Sorting Revisited

by Winfried Gerum

Winfried Gerum

Abstract

The implicit collation in the M language does not always yield a sequence of indices according to user expectations. Under the heading of *internationalization,* the MDC (MUMPS Development Committee) has provided a handle to escape the traditional limitations. The MDC work so far does not help a typical user with her collation problems. Here we present a Generic Key Transform that makes it very easy to specify a wide range of collation types. This can be used with or without the MDC framework.

Careful design of the MUMPS language provides us with a wide range of features that superficially look simple. Behind that simple facade, there is powerful functionality. One of the most popular features is the implicit collating in local and global arrays. Numeric keys are automatically sorted according to their numeric value. Non-numeric keys are sorted "alphabetically." But this "alphabet" is the ASCII character set. In this alphabet, uppercase and lowercase characters are quite different. As long as keys are numeric-only or strictly words of the same case, the result is perfect. But with entries of mixed case, non-English characters, etc., there may be some surprises.

In a previous *Tips 'N' Tricks* column ("How To Sort McMUMPS and other Strange Guys"), I discussed the issues and presented a generic algorithm to transform keys to an alternate format that sorts according to user expectations with the traditional MUMPS collation scheme. That **Generic Key Transform** allows you to specify that some characters (or combination of characters) sort like a different character (or combination of characters). That proved to be quite useful: You can specify that uppercase characters sort as lowercase characters and it allows you to specify that characters like ä sort like ae, etc.

The Generic Key Transform, as published in the former article, did not allow you to specify that a character (or combination of characters) should sort like no character. And, as a reader pointed out, it did not provide the expected collation for strings with embedded numbers.

The MUMPS Development Committee has introduced several new features to the MUMPS language to facilitate *Internationalization* of MUMPS software. Regarding the collation of MUMPS globals, things are as simple as: SET ^$GLOBAL(gvn, "COLLATE") = algoref with ^$GLOBAL being a Structured System Variable (ssvn), gvn a global, and algoref describing a function such that for every key X the expression @(algoref_"(X)") yields a unique value to be used for sorting. The actual storage may use the plain key or the transformed key. These details are not visible to the application program. It works like magic, if your MUMPS implementation supports that and if you can construct a function with the desired property. The transformed key is not subject to the string length limitations that may apply for plain keys.

Even if your MUMPS application does not support this ^$GLOBAL, you can reap the benefits of such a sophisticated collation scheme if you use the key transform on the application level. Albeit, without absolution from the string length restriction.

Anyway, the problem is how to construct such a key transform. Note that the required uniqueness of the key transform guarantees the existence of a reverse transform.

One way is to provide ad hoc solutions for particular problems (e.g. you want to ignore case in alphabetic characters, and you want to ignore all non-alphabetic characters). Then, you could specify a function:

```
Al(X) QUIT $TRANSLATE(X,"ABCDEF
GHIJKLMNOPQRSTUVWXYZ"_X,
"abcdefghijklmnopqrstuvwxyz")_
$CHAR(n)_X
```

with (0 ≤ n < ($ASCII("a"))) and $C(n) being permitted in subscripts. The uniqueness is obvious and the reverse transform is as simple as:

```
RevAl(X) QUIT $PIECE(X,$CHAR(N)
, 2,$LENGTH(X,$CHAR(n)))
```

If the ASCII character set is the only character set you will ever use, this function A1 may be all you need to solve your collation problems. But complexity lurks behind many innocent-looking things.

The issues:

- Sort uppercase like lowercase characters
- Ignore punctuation in sorting
- Sort embedded numbers according to numeric values
- Sort "Mc" in names like "Mac"
- Sort ä, Ä, æ, Æ like ae

Look at:

MUMPS:	note-taking, notetaker, noteworthy
Human:	notetaker, note-taking, noteworthy

Here the "-" should not have a collation value.

One reader pointed out that there is another shortcoming. If a number is embedded within a string, you get funny results:

MUMPS:	Filel, File10, Filell, File2, File3, . . .
Human:	Filel, File2, File3, File10, Filell, . . .

And if you sort Roman numbers (or strings containing Roman numbers) the results are even less useful:

MUMPS:	i ii iii iv ix v vi vii viii x xi xii xiii xiv xix xv xvi xvii xviii xx
Human:	i ii iii iv v vi vii viii ix x xi xii xiii xiv xv xvi xvii xviii xix xx

And what about numbers being represented as words?

MUMPS:	chapter five, chapter four, chapter one, chapter six, chapter three, chapter two
Human:	chapter one, chapter two, chapter three, chapter four, chapter five, chapter six

There is no cure-all, as some strings may be ambiguous: "mix" may be an English word or a Roman number (1009), "dix" may be the French word for 10 or a Roman number (509).

However, the *Generic Key Transform* can be enhanced to accommodate some of the additional requirements presented here:

The basic idea remains the same: The substrings of the key are replaced by counterparts with the desired collation property. A second string is prepared that provides information to do the backward transform.

It is not easy to accommodate numeric values in this context. We have to transform a numeric value in such a way that the string representation has the desired collation property. When we realize that a numeric value may have any number of digits before or after the decimal point, it soon becomes evident that we cannot find a transform for all possible numbers. That holds true, even when "all possible" means "all canonic numbers subject to a string length limitation."

As long as we settle for numbers within a predefined range of digits before and after the decimal point, everything is fine. We simply do a $JUSTIFY according to the selected number of digits and $TRANSLATE spaces to zeroes. That does not do the trick for negative numbers. There you have to take the tens complement to get the right property. The fixed format requires that a sign is used for all numbers. However, "+" and "-" sort in this sequence. So replace "-" by "0" and "+" by "1" to fix this.

The Generic Key Transform does not change numbers that have more than the preselected number of digits on either side of the decimal point. Neither does it give non-canonic numbers any special attention. After the (transformed) number we put an "A" to denote "Arabic" number. In a similar way, Roman numbers can be first transformed to the Arabic notation, then to the format as described with an "R" appended. The function deals with the fact that a Roman number may be represented by either lowercase or uppercase characters. But again, the fact that the transform has to be reversible, means we can accept "canonic numbers" only. The tokens MXM and MDCCCCLXXXX represent the same number (1990).

The functionality described so far is fully implemented in the MUMPS code that follows this article.

Further effort could extend the function to sort words for numbers according to their numeric value instead of alphabetically. There again the question of canonic values arises (eleven-hundred vs. one-thousand-one hundred).

Doing things a little differently, one could eliminate the need to restrict numeric entities to their canonic form.

The remaining challenges are how to find a way to collate arbitrary embedded numbers and what to do with ambiguities like "mix." For most practical purposes, the limitations in this Generic Key Transform will be insignificant. Again, MUMPS provides the perfect framework for easy implementation of very demanding tasks. **M**

```
Routine %GKT
09/18/95 10:25:05 AM
Size is 9877 bytes, 158 lines
```

```
+1 %GKT    ;generic key transform;
+2 +1      ;Gerum,4.7.1995
+3 +2      Q
+4 +3      ;function $$IN^%GKT(InStr,TrTable[,Offset[,Delim
           [,Just1,Just2,Nsys]]])
+5 +4      ;in:InStr string to be transformed
+6 +5      ;   TrTable transformation table (string)
+7 +6      ;   1st char usually ":" delimiter within replacement pair
+8 +7      ;   2nd char usually "," delimiter between replacement pairs
+9 +8      ;   following chars are pairs like
+10 +9     ;  " :_,a:A,A:A,b:B,ä:AE,sch:S"
+11 +10    ;   meaning "a" sorts as "A", "b" as "B", "ä" as "AE" etc.
+12 +11    ;   the "from" part of a replacement pair should be at least one character long.
+13 +12    ;   The "to" part may be any length, including zero.
+14 +13    ;   So the delimiting chars themselves cannot be transformed. The
+14 +15    ;   order within the table is significant; if
+16 +15    ;   several combinations of chars map into the same char the
+17 +16    ;   order within the table decides which one sorts first. Each
+18 +17    ;   character not in the table sorts before occurrences of the
+19 +18    ;   same character resulting from a transform.
+20 +19    ;   Offset for counts string, recommended value 0 if CTRLS are
+21 +20    ;   permitted as subscripts, otherwise 32 (=$ASCII(" "))
+22 +21    ;   default value is 48 (=$ASCII("0"))
+23 +22    ;   If called by name, value actually used is returned.
+24 +23    ;   Delim delimiter between raw transform and counts string,
+25 +24    ;   should be one character. Surplus chars are being discarded.
+26 +25    ;   default value is " "
+27 +26    ;   That character either should never be in an input
+28 %GKT+27 ; string or it should appear in the translation table.
+29 +28    ;   Its $ASCII value should be small.
+30 +29    ;   If called by name, the value actually used is returned.
+31 +30    ;   The following three parameters are used to get the
+32 +31    ;   desired collation for substrings that are numeric.
+33 +32    ;   As it is not possible to fit arbitrary precision numbers
+34 +33    ;   in this collation scheme, the restriction is as follows:
+35 +34    ;   If a numeric substring SS satisfies
+36 +35    ;   SS=+$EXTRACT($JUSTIFY(SS,Just1,Just2),1,$LENGTH(Just1))
+37 +36    ;   and if adjacent characters do not extend the substring
+38 +37    ;   to form a numlit,
+39 +38    ;   it sorts according to its numeric value
+40 +39    ;   Just1
+41 +40    ;   Just2
+42 +41    ;   Nsys String of Characters "ARr"
+43 +42    ;   to determine whether Arabic, Roman (written in uppercase) or
+44 +43    ;   Roman (written in lowercase) numbers should be sorted according
+45 +44    ;   to their numeric value and which system should take precedence.
+46 +45    ;   default is "A" (Arabic, no Roman numbers)
+47 +46    ;   Only for Arabic numbers a sign and a fractional part is accepted
+48 +47    ;out: transformed key, with desired collation properties
+49 IN(InStr,TrTable,Offset,Delim,Just1,Just2,Nsys)  N A,B,C,DbetwPrs,DinPair,F,I,Num,R,X,Y,Z
+50 +1      QUIT:$G(InStr)-"" "" ;missing or trivial string
+51 +2      QUIT:$G(TrTable)?.4E InStr ;missing or trivial translation table
+52 +3      S:$G(Offset)="" Offset=48 ;offset defaults to $ASCII("0")
+53 +4      SET Delim=$EXTRACT($JUSTIFY($G(Delim),1)) ;delimiter defaults to " "
+54 +5      SET Num=$CHAR($A(Delim)+1) ;"numeric" token (make it parameter?)
+55 +6      SET DinPair=$EXTRACT(TrTable) ;delimiter within replacement pair
+56 IN+7    SET DbetwPrs=$EXTRACT(TrTable,2) ;delimiter between replacement pairs
+57 +8      SET (X,Y,Z)= ""
+58 +9      FOR I=1:1:$LENGTH(InStr) DO  SET X=X_C,Y=Y_B,Z=Z_A
+59 +10     .SET C=$EXTRACT(InStr,I),A=$CHAR(Offset)
```

```
+60 +11    .IF -.0123456789[$EXTRACT(InStr,I),Nsys["A" ;consider Arabic numbers?
+61 +12    .IF  IF -.0123456789'[$EXTRACT(Instr,I-1)!(I=1) ;number previously rejected?
+62 +13    .IF  IF $EXTRACT(InStr,I,$LENGTH(InStr))'?.1"-"1.N1"E"1N.E ;beware Exponential notation
+63 +14    .IF  SET C=+$EXTRACT(InStr,I,$LENGTH(InStr))
+64 +15    .IF  IF $EXTRACT(InStr,I,I+$LENGTH(C)-1)=C,$EXTRACT(InStr,I+$LENGTH(C))'=0 ;substring canonic
           number?
+65 +16    .IF  IF C=+$TRANSLATE($EXTRACT($JUSTIFY(C,Just1,Just2),1,Just1)," ");does # fit?
+66 +17    .IF  SET I=I+$LENGTH(C)-1 DO  SET C=Num_C_$CHAR($FIND(Nsys,"A")+64),B="" QUIT
+67 +18    ..IF C<0 SET C=0_$TRANSLATE($JUSTIFY($TRANSLATE(C,"-"),Just1,Just2)," 0123456789",99876543210)
+68 +19    ..ELSE  SET C=1_$TRANSLATE($JUSTIFY($TRANSLATE(C,"-"),Just1,Just2)," ",0)
+69 +20    .IF "ivxlcdm"[$EXTRACT(InStr,I),Nsys["r" ;consider lowercase Roman numbers?
+70 +21    .IF  IF "ivxlcdm"'[$EXTRACT(InStr,I-1)!(I-1) ;number previously rejected?
+71 +22    .IF  SET C=$PIECE($EXTRACT(Instr,I,$LENGTH(InStr))_ ".", $EXTRACT($TRANSLATE($EXTRACT(Instr,I,
           $LENGTH(InStr)),"ivxlcdm")_"."))
+72 +23    .IF  IF EXTRACT(InStr,I-1)?.1P,$EXTRACT(InStr,I+$LENGTH(C))?.1P
+73 +24    .IF  IF C]"",C=$TRANSLATE($$NormRom(C),"IVXLCDM","ivxlcdm") SET R=$$R2A(C)
+74 +25    .IF  IF R=+$TRANSLATE($EXTRACT($JUSTIFY(R,Just1,Just2),1,Just1), " "); does # fit?
+75 +26    .IF  SET I=I+$LENGTH(C)-1,C=Num_1_$TRANSLATE($JUSTIFY(R,Just1,Just2)," ",0)_$CHAR($FIND(Nsys,"r")
           +64),B="" QUIT
+76 +27    .IF "IVXLCDM"[$EXTRACT(InStr,I),Nsys["R" ;consider uppercase Roman number?
+77 +28    .IF  IF "IVXLCDM"'[$EXTRACT(Instr,I-1)!(I=1) ;numbers previously rejected?
+78 +29    .IF  SET C=$PIECE($EXTRACT(InStr,I,$LENGTH(InStr))_ ".",$EXTRACT($TRANSLATE($EXTRACT(InStr,I,
           $LENGTH(InStr)),"IVXLCDM")_"."))
+79 IN+30 .IF  IF EXTRACT(InStr,I-1)?.1P,$EXTRACT(InStr,I+$LENGTH(C))?.1P
+80 +31    .IF  IF C]"",C=$$NormRom(C) SET R=$$R2A(C)
+81 +32    .IF  IF R=+$TRANSLATE($EXTRACT($JUSTIFY(R,Just1,Just2),1,Just1)," ") ;does # fit?
+82 +33    .IF  SET I=I+$LENGTH(C)-1,C=Num_1_$TRANSLATE($JUSTIFY(R,Just1,Just2)," ",0)_$CHAR($FIND(Nsys,"R")
           +64),B="" QUIT
+83 +34    .S C=$EXTRACT(InStr,I)
+84 +35    .IF TrTable'[(DbetwPrs_C) SET (A,B)=$CHAR(Offset) QUIT  ;no transform
+85 +36    .F I=I:1:$LENGTH(InStr) QUIT  :Trtable'((DbetwPrs_C_$EXTRACT(InStr,I+1)) QUIT  :$EXTRACT(InStr,I
           +1)=DinPair DO
+86 +37    ..S C=C_$EXTRACT(InStr,I+1) ;get longest matching entry
+87 +38    .IF TrTable'[(DbetwPrs_C_DinPair) SET B=$CHAR(Offset) QUIT
+88 +39    .S F=$PIECE($PIECE(TrTable,DbetwPrs_C_DinPair,2),DbetwPrs) ;get replacement
+89 +40    .IF F="" SET B=$LENGTH($PIECE(TrTable_DbetwPrs,DbetwPrs_C_DinPair),DinPair_DbetwPrs) ;qet # of
           entry
+90 +41    .IF SET A=$CHAR(Offset+B),(B,C)="" QUIT
+91 +42    .SET B=$LENGTH($PIECE(TrTable_DbetwPrs,DbetwPrs_C_DinPair),DinPair_$EXTRACT(F)) ;get # of entry
+92 +43    .SET C=F,B=$CHAR(Offset+B),A=$CHAR(Offset)
+93 +44    FOR I=$LENGTH(Z):-1:0 IF $A(Z,I)'=offset SET Z=$EXTRACT(Z,1,I) QUIT
+94 +45    FOR I=$LENGTH(Y):-1:0 IF $A(Y,I)'=Offset SET Y=$EXTRACT(Y,1,I) QUIT
+95 +46    IF Z="" QUIT:Y="" X QUIT X_Delim_Y
+96 +47    QUIT X_Delim_Y_Delim_Z
+97 +48    ;function $$IN^%GKT(InStr,TrTable[,Offset[,Delim[,Just1,Just2,Nsys]]])
+98 +49    ;Just2 is currently not used. We keep it just for symmetry with  $$IN
+99 +50    ;reverse function to IN, in and out same as above
+100 OUT(InStr,TrTable,Offset,Delim,Just1,Just2,Nsys) NEW A,B,C,DbetwPrs,DinPair,I,IA,IB,Num,OutStr,X,Y,Z,c
+101 +1    QUIT:$G(InStr)= "" "" ;missing or trivial string
+102 +2    QUIT:$G(Trtable)?.4E InStr ;missing or trivial translation table
+103 OUT+3 SET:$G(Offset)="" Offset=48 ;offset defaults to $ASCII("0")
+104 +4    SET Delim=$EXTRACT($JUSTIFY($G(Delim),1)) ;delimiter defaults to " "
+105 +5    SET Num=$CHAR($A(Delim)+1) ;"numeric" token (make it parameter?)
+106 +6    SET DinPair=$EXTRACT(TrTable) ;delimiter within replacement pair
+107 +7    SET DbetwPrs=$EXTRACT(TrTable,2) ;delimiter between replacement pairs
+108 +8    SET X=$PIECE(InStr,Delim),Y=$PIECE(InStr,Delim,2),Z=$PIECE(InStr,Delim,3)
+109 +9    IF Z="",Y="" QUIT X
+110 +10   SET OutStr="",IB=1,IA=1
+111 +11   FOR I=1:1:$LENGTH(X) SET C=$EXTRACT(X,I) DO  SET OutStr-OutStr_C
+112 +12   .SET A=$EXTRACT(Z,IA),IA=IA+1
+113 +13   .IF $A(A)>Offset SET C=$PIECE(TrTable_DbetwPrs,DinPair_DbetwPrs,$A(A)-Offset+1),C=$PIECE(C,DbetwPrs,
           $LENGTH(C,DbetwPrs)),I=I-1 QUIT
+114 +14   .IF C=Num SET C=$EXTRACT(X,I+2,I+Just1+1),B=$EXTRACT(Nsys,$A(X,I+Just1+2)-65) DO QUIT
+115 +15   ..SET C=$SELECT($EXTRACT(X,I+1):+C,1:-$TRANSLATE(C,"0123456789",9876543210))
+116 +16   ..IF B="A"
+117 +17   ..ELSE IF B="R" SET C=$$A2R(C)
+118 +18   ..ELSE IF B="r" SET C=$TRANSLATE($$A2R(C),"IVXLCDM","ivxlcdm")
+119 +19   ..IF SET I=I+Justl+2,B=""
+120 +20   ..ELSE SET Outstr="<inconsistent input>",I=$LENGTH(X)
+121 +21   .SET B=$EXTRACT(Y,IB),IB=IB+1 QUIT:B=""
+122 +22   .SET B=$A(B)-Offset QUIT:'B ;no replacement
+123 +23   .SET C=$LENGTH($PIECE(TrTable,DinPair_C,1,B),DbetwPrS) ;Piece # of replacement pair
+124 +24   .SET C=$PIECE(TrTable,DbetwPrs,C) ;replacement pair
+125 +25   .SET c=$PIECE(C,DinPair,2) ;replacing string
+126 +26   .SET C=$PIECE(C,DinPair) ;original string
+127 +27   .IF C=""!(c="") SET OutStr="<inconsistent input>",I=$LENGTH(X) QUIT
+128 +28   .SET I=I+$LENGTH(c)-1 ;skip (length of replacement)
+129 +29   FOR IA=IA:1 SET A=$EXTRACT(Z,IA) QUIT:$A(A)'>Offset DO SET OutStr=OutStr_C
+130 +30   .SET C=$PIECE(TrTable_DbetwPrs,DinPair_DbetwPrs,$A(A)-Offset+1),C=$PIECE(C,DbetwPrs,
           $LENGTH(C,DbetwPrs))
+131 OUT+31 QUIT OutStr
+132 +32   ;sort all lowercase as uppercase
+133 +33   ;inverse of $$UIN
+134 UIN(X) QUIT $$IN(X,":,:,!:,-:,A:a,B:b,C:c,D:d,E:e,F:f,G:g,H:h,I:i,J:j,K:k,L:l,M:m,N:n,O:o,P:p,Q:q,R:r,
           S:s,T:t,U:u,V:v,W:w,X:x,Y:y,Z:z,ä:ae,ö:oe,ü:ue,ß:ss,Ä:ae,Ö:oe,Ü:ue",48," ",10,2,"ArR")
+135 UOUT(X) QUIT $$OUT(X,":,:,!:,-:,A:a,B:b,C:c,D:d,E:e,F:f,G:g,H:h,I:i,J:j,K:k,L:l,M:m,N:n,O:o,P:p,Q:q,R:r,
           S:s,T:t,U:u,V:v,W:w,X:x,Y:y,Z:z,ä:ae,ö:oe,ü:ue,ß:ss,Ä:ae,Ö:oe,Ü:ue",48," ",10,2,"ArR")
+136 DUDEN(X) QUIT $$OUT(X,":,:,!:,-:,A:a,B:b,C:c,D:d,E:e,F:f,G:g,H:h,I:i,J:j,K:k,L:l,M:m,N:n,O:o,P:p,Q:q,R:r,
           S:s,T:t,U:u,V:v,W:w,X:x,Y:y,Z:z,ä:a,ö:o,ü:u,ß:ss,Ä:a,Ö:o,Ü:u",48," ",0,0,"A")
+137 PHONE(X) QUIT $$IN(X,":,:,!:,-,A:a,B:b,C:c,D:d,E:e,F:f,G:g,H:h,I:i,J:j,K:k,L:l,M:m,N:n,O:o,P:p,Q:q,R:r,
           S:s,T:t,U:u,V:v,W:w,X:x,Y:y,Z:z,ä:ae,ö:oe,ü:ue,ß:ss,Ä:ae,Ö:oe,Ü:ue",48," ",10,2,"A")
+138 +1     ;-$$A2R(X) : convert Arabic to Roman
+139 A2R(X) NEW I,J,R,T
+140 +1     SET R=$TRANSLATE(X," ")
+141 +2     QUIT:R'?1.N ""
+142 +3     QUIT:'R "nihil"
+143 +4     SET T="",T(1)="I",T(5)="V",T(10)="X",T(50)="L",T(100)="C",T(500)="D",T(1000)="M"
+144 +5     FOR I=1000,500,100,50,10,5,1 DO
+145 +6     .FOR QUIT:'R DO:R<I QUIT:R<I SET R=R-I,T=T_T(I)
+146 +7     ..FOR J=1,10,100 QUIT:J*2'<I IF I-J'>R SET R=R-I+J,T=T_T(J)_T(I) QUIT
+147 +8     ..; one could subtract fives: MVM <—> MXMV etc.
+148 +9     ..; FOR J=1,5,10,50,100 QUIT:J*2'<I IF I-J'>R SET R=R-I+J,T=T_T(J)_T(I) QUIT
+149 +10    QUIT T
+150 +11    ;-$$R2A(X) : convert Roman to Arabic
+151 R2A(X)     NEW I,R,T,Y
+152 +1     SET R=$TR(X,"ivxlcdm","IVXLCDM") QUIT:R="NIHIL" 0
+153 +2     QUIT:$TR(R,"IVXLCDM")]""!(R="") "" ;-invalid format
+154 +3     SET T="",T("I")=1,T("V")=5,T("X")=10,T("L")=50,T("C")=100,T("D")=500,T("M")=1000
+155 R2A+4    FOR I=1:1:$L(R)-1 S Y=T($E(R,I)),T=T+$S(T($E(R,I+1))>Y:-Y,1:Y)
+156 +5     QUIT T+T($E(R,$L(R)))
+157 +6     ;-$$NormRom(x) : normalize Roman
+158 NormRom(X) QUIT $$A2R($$R2A(X))
```

Winfried Gerum is with Winner Software, GmbH in Röttenbach, Germany. You may contact him by phone at 011-49-9195-940022 or by fax at 011-49-9195-940030.

MTA +

Database & Client/Server World =

added value in '96