Najpoužívanejšie Funkcie v Haskeli

- **all::(a→Bool) →[a] →Bool**
  all even [2,4,6] = True
  all even [1,4,6] = False
- **and::[Bool] →Bool**
  and[False,True,True] = False
  -stačí jeden False
- **any::(a→Bool) →[a] →Bool**
  any even [1,3,4] = True
- **concat::[[a]] →[a]**
  concat [[6],[4]] = [6,4]
- **const::a→b→a**
  const x y = x
- **curry::((a,b) →c) →a→b→c**
  curry f x y = f (x,y)
- **cycle::[a] →[b]**
  cycle [1,2] = [1,2,1,2,...]
- **div::Integer→Integer→Integer**
  div 9 2 = 4
  -celočíselné delenie
- **drop::Int→[a] →[a]**
  drop 2 [3,8,9] = [9]
- **dropWhile::(a→Bool) →[a] →[a]**
  dropWhile odd [1,3,2,4] = [2,4]
- **elem::Eq a =>a→[a] →Bool**
  elem 1 [1,2,6] = True
  elem 1 [2,4,8] = False
- **even::Integer→Bool**
  -vracia True pre párne čísla
- **filter::(a→Bool) →[a] →[a]**
  filter odd [1,1,2] = [1,1]
- **flip::(a→b→c) →b→a→c**
  flip f x y = f y x
- **foldr::(a→b→b) →b→[a] →b**
  foldr (*) 3 [1,2,3] = 18
  -výpočet: (1*(2*(3*3)))
- **foldr1::(a→a→a) →[a] →a**
  foldr1 (*) [2,3,4] = 24
- **foldl::(a→b→a) →a→[b] →a**
  foldl (+) 3 [1,2,3] = 9
  -výpočet: (((3+1)+2)+3)
- **foldl1::(a→a→a) →[a] →a**
  fold1 (+) [1,2,3] = 6
- **fst::(a,b) →a**
  fst (a,b) = a
- **gcd::Integer→Integer→Integer**
  gcd x y = najväčší spoločný deliteľ
- **head::[a] →a**
  head [x,y,...] = x
  head [] - nie je definované
- **id:a→a**
  id x = x
- **init::[a] →[a]**
  init [...,x,y,z] = [...,x,y]
- **iterate::(a→a) →a→[a]**
  iterate (3+) 2 = [2,5,8,11,..]
- **last::[a] →a**
  last [...,x,y,z] = [z]
- **lcm::Integer→Integer→Integer**
  lcm x y = najmenší spoločný násobok
- **length::[a] →Int**
  length [] = 0
- **map::(a→b) →[a] →[b]**
  map (^2) [1,3,5] = [1,9,25]
- **max::Ord a => a→a→a**
  max 9 8 = 9
  max 's' 'b' = 's'
- **maximum::Ord a => [a] →a**
  maximum [2,9,5,4] = 9
- **min::Ord a => a→a→a**
  min 7 2 = 2
  min 'a' 'm' = 'a'
- **minimum::Ord a => [a] →a**
  minimum [4,6,2,8] = 2
- **mod::Integer→Integer→Integer**
  mod 9 2 = 1
  -zostatok po celočíselnom delení
- **not::Bool→Bool**
  not False = True
- **notElem::Eq a => a→[a] →Bool**
  notElem 1 [2,9,6] = True
  notElem 1 [9,1,8] = False
- **null::[a] →Bool**
  null [] = True
- **odd::Integer→Bool**
  -vracia True pre nepárne čísla
- **or::[Bool] →Bool**
  or [False,False,True] = True
- **product::[Integer] →Integer**
  product [1,2,3,4] = 24
- **repeat::a→[a]**
  repeat 3 = [3,3,..]
- **replicate::Int→a→[a]**
  replicate 5 3 = [3,3,3,3,3]
- **reverse::[a] →[a]**
  reverse [1,2,3] = [3,2,1]

Najpoužívanejšie Funkcie v Haskeli

- **show::a→String**
  show 56 = "56"
- **snd::(a,b) →b**
  snd (1,2) = 2
- **signum::Integer→Integer**
  signum (-56) = -1
  signum (56) = 1
- **sum::[Integer] →Integer**
  sum [1,2,3] = 6
- **tail::[a] →[a]**
  tail [x,y,z] = [y,z]
- **take::Int→[a] →[a]**
  take 2 [3,4,5] = [3,4]
- **takeWhile::(a→Bool) →[a] →[a]**
  takeWhile odd [1,2,3,4] = [1,3]
- **toLower::Char→Char**
  toLower 'M' = 'm'
- **toUpper::Char→Char**
  toUpper 'd' = 'D'
- **uncurry::(a→b→c) →(a,b) →c**
  uncurry f (x,y) = f x y
- **unzip::[(a,b)] →([a],[b])**
  unzip [('a',1),('b',2)] = ("ab",[1,2])
- **zip::[a] →[b] →[(a,b)]**
  zip "cd" [2,5] = [('c',2),('d',5)]
- **zipWith::(a→b→c) →[a] →[b] →[c]**
  zipWith (*) [3,4] [5,6] = [15,24]
- **(,)::a→b→(a,b)**
  (,) x y = (x,y)
- **(!!)::[a] →Int→a**
  [3,1,7] !! 2 = 7
- **(:)::a→[a] →[a]**
  (:) 1 [2,3] = [1,2,3]
- **(++)::[a] →[a] →[a]**
  (++) [1,2] [3,4] = [1,2,3,4]
- **(||)::Bool→Bool→Bool**
  True || False = True
- **(&&)::Bool→Bool→Bool**
  True && False = False
- **[]::[a]**
  prázdny zoznam

Monadické Funkcie :

- **return:: a -> IO a**
  vracia svoj argument ako výsledok akcie
- **getLine::IO String**
  načíta riadok zo štandardného vstupu
- **getChar::IO Char**
  načíta znak zo štandardného vstupu
- **putStr::String→IO ()**
  vypíše argument na štandardný výstup
- **putStrLn::String→IO ()**
  to isté čo putStr, ale pokračuje na novom riadku
- **read::Read a => String→a**
  načíta súbor
- **(>>)::IO a→ IO b→IO b**
  pre reťazenie akcií (bez použitia výsledku prvého)
- **(>>=)::IO a→(a→IO b) →IO b**
  umožňuje pristúpiť k vnútornému výsledku akcie z funkcie, ktorá vracia akciu
- **readFile::FilePath→IO String**
  načíta súbor
- **writeFile::FilePath→String→ IO()**
  zapíše do súboru

Funkcie pre stromy:

- preorder Empty = []
  preorder (Node v l r) = v:preorder l ++ preorder r
- postorder Empty = []
  postorder (Node v l r) = postorder l ++ postorder r ++ [v]
- inorder Empty = []
  inorder (Node v l r) = inorder l ++ [v] ++ inorder r