INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS - *campus* BAMBUÍ

Engenharia de Computação

Disciplina: Paradigmas de Programação

Aluna: Letícia Leonel

Atividade 3 – Implementação Haskell

O código começa definindo o tipo de dado **Tarefa** como um registro que contém apenas um campo **descricao**, que representa a descrição da tarefa. Em seguida, é definido um sinônimo de tipo **ListaDeTarefas**, que é uma lista de tarefas.

```haskell
module Tarefa where

data Tarefa = Tarefa { descricao :: String } deriving (Show)
type ListaDeTarefas = [Tarefa]
```

A seguir, temos a função **adicionarTarefa**, essa função simplesmente cria uma nova tarefa com a descrição fornecida e a adiciona à lista existente, retornando a nova lista de tarefas.

```haskell
--Adicionando uma nova tarefa
adicionarTarefa :: ListaDeTarefas -> String -> ListaDeTarefas
adicionarTarefa lista descricaoTarefa = lista ++ [Tarefa descricaoTarefa]
```

A função **removerTarefa** recebe uma lista de tarefas e um índice e remove a tarefa correspondente a esse índice da lista

```haskell
--Removendo uma tarefa
removerTarefa :: ListaDeTarefas -> Int -> ListaDeTarefas
removerTarefa lista indice = take (indice - 1) lista ++ drop indice lista
```

A função **exibirListaTarefas** recebe a lista de tarefas e imprime na tela a lista formatada. Se estiver vazia, exibe uma mensagem indicando que não existem tarefas.

```haskell
--Printando as tarefas
exibirListaTarefas :: ListaDeTarefas -> IO ()
exibirListaTarefas lista = do
  putStrLn "--- Lista de Tarefas: ---"
  if null lista
    then putStrLn "Tarefas inexistentes."
    else mapM_ (putStrLn . formatarTarefa) (zip [1..] lista)
```

A função **formatarTarefa** recebe uma tupla contendo um índice e uma tarefa, e retorna uma string formatada representando a tarefa. Ela utiliza o índice e a descrição da tarefa para criar uma string no formato "índice. descrição".

```haskell
--formatando a lista
formatarTarefa :: (Int, Tarefa) -> String
formatarTarefa (indice, tarefa) = show indice ++ ". " ++ descricao tarefa
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS - *campus* BAMBUÍ

Engenharia de Computação

Disciplina: Paradigmas de Programação

Aluna: Letícia Leonel

Atividade 3 – Implementação Haskell

O programa principal é definido pela função **main**. Ela chama a função **exibirMenu** com uma lista vazia de tarefas.

```haskell
--Programa Principal
main :: IO ()
main = do
  putStrLn "Bem-vindo a sua lista de Tarefas!"
  exibirMenu []
```

A função **exibirMenu** é responsável por exibir o menu de opções ao usuário e tratar a entrada fornecida. Dependendo do comando escolhido, diferentes ações são executadas. Se o comando for "1", o usuário é solicitado a digitar a descrição da nova tarefa. Se o comando for "2", o usuário é solicitado a digitar o número da tarefa a ser removida. Se o comando for "3", exibe a lista atual de tarefas. Se o comando for "4", uma mensagem de encerramento é exibida. Se o comando não corresponder a nenhuma das opções anteriores, uma mensagem de erro é exibida. Enquanto isso, a função **exibirMenu** é chamada recursivamente com a mesma lista.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS - *campus* BAMBUÍ
Engenharia de Computação
Disciplina: Paradigmas de Programação
Aluna: Letícia Leonel
Atividade 3 – Implementação Haskell

```haskell
-- Função para exibir o menu de opções e tratar a entrada do usuário
exibirMenu :: ListaDeTarefas -> IO ()
exibirMenu lista = do
  putStrLn "Selecione a opção desejada:"
  putStrLn "1. Adicionar uma nova tarefa."
  putStrLn "2. Remover uma tarefa."
  putStrLn "3. Exibir a lista de tarefas."
  putStrLn "4. Sair."

  putStr "> "
  comando <- getLine

  case comando of
    "1" -> do
      putStrLn "Digite a tarefa:"
      descricaoTarefa <- getLine
      let novaLista = adicionarTarefa lista descricaoTarefa
      putStrLn "Tarefa adicionada com sucesso!"
      exibirMenu novaLista

    "2" -> do
      putStrLn "Digite o número da tarefa a ser removida:"
      indiceStr <- getLine
      let indice = read indiceStr :: Int
      let novaLista = removerTarefa lista indice
      putStrLn "Tarefa removida com sucesso!"
      exibirMenu novaLista
```

```haskell
    "3" -> do
      exibirListaTarefas lista
      exibirMenu lista

    "4" -> putStrLn "Lista de tarefas fechada com sucesso!"

    _ -> do
      putStrLn "Erro!"
      exibirMenu lista
```