Title

stata.com

ustrcompare( ) — Compare or sort Unicode strings

Description

`ustrcompare(`*s1*`,` *s2* `[ ,` *loc* `])` compares two Unicode strings. The function returns −1, 1, or 0 if *s1* is less than, greater than, or equal to *s2*, respectively. If *loc* is not specified, the `locale_functions` setting is used.

`ustrsortkey(`*s*`,` *loc* `[ ,` *loc* `])` generates a null-terminated byte array. The `sort` command on the sort keys of two Unicode strings *s1* and *s2* produces the same order from `ustrcompare(`*s1*`,` *s2*`,` *loc*`)`. If *loc* is not specified, the `locale_functions` setting is used. The result is also diacritic and case sensitive. If you need different behavior, you should use the extended function `ustrsortkeyex()`.

`ustrcompareex(`*s1*`,` *s2*`,` *loc*`,` *st*`,` *case*`,` *cslv*`,` *norm*`,` *num*`,` *alt*`,` *fr*`)` is an extended version of `ustrcompare()`. It provides more options for the comparison behaviors.

`ustrsortkeyex(`*s*`,` *loc*`,` *st*`,` *case*`,` *cslv*`,` *norm*`,` *num*`,` *alt*`,` *fr*`)` is an extended version of `ustrsortkey()`. It provides more options for the comparison behaviors.

The additional options are as follows:

st controls the strength of the comparison:

`-1`	default value for the locale
`1`	primary; base-letter differences, such as “a” and “b”
`2`	secondary; diacritical differences of the same base letter, such as “a” and “ä”
`3`	tertiary; case differences of the same base letter, such as “a” and “A”
`4`	quaternary; used to distinguish between Katakana and Hiragana for JIS 4061 collation standard
`5`	identical; code-point order of the string; rarely useful

Numbers other than those listed above are treated as tertiary.

case controls the uppercase and lowercase letter order. Possible values are `1` (uppercase first), `2` (lowercase first), or `0` (use tertiary strength; advanced option). `-1` means the default value for the locale should be used. Any other values are treated as `0`.

cslv controls whether an extra case level between the secondary level and the tertiary level is generated. Possible values are `0` (off) or `1` (on). `-1` means the default value for the locale should be used. Any other values are treated as `0`. Combining this setting to be “on” and the strength setting to be primary can achieve the effect of ignoring the diacritical differences but preserving the case differences. If the setting is “on”, the result is also affected by the *case* setting.

norm controls whether the normalization check and normalizations are performed. Possible values are `0` (off) or `1` (on). `-1` means the default value for the locale should be used. Any other values are treated as `0`. Most languages do not require normalization for comparison. Normalization is needed in languages that use multiple combining characters, such as Arabic, ancient Greek, or Hebrew. For more information about normalization, see [M-5] **ustrnormalize( )**,

num controls how contiguous digit substrings are sorted. Possible values are `0` (off) or `1` (on). `-1` means the default value for the locale should be used. Any other values are treated as `0`. If the setting is "on", substrings consisting of digits are sorted based on the numeric value. For example, "100" is after "20" instead of before it. Note that digit substrings are limited to 254 digits and that plus or minus signs, decimals, and exponents are not supported.

alt controls how spaces and punctuation characters are handled. Possible values are `0` (use primary weights) or `1` (alternative handling). Any other values are treated as `0`. If the setting is `1` (alternative handling), "onsite", "on-site", and "on site" are considered the same.

fr controls the direction of secondary strength. Possible values are `0` (off) or `1` (on). `-1` means the default value for the locale should be used. All other values are treated as "off". If the setting is "on", the diacritical letters are sorted backward. Note that the setting is "on" by default only for Canadian French locale (`fr_CA`).

When *s1* and *s2* are not scalar, these functions return element-by-element results.

Syntax

real matrix `ustrcompare(`*string matrix s1*`,` *string matrix s2* [`,` *string scalar loc*]`)`

string matrix `ustrsortkey(`*string matrix s* [`,` *string scalar loc*]`)`

real matrix `ustrcompareex(`*string matrix s1*`,` *s2*`,` *string scalar loc*`,`
real scalar st`,` *case*`,` *cslv*`,` *norm*`,` *num*`,` *alt*`,` *fr*`)`

string matrix `ustrsortkeyex(`*string matrix s*`,` *string scalar loc*`,`
real scalar st`,` *case*`,` *cslv*`,` *norm*`,` *num*`,` *alt*`,` *fr*`)`

Remarks and examples

stata.com

Unicode string comparison is locale dependent. For example, z $<$ ö in Swedish but ö $<$ z in German. The comparison is diacritic and case sensitive. If you need different behavior, such as case-insensitive comparison, you should use the extended comparison function `ustrcompareex()`. Unicode string comparison is language sensitive, which is different from the byte value comparison used by `sort`. For example, capital letter "Z" (byte value 90) comes before lowercase "a" (byte value 97) in terms of byte value but comes after "a" in any English dictionary.

An invalid UTF-8 sequence is replaced with the Unicode replacement character `\ufffd`.

Conformability

`ustrcompare(`*s1*`,` *s2* `[ ,` *loc* `])`:

s1:	$r \times c$
s2:	$r \times c$
loc:	1×1
result:	$r \times c$

`ustrsortkey(`*s* `[ ,` *loc* `])`:

s:	$r \times c$
loc:	1×1
result:	$r \times c$

`ustrcompareex(`*s1*`,` *s2*`,` *loc*`,` *st*`,` *case*`,` *cslv*`,` *norm*`,` *num*`,` *alt*`,` *fr*`)`:

s1:	$r \times c$
s2:	$r \times c$
loc:	1×1
st:	1×1
case:	1×1
cslv:	1×1
norm:	1×1
num:	1×1
alt:	1×1
fr:	1×1
result:	$r \times c$

`ustrsortkeyex(`*s*`,` *loc*`,` *st*`,` *case*`,` *cslv*`,` *norm*`,` *num*`,` *alt*`,` *fr*`)`:

s:	$r \times c$
loc:	1×1
st:	1×1
case:	1×1
cslv:	1×1
norm:	1×1
num:	1×1
alt:	1×1
fr:	1×1
result:	$r \times c$

Diagnostics

`ustrcompare()` and `ustrcompareex()` return a negative number other than -1 if an error occurs.

`ustrsortkey()` and `ustrsortkeyex()` return an empty string if an error occurs.

Also see

[M-5] **sort( )** — Reorder rows of matrix

[M-4] **String** — String manipulation functions

[U] **12.4.2 Handling Unicode strings**