

Функции и стойности

ИТ Кариера



Учителски екип
Обучение за ИТ кариера

<https://it-kariera.mon.bg/e-learning>

Съдържание

- Функции
- Чисти функции
- Функции като стойности на функция
- Рекурсия



Функции

- Основен градивен блок в Haskell
- Поредица от команди, която може да бъде преизползвана
 - Всяка функция изпълнява определена задача, приема n на брой параметри и връща резултат спрямо подадените параметри
- Повечето програмни езици използват функции (също се наричат и методи, процедури)

Дефиниране на функция

- В контекста на GHCi функцията се дефинира с ключовата дума `let`, за да е ясно, че се декларира нова функция, а не се извиква съществуваща

```
square x = x * x
```

```
let square x = x * x
```

Дефиниране на функция

- В контекста на GHCi функцията се дефинира с ключовата дума `let`, за да е ясно, че се декларира нова функция, а не се извиква съществуваща

```
square x = x * x
```

име на
функцията

```
let square x = x * x
```


Дефиниране на функция

- В контекста на GHCi функцията се дефинира с ключовата дума `let`, за да е ясно, че се декларира нова функция, а не се извиква съществуваща

```
square x = x * x
```

име на
функцията

```
let square x = x * x
```

Дефиниране на функция

- В контекста на GHCi функцията се дефинира с ключовата дума `let`, за да е ясно, че се декларира нова функция, а не се извиква съществуваща

`square x = x * x`

име на
функцията

параметрите, които функцията приема
се изреждат, разделени от интервал,
без да се обграждат от скоби

тяло на функцията: съдържа
израз, чиято стойност се
връща като резултат

`let square x = x * x`

Дефиниране на функция с повече параметри

```
multMax a b x = (max a b) * x
```


Дефиниране на функция с повече параметри

```
multMax a b x = (max a b) * x
```

първо се изпълнява функцията `max`, която е вградена за Haskell и връща по-големият от два параметъра

Дефиниране на функция с повече параметри

```
multMax a b x = (max a b) * x
```

резултатът от
функцията max се
умножава по x

първо се изпълнява функцията
max, която е вградена за
Haskell и връща по-големият
от два параметъра

Извикване на функция

- Извикване на вградената в Haskell `sqrt` функция, която извежда корен
корен квадратен

```
sqrt 3
```

```
max 3 7
```

- Резултат:

```
1.7320508075688722
```

```
7
```

Извикване на функция

- Извикване на вградената в Haskell `sqrt` функция, която извежда корен
корен квадратен

`sqrt 3`

функцията приема като
параметър едно число

`max 3 7`

- Резултат:

`1.7320508075688722`

`7`

Извикване на функция

- Извикване на вградената в Haskell `sqrt` функция, която извежда корен квадратен

`sqrt 3`

функцията приема като параметър едно число

`max 3 7`

когато параметрите са повече от един, те се изреждат без запетайи или скоби

- Резултат:

`1.7320508075688722`

`7`

Функции – кога има нужда от скоби?

- Групиране:

```
(5 + 2) * (3 - 4)
```

```
max (5 + 2) (sqrt 17)
```

- Подаване на отрицателно число като параметър:

```
max 5 (-5)
```

Функции – кога има нужда от скоби?

- Групиране

$(5 + 2) * (3 - 4)$

Резултат: 7

`max (5 + 2) (sqrt 17)`

Резултат: 7

- Подаване на отрицателно число като параметър

`max 5 (-5)`

Резултат: 5

Задача:

- Дефинирайте функция, която приема 1 параметър n (целочислен тип) и връща като резултат абсолютната стойност на n и прибавя към нея 5

Решение:

```
plusFive n =  
  if n < 0  
  then (-n) + 5  
  else n + 5
```

един възможен подход е да
проверявате дали числото е
по-малко от 0 и съответно да
го преобразувате

```
plusFive' n = (abs n) + 5
```

може и да използвате вградения
в стандартната библиотека за
Haskell abs метод

Резултат: `plusFive 5 -> 10; plusFive (-5) -> 10`

Чисти функции

- Всички функции в Haskell са чисти функции
 - не могат да променят нито локално, нито глобално състояние
 - не могат да зависят от текущо състояние
 - при подаване на едни и същи параметри дадена функция винаги връща един и същ резултат

Чисти функции – примери

- Принтиране на текст в конзола
 - не е чиста функция - променя външно състояние
- Четене на файл
 - не е чиста функция - зависи от външно състояние
- Пресмятане дължината на символен низ
 - чиста функция - не зависи от външно или вътрешно състояние
- Вземане на текущото време
 - не е чиста функция - в зависимост кога е извикана връща различен резултат
- Вземане на произволно число
 - не е чиста функция - връща различен резултат всеки път при извикването си

Функции като стойности на функция

- Функциите в Haskell се възприемат като тип променлива (подобно на `Int`, `Char` в процедурните и обектно-ориентирани езици)
- Могат да се подават като параметри на други функции
- Функции от по-висок ред
- Това са функции, които или приемат като параметър една или повече функции, или връщат като резултат функция

Функции като стойности на функция – примери

```
pass3 f = f 3  
add1 x = x + 1  
pass3 add1
```

Функции като стойности на функция – примери

функцията `pass3` приема като параметър функция `f`, на която подава 3 като параметър

```
pass3 f = f 3  
add1 x = x + 1  
pass3 add1
```

Функции като стойности на функция – примери

```
pass3 f = f 3  
add1 x = x + 1  
pass3 add1
```

функцията pass3 приема като параметър функция f, на която подава 3 като параметър

функцията add1 приема един параметър, към който прибавя 1

Функции като стойности на функция – примери

```
pass3 f = f 3  
add1 x = x + 1  
pass3 add1
```

функцията pass3 приема като параметър функция f, на която подава 3 като параметър

функцията add1 приема един параметър, към който прибавя 1

функцията pass3 е извикана с параметър add1, а по време на изпълнението и add1 се изпълнява с параметър 3

Функции като стойности на функция – примери

- Резултат:

```
pass3 f = f 3  
add1 x = x + 1  
pass3 add1
```

функцията pass3 приема като параметър функция f, на която подава 3 като параметър

функцията add1 приема един параметър, към който прибавя 1

функцията pass3 е извикана с параметър add1, а по време на изпълнението и add1 се изпълнява с параметър 3

4

Функции като стойности на функция – примери

```
compose f g x = f (g x)
```

```
add1 x = x + 1
```

```
mult2 x = 2 * x
```

```
compose add1 mult2 4
```

Функции като стойности на функция – примери

```
compose f g x = f (g x)
add1 x = x + 1
mult2 x = 2 * x
```

функцията `compose` приема 3 параметъра, от които 2 са функции, подава параметърът `x` на функцията `g`, след което подава полученият резултат на функцията `f`

```
compose add1 mult2 4
```

Функции като стойности на функция – примери

```
compose f g x = f (g x)
```

```
add1 x = x + 1
```

```
mult2 x = 2 * x
```

функцията `compose` приема 3 параметъра, от които 2 са функции, подава параметърът `x` на функцията `g`, след което подава полученият резултат на функцията `f`

```
compose add1 mult2 4
```

резултатът след изпълнението е 9

Задача:

- Дефинирайте функция, която приема като параметри две функции и число и последователно извиква двете функции, подавайки им числото като параметър

Решение:

`execute f g a = f (g a)`

скобите са нужни, за да се дефинира приоритета на операциите

Резултат: `execute sqrt sqrt 5`
`-> 1.4953487812212205`

Рекурсия

- Процес, при който функция извиква себе си директно или индиректно (пример: функцията а извиква функцията b, която извиква функцията а)
- Техника за решение на проблеми
 - Разделя всеки проблем на подпроблем от същия тип
- Рекурсията трябва да има дъно (базов случай)
 - Всяка стъпка от рекурсията трябва да се стреми към така дефинираното дъно
- В Haskell рекурсията се използва като заместител на циклите, които езикът не поддържа
 - За целта се използва помощна функция

Рекурсия – примеры

```
pow2 n =  
    if n == 0  
    then 1  
    else 2 * (pow2 (n - 1))
```

Рекурсия – примери

```
pow2 n =  
    if n == 0  
    then 1  
    else 2 * (pow2 (n - 1))
```

функцията pow2 връща като резултат n -тата степен на 2-ката като използва рекурсия, за да я намери

Рекурсия – примери

```
pow2 n =  
  if n == 0  
  then 1  
  else 2 * (pow2 (n - 1))
```

функцията pow2 връща като резултат
n-тата степен на 2-ката като
използва рекурсия за да я намери

случаят в който n достигне
0 е дъното на рекурсията,
тогава функцията връща
резултат 1

Рекурсия – примери

```
pow2 n =  
  if n == 0  
  then 1  
  else 2 * (pow2 (n - 1))
```

функцията pow2 връща като резултат n -тата степен на 2-ката като използва рекурсия, за да я намери

случая, в който n достигне стойност 0 е дъното на рекурсията. Тогава функцията връща резултат 1

В противен случай функцията връща 2 умножено по резултата от функцията за $n - 1$. Така проблемът се разбива на малки подпроблеми и се стига до крайният резултат

Рекурсия – примери

- Функция, слепваща символен низ n на брой пъти
- При $n = 0$ функцията връща празен символен низ
- В противен случай свежда проблема до подпроблем от същия тип

```
repeatString str n =  
    if n == 0  
    then ""  
    else str ++ (repeatString str (n-1))
```

Обобщение

- Функции
- Чисти функции
- Функции като стойности на функция
- Рекурсия



Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист"



Министерство
на образованието
и науката



Национална
програма
„Обучение за
ИТ кариера“

- Курсът се разпространява под свободен лиценз **CC-BY-NC-SA**

