
    IF LN_1-3 <> LA_1-3
       PERFORM GV3_VORLAUF_<Feldname>
    IF LN_1-4 <> LA_1-4
       PERFORM GV4_VORLAUF_<Feldname>
```

DN_GRUPPENNACHLAEUFE Section

```
* Durchlauf vom niedrigsten zum höchsten NACHLAUF, höhere Stufen bedingt
*   Anzahl Gruppen ggf. anpassen
    IF LN_1-4 <> LA_1-4
       PERFORM GN4_NACHLAUF_<Feldname>
    IF LN_1-3 <> LA_1-3
       PERFORM GN3_NACHLAUF_<Feldname>
    IF LN_1-2 <> LA_1-2
       PERFORM GN2_NACHLAUF_<Feldname>
    IF LN_1   <> LA_1
       PERFORM GN1_NACHLAUF_<Feldname>
```

E EINZELVERARBEITUNG

E_EINZELVERARBEITUNG Section

```
**  Steuerung der Einzelverarbeitung - alternativ je Datei!
    CASE = XDAT:    Datei mit dem derzeit niedrigsten Sortierbegriff
*                   Satzinhalt steht im jeweiligen Eingabebereich,
*                   die Grp-Begriffe daraus stehen in LN_BGRF
* Felder aus anderen Eingabedateien nur aus Zwischenspeichern ansprechen
      = 1:  Perform E1_BESTAND_A
      = 2:  Perform E2_BESTAND_B
      = 3:  Perform E3_BESTAND_C
      = 4:  Perform E4_BESTAND_D
    ENDCASE
EXIT
```

E1_BESTAND_A Section

```
* Verarbeitung Datensatz aus Bestand_A
    xxxx
```

E2_BESTAND_B Section

```
* Verarbeitung Datensatz aus Bestand_B
    xxxx
```

E3_BESTAND_C Section

```
* Verarbeitung Datensatz aus Bestand_C
    xxxx
```

E4_BESTAND_D Section

```
* Verarbeitung Datensatz aus Bestand_D
    xxxx
```

F SCHLUSSPROGRAMM

F_SCHLUSSPROGRAMM Section

```
* Verarbeitung am Ende des Programms
```

G Unterprogramme Gruppenverarbeitung

```
*  GV = Vorläufe
*  GN = NACHLAUF
```

GV1_VORLAUF_<Feldname> Section

    xxxx

GV2_VORLAUF_<Feldname> Section

    xxxx

GV3 VORLAUF_<Feldname> Section

    xxxx

GV4 VORLAUF_<Feldname> Section

    xxxx

GN1_NACHLAUF_<Feldname> Section

    xxxx

GN2_NACHLAUF_<Feldname> Section

    xxxx

GN3_NACHLAUF_<Feldname> Section

    xxxx

GN4_NACHLAUF_<Feldname> Section

    xxxx

Weitere individuelle Subroutinen

```
* >> weitere Unterprogramme, aufgerufen aus den Standardroutinen
```

H_Unterprogramme Verarbeitung

J_Unterprogramme der Ausgabe