Universidade Federal de Pelotas
Curso de Engenharia de Computação
Disciplina: 22000350 – Tópicos Especiais em Computação IX
Turma: 2020/2 – M1
Professores: Andre Rauber Du Bois

Trabalho Final: Liga4

Aluno: Vinícius Renato Rocha Geraldo. Data: 18/05/2021
Matrícula: 16104664.

1 Introdução

No trabalho final para disciplina de programação na linguagem Haskell precisamos desenvolver um jogo de tabuleiro relacionando questões estudadas apresentada durante o semestre. Nesse projeto iremos realizar uma descrição do jogo Liga4 demonstrando partes do código descrito, junto do funcionamento do programa para ser executado e replicado. Esse relatório está dividido da seguinte forma havendo uma discussão sobre o trabalho, apresentando partes do código que são de cunho essencial para o funcionamento do jogo, e por fim uma breve conclusão do trabalho utilizando a linguagem Haskell.

2 Discussões

O código implementado para disciplina necessitava conter abstrações do conteúdo apresentado em aula e havendo um desafio para ser realizado durante a criação do algoritmo que descreva o objetivo do trabalho. Assim, foi desenvolvido o jogo de tabuleiro chamado Liga4, da empresa Hasbro, no qual se baseia em um clássico jogo de jogo da velha, porém havendo o desafio de ter que vencer o jogo contendo 4 peças da mesma cor, tais quais apresentadas como na Figura 1. Vemos que no jogo somos capazes de vencer nas posições verticais, horizontais, e nas diagonais para todos as direções de esquerda e direita onde são jogadas que precisamos tratar no jogo desenvolvido.

No desenvolvimento do código inicialmente fazemos a descrição do tabuleiro usando a função *replicate*() passando como entrada um valor inteiro para criar uma matriz do tabuleiro, no nosso jogo como o tabuleiro consta de posições ($6x7$) inicialmente apresenta esse formato ao jogador. Dessa forma, podemos acessar a matriz utilizando apenas as colunas como parâmetro quando fazemos a interação com o jogo, e havendo uma facilidade no momento de atualizar posições para apresentar ao jogador um novo tabuleiro com a posição modificada com sua respectiva peça. Assim, vamos analisar como funciona o menu do jogo para inicialização do jogo como apresentado na Figura 2, onde conseguimos esse acesso quando executamos o comando *main* para chamar a inicialização da bibliotecas de IO associadas em Haskell. Nessa tela vemos que somos apresentados a quatro comandos que são possíveis de executar no terminal para as chamadas das funções a serem executadas, assim precisamos escrever da mesma maneira que está descrita a função no console para executar da maneira correta. Somos capazes de editar o formato do jogo para conter um tabuleiro maior, ou até a quantidade de jogadores que teremos na partida.

Na reformatação do tabuleiro utilizamos o comando $TamanhoTabuleiro\#\#$ passando dois argumentos do tipo inteiro para gerar um novo tabuleiro com o tamanho passado como parâmetro, e na função $QuantosJogadores\#$ podemos modificar a quantidade de jogadores que terá na partida quando executar a função $ComecarJogo$. A configuração inicialmente é feita com o padrão do jogo para dois jogadores, e quando começamos o jogo temos a seguinte apresentação do terminal na Figura 3. São apresentados comandos possíveis para cada jogador para posicionar uma nova peça no tabuleiro utilizando a função $Posicaox$, onde "x" é a coluna que deseja colocar a peça para passar ao próximo jogador. Com a programação utilizando códigos ANSI, conseguimos fazer a distinção da peça do jogador através das cores que foi inicialmente populada com os valores na função $simboloTabuleiro$, onde somos capazes de escrever de maneira colorida no terminal os caracteres programados.

Na seguinte tela quando o jogo é começado, podemos posicionar para qual jogador designado inicialmente e assim ir realizando as jogadas consecutivamente para cada jogador. Podemos posicionar em qualquer lugar das colunas apresentadas apenas descrevendo a função $Posicaox$, e sua posição

desejada, assim posicionando a cor da peça no lugar. Na Figura 4 vemos uma possível solução do jogo para um vencedor, e somos apresentados a um novo menu com o propósito de reiniciar um novo jogo com a função $TenteOutraVez$ ou voltamos para o menu a sua função designada.

3 Conclusões

Portanto, somos capazes de descrever diferentes modelos de códigos utilizando as funcionalidades presentes na abstração da linguagem Haskell. De tal forma, podemos solucionar diferentes problemas divididos em submódulos como descrito nesse trabalho para fazer cada etapa do projeto, e assim gerar um resultado final como apresentação de um jogo de tabuleiro. Nesse tipo de linguagem trabalhos de maneira mais amigável com o código, sendo capaz de interpretar trechos existentes e descrever suas funções assim de forma recursiva, ou apenas utilizando as funções de alta ordem apresentadas na disciplina. O código está comentado com as respectivas funcionalidades, e espero que se divirta com o jogo.

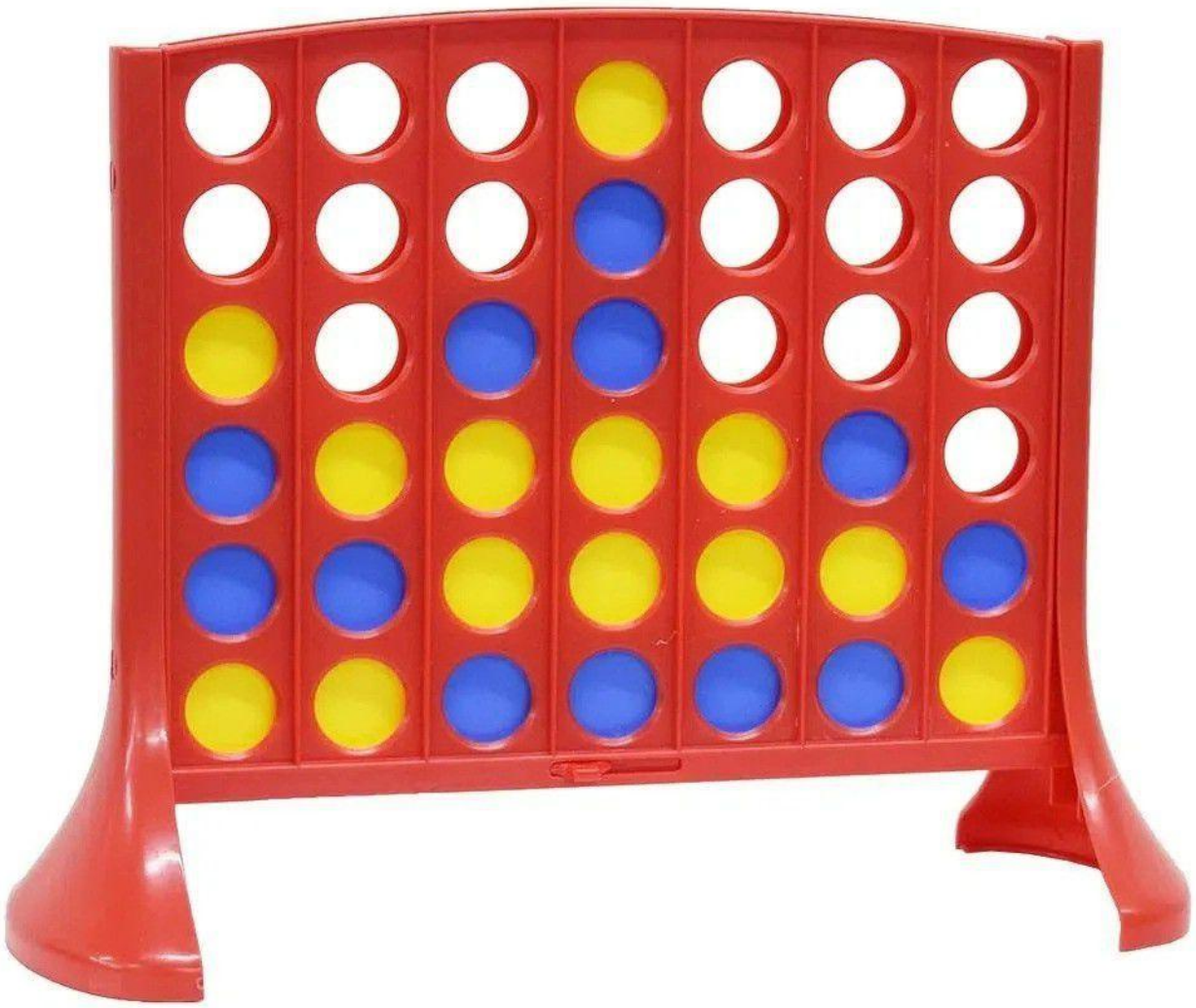

Figura 1: Tabuleiro do jogo

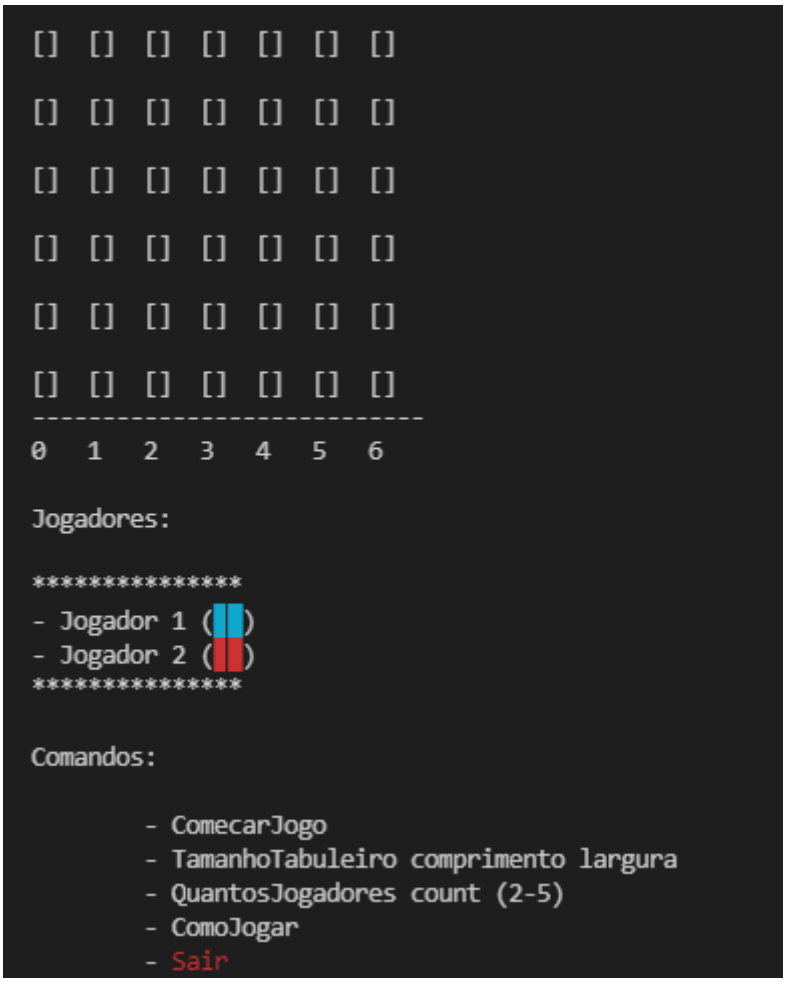

Figura 2: Menu do jogo

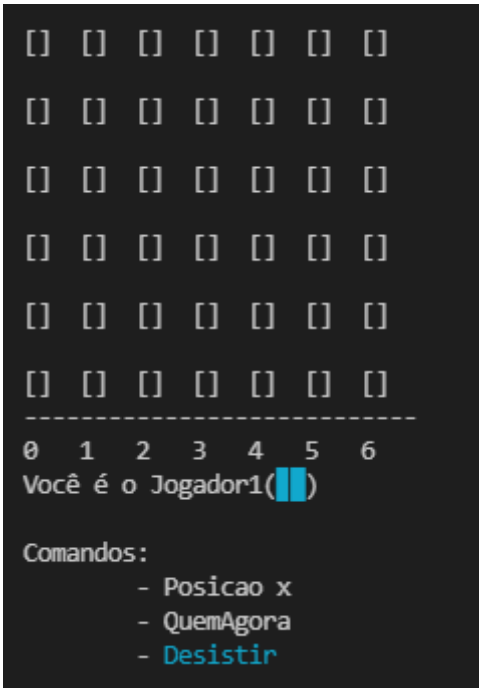

Figura 3: Tela do tabuleiro com o jogo em funcionamento

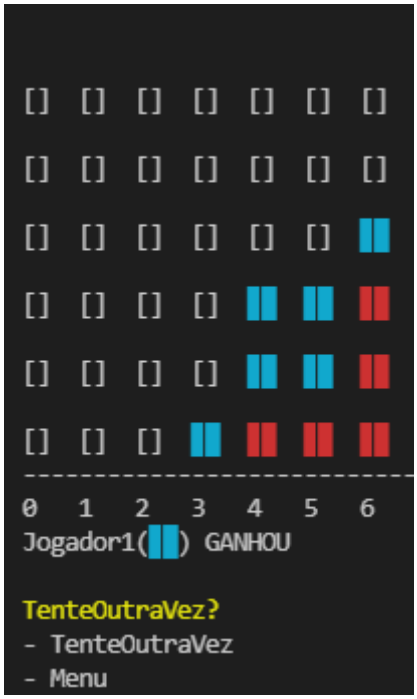

Figura 4: Tela do vencedor