Operatii

- `/=` diferit
- `==` egal
- `<=` si `>=`
- `rem` restul impartirii
- `div` impartire
- `*` inmultire
- `^` ridicare la putere
- `sqrt` radical

Liste

Se adauga lista la alta lista cu `` `++` ``

```
Prelude> [1,2,3] ++ [12]
[1,2,3,12]
```

Se iau elemente din lista cu :

```
-- Double every second number in a list starting on the left.
doubleEveryOther :: [Integer] -> [Integer]
doubleEveryOther [] = []
doubleEveryOther [x] = [x]
doubleEveryOther (x : y : ls) = x : 2 * y :
doubleEveryOther(ls)
```

Multimi

```
-- Sa se determine toate sirurile s = xy obtinute dintr-o lista
xs cu proprietatea ca x si y apartin listei xs si sirul s e
palindom.
--   (ex. pal ["say", "on", "not", "to", "as"] =
                                    ["sayas","noton","tonot"])
pal :: [String] -> [String]
pal ls = [x ++ y | x <- ls, y <- ls, reverse(x ++ y) == x ++ y]
```

Map

Aplica functie fiecarui element din lista `map functie lista`.
Merge apelata si ca `functie `map` lista`

```
Prelude Data.Char> map toUpper "teSt"
"TEST"
```

```
Prelude > map (\x -> x * x) [1, 2, 3, 4, 5]
[1, 4, 9, 16, 25]
```

```
Prelude Data.Char> toLower `map` "AbCCADAS"
"abccadas"
```

Filter

Returneaza numai elementele ce trec prin filtru.

```
Prelude> filter(/= 'a') "aAAbcsa"
"AAbcs"
```

```
Prelude> filter(== 1) [1, 12, 1, 14]
[1,1]
```

```
Prelude Data.Char>  filter(isLetter) "1a2b3c5d"
"abcd"
```

Foldl si prietenii

```
foldr functie element_initial (1:2:3:[])
= 1 `functie` (2 `functie` (3 `functie` element_initial))
```

```
Prelude> foldl (+) 0 [1,2,3,4,5]
15
```

```
foldl (-) 100 [1]         = 99 = ((100)-1)
foldl (-) 100 [1,2]       = 97 = (( 99)-2) = (((100)-1)-2)
foldl (-) 100 [1,2,3]     = 94 = (( 97)-3)
foldl (-) 100 [1,2,3,4]   = 90 = (( 94)-4)
foldl (-) 100 [1,2,3,4,5] = 85 = (( 90)-5)
```

```
Prelude> foldr (\x y -> concat ["(", x, "+", y, ")"]) "0" (map
show [1..13])
"(1+(2+(3+(4+(5+(6+(7+(8+(9+(10+(11+(12+(13+0))))))))))))"
```

```
Prelude> foldl (\x y -> concat ["(", x, "+", y, ")"]) "0" (map
show [1..13])
"((((((((((((0+1)+2)+3)+4)+5)+6)+7)+8)+9)+10)+11)+12)+13)"
```