Linguagens de Programação

Introdução ao Haskell

Samuel da Silva Feitosa

Aula 6

Programação Funcional

Motivação

- Características de linguagens funcionais estão sendo incorporadas em linguagens *mainstream*.
- Frameworks escritos em linguagens atuais usam diversos conceitos de linguagens funcionais.

Motivação

- Diversas empresas usam linguagens funcionais para desenvolver seus produtos.
- Nubank utiliza a linguagem *Closure* no desenvolvimento de praticamente todos os seus produtos.
  - Nubank comprou a empresa que desenvolve essa linguagem.

Algumas empresas que usam Haskell

- **ABN AMRO** análise de riscos financeiros
- **AT&T** automatização de processamento de formulários
- **Bank of America Merril Lynch** transformação de dados
- **Bump** servidores baseados em Haskell
- **Facebook** manipulação da base de código PHP
- **Google** infra-estrutura interna de TI
- **MITRE** análise de protocolos de criptografia
- **NVIDIA** ferramentas usadas internamente
- **Qualcomm, Inc** geração de interfaces de programação para Lua
- **The New York Times** processamento de imagens

Por que Haskell?

- Haskell é da família das linguagens funcionais.
- Linguagem funcional pura.
- Linguagem fortemente tipada.
- Avaliação *lazy*.
- Possui suporte a polimorfismo paramétrico e de sobrecarga.

Exemplo

- Realizar uma ação sobre uma coleção de elementos:

```
Iterator it = list.iterator();
while (it.hasNext()) {
    Element e = it.next();
    action(e) ; // does something
}
```

Exemplo

- Realizar uma ação sobre uma coleção de elementos:

```
map action list
```

Linguagem Pura

- Expressões em Haskell não possuem **efeitos colaterais**, por padrão.
  - Efeitos colaterais: atribuição, entrada e saída, etc.
- Expressões produzem o mesmo resultado em todas as execuções.
- Vantagens:
  - Facilita o teste e a execução concorrente.
- O mesmo não é necessariamente verdade em linguagens como Java, C, Python.

Avaliação *Lazy*

- Expressões são executadas somente quando necessárias para o resultado final.
- Quando calculadas, seu resultado pode ser reutilizado ou descartado pelo compilador.
- É um recurso poderoso, mas pode comprometer a eficiência se não for usado corretamente.

Tipagem Forte

- O sistema de tipos classifica os valores em tipos.
- Essa classificação é utilizada para detectar comportamentos indesejados em programas.
- Realiza a verificação durante a compilação.
- Haskell é uma linguagem com tipagem estática e com um sistema de tipos muito expressivo.

Polimorfismo

- Haskell provê suporte a dois tipos de polimorfismo
  - Paramétrico (*generics*)
  - De sobrecarga (*ad hoc*)
- Haskell possui um mecanismo poderoso de sobrecarga.

Primeiros passos com Haskell

Instalação

- Visual Studio Code
  - https://code.visualstudio.com/Download
  - Instalar o plugin "*Haskell Syntax Highlighting*"
- Haskell Platform (Ubuntu)
  - Utilizar o comando: **apt install haskell-platform**
  - Instala as ferramentas necessárias para desenvolver sistemas simples em Haskell
- Haskell Stack (facilita a instalação no Windows)
  - Ferramenta para gerenciar projetos e bibliotecas
  - Instala todas as dependências automaticamente (incluindo o compilador GHC).
  - https://www.haskellstack.org/

Interpretador

- No terminal, digite `ghci` e você irá obter o prompt do interpretador.

```
Prelude*>
```

- O GHCi é um REPL
  - Read, Eval, Print, Loop.
- Prelude é a biblioteca importada por todo módulo Haskell.

Exemplos

- Operações matemáticas

```
Prelude> 5 * 3
15
```

```
Prelude> 1/2 + 1/3
0.8333333333333333
```

- Funções matemáticas

```
Prelude> sqrt 7
2.6457513110645907
```

Exemplos

- GHCi não permite (por padrão) utilizar múltiplas linhas de código:

```
Prelude> 1/2 +

<interactive>:13:7: error:
    parse error (possibly incorrect indentation or mismatched brackets)
```

- Porém, é possível utilizar um bloco multi-linha:

```
Prelude> :{
Prelude| 1/2 +
Prelude| 1/3
Prelude| :}
0.8333333333333333
```

Exemplos

- Verificando o sistema fortemente tipado
  - Dividir duas *strings* produz um erro

```
Prelude> "hello" / "world"

<interactive>:23:1: error:
    • No instance for (Fractional [Char]) arising from a use of ‘/’
    • In the expression: "hello" / "world"
      In an equation for ‘it’: it = "hello" / "world"
```

- Para fechar o interpretador, você pode usar CTLR+D ou digitar o comando:

```
Prelude> :quit
Leaving GHCi.
```

Documentação das Bibliotecas

- Haskell 2010 - Language Report
  - https://www.haskell.org/onlinereport/haskell2010/
- Bibliotecas Padrão / Prelude
  - Importado automaticamente por padrão em todos os programas em Haskell.
  - https://downloads.haskell.org/~ghc/latest/docs/html/libraries/
- Hackage
  - Coleção de pacotes disponibilizados pela comunidade de desenvolvedores.
  - https://hackage.haskell.org/

Considerações Finais

- Vimos a motivação para utilizar linguagens funcionais
  - Conceitos de LF em linguagens tradicionais como Java, Python, Javascript.
  - Haskell é uma linguagem funcional pura.
- Instalação do VSCode e do Haskell-Platform e Stack (windows).
- Rodamos comandos no interpretador interativo do Haskell (GHCi).