Programação 3

Professor: Carlo Marcelo Revoredo da Silva
Contato: revoredo@gmail.com

UNIVERSIDADE DE PERNAMBUCO

Conteúdo

- Contextualização
- O Paradigma Funcional (PF)
- Prática
  - Instalação
  - Primeiros comandos
  - Exemplo
- Conclusão

Contextualização

- A programação funcional teve inicio em 1930

- Nessa década, Alonzo Church desenvolveu o cálculo LAMBDA
  - Forma matemática para representar a computação
  - Todas as funções são anônimas (não exige um nome, serve para auxiliar o código de uma outra função, resultando em funções dentro de outras funções)
  - Uma função pode ser passada como parâmetro ou retorno de outras funções

Contextualização

- A programação funcional teve inicio em 1930

- Nessa década, Alonso Church desenvolveu o cálculo LAMBDA
  - Forma matemática para representar a computação
  - Todas as funções são anônimas (não exige um nome, serve para auxiliar o código de uma outra função, resultando em funções dentro de outras funções)

- Em 1950 John McCarthy criou a linguagem de programação Lisp baseada em Lambda.

- Em 1970 Robin Milner desenvolveu a primeira linguagem de programação funcional com inferência de tipos e polimorfismo.

- Em 1980 se deu inicio ao desenvolvimento da linguagem de programação Haskell

Paradigma Funcional

- Paradigma baseado em funções
- Incentiva a imutabilidade de dados e de estados
  - Uma função sempre retornar um mesmo valor;
  - As funções não devem ter efeitos colaterais;
  - Desencorajar as mutações;
  - Problemas de referência de memória;

Mutabilidade x Imutabilidade

```
List<Aluno> alunos = new ArrayList<Aluno>();
Aluno a = new Aluno();
a.setNome(“Marcelo”);
alunos.add(a);
a.setNome(“Revoredo”); // mutável
alunos.add(a);
System.out.println(alunos); // [Revoredo, Revoredo]
```

Mutabilidade x Imutabilidade

```
List<Aluno> alunos = new ArrayList<Aluno>();
Aluno a = new Aluno();
a.setNome(“Marcelo”);
alunos.add(a);
a = new Aluno(); // resolveu o problema!
a.setNome(“Revoredo”);
alunos.add(a); // 2 itens com nomes distintos
System.out.println(alunos); // [Marcelo, Revoredo]
```

Paradigma Funcional

- Não segue na contramão com a orientação a objetos, podendo até mesmo ser combinados

- Linguagens criadas com base no paradigma funcional
  - Earling, R, Haskell, Ocaml, F#, Elixir
- Outras:
  - Java (Scala), JavaScript
- Multiparadigma:
  - Lisp

Let’s Code!

Comandos básicos

1. Instalar o HaskellPlatform-8.6.5-core-x86_64-setup.exe
2. No Windows: Iniciar > Executar
3. Digite CMD (acessar o prompt de comando)
4. Digitar: ghci
5. Praticar comandos básicos

Comandos básicos

```
sum(x,y) // função para somar
sum[1,2,3,4] // diversos números

a = [1,2,3] // define uma lista
b = [4,5,6] // define uma lista
concat [a,b] // unir 2 listas

a++[4] // adiciona um item na lista
drop 1 a // remove 1 item da lista
drop 3 a // remove 3 itens da lista
```

Comandos básicos

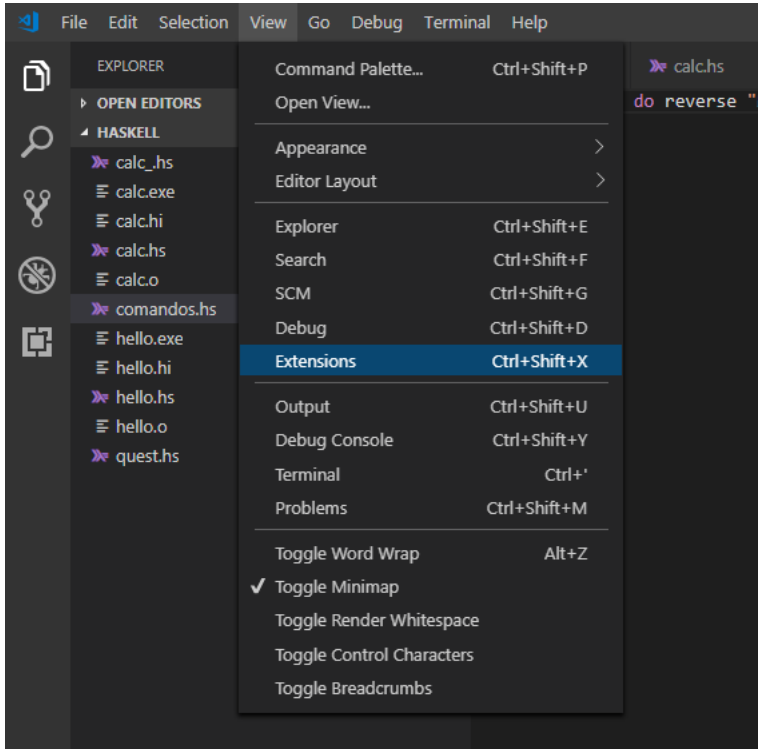
```
a = “marcelo”
reverse a // inverte uma sequência
a = [1,2,3]
reverse a // também funciona com listas
a = “marcelo”
b = “revoredo”
concat[a,” “, b]

a = sequence [“marcelo”]
```

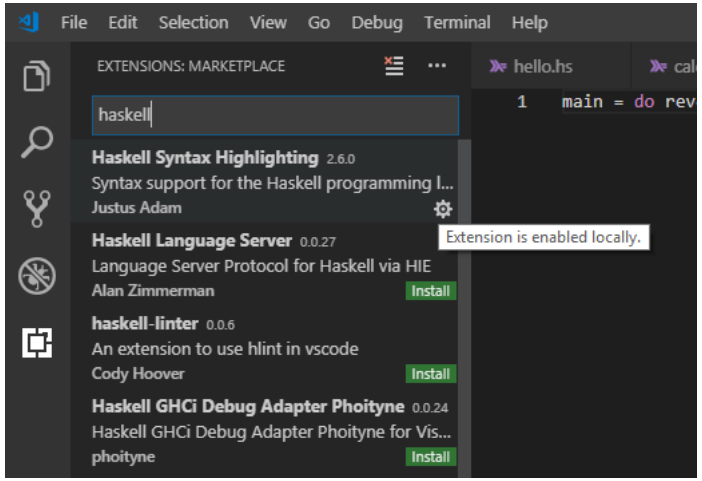
Definindo suas próprias funções

```
somar n1 n2 = n1+n2 // define uma função chamada somar
somar 2 1 // executa somar

Que tal fazer as demais operações aritméticas?
Que tal fazer uma função média?

a = [1,2,3] // define uma lista

inserir n = a++[n] // função para inserir na lista a
remover n = drop n a // função para remover na lista a
```

IDE para Haskell

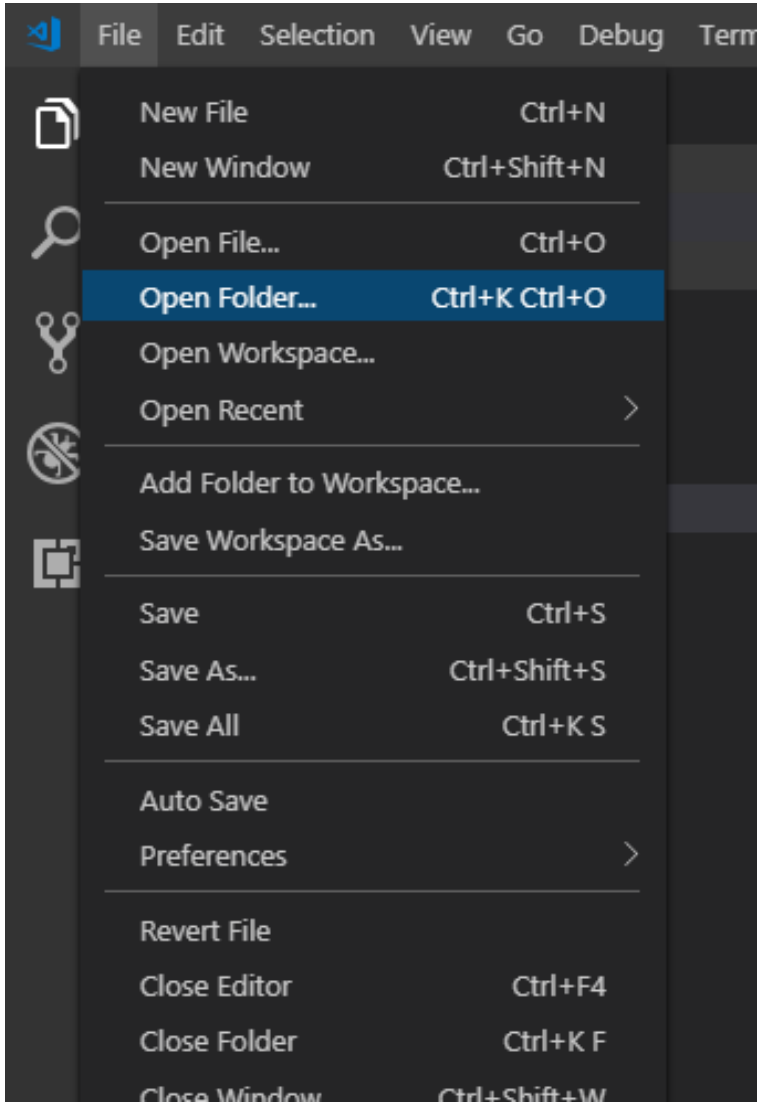
1. Instalar o VSCodeUserSetup-x64-1.34.0.exe
2. No VSCode:
   - View > Extensions (Ctrl + Shift + X)
   - Pesquisar por Haskell
   - Clicar em Haskell Syntax Highlighting
   - Clicar em Install





Preparando o VSCode

1. Crie uma pasta no sistema para armazenar seus scripts
2. No VSCode:
   - File > Open Folder...
   - Selecione a pasta que foi criada



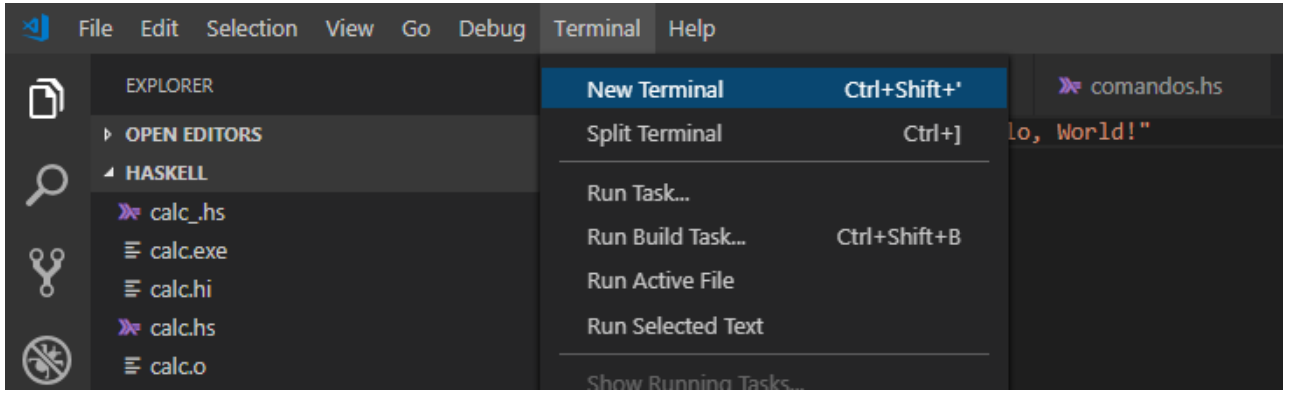
Criando e compilando...

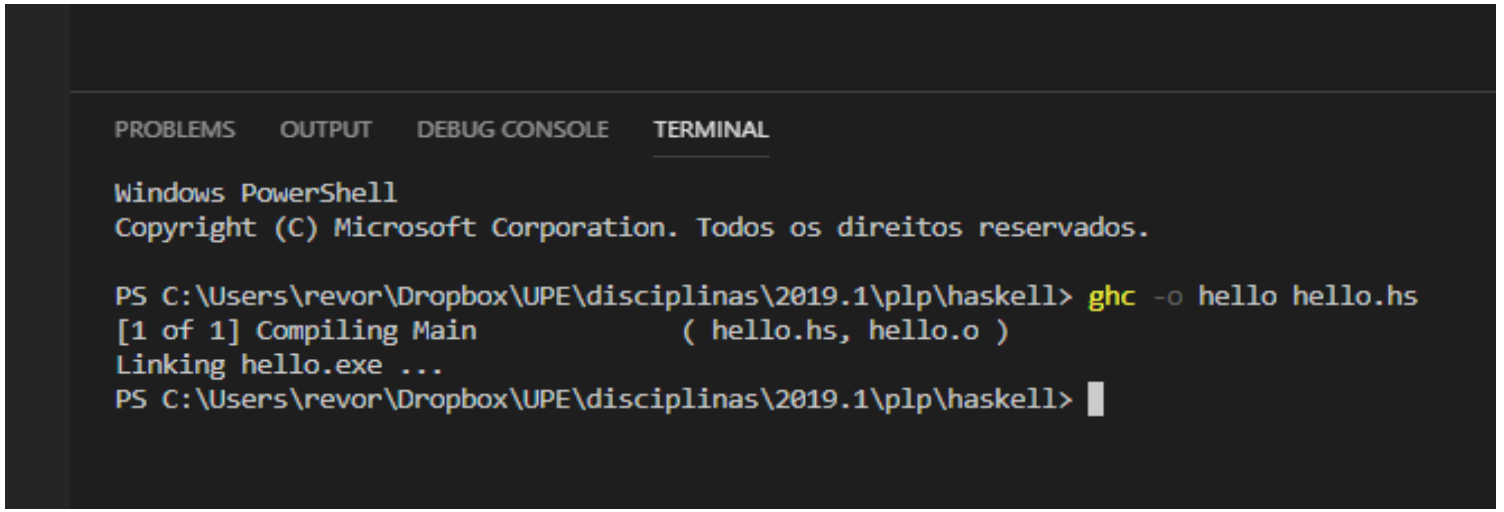
NO VSCode:

1. Crie um arquivo hello.hs
2. Conteúdo do arquivo

```
main = putStrLn "Hello, Haskell!"
```

3. Salvar (Ctrl + S)
4. Abrir o terminal:
5. Terminal > New Terminal
6. Código para compilar:
7. ghc -o hello hello.hs
8. Após isso, execute o CMD e abra o hello.exe





Exemplo de uma calculadora

```
main =
    do putStrLn "Informe o primeiro número: "
       primeiro <- getLine

       putStrLn "Informe o segundo número: "
       segundo <- getLine

       putStrLn "Informe a operação aritmética: "
       operacao <- getLine

       putStrLn "Resultado: "
```

Exemplo de uma calculadora

```
case operacao of
        -- Somar
        "+" -> do
            print((read primeiro :: Float) + (read segundo :: Float))
        -- Subtrair
        "-" -> do
            print((read primeiro :: Float) - (read segundo :: Float))
        -- Multiplicar
        "*" -> do
            print((read primeiro :: Float) * (read segundo :: Float))
        -- Dividir
        "/" -> do
            print((read primeiro :: Float) / (read segundo :: Float))
```

Referências

1. Calculo Lambda

https://github.com/Webschool-io/matematica-para-programadores/tree/master/calculo-lambda

2. Conceitos sobre imutabilidade

https://medium.com/opensanca/imutabilidade-eis-a-quest%C3%A3o-507fde8c6686

3. Palestra sobre os problemas da mutabilidade

https://youtu.be/7Zlp9rKHGD4

4. Conceitos sobre funções anônimas

https://medium.com/trainingcenter/closures-fun%C3%A7%C3%B5es-an%C3%B4nimas-e-fun%C3%A7%C3%B5es-nomeadas-f840782a486c

5. Comandos básicos no GHCI

http://downloads.haskell.org/~ghc/7.4.1/docs/html/users_guide/ghci-commands.html

Download das Ferramentas

1. Haskell

https://www.haskell.org/platform/

2. VSCode

https://code.visualstudio.com/

Grato pela atenção

Disciplina: Programação 3

Professor: Carlo Marcelo Revoredo da Silva

Contato: revoredo@gmail.com

UNIVERSIDADE DE PERNAMBUCO