Podstawowe komendy obsługi:

komenda	skrót	co robi
:load test.hs	: l test hs	czyta skrypt test.hs
:reload	: r	czyta ponownie bieżący skrypt
:set editor test.hs	:s editor test.hs	ustala edytor do skryptu
:edit test.hs	:e test.hs	edytuje test.hs w ustalonym edytorze
:edit	:e	edytuje bieżący skrypt
:quit	:q	wyjsciez GHCI

Komentarze

```
-- jedna linia
{-
blok kodu
-}
```

Elementarne operacje arytmetyczne

```
2+3*4
(2+3)*4
sqrt(3^2 + 4^2)
```

Niektóre Typy

Podstawowe Typy

```
Bool -- typ logiczny zawiera False i True.
String  -- typ łańcuchowy zweiera łańcuchy np.: "ala ma"
Int  -- typ zawierajacy liczby całkowite np.: 1,2, -9, 0
Integer -- typ zawierajacy liczby całkowite dowolnego rzędu
Double - typ odpowiadający liczba rzeczywistym Double 1.545, 8.0098
```

Typy List

```
[False,True,False] :: [Bool]
["One","Two","Three"] :: [String]
[1,2,3,4] :: [Integer]
```

Typy Funkcji

```
not :: Bool -> Bool
even :: Int -> Bool
--np.:
not True
even 5
```

Typy Krotek (Tupli)

```
(False,True) :: (Bool,Bool)
(False,'ala',True) :: (Bool,String,Bool)
(1,3) :: (Int,Int)
```

Typy całkowite

```
div :: a -> a -> a
mod :: a -> a -> a
--np.:
div 24 10
mod 24 10
```

Typy Porównania

```
(==) :: a -> a -> Bool
(/=) :: a -> a -> Bool
--np.:
"abc" == "abc"
[1,2] == [1,2,3]
```

Typy Porządkowe

```
(<) :: a -> a -> Bool
(<=) :: a -> a -> Bool
min :: a -> a -> a
max :: a -> a -> a
--np.:
False < True
"elegant" < "elephant"
[1,2,3] < [1,2]
('as',2) < ('as',1)
min 5 9
```

Typy Numeryczne

```
(+) :: a -> a -> a
(-) :: a -> a -> a
(*) :: a -> a -> a
negate :: a -> a
--np.:
1+3
negate 9
```

Elementarne operacje na listach

```
head [1,2,3,4,5] -- zwraca pierwszy element z niepustej listy
tail [1,2,3,4,5] -- usuwa pierwszy element z niepustej listy
last [1,2,3,4,5] -- zwraca ostatni element listy
init [1,2,3,4,5] -- usuwa pierwszy element listy
take 3 [1,2,3,4,5] -- zwraca pierwsze 3 elementy z niepustej listy
drop 3 [1,2,3,4,5] --usuwa pierwsze 3 elementy z niepustej listy
length [1,2,3,4,5] -- zwraca ilooa elementów listy
sum [1,2,3,4,5]  -- sumuje elementy listy
product [1,2,3,4,5]  -- oblicza iloczyn elementów listy
reverse [1,2,3,4,5] -- odwraca kolejnosc elementów listy
```

Operatory działajace na listach

```
1 [1,2,3] ++ [4,5]  -- operator konkatenacji ++
2 [1,2,3,4,5] !! 2  -- zwraca 2 (licząc od zera) element z niepustej listy
```

Pisanie skryptów

Skrypt test.hs

```
1 podwojenie x = x + x
2 podwojnepodwojenie x = podwojenie (podwojenie x)
```

Wywołanie skryptu test.hs

```
1 podwojnepodwojenie 10
2 take (double 2) [1,2,3,4,5]
```

Skrypt test2.hs

```
1 iloczyn n = product [1..n]  --iloczyn n liczb
2 srednia ns = div (sum ns) (length ns) -- średnia liczb w ns
```

Wywołanie skryptu test2.hs

```
1 iloczyn 10
2 srednia [1,2,3,4,5]
```

Skrypt test3.hs

```
1 a = b + c
2         where
3            b = 1
4            c = 2
5 d = a * 2
```

Wywołanie skryptu test3.hs

```
1 a
```

Skrypt test4.hs

```
1 --Suma liczb listy ns
2 mysum [] = 0
3 mysum (n:ns) = n + sum ns
```

Wywołanie skryptu test4.hs

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

Skrypt test5.hs

```
1 --Iloczyn liczb listy ns
2 myproduct [] = 1
3 myproduct (n:ns) = n * product ns
```

Wywołanie skryptu test5.hs

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

Skrypt test6.hs

```
1 --własne funkcje last
2 mylast1 xs = head (reverse xs)
3 mylast2 xs = xs !! (length xs - 1)
```

Wywołanie skryptu test6.hs

```
1 mylast1 [1,2,3,4]
2 mylast2 [1,2,3,4]
```

Skrypt test7.hs

```
1 n = div a (length xs)
2    where
3      a = 10
4      xs = [1,2,3,4,5]
```

Wywołanie skryptu test7.hs

```
1 n
```

Skrypt test8.hs

```
1 myeven n = (mod n 2) == 0
```

Wywołanie skryptu test8.hs

```
1 myeven 7
```

Skrypt test9.hs

```
1 mysplit n xs = (take n xs, drop n xs)
```

Wywołanie skryptu test9.hs

```
1 mysplit 3 [8,9,0,7,5,4]
```

Skrypt test10.hs

```
1 ulamek n = 1/n
```

Wywołanie skryptu test10.hs

```
1 ulamek 5
```