

Linguagens Funcionais: Haskell

Prática 1

Paradigmas de Linguagens Computacionais

Monitores: Bruno Barros (blbs)
Caio Neves (ccno)
Thiago Lacerda (tbl2)

Hugs

- Interpretador de Haskell
- Suporta todos os recursos da linguagem
- Para iniciá-lo, execute `winhugs.exe`
- Hugs é gratuito. Você pode obtê-lo em www.haskell.org/hugs.
- Versão mais nova encontra-se na página da monitoria

Hugs: Comandos

`:quit`

Sai do Hugs

`:?`

Exibe a ajuda do Hugs

`:load <caminho_do_arquivo>`

Carrega um arquivo para o Hugs (import)

`:reload`

Recarrega o último arquivo carregado

Hugs: Expressões

O [Hugs](#) pode computar o valor de expressões [Haskell](#). Basta apenas digitá-las no console.

```
> 2+3
```

```
5
```

```
> head [3,2,1]
```

```
3
```

```
> tail [3,2,1]
```

```
[2,1]
```

Exercício 1: Expressões

Calcule o valor das seguintes expressões (no [Hugs](#)):

> sum [1,2,3,4]

> product [1,2,3,4]

> "abc" ++ "def"

> fst (2,3)

Hugs: Tipos

Toda expressão de **Haskell** tem um tipo associado e o **Hugs** permite descobri-lo através do comando **:type**

```
> :t 2+3
```

```
2 + 3 :: Num a => a
```

```
> :t head [3,2,1]
```

```
head [3,2,1] :: Num a => a
```

```
> :t tail
```

```
tail :: [a] -> [a]
```

Exercício 2: Tipos

Encontre expressões cujos tipos são:

Char

[Char]

(Int, Int)

(Bool, [Char])

[(Bool, Char)]

Exercício 3: Tipos

Encontre o tipo das seguintes expressões:

head

sum

fst

elem

flip

flip elem

Hugs: Arquivos

`:load <arquivo>`

Carrega o arquivo no Hugs

`:reload`

Recarrega o último arquivo

Exercício 4: Arquivos

Carregue o arquivo [exercicio.hs](#) e calcule as seguintes expressões:

`square 2`

`allEqual 2 3 4`

`allEqual 1 1 1`

`allEqual (square 2) (square (-2)) 4`

`maxi (square 2) 3`

Exercício 5: Arquivos

Modifique o arquivo [exercicio.hs](#), incluindo as seguintes funções:

`fat :: Int -> Int`

Calcula $n!$

`all4Equal :: Int -> Int -> Int -> Int -> Bool`

Compara se quatro números são iguais

`all4Equal2 :: Int -> Int -> Int -> Int -> Bool`

Mesmo que anterior, mas usando a definição de `allEqual`

Exercício 6: Sales

Modifique o arquivo `sales.hs`, incluindo as seguintes funções:

`maxSales :: Int -> Int`

Dada uma semana `n`, retorna a semana com maior número de vendas entre `0` e `n`

`totalSales :: Int -> Int`

Dada uma semana `n`, retorna a soma das vendas entre as semanas `0` e `n`

Exercício 7: Sales

Modifique o arquivo `sales.hs`, incluindo as seguintes funções:

`howManyWeeks :: Int -> Int -> Int`

Dado um valor de vendas e uma semana `n`, determina quantas semanas entre `0` e `n` tiveram essa vendagem

`averageSales :: Int -> Float`

Dada uma semana `n`, calcula a média de vendas entre `0` e `n`

Exercício 8: Strings

Defina as seguintes funções:

`makeSpaces :: Int -> String`

Produz uma String com `n` espaços vazios

`pushRight :: Int -> String -> String`

Adicionar `n` espaços à esquerda de uma String (deve usar `makeSpaces`)

Exercício 9: Tipos

Defina os tipos **Point** e **Triangle** (a partir de **Point**) e as funções **distance** (calcula a distância entre dois pontos), **midPoint** (calcula o ponto médio entre dois pontos) e **perimeter** (calcula o perímetro do triângulo)

```
distance :: Point -> Point -> Float
midPoint :: Point -> Point -> Point
perimeter :: Triangle -> Float
```