

APL64: Using ☐String

Contents

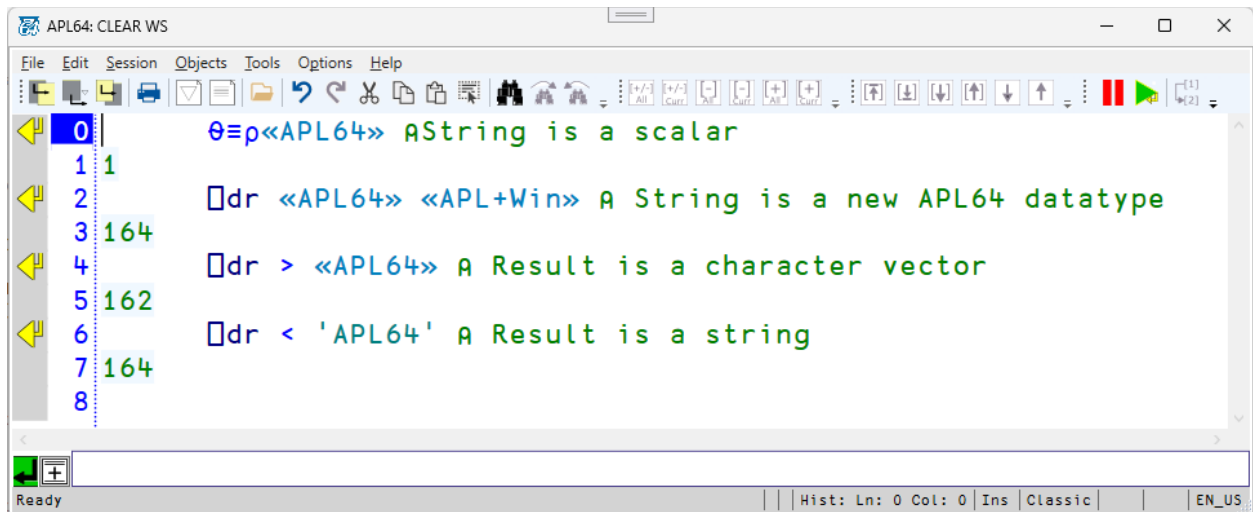
String Datatype in APL64	2
☐ String System Function	2
☐ String Actions	3
COMPARE.....	3
COMPAREORDINAL.....	4
CONCAT.....	5
CONTAINS	6
ENDSWITH.....	7
EQUALS	7
GETCULTUREINFO	8
GETHASHCODE.....	9
INDEXOF.....	9
INDEXOFANY	10
INSERT	11
ISNORMALIZED.....	12
ISNULLOREMPY	12
ISNULLORWHITESPACE	13
JOIN	13
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 destrstring 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
```

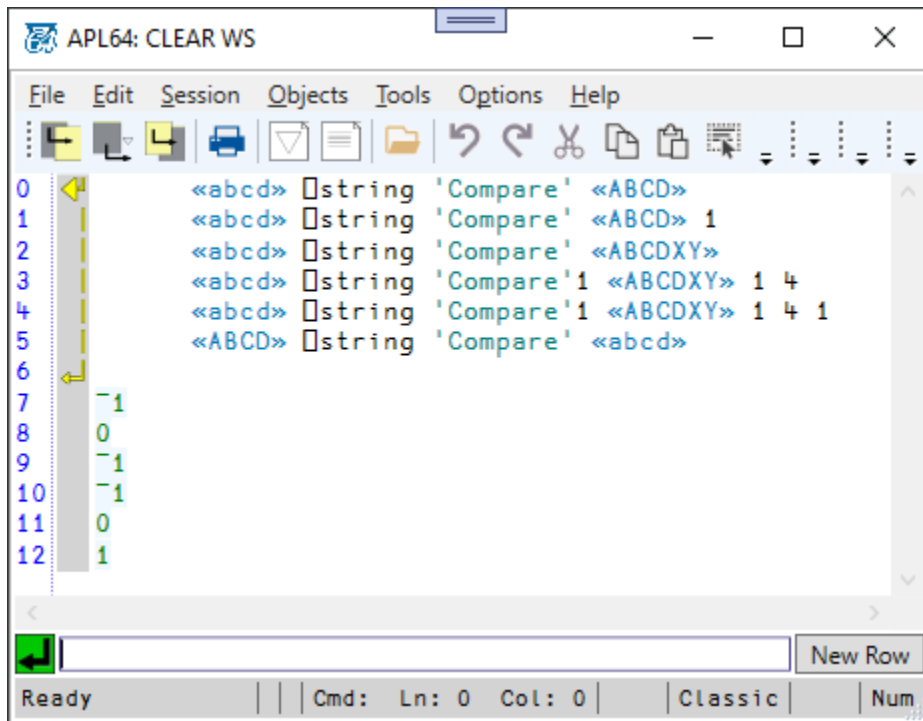
```
Int32←String ⊠STRING "COMPARE" String IgnoreCase
```

```
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»



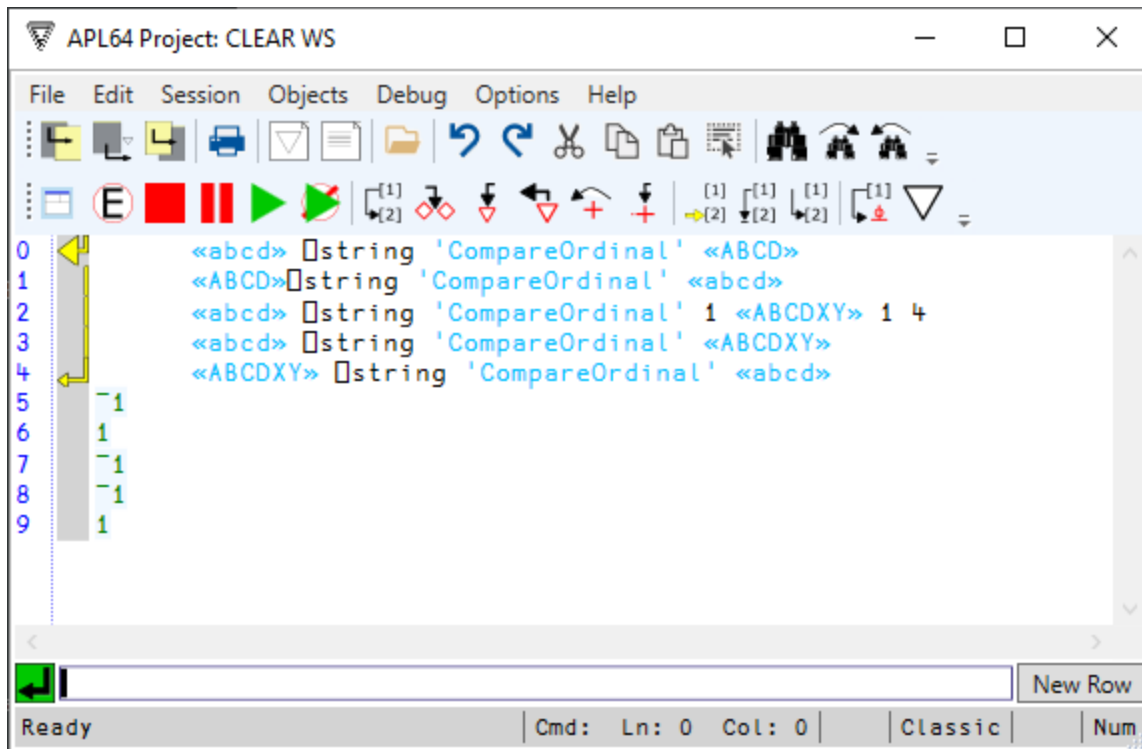
COMPAREORDINAL

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»)

```

```

0  «abcd» ⍵string 'Concat' «xyz»
1  «abcd» ⍵string 'Concat' «xyz» «pq»
2  «abcd» ⍵string 'Concat' «xyz» «pq» «01234»
3  «abcd» ⍵string 'Concat' («xyz» «pq» «01234»)
4  «abcd», «xyz», «pq», «01234»
5  abcdxyz
6  abcdxyzpq
7  abcdxyzpq01234
8  abcdxyzpq01234
9  abcd xyz pq 01234

```

Ready | Cmd: Ln: 0 Col: 0 | Classic | Num

«abcdefghiXYΣΣ» ≡ 'abc' ⍵string 'Concat' ('def' «ghi» 'X' «Y» 'Σ' «Σ»)

```

0  «abcdefghiXYΣΣ» ≡ 'abc' ⍵string 'Concat' ('def' «ghi» 'X' «Y» 'Σ' «Σ»)
1  1
2  |

```

Ready | Hist: Ln: 2 Col: 6 | Ins | Classic | Num | EN_US

CONTAINS

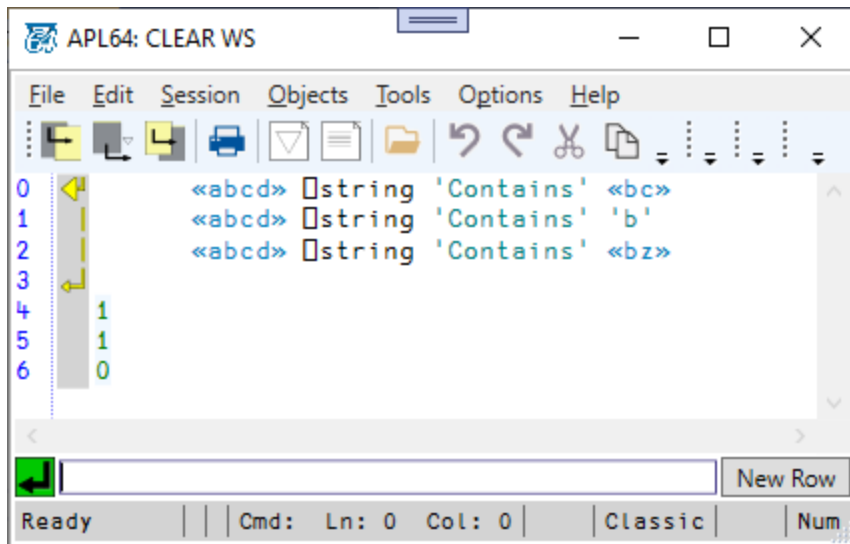
Bool←String ⍵STRING "CONTAINS" String

Bool←String ⍵STRING "CONTAINS" Char

```

«abcd» ⍵string 'Contains' «bc»
«abcd» ⍵string 'Contains' 'b'
«abcd» ⍵string 'Contains' «bz»

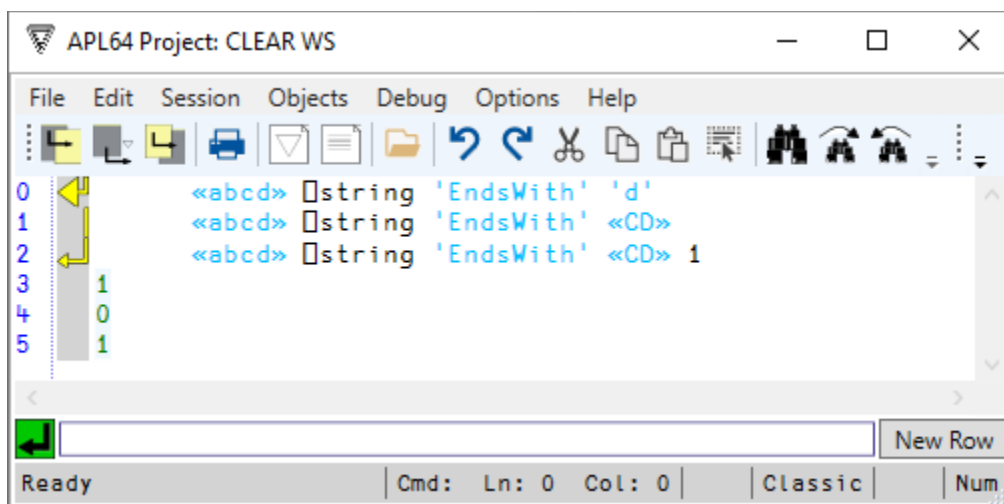
```



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

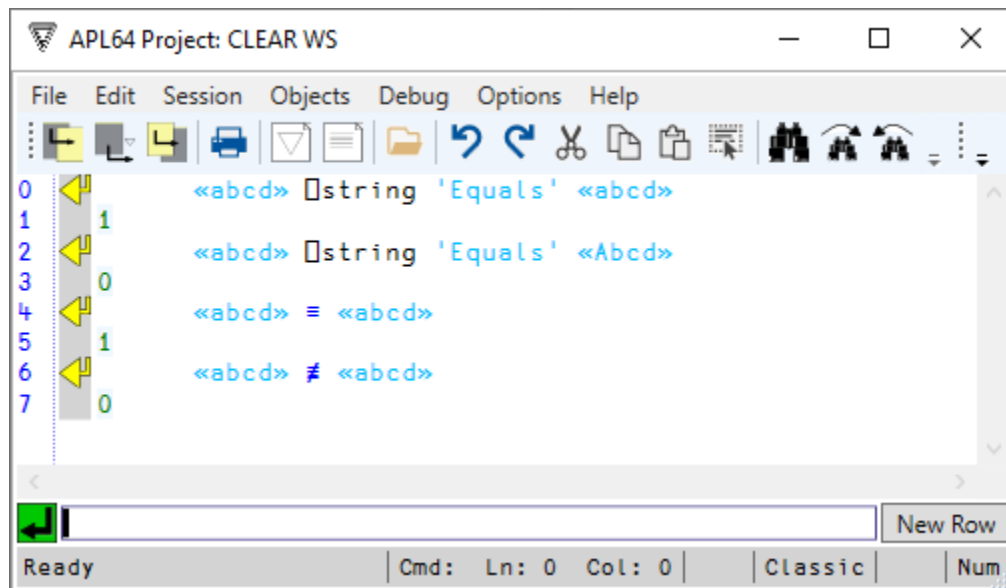


EQUALS

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'  
pci  
⎕dr ci  
ci[1 184 250 348 126;]  
p>⎕←ci[250;2]  
⎕os
```

```

APL64: CLEAR WS
0  ci←string 'GetCultureInfo'
1  pci
2  870 3
3  dr ci
4  326
5  ci[1 184 250 348 126:]
6  English Name      languageCode-CountryCode Display Name
7  English (United Kingdom) en-GB      English (United Kingdom)
8  English (United States) en-US      English (United States)
9  French (France)    fr-FR      French (France)
10 German (Germany)   de-DE      German (Germany)
11 p←ci[250;2]
12 en-US
13 5
14 os
15 Microsoft Windows 10.0.22631
16

```

GETHASHCODE

```
Int32←String STRING "GETHASHCODE"
```

The result values will not transcend an APL64 instance.

```

APL64 Project: CLEAR WS
File Edit Session Objects Debug Options Help
0  <<abcd> string 'GetHashCode'
1  1416555597
2  <<ABCD> string 'GetHashCode'
3  -1490879826
4  <<0> string 'GetHashCode'
5  -699925257

```

INDEXOF

```
Int32←String STRING "INDEXOF" Char
```

```
Int32←String STRING "INDEXOF" String
```

```
Int32←String STRING "INDEXOF" Char StartPosInLarg
```

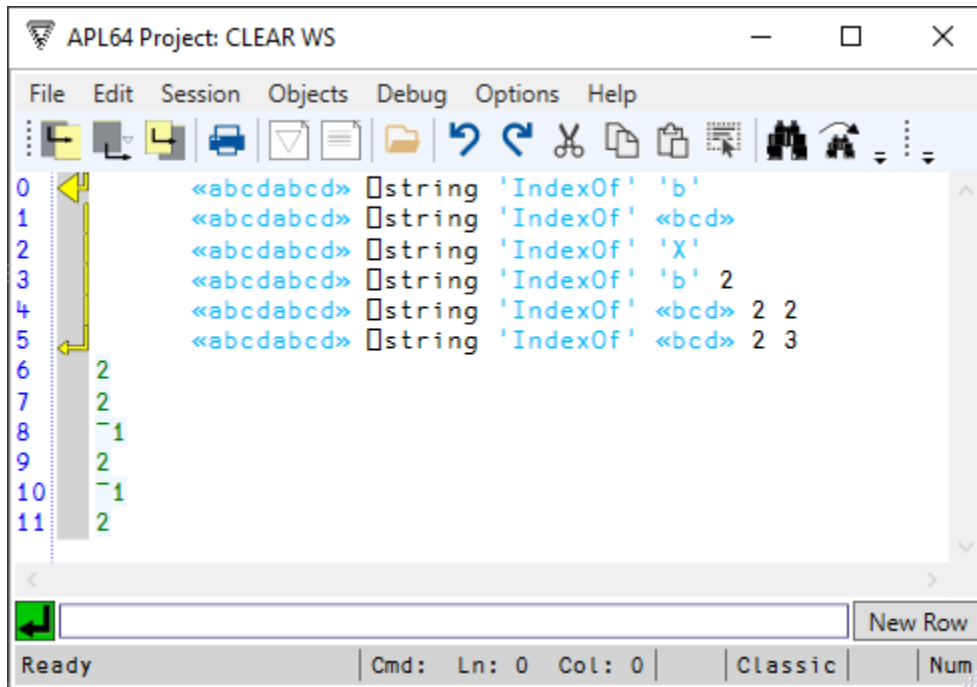
```
Int32←String STRING "INDEXOF" String StartPosInLarg
```

```
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

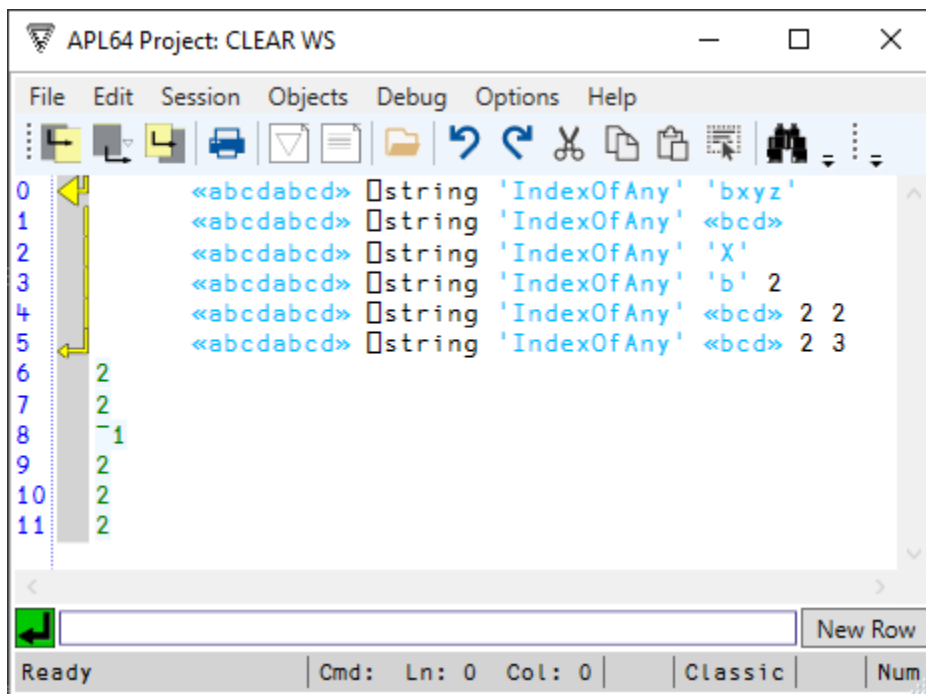
Int32←String ☐ STRING "INDEXOFANY" String

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

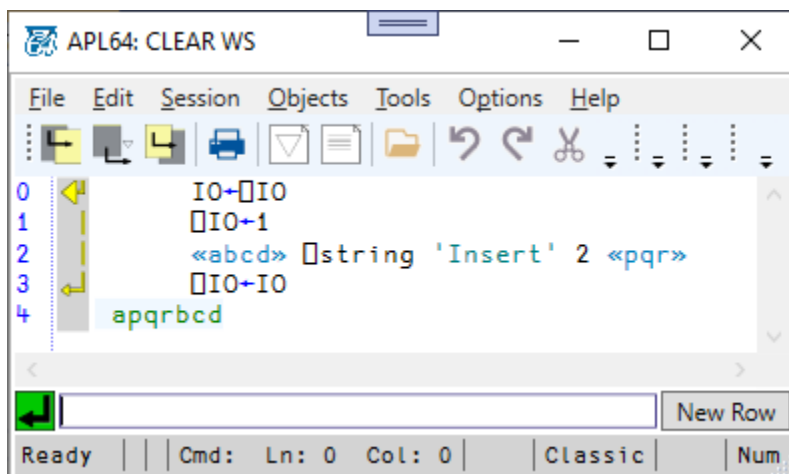


INSERT

String←String ⍺STRING "INSERT" StartPosInLarg String

StartPosInLarg is ⍺IO-sensitive.

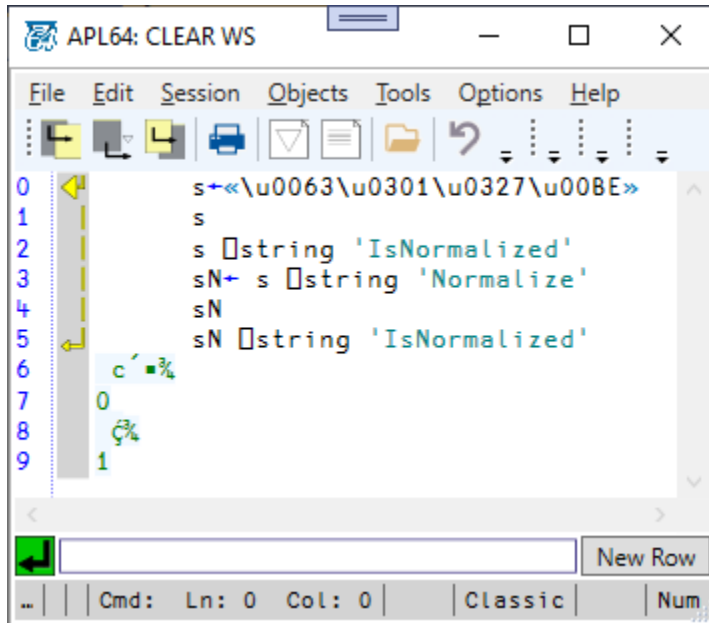
```
IO←⍺IO
⍺IO←1
«abcd» ⍺string 'Insert' 2 «pqr»
⍺IO←IO
```



ISNORMALIZED

String←String □STRING "ISNORMALIZED"

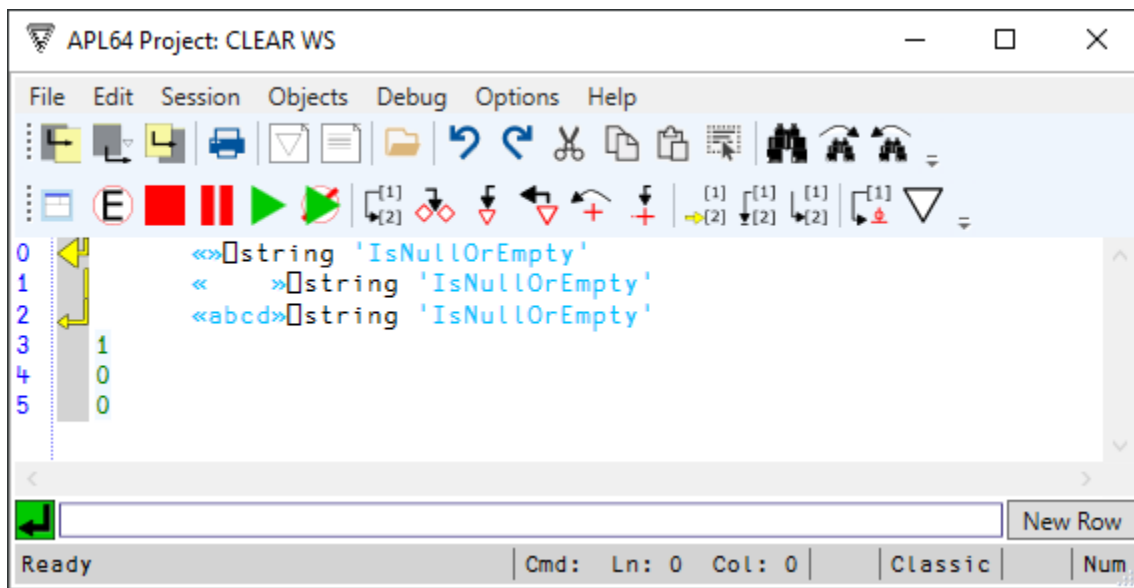
```
s←«\u0063\u0301\u0327\u00BE»
s
s □string 'IsNormalized'
sN← s □string 'Normalize'
sN
sN □string 'IsNormalized'
```



ISNULLOREEMPTY

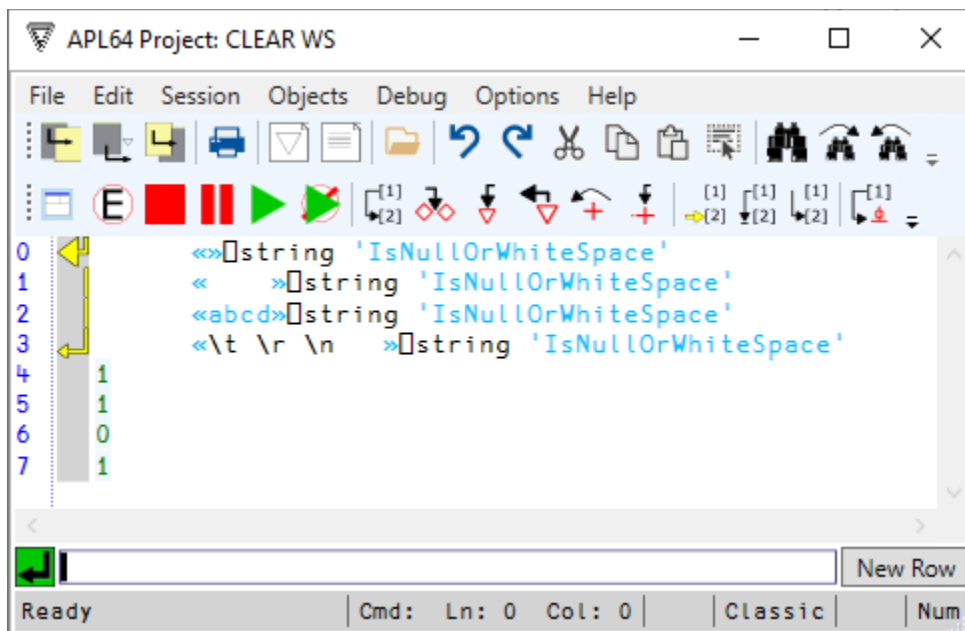
String←String □STRING "ISNULLOREEMPTY"

```
«» □string 'IsNullOrEmpty'
« » □string 'IsNullOrEmpty'
«abcd» □string 'IsNullOrEmpty'
```



ISNULLORWHITESPACE

String←String []STRING "ISNULLORWHITESPACE"



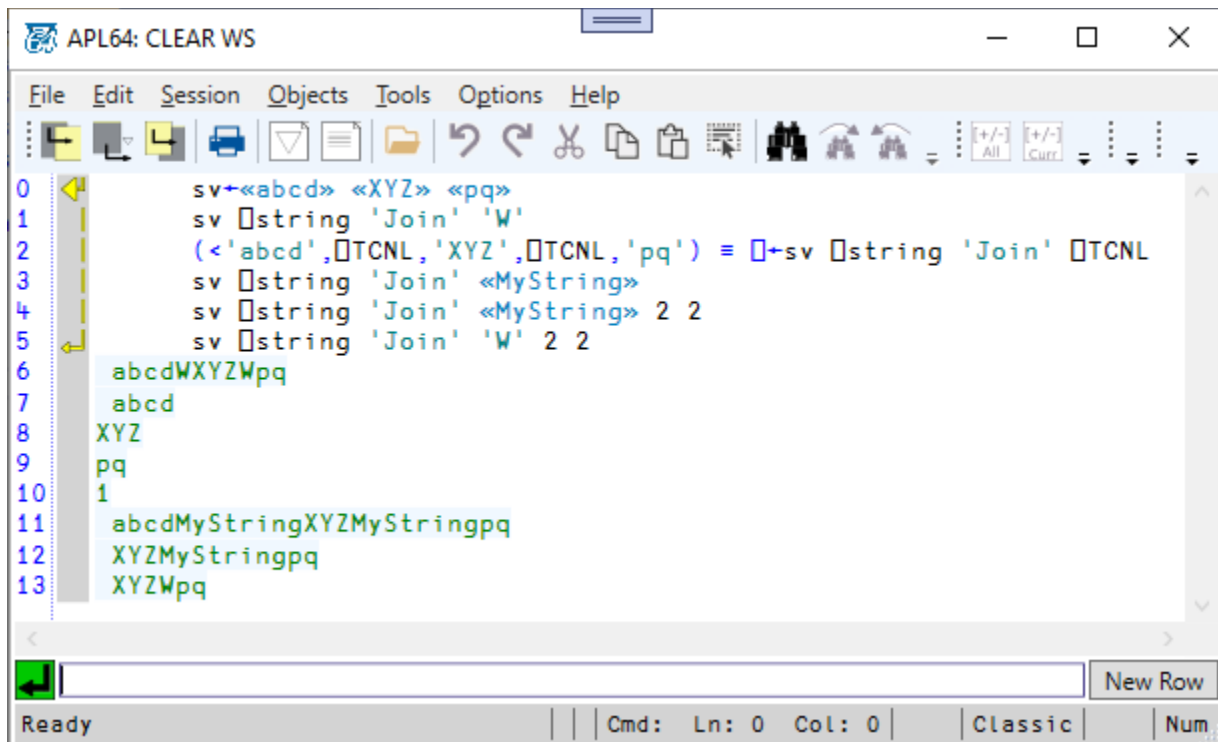
JOIN

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.

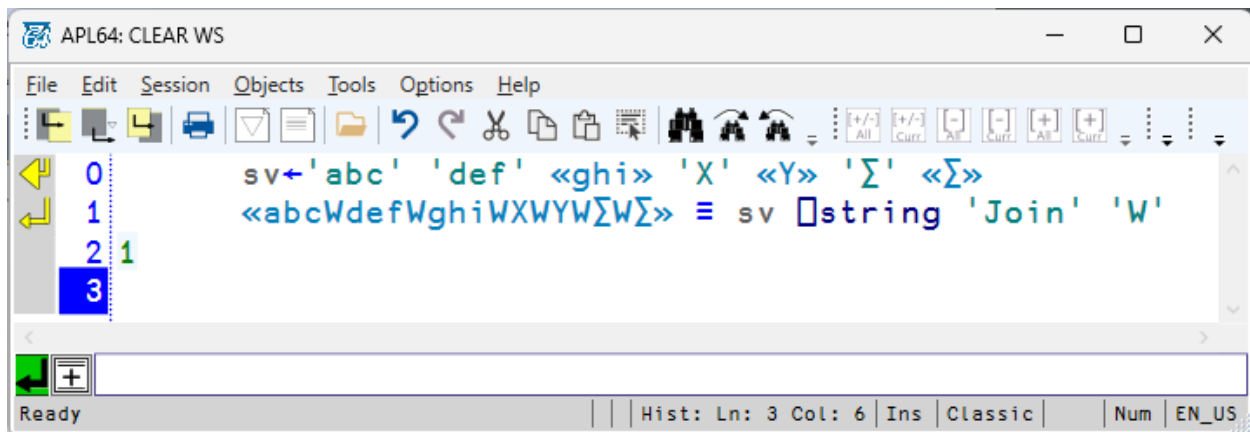
```
sv←«abcd» «XYZ» «pq»
sv ☐string 'Join' 'W'
(<'abcd',☐TCNL,'XYZ',☐TCNL,'pq') ≡ ☐←sv ☐string 'Join' ☐TCNL
sv ☐string 'Join' «MyString»
sv ☐string 'Join' «MyString» 2 2
sv ☐string 'Join' 'W' 2 2
```

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



```
APL64: CLEAR WS
File Edit Session Objects Tools Options Help
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
abcdWXYZWpq
abcd
XYZ
pq
1
abcdMyStringXYZMyStringpq
XYZMyStringpq
XYZWpq
```

```
sv←'abc' 'def' «ghi» 'X' «Y» 'Σ' «Σ»
«abcWdefWghiWXWYWΣWΣ» ≡ sv ☐string 'Join' 'W'
```



LASTINDEXOF

Int32← String ☐ STRING "LASTINDEXOF" Char

Int32← String ☐ STRING "LASTINDEXOF" String

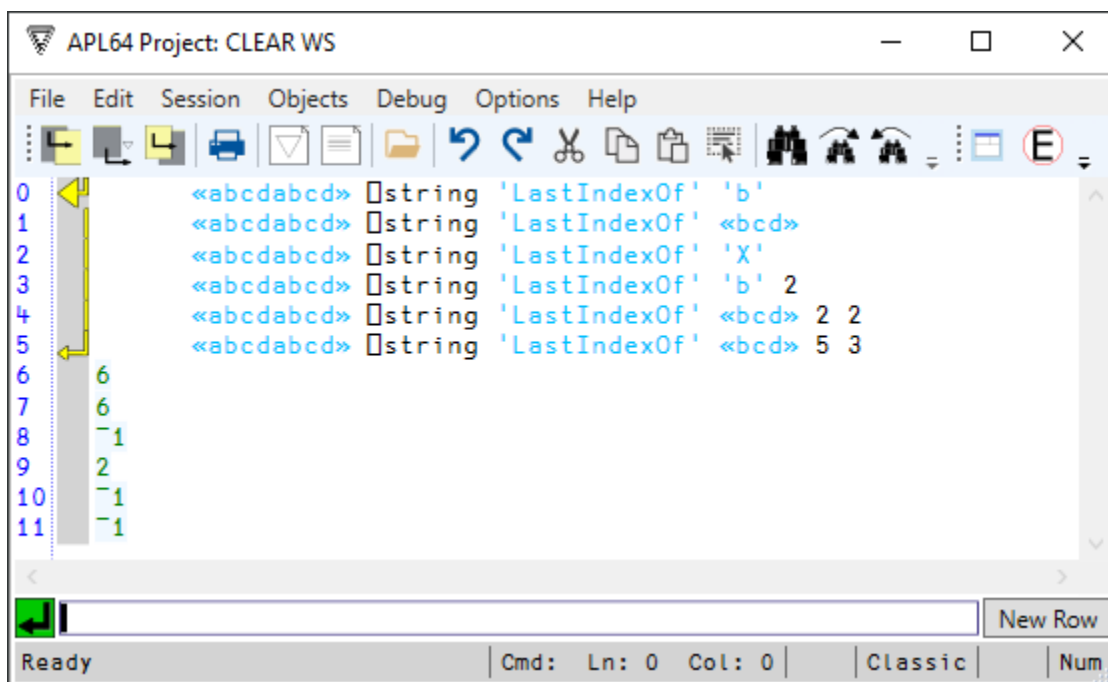
Int32← String ☐ STRING "LASTINDEXOF" Char StartPosInLarg

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

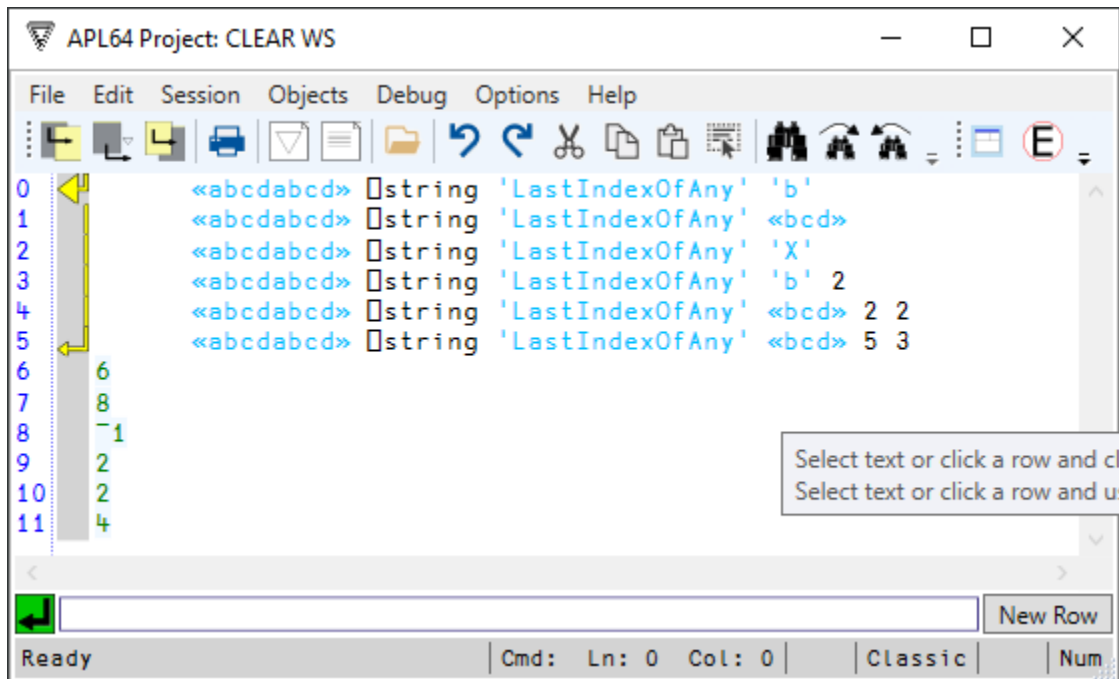
Int32←String ☐ STRING "LASTINDEXOFANY" Char[]

Int32←String ☐ STRING "LASTINDEXOFANY" Char[] StartPosInLarg

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

StartPosInLarg is ☐ IO-sensitive.

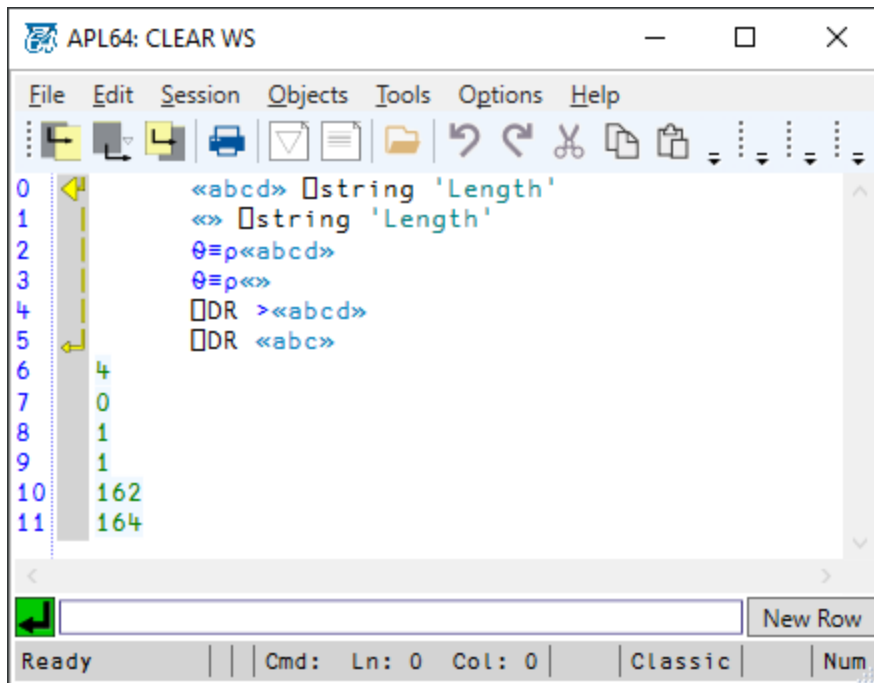
```
«abcdabcd» ☐ string 'LastIndexOfAny' 'b'
«abcdabcd» ☐ string 'LastIndexOfAny' «bcd»
«abcdabcd» ☐ string 'LastIndexOfAny' 'X'
2 ≡ «abcdabcd» ☐ string 'LastIndexOfAny' 'b' 2
«abcdabcd» ☐ string 'LastIndexOfAny' «bcd» 2 2
«abcdabcd» ☐ string 'LastIndexOfAny' «bcd» 5 3
```



LENGTH

Int32←String ☐ STRING "LENGTH"

```
«abcd» ☐ string 'Length'
«» ☐ string 'Length'
θ≡ρ«abcd»
θ≡ρ«»
☐ DR >«abcd»
☐ DR «abc»
```



NORMALIZE

Bool←String ⍵STRING "NORMALIZE"

Refer to the ⍵STRING "ISNORMALIZED" example.

PADLEFT

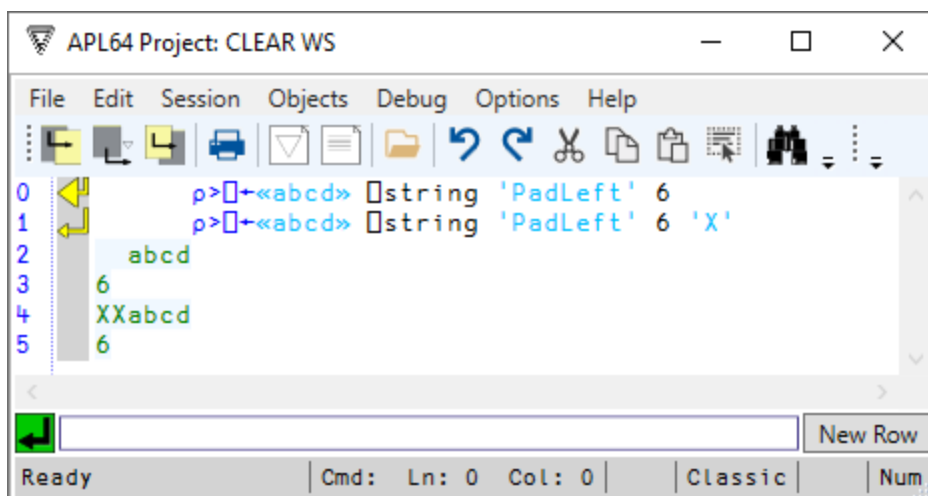
String←String ⍵STRING "PADLEFT"

String←String ⍵STRING "PADLEFT" TotalWidth PadChar

```

ρ>⍵←«abcd» ⍵string 'Padleft' 6
ρ>⍵←«abcd» ⍵string 'Padleft' 6 'X'

```

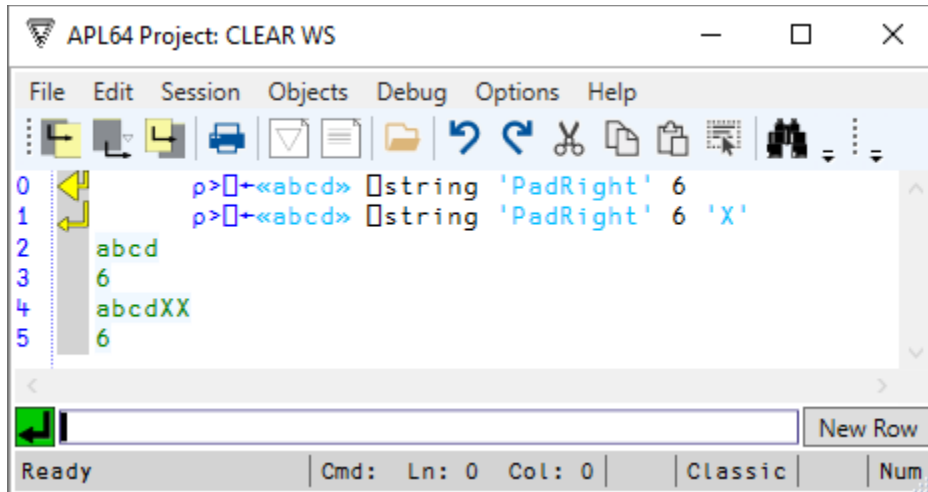


PADRIGHT

String←String □STRING "PADRIGHT" TotalWidth

String←String □STRING "PADRIGHT" TotalWidth PadChar

```
ρ>⊞←«abcd» ⊞string 'Padright' 6
ρ>⊞←«abcd» ⊞string 'Padright' 6 'X'
```



The screenshot shows the APL64 Project: CLEAR WS window. The code editor contains two lines of APL code. The first line is `ρ>⊞←«abcd» ⊞string 'Padright' 6` and the second line is `ρ>⊞←«abcd» ⊞string 'Padright' 6 'X'`. The results are displayed in the console area below the code editor. The first result is `abcd` followed by 6 spaces, and the second result is `abcdXX` followed by 6 spaces. The status bar at the bottom shows 'Ready', 'Cmd: Ln: 0 Col: 0', 'Classic', and 'Num'.

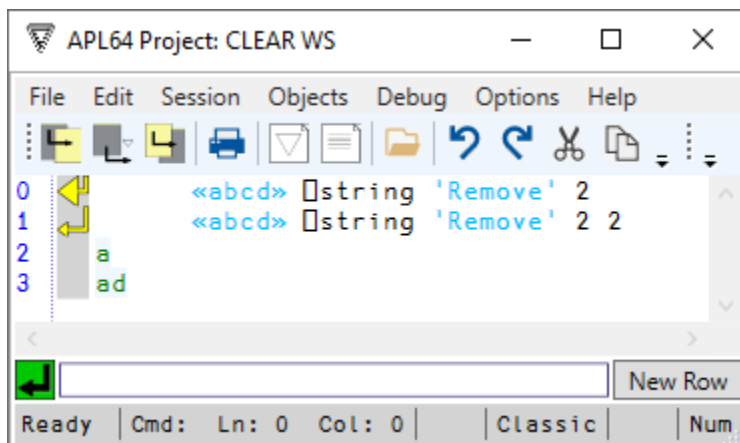
REMOVE

String←String □STRING "REMOVE" StartPos

String←String □STRING "REMOVE" StartPos #CharsToRemove

StartPos is □IO-sensitive.

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



The screenshot shows the APL64 Project: CLEAR WS window. The code editor contains two lines of APL code. The first line is `«abcd» ⊞string 'Remove' 2` and the second line is `«abcd» ⊞string 'Remove' 2 2`. The results are displayed in the console area below the code editor. The first result is `a` followed by 3 spaces, and the second result is `ad` followed by 2 spaces. The status bar at the bottom shows 'Ready', 'Cmd: Ln: 0 Col: 0', 'Classic', and 'Num'.

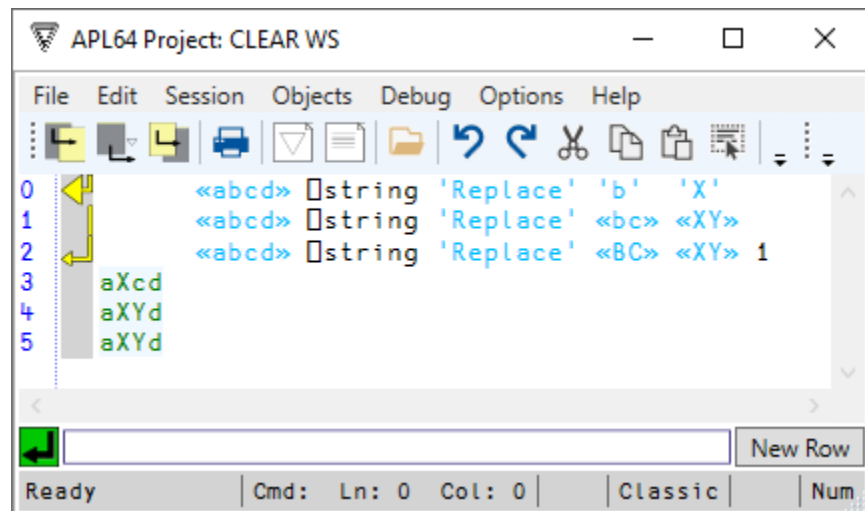
REPLACE

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» «Σ»)	□String	'SortIndex'	'A'
--	---------	-------------	-----

```

0 (⊂,⊂ ⊂.⊂ ⊂1⊂ ⊂2⊂ ⊂A⊂ ⊂B⊂)⊂String 'SortIndex' 'A'
1 1 2 3 4 5 6 7 8 9
2

```

(⊂,⊂ ⊂.⊂ ⊂1⊂ ⊂2⊂ ⊂A⊂ ⊂B⊂)⊂String 'SortIndex' 'D'

```

0 (⊂,⊂ ⊂.⊂ ⊂1⊂ ⊂2⊂ ⊂A⊂ ⊂B⊂)⊂String 'SortIndex' 'D'
1 1 6 5 4 3 2 1
2

```

(⊂,⊂ ⊂.⊂ ⊂1⊂ ⊂2⊂ ⊂A⊂ ⊂B⊂)⊂String 'SortIndex' 'A' ',.21'

```

0 (⊂,⊂ ⊂.⊂ ⊂1⊂ ⊂2⊂ ⊂A⊂ ⊂B⊂)⊂String 'SortIndex' 'A' ',.21'
1 1 1 2 2 3 3 4 4 5 5 6 6
2

```

(⊂,⊂ ⊂.⊂ ⊂1⊂ ⊂2⊂ ⊂A⊂ ⊂B⊂)⊂String 'SortIndex' 'D' ',.21'

```

0 (⊂,⊂ ⊂.⊂ ⊂1⊂ ⊂2⊂ ⊂A⊂ ⊂B⊂)⊂String 'SortIndex' 'D' ',.21'
1 1 5 6 3 4 2 1
2

```

X←2 3 4p⊂ZZ⊂ ⊂AB⊂ ⊂AB⊂ ⊂AB⊂ ⊂AB⊂ ⊂AB⊂ ⊂AB⊂ ⊂AB⊂ ⊂AB⊂ ⊂AB⊂ ⊂AB⊂ ⊂W⊂

```

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

```

```

APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0 X←2 3 4p«ZZZ» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «W»
1 X[2;3;4]←«T»
2 X
3 X⊞String 'SortIndex' 'A'
4 ZZZ AB AB AB
5 AB AB AB AB
6 AB AB AB W
7
8 ZZZ AB AB AB
9 AB AB AB AB
10 AB AB AB T
11 2 1
12
Ready | Hist: Ln: 12 Col: 6 | Ins | Classic | Num | EN_US

```

```

X←2 3 4p«ZZZ» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «W»
X[2;3;4]←«T»
X
X⊞String 'SortIndex' 'D'

```

```

APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0 X←2 3 4p«ZZZ» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «W»
1 X[2;3;4]←«T»
2 X
3 X⊞String 'SortIndex' 'D'
4 ZZZ AB AB AB
5 AB AB AB AB
6 AB AB AB W
7
8 ZZZ AB AB AB
9 AB AB AB AB
10 AB AB AB T
11 1 2
12
Ready | Hist: Ln: 12 Col: 6 | Ins | Classic | Num | EN_US

```

```

X←2 3 4p«ZZZ» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «W»
X[2;3;4]←«T»
X
X⊞String 'SortIndex' 'A' 'TW'

```

```

APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0 X←2 3 4p«ZZZ» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «W»
1 X[2;3;4]←«T»
2 X
3 X⌵String 'SortIndex' 'A' 'TW'
4 ZZZ AB AB AB
5 AB AB AB AB
6 AB AB AB W
7
8 ZZZ AB AB AB
9 AB AB AB AB
10 AB AB AB T
11 2 1
12

```

```

X←2 3 4p«ZZZ» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «W»
X[2;3;4]←«T»
X
X⌵String 'SortIndex' 'A' 'WT'

```

```

APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0 X←2 3 4p«ZZZ» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «AB» «W»
1 X[2;3;4]←«T»
2 X
3 X⌵String 'SortIndex' 'A' 'WT'
4 ZZZ AB AB AB
5 AB AB AB AB
6 AB AB AB W
7
8 ZZZ AB AB AB
9 AB AB AB AB
10 AB AB AB T
11 1 2
12

```

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» «pqrXYXzXY») ≡ «abcdXpqrXYXzXY» □string 'Split' 'XY' 2 0
(«abcd» «pqrXXYX») ≡ «abcdXpqrXXYX» □string 'Split' «X» 2 0
(«abcd» «pqrXYXzXY») ≡ «abcdXpqrXYXzXY» □string 'Split' «XY» 2 0
(«abcd» «pqr» «Y» «») ≡ «abcdXpqrXXYX» □string 'Split' 'X' 10 0
(«abcd» «pqr» «Z» «») ≡ «abcdXpqrXYXzXY» □string 'Split' 'XY' 10 0
(«abcd» «pqr» «Y» «») ≡ «abcdXpqrXXYX» □string 'Split' «X» 10 0
(«abcd» «pqr» «Z» «») ≡ «abcdXpqrXYXzXY» □string 'Split' «XY» 10 0
(«abcd» «pqr» «Y») ≡ «abcdXpqrXXYX» □string 'Split' 'X' 10 1
(«abcd» «pqr» «Z») ≡ «abcdXpqrXYXzXY» □string 'Split' 'XY' 10 1
(«abcd» «pqr» «Y») ≡ «abcdXpqrXXYX» □string 'Split' «X» 10 1
(«abcd» «pqr» «Z») ≡ «abcdXpqrXYXzXY» □string 'Split' «XY» 10 1

```

STARTSWITH

```

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

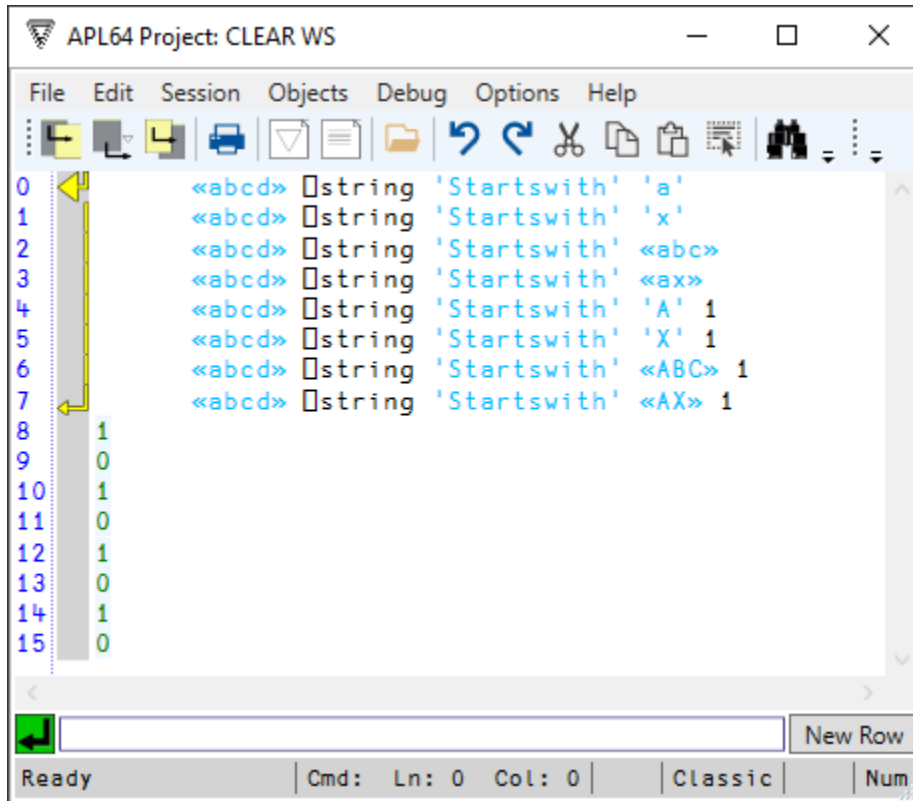
```

```

«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



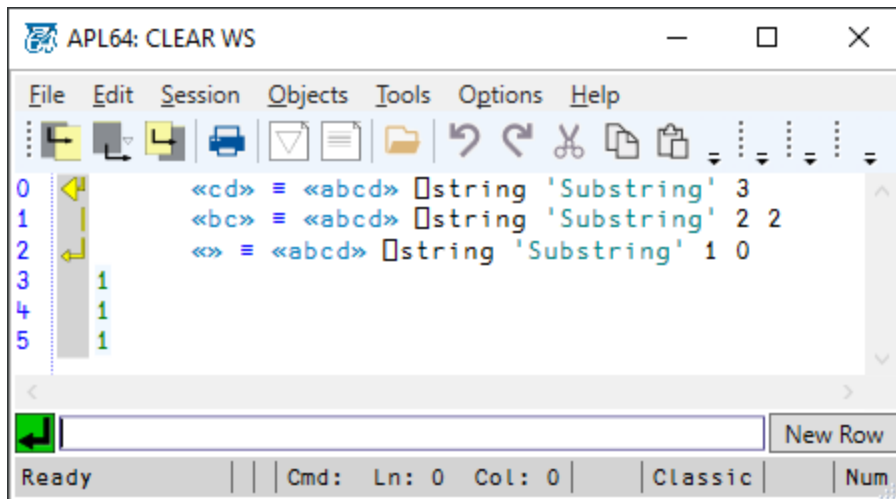
SUBSTRING

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

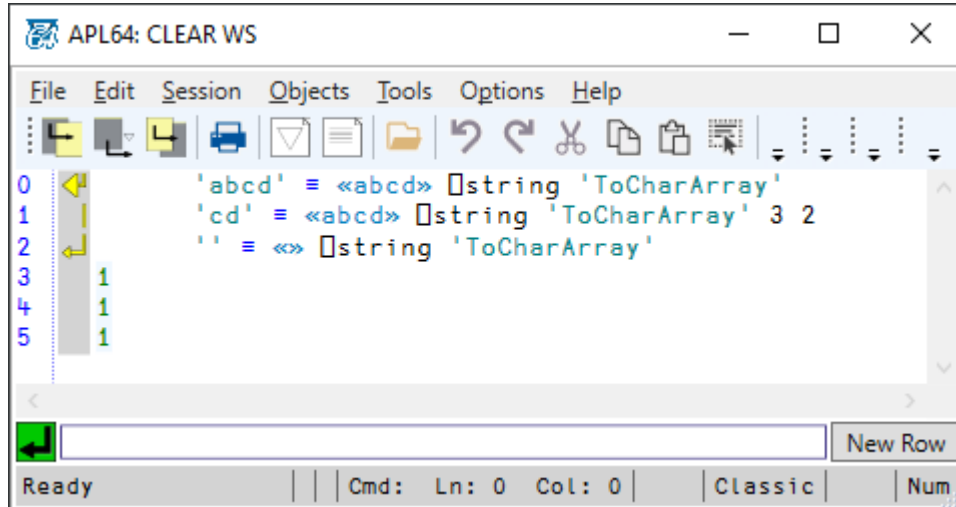


TOCHARARRAY

Char[]←String ⎕STRING "TOCHARARRAY"

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

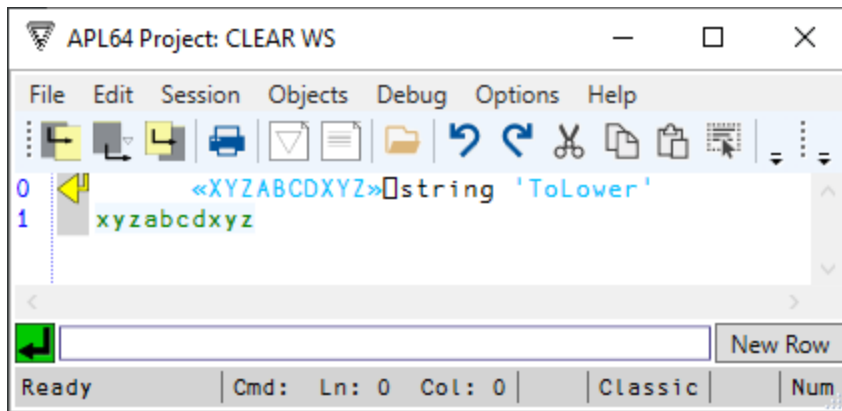
<pre> 'abcd' ≡ «abcd» ⎕string 'ToCharArray' 'cd' ≡ «abcd» ⎕string 'ToCharArray' 3 2 '' ≡ «>» ⎕string 'ToCharArray' </pre>
--



TOLOWER

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

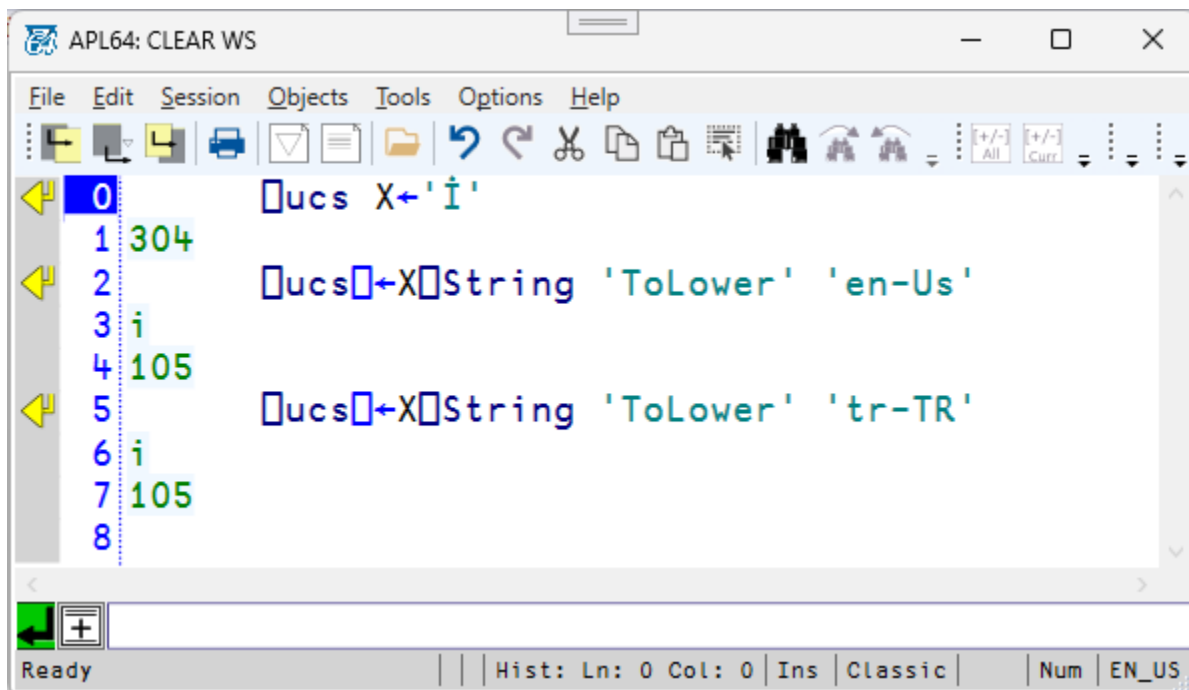
<pre> «XYZABCDXYZ» ⎕string 'ToLower' </pre>



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

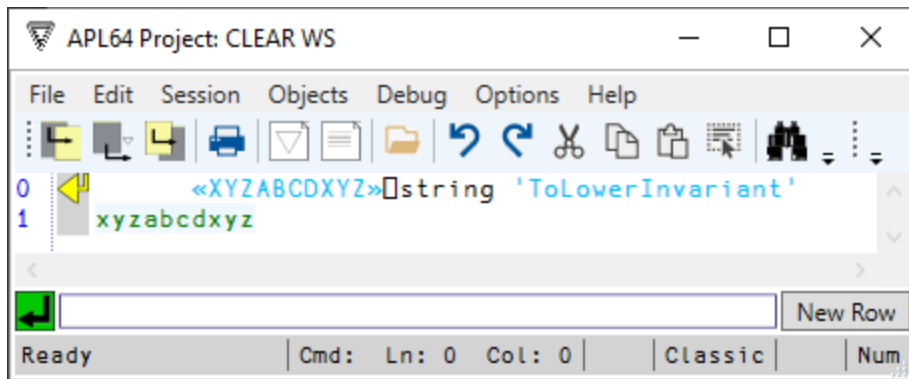
```
⊆ucs⊆←(⊆←«ABC123i»)⊆String 'ToLower' 'en-US'
```



TOLOWERINVARIANT

String←String ⊆STRING "TOLOWERINVARIANT"

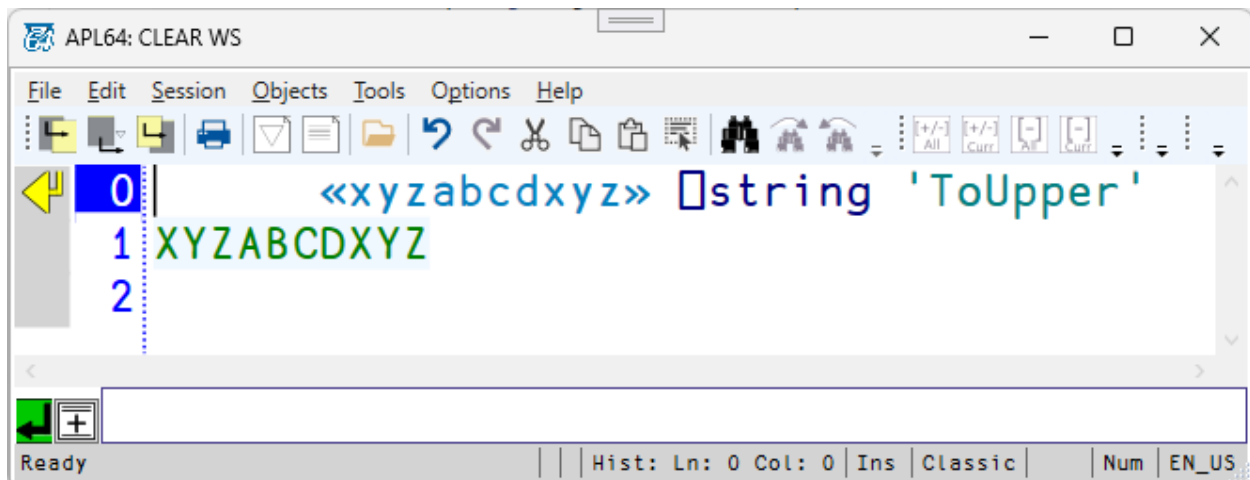
```
«XYZABCDXYZ» ⊆string 'ToLowerInvariant'
```



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'
```

```

0      ⌵ucs X←'i'
1      105
2      ⌵ucs⌵←X⌵String 'ToUpper' 'en-US'
3      I
4      73
5      ⌵ucs⌵←X⌵String 'ToUpper' 'tr-TR'
6      İ
7      304
8

```

Ready | Hist: Ln: 8 Col: 6 | Ins | Classic | Num | EN_US

TOUPPERINVARIANT

String←String ⌵STRING "TOUPPERINVARIANT"

```
«xyzabcdxyz» ⌵string 'ToUpperInvariant'
```

```

0      «xyzabcdxyz»⌵string 'ToUpperInvariant'
1      XYZABCDXYZ

```

Ready | Cmd: Ln: 0 Col: 0 | Classic | Num

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'

```

```

0  « ab cd »⊆string 'Trim'
1  «xabcxdx»⊆string 'Trim' 'x'
2  «xyzxyzabcdxyz»⊆string 'Trim' 'xyz'
3  ab cd
4  abcdx
5  abcd

```

Ready | Cmd: Ln: 0 Col: 0 | Classic | Num

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'

```

```

0  « ab cd »⊆string 'TrimEnd'
1  «xabcxdx»⊆string 'TrimEnd' 'x'
2  «xyzxyzabcdxyzxyz»⊆string 'TrimEnd' 'xyz'
3  ab cd
4  xabcxd
5  xyzxyzabcd

```

Ready | Cmd: Ln: 0 Col: 0 | Classic | Num

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'

```

0  « ab cd »␣string 'TrimStart'
1  «xabcxdx»␣string 'TrimStart' 'x'
2  «xyzxyzabcdxyzxyz»␣string 'TrimStart' 'xyz'
3
4  ab cd
5  abcdxdx
6  abcdxyzxyz

```

The screenshot shows the APL64: CLEAR WS window. The code editor contains three lines of APL code. The first line is « ab cd »␣string 'TrimStart'. The second line is «xabcxdx»␣string 'TrimStart' 'x'. The third line is «xyzxyzabcdxyzxyz»␣string 'TrimStart' 'xyz'. Below the code, the execution results are displayed: ab cd, abcdxdx, and abcdxyzxyz. The status bar at the bottom shows 'Ready', 'Cmd: Ln: 0 Col: 0', 'Classic', and 'Num'.

? (Documentation Summary)

Char[] ← String ␣STRING "?"

␣dr␣String '?'
 ρ␣String '?'
 1500↑␣String '?'

```

0  ␣string '?'
1  ␣STRING Documentation Summary
2
3  Int32 + String ␣STRING 'COMPARE' String
4  Int32 + String ␣STRING 'COMPARE' String IgnoreCase
5  Int32 + String ␣STRING 'COMPARE' StartPosInLarg String StartPosInCompareString #CharsToCompare
6  Int32 + String ␣STRING 'COMPARE' StartPosInLarg String StartPosInCompareString #CharsToCompare IgnoreCase
7
8  Int32 + String ␣STRING 'COMPAREORDINAL' String
9  Int32 + String ␣STRING 'COMPAREORDINAL' StartPosInLarg String StartPosInCompareString #CharsToCompare
10
11 String + String ␣STRING 'CONCAT' String
12 String + String ␣STRING 'CONCAT' String String
13 String + String ␣STRING 'CONCAT' String String String
14 String + String ␣STRING 'CONCAT' String[]
15
16 Bool + String ␣STRING 'CONTAINS' String
17 Bool + String ␣STRING 'CONTAINS' Char
18
19 Bool + String ␣STRING 'ENDSWITH' Char
20 Bool + String ␣STRING 'ENDSWITH' String
21 Bool + String ␣STRING 'ENDSWITH' String IgnoreCase
22
23 Bool + String ␣STRING 'EQUALS' String
24
25 AplArray + String ␣STRING 'GETCULTUREINFO'
26
27 Int32 + String ␣STRING 'GETHASHCODE'
28
29 Char[] + String ␣STRING 'HELP'

```

The screenshot shows the APL64: CLEAR WS window with the documentation for the STRING function. The code editor contains a list of functions and their signatures. The status bar at the bottom shows 'Ready', 'Hist: Ln: 136 Col: 6', 'Ins', 'Classic', 'Num', and 'EN_US'.