

# 1 GHCi Příkazy

Užitečné GHCi příkazy

- `:(e |exit)` - uzavře interpret;
- `:(l |load) file` - nahraje soubor;
- `:(r |reload)` - znovu nahraje poslední soubor;
- `:(i |info) function` - vypíše informace o zadané funkci;
- `:(t |type) expression` - vypíše informace o typu zadaného výrazu;
- `:(? |help)` - vypíše nápovědu o GHCi příkazech.

## 2 Základní datové typy

- Operátory pro porovnání: `==`, `/=`, `<`, `>`, `<=`, `>=`      `ord`, `chr`
- Číselné operátory: `+`, `-`, `*`, `^`, `**`      • `import Data.Char -- toUpper, isDigit`
- `Int`      • `Bool &&`, `||`, `not`, `==`
- `div`, `mod`, `abs`, `negate`      • `Double`, `Float`
- `fromIntegral -- converts Integral to Num`      `abs`, `acos`, `asin`, `sin`, `cos`, `pi`, `negate`
- `Char`      `ceiling`, `floor`, `round`, `truncate`, `exp`, `log`,

## 3 Uživatelem definované datové typy

- Synonyma: `type String = [Char]`      `data Point = Point Float Float`
- Nové typy:      `data Tree1 a = Leaf a`  
    `data Color = Black | White | Red`      | `Branch (Tree1 a) (Tree1 a)`

## 4 Definice funkce

- Technika *Pattern matching* - několik definic (rovníc) s různými *vzory* a *výrazy* na pravé straně.  

```
f pat11 pat12 ... = rhs1
f pat21 pat22 ... = rhs2
...
```

```
factorial 1 = 1
factorial n = n * factorial (n-1)
```

```
length [] = 0
length (x:xs) = 1 + length xs
```
- Výrazy:  

```
max x y = if x > y then x else y
```

```
describeList xs = "The list is " ++ case xs of
  [] -> "empty."
  [x] -> "a singleton list."
  xs -> "a longer list."
```

```
cylinder r h = let sideArea = 2 * pi * r * h
                  topArea = pi * r ^2
                  in sideArea + 2 * topArea
```
- Lokální definice:  

```
initials first last = [f]++". "++[l]++ ". "
  where f = head first
        l = head last
```

## 5 Funkce a operátory pracující se seznamy

- Přístup k elementům seznamu

```
head [5,4,3,2,1] -- 5
tail [5,4,3,2,1] -- [4,3,2,1]
last [5,4,3,2,1] -- 1
init [5,4,3,2,1] -- [5,4,3,2]
[1,2,3] !! 2 -- 3
length [5,4,3,2,1] -- 5
null [1,2,3] -- False
null [] -- True
```

- Spojování seznamů

```
[1,2,3] ++ [4,5] -- [1,2,3,4,5]
concat [[1,2],[3],[4,5]] -- [1,2,3,4,5]
zip [1,2] [3,4,5] -- [(1,3),(2,4)]
zipWith (+) [1,2] [3,4] -- [4,6]
```

- Seznamy čísel

```
minimum [8,4,2,1,5,6] -- 1
maximum [1,9,2,3,4] -- 9
sum [5,2,1,6,3,2,5,7] -- 31
product [6,2,1,2] -- 24
```

- Výběr části seznamu

```
take 3 [5,4,3,2,1] -- [5,4,3]
drop 3 [8,4,2,1,5,6] -- [1,5,6]
takeWhile (> 0) [1,3,0,4] -- [1,3]
dropWhile (> 0) [1,3,0,4] -- [0,4]
filter (> 0) [1,3,0,2,-1] -- [1,3,2]
```

- Transformace seznamů

```
reverse [5,4,3,2,1] -- [1,2,3,4,5]
map (*2) [1,2,3] -- [2,4,6]
```

- Vybrané pěkné funkce

```
4 `elem` [3,4,5,6] -- True
replicate 3 10 -- [10,10,10]
-- cycle and repeat returns infinite list
take 7 (cycle [1,2,3]) -- [1,2,3,1,2,3,1]
take 7 (repeat 5) -- [5,5,5,5,5,5,5]
foldl (+) 10 [1,2,3] -- ((10+1)+2)+3 = 16
scanl (+) 10 [1,2,3] -- [10, 11, 13, 16]
foldr (-) 10 [1,2,3] -- 1-(2-(3-10)) = -8
scanr (-) 10 [1,2,3] -- [-8,9,-7,10]
foldl1 (-) [1,2,3,4] -- -8
scanl1 (-) [1,2,3,4] -- [1,-1,-4,-8]
foldr1 (-) [1,2,3,4] -- -2
scanr1 (-) [1,2,3,4] -- [-2,3,-1,4]
```