Using □String

Contents

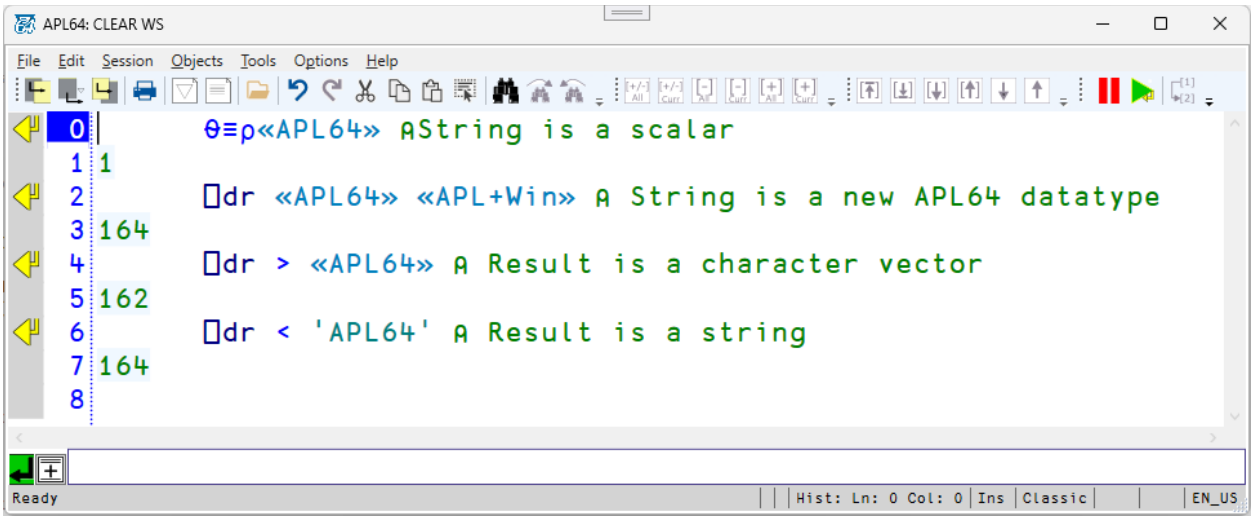
String Datatype in APL64 2
□String System Function 2
□String Actions 3
COMPARE 3
COMPAREORDINAL 4
CONCAT 5
CONTAINS 6
ENDSWITH 7
EQUALS 7
GETCULTUREINFO 8
GETHASHCODE 9
INDEXOF 9
INDEXOFANY 10
INSERT 11
ISNORMALIZED 12
ISNULLOREMPTY 12
ISNULLORWHITESPACE 13
JOIN 13
LASTINDEXOF 15
LASTINDEXOFANY 15
LENGTH 16
NORMALIZE 17
PADLEFT 17
PADRIGHT 18
REMOVE 18
REPLACE 19
SORTINDEX 19
SPLIT 22
STARTSWITH 23
SUBSTRING 24

TOCHARARRAY 25
TOLOWER 25
TOLOWERINVARIANT 26
TOUPPER 27
TOUPPERINVARIANT 28
TRIM 28
TRIMEND 29
TRIMSTART 29
? (Documentation Summary) 30

String Datatype in APL64

APL64 and .Net support the string and character array datatypes, but the string datatype is most frequently used for text data in in .Net.

A string is a scalar, not a character vector. The APL64 enstring (<) and destring functions (>) may be used to convert between string and character arrays.



⎕String System Function

The APL64 ⎕string system function is an interface to the .Net String class. Learn more about the .Net string datatype here.

Purpose:
⎕string exposes many of the methods and properties of the .Net String class.

Syntax: result ← larg ⎕string action rarg

Arguments:
larg are the optional arguments to the action.

action is a text vector that determines how the system function will operate on the left and right argument.

rarg are the optional arguments to the action.

Result:
The result, if any, depends on the action called. For most □string actions the result has the APL64 string datatype.

Remarks:

- A □String action which accepts a scalar String left argument will also accept a rank zero or one, character value left argument.
- A □String action which accepts a String[] vector left argument will also accept a vector left argument with elements which can be a string scalar or rank zero or one character values.

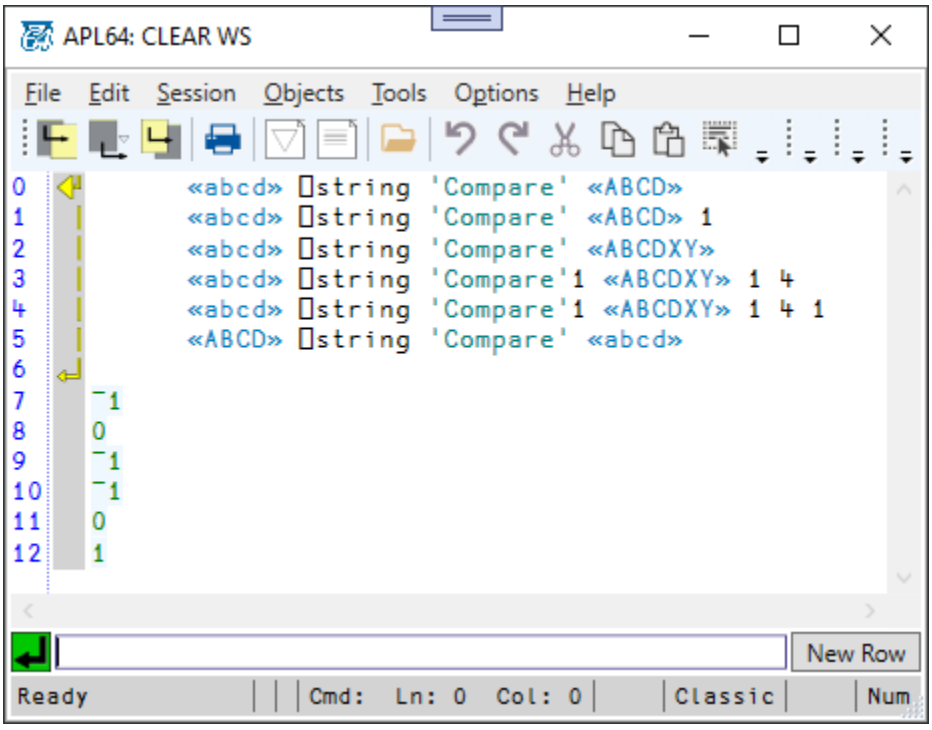
□String Actions

COMPARE

Int32←String □STRING "COMPARE" String
Int32←String □STRING "COMPARE" String IgnoreCase
Int32←String □STRING "COMPARE" StartPosInLarg String StartPosInRarg #CharsToCompare
Int32←String □STRING "COMPARE" StartPosInLarg String StartPosInRarg #CharsToCompare IgnoreCase

StartPosInLarg is □IO-sensitive.

```
«abcd» □string 'Compare' «ABCD»
«abcd» □string 'Compare' «ABCD» 1
«abcd» □string 'Compare' «ABCDXY»
«abcd» □string 'Compare'1 «ABCDXY» 1 4
«abcd» □string 'Compare'1 «ABCDXY» 1 4 1
«ABCD» □string 'Compare' «abcd»
```

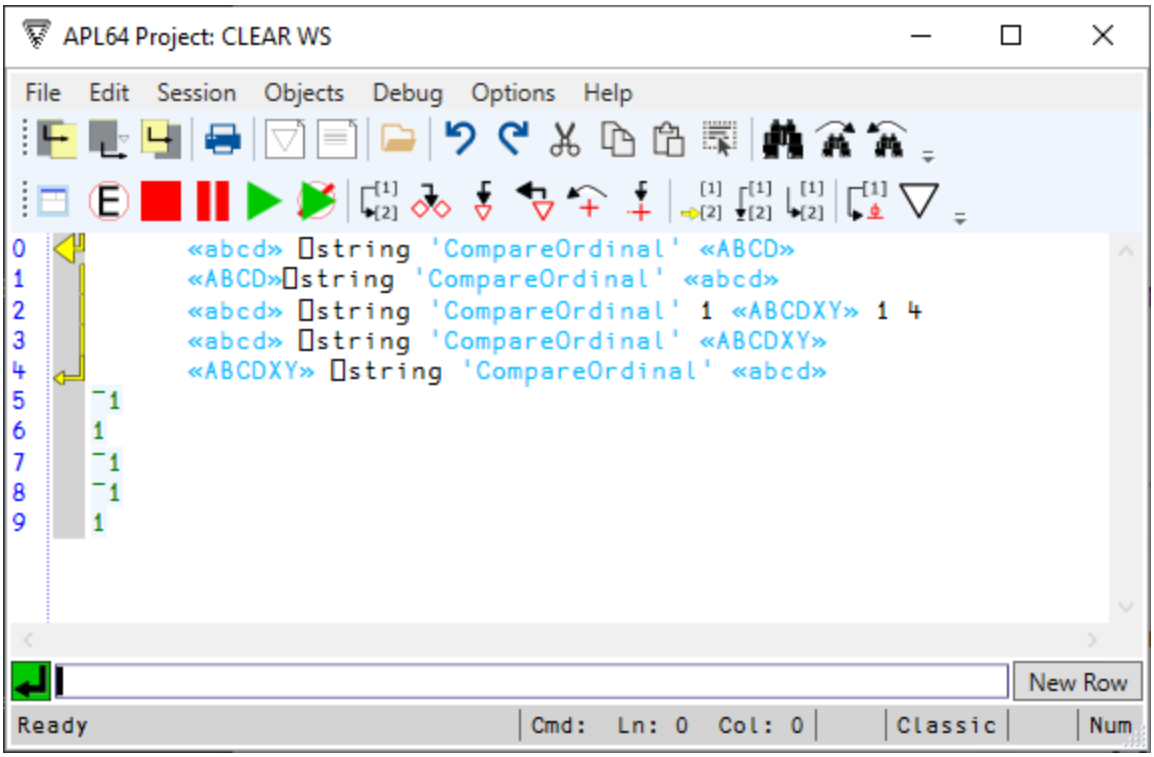


COMPAREORDINAL

Int32←String ⎕STRING "COMPAREORDINAL" String
Int32←String ⎕STRING "COMPAREORDINAL" StartPosInLarg String StartPosInRarg #CharsToCompare

StartPosInLarg is ⎕IO-sensitive.

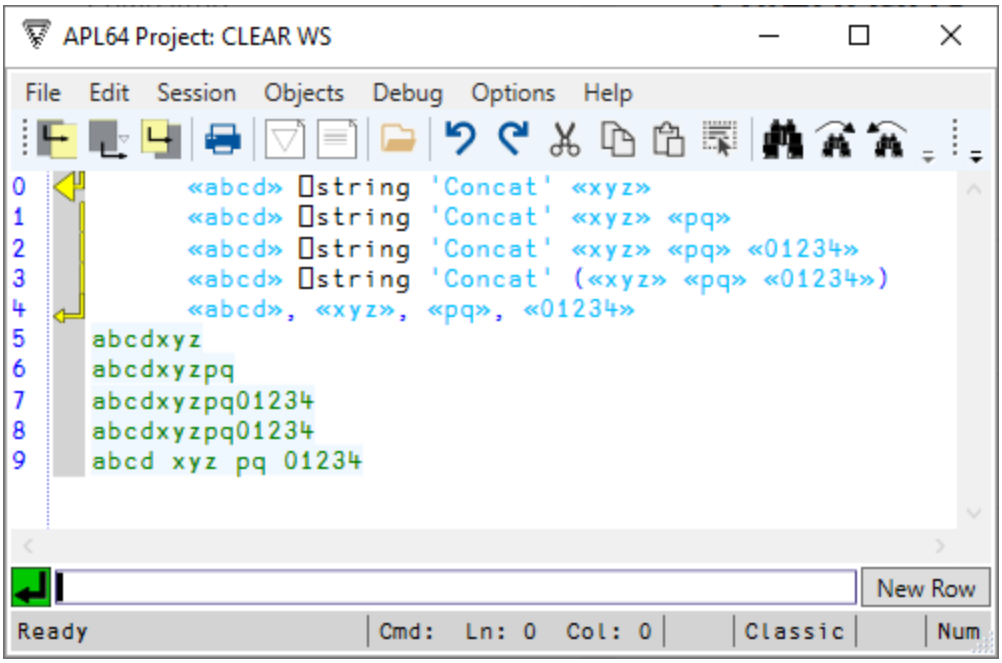
```
«abcd» ⎕string 'CompareOrdinal' «ABCD»
«ABCD» ⎕string 'CompareOrdinal' «abcd»
«abcd» ⎕string 'CompareOrdinal'1 «ABCDXY» 1 4
«abcd» ⎕string 'CompareOrdinal' «ABCDXY»
«ABCDXY» ⎕string 'CompareOrdinal' «abcd»
```



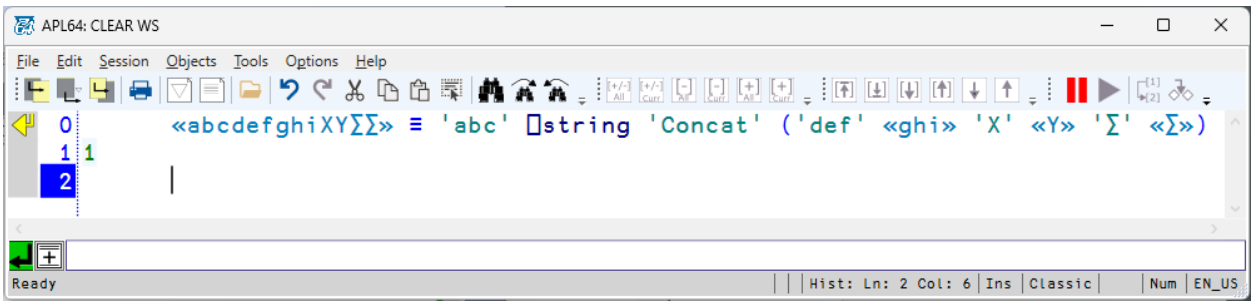
CONCAT

String←String ⍀STRING "CONCAT" String
String←String ⍀STRING "CONCAT" String String
String←String ⍀STRING "CONCAT" String String String
String←String ⍀STRING "CONCAT" String[]

```
«abcd»⍀string 'Concat' «xyz»
«abcd» ⍀string 'Concat' «xyz» «pqr»
«abcd» ⍀string 'Concat' «xyz» «pqr» «01234»
«abcd» ⍀string 'Concat' («xyz» «pqr» «01234»)
(«abcd», «xyz», «pqr», «01234»
```



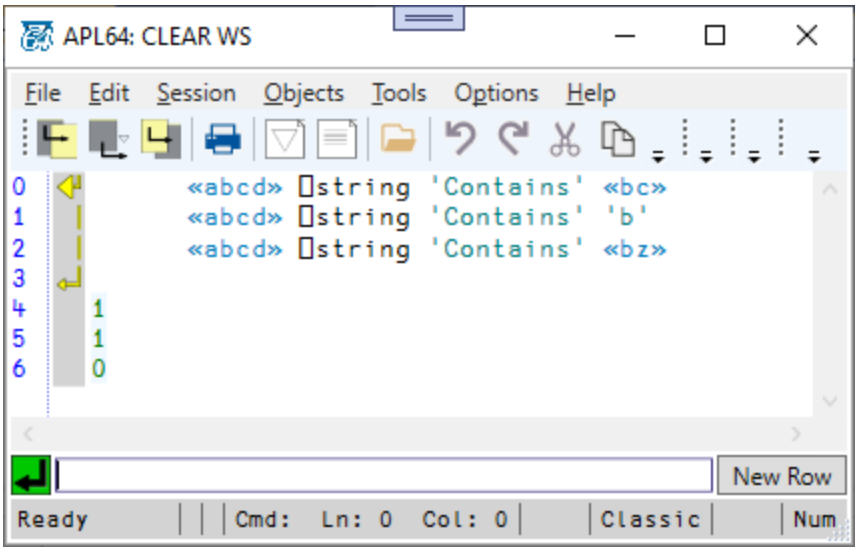

```
«abcdefghiXY∑∑» ≡ 'abc' ⎕string 'Concat' ('def' «ghi» 'X' «Y» '∑' «∑»)
```



CONTAINS

Bool←String ⎕STRING "CONTAINS" String
Bool←String ⎕STRING "CONTAINS" Char

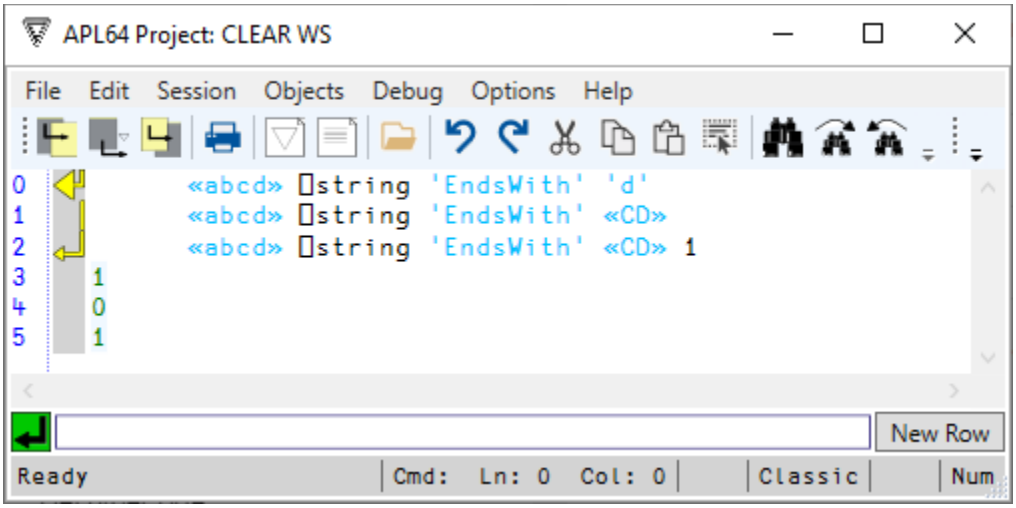
```
«abcd» ⎕string 'Contains' «bc»
«abcd» ⎕string 'Contains' 'b'
«abcd» ⎕string 'Contains' «bz»
```



ENDSWITH

Bool← String ⍷STRING "ENDSWITH" Char
Bool← String ⍷STRING "ENDSWITH" String
Bool← String ⍷STRING "ENDSWITH" String IgnoreCase

```
«abcd» ⍷string 'EndsWith' 'd'
«abcd» ⍷string 'EndsWith' "«CD»
«abcd» ⍷string 'EndsWith' «CD» 1
```

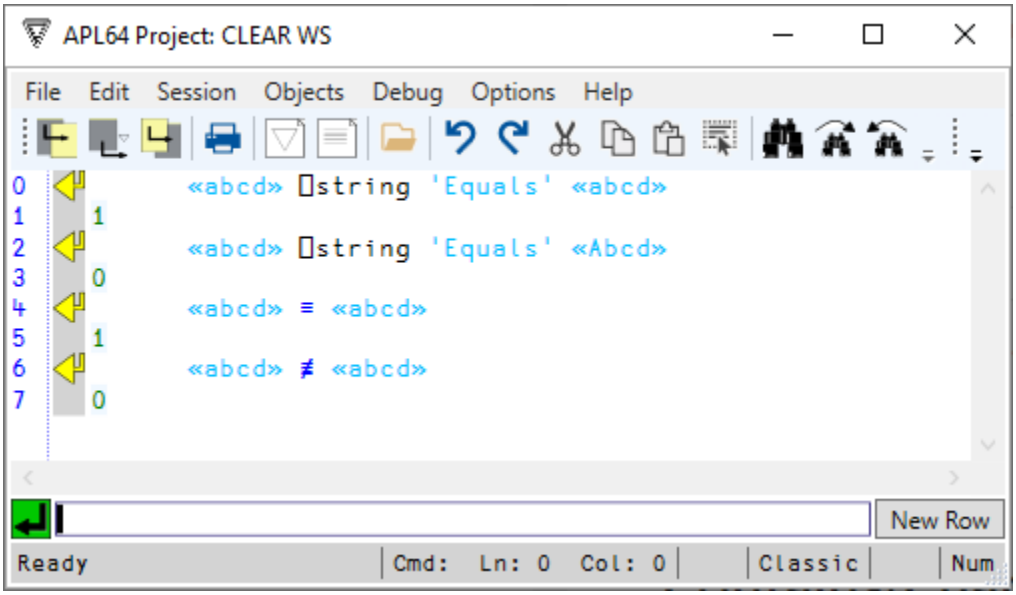


EQUALS

Bool← String ⍷STRING "EQUALS" String

```
«abcd» ⍷string 'Equals' «abcd»
«abcd» ⍷string 'Equals' «Abcd»
«abcd» ≡ «abcd»"
```

```
«abcd» ≢ «abcd»"
```



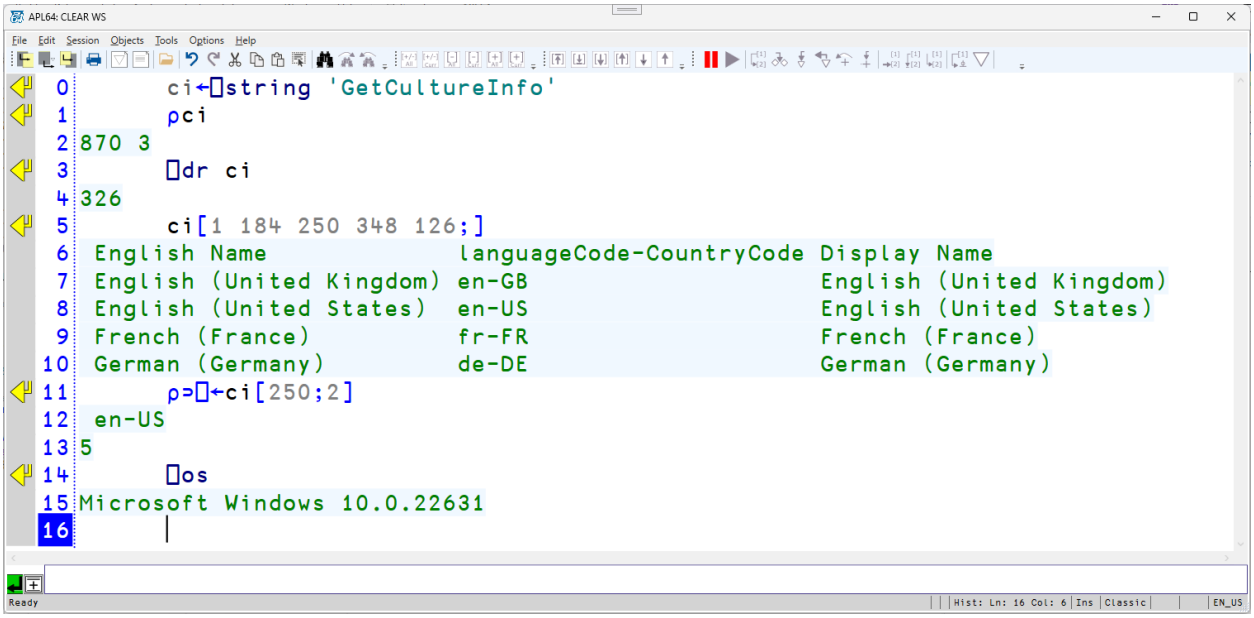
GETCULTUREINFO

AplArray←⎕STRING "GetCultureInfo"

Obtain the supported .Net cultures at application runtime, which vary with the operating system environment.

Use the 'LanguageCode-CountryCode' value when specifying culture information for actions such as ToLower and ToUpper.

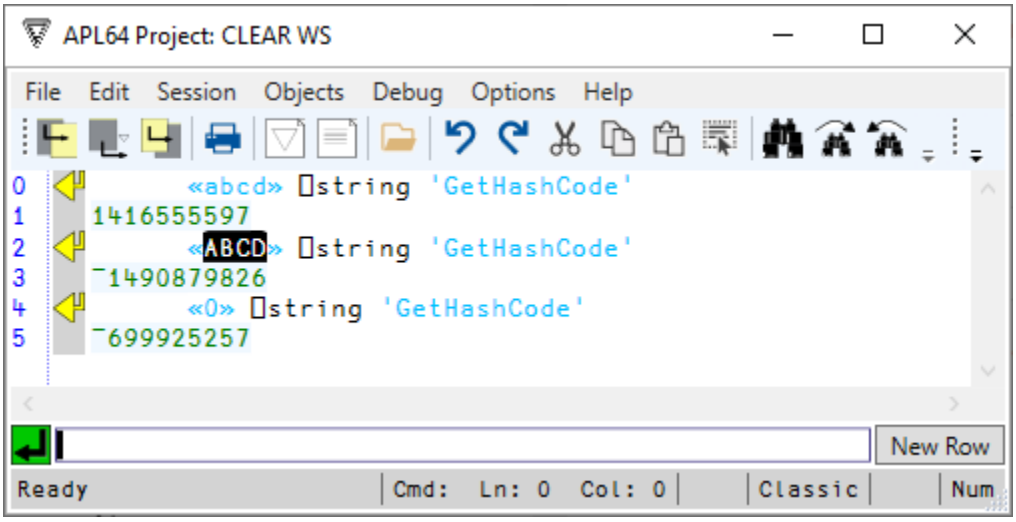
```
ci←⎕string 'GetCultureInfo'
⍴ci
⎕dr ci
ci[1 184 250 348 126;]
⍴⊃⎕←ci[250;2]
⎕os
```



GETHASHCODE

Int32←String ⎕STRING "GETHASHCODE"

The result values will not transcend an APL64 instance.

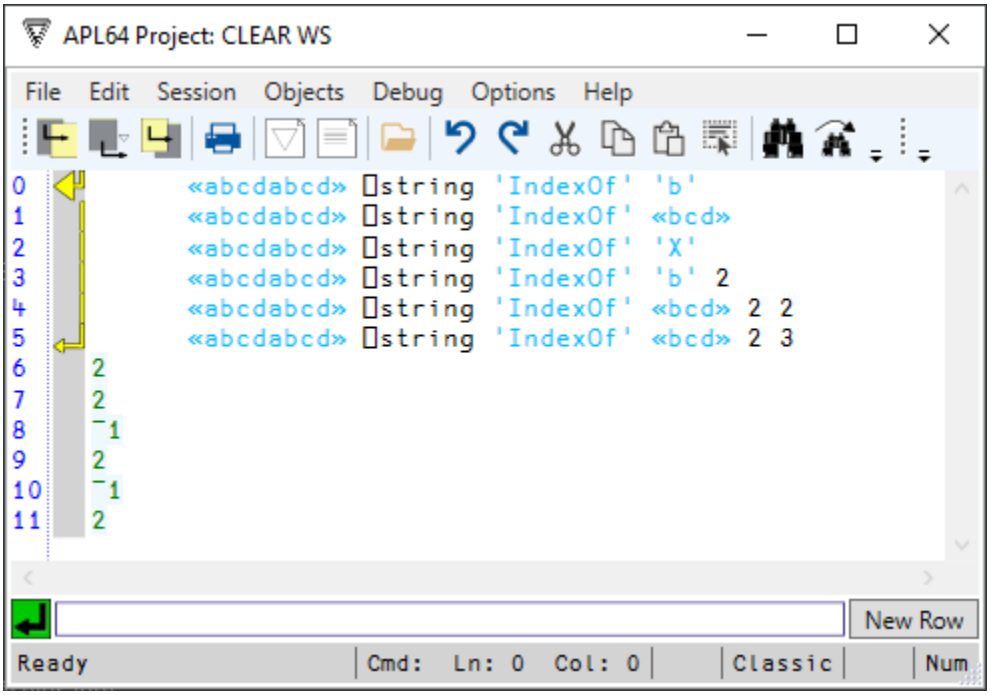


INDEXOF

Int32←String ⎕STRING "INDEXOF" Char
Int32←String ⎕STRING "INDEXOF" String
Int32←String ⎕STRING "INDEXOF" Char StartPosInLarg
Int32←String ⎕STRING "INDEXOF" String StartPosInLarg
Int32← String⎕STRING "INDEXOF" Char StartPosInLarg #CharsToExamine
Int32← String⎕STRING "INDEXOF" String StartPosInLarg #CharsToExamine

StartPosInLarg is ⎕IO-sensitive.

```
«abcdabcd» ⎕string 'IndexOf' 'b'
«abcdabcd» ⎕string 'IndexOf' «bcd»
«abcdabcd» ⎕string 'IndexOf' 'X'
«abcdabcd» ⎕string 'IndexOf' 'b' 2
«abcdabcd» ⎕string 'IndexOf' «bcd» 2 2
«abcdabcd» ⎕string 'IndexOf' «bcd» 2 3
```

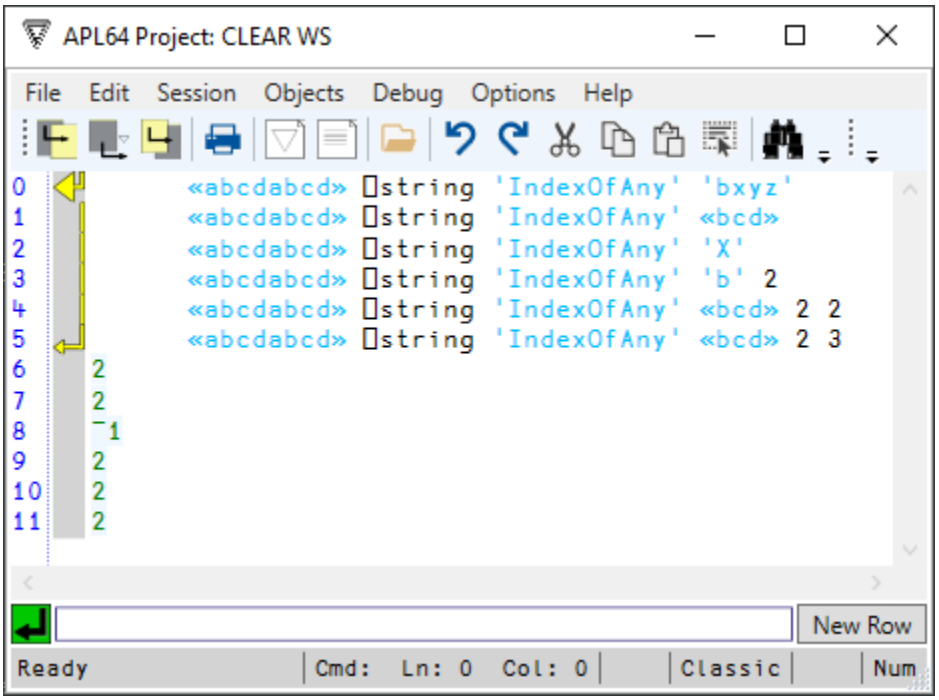


INDEXOFANY

Int32←String ⎕STRING "INDEXOFANY" Char[]
Int32←String ⎕STRING "INDEXOFANY" Char[] StartPosInLarg
Int32←String ⎕STRING "INDEXOFANY" Char[] StartPosInLarg #CharsToExamine
Int32←String ⎕STRING "INDEXOFANY" String
Int32←String ⎕STRING "INDEXOFANY" String StartPosInLarg
Int32←String ⎕STRING "INDEXOFANY" String StartPosInLarg #CharsToExamine

StartPosInLarg is ⎕IO-sensitive.

```
«abcdabcd» ⎕string 'IndexOfAny' 'bxy'
«abcdabcd» ⎕string 'IndexOfAny' «bcd»
«abcdabcd» ⎕string 'IndexOfAny' 'X'
«abcdabcd» ⎕string 'IndexOfAny' 'b' 2
«abcdabcd» ⎕string 'IndexOfAny' «bcd» 2 2
«abcdabcd» ⎕string 'IndexOfAny' «bcd» 2 3
```

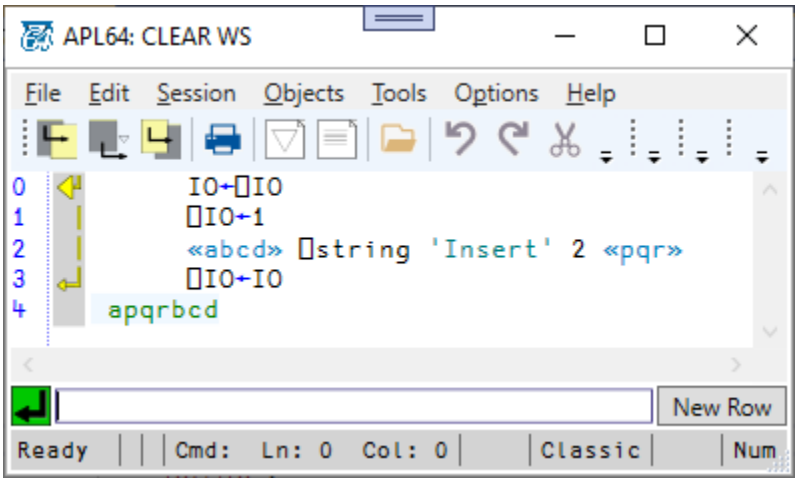


INSERT

String←String ⍞STRING "INSERT" StartPosInLarg String

StartPosInLarg is ⍞IO-sensitive.

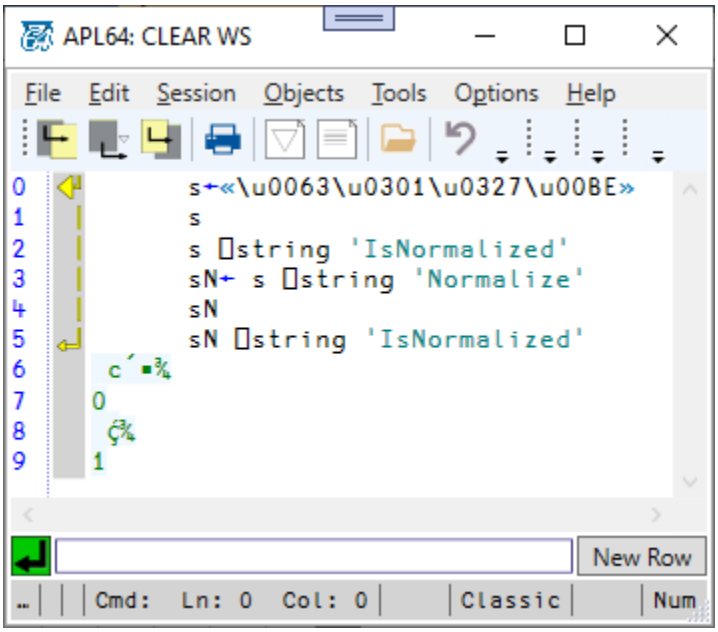
```
⎕IO←⍞IO
⍞IO←1
«abcd» ⍞string 'Insert' 2 «pqr»
⍞IO←⎕IO
```



ISNORMALIZED

String←String ⎕STRING "ISNORMALIZED"

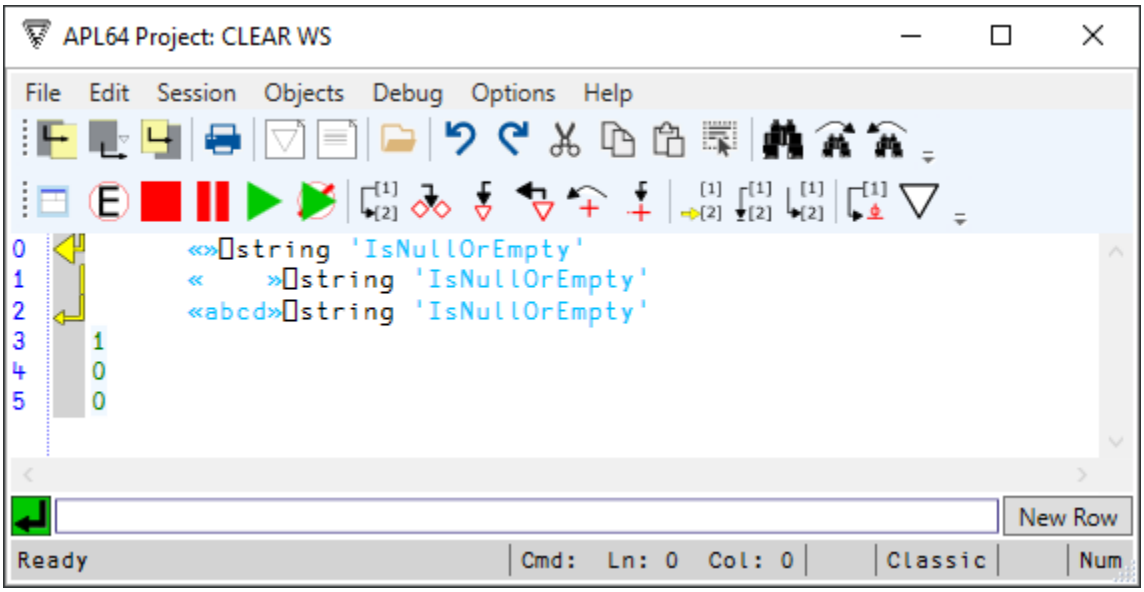
```
s←«\u0063\u0301\u0327\u00BE»
s
s ⎕string 'IsNormalized'
sN← s ⎕string 'Normalize'
sN
sN ⎕string 'IsNormalized'
```



ISNULLOREMPTY

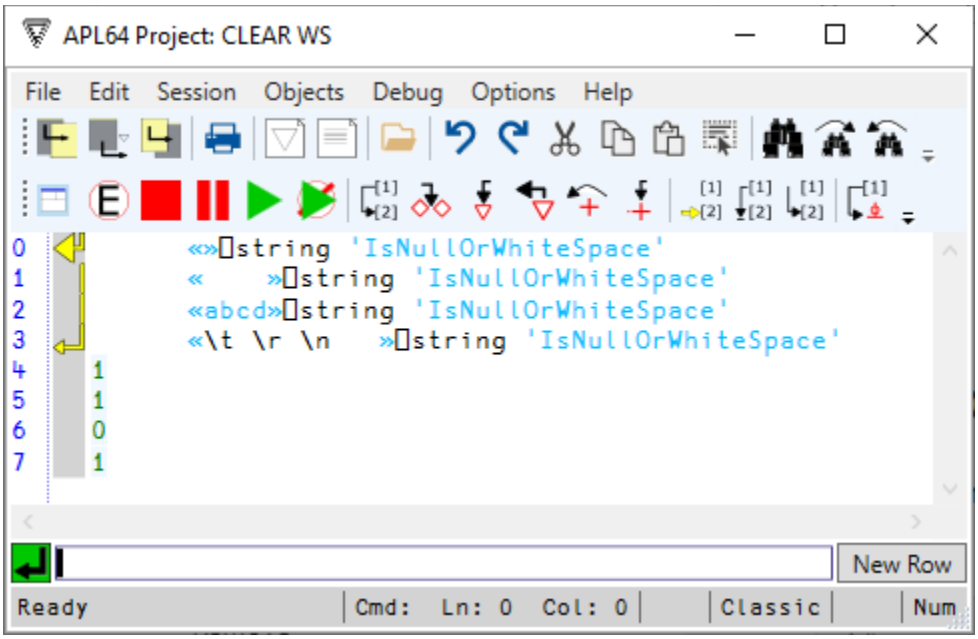
String←String ⎕STRING "ISNULLOREMPTY"

```
«» ⎕string 'IsNullOrEmpty'
«      » ⎕string 'IsNullOrEmpty'
«abcd» ⎕string 'IsNullOrEmpty'
```



ISNULLORWHITESPACE

String←String ⍁STRING "ISNULLORWHITESPACE"



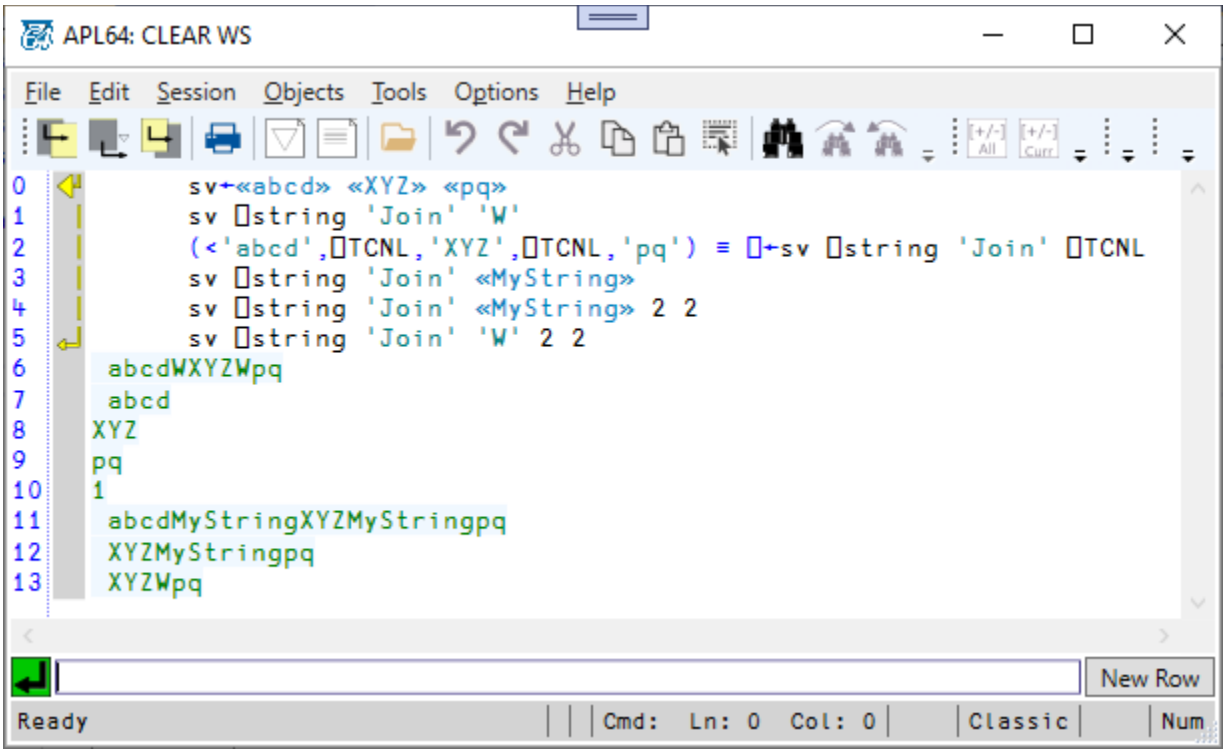
JOIN

String←String[] ⍁STRING "JOIN" (Char)delimChar
String←String[] ⍁STRING "JOIN" (String)delimString
String←String[] ⍁STRING "JOIN" (Char)delimChar StartItem# #Items
String←String[] ⍁STRING "JOIN" (String)delimChar StartItem# #Items

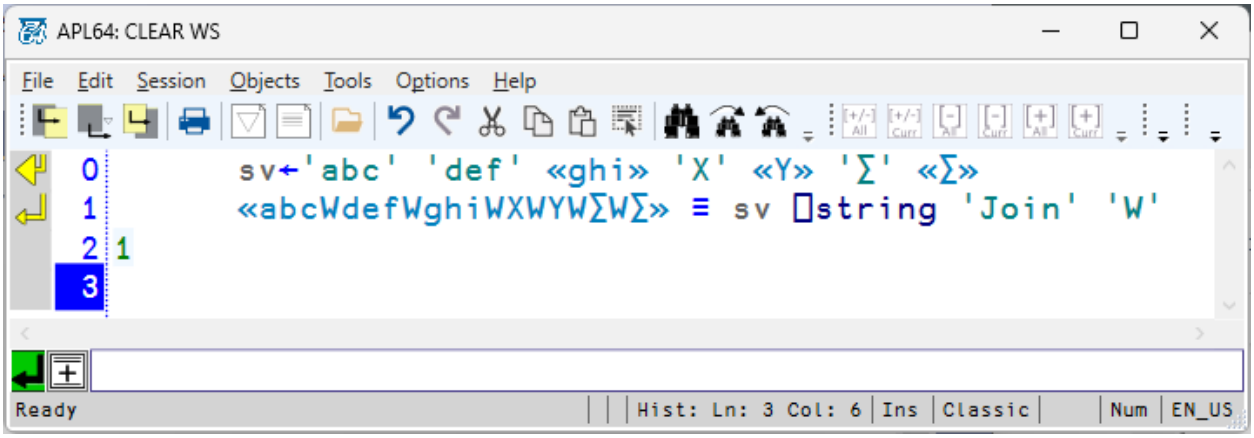
StartItem# is ⎕IO-sensitive.

```
sv←«abcd» «XYZ» «pq»
sv ⎕string 'Join' 'W'
(<'abcd',⎕TCNL,'XYZ',⎕TCNL,'pq') ≡ ⎕←sv ⎕string 'Join' ⎕TCNL
sv ⎕string 'Join' «MyString»
sv ⎕string 'Join' «MyString» 2 2
sv ⎕string 'Join' 'W' 2 2
```

Note: The monadic < and > operators in the example are called Enstring and Destring, respectively, and are used to convert between string and character objects.



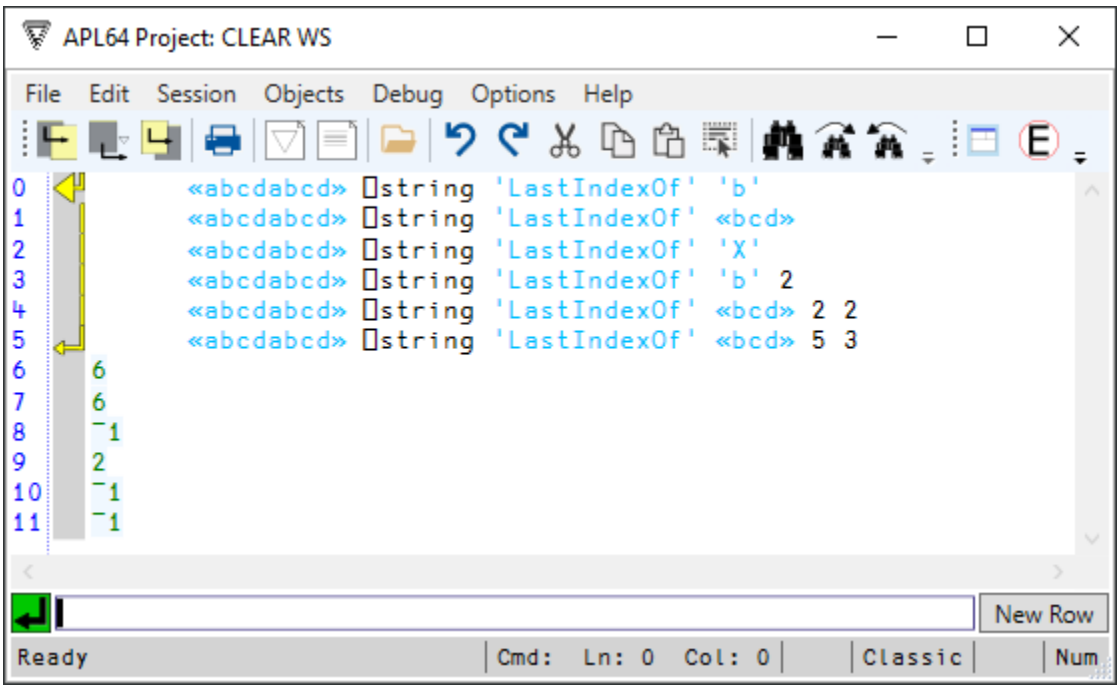

```
sv←'abc' 'def' «ghi» 'X' «Y» '∑' «∑»
«abcWdefWghiWXWYW∑W∑» ≡ sv ⎕string 'Join' 'W'
```



LASTINDEXOF

Int32← String ⎕STRING "LASTINDEXOF" Char
Int32← String ⎕STRING "LASTINDEXOF" String
Int32← String ⎕STRING "LASTINDEXOF" Char StartPosInLarg
Int32← String ⎕STRING "LASTINDEXOF" String StartPosInLarg
Int32← String ⎕STRING "LASTINDEXOF" Char StartPosInLarg #CharsToExamine
Int32← String ⎕STRING "LASTINDEXOF" String StartPosInLarg #CharsToExamine

StartPosInLarg is ⎕IO-sensitive.

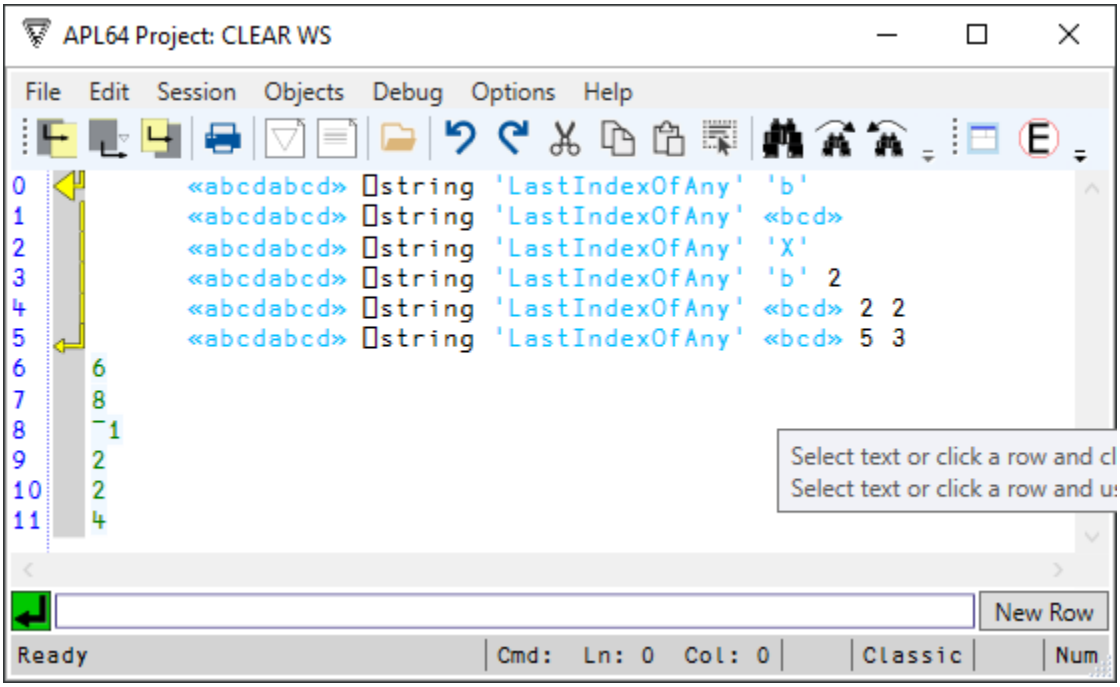


LASTINDEXOFANY

Int32←String ⎕STRING "LASTINDEXOFANY" Char[]
Int32←String ⎕STRING "LASTINDEXOFANY" Char[] StartPosInLarg
Int32←String ⎕STRING "LASTINDEXOFANY" Char[] StartPosInLarg #CharsToExamine

StartPosInLarg is ⎕IO-sensitive.

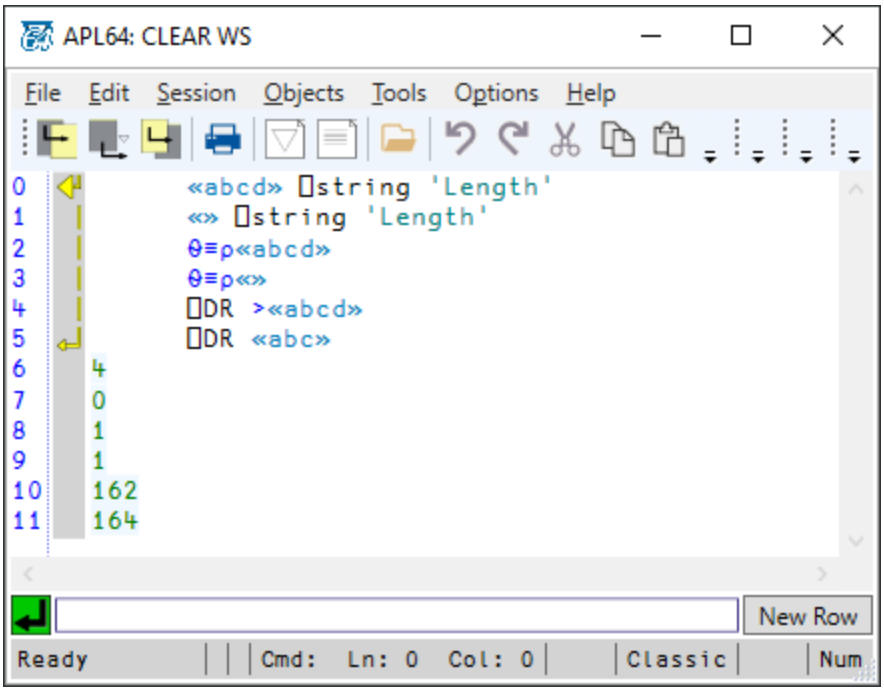
```
«abcdabcd» ⍸string 'LastIndexOfAny' 'b'
«abcdabcd» ⍸string 'LastIndexOfAny' «bcd»
«abcdabcd» ⍸string 'LastIndexOfAny' 'X'
2 ≡ «abcdabcd» ⍸string 'LastIndexOfAny' 'b' 2
«abcdabcd» ⍸string 'LastIndexOfAny' «bcd» 2 2
«abcdabcd» ⍸string 'LastIndexOfAny' «bcd» 5 3
```



LENGTH

Int32←String ⍸STRING "LENGTH"

```
«abcd» ⍸string 'Length'
«» ⍸string 'Length'
⍬≡⍴«abcd»
⍬≡⍴«»
⍸DR >«abcd»
⍸DR «abc»
```



NORMALIZE

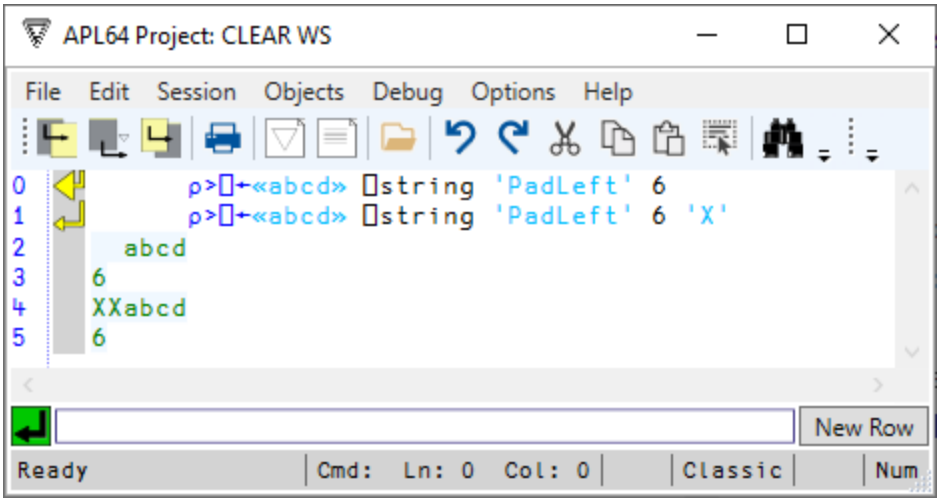
Bool←String ⎕STRING "NORMALIZE"

Refer to the ⎕STRING "ISNORMALIZED" example.

PADLEFT

String←String ⎕STRING "PADLEFT"
String←String ⎕STRING "PADLEFT" TotalWidth PadChar

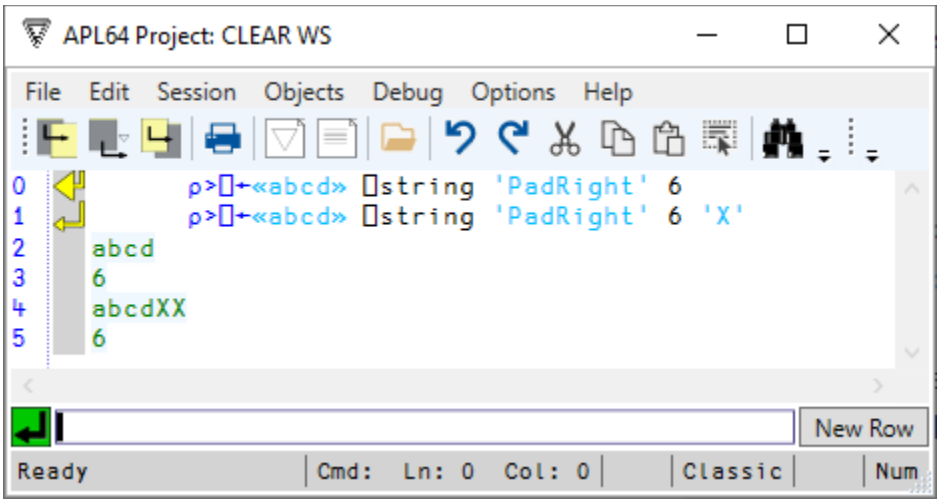
```
⍴>⎕←«abcd» ⎕string 'Padleft' 6
⍴>⎕←«abcd» ⎕string 'Padleft' 6 'X'
```



PADRIGHT

String←String ⍞STRING "PADRIGHT" TotalWidth
String←String ⍞STRING "PADRIGHT" TotalWidth PadChar

```
      ⍴⍞←«abcd» ⍞string 'Padright' 6
      ⍴⍞←«abcd» ⍞string 'Padright' 6 'X'
```

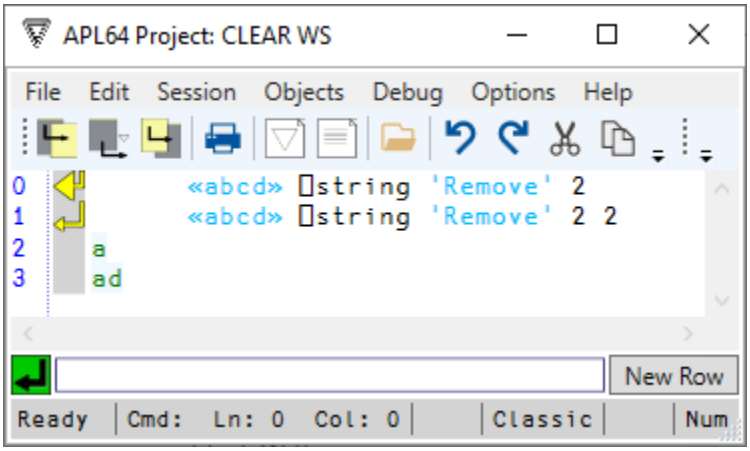


REMOVE

String←String ⍞STRING "REMOVE" StartPos
String←String ⍞STRING "REMOVE" StartPos #CharsToRemove

StartPos is ⍞IO-sensitive.

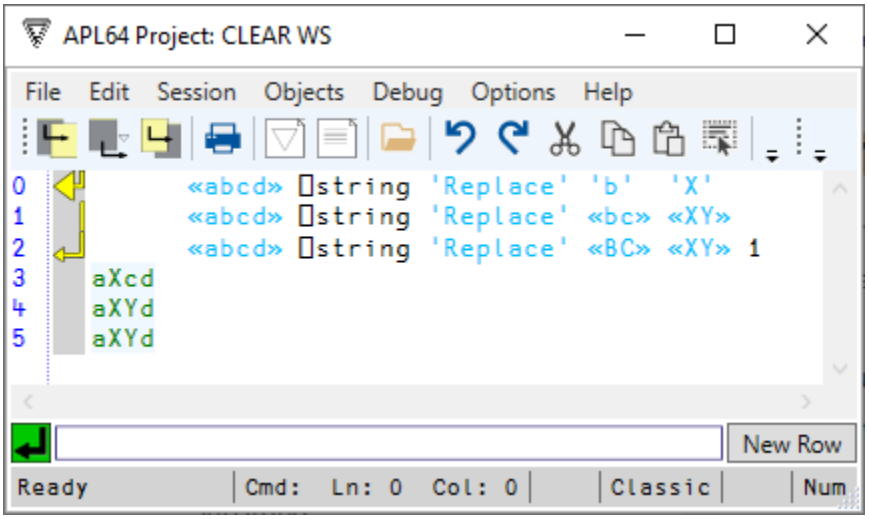
```
      «abcd» ⍞string 'Remove' 2
      «abcd» ⍞string 'Remove' 2 2
```



REPLACE

String←String ⎕STRING "REPLACE" CharTarget CharReplace
String←String ⎕STRING "REPLACE" StringTarget StringReplace
String←String ⎕STRING "REPLACE" StringTarget StringReplace IgnoreCase

```
«abcd» ⎕string 'Replace' 'b' 'X'
«abcd» ⎕string 'Replace' «bc» «XY»
«abcd» ⎕string 'Replace' «BC» «XY» 1
```



SORTINDEX

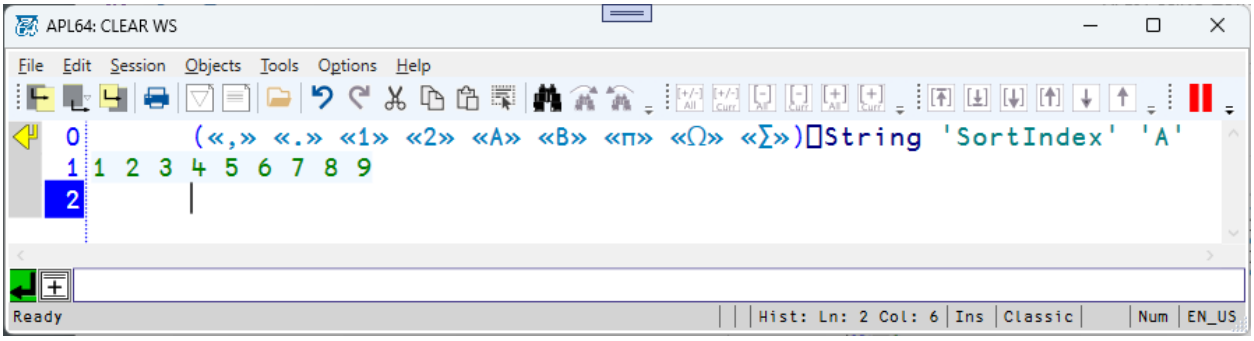
Indices←StringArray ⎕String 'SortIndex' [SortOrder] [Permutation]

StringArray is an array of string elements. StringArray may have any rank.
SortOrder is a character singleton: A(default)/Ascending, D/Descending.
Permutation is an optional text scalar or vector which may be used to adjust the sorting order of characters in the string elements of StringArray.

Indices is a vector of integers which may be used to sort StringArray according to the SortOrder.

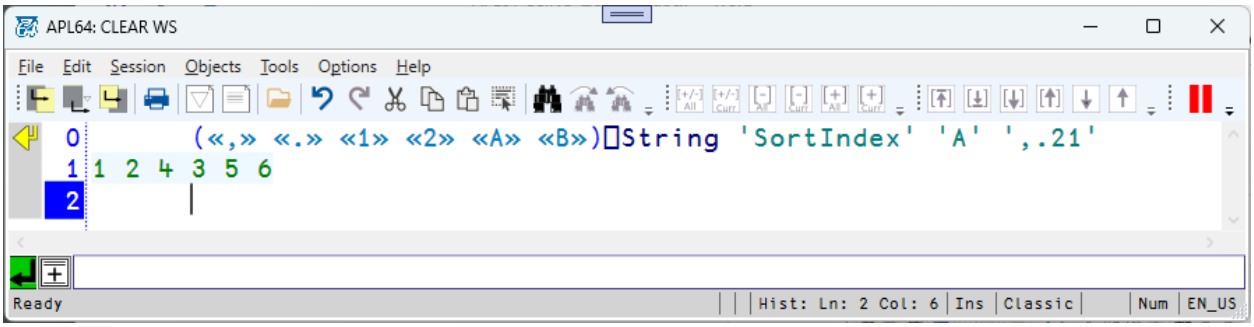
The SortIndex action uses a .Net algorithm to produce the Indices.

```
(«,» «.» «1» «2» «A» «B» «π» «Ω» «∑»)⎕String 'SortIndex' 'A'
```

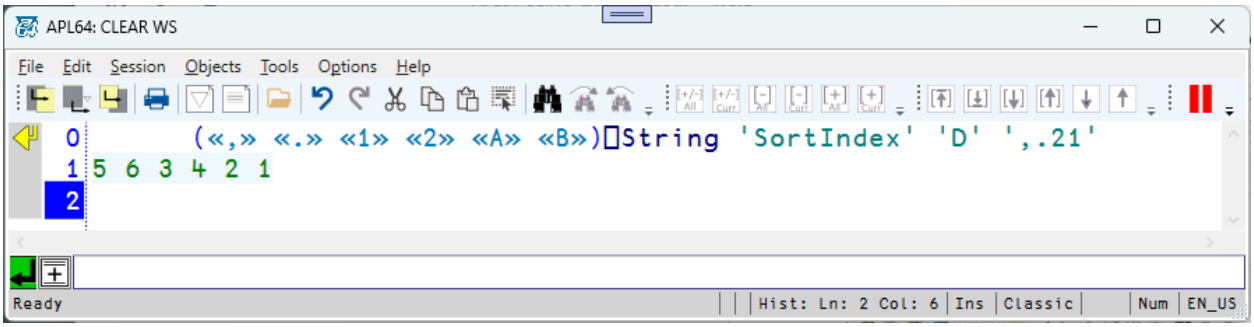


(«,» «.» «1» «2» «A» «B»)⍞String 'SortIndex' 'D'

(«,» «.» «1» «2» «A» «B»)⍞String 'SortIndex' 'A' ',.21'

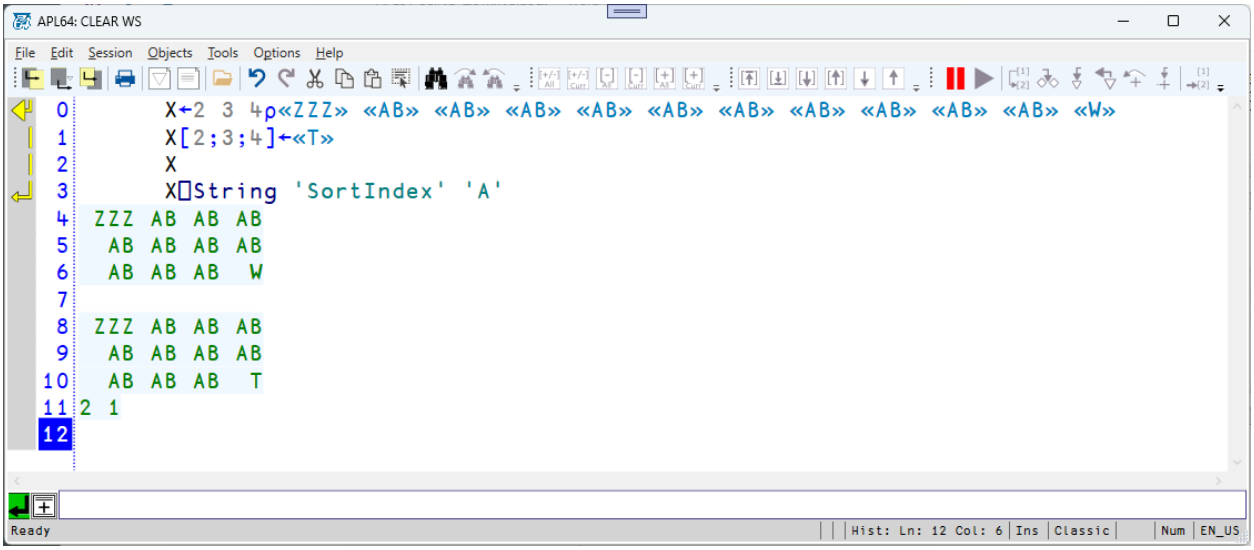


(«,» «.» «1» «2» «A» «B»)⍞String 'SortIndex' 'D' ',.21'

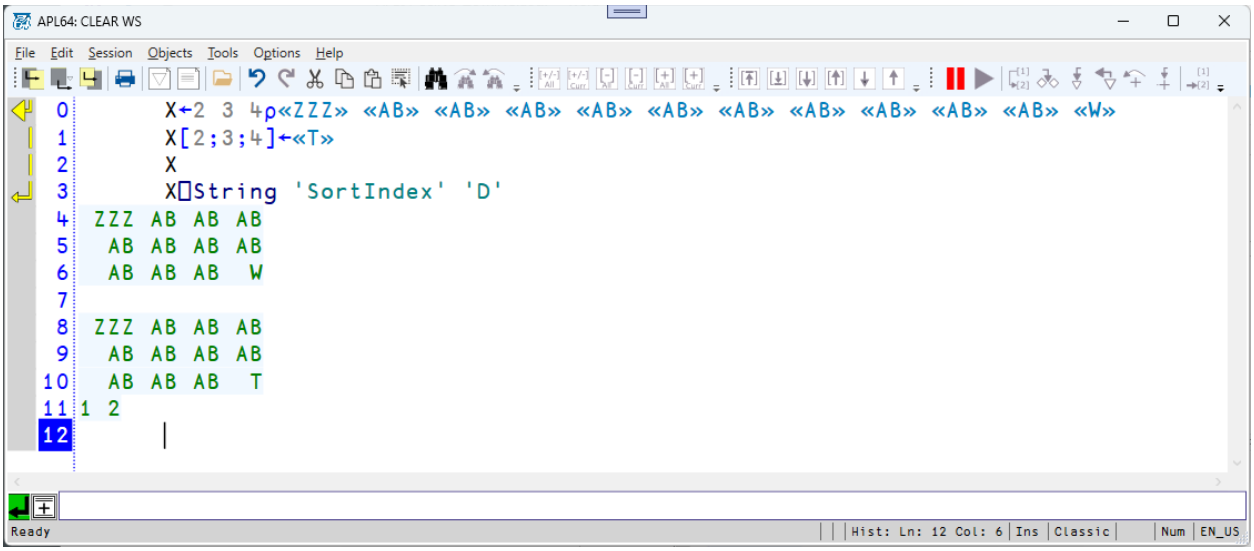


X←2 3 4⍴«ZZZ» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «W»

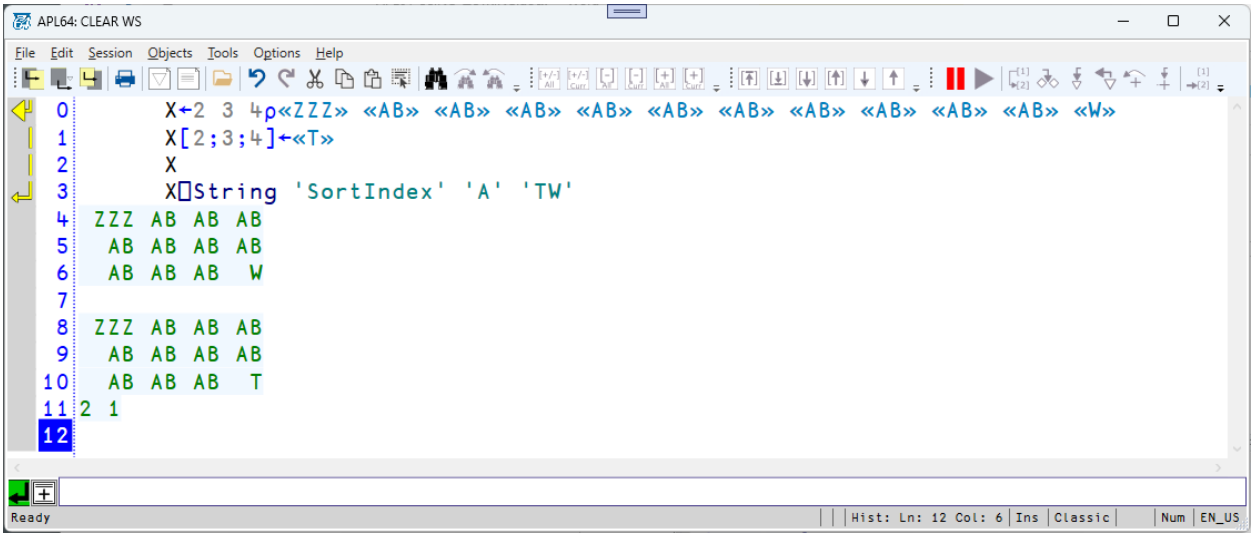
```
X[2;3;4]←«T»
X
X⎕String 'SortIndex' 'A'
```



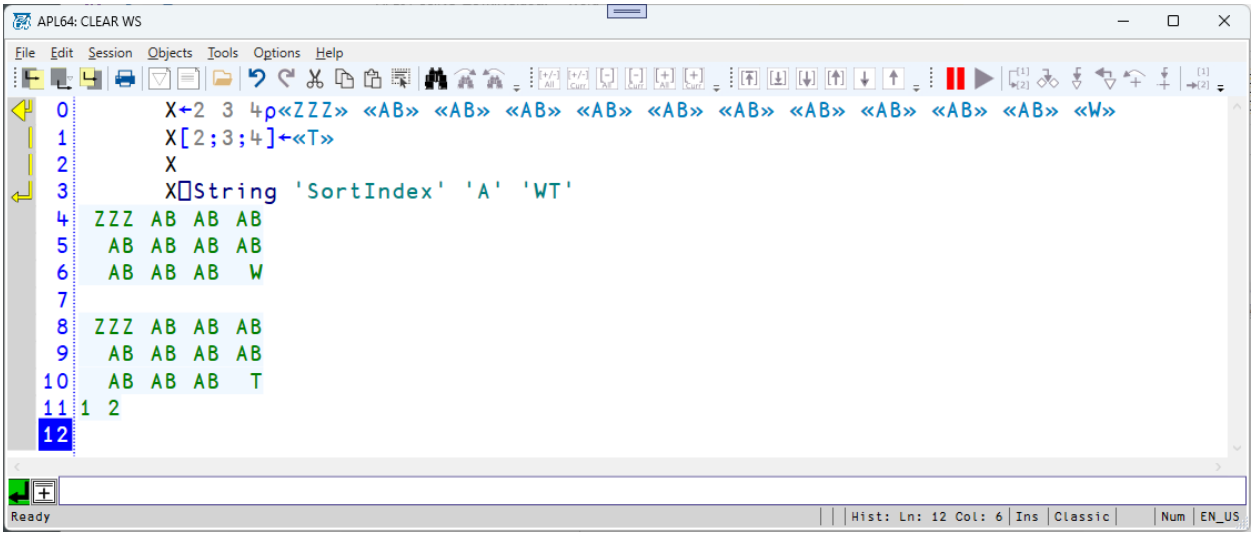

```
X←2 3 4⍴«ZZZ» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «W»
X[2;3;4]←«T»
X
X⎕String 'SortIndex' 'D'
```




```
X←2 3 4⍴«ZZZ» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «W»
X[2;3;4]←«T»
X
X⎕String 'SortIndex' 'A' 'TW'
```




```
X←2 3 4ρ«ZZZ» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «W»
X[2;3;4]←«T»
X
X⎕String 'SortIndex' 'A' 'WT'
```



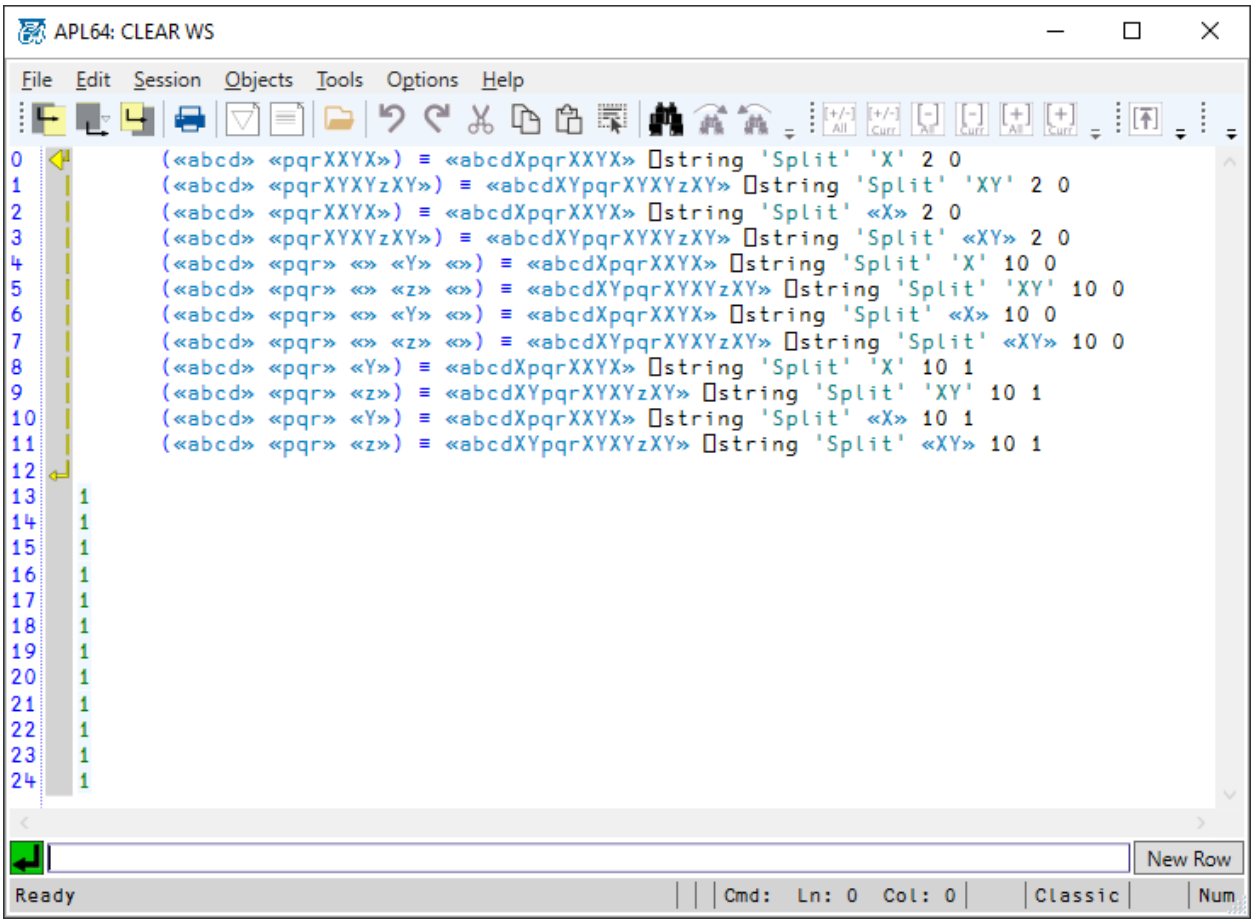
SPLIT

String[]←String ⎕STRING "SPLIT" (Char[])Delimiters Max#Substrings [RemoveEmptyEntries]
String[]←String ⎕STRING "SPLIT" (Char)Delimiter Max#Substrings [RemoveEmptyEntries]
String[]←String ⎕STRING "SPLIT" (String[])Delimiters Max#Substrings [RemoveEmptyEntries]
String[]←String ⎕STRING "SPLIT" (String)Delimiter Max#Substrings [RemoveEmptyEntries]

For no limit on the number of substrings, set Max#Substrings to -1.
The RemoveEmptyEntries argument is optional. If not present, the default is 0.

```
(«abcd» «pqrXXYX») ≡ «abcdXpqrXXYX» ⎕string 'Split' 'X' 2 0
```

```
(«abcd» «pqrXYXYzXY») ≡ «abcdXYpqrXYXYzXY» ⍷string 'Split' 'XY' 2 0
(«abcd» «pqrXXYX») ≡ «abcdXpqrXXYX» ⍷string 'Split' «X» 2 0
(«abcd» «pqrXYXYzXY») ≡ «abcdXYpqrXYXYzXY» ⍷string 'Split' «XY» 2 0
(«abcd» «pqr» «» «Y» «») ≡ «abcdXpqrXXYX» ⍷string 'Split' 'X' 10 0
(«abcd» «pqr» «» «z» «») ≡ «abcdXYpqrXYXYzXY» ⍷string 'Split' 'XY' 10 0
(«abcd» «pqr» «» «Y» «») ≡ «abcdXpqrXXYX» ⍷string 'Split' «X» 10 0
(«abcd» «pqr» «» «z» «») ≡ «abcdXYpqrXYXYzXY» ⍷string 'Split' «XY» 10 0
(«abcd» «pqr» «Y») ≡ «abcdXpqrXXYX» ⍷string 'Split' 'X' 10 1
(«abcd» «pqr» «z») ≡ «abcdXYpqrXYXYzXY» ⍷string 'Split' 'XY' 10 1
(«abcd» «pqr» «Y») ≡ «abcdXpqrXXYX» ⍷string 'Split' «X» 10 1
(«abcd» «pqr» «z») ≡ «abcdXYpqrXYXYzXY» ⍷string 'Split' «XY» 10 1
```

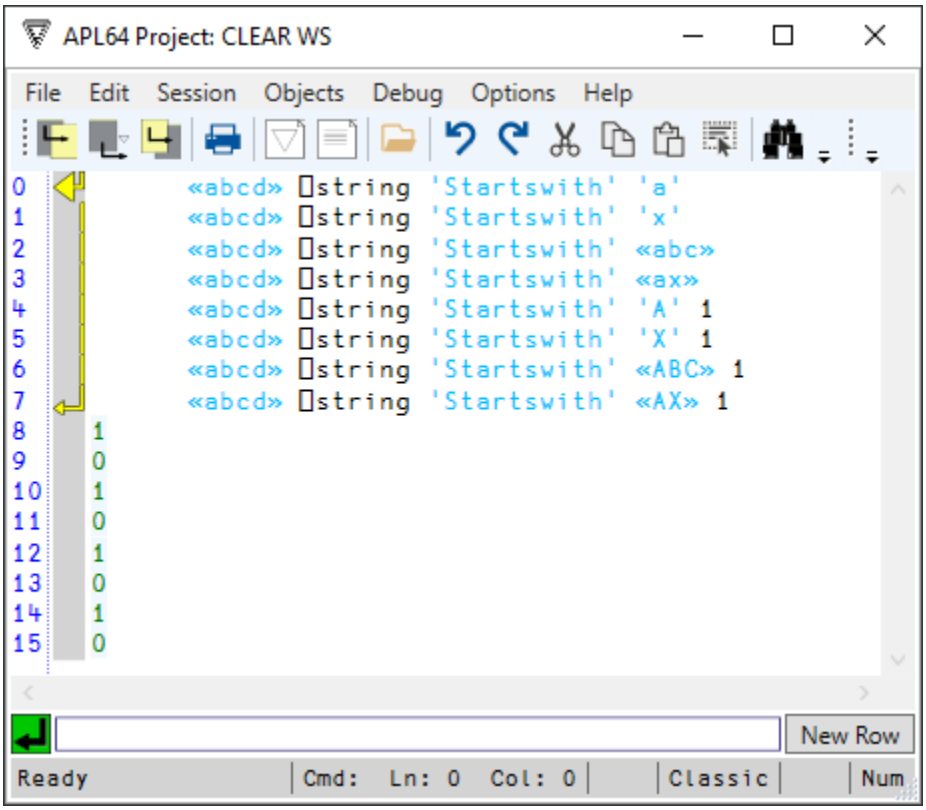


STARTSWITH

Bool←String ⍷STRING "STARTSWITH" Char
Bool←String ⍷STRING "STARTSWITH" String
Bool←String ⍷STRING "STARTSWITH" String IgnoreCase

```
«abcd» ⍷string 'StartsWith' 'a'
«abcd» ⍷string 'StartsWith' 'X'
«abcd» ⍷string 'StartsWith' «abc»
«abcd» ⍷string 'StartsWith' «ax»
```

```
«abcd» ⍷string 'StartsWith' 'A' 1
«abcd» ⍷string 'StartsWith' 'X' 1
«abcd» ⍷string 'StartsWith' «ABC» 1
«abcd» ⍷string 'StartsWith' «AX» 1
```

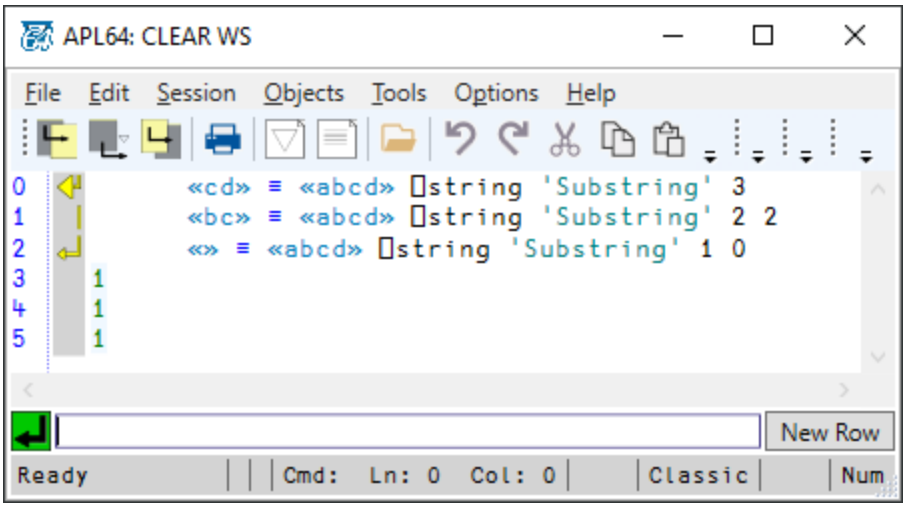


SUBSTRING

String←String ⍷STRING "SUBSTRING" (Int32)StartPos
String←String ⍷STRING "SUBSTRING" (Int32)StartPos (Int32)#Chars

StartPosInLarg is ⍷IO-sensitive.

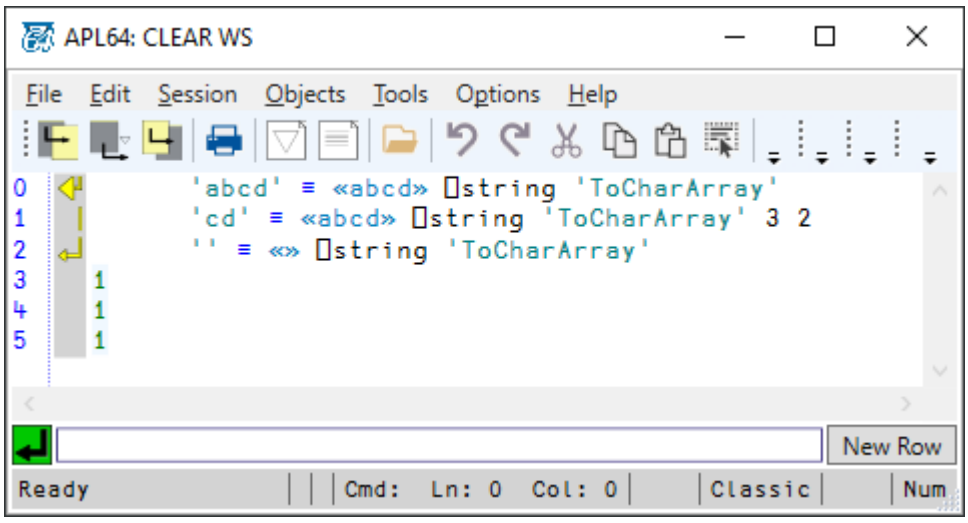
```
«cd» ≡ «abcd» ⍷string 'Substring' 3
«bc» ≡ «abcd» ⍷string 'Substring' 2 2
«» ≡ «abcd» ⍷string 'Substring' 1 0
```



TOCHARARRAY

Char[]←String ⎕STRING "TOCHARARRAY"
Char[]←String ⎕STRING "TOCHARARRAY" (Int32)StartPos (Int32)#Chars

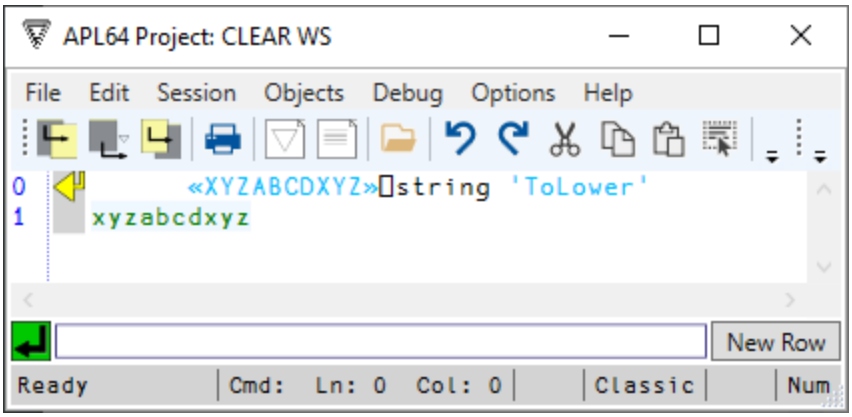
```
'abcd' ≡ «abcd» ⎕string 'ToCharArray'
'cd' ≡ «abcd» ⎕string 'ToCharArray' 3 2
'' ≡ «» ⎕string 'ToCharArray'
```



TOLOWER

Use current culture: String←String ⎕STRING "ToLower"

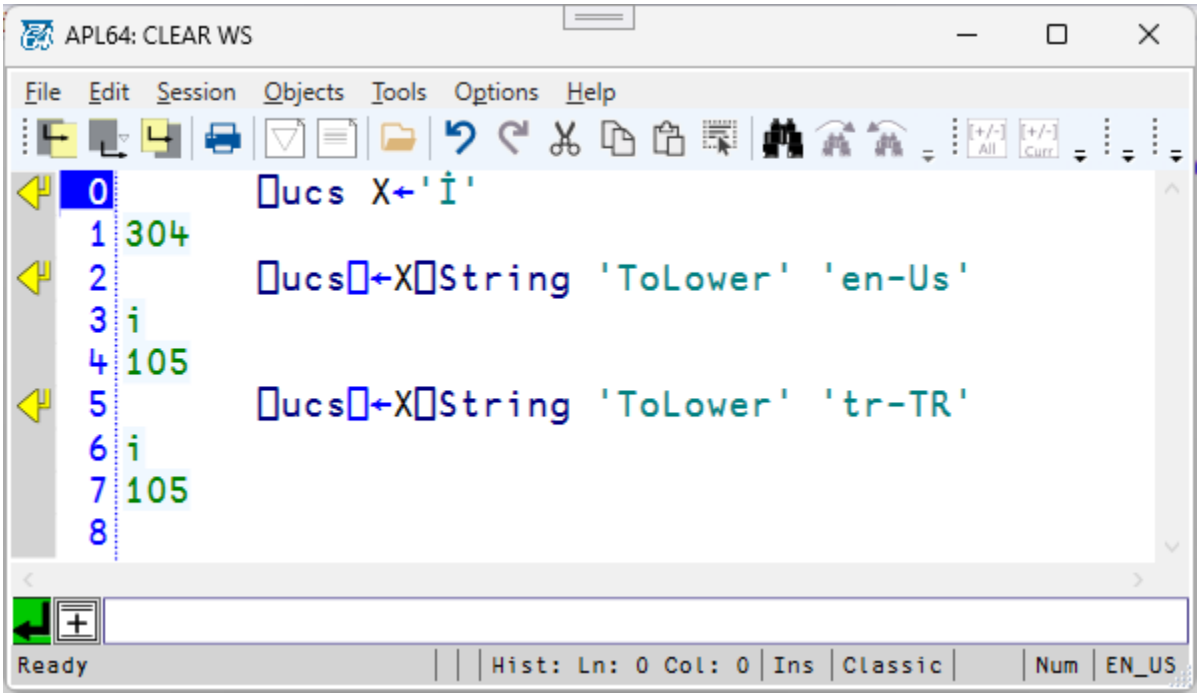
```
«XYZABCDXYZ» ⎕string 'ToLower'
```



Use selected culture: String←String ⎕STRING "ToLower" "CultureName"

The available options for CultureName depend on the operating system environment, e.g. for the Windows operating system environment. Use ⎕string 'GetCultureInfo' at application runtime to obtain a list of available culture names for the current operation system environment. The .Net-supported culture name is not case-sensitive

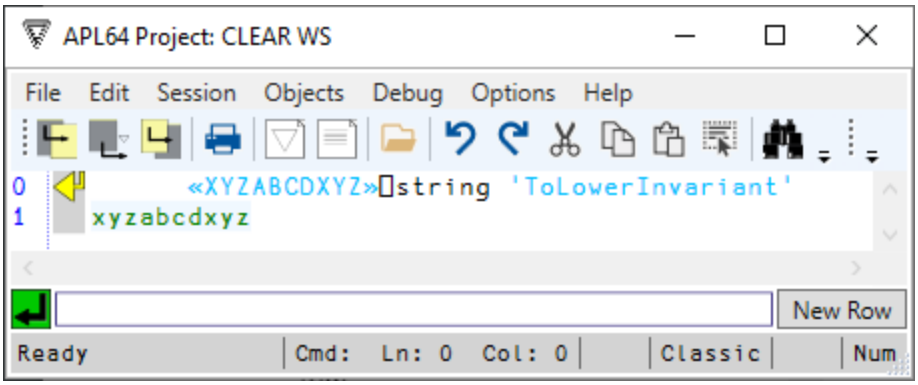
```
⎕ucs⎕←(⎕←«ABC123İ»)⎕String 'ToLower' 'en-US'
```



TOLOWERINVARIANT

String←String ⎕STRING "TOLOWERINVARIANT"

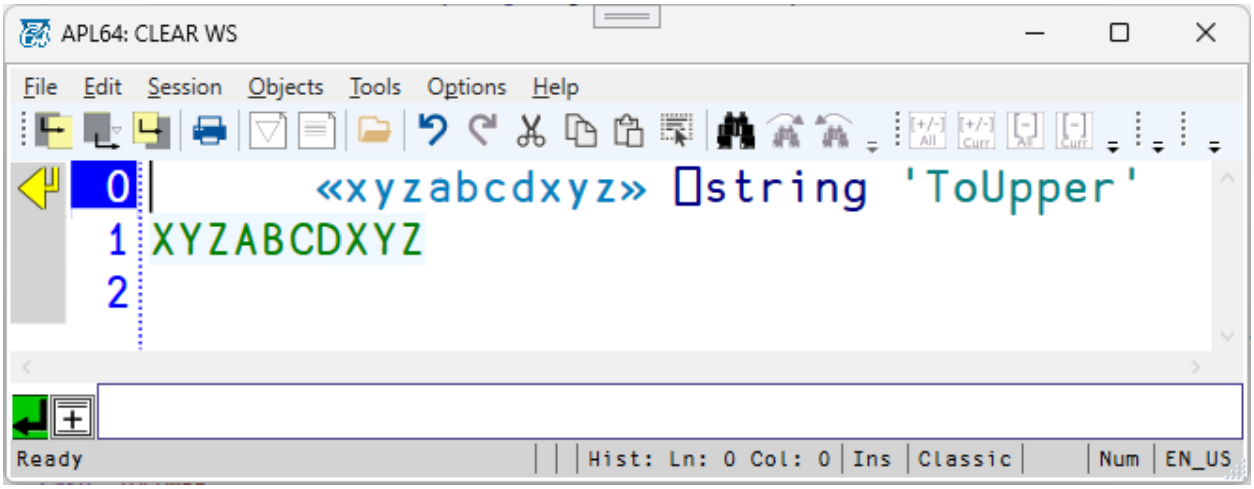
```
«XYZABCDXYZ» ⎕string 'ToLowerInvariant'
```



TOUPPER

Use current culture: String←String ⎕STRING "TOUPPER"

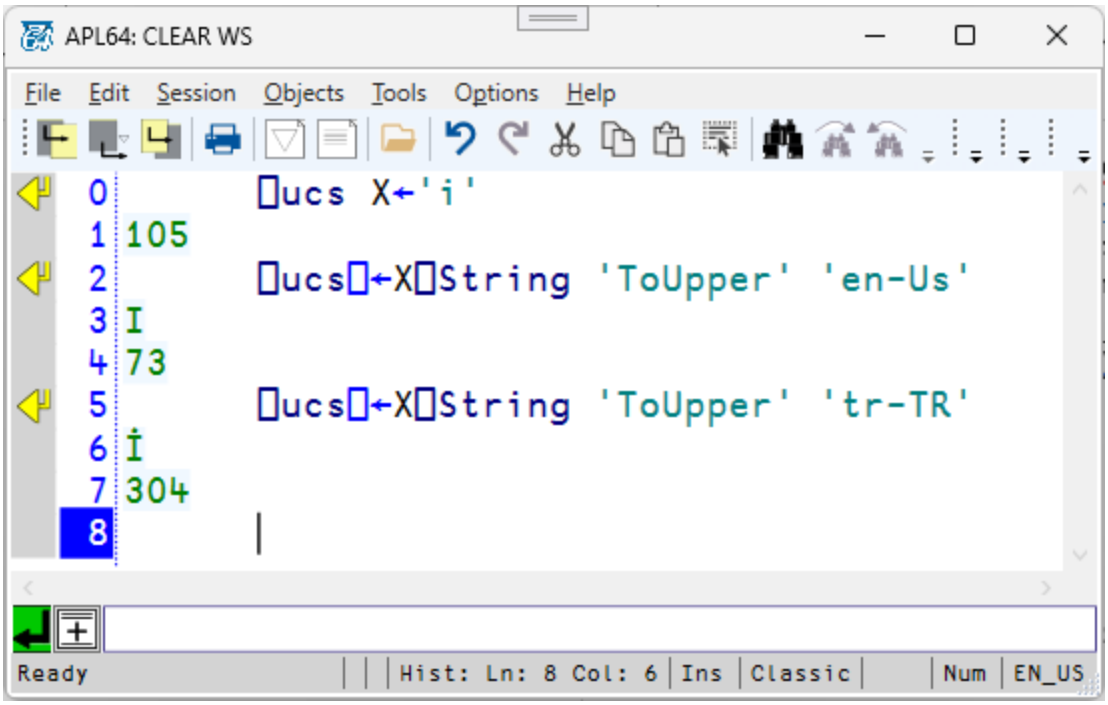
```
«xyzabcdxyz» ⎕string 'ToUpper'
```



Use selected culture: String←String ⎕STRING "TOUPPER" "CultureName"

The available options for CultureName depend on the operating system environment, e.g. for the Windows operating system environment. Use ⎕string 'GetCultureInfo' at application runtime to obtain a list of available culture names for the current operation system environment. The .Net-supported culture name is not case-sensitive

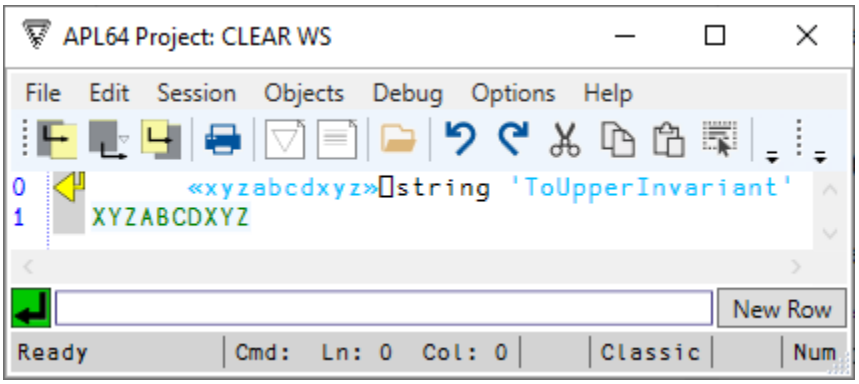
```
⎕ucs⎕←(⎕←«abc123i»)⎕String 'ToUpper' 'en-US'
```



TOUPPERINVARIANT

String←String ⎕STRING "TOUPPERINVARIANT"

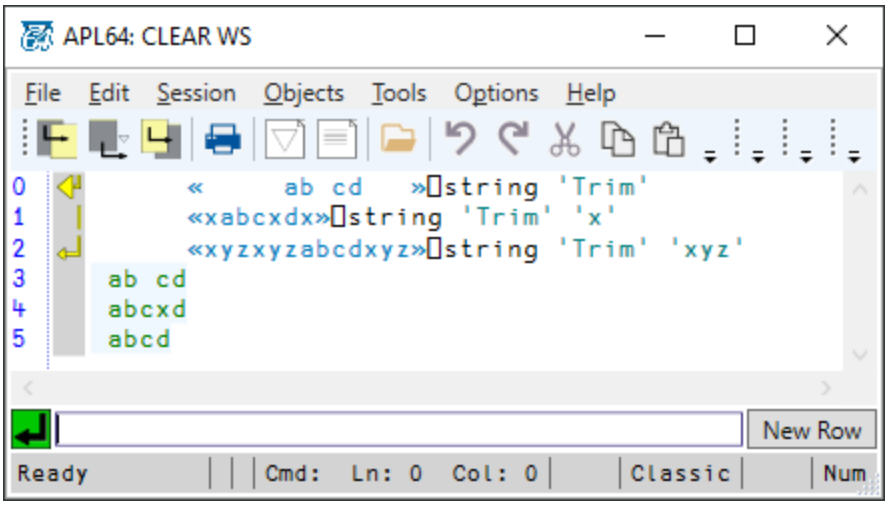
```
«xyzabcdxyz» ⎕string 'ToUpperInvariant'
```



TRIM

String←String ⎕STRING "TRIM"
String←String ⎕STRING "TRIM" (char)charToTrim
String←String ⎕STRING "TRIM" (char[])charsToTrim

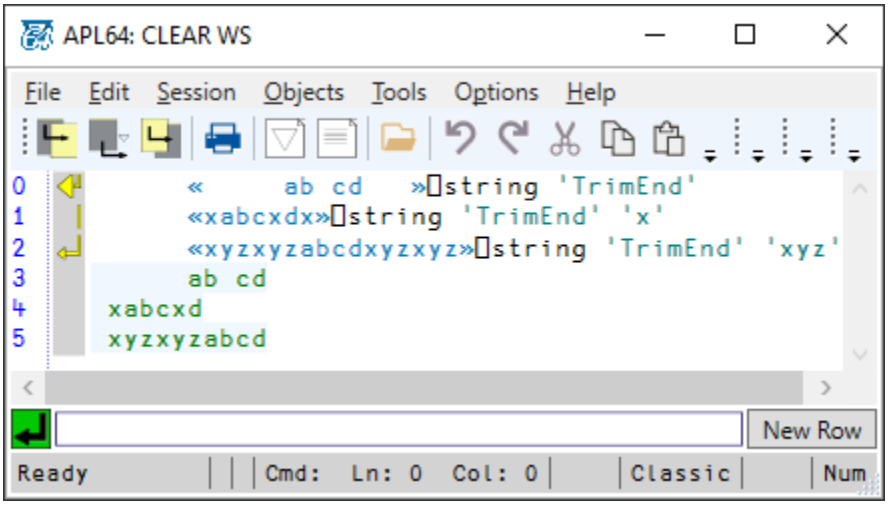
```
«    ab cd  »⎕string 'Trim'
«xabcxdx»⎕string 'Trim' 'x'
«xyzxyzabcdxyz»⎕string 'Trim' 'xyz'
```



TRIMEND

String←String ⎕STRING "TRIMEND"
String←String ⎕STRING "TRIMEND" (char)charToTrim
String←String ⎕STRING "TRIMEND" (char[])charsToTrim

```
«    ab cd   »⎕string 'TrimEnd'
«xabcxdx»⎕string 'TrimEnd' 'x'
«xyzxyzabcdxyzxyz»⎕string 'TrimEnd' 'xyz'
```

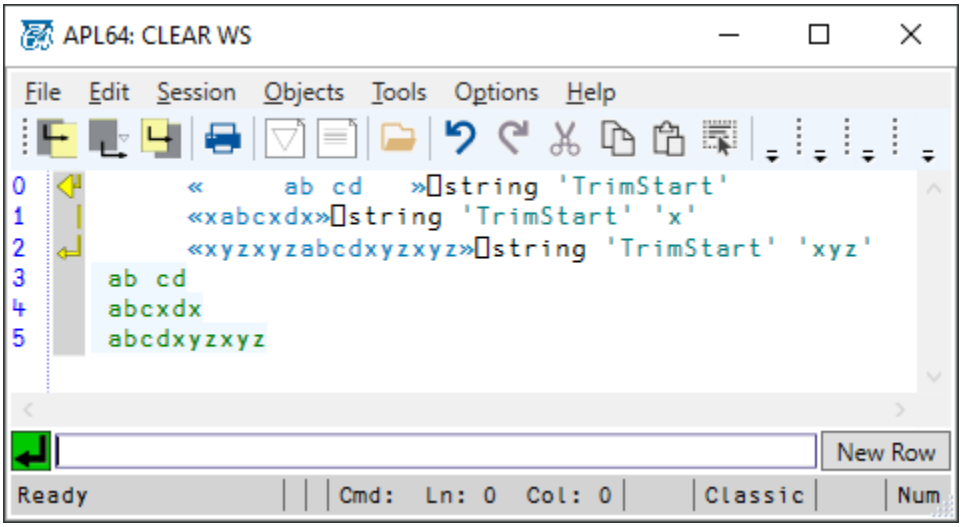


TRIMSTART

String←String ⎕STRING "TRIMSTART"
String←String ⎕STRING "TRIMSTART" (char)charToTrim
String←String ⎕STRING "TRIMSTART" (char[])charsToTrim

```
«    ab cd   »⎕string 'TrimStart'
«xabcxdx»⎕string 'TrimStart' 'x'
```

```
«xyzxyzabcdxyzxyz»⎕string 'TrimStart' 'xyz'
```



? (Documentation Summary)

Char[] ← String ⎕STRING "?"

```
⎕dr⎕String '?'
⍴⎕String '?'
1500↑⎕String '?'
```