Programação Funcional

```
universidade = "Universidade Federal de Alfenas"
professor = "Romário da Silva Borges"
```

Aprenderemos nesta aula...

- Conceitos introdutórios sobre programação funcional;
- Introdução a linguagem Haskell e a alguns comandos básicos;

História do paradigma Funcional

- 1930 - Alonzo Church e o cálculo Lambda;
- 1960 - Criação da linguagem LISP, John McCarthy no MIT;
- 1970 - Meta Linguagem com inferência de Tipos;
- 1987 - Nasce Haskell. Nome tem homenagem a Haskell Brooks Curry;

Programação funcional

- Declarativa;
- Linguagens que implementam a programação funcional:
  Haskell, F#, Scala, JS, Python
- Paradigma Imperativo -> “Como Fazer?”
  Paradigma Funcional -> “O que fazer?”

Programação funcional

Soma de valores em C:

```c
int soma = 0;
for (int i = 1; i <= 10; i++) {
    soma += i;
}
```

Soma de valores em Haskell:

```haskell
sum [1..10]
```

Características: Função pura

Uma função pura é aquela que sempre retorna o mesmo resultado para as mesmas entradas e não causa efeitos colaterais (não altera variáveis, arquivos, nem depende de algo fora dela).

```
const numbers = [1, 2, 3, 4, 5];

// puro
numbers.slice(0, 3); // [1,2,3]

numbers.slice(0, 3); // [1,2,3]

numbers.slice(0, 3); // [1,2,3]

// impuro
numbers.splice(0, 3); // [1,2,3]

numbers.splice(0, 3); // [4,5]

numbers.splice(0, 3); // []
```

Características: Imutabilidade

```
x = 10
y = x + 5
z = y * 2
```

Na programação funcional, os valores não mudam após serem criados. Em vez de modificar uma variável, cria-se um novo valor a partir do anterior

Características: Composição de funções

```
const doubleNumber = (x) => x * 2;

Math.sqrt(doubleNumber(doubleNumber(doubleNumber(2)))); // 4
```

É o ato de juntar funções simples para formar funções mais complexas. O resultado de uma função é passado como entrada para outra.

Características: Avaliação preguiçosa

```
take 5 [1..]    -- retorna [1,2,3,4,5] mesmo sendo uma lista infinita
```

Haskell só calcula o que for realmente necessário, e apenas quando for preciso. Isso permite trabalhar com listas infinitas e otimizar desempenho.

Características: Alta Ordem

```
aplica f x = f x
aplica (*2) 10   -- resultado: 20
```

Funções podem receber outras funções como parâmetro ou retornar funções — tratadas como valores comuns

Características: Inferência de tipos

```
:t 5
5 :: Num a => a   -- Haskell deduz que é um número
```

Haskell descobre automaticamente o tipo de cada valor, mas não permite misturar tipos incompatíveis.

Características: Recursão X laços

```
fatorial 0 = 1
fatorial n = n * fatorial (n - 1)
```

Como não há variáveis mutáveis nem loops tradicionais (for, while), usamos recursão para repetir operações

Algumas aplicações

Exemplo de plataforma construída em Haskell.

Inspirado em conceitos de programação funcional, como imutabilidade (estados e propriedades não devem ser modificados diretamente) , funções puras (componentes funcionais devem produzir o mesmo resultado para as mesmas entradas, sem efeitos colaterais), entre outras características.

O WhatsApp foi construído originalmente em Erlang, uma linguagem funcional, devido a confiabilidade e imutabilidade.

Compilador GHC (Glasgow Haskell Compiler)

- Compilador que gera código-máquina nativo;
- Possui o interpretador ghci;
- O compilador é escrito em Haskell;

Compilador GHC (Glasgow Haskell Compiler)

Alguns comandos básicos:

ghci (inicializa o interpretador)
:l (carrega arquivo)
:r (recompila código)
:quit (sai do interpretador)

Exercicios

Resolva os exercícios abaixo (considerando a inferência de tipos)

1. Crie funções que retornem o dobro e o triplo de um número.
2. Faça uma função que calcule a média entre dois números.
3. Escreva uma função que receba um número e diga se ele é par ou ímpar.
4. Escreva uma função que diga se uma pessoa pode votar (idade >= 16).

Gabarito

1. ```
   dobro x = 2 * x
   triplo x = 3 * x
   ```

2. ```
   media a b = (a + b) / 2
   ```

3. ```
   parOuImpar n =  if mod n 2 == 0
     then "Par"
     else "Ímpar"
   ```

4. ```
   podeVotar idade = idade >= 16
   ```

Programação Funcional