Каррирование

```
> (+) 9 5
14
> sum x y = (+) x y
> sum = (+)

> :t (+)
(+) :: Num a => a -> a -> a

ДМ-6, вывод типов правоассоциативен
(+) :: Num a => a -> (a -> a)

> (+) (9 5)
?

> ((+) 9) 5
?
```

Каррирование

```
> let sumWith9 = (+) 9

> :t a
a :: Integer -> Integer

> sumWith9 5
14
> sumWith9 90
99

> let res = sumWith9 42
51
> :t res
res :: Integer
```

ФВП

```
> let doItAgain f x = f (f x)

> doItAgain sumWith9 3
21

zipWith' :: (a -> b -> c) -> [a] -> [b] -> [c]
?

> zipWith (+) [1, 2, 3, 4] [4, 3, 2, 1]
[5, 5, 5, 5]

> zipWith (++) ["CSKA", "Zenith"] ["champion", "champion"]
```

ФВП

```
flip' :: (a -> b -> c) -> b -> a -> c
flip' f y x = f x y

> flip' zip [1,2,3,4,5] "hello"
[('h',1),('e',2),('l',3),('l',4),('o',5)]

> zipWith (flip' div) [2,2..] [10,8,6,4,2]
[5,4,3,2,1]
```

ФВП

```
map :: (a -> b) -> [a] -> [b]
map _ [] = []
map f (x:xs) = f x : map f xs

> map (+3) [1,5,3,1,6]
[4,8,6,4,9]

> map (++ "!") ["BIFF", "BANG", "POW"]
["BIFF!","BANG!","POW!"]

> map (replicate 3) [3..6]
[[3,3,3],[4,4,4],[5,5,5],[6,6,6]]

> map (map (^2)) [[1,2],[3,4,5,6],[7,8]]
[[1,4],[9,16,25,36],[49,64]]

> map fst [(1,2),(3,5),(6,3),(2,6),(2,5)]
[1,3,6,2,2]
```

ФВП

```
filter :: (a -> Bool) -> [a] -> [a]
filter _ [] = []
filter p (x:xs)
  | p x       = x : filter p xs
  | otherwise = filter p xs

> filter (>3) [1,5,3,2,1,6,4,3,2,1]
[5,6,4]
> filter (==3) [1,2,3,4,5]
[3]
> filter even [1..10]
[2,4,6,8,10]
> let notNull x = not (null x) in filter notNull [[1,2,3],[],
[3,4,5],[2,2],[],[],[]]
[[1,2,3],[3,4,5],[2,2]]
> filter (`elem` ['a'..'z']) "u LaUgH aT mE BeCaUsE I aM
diFfeRent"
"uagameasadifeent"
> filter (`elem` ['A'..'Z']) "i lauGh At You BecAuse u r aLL the
Same"
"GAYBALLS"
```

\-выражения

```
square x = x * x
oddsSquares = filter (\x -> odd x) (map square [1..100])

oddsSquares = filter (\x -> odd x) (map (\x->x*x) [1..100])

square = \x → x * x

flip' :: (a -> b -> c) -> b -> a -> c
flip' f = \x y -> f y x
```

Показать список

```
map (\(x,y) -> show x ++ "). " ++ y) $ zip [1..]
["Трус", "Балбес", "Бывалый"]
```

\-выражения

```
calcDistance (x,y) (a,b) = sqrt (w^2 + h^2)
  where (w,h) = (x-a, b-y)

points = [(1,1),(5,3),(2,5),(7,4),(5,5),(3,2),(0,4)]

sort [] = []
sort ((x,o):xs) =
    let smallerSorted = sort (filter (\(y,o) -> y <= x) xs)
        biggerSorted = sort (filter (\(y,o) -> y > x) xs)
    in  smallerSorted ++ [(x,o)] ++ biggerSorted

nearest3 dest = take 3 (sort (map (\p -> (calcDistance p dest, p))
points))

> nearest3 (2,2)

nearest3 dest = take 3 (filter (\(dist, p) -> dist < 2) (sort (map
(\p -> (calcDistance p dest, p)) points)))
nearest3 dest = filter (\(dist, p) -> dist < 2) (take 3 (sort (map
(\p -> (calcDistance p dest, p)) points)))
 > nearest3 (2,2)
```

\-выражения (ECMA-5, ff,ch,o,ie9)

```
var metroStations = [{lat: 52.234, lng: 30.482, name:
'Kashirskaya'},..]
    , calcDistance = function(point, station) {
        station.dist = 180 * arctan … // distance algorythm (mutable)
        return station;
    };

nearest3 = function(point) {
    return metroStations
            .map(function(m) {return calcDistance(point, m);})
            .sort(function(a,b) {return a.dist > b .dist;})
            .slice(0,3)
            .filter(function(m) {return m.dist < 2000;})
};

nearest3({lattitude: 52.0, longitude: 31.0})
```

Применение функций

```
($) :: (a -> b) -> a -> b
f $ x = f x
```

f (g (z x)) === f $ g $ z x правоассоцитивно

```
nearest3 dest = filter (\(dist, p) -> dist < 2) $ take 3 $ sort $ map (\p -> (calcDistance p dest,
p)) points
```

No more LISP!

```
nearest3 dest = filter (\(dist, p) -> dist < 2) (take 3 (sort (map (\p -> (calcDistance p dest, p))
points)))
```

```
> sum 9 (sum 6 (sum 3 1))
> sum 9 $ sum 6 $ sum 3 1
```

Композиция функций

```
(.) :: (b -> c) -> (a -> b) -> a -> c
f . g = \x -> f (g x)

> map (\xs -> negate (sum (tail xs))) [[1..5],[3..6],[1..7]]

[-14,-15,-27]

> map (negate . sum . tail) [[1..5],[3..6],[1..7]]
[-14,-15,-27]
```

Композиция функций, putStr

```
nearest3 dest = take 3 . sort . map (\p -> (calcDistance p
dest, p))

main = do
  putStr . show $ nearest3 (2,2) points

{-
main = do
  putStrLn . show $ nearest3 (2,2) points
-}
```

mapM

```
main = do
  mapM putStrLn ["adf\taff","dfaf\tsf","asdf\tf","asddf\tf"]

> main
adf   aff
dfaf  sf
asdf  f
asddf f
[(),(),(),()]


main = do
  mapM putStrLn ["adf\taff","dfaf\tsf","asdf\tf","asddf\tf"]
  Return ()

main = do
  mapM_ putStrLn ["adf\taff","dfaf\tsf","asdf\tf","asddf\tf"]
```

Задание

Числа Фиббоначе, бесконечный список

[1, 1, 2, 3, 5, 8, 13, ...]

КР!

Список треугольных чисел

Список пирамидальных чисел

n-е треугольное число tn равно количеству одинаковых монет, из которых можно построить равносторонний треугольник, на каждой стороне которого укладывается n монет.

n-е пирамидальное число pn равно количеству одинаковых шаров, из которых можно построить пирамиду с треугольным основанием, на каждой стороне которой укладывается n шаров