

RPG and Unicode – Teil 7

String-Funktionen mit Anzahl Zeichen und UTF-8

Im letzten Artikel haben wir uns mit der Verarbeitung von UTF-8 Daten mit alphanumerischen Built-In-Funktionen beschäftigt. Per Default müssen alle notwendigen Angaben in Anzahl-Byte erfolgen, was gerade bei UTF-8-Daten bei denen ein Zeichen 1-4 Byte einnehmen kann, nicht ganz trivial ist. Einfacher wäre es, wenn man die Anzahl Zeichen verwenden könnte. ... und genau das ist auch möglich!

Erweiterung in alphanumerischen Built-In-Funktionen

Damit die Start-Positionen, Längen oder Rückgabe-Werte in Anzahl Zeichen anstatt in Anzahl Byte ermittelt und ausgegeben werden können, wurden alle alphanumerischen Built-In-Functions um einen optionalen Parameter erweitert. Dieser Parameter, sofern er übergeben wird, muss immer als letzter Parameter angegeben werden.

Dieser Parameter kann zwei Ausprägungen haben:

- ***STDCHARSIZE** Wird der optionale Parameter nicht oder mit *STDCHARSIZE übergeben, so werden die Start-Positionen, Längen und Rückgabe-Werte in Anzahl Byte erwartet bzw. zurückgegeben.
- ***NATURAL** Wird der optionale Parameter mit *NATURAL übergeben, so werden die Start-Positionen, Längen und Rückgabe-Werte in Anzahl Zeichen erwartet bzw. zurückgegeben.

Die Erweiterung gilt für die im Folgenden aufgelisteten Built-In-Funktionen

Built-In-Variable	Beschreibung	Default	Option	
			*STDCHARSIZE	*NATURAL
%CHECK	Zeichen prüfen	Anzahl Byte	Anzahl Byte	Anzahl Zeichen
%CHECKR	Zeichen von rückwärts prüfen			
%CONCAT	Werte mit Trennzeichen verknüpfen			
%CONCATARR	Array-Elemente mit Trennzeichen verknüpfen			
%LEFT	Substring Zeichen von der linken Seite			
%LOWER	Zeichen in Kleinschrift konvertieren			
%REPLACE	Zeichenkette ersetzen (nur 1. Vorkommen)			
%RIGHT	Substring Zeichen von der rechten Seite			
%SCAN	Zeichen suchen			
%SCANR	Zeichen von rückwärts suchen			
%SCANRPL	Suchen und Ersetzen (alle Vorkommen)			
%SPLIT	Zeichenkette aufsplitten			
%SUBSTR	Substring			
%TRIM	Zeichen entfernen			
%UPPER	Zeichen in Großschrift konvertieren			
%XLATE	Zeichen konvertieren			

Abbildung 1: Alphanumerische Built-In-Funktionen mit Erweiterung für Unicode-Verarbeitung in Anzahl Zeichen

Schauen wir uns ein Beispiel an, in dem der optionale-Parameter mit *NATURAL verwendet wird.

Wie im vorherigen Artikel gehen wir wieder von der folgenden Datenstruktur (DSName) aus. Diese Datenstruktur wird von einer Feldgruppe (Array) überlagert und enthält die 5 Unterfelder à 50 Byte! (nicht Zeichen) mit CCSID 1208 (also UTF-8). Die Unterfelder wurden mit diversen Namen initialisiert. Lediglich der erste Name (Peter Hausberg) besteht nur aus den Buchstaben A-Z. Alle anderen Namen beinhalten Umlaute, akzentuierte Buchstaben und oder ein ß.

RPG and Unicode – Teil 7

String-Funktionen mit Anzahl Zeichen und UTF-8

```

DCL-DS DSName      Qualified;
  Name1           VarChar(50) CCSID(1208)  Inz('Peter Hausberg');
  Name2           VarChar(50) CCSID(1208)  Inz('Anna Weißenberg');
  Name3           VarChar(50) CCSID(1208)  Inz('Günther Roßberg');
  Name4           VarChar(50) CCSID(1208)  Inz('Käthe Küßbergen');
  Name5           VarChar(50) CCSID(1208)  Inz('Renée Lébergus');
  Arr             VarChar(50) CCSID(1208)  Dim(5) Pos(1);
End-Ds;

```

Abbildung 2: Datenstruktur mit UTF-8-Daten

Built-In-Funktion %SUBST(), Ermitteln Start-Position mit %SCAN und *NATURAL

In dem folgenden Beispiel wird die Datenstruktur DSName über eine Schleife gelesen. Für jedes Unterfeld wird mit Hilfe der Funktion %SCAN() die Position des Wertes 'berg' ermittelt. Die ermittelte Position wird in der Built-In-Funktion %SUBST() als Start-Position eingesetzt.

Bei beiden Built-In-Funktionen wurde der optionale Parameter-Wert *NATURAL am Ende der Built-In-Funktion angegeben.

```

Dcl-Proc Int_Subst_Natural;

  DCL-DS DSName      Qualified;
    Name1           VarChar(50) CCSID(1208)  Inz('Peter Hausberg');
    Name2           VarChar(50) CCSID(1208)  Inz('Anna Weißenberg');
    Name3           VarChar(50) CCSID(1208)  Inz('Günther Roßberg');
    Name4           VarChar(50) CCSID(1208)  Inz('Käthe Küßbergen');
    Name5           VarChar(50) CCSID(1208)  Inz('Renée Lébergus');
    Arr             VarChar(50) CCSID(1208)  Dim(5) Pos(1);
  End-Ds;

  DCL-S StrSearch    VarChar(50) CCSID(1208)  Inz('berg');

  DCL-S Index        Uns(3);
  DCL-S StrPos       Uns(3);
  DCL-S Result       VarChar(50) CCSID(1208);

  //-----
  For Index = 1 to %Elem(DSName.Arr);
    StrPos = %Scan(StrSearch: DSName.Arr(Index): *Natural);
    DspText = DSName.Arr(Index) + ' ' + StrSearch +
              ' Position: '      + %Char(StrPos);
    Dsply DspText;
    Result = %Subst(DSName.Arr(Index): StrPos: *Natural);
    DspText = DSName.Arr(Index) + ' String beginning with ' +
              StrSearch          + ' = ' + Result;
    Dsply DspText;
  EndFor;

  return;
End-Proc;

```

Abbildung 3: Verarbeitung von UTF-8 Daten mit *NATURAL in Built-In-Funktionen

Führt man die Prozedur aus erhält man das folgende Ergebnis:

RPG and Unicode – Teil 7

String-Funktionen mit Anzahl Zeichen und UTF-8

```
DSPLY Peter Hausberg berg Position: 11
DSPLY Peter Hausberg String beginning with berg = berg
DSPLY Anna Weißberg berg Position: 12
DSPLY Anna Weißberg String beginning with berg = berg
DSPLY Günther Roßberg berg Position: 12
DSPLY Günther Roßberg String beginning with berg = berg
DSPLY Käthe Küßbergen berg Position: 10
DSPLY Käthe Küßbergen String beginning with berg = berge
DSPLY Renée Lébergus berg Position: 9
DSPLY Renée Lébergus String beginning with berg = bergus
```

Abbildung 4: Ergebnis aus Verarbeitung mit *NATURAL in Built-In-Funktionen

Das Ergebnis ist korrekt, d.h. alle ausgegebenen Strings beginnen mit ‚berg‘. Wenn man sich allerdings die Start-Positionen anschaut, so wird man feststellen, dass es sich bei Position (im Gegensatz zu dem Beispiel im letzten Artikel) nicht mehr um die Anzahl in Byte, sondern tatsächlich um die Anzahl Zeichen handelt.

Build-In-Funktion %SUBST() und hardcodierte Start-Positionen mit *NATURAL

Wenn man also jetzt, anstatt die Positionen zu berechnen, diese als Anzahl Zeichen hardcodiert, sollte, sofern der optionale Wert *NATURAL bei den Built-In-Funktionen angegeben wurde, ebenfalls das korrekte Ergebnis angezeigt werden.

In dem folgenden Beispiel wird wiederum die zuvor gezeigte Datenstruktur mit der überlagernden Feldgruppe (Array) und mit den UTF-8-Unterfeldern mit dem gleichen Inhalt wie zuvor verarbeitet.

In einer zweiten Daten-Struktur ebenfalls mit überlagernder Feldgruppe (Array) ist die Position in Anzahl Zeichen hinterlegt, an der sich der Text ‚berg‘ in den Unterfeldern der ersten Datenstruktur befindet.

Die Datenstruktur wird in einer Schleife gelesen, und für jeden Namen mit Hilfe der Built-In-Funktion %SUBST() der mit ‚berg‘ beginnende Text ermittelt und ausgegeben.

In der Built-In-Funktion %SUBST() wird zum einen der Wert aus der zweiten Datenstruktur angegeben und zum anderen der optionale Parameter-Wert *NATURAL hinzugefügt.

RPG and Unicode – Teil 7 String-Funktionen mit Anzahl Zeichen und UTF-8

```
Dcl-Proc Int_Subst_Natural_Fix;

  DCL-DS DSName      Qualified;
  Name1      VarChar(50) CCSID(1208)  Inz('Peter Hausberg');
  Name2      VarChar(50) CCSID(1208)  Inz('Anna Weißenberg');
  Name3      VarChar(50) CCSID(1208)  Inz('Günther Roßberg');
  Name4      VarChar(50) CCSID(1208)  Inz('Käthe Küßbergen');
  Name5      VarChar(50) CCSID(1208)  Inz('Renée Lébergus');
  Arr        VarChar(50) CCSID(1208)  Dim(5) Pos(1);
End-Ds;

  DCL-DS DSPos       Qualified;
  Pos1       Uns(3)   Inz(11);
  Pos2       Uns(3)   Inz(12);
  Pos3       Uns(3)   Inz(12);
  Pos4       Uns(3)   Inz(10);
  Pos5       Uns(3)   Inz(9);
  Arr        Uns(3)   Dim(5) Pos(1);
End-Ds;

  DCL-S StrSearch    VarChar(50) CCSID(1208)  Inz('berg');

  DCL-S Index        Uns(3);
  DCL-S StrPos       Uns(3);
  DCL-S Result       VarChar(50) CCSID(1208);

  -----
  For Index = 1 to %Elem(DSName.Arr);
    StrPos = DSPos.Arr(Index);
    DspText = DSName.Arr(Index) + ' ' + StrSearch +
              ' Position: ' + %Char(StrPos);
    Dsply DspText;
    Result = %Subst(DSName.Arr(Index): StrPos: *Natural);
    DspText = DSName.Arr(Index) + ' String beginning with ' +
              StrSearch + ' = ' + Result;
    Dsply DspText;
  EndFor;

  return;
End-Proc;
```

Abbildung 5: Verarbeitung von UTF-8 Daten mit *NATURAL und hardcodierten Werte in Built-In-Funktionen

Führt man die Prozedur aus so erhält man das folgende Ergebnis.

RPG and Unicode – Teil 7

String-Funktionen mit Anzahl Zeichen und UTF-8

```
DSPLY Peter Hausberg berg Position: 11
DSPLY Peter Hausberg String beginning with berg = berg
DSPLY Anna weißenberg berg Position: 12
DSPLY Anna weißenberg String beginning with berg = berg
DSPLY Günther Roßberg berg Position: 12
DSPLY Günther Roßberg String beginning with berg = berg
DSPLY Käthe Küßbergen berg Position: 10
DSPLY Käthe Küßbergen String beginning with berg = berge
DSPLY Renée Lébergus berg Position: 9
DSPLY Renée Lébergus String beginning with berg = bergus
```

Abbildung 6: Ergebnis aus Verarbeitung mit *NATURAL und hardcodierten Positionen in Built-In-Funktionen

Alle Werte wurden korrekt ermittelt, d.h. alle Ausgäbe-Texte beginnen mit 'berg'.

Um Unicode-Werte mit alphanumerischen Built-In-Funktionen und der Anzahl Zeichen verarbeiten zu können, muss also nur der optionale Parameter-Wert *NATURAL hinzugefügt werden.

... nur wie leicht vergisst man das das eine oder andere Mal?

Gibt es keine Einstellung, über die man *NATURAL für die alphanumerischen Built-In-Funktionen als Unterlassungs-Wert setzen kann?

In den Ausnahmefällen, bei denen mit der Anzahl an Byte benötigt könnte der optionale Parameter-Wert *STDCHARSIZE angegeben werden.

An dieser Stelle kommen die Schlüssel-Worte CHARCOUNT und CHARCOUNTTYPES in den Control-Options (H-Bestimmungen) zum Zug.

Control-Options (H-Bestimmungen) CHARCOUNT und CARCOUNTTYPES

Sofern man grundsätzlich für UTF-8 oder andere Mixed-Data in den alphanumerischen Built-In-Funktionen mit Anzahl Zeichen anstatt Anzahl Byte arbeiten möchte, kann man dies in den Control-Options (H-Bestimmungen) global setzen.

- CHARCOUNT Bei dem Schlüssel-Wort CHARCOUNT können zwei Ausprägungen angegeben werden:
 - *STDCHARSIZE
Wird CHARCOUNT nicht angegeben oder mit der Ausprägung *STDCHARSIZE, entspricht dies dem Standard, d.h. die Positionen, Längenangaben und Rückgabe-Werte werden in Anzahl Byte erwartet und auch zurückgegeben
 - *NATURAL
Wird der Wert *NATURAL angegeben, so wird in den alphanumerischen Built-In-Funktionen die Anzahl an Zeichen verwendet.

Allerdings erfolgt dies Umsetzung nicht bei jedem alphanumerischen Datentypen, sondern nur bei den Datentypen, die in dem Schlüssel-Wort CHARCOUNTTYPES ebenfalls in den Control-Options (H-Bestimmungen) hinterlegt sind.
- CHARCOUNTTYPES In der Option CHARCOUNTTYPES werden die Datentypen aufgelistet, für die bei der Verwendung von alphanumerischen Built-In-Funktionen die Angabe von Positionen, Längen und in Rückgabe-Werten in Anzahl Zeichen erfolgen soll.

RPG and Unicode – Teil 7

String-Funktionen mit Anzahl Zeichen und UTF-8

Die folgenden Typen können angegeben werden:

- *UTF8
- *UTF16
- *JOB RUN
- *MIXEDEBCDIC
- *MIXEDASCII

Für die Verwendung von Anzahl Zeichen anstatt Anzahl Byte in Unicode sind die ersten beiden Typen, also *UTF8 und *UTF16 erforderlich.

Die folgende Abbildung zeigt Control-Options/H-Bestimmungen in denen sowohl das Schlüssel-Wort CHARCOUNT als auch das Schlüssel-Wort CHARCOUNTTYPES angegeben wurde. Mit diese Control-Options wird für UTF-8 oder UTF-16 Texten in den alphanumerischen Built-In-Funktionen die Anzahl Zeichen verwendet.

```
Ctl-Opt DEBUG DECEDIT('0,') DATEDIT(*DMY.)
        CCSID(*EXACT)
        CCSIDCVT(*LIST: *EXCP)
        CHARCOUNT(*NATURAL) CHARCOUNTTYPES(*UTF8: *UTF16)
/If defined (*CRTBNDRPG)
DftActgrp(*No) ActGrp('BIFLENU81')
/EndIf
;
```

Abbildung 7: Control-Options/H-Bestimmungen mit CHARCOUNT und CHARCOUNTTYPES

Compiler-Direktive /CHARCOUNT

Mit Hilfe der Control-Options CHARCOUNT und CHARCOUNTTYPES kann für die ganze Quelle festgelegt werden, ob die alphanumerischen Built-In-Functions bei Unicode-Werten mit der Anzahl Byte oder der Anzahl Zeichen arbeiten.

Es könnte jedoch sein, dass man die Einstellungen nicht nur universell, sondern vielleicht auch pro Prozedur, die in der gleichen Quelle hinterlegt sind, handeln möchte.

Dies kann man mit Hilfe der Compiler-Direktive /CHARCOUNT bewerkstelligen. Für die Compiler-Direktive können zwei Ausprägungen angegeben werden:

- /CHARCOUNT STDCHARSIZE Alle alphanumerischen Built-In-Funktionen, die nach der Compiler-Direktive codiert wurden und mit denen UTF-8 Daten verarbeitet werden, verwenden Anzahl Byte.
Die Regel gilt entweder bis zum Ende der Quelle oder zur nächsten /CHARCOUNT Compiler-Direktive, die das Verhalten wieder ändert.
- /CHARCOUNT NATURAL Alle alphanumerischen Built-In-Funktionen, die nach der Compiler-Direktive codiert wurden und mit denen UTF-8 Daten verarbeitet werden, verwenden die Anzahl Zeichen.
Die Regel gilt entweder bis zum Ende der Quelle oder zur nächsten /CHARCOUNT Compiler-Direktive, die das Verhalten wieder ändert.

Anmerkung: Die Compiler-Direktive kann nur gesetzt werden, wenn das Schlüssel-Wort CHARCOUNTTYPES in den Control-Options (H-Bestimmungen) gesetzt wurde!

RPG and Unicode – Teil 7

String-Funktionen mit Anzahl Zeichen und UTF-8

In dem folgende Code-Beispiel wird die erste Prozedur `Int_Subst_Std_Natural` mit `/CHARCOUNT NATURAL` ausgeführt, d.h. Positionen, Längen-Angaben sowie Rückgabe-Werte von alphanumerischen Built-In-Funktionen werden als Anzahl Zeichen interpretiert.

Zu Beginn der zweiten Funktion `Int_Subst_Std_Byte` wird eine zweite Compiler-Direktive eingebunden, durch die die erste Compiler-Direktive wieder zurückgesetzt wird, d.h. in der Funktion `Int_Subst_Std_Byte` und allen folgenden Code-Zeilen werden, werden die Positionen, Längen-Angaben und Rückgabe-Werte aus alphanumerischen Built-In-Funktionen als Anzahl Byte verarbeitet.

```
Ctl-Opt CharCountTypes(*UTF8: *UTF16);
//*****
/CharCount Natural
Dcl-Proc Int_Subst_Std_Natural;

- 37 lines excluded. -----

    return;
End-Proc;
//*****
- 4 lines excluded. -----
/CharCount StdCharSize
Dcl-Proc Int_Subst_Std_Byte;

- 37 lines excluded. -----

    return;
End-Proc;
```

Abbildung 8: Compiler-Direktive /CHARCOUNT

Soweit zu der Verarbeitung von UTF-8-Daten in RPG.

Fazit

Wenn man alles zusammennimmt und richtig einsetzt, also:

- Control-Options (H-Bestimmungen)
 - CCSID(*EXACT) Keine Konvertierung zur Laufzeit von hardcodierten Werten
 - CCSIDCVT Anzeige Konvertierungs-Probleme in Kompilierungsliste bzw. Abbruch bei der Konvertierung in Substitutions-Zeichen
 - CHARCOUNT Positionen, Längen und Rückgabe-Werte in alphanumerischen Built-In-Funktionen entweder in Anzahl Byte (Standard) oder Anzahl Zeichen
 - CHARCOUNTTYPES Auflistung der Datentypen, für die bei alphanumerischen Built-In-Funktionen bei Schlüssel-Wort `CHARCOUNT(*NATURAL)` Anzahl Byte verwendet werden.
 - OPENOPT Über das Schlüssel-Wort `OPENOPT` kann gesteuert werden, ob bei der Verarbeitung von den in F-Bestimmungen definierten Dateien mit Native I/O die Daten grundsätzlich in die Job CCSID (Default) konvertiert werden, oder ob sie unkonvertiert verarbeitet werden.
- Datei-Deklarationen (F-Bestimmungen)

RPG and Unicode – Teil 7

String-Funktionen mit Anzahl Zeichen und UTF-8

- DATA Anstatt OPENOPT(*NOCVT) für alle Dateien zu verwenden, kann in den File-Deklarationen (F-Bestimmungen) bei jeder Datei individuell entschieden werden, ob die Daten in die Job-CCSID konvertiert werden (Default) oder unkonvertiert verarbeitet werden.
- Definitionen (D-Bestimmungen)
 - CCSID(Nummer) Definition von standalone Variablen oder Datenstruktur-Unterfelder mit der gewünschten CCSID z.B. CCSID(1208) = Unicode
 - CCSID(*EXACT) Angabe bei der Definition von externen Datenstrukturen oder Datenstrukturen, die mit LIKERECD definiert werden, um eine Konvertierung der Daten in die Job-CCSID zu vermeiden
- C-Bestimmungen
 - %CHARCOUNT() Ermittlung der String Länge in Anzahl Zeichen. (%LEN() liefert Anzahl Byte)
 - Alle alphanumerischen Built-In-Funktionen

Über einen optionalen zusätzlichen Parameter kann man festlegen, ob für diese Built-In-Funktion die Positionen, Längen und Rückgabewerte in Anzahl Zeichen oder Anzahl Spalten verwendet werden soll.

Ein globales Handling kann durch die Schlüssel-Worte CHARCOUNT und CHARCOUNTTYPES in den Control-Options (H-Bestimmungen) gesetzt werden.

Mit Hilfe der Compiler-Direktive /CHARCOUNT kann das allgemeine Handling innerhalb der Quelle geändert werden.

... ist die korrekte Verarbeitung von UTF-8 oder UTF-16 Daten in RPG kein Problem.

Und nun viel Spaß beim Verarbeiten von Unicode-Daten in RPG!