ZMIENNE GLOBALNE, INSTRUKCJE PĘTLOWE

1. Zmienne globalne

Zmienne globalne definiujemy w następujący sposób:

```
(defglobal
     ?*<zmienna1>*  =  <wartosc1>
     . . .
     ?*<zmiennaN>* = <wartoscN>)
```

Zmienna zdefiniowana za pomocą konstrukcji (defglobal ...) jest rozpoznawana globalnie, przez wszystkie reguły.

Różnice między zmiennymi globalnymi a lokalnymi:

a) Zmienne globalne nie mogą być użyte tak samo jak zmienne lokalne po lewej stronie reguły, czyli niedopuszczalne jest użycie zmiennej globalnej w taki sposób jak w poniższej regule:

```
(defrule regula-niedopuszczalna
     (x ?*x*)
=>
...)
```

b) Zmienna globalna może być natomiast użyta po lewej stronie reguły jako część funkcji (test ...), np.:

```
(defrule regula-dopuszczalna
     (y ?y)
     (test (> ?y ?*x*))
=>
...)
```

Tak więc, jeśli ?*x* = 5 oraz mamy fakt (y 6), to reguła zostanie uaktywniona.

c) Jeżeli wartość zmiennej globalnej zostanie zmieniona podczas wykonywania programu, to reguły, w których jest ona użyta po lewej stronie, „dostosują się" do zmian. Np. jeśli zmienimy wartość ?*x* na 10, powyższa reguła zostanie uaktywniona przez fakt (y 6), choć poprzednio byłaby przez taki fakt uaktywniona.

Zmienne globalne mogą być wstawiane przy pomocy komendy (bind ...) po prawej stronie reguły, lub bezpośrednio z poziomu CLIPS'a. Zmienne globalne mogą być usunięte za pomocą komendy (clear). Po podaniu polecenia (clear) należy w programie zmodyfikować wyrażenie (defglobal ...) tak, aby nie zawierało niepożądanych zmiennych i tak zmieniony program załadować do CLIPS'a. Inny sposób usunięcia zmiennej globalnej to polecenie:

```
(undefglobal <nazwa-zmiennej-globalnej>)
```

Jeżeli podczas wykonywania programu zmienimy wartość zmiennej globalnej, to po podaniu komendy (reset) zostanie przywrócona jej pierwotna wartość (zdefiniowana w programie).

Przykład:

```
(defglobal
     ?*gx* = 5
     ?*gy* = 10
     ?*gz* = 50)

(defrule start
     (initial-fact)
=>
     (printout t "Podaj wartosc x = ")
     (bind ?x (read))
     (printout t "Podaj wartosc y = ")
     (bind ?y (read))
     (printout t "Podaj wartosc z = ")
     (bind ?z (read))
     (assert (x ?x)(y ?y)(z ?z)))

(defrule regula1
     (x ?x)(y ?y)(z ?z)
     (test (> ?x ?*gx*))
=>
     (printout t "Zmienna lokalna x = " ?x " jest wieksza od zmiennej globalnej
     gx = " ?*gx* crlf))

(defrule regula2
     (x ?x)(y ?y)(z ?z)
     (test (= ?*gy* 10))
=>
     (printout t "Zmienna globalna gy jest rowna 10" crlf)
     (printout t "Zmieniamy wartosc zmiennej globalnej gy oraz gz = " ?*gz* crlf)
     (bind ?*gy* 100)
     (printout t "Podaj nowa wartosc zmiennej globalnej gz = ")
     (bind ?*gz* (read))
     (bind ?tmp 1)
     (assert (tmp ?tmp)))

(defrule regula3
     (x ?x)(y ?y)(z ?z)(tmp ?tmp)
=>
     (printout t "Nowa wartosc gy wynosi = " ?*gy* crlf)
     (printout t "Nowa wartosc gz wynosi = " ?*gz* crlf))
```

2. Funkcja while

Składnia pętli while jest nastepująca:

```
(while  (<warunek>)
     (<akcja1>)
     ...
     (<akcjaN>))
```

Przykład:

```
(defrule start
     (initial-fact)
=>
     (printout t "Program oblicza wartosc silni z n" crlf)
     (printout t "Podaj n = ")
     (bind ?n (read))
     (assert (n ?n)))

(defrule licz
     (n ?n)
=>
     (bind ?silnia 1)
     (bind ?licznik 1)
     (while (<= ?licznik ?n)
     (bind ?silnia (* ?silnia ?licznik))
     (bind ?licznik (+ ?licznik 1)))
     (assert (silnia ?silnia)))

(defrule wypisz
     (silnia ?silnia)
=>
     (printout t "Silnia = " ?silnia crlf))
```

3. Zadania

a) napisać program wypisujący kolejno liczby od 1 do zadanej wartości *n*
np. po podaniu n=7, program powinien wypisać:

1
2
3
4
5
6
7

b) napisać program obliczający sumę kwadratów dla podanej wartości n
np. po podaniu n=5, program powinien obliczyć suma=$1^2+2^2+3^2+4^2+5^2=55$

c) napisać program obliczający n-ty wyraz ciągu Fibonacciego

$F_0 = 1;\ F_1 = 2;\ F_2 = 3;\ F_3 = 5;\ \ldots$
$F_n = F_{n-1} + F_{n-2}$

np. po podaniu n=6, program powinien podać liczbę 21

d) zmodyfikować program z punktu 3c, aby program korzystał z funkcji (fibo ?n) utworzonej przez użytkownika, która to funkcja obliczałaby n-ty wyraz ciągu.