Inhaltsverzeichnis

1. Vorwort

Wie im Abschluss vom Projekt mit dem COBOL Programm („Projekt csv nach xml“) erwähnt, bin ich noch nicht zufrieden mit Aussehen und Inhalt vom COBOL-Programm. Für die Verarbeitung der 700 Datensätze reicht das was wir bisher gemacht haben völlig aus, es ist nur nicht wirklich guter Code.

Zum einen ist er nicht besonders lesbar. Wenn wir da noch weitere Sections einbauen wollen, wird das schnell unübersichtlich. Das geht deutlich besser, ich hoffe auch Ihr werdet mir am Ende zustimmen.

Zum anderen haben wir noch jede Menge Code eingebaut, der kostbare Verarbeitungs-Zeit raubt, ich hoffe, dass ich das nachstellen kann.

Ausgangspunkt ist der Quellcode aus Kapitel 4.5 aus der Dokumentation (Projektdokumentation). Ich werde diesen Kapitel für Kapitel erweitern, die Zwischenschritte finden sich dann auch wieder nach Kapiteln sortiert als zip-Datei im Download-Bereich.

Das war der Vorrede genug, auf geht es in die erste Iteration.

2. Vorarbeiten

In den Vorarbeiten tun wir mehrere Dinge.

Zunächst bringen wir uns alle auf den aktuellen Stand des Quellcodes.

Dann bauen wir einen Zeitstempel für den Start und das Ende ein, damit wir die Laufzeit dazwischen berechnen können.

Im Anschluss erweitern wir die Eingabe der csv-Datei durch exzessives Vervielfachen der Inhalte, sodass wir eine ordentliche Testbasis haben.

Zum Schluss geben wir auch noch die Anzahl der gelesenen Sätze aus, dann haben wir einen guten Überblick über unser Programm.

2.1. Der aktuelle Quellcode

Damit wir alle auf dem gleichen Stand starten, hier noch einmal der Quellcode aus dem Kapitel 4.5 der oben verlinkten Dokumentation.

```
1       ******************************************************************
2       * Author:    papa
3       * Date:      Juni 2021
4       * Purpose:   Konvertieren von Inhalten aus csv nach xml
5       *            Ausbaustufe 4.5 Umlaute behandeln
6       * Tectonics: cobc
7       ******************************************************************
8        IDENTIFICATION DIVISION.
9        PROGRAM-ID. CSV-NACH-XML.
10       ENVIRONMENT DIVISION.
11       INPUT-OUTPUT SECTION.
12       FILE-CONTROL.
13
14       SELECT EINGABE ASSIGN TO 'kennzeichen.csv'
15       ORGANIZATION IS LINE SEQUENTIAL
16       FILE STATUS IS EINGABE-STATUS.
17
18       SELECT AUSGABE ASSIGN TO 'xml-ausgabe.xml'
19       ORGANIZATION IS LINE SEQUENTIAL
20       FILE STATUS IS AUSGABE-STATUS.
21
22       DATA DIVISION.
23       FILE SECTION.
24
25       FD EINGABE.
26       01 EINGABE-FILE.
27           05 EINGABE-GANZ             PIC X(130).
28
29       FD AUSGABE.
30       01 AUSGABE-FILE.
31           05 AUSGABE-ZEILE            PIC X(130).
32
```

```
33         WORKING-STORAGE SECTION.
34
35        *Definition der Schalter
36         01 EINGABE-ENDE-ERREICHT  PIC X(1).
37             88 EINGABE-ENDE-JA        VALUE "J".
38             88 EINGABE-ENDE-NEIN      VALUE "N".
39
40         01 FEHLER-SCHALTER        PIC X(1).
41             88 FEHLER-JA              VALUE "J".
42             88 FEHLER-NEIN            VALUE "N".
43
44         01 ERSTER-SATZ            PIC X(1).
45             88 ERSTER-SATZ-JA         VALUE "J".
46             88 ERSTER-SATZ-NEIN       VALUE "N".
47
48        *Definition der Variablen
49         01 VARIABLEN.
50             05 EINGABE-STATUS        PIC 9(2).
51             05 AUSGABE-STATUS        PIC 9(2).
52             05 WS-EINGABE            PIC X(130).
53             05 TEMP-AUSGABE-ZEILE    PIC X(130).
54             05 TEMP-LOG              PIC X(130).
55             05 WS-INHALT-ABK         PIC X(3).
56             05 WS-INHALT-STADT       PIC X(82).
57             05 WS-INHALT-ABGEL       PIC X(82).
58             05 WS-INHALT-BULAND      PIC X(82).
59             05 WS-TEMP-AUSGABE       PIC X(82).
60
61        *Definition der Konstanten
62         01 KONSTANTEN.
63             05 NUM-10                PIC 9(2) VALUE 10.
64             05 WS-AUSGABE.
65                10 AUS-VERSION.
66                   15 FILLER             PIC X(14) VALUE "<?xml version=".
67                   15 FILLER           PIC X(01) VALUE X'22'.
68                   15 FILLER           PIC X(03) VALUE "1.0".
69                   15 FILLER           PIC X(01) VALUE X'22'.
70                   15 FILLER           PIC X(02) VALUE "?>".
71                10 AUS-KENNZEICHEN-AUF PIC X(13) VALUE "<kennzeichen>".
72                10 AUS-KENNZEICHEN-ZU  PIC X(14) vALUE "</kennzeichen>".
73                10 AUS-RECORD_AUF      PIC X(13) VALUE "    <record>".
74                10 AUS-RECORD_ZU       PIC X(14) VALUE "    </record>".
75                10 AUS-ABK-AUF          PIC X(13) VALUE "        <Abk>".
76                10 AUS-ABK-ZU           PIC X(07) VALUE "</Abk>".
77                10 AUS-STADT-AUF        PIC X(25) VALUE
78                   "        <Stadt_Landkreis>".
79                10 AUS-STADT-ZU          PIC X(18) VALUE
80                   "</Stadt_Landkreis>".
81                10 AUS-ABGEL-AUF         PIC X(24) VALUE
82                   "        <abgeleitet_von>".
83                10 AUS-ABGEL-ZU          PIC X(17) VALUE
84                   "</abgeleitet_von>".
85                10 AUS-BULAND-AUF        PIC X(16) VALUE
86                   "        <Buland>".
87                10 AUS-BULAND-ZU         PIC X(17) VALUE
88                   "</Buland>".
89             05 CHAR-SEMICOLON           PIC X(1) VALUE ";".
```

```
 90       *    EINGEHENDE ZEICHEN
 91       *    GROSSES Ä
 92            05 HEXA-C4                    PIC X(01) VALUE X'C4'.
 93       *    GROSSES Ü
 94            05 HEXA-DC                    PIC X(01) VALUE X'DC'.
 95       *    GROSSES Ö
 96            05 HEXA-D6                    PIC X(01) VALUE X'D6'.
 97       *    KLEINES ä
 98            05 HEXA-E4                    PIC X(01) VALUE X'E4'.
 99       *    KLEINES ü
100            05 HEXA-FC                    PIC X(01) VALUE X'FC'.
101       *    KLEINES ö
102            05 HEXA-F6                    PIC X(01) VALUE X'F6'.
103       *    DAS SCHARFE S
104            05 HEXA-DF                    PIC X(01) VALUE X'DF'.
105       *    FILLER, ALLEN ZEICHEN IN DER AUSGABE VORANGESTELLT
106            05 HEXA-C3                    PIC X(01) VALUE X'c3'.
107       *    AUSGEHENDE ZEICHEN
108       *    GROSSES Ä
109            05 HEXA-84                    PIC X(01) VALUE X'84'.
110       *    GROSSES Ü
111            05 HEXA-9C                    PIC X(01) VALUE X'9c'.
112       *    GROSSES Ö
113            05 HEXA-96                    PIC X(01) VALUE X'96'.
114       *    KLEINES ä
115            05 HEXA-A4                    PIC X(01) VALUE X'a4'.
116       *    KLEINES ü
117            05 HEXA-BC                    PIC X(01) VALUE X'bc'.
118       *    KLEINES ö
119            05 HEXA-B6                    PIC X(01) VALUE X'b6'.
120       *    DAS SCHARFE S
121            05 HEXA-9F                    PIC X(01) VALUE X'9F'.
122
123       *interne Tabellen inklusive des REDEFINES-Feldes
124        01  RED-EIN-FELD                  PIC X(82).
125        01  TABELLE-1 REDEFINES RED-EIN-FELD
126                      OCCURS 82 TIMES INDEXED BY TAB-1-IND.
127            05 EIN-BYTE                   PIC X(1).
128        01  RED-AUS-FELD                  PIC X(83).
129        01  TABELLE-2 REDEFINES RED-AUS-FELD
130                      OCCURS 83 TIMES INDEXED BY TAB-2-IND.
131            05 AUS-BYTE                   PIC X(1).
132
133        PROCEDURE DIVISION.
134
135       *Beginn der Steuerung
136        STEUER SECTION.
137            PERFORM INITIALISIERUNG
138            PERFORM OEFFNEN-DATEIEN
139            PERFORM LESEN-DATENSATZ
140            PERFORM WITH TEST BEFORE UNTIL EINGABE-ENDE-JA OR FEHLER-JA
141                PERFORM KONVERTIERE-DATENSATZ
142                PERFORM LESEN-DATENSATZ
143            END-PERFORM
144            PERFORM ABSCHLUSSARBEITEN
145            PERFORM SCHLIESSEN-DATEIEN
146            STOP RUN.
```

```
147        *Ende der Steuerung
148
149        *Beginn der Prozeduren
150         INITIALISIERUNG SECTION.
151             DISPLAY "IN INITIALISIERUNG"
152
153        *    Alle Variablen auf einen Schlag initialisieren
154             INITIALIZE VARIABLEN
155
156        *    Setzen der Schalter auf Anfangszustand
157             SET EINGABE-ENDE-NEIN TO TRUE
158             SET FEHLER-NEIN TO TRUE
159             SET ERSTER-SATZ-JA TO TRUE
160         .
161         OEFFNEN-DATEIEN SECTION.
162             DISPLAY "IN OEFFNEN-DATEIEN"
163             OPEN INPUT EINGABE
164             OPEN OUTPUT AUSGABE
165         .
166         LESEN-DATENSATZ SECTION.
167             DISPLAY "IN LESEN-DATENSATZ"
168
169             READ EINGABE INTO WS-EINGABE
170                AT END SET EINGABE-ENDE-JA TO TRUE
171                NOT AT END DISPLAY EINGABE-GANZ
172             END-READ
173
174             EVALUATE EINGABE-STATUS
175                WHEN ZEROES
176                WHEN NUM-10
177                   CONTINUE
178                WHEN OTHER
179                   DISPLAY "EINGABE-STATUS = " EINGABE-STATUS
180                   SET FEHLER-JA TO TRUE
181             END-EVALUATE
182         .
183         KONVERTIERE-DATENSATZ SECTION.
184             DISPLAY "IN KONVERTIERE-DATENSATZ"
185
186             IF ERSTER-SATZ-JA
187                 MOVE AUS-VERSION            TO TEMP-AUSGABE-ZEILE
188                 PERFORM SCHRIEBEN-DATENSATZ
189                 MOVE AUS-KENNZEICHEN-AUF    TO TEMP-AUSGABE-ZEILE
190                 PERFORM SCHRIEBEN-DATENSATZ
191                 SET ERSTER-SATZ-NEIN        TO TRUE
192             END-IF
193
194             MOVE AUS-RECORD_AUF             TO TEMP-AUSGABE-ZEILE
195             PERFORM SCHRIEBEN-DATENSATZ
196
197             UNSTRING WS-EINGABE DELIMITED BY CHAR-SEMICOLON
198                INTO WS-INHALT-ABK,
199                     WS-INHALT-STADT,
200                     WS-INHALT-ABGEL,
201                     WS-INHALT-BULAND
202             END-UNSTRING
203
```

```
204            MOVE WS-INHALT-ABK        TO RED-EIN-FELD
205            PERFORM UMLAUTE-UMSETZEN
206
207            STRING AUS-ABK-AUF
208                   WS-TEMP-AUSGABE
209                   AUS-ABK-ZU
210            INTO TEMP-AUSGABE-ZEILE
211            PERFORM SCHRIEBEN-DATENSATZ
212
213            MOVE WS-INHALT-STADT      TO RED-EIN-FELD
214            PERFORM UMLAUTE-UMSETZEN
215
216            STRING AUS-STADT-AUF
217                   WS-TEMP-AUSGABE
218                   AUS-STADT-ZU
219            INTO TEMP-AUSGABE-ZEILE
220            PERFORM SCHRIEBEN-DATENSATZ
221
222            MOVE WS-INHALT-ABGEL      TO RED-EIN-FELD
223            PERFORM UMLAUTE-UMSETZEN
224
225            STRING AUS-ABGEL-AUF
226                   WS-TEMP-AUSGABE
227                   AUS-ABGEL-ZU
228            INTO TEMP-AUSGABE-ZEILE
229            PERFORM SCHRIEBEN-DATENSATZ
230
231            MOVE WS-INHALT-BULAND     TO RED-EIN-FELD
232            PERFORM UMLAUTE-UMSETZEN
233
234            STRING AUS-BULAND-AUF
235                   WS-TEMP-AUSGABE
236                   AUS-BULAND-ZU
237            INTO TEMP-AUSGABE-ZEILE
238            PERFORM SCHRIEBEN-DATENSATZ
239 #
240            MOVE AUS-RECORD_ZU           TO TEMP-AUSGABE-ZEILE
241            PERFORM SCHRIEBEN-DATENSATZ
242            .
243
244         UMLAUTE-UMSETZEN SECTION.
245            DISPLAY "IN UMLAUTE-UMSETZEN"
246
247        *   Initialisieren des Index für die 2. Tabelle:
248            SET TAB-2-IND TO 0
249
250        *   Schleife um die 1. Tabelle:
251            PERFORM VARYING TAB-1-IND FROM 1 BY 1
252                    UNTIL TAB-1-IND > 83
253
254        *       Index der 2. Tabelle erhöhen:
255                SET TAB-2-IND UP BY 1
256        *       Übergabe Inhalte von einer Tabelle zur anderen:
257                MOVE EIN-BYTE(TAB-1-IND) TO AUS-BYTE(TAB-2-IND)
258
259        *       Behandlung großes Ä
260                IF AUS-BYTE(TAB-2-IND) = HEXA-C4
```

```
261                     MOVE HEXA-C3 TO AUS-BYTE(TAB-2-IND)
262                     SET TAB-2-IND UP BY 1
263                     MOVE HEXA-84 TO AUS-BYTE(TAB-2-IND)
264                 END-IF
265
266         *       Behandlung großes Ö
267                 IF AUS-BYTE(TAB-2-IND) = HEXA-D6
268                     MOVE HEXA-C3 TO AUS-BYTE(TAB-2-IND)
269                     SET TAB-2-IND UP BY 1
270                     MOVE HEXA-96 TO AUS-BYTE(TAB-2-IND)
271                 END-IF
272
273         *       Behandlung großes Ü
274                 IF AUS-BYTE(TAB-2-IND) = HEXA-DC
275                     MOVE HEXA-C3 TO AUS-BYTE(TAB-2-IND)
276                     SET TAB-2-IND UP BY 1
277                     MOVE HEXA-9C TO AUS-BYTE(TAB-2-IND)
278                 END-IF
279
280         *       Behandlung kleines ä
281                 IF AUS-BYTE(TAB-2-IND) = HEXA-E4
282                     MOVE HEXA-C3 TO AUS-BYTE(TAB-2-IND)
283                     SET TAB-2-IND UP BY 1
284                     MOVE HEXA-A4 TO AUS-BYTE(TAB-2-IND)
285                 END-IF
286
287         *       Behandlung kleines ö
288                 IF AUS-BYTE(TAB-2-IND) = HEXA-F6
289                     MOVE HEXA-C3 TO AUS-BYTE(TAB-2-IND)
290                     SET TAB-2-IND UP BY 1
291                     MOVE HEXA-B6 TO AUS-BYTE(TAB-2-IND)
292                 END-IF
293
294         *       Behandlung kleines ü
295                 IF AUS-BYTE(TAB-2-IND) = HEXA-FC
296         *           Ersetzen HEX fc durch c3
297                     MOVE HEXA-C3 TO AUS-BYTE(TAB-2-IND)
298         *           Index um 1 erhöhen um in des nächste Feld zu gelangen
299                     SET TAB-2-IND UP BY 1
300         *           Ersetzen leer durch HEX bc
301                     MOVE HEXA-BC TO AUS-BYTE(TAB-2-IND)
302                 END-IF
303
304         *       Behandlung scharfes s (ß)
305                 IF AUS-BYTE(TAB-2-IND) = HEXA-DF
306                     MOVE HEXA-C3 TO AUS-BYTE(TAB-2-IND)
307                     SET TAB-2-IND UP BY 1
308                     MOVE HEXA-9F TO AUS-BYTE(TAB-2-IND)
309                 END-IF
310
311         *    Ende der Schleife um die 1. Tabelle:
312              END-PERFORM
313
314         *    Das Feld RED-AUS-FELD beinhaltet die Werte Tabelle-2,
315         *    daher kann das jetzt an das Feld WS-TEMP-AUSGABE
316         *    übergeben werden:
317              DISPLAY "RED-AUS-FELD = " RED-AUS-FELD
```

```
318             MOVE RED-AUS-FELD TO WS-TEMP-AUSGABE
319         .
320
321         SCHRIEBEN-DATENSATZ SECTION.
322             DISPLAY "IN SCHRIEBEN-DATENSATZ"
323
324             WRITE AUSGABE-FILE FROM TEMP-AUSGABE-ZEILE
325
326             IF AUSGABE-STATUS > ZEROES
327                 DISPLAY "AUSGABE-STATUS = " AUSGABE-STATUS
328                 SET FEHLER-JA TO TRUE
329             END-IF
330
331             INITIALIZE TEMP-AUSGABE-ZEILE
332         .
333         ABSCHLUSSARBEITEN SECTION.
334             DISPLAY "IN ABSCHLUSSARBEITEN"
335             MOVE AUS-KENNZEICHEN-ZU         TO TEMP-AUSGABE-ZEILE
336             PERFORM SCHRIEBEN-DATENSATZ
337
338         .
339         SCHLIESSEN-DATEIEN SECTION.
340             DISPLAY "IN SCHLIESSEN-DATEIEN"
341             CLOSE EINGABE
342             CLOSE AUSGABE
343         .
344         END PROGRAM CSV-NACH-XML.
345
```

Also erstmal noch keine Neuigkeiten. Ich habe beim Kopieren gesehen, dass in Zeile 239 ein „#“ im Kommentarbereich steht, das werde ich als erstes eliminieren – das sollte gar nicht erst auf der Homepage landen, Asche auf mein Haupt.

2.2. Ermitteln Zeitstempel

Zu Programmstart und -ende wollen wir den Aufruf einer Funktion einbauen, die uns jeweils einen Zeitstempel gibt, den wir dann in den ABSCHLUSSARBEITEN auswerten und ausgeben können.

Dazu bietet uns COBOL die Funktion `CURRENT-DATE`, über die wir leider keinen vollwertigen Zeitstempel im Sinne eines Datenbank-Timestamps mit 30 Stellen bekommen, wir haben hier nur 2 Stellen hinter der Sekunde zur Verfügung, also sind wir auf Zenti-Sekunden genau.

Zur Nutzung der Funktion brauchen wir die Deklaration des Zeitstempels:

```
05 WS-CURRENT-DATE.
   10 WS-CURRENT-DATE.
      15 WS-CURRENT-YEAR       PIC  9(4).
      15 WS-CURRENT-MONTH      PIC  9(2).
      15 WS-CURRENT-DAY        PIC  9(2).
   10 WS-CURRENT-TIME.
      15 WS-CURRENT-HOUR       PIC  9(2).
      15 WS-CURRENT-MINUTE     PIC  9(2).
      15 WS-CURRENT-SECOND     PIC  9(2).
      15 WS-CURRENT-MS         PIC  9(2).
   10 WS-DIFF-FROM-GMT         PIC S9(4).
```

Der Code für den Aufruf der Funktion ist

```
MOVE FUNCTION CURRENT-DATE TO WS-CURRENT-DATE
```

Da wir uns Start und Ende merken wollen brauchen wir die Deklaration zweimal:

```
05 WS-CURRENT-DATE-START.
   10 WS-CURRENT-DATE.
      15 WS-CURRENT-YEAR          PIC  9(4).
      15 WS-CURRENT-MONTH         PIC  9(2).
      15 WS-CURRENT-DAY           PIC  9(2).
   10 WS-CURRENT-TIME-START.
      15 WS-CURRENT-HOUR-S        PIC  9(2).
      15 WS-CURRENT-MINUTE-S      PIC  9(2).
      15 WS-CURRENT-SECOND-S      PIC  9(2).
      15 WS-CURRENT-MS-S          PIC  9(2).
   10 WS-DIFF-FROM-GMT            PIC S9(4).

05 WS-CURRENT-DATE-ENDE.
   10 WS-CURRENT-DATE.
      15 WS-CURRENT-YEAR          PIC  9(4).
      15 WS-CURRENT-MONTH         PIC  9(2).
      15 WS-CURRENT-DAY           PIC  9(2).
   10 WS-CURRENT-TIME-ENDE.
      15 WS-CURRENT-HOUR-E        PIC  9(2).
      15 WS-CURRENT-MINUTE-E      PIC  9(2).
      15 WS-CURRENT-SECOND-E      PIC  9(2).
      15 WS-CURRENT-MS-E          PIC  9(2).
   10 WS-DIFF-FROM-GMT            PIC S9(4).
```

Ich höre Euch rufen – „hah, Fehler! Die Felder sind nicht eindeutig! Alle Felder in `WS-CURRENT-DATE` sind doppelt, das geht doch nicht!“. Und ich sage Euch – Ihr habt Recht. Und auch wieder nicht. Klären wir das auf.

Solange Ihr die Felder nicht ansprecht, können sie unterhalb einer übergeordneten Struktur gleich heißen. Da wir uns hier nur mit der Uhrzeit auseinandersetzen, reicht uns auf die Unterscheidung auf der 10er Stufe nach `WS-CURRENT-TIME-START` und `WS-CURRENT-TIME-ENDE`.

Sobald ein Feld auf der 15er Stufe angesprochen werden soll, muss es entweder eindeutig gemacht werden, oder der Zusatz `OF` und die übergeordnete Struktur beinhalten.

Ein Beispiel. Solange wir nichts mit dem Feld `WS-CURRENT-YEAR` machen, müssen wir es nicht unterschiedlich in den beiden Deklarationen für Start und Ende behandeln. Wollen wir es aber ausgeben, sieht das anders aus.

```
DISPLAY "Ende Jahr = " WS-CURRENT-YEAR
```

Das lässt der Compiler nicht zu. Dafür müssten wir eines der Felder `WS-CURRENT-YEAR` ändern. Was also gehen würde wäre `WS-CURRENT-YEAR-ENDE`:

```
…
               10 WS-CURRENT-DATE.
                  15 WS-CURRENT-YEAR-ENDE       PIC  9(4).
…
DISPLAY "Ende Jahr = " WS-CURRENT-YEAR-ENDE
```

Alternativ muss via OF mitgegeben werden, welche übergeordnete Struktur zuständig ist. In unserem Fall ist aber auch die 10er Stufe gleich, also müssen wir die 5er Stufe angeben

```
…
           05 WS-CURRENT-DATE-ENDE.
              10 WS-CURRENT-DATE.
                 15 WS-CURRENT-YEAR            PIC  9(4).
…
DISPLAY "Ende Jahr = " WS-CURRENT-HOUR OF WS-CURRENT-DATE-ENDE
```

Kompliziert und fehleranfällig, besser ist es also, Variablen eindeutig zu machen. Probiert es aus!

Den Aufruf der Funktion für den Start-Zeitstempels packen wir in die INITIALISIEREN SECTION, dort nach der Initialisierung der Variablen:

```
       INITIALISIERUNG SECTION.
…
          INITIALIZE VARIABLEN

          MOVE FUNCTION CURRENT-DATE TO WS-CURRENT-DATE-START
```

Der Rest der Arbeiten findet in der ABSCHLUSSARBEITEN SECTION statt.

Auch den Code nehme ich wieder vorweg, bei mir sieht er so aus:

```
 1         ABSCHLUSSARBEITEN SECTION.
 2             DISPLAY "IN ABSCHLUSSARBEITEN"
 3             MOVE AUS-KENNZEICHEN-ZU         TO TEMP-AUSGABE-ZEILE
 4             PERFORM SCHRIEBEN-DATENSATZ
 5
 6             MOVE FUNCTION CURRENT-DATE TO WS-CURRENT-DATE-ENDE
 7
 8        *    Umrechnung START-Zeit in Sekunden --> LZ-START-IN-SEK
 9             COMPUTE LZ-H-IN-SEK       =   WS-CURRENT-HOUR-S   * 3600
10             COMPUTE LZ-MIN-IN-SEK     =   WS-CURRENT-MINUTE-S * 60
11             COMPUTE LZ-START-IN-SEK   =   LZ-H-IN-SEK
12                                         + LZ-MIN-IN-SEK
13                                         + WS-CURRENT-SECOND-S
14
15        *    Umrechnung ENDE-Zeit in Sekunden --> LZ-ENDE-IN-SEK
16             COMPUTE LZ-H-IN-SEK       =   WS-CURRENT-HOUR-E   * 3600
17             COMPUTE LZ-MIN-IN-SEK     =   WS-CURRENT-MINUTE-E * 60
18             COMPUTE LZ-ENDE-IN-SEK    =   LZ-H-IN-SEK
19                                         + LZ-MIN-IN-SEK
20                                         + WS-CURRENT-SECOND-E
21
22        *    Programmlauf über das Tagesende hinaus?
23             IF WS-CURRENT-HOUR-E < WS-CURRENT-HOUR-S
24                 ADD 86400               TO LZ-ENDE-IN-SEK
25             END-IF
26
27        *    Laufzeit in Sekunden --> LZ-DIFF-IN-SEK
28             COMPUTE LZ-DIFF-IN-SEK =   LZ-ENDE-IN-SEK
29                                      - LZ-START-IN-SEK
30
31        *    Aufbereiten Start-Zeitpunkt
32             MOVE WS-CURRENT-HOUR-S     TO LZ-START-STUNDE
33             MOVE WS-CURRENT-MINUTE-S   TO LZ-START-MINUTE
```

```
34            MOVE WS-CURRENT-SECOND-S   TO LZ-START-SEKUNDE
35            MOVE WS-CURRENT-MS-S       TO LZ-START-ZENTI-SEK
36
37       *    Aufbereiten Ende-Zeitpunkt
38            MOVE WS-CURRENT-HOUR-E     TO LZ-ENDE-STUNDE
39            MOVE WS-CURRENT-MINUTE-E   TO LZ-ENDE-MINUTE
40            MOVE WS-CURRENT-SECOND-E   TO LZ-ENDE-SEKUNDE
41            MOVE WS-CURRENT-MS-E       TO LZ-ENDE-ZENTI-SEK
42
43       *    Aufbereiten Differenz
44       *    Rückrechnen Differenz in Stunden
45            MOVE LZ-DIFF-IN-SEK        TO LZ-REST-IN-SEK
46
47            COMPUTE LZ-PRUEF-STUNDEN = LZ-REST-IN-SEK / 3600
48
49            MOVE LZ-PRUEF-STUNDEN      TO LZ-DIFF-IN-STD
50
51       *    Abziehen der vollen Stunden umgerechnet in Sekunden
52            COMPUTE LZ-REST-IN-SEK  =    LZ-REST-IN-SEK
53                                       - (LZ-DIFF-IN-STD * 3600)
54
55       *    Rückrechnen Differenz in Minuten
56            COMPUTE LZ-PRUEF-MINUTEN = LZ-REST-IN-SEK / 60
57
58            MOVE LZ-PRUEF-MINUTEN    TO LZ-DIFF-IN-MIN
59
60       *    Abziehen der vollen Minuten umgerechnet in Sekunden
61            COMPUTE LZ-REST-IN-SEK  =    LZ-REST-IN-SEK
62                                       - (LZ-DIFF-IN-MIN * 60)
63
64       *    Der Rest sind Sekunden
65            MOVE LZ-REST-IN-SEK     TO LZ-DIFF-IN-SEK
66
67       *    Für die Differenz in Zenti-Sekunden brauchen wir 3 Stellen
68            MOVE WS-CURRENT-MS-S    TO LZ-TEMP-MS-S
69            MOVE WS-CURRENT-MS-E    TO LZ-TEMP-MS-E
70
71            IF WS-CURRENT-MS-E         < WS-CURRENT-MS-S
72                COMPUTE LZ-TEMP-MS-E   = WS-CURRENT-MS-E + 100
73       *        Sekunde um 1 verringern
74                COMPUTE LZ-DIFF-IN-SEK       =  LZ-DIFF-IN-SEK - 1
75            END-IF
76
77            COMPUTE LZ-DIFF-MS    =   LZ-TEMP-MS-E
78                                    - LZ-TEMP-MS-S
79
80       *    Aufbereiten Differenz
81            MOVE LZ-DIFF-IN-STD        TO LZ-DIFF-STUNDE
82            MOVE LZ-DIFF-IN-MIN        TO LZ-DIFF-MINUTE
83                             MOVE LZ-DIFF-IN-SEK        TO LZ-DIFF-SEKUNDE
84            MOVE LZ-DIFF-MS            TO LZ-DIFF-ZENTI-SEK
85
86            DISPLAY "Start                = " LAUFZEIT-START
87            DISPLAY "Ende                 = " LAUFZEIT-ENDE
88            DISPLAY "Laufzeit in Sekunden = " LAUFZEIT-DIFFERENZ
89             .
```

Was passiert hier? Zunächst wird die Funktion zur Ermittlung des Ende-Zeitstempels aufgerufen (Zeile 6). Damit haben wir beide Werte im Zugriff.

Um mit beiden Zeiten arbeiten können, werden sowohl Start- als auch Ende-Zeit in Sekunden umgerechnet. Die Stunden werden mit 3.600 multipliziert, die Minuten mit 60 und in Zeilen 11 und 18 werden die Felder `LZ-START-IN-SEK` bzw `LZ-ENDE-IN-SEK` mit den Summen gefüllt.

Sollten wir über das Tagesende hinausgekommen sein, addieren wir noch die Summe aller Sekunden eines ganzen Tages auf `LZ-ENDE-IN-SEK` (Zeile 24).

In Zeile 28 subtrahieren wir das Ende vom Start und haben dann die Laufzeit in Sekunden im Feld `LZ-DIFF-IN-SEK` gespeichert.

Zeilen 32 bis 41 bringen die Inhalte von Start- und Ende-Zeit in die Ausgabezeilen.

Die Sekunden aus `LZ-DIFF-IN-SEK` müssen wir jetzt wieder zurückrechnen in Stunden, Minuten und Sekunden.

Ich habe mich dazu entschlossen, eine Restwertbetrachtung zu machen. Zuerst wird der Inhalt von `LZ-DIFF-IN-SEK` in das Hilfsfeld `LZ-REST-IN-SEK` kopiert (Zeile 45).

Jetzt werden die vollen Stunden ermittelt, indem der `LZ-REST-IN-SEK` durch 3.600 geteilt wird (Zeile 47). Ein möglicher Rest ist mir dabei nicht wichtig.

Die vollen Stunden werden jetzt wieder mit 3.600 multipliziert und das Ergebnis wird von `LZ-REST-IN-SEK` abgezogen (Zeile 52). Damit bleiben nur noch Minuten und Sekunden umgerechnet in `LZ-REST-IN-SEK` übrig.

Zeilen 56 bis 62 machen wir das Gleiche noch einmal für Minuten, also Division mit 60. In Zeile 61 verbleiben dann im Feld `LZ-REST-IN-SEK` nur noch Sekunden, die wir dann einfach übernehmen.

Für die Nachkommastellen in den Zenti-Sekunden sind wir dann wieder im Hunderter-System, können also einfach das Ende vom Start abziehen.

Allerdings müssen wir bedenken, dass die Zenti-Sekunden vom Start größer sein können, als die vom Ende. Als zum Beispiel `START = 99`, `ENDE = 02`. Um auch da richtig zu rechnen, wird in Zeile 71 geprüft, ob das Ende kleiner ist als der Beginn. Ist das der Fall, wird auf das Ende 100 addiert und der Start kann abgezogen werden.

In unserem Beispiel also `START = 099` und `ENDE = 102`. Dazu müssen wir zwingend ein 3-stelliges Feld haben, sonst klappt das Abziehen nicht. Da wir uns damit an der nächsten Sekunde bedient haben, müssen wir vom Ergebnis `LZ-DIFF-IN-SEK` noch eine anziehen.

Der Rest ist dann die Ausgabe der Werte selbst.

Noch ein Hinweis, wenn Ihr mit unterschiedlichen Datumswerten testen wollt, könnt Ihr den Aufruf der Funktion auskommentieren und eigene Werte in die Variablen geben. Der Aufbau dazu ist:

```
*      MOVE FUNCTION CURRENT-DATE TO WS-CURRENT-DATE-ENDE
       MOVE "2021062723015999+020" TO WS-CURRENT-DATE-START
       MOVE "2021062702590202+020" TO WS-CURRENT-DATE-ENDE
```

Die Deklarationen der benötigten Variablen sind

```
 1            05 WS-CURRENT-DATE-START.
 2               10 WS-CURRENT-DATE.
 3                  15 WS-CURRENT-YEAR         PIC  9(4).
 4                  15 WS-CURRENT-MONTH        PIC  9(2).
 5                  15 WS-CURRENT-DAY          PIC  9(2).
 6               10 WS-CURRENT-TIME-START.
 7                  15 WS-CURRENT-HOUR-S       PIC  9(2).
 8                  15 WS-CURRENT-MINUTE-S     PIC  9(2).
 9                  15 WS-CURRENT-SECOND-S     PIC  9(2).
10                  15 WS-CURRENT-MS-S         PIC  9(2).
11               10 WS-DIFF-FROM-GMT           PIC S9(4).
12
13            05 WS-CURRENT-DATE-ENDE.
14               10 WS-CURRENT-DATE.
15                  15 WS-CURRENT-YEAR         PIC  9(4).
16                  15 WS-CURRENT-MONTH        PIC  9(2).
17                  15 WS-CURRENT-DAY          PIC  9(2).
18               10 WS-CURRENT-TIME-ENDE.
19                  15 WS-CURRENT-HOUR-E       PIC  9(2).
20                  15 WS-CURRENT-MINUTE-E     PIC  9(2).
21                  15 WS-CURRENT-SECOND-E     PIC  9(2).
22                  15 WS-CURRENT-MS-E         PIC  9(2).
23               10 WS-DIFF-FROM-GMT           PIC S9(4).
24
25            05 LAUFZEIT-START.
26               10 LZ-START-STUNDE            PIC 9(2).
27               10 FILLER                     PIC X(1) VALUE ":".
28               10 LZ-START-MINUTE            PIC 9(2).
29               10 FILLER                     PIC X(1) VALUE ":".
30               10 LZ-START-SEKUNDE           PIC 9(2).
31               10 FILLER                     PIC X(1) VALUE ",".
32               10 LZ-START-ZENTI-SEK         PIC 9(2).
33            05 LAUFZEIT-ENDE.
34               10 LZ-ENDE-STUNDE             PIC 9(2).
35               10 FILLER                     PIC X(1) VALUE ":".
36               10 LZ-ENDE-MINUTE             PIC 9(2).
37               10 FILLER                     PIC X(1) VALUE ":".
38               10 LZ-ENDE-SEKUNDE            PIC 9(2).
39               10 FILLER                     PIC X(1) VALUE ",".
40               10 LZ-ENDE-ZENTI-SEK          PIC 9(2).
41            05 LAUFZEIT-DIFFERENZ.
42               10 LZ-DIFF-STUNDE             PIC 9(2).
43               10 FILLER                     PIC X(1) VALUE ":".
44               10 LZ-DIFF-MINUTE             PIC 9(2).
45               10 FILLER                     PIC X(1) VALUE ":".
46               10 LZ-DIFF-SEKUNDE            PIC 9(2).
47               10 FILLER                     PIC X(1) VALUE ",".
48               10 LZ-DIFF-ZENTI-SEK          PIC 9(2).
49
```

```
50            05 LZ-H-IN-SEK                 PIC 9(6).
51            05 LZ-MIN-IN-SEK               PIC 9(6).
52            05 LZ-START-IN-SEK             PIC 9(6).
53            05 LZ-ENDE-IN-SEK              PIC 9(6).
54            05 LZ-DIFF-IN-STD              PIC 9(6).
55            05 LZ-DIFF-IN-MIN              PIC 9(6).
56            05 LZ-DIFF-IN-SEK              PIC 9(6).
57            05 LZ-DIFF-MS                  PIC 9(3).
58            05 LZ-REST-IN-SEK              PIC 9(6).
59            05 LZ-PRUEF-STUNDEN            PIC 9(6).
60            05 LZ-PRUEF-MINUTEN            PIC 9(6).
61            05 LZ-PRUEF-SEKUNDEN           PIC 9(6).
62            05 LZ-TEMP-MS-S                PIC 9(3).
63            05 LZ-TEMP-MS-E                PIC 9(3).
```

Wenn ich das jetzt umwandle und laufen lassen, sieht die Statistik bei mir so aus:

```
Anzahl Datens▒tze    =          716
Start                = 11:35:04,13
Ende                 = 11:35:04,35
Laufzeit in Sekunden = 00:00:00,22
```

Damit haben wir die Basis für eine Messung der Verbesserung geschaffen, sehr gut.

2.3. Vergrößern der Datenbasis

Damit kommen wir zur 2. Sache, die ich oben ansprach. Wir brauchen mehr Futter!

Auf meiner Maschine bin ich nicht in der Lage wirklich vernünftige Programmlaufzeiten zu ermitteln, die 700 Datensätzen sind einfach zu schnell durch. Jetzt geht es also um Quantität, nicht um Qualität.

Dazu kopiere ich die Eingabe (`kennzeichen.csv`) und füge sie unter neuem Namen wieder ein (`kennzeichen-lang.csv`). Die Datensätze darin kopiere ich und füge sie immer und immer wieder ein, bis ich ein ordentliches Set von 100.000 bis 200.000 Daten zusammen habe. Je mehr desto besser.

Damit wir uns dann auch ausgeben lassen können wir viele Datensätze gelesen wurden, müssen wir noch etwas werkeln.

Zunächst brauchen wir 2 Variablen

```
05 ANZAHL-SAETZE                PIC 9(8).
05 ANZAHL-SAETZE-DIPLAY         PIC ZZ.ZZZ.ZZ9.
```

Das Z steht für eine Stelle in einem numerischen wert, die als leer angezeigt wird, wenn sie eine führende Null ist.

Wenn wir das aber so eingeben wird der Compiler sich über das Aussehen von Variable `ANZAHL-SAETZE-DIPLAY` beschweren.

Im Amerikanischen sind die Tausende-Trennpunkte keine Punkte sondern Semikola, und der Punkt steht als Dezimal-Trenner. Also `ZZ,ZZZ,ZZ9.99` für einen numerischen Wert mit 2 Nachkomma-Stellen.

Das unterschiedliche Aussehen können wir dem Programm aber mitteilen, dazu müssen wir ganz nach oben und zwischen `ENVIRONMENT DIVISION.` und der `INPUT-OUTPUT SECTION.` die 3 rot markierten Zeilen einfügen:

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
DECIMAL-POINT IS COMMA.
INPUT-OUTPUT SECTION.
```

Keine Fehlermeldung mehr, richtig? Prima.

Dann gehen wir noch in die LESEN-DATENSATZ SECTION und ändern die READ-Anweisung

```
READ EINGABE INTO WS-EINGABE
   AT END SET EINGABE-ENDE-JA TO TRUE
   NOT AT END ADD 1 TO ANZAHL-SAETZE
END-READ
```

In der ABSCHLUSSARBEITEN SECTION geben wir dann das Ergebnis mit aus. An welcher Stelle ist eigentlich egal, ich habe es ganz nach oben gesetzt:

```
MOVE ANZAHL-SAETZE            TO   ANZAHL-SAETZE-DIPLAY
DISPLAY "Anzahl Datensätze    =  " ANZAHL-SAETZE-DIPLAY
```

Damit wir jetzt auch die vielen Datensätze testen können, müssen wir die SELECT-Anweisung vom Anfang des Programms noch ändern:

```
SELECT EINGABE ASSIGN TO 'kennzeichen-lang.csv'
```

Und schon kann es losgehen. Ich komme bei 100.101 Datensätzen auf 24 Sekunden Laufzeit:

```
Anzahl Datens▒tze    =     100.101
Start                = 08:02:57,01
Ende                 = 08:03:21,63
Laufzeit in Sekunden = 00:00:24,62
```

Damit lässt sich doch was anfangen. Das „ä“ sieht nicht schön aus, aber das ist erstmal okay.

Den Quellcode habe ich jetzt hier nicht mehr komplett eingebunden, Ihr findet Ihn in der zip-Datei mit der Kapitelnummer 2.2 als Zusatz.

3. Verbesserungen

Im vorigen Kapitel haben wir die Grundlage geschaffen, die Wirksamkeit der zeitlichen Verbesserungen messen zu können. In diesem Kapitel gehen wir die Verbesserungen an.

Im ersten Schritt widmen wir uns um die Strukturierung des Codes. Ich denke, hier werden noch keine Effekte messbar sein, denn am Code selbst werde ich nicht viel verbessern. Er wird aber – zumindest für mich – deutlich besser lesbar sein, ich hoffe, Ihr seht das genauso.

Danach werden wir eine Log-Datei einführen und die DISPLAYs entfernen und gegen Schreibzugriffe in das Log austauschen.

3.1. Strukturierung des Codes

Bei der Strukturierung von Code ist die Lesbarkeit ein entscheidender Faktor. Ichhabe es im anderen Projekt schon gesagt, hier hat jedes Unternehmen, teilweise aber auch jede Abteilung eigene Regeln.

Ich stelle jetzt mal ein paar auf und wir setzen die dann exemplarisch in Code-Teilen um, den ´neuen Code nehmt dann bitte aus der Download-Datei.

Kommentare:

Kommentare dienen dem Verständnis. COBOL ist durch seine Struktur gut lesbar, weil die Anweisungen in klaren Worten erfolgen. Dennoch gibt es Stellen, an denen Ihr länger gebraucht habt, bis der Code funktioniert hat. Das sind dann Stellen, die Ihr unbedingt kommentieren solltet.

Baut Kommentar-Kästen um wichtige Stellen. Kommentar-Kästen sind wichtige Stilmittel um besondere Stellen hervorzuheben. Wir haben einen solchen Kasten am Anfang des Programms gesehen:

```
******************************************************************
* Author:    papa
* Date:      Juni 2021
* Purpose:   Konvertieren von Inhalten aus csv nach xml
*            Ausbaustufe Tuningmaßnahmen
*            - Umstrukturien Code
*            -
* Tectonics: cobc
******************************************************************
```

Ich habe mir angewöhnt, vor einer Section einen Kommentar-Kasten zu platzieren, um die Inhalte der Section zu beschreiben. Für die UMLAUTE_UMSETZEN SECTION wäre das

```
******************************************************************
*    UMLAUTE-UMSETZEN SECTION.
*    - iterieren über interne Tabelle TABELLE-1
*    - Prüfung, ob aktuelles Zeichen ein Umlaut ist
*      - wenn ja,
*          - Füllzeichen einfügen
*          - Umlaut transformieren
*          - Rückgabe
******************************************************************
```

Der Kasten ist aber nicht nur eine Quelle für Informationen, sondern gleichzeitig auch ein Trenner zwischen 2 Sections:

```
310           INTO TEMP-AUSGABE-ZEILE
311           PERFORM SCHRIEBEN-DATENSATZ
312
313           MOVE AUS-RECORD_ZU           TO TEMP-AUSGABE-ZEILE
314           PERFORM SCHRIEBEN-DATENSATZ
315           .
316
317       ******************************************************************
318       *     UMLAUTE-UMSETZEN SECTION.
319       *     - iterieren über interne Tabelle TABELLE-1
320       *     - Prüfung, ob aktuelles Zeichen ein Umlaut ist
321       *       - wenn ja,
322       *           - Füllzeichen einfügen
323       *           - Umlaut transformieren
324       *           - Rückgabe
325       ******************************************************************
326        UMLAUTE-UMSETZEN SECTION.
327           DISPLAY "IN UMLAUTE-UMSETZEN"
328
329       *     Initialisieren des Index für die 2. Tabelle:
330           SET TAB-2-IND TO 0
331
332       *     Schleife um die 1. Tabelle:
333           PERFORM VARYING TAB-1-IND FROM 1 BY 1
```

Das Prinzip der Trennung oder des Blickfangs können wir auch zur Wiedererkennung von wichtigen Abschnitten im Code nutzen. Am Beispiel der PROCEDURE DIVISION, statt

```
199        01  RED-AUS-FELD                PIC X(83).
200        01  TABELLE-2 REDEFINES RED-AUS-FELD
201                    OCCURS 83 TIMES INDEXED BY TAB-2-IND.
202            05 AUS-BYTE                  PIC X(1).
203
204        PROCEDURE DIVISION.
205
206       *Beginn der Steuerung
207        STEUER SECTION.
208            PERFORM INITIALISIERUNG
209            PERFORM OEFFNEN-DATEIEN
210            PERFORM LESEN-DATENSATZ
211            PERFORM WITH TEST BEFORE UNTIL EINGABE-ENDE-JA OR FEHLER-JA
212                PERFORM KONVERTIERE-DATENSATZ
213                PERFORM LESEN-DATENSATZ
```

eine Kommentarzeile vorher und nachher ist für mich eine optische Verbesserung:

```
199        01  RED-AUS-FELD                PIC X(83).
200        01  TABELLE-2 REDEFINES RED-AUS-FELD
201                    OCCURS 83 TIMES INDEXED BY TAB-2-IND.
202            05 AUS-BYTE                  PIC X(1).
203
204       ******************************************************************
205        PROCEDURE DIVISION.
206       ******************************************************************
207
208       *Beginn der Steuerung
209        STEUER SECTION.
210            PERFORM INITIALISIERUNG
211            PERFORM OEFFNEN-DATEIEN
212            PERFORM LESEN-DATENSATZ
213            PERFORM WITH TEST BEFORE UNTIL EINGABE-ENDE-JA OR FEHLER-JA
214                PERFORM KONVERTIERE-DATENSATZ
215                PERFORM LESEN-DATENSATZ
```

Eine weitere optische Verbesserung sind gleiche Abstände in den Deklarationen und den Sections. Was meine ich damit? Nehmen wir zwei Beispiele. Einmal die Deklarationen, mit Bild 1:

```
53      *Definition der Variablen
54      01 VARIABLEN.
55          05 EINGABE-STATUS        PIC 9(2).
56          05 AUSGABE-STATUS        PIC 9(2).
57          05 WS-EINGABE            PIC X(130).
58          05 TEMP-AUSGABE-ZEILE    PIC X(130).
59          05 TEMP-LOG              PIC X(130).
60          05 WS-INHALT-ABK         PIC X(3).
61          05 WS-INHALT-STADT       PIC X(82).
62          05 WS-INHALT-ABGEL       PIC X(82).
63          05 WS-INHALT-BULAND      PIC X(82).
64          05 WS-TEMP-AUSGABE       PIC X(82).
65          05 WS-CURRENT-DATE-START.
66             10 WS-CURRENT-DATE.
67                15 WS-CURRENT-YEAR         PIC  9(4).
68                15 WS-CURRENT-MONTH        PIC  9(2).
69                15 WS-CURRENT-DAY          PIC  9(2).
70             10 WS-CURRENT-TIME-START.
71                15 WS-CURRENT-HOUR-S       PIC  9(2).
72                15 WS-CURRENT-MINUTE-S     PIC  9(2).
73                15 WS-CURRENT-SECOND-S     PIC  9(2).
74                15 WS-CURRENT-MS-S         PIC  9(2).
75             10 WS-DIFF-FROM-GMT           PIC S9(4).
```

Bild 2 mit der Vereinheitlichung:

```
53      *Definition der Variablen
54      01 VARIABLEN.
55          05 EINGABE-STATUS                PIC 9(02).
56          05 AUSGABE-STATUS                PIC 9(02).
57          05 WS-EINGABE                    PIC X(130).
58          05 TEMP-AUSGABE-ZEILE            PIC X(130).
59          05 TEMP-LOG                      PIC X(130).
60          05 WS-INHALT-ABK                 PIC X(03).
61          05 WS-INHALT-STADT               PIC X(82).
62          05 WS-INHALT-ABGEL               PIC X(82).
63          05 WS-INHALT-BULAND              PIC X(82).
64          05 WS-TEMP-AUSGABE               PIC X(82).
65          05 WS-CURRENT-DATE-START.
66             10 WS-CURRENT-DATE.
67                15 WS-CURRENT-YEAR         PIC 9(04).
68                15 WS-CURRENT-MONTH        PIC 9(02).
69                15 WS-CURRENT-DAY          PIC 9(02).
70             10 WS-CURRENT-TIME-START.
71                15 WS-CURRENT-HOUR-S       PIC 9(02).
72                15 WS-CURRENT-MINUTE-S     PIC 9(02).
73                15 WS-CURRENT-SECOND-S     PIC 9(02).
74                15 WS-CURRENT-MS-S         PIC 9(02).
75             10 WS-DIFF-FROM-GMT           PIC S9(04).
```

Das zweite Bild erscheint mir persönlich etwas ruhiger. Und auch bei der Strukturierung des Codes können wir ruhigere Bilder erzeugen. Als Beispiel hier Bild 1:

```
228     *    Setzen der Schalter auf Anfangszustand
229          SET EINGABE-ENDE-NEIN TO TRUE
230          SET FEHLER-NEIN TO TRUE
231          SET ERSTER-SATZ-JA TO TRUE
```

Daneben direkt Bild 2:

```
228     *    Setzen der Schalter auf Anfangszustand
229          SET EINGABE-ENDE-NEIN      TO TRUE
230          SET FEHLER-NEIN            TO TRUE
231          SET ERSTER-SATZ-JA         TO TRUE
```

Für den nächsten Block an Änderungen muss ich etwas ausholen.

Unsere Sections sind nicht wirklich geordnet. Wenn wir eine neue Section brauchen, benennen wir diese und bauen sie irgendwo ein. Das ist nicht wirklich übersichtlich und folgt keiner Reihenfolge.

Auch hierfür gibt es in den meisten Unternehmen Regeln, wo welche Section zu postieren ist und wie die Namenskonventionen dafür sind. Ich stelle hier mal ein paar Regeln vor, Ihr könnt selber entscheiden, was Ihr davon annehmen wollt. Für den Code selbst hat das aber keine Relevanz.

Schauen wir uns die Programmstruktur an, wir haben das ja in der STEUER SECTION gebündelt:

```
*Beginn der Steuerung
 STEUER SECTION.
     PERFORM INITIALISIERUNG
     PERFORM OEFFNEN-DATEIEN
     PERFORM LESEN-DATENSATZ
     PERFORM WITH TEST BEFORE UNTIL EINGABE-ENDE-JA OR FEHLER-JA
         PERFORM KONVERTIERE-DATENSATZ
         PERFORM LESEN-DATENSATZ
     END-PERFORM
     PERFORM ABSCHLUSSARBEITEN
     PERFORM SCHLIESSEN-DATEIEN
     STOP RUN.
```

Wenn wir das mal generalisierter betrachten, haben wir ein paar Sections für die **Vorbereitung** (INITIALISIERN, OEFFNEN-DATEIEN), dann haben wir die eigentliche **Verarbeitung** (LESEN-DATENSATZ, in die Schleife gehen, KONVERTIEREN-DATENSATZ, LESEN-DATENSATZ) und wenn alles abgearbeitet ist, oder es ist ein Fehler aufgetreten, dann die **Nacharbeiten** (ABSCHLUSSARBEITEN, SCHLIESSEN-DATEIEN).

Dies ist eine gängige Struktur in einem COBOL-Programm, egal welche Arbeiten es zu erledigen hat.

Aus diesem Grund lässt man die Sections gerne mit einem Buchstaben beginnen, gefolgt von einer 3- oder 4-stelligen Zahl. Der Buchstabe soll die Gruppe anzeigen, zu der die Section gehört. Zum Beispiel könnten alle Vorarbeiten mit „A“ anfangen, die Verarbeitung eventuell mit „V“ und die Nacharbeiten mit „Z“. Das Lesen und Schreiben von Dateien hat eine Sonderstellung, hier könnte man den Buchstaben „H“ für Hilfs-Sections definieren.

Die Nummerierung könnte anzeigen, auf welcher Ebene die Section aufgerufen wird. So wäre eine V100 die erste Section in der Gruppe „Verarbeitung“, die V200 die nächste. Soll in der V200 eine weitere Section aufgerufen werden, wäre das die V210 und so weiter.

Je größer ein Programm ist, desto eher kommt man mit der Auswahl der Nummerierungen in Bedrängnis. Aus eigener Erfahrung kann ich sagen, dass die Umbenennung eines Moduls mit mehreren tausend Zeilen Code kein Spaß ist.

Ich nehme mal den Vorschlag von oben an und nummeriere die Sections neu. Dabei kommt dann folgendes heraus:

```
      ****************************************************************
      *S001-STEUER SECTION
      *    - Programmsteuerung
      ****************************************************************
       S100-STEUER SECTION.
           PERFORM A100-INITIALISIERUNG
           PERFORM A200-OEFFNEN-DATEIEN
           PERFORM V100-VERABEITUNG
           PERFORM Z100-ABSCHLUSSARBEITEN
           PERFORM H300-SCHLIESSEN-DATEIEN
           STOP RUN.
```

Die Schleife um die Inhalte der Eingabe wandert in die V100:

```
      ****************************************************************
      *V100-VERARBEITUNG SECTION.
      *    - Schleife um das Lesen der csv-Daten
      ****************************************************************
       V100-VERARBEITUNG SECTION.
…
           PERFORM H100-LESEN-DATENSATZ
           PERFORM WITH TEST BEFORE UNTIL EINGABE-ENDE-JA OR FEHLER-JA
               PERFORM V110-KONVERTIERE-DATENSATZ
               PERFORM H100-LESEN-DATENSATZ
           END-PERFORM
       .
```

Und wo liegt jetzt der Vorteil? Stellen wir uns vor, wir sind Entwickler in einem Unternehmen und hätten die Aufgabe, eine Erweiterung oder Fehlerbereinigung an dem Programm vorzunehmen.

Egal welche Programme ich vorher bearbeitet habe, ich weiß zum Beispiel, dass mit der V100 immer die Verarbeitung beginnt und finde mich relativ schnell zurecht. Sollte aber der Fehler in der Behandlung der Eingabe sein, würde ich zuerst die A-Sections überprüfen.

Damit haben wir alles zu Kommentaren, Strukturierung und Benamung der Sections gesagt, ich habe den bisher erstellen Code entsprechend umgebaut, den Code dazu gibt es im Download-Bereich.

Damit wir allerdings unseren Fortschritt sehen, hier die entsprechenden Laufzeit-Angaben für den aktuellen Lauf:

```
Anzahl Datens▒tze    =     100.101
Start                = 07:55:07,82
Ende                 = 07:55:31,91
Laufzeit in Sekunden = 00:00:24,09
```

Okay, damit haben wir noch keine Verringerung der Laufzeit bewirkt.

3.2. Einführen einer Log-Datei

Als erste echte Laufzeitverbesserung bauen wir die DISPLAY-Anweisungen um. Die Ausgabe auf die Konsole kostet richtig Zeit, das werden wir gleich sehen. Auch sind Displays nur von kurzer Dauer, sobald das Programm neu gestartet wird, sind die Informationen nicht mehr vorhanden.

Daher erstellen wir uns jetzt eine Log-Datei. Hier legen wir alle für uns wichtigen Informationen ab, genau wie die xml-Datei steht sie uns nach Programmende zur Verfügung. Einen kleinen Haken hat die Log-Datei allerdings. Sie steht uns nicht sofort bei Programmstart zur Verfügung, wir müssen sie zuerst noch öffnen. Da wir das erst in A200 machen, können wir Dinge aus A100 nicht in das Log schreiben, da müssten wir dann auf Displays ausweichen.

Wir könnten die beiden A-Sections umdrehen und zuerst die Dateien öffnen lassen, allerdings wären dann die genutzten Variablen und Schalter nicht initialisiert, auch nicht schön. Ich bleibe bei der Reihenfolge, sie ist mir einfach vertraut. Wenn Ihr das anders machen wollt – nur Mut, es gibt da kein „richtig oder falsch“.

Im Zuge dieser Umbaumaßnahmen, richten wir auch gleich noch einen „Analyse-“ oder „Debug-Schalter“ ein. Damit unterscheiden wir, ob wir uns in der Analyse-Phase befinden, in der wir im Test ganz viele Informationen brauchen, wo sich das Programm gerade befindet oder was der konkrete Werte einer Variablen ist, eben Displays. Oder sind wir in der Phase, wo das Programm nach unserer Meinung völlig korrekt arbeitet und zwischendurch keine Lebenszeichen abgeben muss. Also der „Echtbetrieb“.

Da wir nach den Umbaumaßnahmen bei über 800 Code-Zeilen sind, habe ich nachfolgend den Code aufgeführt, zerhacke ihn aber an Stellen, die ich kommentieren will. Nehmt Euch den Quellcode aus der zip-Datei und verfolgt parallel was passiert.

Der Anfang ist nicht besonders spannend:

```
26 …
27         SELECT LOG            ASSIGN TO 'log.txt'
28                ORGANIZATION   IS LINE SEQUENTIAL
29                FILE STATUS    IS AUSGABE-STATUS.
30 …
31 …
44         FD LOG.
45         01 LOG-FILE.
46             05 LOG-ZEILE                    PIC X(130).
47 …
66 …
67         01 PROGRAMMSTART                    PIC X(01).
68             88 PGM-START-JA                     VALUE "J".
69             88 PGM-START-NEIN                   VALUE "N".
70
71         01 PROGRAMMENDE                     PIC X(01).
72             88 PGM-ENDE-JA                      VALUE "J".
73             88 PGM-ENDE-NEIN                    VALUE "N".
74
75         01 ANALYSE-SCHALTER                 PIC X(01).
76             88 ANALYSE-JA                       VALUE "J".
```

```
 77            88 ANALYSE-NEIN                     VALUE "N".
 78 …
 83 …
 84            05 LOG-STATUS                    PIC 9(02).
```

Die Section-Namen werden als Konstanten für die Ausgabe im Log gespeichert.

```
178 …
179       ****************************************************************
180       *    Section-Namen zur Ausgabe ins Log
181       ****************************************************************
182            05 KONST-V100                    PIC X(20) VALUE
183               "in V100-VERARBEITUNG".
184            05 KONST-V110                    PIC X(29) VALUE
185               "in V110-KONVERTIERE-DATENSATZ".
186            05 KONST-V111                    PIC X(24) VALUE
187               "in V111-UMLAUTE-UMSETZEN".
188            05 KONST-H100                    PIC X(24) VALUE
189               "in H100-LESEN-DATENSATZ".
190            05 KONST-H200                    PIC X(27) VALUE
191               "in H200-SCHREIBEN-DATENSATZ".
192            05 KONST-Z100                    PIC X(25) VALUE
193               "in Z100-ABSCHLUSSARBEITEN".
194       ****************************************************************
```

Bei den Definitionen für die einzelnen Log-Zeilen gibt es auch nichts anzumerken. Jede 05er Stufe entspricht eine Log-Zeile.

```
232
233       ****************************************************************
234       *    Log-Definitionen
235       ****************************************************************
236            05 LOG-HEADER.
237               10 FILLER                     PIC X(10) VALUE ALL "*".
238               10 FILLER                     PIC X(17) VALUE ALL " ".
239               10 FILLER                     PIC X(26) VALUE
240                  "Programmstart CSV-NACH-XML".
241               10 FILLER                     PIC X(17) VALUE ALL " ".
242               10 FILLER                     PIC X(10) VALUE ALL "*".
243            05 LOG-ENDE.
244               10 FILLER                     PIC X(10) VALUE ALL "*".
245               10 FILLER                     PIC X(17) VALUE ALL " ".
246               10 FILLER                     PIC X(26) VALUE
247                  "Programmende CSV-NACH-XML ".
248               10 FILLER                     PIC X(17) VALUE ALL " ".
249               10 FILLER                     PIC X(10) VALUE ALL "*".
250            05 LOG-LAUFZEIT-START.
251               10 FILLER                     PIC X(10) VALUE ALL "*".
252               10 FILLER                     PIC X(17) VALUE ALL " ".
253               10 FILLER                     PIC X(14) VALUE
254                  "Start um      ".
255               10 LZ-START-STUNDE            PIC 9(02).
256               10 FILLER                     PIC X(01) VALUE ":".
257               10 LZ-START-MINUTE            PIC 9(02).
258               10 FILLER                     PIC X(01) VALUE ":".
259               10 LZ-START-SEKUNDE           PIC 9(02).
260               10 FILLER                     PIC X(01) VALUE ",".
```

```
261              10 LZ-START-ZENTI-SEK         PIC 9(02).
262              10 FILLER                     PIC X(18) VALUE ALL " ".
263              10 FILLER                     PIC X(10) VALUE ALL "*".
264           05 LOG-LAUFZEIT-ENDE.
265              10 FILLER                     PIC X(10) VALUE ALL "*".
266              10 FILLER                     PIC X(17) VALUE ALL " ".
267              10 FILLER                     PIC X(14) VALUE
268                 "Ende um       ".
269              10 LZ-ENDE-STUNDE             PIC 9(02).
270              10 FILLER                     PIC X(01) VALUE ":".
271              10 LZ-ENDE-MINUTE             PIC 9(02).
272              10 FILLER                     PIC X(01) VALUE ":".
273              10 LZ-ENDE-SEKUNDE            PIC 9(02).
274              10 FILLER                     PIC X(01) VALUE ",".
275              10 LZ-ENDE-ZENTI-SEK          PIC 9(02).
276              10 FILLER                     PIC X(18) VALUE ALL " ".
277              10 FILLER                     PIC X(10) VALUE ALL "*".
278           05 LOG-LAUFZEIT-DIFFERENZ.
279              10 FILLER                     PIC X(10) VALUE ALL "*".
280              10 FILLER                     PIC X(17) VALUE ALL " ".
281              10 FILLER                     PIC X(14) VALUE
282                "Laufzeit      ".
283              10 LZ-DIFF-STUNDE             PIC 9(02).
284              10 FILLER                     PIC X(01) VALUE ":".
285              10 LZ-DIFF-MINUTE             PIC 9(02).
286              10 FILLER                     PIC X(01) VALUE ":".
287              10 LZ-DIFF-SEKUNDE            PIC 9(02).
288              10 FILLER                     PIC X(01) VALUE ",".
289              10 LZ-DIFF-ZENTI-SEK          PIC 9(02).
290              10 FILLER                     PIC X(18) VALUE ALL " ".
291              10 FILLER                     PIC X(10) VALUE ALL "*".
292           05 LOG-ANZAHL-SAETZE.
293              10 FILLER                     PIC X(10) VALUE ALL "*".
294              10 FILLER                     PIC X(17) VALUE ALL " ".
295              10 FILLER                     PIC X(15) VALUE
296                 "gelesen       ".
297              10 LOG-SAETZE-DIPLAY          PIC ZZ.ZZZ.ZZ9.
298              10 FILLER                     PIC X(18) VALUE ALL " ".
299              10 FILLER                     PIC X(10) VALUE ALL "*".
300
301           05 LOG-LEERZEILE                 PIC X(80) VALUE ALL " ".
302           05 LOG-TRENNZEILE                PIC X(80) VALUE ALL "*".
303           05 LOG-KOMMENTARZEILE.
304              10 FILLER                     PIC X(10) VALUE ALL "*".
305              10 FILLER                     PIC X(60) VALUE ALL " ".
306              10 FILLER                     PIC X(10) VALUE ALL "*".
307       ****************************************************************
```

Die 3 Schalter auf Startposition setzen:

```
347        A100-INITIALISIERUNG SECTION.
357 …
358             SET PGM-START-JA                 TO TRUE
359             SET PGM-ENDE-NEIN                TO TRUE
360
361        *    Sollen DISPLAYS ausgegeben werden, muss der ANALYSE-Schalter
362        *    auf ANALYSE-JA gesetzt werden, falls nicht, ANALYSE-NEIN
363             SET ANALYSE-JA                   TO TRUE
364 …
```

Mit OPEN OUTPUT wird eine neue Datei angelegt oder eine bestehende ersetzt. Mit OPEN EXTEND bleibt der bisher erstellte Inhalt bestehenden, neue Einträge werden hinten angefügt. In andeeren Sprachen ist das der „append“-Befehl.

```
369         A200-OEFFNEN-DATEIEN SECTION.
370 …
371             OPEN OUTPUT AUSGABE, LOG
372
373        *    wenn die Logdatei fortgeschrieben werden soll, muss sie mit
374        *    dem Zusatz EXTEND geöffnet werden:
375        *****OPEN EXTEND LOG
```

Mit dem Analyse-Schalter haben wir jetzt die Möglichkeit Log-Einträge nur dann zu schreiben, wenn wir auch im Analyse-Modus unterwegs sind. Die Abfrage ist dann extrem simpel:

```
388         H100-LESEN-DATENSATZ SECTION.
389
390             IF ANALYSE-JA
391                MOVE KONST-H100               TO TEMP-LOG
392                PERFORM H400-SCHREIBEN-LOG
393             END-IF
394
```

Das bauen wir in alle Sections ein, in denen wir Displays hatten. Aber aufpassen, wir hatten auch in den A-Sections Displays, solange die Datei in A200 noch nicht geöffnet wurde, gibt der Versuch des Schreibens einen Fehler! Also in die A-Sections noch keinen Aufruf des Logs einbauen.

Das Schließen der Datei ist auch klar:

```
433         H300-SCHLIESSEN-DATEIEN SECTION.
434
435             CLOSE EINGABE, AUSGABE, LOG
```

Kurz noch zur Z100, auch hier bauen wir die Abfrage nach dem Analyse-Schater ein.

```
699         Z100-ABSCHLUSSARBEITEN SECTION.
700
701             IF ANALYSE-JA
702                MOVE KONST-Z100               TO TEMP-LOG
703                PERFORM H400-SCHREIBEN-LOG
704             END-IF
705
```

Am Ende ersetzen wir die Displays durch die MOVE-Anweisungen zur Befüllung der Log-Inhalte und rufen die H400 auf

```
788        *    Aufbereiten Differenz
789             SET  PGM-ENDE-JA                TO TRUE
790             MOVE LZ-DIFF-IN-STD             TO LZ-DIFF-STUNDE
791             MOVE LZ-DIFF-IN-MIN             TO LZ-DIFF-MINUTE
792             MOVE LZ-DIFF-IN-SEK             TO LZ-DIFF-SEKUNDE
793             MOVE LZ-DIFF-MS                 TO LZ-DIFF-ZENTI-SEK
794
795             PERFORM H400-SCHREIBEN-LOG
796         .
```

Damit bleibt nur noch die neue Section `H400-SCHREIBEN-LOG` zu besprechen.

In der A100 wurden die Schalter `PGM-START-JA` und `PGM-ENDE-NEIN` gesetzt.

Wann immer der erste Aufruf der H400 erfolgt (bei `ANALYSE-JA` ist der erste Aufruf aus V100, bei `ANALYSE-NEIN` erst in Z100, in obiger Tabelle Zeile 795) ist der Programmstart aktiv. Damit werden die Kopfzeilen des Log in Zeilen 448 bis 452 geschrieben. Der Schalter wird dann auf `PGM-START-NEIN` gesetzt, sodass diese Zeilen nicht mehr durchlaufen werden.

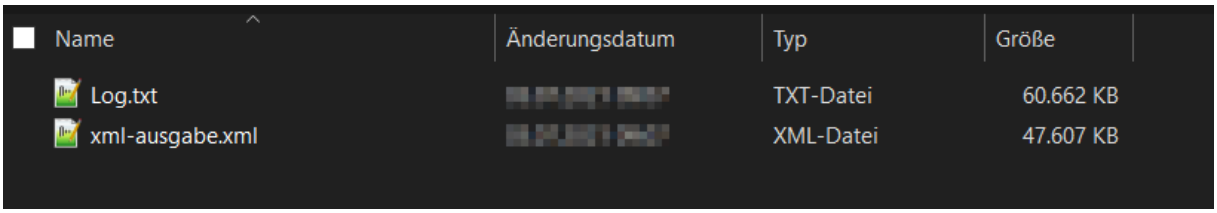
Der Schalter `PGM-ENDE-JA` wird erst in der Z100 (in der obigen Tabelle Zeile 798) gesetzt. Damit ist sichergestellt, dass die Ausgabe der letzten Zeilen erst erfolgen, wenn die Aufbereitung in Z100 stattgefunden hat.

Wann immer also H400 aufgerufen wird, nur das erste mal werden die Kopfzeilen ausgegeben und beim letzten Mal die Laufzeitinformationen. In allen anderen Fällen wir nur die Zeile 465 zum Schreiben des Log-Eintrags aufgerufen.

```
437        ****************************************************************
438        *H400-SCHREIBEN-LOG SECTION.
439        *    - Schreiben des Logs
440        *    - Bei Programmstart Ausgabe der Überschrift
441        *    - Bei Programmende Ausgabe der Laufstatistik
442        *    - Rücksetzen Feld TEMP-LOG
443        ****************************************************************
444         H400-SCHREIBEN-LOG SECTION.
445
446        *    Programmstart
447             IF PGM-START-JA
448                 WRITE LOG-FILE             FROM LOG-TRENNZEILE
449                 WRITE LOG-FILE             FROM LOG-KOMMENTARZEILE
450                 WRITE LOG-FILE             FROM LOG-HEADER
451                 WRITE LOG-FILE             FROM LOG-KOMMENTARZEILE
452                 WRITE LOG-FILE             FROM LOG-TRENNZEILE
453
454                 SET PGM-START-NEIN           TO TRUE
455             END-IF
456
457             IF LOG-STATUS                     > ZEROES
458                 DISPLAY "LOG-STATUS = "         LOG-STATUS
459                 SET FEHLER-JA              TO TRUE
460             END-IF
461
```

```
462        *    Programmende
463             IF PGM-ENDE-NEIN
464        *        Schreiben Log
465                  WRITE LOG-FILE             FROM TEMP-LOG
466                  IF LOG-STATUS                 > ZEROES
467                     DISPLAY "LOG-STATUS = "      LOG-STATUS
468                     SET FEHLER-JA             TO TRUE
469                  END-IF
470             ELSE
471                  MOVE WS-CURRENT-HOUR-E       TO LZ-ENDE-STUNDE
472                  MOVE WS-CURRENT-MINUTE-E     TO LZ-ENDE-MINUTE
473                  MOVE WS-CURRENT-SECOND-E     TO LZ-ENDE-SEKUNDE
474                  MOVE WS-CURRENT-MS-E         TO LZ-ENDE-ZENTI-SEK
475
476                  MOVE ANZAHL-SAETZE           TO LOG-SAETZE-DIPLAY
477
478                  WRITE LOG-FILE             FROM LOG-TRENNZEILE
479                  WRITE LOG-FILE             FROM LOG-KOMMENTARZEILE
480                  WRITE LOG-FILE             FROM LOG-ENDE
481                  WRITE LOG-FILE             FROM LOG-KOMMENTARZEILE
482                  WRITE LOG-FILE             FROM LOG-TRENNZEILE
483                  WRITE LOG-FILE             FROM LOG-KOMMENTARZEILE
484                  WRITE LOG-FILE             FROM LOG-LAUFZEIT-START
485                  WRITE LOG-FILE             FROM LOG-KOMMENTARZEILE
486                  WRITE LOG-FILE             FROM LOG-LAUFZEIT-ENDE
487                  WRITE LOG-FILE             FROM LOG-KOMMENTARZEILE
488                  WRITE LOG-FILE             FROM LOG-LAUFZEIT-DIFFERENZ
489                  WRITE LOG-FILE             FROM LOG-KOMMENTARZEILE
490                  WRITE LOG-FILE             FROM LOG-ANZAHL-SAETZE
491                  WRITE LOG-FILE             FROM LOG-KOMMENTARZEILE
492                  WRITE LOG-FILE             FROM LOG-TRENNZEILE
493
494             END-IF
495             INITIALIZE TEMP-LOG
496          .
```

Das war es hier. Wenn wir das jetzt laufen lassen, sollte im Ordner in dem der Quellcode ist, jetzt die Log-Datei auftauchen:

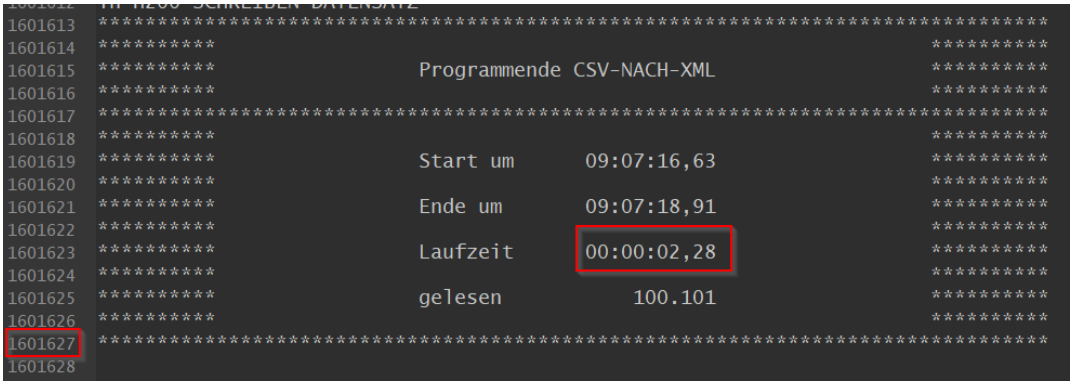


Und jetzt würde ich gerne Eure Gesichter sehen, wenn wir in das Log schauen. Bitte ganz nach unten blättern.

Zur Erinnerung, bis jetzt steht die Marke bei gut 100k eingelesenen Datensätzen bei 24 Sekunden:

```
Anzahl Datens▒tze     =     100.101
Start                 = 07:55:07,82
Ende                  = 07:55:31,91
Laufzeit in Sekunden  = 00:00:24,09
```

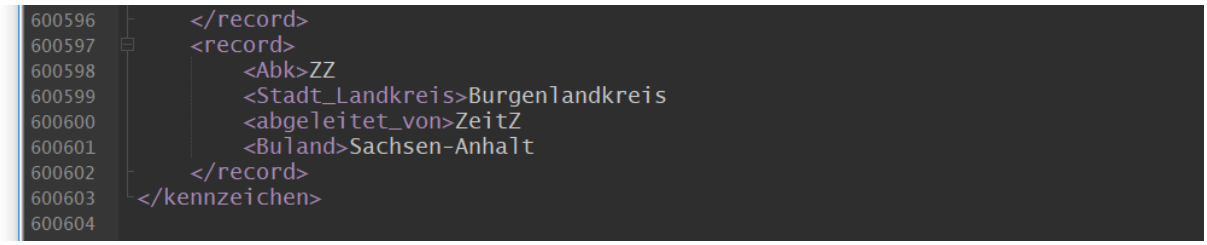
Mit `ANALYSE-JA` komme ich auf 2,28 Sekunden:



Bitte schaut auch mal auf die linke Seite, in nicht mal 3 Sekunden wurden 1,6 Millionen (!) Log-Einträge geschrieben, wie krass ist das denn bitte?

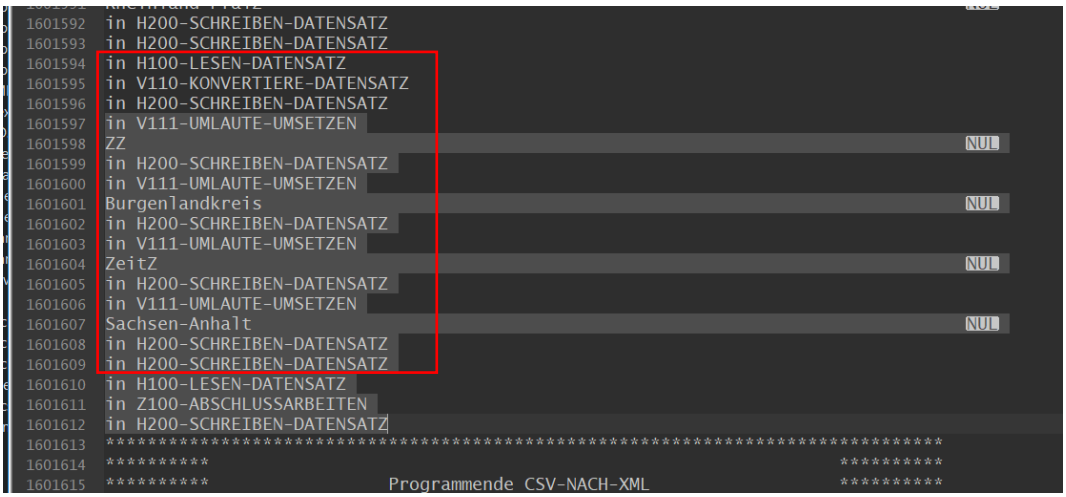
Haben wir da einen Programmfehler? Sind die Zahlen plausibel?

Überschlagen wir das kurz, wir haben 100k Eingabe. Was sagt die xml-Datei?



600k xml, das passt. Aus jedem csv-Satz werden 6 Einträge in der xml-Datei.

Schauen wir uns das Log für den letzten csv-Satz an:



16 Einträge für einen Datensatz, 16 * 100k = 1.600k = 1,6 Mio - auch das passt.

Und wenn wir das Schreiben des Logs auch abklemmen? Also `ANALYSE-NEIN`?

```
********************************************************************************
**********                                                          **********
**********                  Programmstart CSV-NACH-XML              **********
**********                                                          **********
********************************************************************************
********************************************************************************
**********                                                          **********
**********                  Programmende CSV-NACH-XML               **********
**********                                                          **********
********************************************************************************
**********                                                          **********
**********                  Start um      07:42:25,02               **********
**********                                                          **********
**********                  Ende um       07:42:26,22               **********
**********                                                          **********
**********                  Laufzeit      00:00:01,20               **********
**********                                                          **********
**********                  gelesen           100.101               **********
**********                                                          **********
********************************************************************************
```

1,2 Sekunden – Wahnsinn! Vor dieser Leistung habe ich Respekt!

Wenn wir das mit unserer originalen csv-Datei mit 716 Datensätze machen brauchen wir 0,05 Sekunden. Ohne Worte.

```
********************************************************************************
**********                                                          **********
**********                  Programmstart CSV-NACH-XML              **********
**********                                                          **********
********************************************************************************
********************************************************************************
**********                                                          **********
**********                  Programmende CSV-NACH-XML               **********
**********                                                          **********
********************************************************************************
**********                                                          **********
**********                  Start um      07:45:58,97               **********
**********                                                          **********
**********                  Ende um       07:45:59,02               **********
**********                                                          **********
**********                  Laufzeit      00:00:00,05               **********
**********                                                          **********
**********                  gelesen               716               **********
**********                                                          **********
********************************************************************************
```

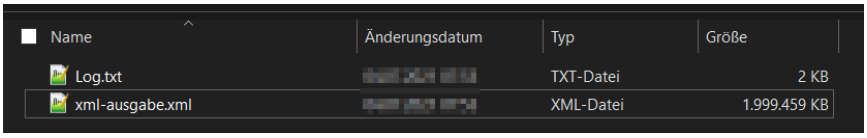
Noch eine Steigerung, ich habe noch eine csv-Datei mit über 4 Mio. Einträgen:

```
Y;Bundeswehr;Y war zur Gründung noch frei;bundesweit
Z;Landkreis Zwickau;Zwickau;Sachsen
ZE;Landkreis Anhalt-Bitterfeld;ZErbst;Sachsen-Anhalt
ZEL;Landkreis Cochem-Zell;ZELl;Rheinland-Pfalz
ZI;Landkreis Görlitz;ZIttau;Sachsen
ZIG;Schwalm-Eder-Kreis;ZIeGenhain;Hessen
ZP;Erzgebirgskreis;ZschoPau;Sachsen
ZR;Landkreis Greiz;ZeulenRoda;Thüringen
ZW;Stadt Zweibrücken und Landkreis Südwestpfalz;ZWeibrücken;Rheinland-Pfalz
ZZ;Burgenlandkreis;ZeitZ;Sachsen-Anhalt
```

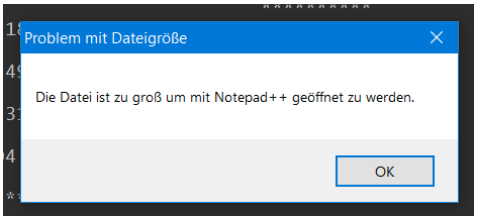
Damit ist das Programm jetzt eine ganze Weile beschäftigt:

```
**********                                                          **********
**********                 Programmstart CSV-NACH-XML               **********
**********                                                          **********
**********                 Programmende CSV-NACH-XML                **********
**********                                                          **********
**********                 Start um      07:53:18,23                **********
**********                 Ende um       07:53:49,80                **********
**********                 Laufzeit      00:00:31,57                **********
**********                 gelesen         4.204.201                **********
**********                                                          **********
```

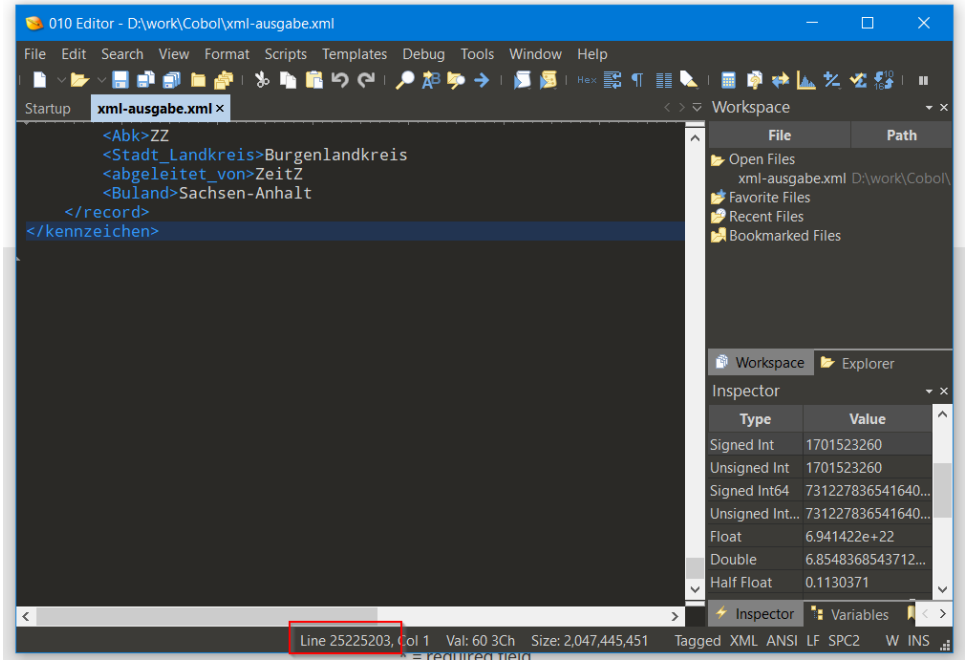
Dabei wurden fast 2 GB an xml-Datei erzeugt:



Das ist zu groß für Notepad++:



Die Anzeige mit dem 010 Editor ergibt stattliche 25.225.203 Zeilen:



Zufällig ist mein Notebook leer gewesen und ich habe den 700er Test mit angeschlossenem Stromkabel gemacht. Ergebnis 0,01 Sekunden:

```
**********************************************************************************
**********                                                              **********
**********                  Programmstart CSV-NACH-XML                  **********
**********                                                              **********
**********************************************************************************
**********************************************************************************
**********                                                              **********
**********                  Programmende CSV-NACH-XML                   **********
**********                                                              **********
**********************************************************************************
**********                                                              **********
**********                  Start um      08:35:47,18                   **********
**********                                                              **********
**********                  Ende um       08:35:47,19                   **********
**********                                                              **********
**********                  Laufzeit      00:00:00,01                   **********
**********                                                              **********
**********                  gelesen               716                   **********
**********                                                              **********
**********************************************************************************
```

Wow – und mit Strom und ganz großer Datenmenge?

```
**********************************************************************************
**********                                                              **********
**********                  Programmstart CSV-NACH-XML                  **********
**********                                                              **********
**********************************************************************************
**********************************************************************************
**********                                                              **********
**********                  Programmende CSV-NACH-XML                   **********
**********                                                              **********
**********************************************************************************
**********                                                              **********
**********                  Start um      08:39:56,73                   **********
**********                                                              **********
**********                  Ende um       08:40:26,58                   **********
**********                                                              **********
**********                  Laufzeit      00:00:29,85                   **********
**********                                                              **********
**********                  gelesen         4.204.201                   **********
**********                                                              **********
**********************************************************************************
```

Das hat jetzt nicht so viel gebracht. Aber die 100k brauchen jetzt keine Sekunde mehr:

```
**********************************************************************************
**********                                                              **********
**********                  Programmstart CSV-NACH-XML                  **********
**********                                                              **********
**********************************************************************************
**********************************************************************************
**********                                                              **********
**********                  Programmende CSV-NACH-XML                   **********
**********                                                              **********
**********************************************************************************
**********                                                              **********
**********                  Start um      08:42:05,54                   **********
**********                                                              **********
**********                  Ende um       08:42:06,37                   **********
**********                                                              **********
**********                  Laufzeit      00:00:00,83                   **********
**********                                                              **********
**********                  gelesen           100.101                   **********
**********                                                              **********
**********************************************************************************
```

3.3. Separate Behandlung Feld Bundesland

Mit den bisher gemachten Umbaumaßnahmen haben wir vermutlich den größten Zugewinn an Geschwindigkeit erreicht. Eine mögliche Verbesserung kann ich mir noch vorstellen, statt alle Bundesländer immer wieder neu auf Umlaute zu prüfen, könnten wir vielleicht auf die 17 unterschiedlichen Varianten abfragen. Warum 17? Wir haben doch nur 16 Bundesländer? In der csv-Datei sind 6 Einträge für „bundesweit“, zum Beispiel das Kennzeichen für das THW.

Damit wir später die unterschiedlichen Laufzeiten nebeneinander stellen können, lagern wir die Behandlung des Feldes „Bundesland“ in verschiedene Sections aus.

In der V110 ziehen wir den Teil zur Behandlung der Bundesländer raus

```
       V110-KONVERTIERE-DATENSATZ SECTION.
…
           MOVE    WS-INHALT-BULAND         TO RED-EIN-FELD
            PERFORM V111-UMLAUTE-UMSETZEN

            STRING  AUS-BULAND-AUF
                    WS-TEMP-AUSGABE
                    AUS-BULAND-ZU
            INTO    TEMP-AUSGABE-ZEILE
            PERFORM H200-SCHREIBEN-DATENSATZ
```

und packen ihn in eine eigene, neue Section:

```
      ****************************************************************
      *V112-BEHANDELN-BULAND SECTION.
      *    - Bundesland setzen
      *    - Variante mit Umsetzen Umlaute
      ****************************************************************
       V112-BEHANDELN-BULAND SECTION.

           IF ANALYSE-JA
              MOVE KONST-V112               TO TEMP-LOG
              PERFORM H400-SCHREIBEN-LOG
           END-IF

           MOVE    WS-INHALT-BULAND         TO RED-EIN-FELD
           PERFORM V111-UMLAUTE-UMSETZEN

           STRING  AUS-BULAND-AUF
                   WS-TEMP-AUSGABE
                   AUS-BULAND-ZU
           INTO    TEMP-AUSGABE-ZEILE
           PERFORM H200-SCHREIBEN-DATENSATZ

           .
```

Den Aufruf der V112 packen wir an die Stelle in der V110, an der wir eben den Code-Teil herausgezogen haben:

```
       V110-KONVERTIERE-DATENSATZ SECTION.
…
           PERFORM V112-BEHANDELN-BULAND
```

Als neue Section bauen wir uns eine `V113-BEHANDELN-BULAND`.

Der Kern der STRING-Funktion bleibt gleich. Wir vergleichen aber den ganzen Inhalt von WS-INHALT-BULAND am Stück mit Konstanten, die wir noch definieren müssen.

Die eigentliche Verarbeitung erfolgt dann wieder über ein EVALTUATE, allerdings fragen nicht das Feld direkt ab, sondern prüfen mit EVALUATE TRUE.

Zuerst die Deklarationen:

```
*****************************************************************
*    Bundesländer bw-59 th-44 by-150 sa-52 br-46 he-44 mv-44 ns-59
*    nw-81 rp-43 sl-9 sx-58 sh-18 bu-6
*****************************************************************
     05 BULAND-KONSTANTEN.
        10 BULAND-BW-ABFRAGE          PIC X(17) VALUE
           "Baden-Württemberg".
        10 BULAND-TH-ABFRAGE          PIC X(10) VALUE
           "Thüringen".
        10 BULAND-BY                  PIC X(06) VALUE
           "Bayern".
        10 BULAND-BW.
           15 FILLER                  PIC X(07) VALUE
          "Baden-W".
           15 FILLER                  PIC X(02) VALUE X'c3bc'.
           15 FILLER                  PIC X(09) VALUE
          "rttemberg".
        10 BULAND-TH.
           15 FILLER                  PIC X(02) VALUE
          "Th".
           15 FILLER                  PIC X(02) VALUE X'c3bc'.
           15 FILLER                  PIC X(06) VALUE
          "ringen".
        10 BULAND-SA                   PIC X(14) VALUE
           "Sachsen-Anhalt".
        10 BULAND-NW                   PIC X(19) VALUE
           "Nordrhein-Westfalen".
        10 BULAND-SX                   PIC X(07) VALUE
           "Sachsen".
        10 BULAND-RP                   PIC X(15) VALUE
           "Rheinland-Pfalz".
        10 BULAND-NS                   PIC X(13) VALUE
           "Niedersachsen".
        10 BULAND-BR                   PIC X(11) VALUE
           "Brandenburg".
        10 BULAND-MV                   PIC X(22) VALUE
          "Mecklenburg-Vorpommern".
        10 BULAND-BE                   PIC X(06) VALUE
           "Berlin".
        10 BULAND-BU                   PIC X(10) VALUE
           "bundesweit".
        10 BULAND-HE                   PIC X(06) VALUE
           "Hessen".
        10 BULAND-SH                   PIC X(18) VALUE
           "Schleswig-Holstein".
        10 BULAND-HB                   PIC X(06) VALUE
           "Bremen".
        10 BULAND-HH                   PIC X(07) VALUE
          "Hamburg".
        10 BULAND-SL                   PIC X(08) VALUE
           "Saarland".
```

Bei den beiden Bundesländern Baden-Württemberg und Thüringen müssen wir das „ü“ umsetzen. Deshalb müssen wir für diese beiden Länder noch einen extra-Schlenker definieren.

Der Vergleich mit dem Inhalt der csv-Datei für die beiden Länder muss das „ü“ beinhalten, die Ausgabe muss dann wieder mit hex’c3bc‘ erfolgen.

Der Code für die V113 am Beispiel von Bayern und Baden-Württemberg ist folgender:

```
      ****************************************************************
      *V113-BEHANDELN-BULAND SECTION.
      *    - Bundesland setzen
      *    - Variante mit Evaluate auf 17 KOnstanten
      ****************************************************************
       V113-BEHANDELN-BULAND SECTION.

           IF ANALYSE-JA
              MOVE KONST-V113               TO TEMP-LOG
              PERFORM H400-SCHREIBEN-LOG
           END-IF

           EVALUATE TRUE
           WHEN WS-INHALT-BULAND            =  BULAND-BY
                STRING  AUS-BULAND-AUF
                        BULAND-BY
                        AUS-BULAND-ZU
                INTO    TEMP-AUSGABE-ZEILE
                PERFORM H200-SCHREIBEN-DATENSATZ
           WHEN WS-INHALT-BULAND            =  BULAND-BW-ABFRAGE
                STRING  AUS-BULAND-AUF
                        BULAND-BW
                        AUS-BULAND-ZU
                INTO    TEMP-AUSGABE-ZEILE
                PERFORM H200-SCHREIBEN-DATENSATZ
…
            .
```

Die Behandlung der anderen Bundesländer ist analog. Den Aufruf der V113 erfolgt direkt unter V112. Damit haben wir die Möglichkeit eine von beiden Sections auszukommentieren, wenn die andere gerade laufen soll.

```
      *    - Variante mit Evaluate auf 17 KOnstanten
      ****************************************************************
       V110-KONVERTIERE-DATENSATZ SECTION.
…
      *     PERFORM V112-BEHANDELN-BULAND
           PERFORM V113-BEHANDELN-BULAND
```

Schauen wir auf die Laufzeit an, dazu schalten wir erst einmal den ANLYSE-Schalter wieder auf `ANALYSE-JA`.

Der erste Durchlauf ist mit V112, Analyse-Ja und den 100.101 csv-Einträgen liegt bei 1,08 Sekunden:

```
1701701 in H200-SCHREIBEN-DATENSATZ
1701702 in V111-UMLAUTE-UMSETZEN
1701703 ZeitZ
1701704 in H200-SCHREIBEN-DATENSATZ
1701705 in V112-BEHANDELN-BULAND
1701706 in V111-UMLAUTE-UMSETZEN
1701707 Sachsen-Anhalt
1701708 in H200-SCHREIBEN-DATENSATZ
1701709 in H200-SCHREIBEN-DATENSATZ
1701710 in H100-LESEN-DATENSATZ
1701711 in Z100-ABSCHLUSSARBEITEN
1701712 in H200-SCHREIBEN-DATENSATZ
1701713 ********************************************************************************
1701714 **********                                                      **********
1701715 **********                Programmende CSV-NACH-XML             **********
1701716 **********                                                      **********
1701717 ********************************************************************************
1701718 **********                                                      **********
1701719 **********                Start um      10:44:39,38             **********
1701720 **********                                                      **********
1701721 **********                Ende um       10:44:40,46             **********
1701722 **********                                                      **********
1701723 **********                Laufzeit      00:00:01,08             **********
1701724 **********                                                      **********
1701725 **********                gelesen           100.101             **********
1701726 **********                                                      **********
1701727 ********************************************************************************
```

Für die V113 sind wir bei 1,01 Sekunden:

```
1501364 in H200-SCHREIBEN-DATENSATZ
1501365 in V111-UMLAUTE-UMSETZEN
1501366 ZeitZ
1501367 in H200-SCHREIBEN-DATENSATZ
1501368 in V113-BEHANDELN-BULAND
1501369 in H200-SCHREIBEN-DATENSATZ
1501370 in H200-SCHREIBEN-DATENSATZ
1501371 in H100-LESEN-DATENSATZ
1501372 in Z100-ABSCHLUSSARBEITEN
1501373 in H200-SCHREIBEN-DATENSATZ
1501374 ********************************************************************************
1501375 **********                                                      **********
1501376 **********                Programmende CSV-NACH-XML             **********
1501377 **********                                                      **********
1501378 ********************************************************************************
1501379 **********                                                      **********
1501380 **********                Start um      10:50:03,39             **********
1501381 **********                                                      **********
1501382 **********                Ende um       10:50:04,40             **********
1501383 **********                                                      **********
1501384 **********                Laufzeit      00:00:01,01             **********
1501385 **********                                                      **********
1501386 **********                gelesen           100.101             **********
1501387 **********                                                      **********
1501388 ********************************************************************************
```

Immerhin, eine kleine Verbesserung zeigt sich hier.

Die reduzierte Zahl der Log-Einträge erklärt sich dadurch, dass wir die V111 nicht mehr aufrufen, in der 1 Protokollsatz geschrieben wird und die Ausgabe des Bundeslands erfolgt ebenfalls nicht mehr. Macht pro csv-Satz 2 Aufrufe weniger, in Summe sind das dann 200.202 Log-Einträge.

Da wir jetzt wissen, dass das funktioniert, schalten wir die Analyse wieder aus.

Die Werte sind dann für die V112:

und V113:

Die leichte Verbesserung bestätigt sich also auch hier.

Zum Schluss noch einmal die V113 mit unserer originalen Eingabe von 716 Datensätzen in der csv-Datei:

Für uns nicht mehr messbar 😉

4. Abschluss Projekt

Ein toller Erfolg.

An dieser Stelle steht immer der Ausblick, wie es weiter gehen könnte, was ist noch zu verbessern oder verändern. In diesem Fall kann ich aber keine weiteren Empfehlungen abgeben, ich denke, die Stärken, aber auch die Schwächen von COBOL wurden deutlich.

Von Paul Watzlawick soll der Ausspruch stammen: „Wer als Werkzeug nur einen Hammer hat, sieht in jedem Problem einen Nagel.“

Daher mein Appell an alle Entwickler, nehmt Euch die Zeit Euren persönlichen Werkzeugkasten mit so vielen Programmiersprachen wie möglich zu füllen. Probiert alles aus, seid neugierig und saugt auf was immer Ihr zu hören bekommt.

Dieses Programm ist stark mit der einen Aufgabe der Inhalte aus der kennzeichen.csv verknüpft. Wie wir schon festgestellt haben, ist eine Generalisierung mit COBOL nicht wirklich möglich. Wenn die Datei um eine Spalte erweitert werden soll, weil wir uns zum Beispiel entschlossen haben, auch die Zahl der Einwohner oder die Koordinaten mit aufzunehmen, müssen wir das ganze Programm umbauen.

Mir schwebt als nächstes Projekt vor, ein Analyseprogramm zu schreiben, das die notwendigen Angaben wie Anzahl Spalten und maximale Länge der einzelnen Werte der Felder ausgibt. Damit könnten wir eine Entscheidungshilfe zum Bau des eigentlichen COBOL-Programms bekommen.

Das war es jetzt aber von meiner Seite, ich hoffe, es hat Euch gefallen. Bei Fragen und Anregungen, aber auch, wenn Euch Fehler auffallen, würde ich mich über Rückmeldung über papa@papa-programmiert.de freuen.

Viel Spaß beim Programmieren!