Universidade Federal de Juiz de Fora
Departamento de Ciência da Computação
Linguagem de Programação

Bingo

2º Trabalho Prático em Haskell

Bruno de Oliveira Jucá - MAT 201965013AC

Rômulo Chrispim de Mello - MAT 201935038

Professor: Leonardo Vieira dos Santos Reis

Relatório do segundo trabalho da disciplina DCC019 - Linguagem de Programação.

Juiz de Fora

Dezembro de 2022

1 Introdução

O presente relatório tem como objetivo descrever o programa desenvolvido em Haskell para representar um jogo de Bingo. Serão apresentadas as principais decisões de projeto e as estratégias utilizadas para resolução dos problemas apresentados.

O trabalho está estruturado da seguinte maneira: a Seção 2 descreve como o problema foi modelado; a Seção 3 apresenta como cada funcionalidade foi implementada; por fim, a Seção 4 traz as conclusões e as considerações finais do trabalho.

Obs: Para compilar o programa é necessário que seja possível importar a biblioteca *System.Random*. Foi possível instalá-la através do comando:

```
cabal install --lib random
```

Posteriormente, é possível compilar e executar com:

```
ghc bingo.hs && ./bingo
```

2 Modelagem do Problema

Para modelar o problema e representá-lo internamente, foram utilizadas estruturas bem simples. O jogo de Bingo consiste em sortear um número. Os participantes precisam preencher o número em suas cartelas, caso o possuam. Ganha quem completar uma coluna, uma linha ou a cartela inteira, dependendo da definição.

O jogo precisa representar uma lista, que marca quais números já foram sorteados (de 1 a 75). Além disso, cada jogador, nesta modelagem, é representado por tuplas de tipo (*String*, *String*, [*Int*]). O primeiro valor representa o nome do jogador, o segundo o tipo de sua cartela (que pode ser "l"ou "c") e o terceiro uma lista contendo 25, sendo:

- 5 entre 1 e 15;
- 5 entre 16 e 30;
- 5 entre 31 e 45;
- 5 entre 46 e 60;
- 5 entre 61 e 75;

3 Desenvolvimento das Funcionalidades

3.1 Inicialização do Jogo

A inicialização do jogo se da por meio da função *main*. Por meio de interação com o usuário é definida a quantidade de jogadores. Posteriormente são geradas 3 listas:

- *names* - lista com o nome de cada um dos jogadores
- *cardType* - lista com o tipo de cartela de cada um dos jogadores
- *cards* - lista de 25 números, gerada pelo programa, para representar a cartela de cada jogador

A lista de *cards* é gerada por meio da função *generateCard* que por sua vez utiliza a função *generateRandomList* para gerar 5 listas de 5 valores aleatórios entre os limites especificados anteriormente. Concatenando essas listas, a função retorna os 25 valores aleatórios totais (ordenados).

A função *generateRandomList* funciona de maneira recursiva, e possui uma lista de valores já sorteados como parâmetro (inicialmente fornecida vazia), para manter registro e não repetir valores já sorteados. Desse modo a lista final retornada não possui valores repetidos.

As listas geradas são transformadas em uma lista de tuplas por meio da função *zip3* para representarem os jogadores.

Posteriormente o programa chama a função *playRound* que exibe as cartelas e já sorteia o primeiro número do jogo.

3.2 Sorteio de Novo Número

A função *playRound* é uma função recursiva que cria o *loop* principal do jogo. Ela recebe dois parâmetros: o primeiro é uma lista com os números já sorteados e o segundo e uma lista de tuplas de jogadores ([*String*, *String*, [*Int*]]).

A função já começa sorteando um novo valor, levando em conta uma lista de já sorteados, por meio da função *generateRandomNum*, que funciona de maneira análoga à *generateRandomList* já descrita. A única diferença é que é retornado somente um valor, diferente dos que foram fornecidos na lista de já sorteados.

O programa atualiza a lista de sorteados com o novo valor (*updatedCalledNumbers*) e exibe as cartelas em tela, marcando com um "X"os valores de cartela que já foram sorteados, checando com a lista de valores já sorteados.

3.3 Finalização do Jogo

O jogo deve terminar quando um dos jogadores completar uma linha ou uma coluna, de acordo com sua escolha no início.

Para realizar a verificação, as cartelas dos jogadores são mapeadas para a função *checkBingo* junto com a lista de valores já sorteados.

Com base no tipo de cartela a função define o tipo de verificação que será realizado. As verificações são feitas de forma igual depois que as cartelas são divididas em colunas e linhas. Caso a interseção de alguma das linhas ou colunas com os números já chamados tenha tamanho 5, significa que aquela coluna ou linha foi completada, portanto a função retorna *True*. Caso contrário retorna *False*.

Para dividir a cartela em linhas foram utilizadas as funções *take* e *crop* da maneira necessária.

Para dividir a cartela em colunas foi feito um *zip* com uma lista enumerada para representar o índice e dessa forma foram obtidas as posições necessárias por meio do módulo da divisão do índice.

Por fim, filtrando a lista gerada pelo mapeamento, resta o ganhador, se houver algum. Foi utilizada uma lista para que o programa identifique caso haja mais de um ganhador simultâneo.

Caso não haja ganhador, *playRound* é executada novamente, após o *input* do usuário, tendo como parâmetro, agora, a lista atualizada de valores sorteados. Isso continua ocorrendo até que haja algum ganhador.

4 Conclusões e considerações finais

Este relatório apresenta como foi desenvolvido o trabalho requisitado pela disciplina para a produção de um jogo de Bingo em Haskell. O programa apresentou o comportamento esperado, pôs em prática o conhecimento adquirido em sala de aula e, aparentemente, simula com sucesso um jogo de Bingo.