Matematičke funkcije u CLIPS-u

- **+** vraća sumu svojih argumenata.
- **-** vraća razliku prvog argumenta i sume svih argumenata koji slede.
- ***** vraća proizvod svojih argumenata.
- **/** vraća količnik prvog argumenta i proizvoda svih narednih argumenta.

- `sqrt` vraća kvadratni koren argumenta.
- `div` vraća celobrojni količnik prvog argumenta i proizvoda svih narednih argumenta.
- `max` vraća svoj maksimalni argument.
- `min` vraća svoj minimalni argument.
- `abs` vraća apsolutnu vrednost svog argumenta.
- `mod` vraća ostatak pri deljenju prvog argumenta drugim argumentom.
- `float` konvertuje ceo broj u realan i vraća njegovu vrednost.
- `integer` konvertuje realan broj u ceo i vraća njegovu vrednost.

Uobičajeni zapis	CLIPS zapis
2+3+7	(+ 2 3 7)
5^2	(** 5 2)
5+12/2	(+ 5 (/ 12 2))
(23-2)/(10-3)>0	(> (/ (- 23 2) (- 10 3)) 0)

```
CLIPS> (defrule sabiranje
       (brojevi ?x ?y)
        =>
       (assert (zbir_je (+ ?x ?y)))
       )
```

```
CLIPS> (assert (brojevi 23 56))
==> f-5     (brojevi 23 56)
==> Activation 0    sabiranje: f-5
<Fact-5>
CLIPS> (run)
FIRE    1 sabiranje: f-5
==> f-6     (zbir_je 79)
```

Funkcija **`bind`**

Korišćenjem funkcije **`bind`** na desnoj strani pravila može se promenljivoj dodeliti vrednost nekog izraza.

PRIMER:

```
(defrule pitagora
(katete ?x ?y)
=>
(bind ?z (sqrt (+ (** ?x 2) (** ?y 2))))
(printout t "Hipotenuza je " ?z crlf))
```

PRIMER:

```
CLIPS> (assert (katete 4 3))
==> f-0     (katete 4 3)
==> Activation 0   pitagora:
  f-0
<Fact-0>

CLIPS> (run)
FIRE    1 pitagora: f-0
Hipotenuza je 5.0
```

Zadatak 8

Ako je dat sledeći deftemplate

```
(deftemplate pravougaonik
  (slot oznaka)
  (slot str-a)
  (slot str-b))
```

Napisati pravilo koje će izračunavati površinu figure.

READ

- CLIPS može čitati informacije koje korisnik ukucava sa tastature korišćenjem **`read`** funkcije.

- **`read`** prestaje sa čitanjem kada naiđe na delimiter.

Delimiteri u CLIPSu su:

- svaki ASCII karakter koji se ne može otprintati (to uključuje space, tab, carriage return, line feed)
- znaci navoda “ ”
- zagrade ( )
- ampersand &
- vertikalna linija |
- manje <
- tilda ~
- tačka zarez ;

- **`read`** funkcija, dakle, čita samo jedno polje. Ako pokušate da **`read`** funkcijom pročitate

  `Osnovna boja je crvena`

  biće pročitamo samo prvo polje O**snovna**. Da bi čitav unos bio pročitan neophodno je da bude između navodnika. Sve što se unese između navodnika biće tretirano kao string.

PRIMER:

```
CLIPS> (read)
Crvena
Crvena

CLIPS> (read)
Osnovna boja je crvena
Osnovna

CLIPS> (read)
"Osnovna boja je crvena"
"Osnovna boja je crvena"
```

READLINE

- **`readline`** funkcija je slična **`read`** funkciji, ali ona dozvoljava da ulaz bude čitav string umesto samo jednog polja.

- **`readline`** funkcija prestaje sa čitanjem samo kada naiđe na carriage return, tačku zarez ili EOF (u slučaju kada čita iz fajla).

- **`readline`** funkcija vraća string.

PRIMER:

```
CLIPS> (readline)
Osnovna boja je crvena
"Osnovna boja je crvena"
```

Šta dalje sa stringom?

- String se može transformisati multifield vrednost funkcijom **`explode$`**.

  Sintaksa je:

```
(explode$ <string-expression>)
```

```
CLIPS>
(defrule ucitaj-ime-korisnika
=>
(printout t “Unesite vase ime i prezime: “)
(bind ?odgovor (explode$ (readline)))
(assert (ime-korisnika-je ?odgovor)))

CLIPS>(reset)
CLIPS> (run)
Unesite ime i prezime: Visnja Simic
CLIPS>(facts)
f-0     (initial-fact)
f-1     (ime-korisnika-je Visnja Simic)
For a total of 2 facts.
```

Ostale string funkcije

- String se može transformisati u činjenicu sa više polja i potom ubaciti u listu činjenica funkcijom **`assert-string`**. Sintaksa je:

```
(assert-string <string-expression>)
```

PRIMER:

```
CLIPS> (deftemplate primer (slot x) (slot y))
CLIPS> (assert-string "(osnovna boja je crvena)")
CLIPS> (assert-string "(svetlo \"zuto\")")
CLIPS> (assert-string "(a\\b \"c\\\\d\")")
CLIPS> (assert-string "(primer (x 3))")
CLIPS> (assert-string "(primer (x 5)(y 7))")
CLIPS> (facts)
f-0 (osnovna boja je crvena)
f-1 (svetlo "zuto")
f-2 (a\b "c\d")
f-3 (primer (x 3) (y nil))
f-4 (primer (x 5) (y 7))
For a total of 5 facts.
```

Ostale string funkcije

- **`str-cat`** funkcija spaja svoje argumente u jedan string. Sintaksa:

  `(str-cat <izraz>*)`

  Svaki **`<izraz>`** treba da bude tipa: symbol, string, float ili integer.

```
CLIPS> (str-cat "bla" - bla)
"bla-bla"
```

- **sub-string** funkcija vraća string koji počinje od pozicije **<broj1>** i završava se na poziciji **<broj2>** u stringu **<string1>**. Sintaksa

  (sub-string <broj1> <broj2> <string1>)

  Ako je prvi argument veći od drugog vraća se prazan string **""**.

```
CLIPS> (sub-string 3 8 "abcdefghijkl")
"cdefgh"
```

- **str-index** funkcija vraća početnu poziciju jednog stringa **<string1>** u drugom stringu **<string2>**

(str-index <string1> <string2>)

kada se prvi string ne sadrži u drugom stringu funkcija vraća FALSE.

```
CLIPS> (str-index "def" "abcdefgh")
4
```

- **str-length** funkcija vraća dužinu stringa ili simbola u obliku celog broja. Sintaksa:

```
(str-length <string-ili-symbol>)
  CLIPS> (str-length "abcd")
  4
  CLIPS> (str-length x@z)
  3
```

- **`str-compare`** funkcija poredi dva stringa ili simbola i vraća 0 ako su jednaki, -1 ako prvi string prethodi drugom stringu po ASCII redu, a 1 ako drugi string prethodi prvom stringu. Poređenje se vrši karakter po karakter dok se ne dođe do kraja stringova ili do različitih karaktera.

  Sintaksa:

```
(str-compare <string-ili-symbol1>
             <string-ili-symbol2>)
```

PRIMER:

```
CLIPS> (str-compare "abcd" "abcd")
0
CLIPS> (str-compare "b" "aa")
1
CLIPS> (str-compare "a" "b")
-1
```

PRIMER 1:

```
CLIPS> (defrule test-readline
(initial-fact)
=>
(printout t "Unesi string" crlf)
(bind ?string (readline))
(assert-string
  (str-cat "(" ?string ")")
)
)
```

PRIMER 1:

```
CLIPS> (reset)
CLIPS> (run)
Unesi string
Osnovna boja je crvena

CLIPS> (facts)
f-0 (initial-fact)
f-1 (Osnovna boja je crvena)
For a total of 2 facts.
```