PREDIKÁTY

- sú procedúry, ktoré vracajú ako svoju hodnotu symboly #t alebo #f.
- relačné predikáty: =, <, >
- logické spojky: or, and, not -unárny predikát

```
> ( < 1 2 )
#t
> ( > =  4  (+ 2 3) )
 #f
(and (=4 (* 2 2)) (or(> 0.5 (cos 0)) (< (sin 1) 0)))
#f
```

Užívatelský predikát: ***trojuholnik?***

```
(define (trojuholnik?  a b c)
   (and  (>  (+ a b)  c)
         (>  (+ b c)  a)
         (>  (+ a c)  b) ) )
```

Procedura zabudovaná v predikátoch: ***remainder*** (zvyšok vzniknutý po delení dvoch čísel) – určujú párnosť alebo nepárnosť daného čísla. Tretí predikát reprezentuje implementáciu XOR logickej spojky:

```
(define  (odd?  x)
   (=  (remainder  x 2)  1) )

(define  (even?  x)
   (not  (odd?  x) ) )

(define  (xor  x y)
   (or  (and  x  (not  y) )
        (and  y  (not  x) ) ) )
```

ŠPECIÁLNA FORMA if

akceptuje 3 parametre:

1. parameter je predikát
2. parameter je výraz, bude vtedy vyhodnotený, keď predikát je splnený
3. parameter je alternatívny výraz a je vyhodnotený vtedy, ak predikát nie je splnený.

Formálny tvar špeciálnej formy if:

(if <predikát> <výraz> <alt-výraz>)

Majme nasledujúce interakcie s j.Scheme (skúška)

```
(if   (> 1 2)   (+  3 4)   (-  2 5) )
(if   (even?  4)   (+  4 1)   (-  4 1) )
```

Výpočet absolútnej hodnoty čísla:

```
(define   (abs  x)
   (if   (>  x  0)   x
         (-   x) ) )

(define   (maximum  x  y)
   (if   (>   x  y)   x  y) )
```

V niektorých prípadoch môže špeciálna forma ***if*** akceptovať iba dva výrazy , a to predikát a výraz:
(if < predikát> <výraz>)

Pokiaľ predikát je splnený, vracia forma výsledok vyhodnotenia výrazu. Inak–výsledok nedefinovaný. Táto varianta dáva zmysel iba v prípade procedúr, ktoré pracujú na základe svojho vedľajšieho účinku, čo my nebudeme používať.

ŠPECIÁLNA FORMA cond

Umožňuje vetviť na viac alternatívnych vetví:

```
(cond   (<predikát1>  <výraz1>)
        (<predikát2>  <výraz2>)
         .....
        (else  <alt-výraz> ) )
```

Vyhodnotenie skončí pokiaľ nenarazíme na predikát vyhodnotený na #t.
Namiesto posledného predikátu je možné použiť kľúčové slovo *else* v tomto kontexte ktoré zastupuje predikát, ktorý je vždy splnený.

Príklad
Pre porovnanie - procedúra signum zapísaná prostredníctvom sekvencie príkazov *if* a pomocou príkazu *cond:*

```
(define   (signum  x)
   (if   (<  x  0)  -1
         (if   ( >   x  0)  1  0) ) )

(define   (signum  x)
   (cond   ( (<  x  0)  -1)
           ( ( >   x  0)  1)
           (else  0) ) )
```

Záver:
V programe môžeme používať logické hodnoty #t, #f.
Procedúry, ktoré vracajú logické hodnoty, voláme predikáty. Ich názvy spravidla končia otáznikom.
Pre vetvenie výpočtu na 2 vetvy používame šp.f. ***if.***
Pre vetvenie výpočtu na viac vetiev používame šp.f. ***cond***.

Príklady
1. Nájdite výraz, ktorý potvrdí, že *and* a *or* nie sú obyčajné procedúry.

riešenie
Keď voláme procedúru, vždy sa vyhodnotia všetky prvky zoznamu. Vyhodnotenie zoznamu vráti hodnotu #f:

```
(and   (> 1  2)   test)
```

Ak by *and* bola obyčajná procedúra, vtedy by vyhodnotenie muselo skončiť chybou, nakoľko *test* je nenaviazaný symbol. (analogicky je to pri *or)*

2 Dokážte že *if* nie je procedúra.

Analogicky postupujeme ako v predch. priklade.
Stačí vyhodnotiť napr. tento výraz:

(if (> 1 0) 1 pokus)

3. Napíšte predikáty: *kladne?*, *zaporne?*, *medzi?* , ktoré vrátia logickú hodnotu na základe toho, či je číslo kladné, záporné, resp. nachádza sa medzi dvoma číslami: Napríklad:

(kladne? -6)
vyhodnotenie: #f

(zaporne? -6)
vyhodnotenie: #t

(medzi? 4 2 5)
vyhodnotenie: #t

(medzi? 9 2 5)
vyhodnotenie: #f

4. Napíšte procedúru *maximum?* Pre nájdenie najväčšieho z troch čísiel (otestujte všetky vetvy výpočtu). Napríklad:

(maximum3 4 1 7)
vyhodnotí sa na 7

5. Napíšte procedúru *sucet2vacsich,* ktorá vráti súčet dvoch väčších čísiel a má tri argumenty.