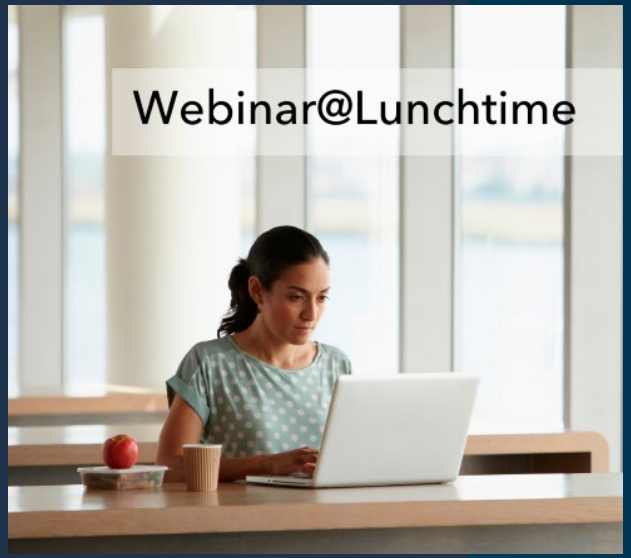


Webinar@Lunchtime

Spezielle Alphanumerische Funktionen und deren Feinheiten

THE POWER TO KNOW.

Inhaltsübersicht

1. Speicherung alphanumerischer Werte

Unterschiedliche Speicherarten für alphanumerische Werte

Bytebasierte Speicherung:
1 Byte pro Zeichen

„Herkömmliche“ alphanumerische Funktionen

Zeichenbasierte Speicherung:
Mehrere Bytes pro Zeichen
(z. B. kyrillische oder japanische Schriftzeichen)

K-Funktionen

z. B. bei UTF-8 Codierung der Datei

(K-)Funktionen für alphanumerische Werte

Alphanumerische Zeichen mit mehr als 1 Byte Speicherplatz

Unterscheidung alphanumerischer Speicherung nach:

SBCS	DBCS	MBCS
Single-byte character set	Double-byte character set	Multi-byte character set
I18N Level 0	I18N Level 1	I18N Level 2

Bisheriger Standard

Mehr Informationen unter: SAS® 9.4 National Language Support (NLS): Reference Guide, Fifth Edition

K-Funktionen unterstützen DBCS und MBCS!

z. B. Ksubstr, KCompress, KIndex, KScan, KLength …

Problem: „Herkömmliche“ Funktion bei **zeichen**basierter Speicherung

Der Wert ist nicht gespeichert mit 1 Byte pro Zeichen!

Für die „normale“ Substr-Funktion ist z. B.
1. Zeichen = 1. Byte
2. Zeichen = 2. Byte
usw.

„Normale“ Substr-Funktion arbeitet hier nicht richtig!

Beispiel:
STR=’E282AC313233’x , 6 Bytes für 4 Zeichen

Hexadezimal-Darstellung	E282AC	31	32	33
	€	1	2	3

Welche Funktionen sind Level2 fähig?

In der Online-Hilfe zu finden

Auszug: Darstellung verschiedener alphanumerischer Funktionen und Einsatzmöglichkeit bei zeichenbasierter Speicherung

SAS String Functions

Function	Description	I18N Level 0	I18N Level 1	I18N Level 2
ANORM420 Function	Returns a normalized string from an input string encoded in EBCDIC420.		X	
ANYALNUM Function in *SAS Functions and CALL Routines: Reference*	Searches a character string for an alphanumeric character, and returns the first position at which the character is found.		X	
ANYALPHA Function in *SAS Functions and CALL Routines: Reference*	Searches a character string for an alphabetic character, and returns the first position at which the character is found.		X	
ANYCNTRL Function in *SAS Functions and CALL Routines: Reference*	Searches a character string for a control character, and returns the first position at which that character is found.			X
ANYDIGIT Function in *SAS Functions and CALL Routines: Reference*	Searches a character string for a digit, and returns the first position at which the digit is found. **Note:** This function is assigned an I18N Level 1 status unless a VARCHAR variable is used, or if the function is threaded or runs in DS2. If these exceptions occur, then this function is assigned an I18N Level 2 status.		X	

2. Kennen Sie DIE?

Length-, Any- und Not-Funktionen

LENGTH-Funktionen

Länge alphanumerischer Werte bestimmen

Nachfolgende Leerzeichen werden nicht mitberechnet bei LENGTH und LENGTHN.

```
Data test;
   text="abc";
   text_nichts="";
   text_nachfolgLeerz="abc  ";
run;
```

Ergebnisse der unterschiedlichen Funktionen

Funktionen	Length	LengthN	LengthC
text	3	3	3
Text_nichts	1	0	1
Text_nachfolgLeerz	3	3	5

LENGTH**N**: Bei Missingwerten ist die Länge 0, nicht1!

LENGTH**C**: Berechnet auch nachfolgende Leerzeichen

Syntaxbeispiel:
laenge=length(text);

ANY Funktionen

Gibt es bestimmte Zeichen in meiner alphanumerischen Variablen/in einem Text?

Anwendungsbeispiel:

Sie wollen prüfen, ob in einer langen Zeichenkette nur Zahlen stehen oder auch andere Zeichen, weil Sie den Wert typkonvertieren wollen zu numerisch.

Any-Funktionen	
Anyalnum	Anyname: gültige Zeichen für SAS Var-Namen
Anyalpha	Anyprint: Druckbare Zeichen
Anydigit	Anypunct: Punktzeichen z. B. : ; ! ?
AnyUPPER	Anyspace: z. B. Blank, Tab
Anylower	Anycntrl: Steuerungszeichen
Anyxdigit: Hexadezimale Darstellung einer Zahl	
Anygraph: graphische Zeichen	
Anyfirst: gültiges Zeichen, das ZUERST in SAS Variablennamen stehen darf (z. B. Buchstabe oder Unterstrich)	

ANY Funktionen
NOT Funktionen

Die ANY-Funktionen gibt es auch verneint als NOT-Funktionen

Beispiele:
NOTALNUM
NOTDIGIT
NOTALPHA
NOTPUNCT
...

Any or Not

Die ANY-Funktionen geben Ihnen die Position des zuerst gefundenen gewünschten Zeichens aus.

Beispiel: var=„ABC12“; Anydigit(var) = 4
Anyalnum(var) = 1

Die NOT-Funktionen geben die Position aus, an der zuerst ein Zeichen steht, das nicht gewünscht ist.

Notdigit(var) = 1

3. Funktions-Modifier: Erleichtern Ihre Programmierung

Beispiel für die COMPRESS Funktion

Modifier in Funktionen

Beispiele für die COMPRESS Funktion

Unterschiedliche Modifier bei unterschiedlichen Funktionen –

Nutzen Sie die ONLINE Hilfe!

Modifier sind Buchstaben, die als Platzhalter für bestimmte Zeichengruppen bzw. Funktionalitäten stehen.

Sie müssen an bestimmten Stellen innerhalb der Funktionsklammer stehen.

COMPRESS Funktion

Zunächst Standardfunktionalitäten

TEXT=123 das ist ein Beispiel _SAS!

Beschreibung	Syntax	Ergebnis
Leerzeichen entfernen	COMPRESS(text)	TEXT=123dasisteinBeispiel_SAS!
Kleines e entfernen	COMPRESS(text, “e“)	TEXT=123 das ist in Bispil _SAS!

Modifier: Beispiel COMPRESS-Funktion

Zeichen entfernen oder behalten

Syntax	Erklärung
COMPRESS (Variable)	Entfernt standardmäßig Leerzeichen in einer Variablen
COMPRESS (Variable, *“Zeichen“*)	Entfernt das angegebene Zeichen (2. Stelle)
COMPRESS (Variable, , *“Modifier“*)	Entfernt eine bestimmte Art von Zeichen (2 Kommas, der Modifier steht an 3. Stelle)
COMPRESS (Variable, , *“Modifier“* “K“)	**BEHÄLT** (k=“**k**eep”) eine bestimmte Art von Zeichen
COMPRESS (Variable, *“Zeichen“* , “k“)	**BEHÄLT** („keep“) das angegebene Zeichen

COMPRESS Funktion

… und jetzt mit Modifier

Modifier bei Compress: 3. Stelle (nach dem zweiten Komma!)

Beschreibung	Syntax	Ergebnis
Alle BUCHSTABEN entfernen	Compress (text,, “a“) Modifier a=alphabetic	TEXT=123 _!
Alle ZAHLEN entfernen	COMPRESS (text, , “d“) Modifier d=digits	TEXT= das ist ein Beispiel _SAS!
Kleines e und ZAHLEN entfernen	COMPRESS(text, “e“, “d“)	TEXT= das ist in Bispil _SAS!

Text=123 das ist ein Beispiel _SAS!

COMPRESS-Funktion

Zeichen behalten – mit dem Modifier K

Beschreibung	Syntax	Ergebnis
Nur die Zahlen behalten	COMPRESS(text,, “kd“)	TEXT=123
Nur SAS erlaubte Zeichen für Datei- und Variablennamen behalten	COMPRESS(text,, “kn“) n wie Name convention	TEXT=123dasisteinBeispiel_SAS
Kleines e und Zahlen behalten	COMPRESS(text, „e“, “kd“)	TEXT=123eee

Beispiele für weitere Modifier:

i=ignore (Groß-/Kleinschreibung)

u=uppercase

p=punctuation (!;:.)

l=lowercase

4. Ersetzungsfunktionen

Ersetzungsfunktionen

Achtung:

Neue Variablen mit TRANWRD oder TRANSTRN Funktionen erstellen:
Standardlänge = 200B!

Funktion	Beschreibung
TRANWRD	Wort ersetzen
TRANSTRN	Zeichen ersetzen Mit TRIMN Funktion können Missing-Werte komplett entfernt werden
TRANSLATE	Zeichen ersetzen
SUBSTR(...)= Linke Seite	Zeichen an bestimmter Stelle ersetzen

Tranwrd Funktion

Wort ersetzen

Syntax: TRANWRD(*text*, *Ziel*, *Ersatz*)

- *text*: Alphanumerische Spalte
- *Ziel*: Suchstring im *text*
- *Ersatz*: Zeichenfolge, die *Ziel* ersetzt

```
Summary2=tranwrd(Summary, 'hurricane', 'storm');
```

Summary	Summary2 200B
Category 3 hurricane initially...	Category 3 storm initially...
The largest (in size) Atlantic hurricane on record...	The largest (in size) Atlantic storm on record...

Translate Funktion

Einzelne Zeichen ersetzen

Translate ersetzt Zeichen in einer Zeichenkette.

Syntax: **TRANSLATE**(*source*, *to-1*, ***from-1*** <, …***to-n***, *from-n*>)

Translate ändert das erste Zeichen von ***from*** *zu dem ersten Zeichen von* ***to usw.***

Beispiel:

```
T_ersetzen=translate (text, '12', 'AB')
```

Text 6B	T_ersetzen 6B
ABCDAB	12CD12

Ersetzen mit der SUBSTR Funktion (linke Seite)

An bestimmten Positionen Zeichen ersetzen

SUBSTR-Funktion (linke Seite einer Zuweisung) ersetzt Zeichen an einer bestimmten Stelle in einer alphanumerischen Variablen.

Syntax: SUBSTR(*Zeichenfolge, Start <,Länge>*) = Wert;

Beispiel: Ersetzen zweier Zeichen ab Position 11.

```
Substr(Location,11,2)= 'OH';
```

Location $ 18	Location $ 18
Columbus, GA 43227	Columbus, OH 43227

11

Die SUBSTR-Funktion (Linke Seite)

Ergänzende Informationen

Start	Ausgangsposition, um Zeichen mit *Wert* zu ersetzen.
Länge	Anzahl der Zeichen, die in *Zeichenfolge* ersetzt werden sollen. Fehlt *Länge* werden alle Zeichen ab *Start* bis zum Ende von *Zeichenfolge* ersetzt. *Länge* kann nicht größer sein als die Restlänge von *Zeichenfolge* (einschließlich endständiger Leerzeichen) nach *Start*.

Transtrn Funktion

Zeichen ersetzen + Besonderheiten

Ersetzt oder entfernt alle Vorkommen einer Teilzeichenkette in einer Zeichenkette.

Syntax: **TRANSTRN**(*source*, *Ziel*, *Ersetzungswert*)

Beispiel:

```
Ersetzen=TRANSTRN(Wert,'B','1');
```

WERT 5B	Ersetzen 200B
ABCBA	A1C1A

Besonderheit bei der TRANSTRN Funktion (1)

Missings zu keinem Blank „verkürzen“ – TRIMN Funktion hilft

Was ist, wenn Sie B mit „nichts“ ersetzen wollen, also keine Leerzeichen haben wollen?

Vorschlag 1)

```
Ersetzen1=TRANSTRN(Wert, “B“, ““);
```

Vorschlag 2)

```
Ersetzen2=TRANSTRN(Wert, “B“, TRIM(““) );
```

Trim entfernt nachfolgende Leerzeichen, aber bei Missings wird trotzdem ein Leerzeichen durch TRIM bleiben.

WERT 5B	Ersetzen1 200B	Ersetzen2 200B
ABCBA	A C A ☹	A C A ☹

Besonderheit bei der TRANSTRN Funktion(2)

Missings zu keinem Blank „verkürzen“ – TRIMN Funktion hilft

So klappt es:

TIPP: TRIMN-Funktion - Ein Leerzeichen als Missing wird tatsächlich entfernt (als Kennzeichen für einen Missing), nicht bei der TRANSLATE Funktion möglich.

Vorschlag 3) **Ersetzen3= TRANSTRN(Wert, “B“, TRIMN(‘‘));**

WERT 5B	Ersetzen3 200B
ABCBA	ACA ☺

TRIM Funktion vs TRIMN Funktion

Sehen Sie dazu auch das Beispiel bei der TRANSTRN Funktion

Interessant auch beim Konkatenieren (=Verbinden) von Werten

- Beide Funktionen entfernen nachfolgende Leerzeichen
- Aber in folgender Situation gibt es einen Unterschied: **Wenn Sie einen Missing ganz entfernen wollen**, denn ein alphanum. Leerzeichen ist ein BLANK.

  - **TRIM** Funktion bei einem Missing:
    Ergebnis hätte ein Blank (=Missing)

  - **TRIMN** Funktion bei einem Missing:
    Würde dagegen wirklich auf „nichts“ kürzen, also auch kein Blank mehr ausgeben.

5. Ein Besuch bei den Verwandten lohnt sich

Funktionsfamilien: COUNT und FIND

Anwendungsszenario

tornado_2017narrative

Narrative
This is a continuation of the northeast Brooks county to strength, it swept about 35 manufactured homes into a pile Seven people lost their lives. The tornado then went on to destroy a collapsing in two walls and removing most of the second story. Another home built of concrete blocks was destroyed. A nearby farm had several concrete anchors for a large metal structure pulled from the ground. Max winds were estimated near 140 mph. Damage cost was estimated
A large, long-track tornado traveled over 70 miles across Dougherty, Worth, Turner, and Wilcox Counties. The tornado ... of Abbeville. The tornado caused significant damage al ... mage was observed along the entire path which was up to ... the path were uprooted or snapped. In Dougherty County, the t ... ed Newton, the tornado was approximately 1.25 miles wide. There wa ... ate damage to a few homes in this area, consistent with EF2 da ... The tornado ... Adam Springs area, destroying nearly every tree in its path and causing EF2 damage to several houses. Most houses in this area had significant damage from falling trees. The tornado then moved through several mobile home parks just west of U.S. 319, destroying many mobile homes and causing the 4 fatalities. Damage consistent with an EF3 tornado was observed just east of U.S. 319. The tornado caused a large portion of a warehouse at the Proctor and Gamble Plant to collapse and tossed several semi-trailers across Mock Road. Additional EF3 damage was observed at the Marine Corp Logistics Base, where multiple anchored double-wide trailers were completely destroyed. In addition, several concrete light poles were snapped near the base, and a large solid concrete building had its solid concrete roof shifted more than 2 inches. A well built concrete block church on Sylvester Rd was

Wie viele Wörter werden durchschnittlich über Tornados geschrieben?

Wie oft wird durchschnittlich EF innerhalb des Textes genannt?

COUNT Funktionen

Zählt wie oft eine angegebene Teilzeichenkette innerhalb einer Zeichenkette vorkommt.	**COUNT(***string***,** *substring* <**,** *modifier(s)*>**)**
Zählt die Anzahl von Zeichen in einer Zeichenkette, die innerhalb einer Zeichen-Liste vorkommen oder nicht.	**COUNTC(***string***,** *character-list* <**,** *modifier(s)*>**)**
Zählt die Anzahl der Wörter	**COUNTW(***string* <**,** *delimiter(s)*> <**,** *modifier(s)*>**)**

COUNT and COUNTW Funktionen

```
NumEF=count(Narrative,'EF');
```

Zeichenfolge EF im String NARRATIVE zählen

```
NumWord=countw(Narrative,' ');
```

Wörter im String NARRATIVE zählen. Ein neues Wort beginnt immer nach einem Blank

- Dieses Argument gibt an: Nur das Leerzeichen ist das Trennzeichen zum Trennen von Wörtern.
- Mehrere Trennzeichen können angegeben werden.
- Ohne Angabe von Trennzeichen nimmt SAS standardmäßig folgende Trennzeichen an:
  blank ! $ % & () * + , - . / ; < ^ |

Die unbekannteren Funktions-Verwandten

COUNTC-Funktion

Zählt die Anzahl von Zeichen in einer Zeichenkette, die in einer Liste vorkommen oder nicht.

COUNTC(String, ‘Zeichenliste‘ <, ‘Modifier‘> ;

Unterschied zur COUNT-Funktion: COUNT zählt **Teilzeichenketten/Wörter.**

Ergebniszählung

3

0

6

1

```
data testcountc;
 text='das ist der SAS Code   ';
 countc_ae=countc(text,'ae');
 count_ae=count(text,'ae');
 countc_das=countc(text,'das');
 count_das=count(text,'das');
 run;
```

FIND Funktionen

Gibt die Startposition an, wo eine Teilzeichenkette gefunden wird.	**FIND(***string***,** *substring* <, *modifier(s)*> <, *start-position*>**)**
Gibt die Startposition an, wo ein Zeichen aus einer Zeichenliste innerhalb einer Zeichenkette gefunden wird.	**FINDC(***string***,** *character-list* <, *modifier(s)*> <, *start-position*>**)**
Gibt die Startposition an, wo ein Wort gefunden wird ODER das wievielte Wort es ist.	**FINDW(***string***,** *word***,** <, *delimiter(s)*> <, *modifier(s)*> <, *start-position*>**)**

Die FIND-Funktion – Beispiel

21

```
data find;
   Text='AUSTRALIA, DENMARK, US';
   Pos1=find(Text,'US');
   Pos2=find(Text,' US');
   Pos3=find(Text,'us');
   Pos4=find(Text,'us','I');
   Pos5=find(Text,'us','I',10);
run;
```

Pos1	Pos2	Pos3	Pos4	Pos5
N 8	N 8	N 8	N 8	N 8
2	20	0	2	21

Hilfreiche Modifier für die FIND Funktionsfamilie

Diese Modifier + Startposition sind bei allen Find Funktionen vorhanden.

- I ignoriert Groß-/Kleinschreibung
- T ignoriert nachfolgende (trailing) Leerzeichen in den Werten von Zeichenfolge und Teilzeichenfolge

Startpos gibt an, ab welcher Position in der Zeichenfolge mit der Suche nach der Teilzeichenfolge begonnen werden soll.

ACHTUNG:

Die FIND Funktion hat sehr wenige Modifier vs FIND**C** oder FIND**W** Funktion.

FINDW Funktion

Viele Zeichen können als Delimiter angegeben werden.

```
EFWordNum=findw(Narrative,'EF','012345- .,','e');
```

e oder E Modifier:
Zählt die Wörteranzahl, bis das angegebene Wort gefunden wird und gibt nicht die Startposition des gefundenen Wortes aus.

Auch das geht!

Zeichen finden, die NICHT enthalten sind: Modifier V

Sie wollen z. B. wissen, ob andere Zeichen enthalten sind als die angegebenen („das ist“).

Rückgabewert= Startposition, wo das erste andere Zeichen steht.

```
Wert='das ist ein langes Beispiel';
Ergebnis=findc(wert,'das ist','v');
```

Wert	Ergebnis
das ist ein langes Beispiel	9

Ausgewählte Modifier

FindC und FindW Funktion

FINDC Funktion

b/B: Suche startet von rechts, nicht von links

k/K: sucht nach Zeichen, die nicht in der Zeichenliste stehen.

FINDW Funktion

e/E: Zählt die Wörter bis zum angegebenen Wort (nicht die Position)

k/K: Bewirkt, dass alle Zeichen, die nicht in der Liste stehen, als Trennzeichen behandelt werden

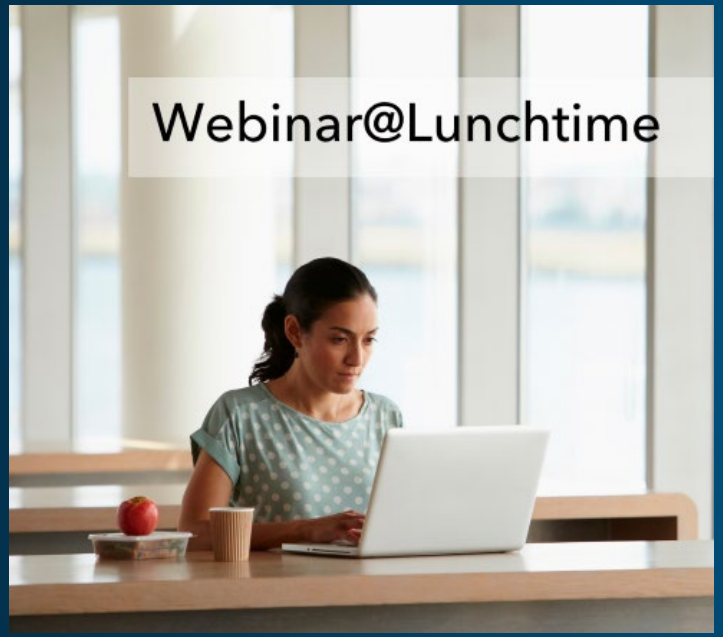
q/Q: Ignoriert Delimiter innerhalb von Teilzeichenketten, die in Anführungszeichen stehen.

Wichtige Punkte bei alphanumerischen Funktionen

Hier zwar nicht behandelt, aber generell interessant – auch für alphanumerische Werte.

Zusammenfassung

- Funktions-Modifier: Können Ihr Programmierleben erleichtern
- Funktionsfamilien wegen der Feinheiten berücksichtigen
  (z. B. COUNT, COUNTC, COUNTW)
- Mögliche Standardlänge beachten bei Erstellung neuer Variablen: z. B. 200 B bei FIND, TRANWRD u.a.
  .
- Gleiche Funktionsweise aber für unterschiedliche Datentypen, z. B. COALESCE**C** oder COALESCE, IF**C** oder IFN, CHOOSE**C** oder CHOOSE …



Nächstes Webinar@Lunchtime:

17. Februar 2022

Rund um das Thema MISSINGS

www.sas.de/lunchtime