Eine simple Gedankenstütze zur richtigen Anwendung von SAS Formaten/Informaten

Matthias Lehrkamp
PAREXEL International
Spandauer Damm 130
14050 Berlin
matthias.lehrkamp@parexel.com

Zusammenfassung

Daten-Standardisierungen wie z.B. CDISC distanzieren sich immer mehr von SAS-Formaten und bevorzugen die Erstellung neuer Variablen. Trotz dessen erfreut sich die Anwendung von SAS-Formaten bzw. -Informaten mit ihren zahlreichen Möglichkeiten großer Beliebtheit, insbesondere bei der Typenkonvertierung von Variablen. Hierbei entstehen Fehler, weil das Konstrukt der SAS-Formate bzw. -Informate im Allgemeinen nicht verstanden wird und die Anwendung immer mit Fragen verbunden ist.
Dieser Artikel wird Ihnen am Ende eine Gedankenstütze mitgeben, mit der SAS-Formate bzw. Informate in Zukunft leicht zu handhaben sind. Denn einer der häufigsten Fehler bei der Anwendung von SAS-Informaten sorgt auch heute noch – und das weltweit – für Schockzustände die es zu vermeiden gilt.

Schlüsselwörter: FORMAT, INFORMAT, INPUT, PUT, Mapping, Validierung, CDISC, PROC FORMAT, w.d Informat

1 SAS Formate in der klinischen Forschung

Elektronischer Speicherplatz ist seit Längerem kostengünstig. Daher steigt der Wunsch nach direkt lesbaren Daten, das heißt, es sollten nur allgemein bekannte Abkürzungen benutzt werden. Vor allem sollen keine Formate in Datensätzen verwendet werden. Stattdessen werden Codes und vollständige Namen in separaten Variablen abgespeichert. Formate wurden früher gerne verwendet, da die Daten speicherplatzsparend abgelegt werden konnten und beim Betrachten der Daten voll lesbar waren. Oft gab es aber Probleme mit dem Mitliefern und dem Einlesen der Formate. Im Moment wird noch das SAS Transportfile Format XPT [1] in Version 5 am häufigsten verwendet, um klinische Daten zu übertragen. Dieses Format ist ein offener Standard und wurde von SAS entwickelt. Damit können die Datensätze mit entsprechender Software auf fast jedem Computer gelesen werden. Die Bestrebungen zielen aber darauf ab, die Daten in einer ASCII Datei zu speichern, so dass die Daten mit jedem Editor lesbar sind. Das Clinical Data Interchange Standards Consortium (CDISC [2]) hat eine feste Struktur entwickelt, um klinische Daten zu speichern, genannt Study Data Tabulation Model (SDTM). Dieses Modell ist so festgelegt, dass gleiche Daten in unterschiedlichen Studien auf dieselbe Weise gespeichert werden und daher schneller auffindbar sind. Am 22.

April 2014 hat CDISC die Spezifikation Dataset-XML [3]veröffentlicht, um Daten als XML (Extensible Markup Language) Dateien zu speichern. Dieses Dateiformat lässt sich mit jedem Editor öffnen und kann mit einer zusätzlichen Datei, einem sogenannten Schema, leserlich aufgehübscht werden.
Ob nun das XPT oder das XML Format verwendet wird, in beiden Fällen ist das Verwenden von Formaten untersagt. Trotzdem sind Formate sehr nützlich und können die Bearbeitung der Daten erheblich vereinfachen. Die Hauptanwendungsgebiete liegen beim Daten-Mapping und der Daten-Validierung, sowie bei der Typenkonvertierung von Zahlenwerten.

2 SAS Formate

SAS Formate bestimmen die Darstellung der einzelnen Werte. Im Datensatz bleiben die ursprünglichen Werte erhalten. Bei Abfragen müssen daher die Werte im Datensatz verwendet werden, während beim Öffnen des Datensatzes oder beim Herausschreiben der Daten die Label des Formats verwendet werden. Das ist natürlich eine zusätzliche Fehlerquelle, da der Betrachter die richtigen Werte nur sieht, wenn er die Formateinstellung ändert. In diesem Abschnitt wird auf die Erstellung eigener Formate eingegangen, SAS eigene Formate vorgestellt und die `PUT` Funktion beschrieben.

2.1 Eigene Formate mit PROC FORMAT erstellen

Eigene Formate können über das `VALUE` Statement in **`PROC FORMAT`** definiert werden. Die Verwendung von Formaten ist nur im `PUT` Statement, in der `PUT` Funktion, der `%SYSFUNC` Makro Funktion, in einigen Prozeduren als Option (`FORMAT=`), sowie im `FORMAT`- und `ATTRIB` Statement möglich.

Die Zuweisung eines Formats erfolgt über seinen Namen: <$>*format*<*w*>.<*d*>

format: Der Formatname besteht aus einer Zeichenkette, die der „SAS Name“ Konvention genügt und nicht länger als 32 Zeichen ist. Das Dollarzeichen „$“ zählt zu den 32 Zeichen hinzu. Der erste Buchstabe muss also ein Buchstabe oder ein Unterstrich „_“ sein. Nachfolgende Zeichen können zusätzlich Zahlen enthalten. Im Unterschied zu Variablennamen, darf ein Formatname nicht auf eine Zahl enden.

$: Zusatz, um Character-Formate zu kennzeichnen. Character-Formate haben als Eingangswert ein Character-Wert.

w: Länge der Zeichenkette (maximal 32 inklusive $)

d: Dezimal-Skalierungsfaktor, bei numerischen Formaten ist es die Anzahl der Nachkommastellen.

In dem folgenden Codebeispiel wird ein Character-Format `$country` und ein numerisches Format `country` mit Hilfe der Prozedur **`PROC FORMAT`** erstellt. Damit der Unterschied zwischen einer Variablen mit und ohne Format sichtbar wird, werden die Variablen zuvor kopiert und die Formate auf den neu erzeugten Variablen angewendet. Die

Zuweisung der Formate erfolgt über das **FORMAT** Statement. Zusätzlich wird eine Variable `check` angelegt, um zu testen, welcher Wert in Abfragen benutzt werden muss.

`country`

	subjid	country	countryn
1	1001	AUT	040
2	2001	DEU	276
3	2002	DEU	276
4	3001	BEL	056
5	4001	ITA	380

```
/* create formats */
PROC FORMAT;
  VALUE $country
    "AUT" = "Austria"
    "DEU" = "Germany"
    "ITA" = "Italy";
  VALUE country
    040 = "AUT"
    276 = "DEU"
    380 = "ITA";
RUN;

/* Set country formats */
DATA country_format;
  SET country;
  FORMAT cfl $country. cfn country.;
  cfl= country;
  cfn= countryn;
  IF cfl="Italy" THEN check= 1;
  ELSE IF cfl="ITA" THEN check= 2;
RUN;
```

`country_format`

	subjid	country	countryn	cfl	cfn	check
1	1001	AUT	040	Austria	AUT	.
2	2001	DEU	276	Germany	DEU	.
3	2002	DEU	276	Germany	DEU	.
4	3001	BEL	056	BEL	56	.
5	4001	ITA	380	Italy	ITA	2

Wie im Ausgabedatensatz `country_format` ersichtlich ist, reagiert SAS nicht auf das im Format definierte Label `"Italy"`, sondern auf den gespeicherten Originalwert `"ITA"`.

Für alle Werte, die nicht im Format mit angegeben sind, wird der Originalwert übernommen. Im Beispiel Code wurde BEL (056) für Belgium nicht umgewandelt, da BEL (056) nicht in den Formaten enthalten ist. Die Variable `check` enthält für die 5. Beobachtung eine 2, das heißt, dass die Abfrage ausschließlich mit `cfl="ITA"` funktioniert. Bei Bedingungen muss also auf den Originalwert abgefragt werden.

2.2 SAS interne Formate

Neben der Möglichkeit eigene Formate zu erstellen, gibt es bereits eine Menge Formate, die von SAS bereitgestellt werden. Die wichtigsten Formate um numerische Werte darzustellen, sind in Tabelle 1 zusammengefasst.

Tabelle 1: SAS interne Formate – Zahlenformate

	Format	Wert	Ausgabe	Kommentar
1	beat5.	100.567	100.6	SAS wählt die beate Notation.
2	8.1	100.567	100.6	*Anzahl-Zeichen.Dezimalstellen*
3	z8.1	100.567	000100.6	Mit führenden Nullen
4	comma8.1	2345.67	2,345.7	Mit Tauaendertrennzeichen
5	dollar9.2	2345.67	$2,345.67	Mit Währung

Ein Datum wird in SAS als Zahl abgespeichert. Diese gibt die Anzahl der Tage an, die zwischen dem 1. Januar 1960 und dem festgelegtem Datum liegen. Demnach wird der 30. Dezember 1959 als -2, der 1. Januar 1960 als 0 und der 3. Januar 1960 als 2 gespeichert. Die Speicherung als Zahl hat den Vorteil, dass mit einem Datum gerechnet werden kann und die Speicherung weniger physikalischen Speicherplatz benötigt. Da die Zahlendarstellung für uns schlecht lesbar ist, bietet SAS einige Formate an, um das Datum lesbar zu machen. Die wichtigsten sind in Tabelle 2 abgebildet.

Tabelle 2: SAS interne Formate – Datumsformate

	Format	Wert	Ausgabe	Kommentar
1	date9.	20173	26MAR2015	Standard in Analyaen
2	b8601da8.	20173	20150326	geeignet für Dateinamen
3	ia8601da.	20173	2015-03-26	CDISC SDTM Vorgabe
4	ddmmyyp10.	20173	26-03-2015	Europäiachea Format
5	mmddyy10.	20173	03/26/2015	USA
6	downame.	20173	Thuraday	Wochentag
7	monname.	20173	March	Monat

Uhrzeiten werden ebenfalls numerisch abgebildet. Als Zahl werden die Sekunden, die seit Mitternacht verstrichen sind, angegeben. Um kleinere Zeiteinheiten darzustellen, werden Gleitkommazahlen verwendet. Die Kombination von Datum und Uhrzeit ist ebenfalls möglich. Dabei werden die Sekunden, die zwischen dem 1. Januar 1960 um 00:00 Uhr und dem festgelegten Zeitpunkt liegen, erfasst. Eine Interaktion zwischen

einem Datum (Tage) und einem Datum mit Uhrzeit (Sekunden) ist wegen der unterschiedlichen Speicherung (Tage und Sekunden) nicht möglich!

2.3 Die PUT Funktion

Die `PUT` Funktion ermöglicht das Konvertieren von Werten mit Hilfe von SAS Formaten, aber auch eigenen Formaten. Die Formatlabel können somit in einer neuen Variable gespeichert werden. Die Funktion hat zwei Eingabeparameter.

PUT(*expression*, *format.* <L | C | R>)

expression: Konstante, Variable oder ein Ausdruck, deren Wert umformatiert werden soll.

format: Name des Formats, optional kann eine Ausrichtung festgelegt werden.

Die `PUT` Funktion kann ausschließlich mit Formaten benutzt werden und der Rückgabewert ist immer vom Typ Character. Mit der Funktion `PUT` und einem Format können numerische oder Character-Werte in Character umgewandelt werden. Zum Format kann optional die Ausrichtung (L – links ausgerichtet, C – zentriert, R – rechts ausgerichtet) der Ausgabe festgelegt werden. Die Standardausrichtung für Character-Werte ist links (L) und für numerische Werte ist rechts (R).

```
DATA testput;
  x= 20173;
  y= PUT(x, date9.);
  z= PUT(x, 8.);
  OUTPUT;
  z= PUT(x, 8. -L);
  OUTPUT;
RUN;
```

testput

	x	y	z
1	20173	26MAR2015	20173
2	20173	26MAR2015	20173

3 SAS Informate

SAS Informate sind zum Interpretieren von eingelesenen Daten konzipiert worden. Gespeichert werden somit die Informat Label nicht der eingelesene Wert. Mit dem entsprechendem Informat wird also aus dem „M“ ein „Male“ und aus dem „F“ ein „Female“. Diese Art der Verwendung findet praktisch nicht statt, da der Originalwert dabei verlorengeht. Eine Fehlinterpretation ist somit nicht mehr nachvollziehbar. In der Praxis werden alle Werte, so wie sie eingegeben werden, eingelesen und gespeichert und erst hinterher angepasst. In der klinischen Forschung erhält man somit zwei Datenbanken. Zum einen den Datenextrakt aus der Originaldatenbank und zum anderen die Datenbank zum Einreichen der Studiendaten. Letzteres ist entweder SDTM von CDISC oder ein hausinternes Firmenformat. Eine Ausnahme für die Verwendung von Informaten ist das Einlesen von Textdateien mit PROC IMPORT, dabei erzeugt SAS einen DATA-Step der mit Informaten gefüttert wird. Der Programmcode wird zum Nachvollziehen im Log

ausgegeben. Wer sich gut mit Informaten auskennt, kann diesen Code nutzen und entsprechend anpassen.
Informate sind aber in Kombination mit der INPUT Funktion besonders nützlich. Mit deren Hilfe können Zahlen, die als Text gespeichert sind, ins numerische umgewandelt werden. Zusätzlich können die Daten dabei validiert werden.

3.1 Eigene Informate mit PROC FORMAT erstellen

Eigene Informate lassen sich ebenfalls über PROC FORMAT erstellen, jedoch muss für Informate das INVALUE Statement verwendet werden.

```
/* create informats */
PROC FORMAT;
  INVALUE $country
    "AUT" = "Austria"
    "DEU" = "Germany"
    "ITA" = "Italy";
  INVALUE country
    "AUT" = 040
    "DEU" = 276
    "ITA" = 380;
RUN;
```

Die Verwendung von Informaten ist auf das INPUT Statement, die INPUT Funktion, dem INFORMAT- und ATTRIB Statement beschränkt.

Die Zuweisung eines Informats erfolgt über dessen Namen: <$>*informat*<*w*>.<*d*>

informat: Der Informatname besteht aus einer Zeichenkette, die der „SAS Name“ Konvention genügt und nicht länger als 32 Zeichen ist. Das Dollarzeichen „$“ zählt zu den 32 Zeichen hinzu. Der erste Buchstabe muss also ein Buchstabe oder ein Unterstrich „_“ sein. Nachfolgende Zeichen können zusätzlich Zahlen enthalten. Im Unterschied zu Variablennamen darf ein Informatname nicht auf eine Zahl enden.

$: Zusatz, um Character-Informate zu kennzeichnen. Bei Character-Informaten ist der Ausgabewert vom Typ Character.

w: Länge der Zeichenkette (maximal 32 inklusive $)

d: Dezimal Skalierungsfaktor, bei numerischen Informaten wird bei ganzen Zahlen durch 10^d dividiert!

Wer jetzt an einen Schreibfehler bei der Beschreibung von *d* glaubt, ist einem Irrtum aufgesessen. Der nächste Abschnitt lüftet den wohl am häufigsten vorkommenden Anwendungsfehler bei der Anwendung von SAS-Informaten weltweit.

3.2 Häufigster Anwendungsfehler bei der Anwendung von SAS-Informaten

Tatsächlich kommt es durch die kuriose Festlegung des Parameters *d* zum wohl häufigsten Anwendungsfehler von SAS-Informat Anweisungen weltweit. Das folgende Beispiel liest vier verschiedene Zahlen ein. Zwei Zahlen im Hunderterbereich, eine Zahl im Zehnerbereich und eine ganze Zahl, nämlich die 7. Alle Zahlen werden als Textwert eingelesen und sollen ins numerische konvertiert werden. Getestet wird die `INPUT` Funktion mit vier verschiedenen Informaten.

```
DATA country;
  INPUT numchar $8.;
  input62= INPUT(numchar, 6.2);
  input6= INPUT(numchar, 6.);
  input52= INPUT(numchar, 5.2);
  input5= INPUT(numchar, 5.);
  DATALINES;
123.45
7
23.58
300.75
;
```

Der geübte SAS Anwender weiß, dass der Punkt als Zeichen mitgezählt werden muss. Damit entfallen schon mal die beiden angegebenen Informate `5.2` und `5.`, da deren Zeichenlänge für die Zahlen `123.45` und `300.75` zu kurz ist. Da maximal 2 Nachkommastellen vorhanden sind, ist auch schnell der Parameter *d* festgelegt, nämlich 2.

	numchar	input62	input6	input52	input5
1	123.45	123.45	123.45	123.4	123.4
2	7	0.07	7	0.07	7
3	23.58	23.58	23.58	23.58	23.58
4	300.75	300.75	300.75	300.7	300.7

Die Ausgabe der Informate `5.2` und `5.` führt wie erwartet zum Abschneiden der beiden Zahlen `123.45` und `300.75`, da wegen des Trennzeichens nur vier Zahlenwerte eingelesen werden können. Auch zu sehen ist, dass bei dem Informat `6.2` die Zahl `7` (2. Zeile) nicht richtig eingelesen wurde. Die Zahl wurde fälschlicherweise als `0.07` interpretiert. Das liegt aber an der Festlegung des Parameter *d*. Die SAS Hilfe zum `w.d` Informat [4] (Alias: BEST*w.d*, D*w.d*, E*w.d*, F*w.d*) schreibt zum Parameter *d* folgendes: „*specifies the power of 10 by which to divide the value. If the data contain decimal points, the d value is ignored*“. Übersetzt bedeutet das, dass die eingelesene Zahl durch 10^d dividiert wird. Der Parameter *d* wird allerdings ignoriert, sofern die Zahl ein Dezimaltrennzeichen enthält. Im Klartext, ganze Zahlen werden durch 10^d dividiert, der Rest bleibt wie er ist. Da 7 die einzige ganze Zahl in diesem Beispiel ist, entsteht genau hier der Fehler. Beim Einlesen von Lochkarten (bis 1990) war dieses Informat sicherlich

ganz nützlich, bis heute hat sich jedoch niemand getraut, dieses Informat zu korrigieren, es ist in der Beschreibung weiterhin erhalten geblieben. Da fallen mir doch gleich die Worte ein: „*It's not a bug, it's a feature*“. Und da der Mix von ganzen und reellen Zahlen nicht so häufig vorkommt, ist es auch entsprechend schwer, diesen Fehler zu entdecken.

3.3 Beispiel mit eigenen Informaten

In dem folgenden Codebeispiel wird ein Character-Informat `$country` und ein numerisches Informat `country` mit Hilfe der Prozedur `PROC FORMAT` und dem `INVALUE` Statement erstellt. Die Ländercodes mit 3 Zeichen werden wie beim Format umgewandelt. In folgendem Beispiel jedoch vom Code zur Zahl, sowie beim Format vom Code zum Langnamen. Damit der Unterschied zum Format sichtbar wird, wird eine Variable `check` angelegt, um zu zeigen, dass sich beim Informat die Werte ändern und nicht nur ein Label draufgesetzt wird.

```
/* read data using informats */
DATA country;
  INPUT subjid $4.
     @6 cfl $country.
     @6 cfn country.;
  FORMAT cfn z3.;
  IF cfl="Italy" THEN check= 1;
  ELSE IF cfl="ITA" THEN check= 2;
  DATALINES;
1001 AUT
2001 DEU
2002 DEU
4001 ITA
;
```

country_format

	subjid	cfl	cfn	check
1	1001	Austria	040	.
2	2001	Germany	276	.
3	2002	Germany	276	.
4	4001	Italy	380	1

Die Variable `check` enthält dieses Mal die 1, das heißt, dass die Abfrage mit `cfl="Italy"` bereits zum Erfolg führt. Die eingelesenen Werte werden also durch die Werte im Informat ersetzt.
Aufgrund der numerischen Ausgabewerte des Formates `country` ist die Variable `cfn` numerisch, wurde aber zusätzlich mit einem Format z3. versehen, damit die führenden Nullen mit ausgegeben werden.

3.4 SAS interne Informate

Zu den meisten Formaten gibt es auch ein passendes Informat. Aber vorsichtig, die Formate und Informate können durchaus voneinander abweichen, wie das Beispiel zu den *w.d* Informat im Abschnitt 3.2 bereits gezeigt hat. In Tabelle 3 gibt es einen Überblick über die wichtigsten Informate.

Tabelle 3: Die wichtigsten Informate

	Informat	Input	Ausgabe	Kommentar
1	$5.	␣␣xY␣	xY␣␣␣	Einlesen von Zeichen
2	$char5.	␣␣xY␣	␣␣xY␣	Einlesen mit Leerzeichen
3	8.	100.567	100.567	Zahlen mit 8 Zeichen
4	is8601da.	2015-03-26	20173	Liest ISO 8601 Datum (Tage)
5	is8601dt.	2015-03-26T11:30	1742988600	Liest ISO 8601 Datum/Zeit (s)
6	anydtdte8.	03/26/14	26MAR2014	Verschiedene Datumsformate

Das Informat anydtdte*w*. kann ein Datum oder ein Datum mit Uhrzeit in den unterschiedlichsten Formaten einlesen und ist daher besonders nützlich.

3.5 Die INPUT Funktion

Die `INPUT` Funktion ermöglicht das Konvertieren von Werten mit Hilfe von SAS Informaten. Ausgabe kann je nach dem Informat numerisch oder character sein. Der Eingabewert muss aber immer vom Typ Character sein.

INPUT(*expression*, <? | ??> *informat*.)

expression: Character-Konstante, Variable oder Ausdruck, deren Wert umformatiert werden soll.

format: Name des Informats, mit Hilfe der optionalen Fragezeichen können Fehlermeldungen im Log (?) und die Fehlersetzung _ERROR_=1 (??) unterdrückt werden.

Die `INPUT` Funktion arbeitet ausschließlich mit Informaten, wobei der Eingabewert immer vom Typ Character sein muss. Mit der Funktion `INPUT` und einem Informat ist es möglich, Character-Werte ins numerische umzuwandeln oder die Werte in andere Textwerte zu mappen. Da Informate zum Einlesen von Daten vorgesehen sind, werden diese Daten überprüft und gegebenenfalls Fehler ausgegeben. Diese können mit Hilfe der optionalen Fragezeichen unterdrückt werden.

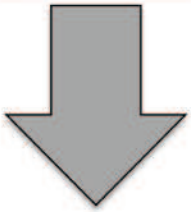
Bei der Benutzung von einem Fragezeichen (`?`) wird lediglich die Zeile mit der NOTE aus dem Log entfernt. Die Datenzeile im Log und das setzen der `_ERROR_` Variable im PDV wird trotzdem durchgeführt. Erst beim Setzen von zwei Fragezeichen (`??`) wird auch die Datenzeile und das Setzen der Fehlervariable im PDV unterdrückt. Da es bei SQL keine Fehlervariable gibt und auch keine Datenzeile, wird hier nur ein Fragezeichen (`?`) benötigt.

```
 /* create informat:
    3 letter ISO country code
    to long country name */
PROC FORMAT;
  INVALUE $country
    "AUT" = "Austria"
    "DEU" = "Germany"
    "ITA" = "Italy"
    OTHER = _ERROR_;
RUN;

/* Convert 3 letter ISO country code
   into long country name */
DATA countryLong;
  SET country;
  countryl= INPUT(country,$country.);
RUN;
```

country

	subjid	country
1	1001	AUT
2	2001	DEU
3	2002	DEU
4	3001	BEL
5	4001	ITA



countryLong

	subjid	country	countryl
1	1001	AUT	Austria
2	2001	DEU	Germany
3	2002	DEU	Germany
4	3001	BEL	
5	4001	ITA	Italy

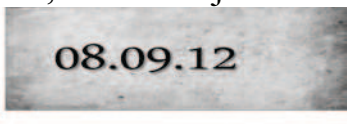
Log:

```
181  DATA countryLong;
182    SET country;
183    countryl= INPUT(country
                       ,$country.);
184  RUN;

NOTE: Invalid argument to function INPUT at line 183 column 13.
subjid=3001 country=BEL countryl=  _ERROR_=1 _N_=4
```

4 Die Gedankenstütze

Um sich zu merken, wann und wie welche Funktion und welches Format angewendet wird, muss man sich nur in die Lage von SAS versetzen. Wir stellen uns also vor, wir wären jetzt der Rechner, bzw. SAS. Als Rechner können wir natürlich mit character und numerischen Werten umgehen. Die Werte-Eingabe und die Werte-Ausgabe passiert über ein Blatt Papier, also immer nur als Character-Wert. Welche Information benötigen wir, wenn wir jetzt ein Blatt Papier mit dem folgenden Wert finden?

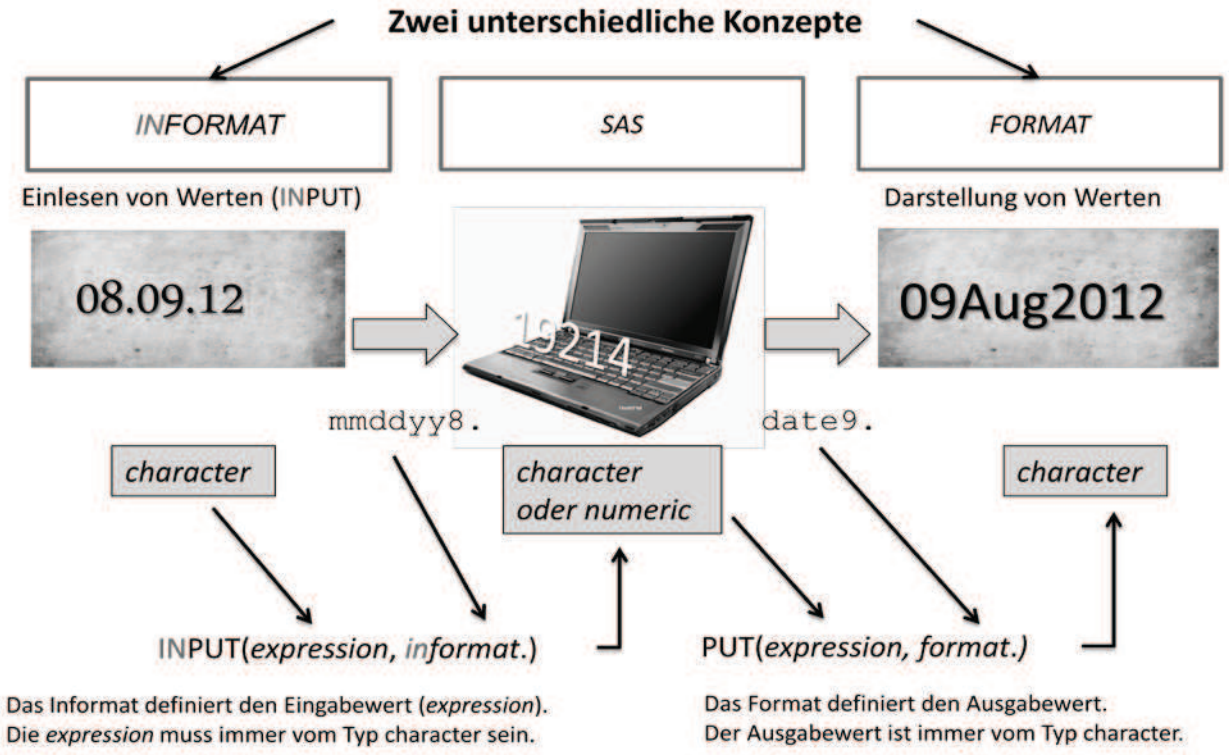


Richtig, wir müssen die Formatierung des Wertes auf dem Papier kennen. In diesem Fall ist es ein Datum im Format mmddyy (zuerst der Monat, dann der Tag und zum Schluss das Jahr). Auf dem Zettel steht also das Datum 09Aug2012. Ein Datum speichern wir am liebsten als Zahlenformat ab, damit wir mit dem Datum auch rechnen können. Der INPUT Funktion müssen wir also den Wert und das Format des Wertes übergeben.

```
x= INPUT("08.09.12", mmddyy8.);
```

Bei der Ausgabe des Wertes stellen wir uns am besten einen Drucker vor. Dieser gibt nur Papier aus, also nur Character-Werte. Damit nicht immer alle Ausgaben gleich aussehen, müssen wir über das Format bestimmen, wie die Ausgabe aussehen soll. Der `PUT` Funktion müssen wir daher den Wert und die gewünschte Formatierung für die Ausgabe mitgeben.

```
y= PUT(x, date9.); → Ausgabe: 09AUG2012
```



5 Nachtrag

Es soll nicht unerwähnt bleiben, dass es ebenfalls möglich ist, Formate mit numerischen Output, sowie Informate mit numerischen Input zu erstellen. Die so erstellten Formate können aber nur in einigen Prozeduren benutzt werden. Die Informate hingegen können nur beim Einlesen von Daten verwendet werden. Beim Konvertieren der Daten wird die Format/Informat-Auswahl durch die `PUT` und `INPUT` Funktion beschränkt. Da diese Format/Informat nur sehr selten verwendet werden und für Anfänger ungeeignet sind, wurde in diesem Artikel nicht näher darauf eingegangen. Informationen und Beispiele gibt es in der SAS Hilfe zu PROC FORMAT, für Formate der Unterpunkt „VALUE Statement“ [5] und für Informate der Unterpunkt „INVALUE Statement“ [6].

6 Zusammenfassung

Die Eingabe (Input) ist immer vom Typ Character, und kann mit der Funktion `INPUT` durchgeführt werden. Der Funktion muss der Eingabewert und dessen Format (Informat) übergeben werden. Unbedingt beachten sollte man, dass Informate sich durchaus von den Formaten, trotz gleicher Namensgebung, unterscheiden können. Eigene Informate können über das `INVALUE` Statement erstellt werden. Alle Eingabehilfen (`INFORMAT`, `INPUT`, `INVALUE`) enthalten stets das IN am Namensanfang.

Bei der Ausgabe von Werten muss die `PUT` Funktion benutzt werden. Mit Hilfe eines Formats wird beschrieben, wie der Ausgabewert aussehen soll. Dieser ist stets vom Typ Character. Eigene Formate werden mit dem `VALUE` Statement erzeugt.

Um mit Formaten einfach umgehen zu können, muss man sich nur in die Lage von SAS versetzen und vielleicht noch an das Beispieldatum 08.09.12 denken. Dies nur als kleine Gedankenstütze ;-)

Literatur

[1] SAS XPT Spezifikation: https://support.sas.com/techsup/technote/ts140.pdf

[2] CDISC: http://www.cdisc.org/

[3] Dataset-XML Standard: http://www.cdisc.org/dataset-xml

[4] SAS w.d Informat: http://support.sas.com/documentation/cdl/en/leforinforref/64790/HTML/default/viewer.htm#n14sqpf1cubqknn1vmkzkp1oph87.htm

[5] SAS Value-Statement: http://support.sas.com/documentation/cdl/en/proc/67327/HTML/default/viewer.htm#p1upn25lbfo6mkn1wncu4dyh9q91.htm

[6] SAS Invalue-Statement: http://support.sas.com/documentation/cdl/en/proc/67327/HTML/default/viewer.htm#p1pmw90bl3jzgdn1w4202kclxtho.htm