Trabalho Prático 1 de PM

Tabuleiro de Xadrez

Douglas Rodrigues de Almeida
douglasralmeida@live.com

1. Introdução

O objetivo do trabalho foi a criação de um programa que simulasse os movimentos da peça cavalo em um tabuleiro de xadrez.

O cavalo possui movimentos limitados no tabuleiro. A partir de uma posição (x, y), ele somente poderá assumir oito posições possíveis, descritas no quadro abaixo:

(x+1,y+2), (x+2,y+1), (x+2,y-1), (x+1,y-2), (x-1,y-2), (x-2,y-1), (x-2,y+1) e (x-1,y+2).

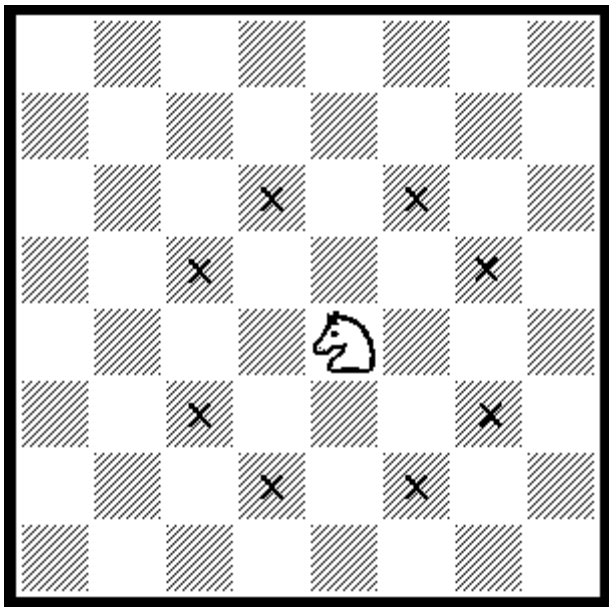

Figura 1: Movimentos possíveis do cavalo

O programa realiza duas simulações: a primeira simula os movimentos do cavalo a partir de uma posição inicial aleatória até a uma posição de onde não é mais possível movimentar-se. A segunda simulação soluciona o desafio de movimentar o cavalo em todas as casas do tabuleiro passando em cada casa uma única vez.

2. Implementação

Classes e Módulos

O programa foi dividido em classes e módulos, dentre eles.

Peca: A interface Peca representa uma peça qualquer do tabuleiro. Ela possui as seguintes funções:

- getMovimento: Obtém um movimento possível a partir de um ID.
- getPosicao: Obtém a posição atual da peça

- getQtMovimento: Obtém a quantidade de movimentos possíveis de uma peça.
- setPosicao: Altera a posição atual da peça.

Cavalo: A classe Cavalo herda a classe Peca e representa uma peça Cavalo do tabuleiro. Ela possui todas as funções da classe pai e possui os seguintes atributos:

- Movimentos[]: Vetor que armazena todos os movimentos possíveis do cavalo.
- Posicao: Ponto (x, y) que armazena a posição atual do cavalo no tabuleiro.

Funções:

- getMovimento: Obtém um movimento possível a partir de um ID.
  - Coesão: Realiza sua função de forma independente.
  - Acoplamento: Não chama nenhuma outra função. É chamado por duas funções diferentes.
- getPosicao: Obtém a posição atual da peça
  - Coesão: Realiza sua função de forma independente.
  - Acoplamento: Não chama nenhuma outra função. É chamado por três funções diferentes.
- getQtMovimento: Obtém a quantidade de movimentos possíveis de uma peça.
  - Coesão: Realiza sua função de forma independente.
  - Acoplamento: Não chama nenhuma outra função. É chamado por duas funções diferentes.
- setPosicao: Altera a posição atual da peça.
  - Coesão: Realiza sua função de forma independente.
  - Acoplamento: Não chama nenhuma outra função. É chamado por três funções diferentes.

Tabuleiro: A classe Tabuleiro representa o tabuleiro de xadrez. Ela possui os seguintes atributos:

- altura: Armazena a altura do tabuleiro.
- largura: Armazena a largura do tabuleiro.
- numvisitas: Armazena o número de visitas realizadas no tabuleiro.
- casas[][]: Matriz que representa o estado das casas do tabuleiro. Cada elemento da matriz indica o momento que a casa foi visitada.

Além dos atributos acima, a classe Tabuleiro possui as seguintes funções:

- ehPosicaoOcupavel: Informa se uma casa do tabuleiro pode ser visitada.
  - Coesão: Realiza sua função de forma independente.
  - Acoplamento: Não chama nenhuma outra função. É chamado por uma função.

- foiPercorrido: Informa se o tabuleiro já foi percorrido.
  - Coesão: Depende de outras funções para realizar sua função.
  - Acoplamento: Chama duas funções diferentes. É chamado por uma função.
- getAltura: Informa a altura do tabuleiro.
  - Coesão: Não depende de outras funções para realizar sua função.
  - Acoplamento: Não chama nenhuma outra função. É chamado por quatro funções diferentes.
- getLargura: Informa a largura do tabuleiro.
  - Coesão: Não depende de outras funções para realizar sua função.
  - Acoplamento: Não chama nenhuma outra função. É chamado por quatro funções diferentes.
- imprimir: Imprime o tabuleiro na tela.
  - Coesão: Depende de outras funções para realizar sua função.
  - Acoplamento: Chama três funções diferentes. É chamado por uma função.
- imprimirDetalhes: Imprime o número de visitas já realizadas no tabuleiro.
  - Coesão: Depende de outras funções para realizar sua função.
  - Acoplamento: Chama uma função. É chamado por uma função.
- inserir: Insere uma nova peça no tabuleiro.
  - Coesão: Depende de outras funções para realizar sua função.
  - Acoplamento: Chama uma função. É chamado por duas funções diferentes.
- movimentar: Movimenta uma peça do tabuleiro.
  - Coesão: Depende de outras funções para realizar sua função.
  - Acoplamento: Chama três funções diferentes. É chamado por duas funções diferentes.
- podeMovimentar: Informa se um movimento específico de uma peça pode ser executado.
  - Coesão: Depende de outras funções para realizar sua função.
  - Acoplamento: Chama cinco funções diferentes. É chamado por duas funções diferentes.

Simulador: A classe Simulador realiza as simulações no tabuleiro. Ela possui três atributos:

- tab: Objeto da classe Tabuleiro que representa o tabuleiro da simulação.
- tempoInicial: Armazena o tempo inicial da simulação.
- tempoTotal: Armaena o tempo gasto pela simulação.

Além dos atributos, a classe Simulador possui as seguintes funções:

- escolheProxMovAleatorio: Escolhe aleatoriamente um movimento para uma peça daqueles possíveis.
    - Coesão: Depende de outras funções para realizar sua função.
    - Acoplamento: Chama seis funções diferentes. É chamado por uma função.
- imprime: Imprime o resultado da simulação com o estado final do tabuleiro, a quantidade de visitas realizadas e o tempo gasto pela simulação em milissegundos.
    - Coesão: Depende de outras funções para realizar sua função.
    - Acoplamento: Chama quatro funções diferentes. É chamado por uma função.
- passear: Realiza um passeio completo no tabuleiro por uma peça visitando todas as casas do tabuleiro uma única vez.
    - Coesão: Depende de outras funções para realizar sua função.
    - Acoplamento: Chama sete funções diferentes, incluindo ela mesma. É chamado por uma função.
- sortearInicio: Sortea uma posição do tabuleiro.
    - Coesão: Depende de outras funções para realizar sua função.
    - Acoplamento: Chama três funções diferentes. É chamado por uma função.
- simula: Simula um passeio aleatório de uma peça no tabuleiro a partir de uma posição inicial aleatória, passando em cada casa uma única vez até uma posição final, de onde não é possível movimentar.
    - Coesão: Depende de outras funções para realizar sua função.
    - Acoplamento: Chama seis funções diferentes. É chamado por uma função.
- solucionaDesafio: Soluciona o desafio do trabalho, realizada um passeio completo no tabuleiro a partir da posição (0, 0) e passando em cada casa uma única vez.
    - Coesão: Depende de outras funções para realizar sua função.
    - Acoplamento: Chama quatro funções diferentes. É chamado por uma função.

Algoritmos

O programa possui dois algoritmos para solucionar o problema da movimentação das peças.

Movimentação Aleatória: Este algoritmo é usado para movimentação aleatória de uma peça no tabuleiro.

```
coloca a peça P numa posição aleatória do tabuleiro
equanto a peça P pode ser movimentada
    escolhe um movimento possível aleatoriamente
    movimenta a peça P
```

Passeio Completo: este algoritmo de tentativa e erro utiliza a técnica de *backtracking* para fazer um passeio completo de uma peça no tabuleiro visitando todas as casas do tabuleiro uma única vez. Ele utiliza a recursividade para tentar todas as alternativas possíveis até encontrar uma solução válida.

O processo pode ser descrito como uma árvore de pesquisa exaustiva, cujos nós correspondem a soluções parciais do problema. Caminhando para as folhas da árvore, corresponde a obter soluções parciais na direção da solução final; caminhar em direção a raiz corresponde a *backtracking* para alguma solução parcial geral obtida anteriormente, a partir da qual seja interessante prosseguir em direção às folhas novamente.

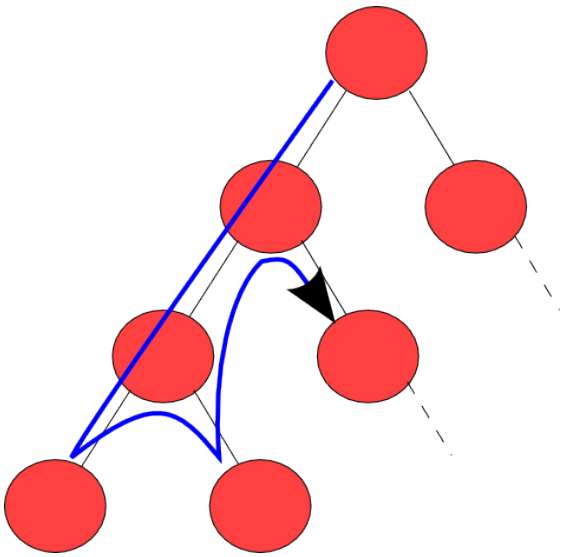

Figura 2: Árvore de pesquisa exaustiva

Descrição do algoritmo:

```
registra P na posição (0, 0) do tabuleiro
tenta
    repete
        seleciona próximo candidato ao movimento
        se movimento é possível então
            movimenta
            se tabuleiro está cheio então
                encerra algoritmo
            do contrário
                chama tenta recursivamente
                se movimento não foi sucedido então
                    apaga registro anterior
    até (encerrar algoritmo) ou (acabarem candidatos a
movimento)
```

3. Avaliação e Resultados

Foram feitas dezenas de testes do programa. Durante estes testes foi verificado que a posição inicial da peça interfere no tempo de duração do programa e na quantidade de visitas usando a simulação aleatória. Não foi possível analisar o tempo de duração por ser muito curto. Todos duraram entre dois e três milissegundos.

Nos testes realizados na solução do desafio, a posição inicial também interferiu no resultado. Dependendo da casa inicial, o desafio demora muito tempo para ser executado.

4. Pesquisa sobre coesão e acoplamento

Coesão Lógica

Um módulo possui coesão lógica quando executa uma série de operações relacionadas, uma das quais é selecionada pelo módulo que faz a chamada.

Ex: Um módulo que executa todas as funções de entrada e saída.

Coesão Temporal

Um módulo tem coesão temporal quando ele realiza uma série de operações relacionadas com o tempo.

Ex: Um módulo chamado abrir_arquivo_e_iniciar_tabela_vendas_e_ler_primeiro_registro_arquivo.

Coesão Processual

Um módulo possui coesão procedural quando ele realizada uma série de operações relacionadas pela sequência de etapas a serem cumpridas pelo produto.

Ex: Um módulo chamado ler_registro_arquivo_e_atualizar_arquivo_de_manutencao.

Coesão Sequencial

Um módulo possui coesão sequencial quando a saída de um elemento serve de entrada para o próximo elemento.

Coesão Funcional

Um módulo possui coesão funcional quando ele realizada exatamente uma operação ou atinge um único objetivo.

Exs: obter_temperatura; escrever_no_disco.

Acoplamento de conteúdo

Dois módulos são acoplados por conteúdo se um deles fizer referência direta ao conteúdo do outro.
Ex: O módulo p modifica uma instrução do módulo q.

Acoplamento comum

Dois módulos apresentam acoplamento comum quando ambos tiverem acesso aos mesmos dados globais.

Acoplamento de controle

Dois módulos apresentam acoplamento de controle quando um módulo controlar explicitamente a lógica do outro módulo.

Ex: Módulo a manda uma mensagem para módulo B. Módulo B usa um parâmetro da mensagem para decidir o que fazer.

Acoplamento de carimbo

Dois módulos apresentam acoplamento de carimbo se uma estrutura de dados for passada na forma de um argumento, porém, o módulo é chamado a operar apenas sobre alguns dos componentes individuais dessa estrutura.

Ex: Em C, passa-se como argumento um ponteiro para um registro para operar apenas em alguns de seus campos.

Acoplamento de dados

Dois módulos apresentam acoplamento de dados se todos os argumentos forem dados homogêneos, ou seja, todo argumento é um argumento simples ou uma estrutura de dados na qual todos os elementos são usados pelo módulo chamado.

5. Referências

SCHACH, S. R. 2009. Engenharia de Software: Os Paradigmas Clássico e Orientado a Objetos. 7ª Edição, AMGH Editora.

ZIVIANI, N. 2009. Projeto de Algoritmos com Implementações em Pascal e C, 2ª Edição, Editora Cengage Learning.