Harbour stringovi

String je proizvoljan niz simbola. Piše kao niz simbola izmedju navodnika ili apostrofa:

```
String := "Ovo je string"
specSym := '!#%"&'
```

FORMATIRANJE

```
Tc(), Tf(), Tk() -> Formatiranje brojeva po definiciji formata
Str(n, l, d)     -> Konvertuje broj u string dužine L i sa D decimala  Str(123, 10, 2)
Val(s)           -> Konvertuje string u numeri?ku vrednost             Val("123")
Left(s, l)       -> Prvih L simbola stringa S                          Left("Abeceda", 3)
Right(s, l)      -> Zadnjih L simbola stringa S                        Right("Abeceda", 3)
SubStr(s,p,l)    -> Deo stringa S od pozicije P, ducine L              SubStr("Abeceda", 3,2)
LTrim(s)         -> String S bez vode?ih blanko simbola                LTrim("   123  ")
RTrim(s)         -> String S bez blanko simbola na kraju               RTrim("   123  ")
AllTrim (s)      -> String S bez vode?ih i krajnjih blanko simbola     AllTrim("   123  ")
Trim(s)          -> String S bez blanko simbola na kraju               Trim("   123  ")
PadL(s, l)       -> Popunjava string S sa L blanko znakova sleva       PadL("ABC", 10)
PadR(s, l)       -> Popunjava string S sa L blanko znakova sdesna      PadR("ABC", 10)
PadC(s, l)       -> Popunjava string S sa L blanko znakova centrirano  PadC("ABC", 10)
Pad(s, l, c)     -> Popunjava string S sa L znakova C sdesna           Pad("ABC", 10, "*")
At (s1, s2)      -> Pozicija stringa S1 u stringu S2                   At(";", "Abce;da")
Len (s)          -> Dužina (broj simbola) stringa S                    Len("Abeceda")
Replicate(c, n)  -> Kreira string od N znakova C                       Replicate(“*”, 5)
Space(n)         -> Kreira string od N blanko znakova                  Space(5)
StrTran(s, x, y) -> Menja u stringu S sva pojavljivanja X u Y          StrTran("Abeceda", "e",
GetV(@cString, cDelimiter) -> U?itavanje stringa do delimitera. Pri tom se menja i origi
c := "Ovo je primer"
x := GetV (@c, " ")  // "Ovo"
? c                  // "je primer"
```

Za stringove su najvažniji operatori spajanja (konkatenacije). Operator + oznacava normalno spajanje, a – spajanje bez blanko simbola na kraju prvog operanda. Na primer:

```
x := "ABC "
y := "DE"
s1 := x + y    // s1 == "ABC DE"
s2 := x - y    // s2 == "ABCDE"
```

Za stringove je vezan jedan važan relacioni operator: $ (podstring). Pomoću njega se ispituje da li je string sadrzan u nekom drugom:

```
"ABC" $ "ABCDEF"   --> TACNO
"aBC" $ "ABCD"     --> NETACNO
"%"   $ "x % y"    --> TACNO
```

Veza izmedju celih brojeva i stringova je uobicajena funkcija Chr() konvertuje dati ceo broj u odgovarajući simbol po ASCII tabeli, a Asc() konvertuje prvi simbol u stringu u kod poASCII tabeli

TRANSFORM

Postoji joŠ jedna veoma vaŽna funkcija koja omoguĆava formatiranje podataka proizvoljnog tipa: funkcija Transform().

Ova funkcija podatak iz bilo kog tipa pretvara u string, sliČno kao Što funkcija Str() pretvara brojeve u stringove. Medjutim, Transform ima daleko veće mogućnosti. Prvo, prihvata i ostale tipove podataka, a ne samo brojeve; drugo, omogućava daleko precizniju kontrolu onoga žto se dobija kao rezultat.

Transform funkcija se kontroliše na dva načina: preko stringova za formatiranje i funkcija za formatiranje. Stringovi za formatiranje služe za operacije nad simbolima ulazne vrednosti. String za formatiranje je, u stvari, niz simbola od kojih neki imaju specijalno značenje. Na primer, simbol “9” u format stringu označava cifru. Ulaz se “uparuje” sa format stringom i na osnovu toga se pravi izlaz. Na primer:

```
? Transform (12345, "999,999")   -> "12,345"
```

Harbour podržava još specijalnih simbola:

```
A,N,X,9,# -    Ozna?ava jedan simbol ulaznog podatka. Rezultat zavisi od tipa ulazne vrednosti
L         -    Prikazuje logi?ki podatak kao T ili F
Y         -    Prikazuje logi?ki podatak kao Y ili N
!         -    Konvertuje slovo u odgovaraju?e veliko
$         -    Prikazuje $ umesto vode?ih nula u broju
*         -    Prikazuje * umesto vode?ih nula u broju
.         -    Odre?uje decimalnu ta?ku
,         -    Zarez
```

Evo nekih primera upotrebe prethodnih specijalnih simbola:

```
Transform(1234.56, "999,999.-")  ->  12,34.-
Transform(1234, "$$$$$$")        ->   $$1234
Transform(1234, "***/***")       ->  **1/234
Transform(-1234, "######")       ->    -1234
Transform("abcdef", "A=A=A")     ->    a=c=e
```

Kao što se vidi u poslednjem primeru, kada se formatira string, simbol u formatu koji nije specijalan jednostavno “prekrije” odgovarajući ulazni simbol.

Drugi način formatiranja je preko funkcija za formatiranje. Funkcije za formatiranje deluju na celu ulaznu vrednost, a ne pojedinačno na simbole. Zadaje se takođe u okviru stringa za formatiranje, samo što se pre kodova funkcija nalazi simbol @.

Na primer, da bi se sva slova u ulaznom stringu konvertovala u velika, zadaje se sledeća naredba:

```
? Transform("abcdef", "@!")
```

Funkcija Transform raspoznaje sledeće kodove – oznake funkcija za formatiranje:

```
B         -    Broj poravnava u levo
C         -    Prikazuje CR posle pozitivnog broja
D         -    Prikazuje datum u teku?em formatu
E         -    Prikazuje datum u evropskom formatu; kod brojeva zamenjuje ta?ku i zarez kao de
R         -    Simboli bez zna?enja u stringu za formatiranje ne brišu ulazne, ve? se ubacuju
X         -    Prikazuje DB posle negativnog broja
Z         -    Prikazuje nule kao blanko simbole
(         -    Zatvara negativne brojeve u zagrade
)         -    Zatvara negativne brojeve u zagrade bez vode?ih blanko simbola
!         -    Sva slova konvertuje u velika
```

Posle oznake za format funkciju (@), mogu se odjednom navesti i više kodova za funkcije. Na primer, string ”@BC” označava da broj treba poravnati u levo i da na kraju treba ispisati CR ako je broj pozitivan. Ponekad je potrebno kombinovati funkcije i simbole za formatiranje i tada treba prvo pisati funkcije, a onda simbole, posle jednog blanko simbola. Na primer

```
Transform (cResult, "@! AA-AAA/AAA")
```