OPERACJE NA ŁAŃCUCHACH, FUNKCJA SWITCH

1. Operacje na łańcuchach

Punkt ten zawiera wykaz wszystkich operacji na łańcuchach, które są dostępne w CLIPS-ie.

(str-cat ...) – wiąże w jeden łańcuch pojedyncze elementy

```
np. (str-cat To jest lancuch)
”Tojestlancuch”

(str-cat To „ jest „ lancuch)
”To jest lancuch”
```

(str-index ...) – zwraca wartość mówiącą na której pozycji łańcucha znajduje się pierwszy znak podanego podłańcucha, np.:

```
(str-index ”abc” ”abcdef”)
1 – pierwszy znak podłańcucha ”abc” jest na pierwszej pozycji łańcucha ”abcdef”

(str-index ”cd” ”abcdef”)
3 – pierwszy znak podłańcucha ”cd” jest na trzeciej pozycji łańcucha ”abcdef”

(str-index ”xyz” ”abcdef”)
FALSE – podłańcuch ”xyz” nie wystepuje w łańcuchu ”abcdef”
```

(sub-string ...) – zwraca podłańcuch z podanego łańcucha, np.:

```
(sub-string 2 4 „abcdef”)
”bcd” – podłańcuch łańcucha ”abcdef” od indeksu 2 do indeksu 4

(sub-string 4 10 ”abcdef”)
”def” – funkcja zwraca tylko tyle znaków, ile może

(sub-string 7 5 „abcdef”)
” ” – funkcja zwraca pusty łańcuch, ponieważ indeks początkowy jest większy od
indeksu końcowego
```

(str-compare ...) – porównuje dwa łańcuchy i zwraca kod zależny od wyniku porównania

```
(str-compare ”cios” ”drzewo”)
-1; kod = -1, ponieważ pierwszy łańcuch < drugiego łańcucha (”c” jest wcześniej
niż ”d”)

(str-compare ”cios” ”bela”)
1; kod = 1, ponieważ pierwszy łańcuch > drugiego łańcucha (”c” jest później niż
”b”)

(str-compare ”papuga” ”papuga”)
0; kod=0, ponieważ pierwszy łańcuch = drugi łańcuch
```

(str-length ...) – zwraca długość łańcucha

```
(str-length ”Kowalski”)
8
```

(upcase …) – zamienia małe litery na wielkie, np:

```
(upcase ”hello”)
”HELLO”
```

(lowcase …) – zamienia wielkie litery na małe np.

```
(lowcase ”HELLO”)
”hello”
```

Przykład demonstrujący wykorzystanie operacji na łańcuchach:

```
(defrule start
(initial-fact)
=>
(printout t "Podaj 1 wyraz tekstu : ")
(bind ?tekst1 (read))
(printout t "Podaj 2 wyraz tekstu : ")
(bind ?tekst2 (read))
(printout t "Podaj 3 wyraz tekstu : ")
(bind ?tekst3 (read))
(assert (tekst1 ?tekst1)(tekst2 ?tekst2)(tekst3 ?tekst3)))

(defrule operacje-na-lancuchach
(tekst1 ?tekst1)(tekst2 ?tekst2)(tekst3 ?tekst3)
=>
(bind ?a1 (str-cat ?tekst1 ?tekst2 ?tekst3))
(printout t "Str-cat =" ?a1 crlf)
(printout t "Podaj podlancuch = ")
(bind ?podlancuch (read))
(bind ?a2 (str-index ?podlancuch ?a1))
(printout t "Podlancuch = " ?a2 crlf)
(printout t "Podaj start substring = ")
(bind ?start (read))
(printout t "Podaj stop substring = ")
(bind ?stop (read))
(bind ?a3 (sub-string ?start ?stop ?a1))
(printout t "Substring od " ?start " do " ?stop " = " ?a3 crlf)
(bind ?dlugosc (str-length ?a1))
(printout t "Dlugosc = " ?dlugosc crlf)
(bind ?up (upcase ?a1))
(printout t "Upcase = " ?up crlf)
(bind ?low (lowcase ?a1))
(printout t "Lowcase = " ?low crlf)
(printout t "Podaj 1 wyraz do porownania = ")
(bind ?w1 (read))
(printout t "Podaj 2 wyraz do porownania = ")
(bind ?w2 (read))
(bind ?por (str-compare ?w1 ?w2))
(printout t "Wynik porownania to = " ?por crlf)
)
```

2. Funkcja (switch ...)

Funkcja (switch ...) pozwala na selekcję i wykonanie grupy czynności, spośród wielu czynności, w zależności od zadanej wartości. Jej składnia jest nastepująca:

```
(switch <wyrażenie-testujące>
      (case <warunek-1> then <akcja-1>)
      .
      .
      .
      (case <warunek-i> then <akcja-i>)
      .
      .
      .
      (case <warunek-N> then <akcja-N>)
[default <akcje>])
```

Funkcja (switch ...) musi zawierać co najmniej dwa wyrażenia (case ...), aby mogła być dokonana selekcja, natomiast wyrażenie „default akcje" jest opcjonalne. Każdy z warunków występujących po słowie kluczowym „case" musi być oczywiście inny.

Funkcja (switch ...) oblicza najpierw wyrażenie testujące, a następnie odpowiedni i-ty warunek w kolejności zgodnej z definicją. Jeżeli i-ty warunek jest zgodny z wyrażeniem testującym, to jest wykonywana i-ta akcja. Jeżeli żaden z warunków nie jest spełniony, to wykonywane są akcje „default" – jeżeli są określone (zdefiniowane). Wartością zwracaną jest ostatnia akcja obliczona przez funkcje (switch ...). Jeżeli żadna z akcji nie została wykonana, to zwracana jest wartość FALSE.

Przykład na zastosowanie funkcji switch

```
(defrule start
      (initial-fact)
=>
      (printout t "Wybieram wariant :" crlf)
      (printout t "1-numer jeden" crlf)
      (printout t "2-numer dwa" crlf)
      (printout t "3-numer trzy" crlf)
      (bind ?x (read))
      (assert (x ?x)))

(defrule wybierz
      (x ?x)
=>
      (switch ?x
      (case 1 then (printout t "Wybrales wariant pierwszy" crlf))
      (case 2 then (printout t "Wybrales wariant drugi" crlf))
      (case 3 then (printout t "Wybrales wariant trzeci" crlf))
      (default (printout t "Nie wybrales wariantu z listy" crlf))))
```

Zadania do wykonania:

1. Napisać program umożliwiający wprowadzenie 3 dowolnych wyrazów. Po wprowadzeniu wyrazy należy połączyć, tak aby 1 wyraz znajdował się na końcu nowo powstałego ciągu, a wyraz 3 na początku. Po dokonaniu operacji połączenia wyrazów na nowo utworzonym ciągu znaków zastosować 4 dowolne funkcje służące do operacji na stringach. Program powinien informować użytkownika, która operację wykonuje.

2. Napisać program składający się z trzech funkcji obliczających kolejno:

- Pole
- Obwód
- Przekątną prostokąta

Program powinien pobrać od użytkownika wartości długości boków a i b, a następnie przy użyciu instrukcji switch, obliczyć żądaną wartość przy użyciu jednej z funkcji zadeklarowanych przez użytkownika.

Przykładowe działanie programu może być następujące:

Podaj bok a prostokata = 5
Podaj bok b prostokata = 6

Co chcesz obliczyć:
1-Pole Prostokąta
2-Obwód Prostokąta
3-Przekątną Prostokąta