Infiksna i prefiksna notacija

```
> 2+3*4
14

> (2+3)*4
20

> sqrt (3^2 + 4^2)
5.0
```

Standardna biblioteka Prelude.hs

Veliki broj funkcija, pregledati njihov spisak u Prelude.hs datoteci koja se nalazi na mestu gde je instaliran Haskell.

- Npr. Funkcija za odabir prvog elementa liste:

```
> head [1,2,3,4,5]
1
```

- Rep liste:

```
> tail [1,2,3,4,5]
[2,3,4,5]
```

- N-ti element liste:

```
> [1,2,3,4,5] !! 2
3
```

- Zadržavanje prvih N elemenata liste:

```
> take 3 [1,2,3,4,5]
[1,2,3]
```

Funkcije

U matematici	U Haskelu
`f(x)`	`f x`
`f(x,y)`	`f x y`
`f(g(x))`	`f (g x)`
`f(x,g(y))`	`f x (g y)`
`f(x)g(y)`	`f x * g y`

Skript datoteke

Otvoriti editor i ukucati Haskel kod. Sačuvati pod ekstenzijom .hs. Nakon toga se može učitati iz radnog Haskel okruženja.

```
double x    = x + x

quadruple x = double (double x)
```

Korisne komande Haskel okruženja, ne prog. jezika

Komanda	Značenje
:load *name*	učitava skript
:reload	osvežava učitani
:edit *name*	otvara skript za edit
:edit	isto samo aktivni
:type *expr*	prikazuje tip podatka proizvoljnog izraza
:?	prikazuje sve komande
:quit	izlazi iz okruženja

Osnovni tipovi podataka

`Bool`	- Logički
`Char`	- Karakter
`String`	- String
`Int, Integer`	- Ceo broj
`Float`	- Realan broj

Tip lista

```
[False,True,False] :: [Bool]

['a','b','c','d']  :: [Char]
```

```
[['a'],['b','c']] :: [[Char]]
```

N-torka (kao struktura)

```
(False,True)      :: (Bool,Bool)

(False,'a',True) :: (Bool,Char,Bool)
```

Funkcijski tip – potpis funkcije

```
not      :: Bool → Bool

isDigit :: Char → Bool
```

```
add          :: (Int,Int) → Int
add (x,y)     = x+y

zeroto       :: Int → [Int]
zeroto n      = [0..n]
```

Karijeve funkcije

Svojstvo da funkcija može vraćati drugu funkciju kao rezultat izvršavanja:

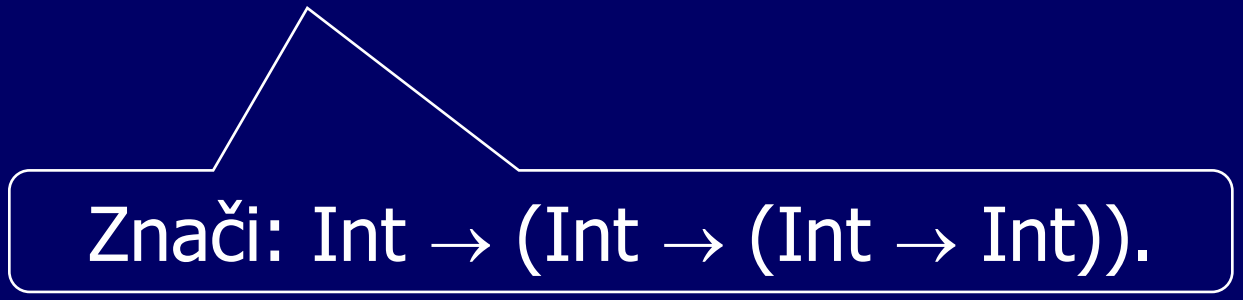
```
add'     :: Int → (Int → Int)
add' x y = x+y
```

add' prihvata 1 argument i vraća funkciju koja posle prihvata još jedan. Parcijalna primena.

Asocijativnost

- Strelica $\rightarrow$ je desno asocijativna.

```
Int → Int → Int → Int
```



Generički tipovi

```
length :: [a] → Int
```

Hoćemo da funkcija length radi za sve prebrojive tipove, a ne da pišemo za svaki tip novu.

- Mnoge funkcije iz standardne biblioteke su već generičke:

```
fst  :: (a,b) → a

head :: [a] → a

take :: Int → [a] → [a]

zip  :: [a] → [b] → [(a,b)]

id   :: a → a
```

Ograničenja na tipove

Nekad ne želimo da se generičke metode mogu primeniti baš na svim tipovima.

```
sum :: Num a ⇒ [a] → a
```

Ne bi trebalo dozvoliti sumiranje nenumeričkih tipova.

- Tri ugrađena ograničenja za tipove:

  `Num` - Numerički

  `Eq` - Poredbeni

  `Ord` - Uređeni

- Npr:

```
(+)  :: Num a ⇒ a → a → a
(==) :: Eq a  ⇒ a → a → Bool
(<)  :: Ord a ⇒ a → a → Bool
```

Vežbe

(1) Koji su tipovi sledećih izraza?

```
['a','b','c']

('a','b','c')

[(False,'0'),(True,'1')]

([False,True],['0','1'])

[tail,init,reverse]
```

(2)

```
second xs       = head (tail xs)

swap (x,y)      = (y,x)

pair x y        = (x,y)

double x        = x*2

palindrome xs = reverse xs == xs

twice f x       = f (f x)
```

(3) Proverite odgovore korišćenjem :type komande

Uslovni izraz if-then-else

```
abs  :: Int → Int
abs n = if n ≥ 0 then n else -n
```

Uslovni izraz sa više mogućnosti

```
abs n | n ≥ 0     = n
      | otherwise = -n
```

Interna reprezentacija liste

Interno je svaka lista predstavljena višestrukom upotrebom funkcije (:) "cons" koja dodaje jedan element na početak liste.

```
[1,2,3,4]
```

Means 1:(2:(3:(4:[]))).

Liste se dekomponuju na sledeći način.

```
head        :: [a] → a
head (x:_)   = x

tail        :: [a] → [a]
tail (_:xs) = xs
```

Generisanje skupova

Slično kao u matematici.

$\{x^2 \mid x \in \{1...5\}\}$

```
[x^2 | x ← [1..5]]
```

```
> [(x,y) | x ← [1,2,3], y ← [4,5]]

[(1,4),(1,5),(2,4),(2,5),(3,4),(3,5)]
```

Korišćenjem ove tehnike obrisati unutrašnje zagrade iz liste listi?

```
concat    :: [[a]] → [a]
concat xss = [x | xs ← xss, x ← xs]
```

```
> concat [[1,2,3],[4,5],[6]]

[1,2,3,4,5,6]
```

Postavljanje ograničenja na generator:

```
factors  :: Int → [Int]
factors n =
   [x | x ← [1..n], n `mod` x == 0]
```

Npr:

```
> factors 15

[1,3,5,15]
```

Korišćenjem ove tehnike izgenerisati spisak svih prostih brojeva do zadatog broja N?

```
primes  :: Int → [Int]
primes n = [x | x ← [2..n], prime x]
```

Npr:

```
> primes 40

[2,3,5,7,11,13,17,19,23,29,31,37]
```

Funkcija zip

```
zip :: [a] → [b] → [(a,b)]
```

Primer:

```
> zip ['a','b','c'] [1,2,3,4]

[('a',1),('b',2),('c',3)]
```

Primer formiranja uzastopnih parova elemenata date liste:

```
pairs    :: [a] → [(a,a)]
pairs xs = zip xs (tail xs)
```

Npr:

```
> pairs [1,2,3,4]

[(1,2),(2,3),(3,4)]
```

Vežbe

(1) Trojka (x,y,z) pozitivnih celih brojeva se zove Pitagorina ako važi $x^2 + y^2 = z^2$. Korišćenjem tehnike generisanja skupova napisati funkciju:

```
pyths :: Int → [(Int,Int,Int)]
```

Koja za dati pozitivan broj n, određuje sve Pitagorine trojke na skupu celih brojeva {1,…,n}:

```
> pyths 5
[(3,4,5),(4,3,5)]
```

(2) Pozitivan ceo broj je savršen ako je jednak sumi svojih faktora. Npr. Faktori broja 6 su 1, 2 i 3 pa je on savršen.

```
perfects :: Int → [Int]
```

Uraditi ovo korišćenjem tehnike generisanja skupova na zadatom intervalu vrednosti {1,…,n}:

```
> perfects 500

[6,28,496]
```

(3) Korišćenjem tehnike generisanja skupova napisati funkciju za računje skalarnog proizvoda dve liste celih brojeva:

$$\sum_{i=0}^{n-1} (xs_i * ys_i)$$

Algoritam Quicksort

Ideja:

- Prazna lista je sortirana lista;
- Ne prazna lista se može sortirati tako što se sortiraju svi elementi manji od glave i svi elementi veći ili jednaki od glave, a potom se rezultuja sortirana lista dobije nadovezivanjem prve liste, glave i druge liste.

Lambda izrazi

Anonimne (ad-hoc) funkcije.

$\lambda x \rightarrow x+x$

Anonimna funkcija koja prihvata x i vraća rezultat x+x.

Funkcije višeg reda

Opšta definicija: funcije višeg reda kao argumente prihvataju neke druge funkcije.

```
twice    :: (a → a) → a → a
twice f x = f (f x)
```

Funkcija mapiranja

Šta je zajedničko sledećim funkcijama: funkcija koja povećava sve elemente liste za 1, funkcija koja množi sve elemente liste sa 2 itd.

```
map :: (a → b) → [a] → [b]
```

Npr:

```
> map (+1) [1,3,5,7]

[2,4,6,8]
```

Jednostavnija implementacija pomoću tehnike generisanja skupova:

```
map f xs = [f x | x ← xs]
```

Ili varijanta putem rekurzije:

```
map f []     = []
map f (x:xs) = f x : map f xs
```

Filter funkcija

Šta je zajedničko funkcijama: eliminiši iz liste sve parne, sve neparne, sve racionalne brojeve itd.

```
filter :: (a → Bool) → [a] → [a]
```

Npr:

```
> filter even [1..10]

[2,4,6,8,10]
```

Definicija filter funkcije preko tehnike generisanja skupova:

```
filter p xs = [x | x ← xs, p x]
```

Ili varijanta preko rekurzije:

```
filter p []     = []
filter p (x:xs)
   | p x        = x : filter p xs
   | otherwise  = filter p xs
```

Foldr funkcija

Šta je zajedničko za funkciju koja sabira elemente liste, množi ih, nadovezuje listu listi u jednu listu itd:

```
f []     = v
f (x:xs) = x ⊕ f xs
```

```
sum []     = 0
sum (x:xs) = x + sum xs
```

v = 0
⊕ = +

```
product []     = 1
product (x:xs) = x * product xs
```

v = 1
⊕ = *

```
and []     = True
and (x:xs) = x && and xs
```

v = True
⊕ = &&

Definisati funkciju koja obrće listu korišćenjem foldr funkcije višeg reda.

Definisati preko foldr funkciju koja pronalazi minimalni element liste.