Linguagem de Programação

Alunos: Aloízio Pita de Castro Júnior - 201365006C
Vinícius de Oliveira Corbelli - 202065093C
Professor: Leonardo Reis

Sumário

- O Problema: Campo minado
- Estratégia adotada
- Implementação da solução

O Problema: Campo minado

Foi proposta na disciplina de Linguagens de Programação (01/2023 - UFJF) a elaboração de uma versão do jogo Campo Minado em Haskell. A implementação desse jogo em Haskell permite explorar os conceitos e recursos dessa linguagem funcional, como imutabilidade, recursão e listas. O desafio consiste em desenvolver um programa capaz de gerar um tabuleiro com posições aleatórias para as minas, permitindo ao jogador interagir com o jogo, abrindo posições e marcando-as como minas. O objetivo principal é criar uma experiência de jogo divertida e desafiadora, mantendo a lógica e as regras do Campo Minado tradicional. Durante o processo de implementação, serão explorados conceitos como manipulação de listas, estruturas de dados, funções de alta ordem e recursão para a resolução de problemas específicos do jogo.

Estratégia Adotada

O desenvolvimento do trabalho pode ser separado em duas etapas:

- Implementação de funções básicas;
- Implementação dos testes e regras do jogo.

Chamamos de funções básicas aquelas que formam a base do campo minado, como na criação do tabuleiro, interação com o usuário, impressão do tabuleiro, sorteio de posições e inclusões das bombas no tabuleiro e a atualização do tabuleiro quando ocorrer uma alteração realizada pelo usuário.

Já os testes e regras do jogo são as funções que garantem que o campo minado se comporte como apresentado no enunciado, com os testes necessários de vitória, derrota, jogadas inválidas, etc.

Ficou definido que o tabuleiro seria uma matriz com cada posição tendo um elemento char, que é impresso na tela para o usuário, e um elemento boolean, oculto, que guarda a informação da posição ter ou não uma bomba.

Implementação da Solução

O jogo começa com a função main sendo chamada. Ele imprime uma mensagem de boas-vindas e, em seguida, obtém o tamanho do tabuleiro e a quantidade de bombas do jogador por meio das funções getSizeTab e getSizeBomb, respectivamente.

As funções getSizeTab e getSizeBomb são responsáveis por obter o tamanho do tabuleiro e a quantidade de bombas do jogador. Elas usam getLine para receber a entrada do usuário e, em seguida, validam a entrada para garantir que ela esteja dentro dos limites adequados.

A função createBoard é usada para criar um tabuleiro vazio, preenchido com asteriscos (*) e definindo cada célula como não contendo uma bomba.

A função generateBombs é usada para gerar aleatoriamente as posições das bombas no tabuleiro. Ela garante que as posições das bombas sejam únicas e, em seguida, atualiza o tabuleiro com as bombas.

A função printBoard é usada para imprimir o tabuleiro atual. Ela oculta as bombas e exibe asteriscos (*) para as células ainda não abertas. A função printColumnLetters imprime as letras das colunas acima do tabuleiro e printIndexedRow imprime cada linha do tabuleiro com seus índices.

Implementação da Solução

Após definir essas funções, a função playGame é chamada, e o jogo começa. Ela exibe as opções disponíveis para o jogador e aguarda a entrada. Dependendo da opção selecionada (+ para marcar, - para desmarcar e uma letra de coluna seguida por um número de linha para abrir), a função realiza a ação correspondente no tabuleiro. O jogo continua até que o jogador vença ou perca.

Durante o jogo, a função playGame monitora constantemente o estado do tabuleiro e atualiza as informações exibidas ao jogador, garantindo que ele esteja ciente do andamento da partida. Além disso, ela valida as entradas do jogador, verificando se estão dentro dos limites e se correspondem às opções disponíveis.

Caso o jogador decida marcar uma célula com o símbolo "+", a função playGame verifica se a célula selecionada está vazia e, em caso afirmativo, a marcação é feita. Por outro lado, se o jogador optar por desmarcar uma célula com o símbolo "-", a função verifica se a célula selecionada já está marcada e, caso esteja, remove a marcação.

Já no caso de o jogador optar por abrir uma célula específica, representada por uma letra de coluna seguida por um número de linha, a função playGame realiza diversas verificações. Ela verifica se a célula selecionada está fechada e se a coordenada fornecida é válida dentro dos limites do tabuleiro. Em seguida, ela realiza a abertura da célula selecionada, revelando seu conteúdo.

Implementação da Solução

Enquanto o jogo prossegue, a função playGame verifica constantemente se o jogador atingiu a condição de vitória, verificando se todas as células não minadas foram abertas. Em caso de vitória, ela encerra o jogo e exibe uma mensagem de parabéns ao jogador. Porém, se o jogador abrir uma célula minada, a função playGame encerra o jogo e revela todas as minas presentes no tabuleiro, exibindo uma mensagem de derrota.

Dessa forma, a função playGame desempenha um papel crucial no fluxo e na dinâmica do jogo, garantindo uma interação suave entre o jogador e o tabuleiro. Através dela, o jogador pode explorar e marcar as células do tabuleiro, levando-o à vitória ou à derrota em sua busca pela descoberta das células seguras.