# Декларативное программирование

Знакомство с Haskell.
Простые функции

# Введение

Ахмадеева Ирина Равильевна

i.akhmadeeva@g.nsu.ru

https://t.me/i_r_akhmadeeva

| | |
|---|---|
| 1 семестр | Haskell |
| 2 семестр | SQL |

# Виды заданий

1. Упражнение: небольшое задание на занятии
2. «Пятиминутка»: проверочная работа на 5 мин
3. Домашнее задание: задание на 1-2 недели
4. Контрольная работа: проверочная работа на 30-45 мин
5. Проект: большое индивидуальное задание по выбору (2 семестр)

# Оценка за 1 семестр

1. Упражнение: 1 балл * 10
2. «Пятиминутка» : 1 балл * 10
3. Домашнее задание : 5 баллов * 10
4. Контрольная работа : 10 баллов * 3

# Материалы

- Лекции
- http://learnyouahaskell.com/chapters
- Изучай Haskell во имя добра! М.Липовача
- https://haskell.mooc.fi/

# Инструменты

- Glasgow Haskell Compiler (GHC)
- GHCi — GHC's interactive environment

Запускаем GHCi: в командной строке пишем ghci

```
> ghci
Prelude>
```

Prelude — стандартная библиотека полезных функций

# Выражения

Мы можем использовать ghci в качестве калькулятора.

```
ghci> 12 + 30
42
```

Вычислите значения следующих выражений:

1. 2 + 15
2. 49 * 100
3. 189 - 147
4. 5 / 2
5. 5 `div` 2
6. 5 * -3
7. 5*-3
8. 5*(-3)
9. False || True
10. not (True && True)
11. 1000^1000
12. 7^1
13. 7**1
14. 7^(-1)
15. 7**(-1)
16. 5 == 7
17. 's' /= 'S'
18. 5 == True
19. "Hello, " ++ "world"
20. "string" ++ True
21. True + 1
22. True +

# Некоторые типы данных

| Тип | Значения | Описание | Операции |
|---|---|---|---|
| Int | 1, 2, -3, … | Целые числа | +, -, *, div, mod,… |
| Integer | 1, -2, 100000000000, … | Неограниченные целые числа | +, -, *, div, mod,… |
| Double | 0.1, 1.2e5, … | Числа с плавающей точкой | +, -, *, /, sqrt,… |
| Bool | True, False | Булевы значения | &&, \|\|, not,… |
| Char | 'a', 'b', 'c', … | Символы | isUpper, isDigit (Data.Char) |
| String или [Char] | "abcd", "hello", "", … | Строки | reverse, ++,… |

https://hackage.haskell.org/package/base/docs/Prelude.html

# Функции

Haskell – это функциональный язык программирования

Функция:

- принимает аргумент(ы)
- возвращает значение
- при вызове с одним и тем же аргументом, должна возвращать одно и то же значение

# Применение функции

Имя функции Аргумент(ы)

```
succ 8
min 3 5
min (max 3 5) (min 4 9)
min 3 5 + max 4 9
```

| Haskell | C, python, … |
|---|---|
| f x | f(x) |
| f x y | f(x, y) |
| g h f 1 | g(h, f, 1) |
| g h (f 1) | g(h, f(1)) |
| g (h f 1) | g(h(f, 1)) |
| g (h (f 1)) | g(h(f(1)) |
| f a + g b | f(a) + g(b) |

# Некоторые функции

succ

min

max

div

quot

sqrt

log

logBase

https://hackage.haskell.org/package/base/docs/Prelude.html

# Остаток от деления

Вычислите, сравните результат:

- mod 42 10
- 42 `mod` 10
- 42 `rem` 10
- (-42) `rem` 10
- (-42) `mod` 10
- -42 `mod` 10

# Остаток от деления

Вычислите, сравните результат:

- mod 42 10
- 42 `mod` 10
- 42 `rem` 10
- (-42) `rem` 10
- (-42) `mod` 10
- -42 `mod` 10

```
(x `quot` y)*y + (x `rem` y) == x
(x `div`  y)*y + (x `mod` y) == x
```

# Префиксная и инфиксные формы

Префиксная форма записи:

- succ 1
- min 1 2
- (+) 2 2
- (&&) True False
- (**) 2 (-1)

Инфиксная форма записи:

- 5 + 2
- 7 / 3
- 42 `div` 10
- 42 `quot` 10
- (-42) `div` 10
- (-42) `quot` 10

# Связывание

Знак равенства задает связывание имя (слева) связывается со значением (справа):

```
x = 42

y = 5

z = 10 :: Int
```

# Объявление и определение функции

тип функции

```
square :: Int -> Int
```

аргумент(ы)

```
square v = v * v
```

имя функции (с маленькой буквы)

```
ghci> :{
| square :: Int -> Int
| square v = v * v
| :}
```

# Сохранение функций в файл

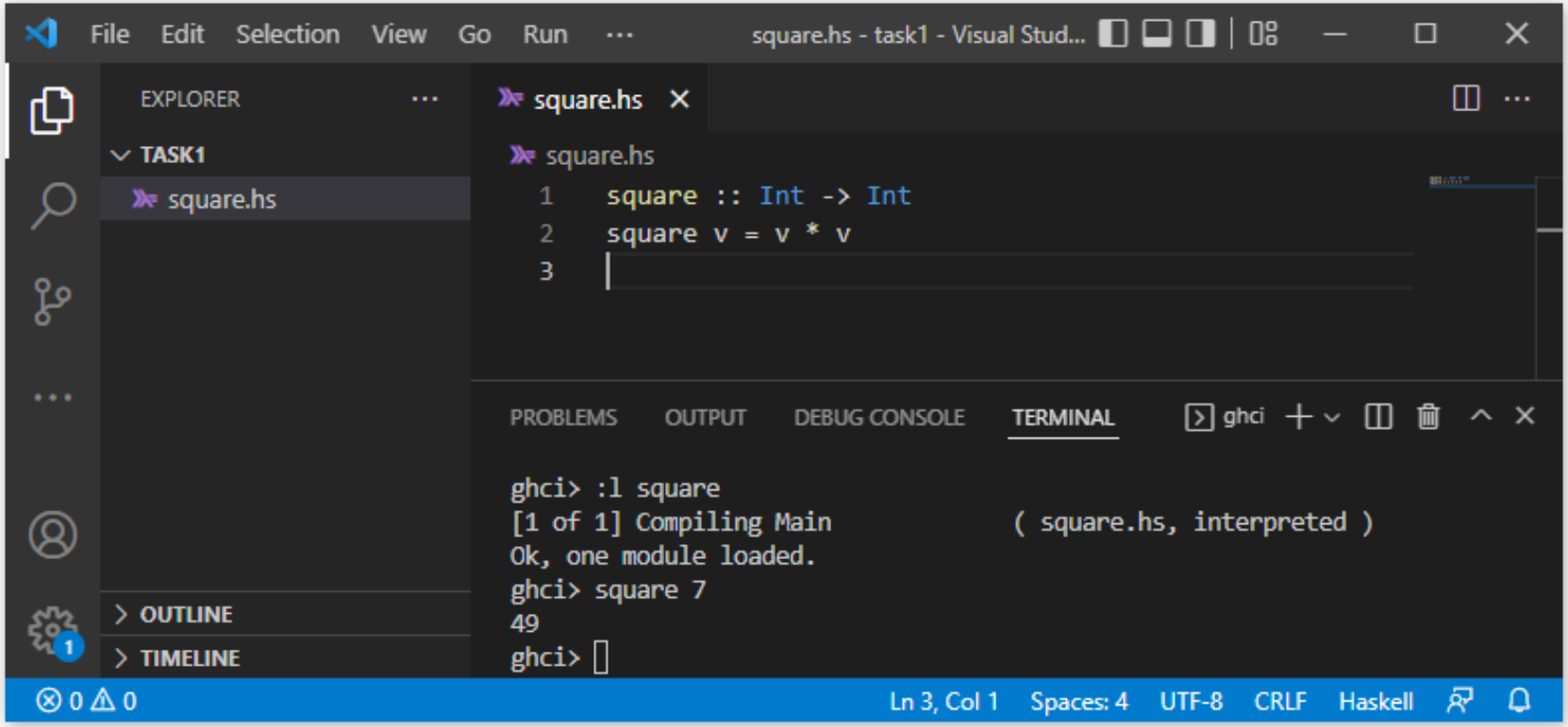
```
square :: Int -> Int
square v = v * v
```

square.hs

```
ghci> :l square.hs
[1 of 1] Compiling Main             ( square.hs, interpreted )
Ok, one module loaded.
ghci> square 7
49
```

# Visual Studio Code

- Расширение: **Haskell Syntax Highlighting**



# Полезные команды ghci

- :? напечатать список доступных команд и их краткое описание
- :q выйти из GHCi
- :l <имя файла> загрузить программу с указанным именем
- :r перегрузить текущий модуль;
- :t <выражение> напечатать тип выражения
- :i <имя функции> напечатать сигнатуру функции и указать, в каком файле была определена данная функция

# Упражнение 1

Реализуйте функцию, **утраивающую** значение своего аргумента.

Функция должна работать с числами типа **Double**.

Сохраните определение в **файл**.

# Упражнение 2

Определите функции, вычисляющие следующие выражения:

$$y = \frac{x^2}{1+x}$$

$$y = \sqrt{3 * x - x^3}$$

$$y = lg\left(x^2 - 21\right).$$

# Условное выражение

```
f x = if x > 0 then 1 else (-1)

absoluteValue n = if n < 0 then -n else n
```

Упражнение 3: с использованием условных выражений реализуйте функцию sign, которая возвращает:

- 1, если ей передано положительное число,
- (-1), если отрицательное,
- 0 в случае, когда передан 0.

# Сопоставление с образцом

Определение функции может состоять из нескольких уравнений. Уравнения сопоставляются по порядку с аргументами, пока не будет найдено подходящее. Это называется сопоставлением с образцом.

```
greet :: String -> String -> String
greet "Finland" name = "Hei, " ++ name
greet "Italy" name = "Ciao, " ++ name
greet "England" name = "How do you do, " ++ name
greet _ name = "Hello, " ++ name
```

Упражнение 4: добавьте поддержку России для функции greet.

https://stackoverflow.com/questions/3611656/haskell-io-with-non-english-characters

# Охранные выражения

```
howMuch n | n > 10 = "a whole bunch"
          | n > 0 = "not much"
          | otherwise = "we're in debt!"
```

Упражнение 5: используя охранные выражения, реализуйте функцию sign', которая возвращает:

- 1, если ей передано положительное число,
- (-1), если отрицательное,
- 0 в случае, когда передан 0.

# Преобразование типов

```
half :: Int -> Double
half n = n / 2
```

# Преобразование типов

```
half :: Int -> Double
half n =(fromIntegral n) / 2
```

# Домашнее задание

1. Прочитать раздел 1.9 https://haskell.mooc.fi/part1#how-do-i-get-anything-done
2. Выполнить Задание 1 в Google Classroom