

Nome:

Turma:

Professor: Giuliano Paes Carnielli

Data:

Avaliação Prática de Estrutura de Dados: Projeto Jogo de Xadrez

Instruções:

1. Esta atividade deve ser desenvolvida em grupos de 5 ou 6 alunos.
2. Todo grupo deve ter um nome único.
3. A entrega deve ser feita, por email, como um pacote ZIP contendo um projeto do Code::Blocks.
4. O título do email deve ter o formato: [ED] - Projeto Pratico <1/2> - <nome do grupo>
5. Um trabalho estará sujeito a anulação imediata caso: não compile, não apresente minimamente as funções esperadas.
6. Os nomes de todos os membros do grupo devem aparecer no cabeçalho de arquivos de código e no corpo do email usado para entregar a avaliação.
7. O código deve ser organizado e bem documentado, com o risco de decréscimo na nota.
8. O plágio ocasionará a anulação de todas as avaliações envolvidas.
9. **BOM TRABALHO.**

1. Jogo de Xadrez:

Desenvolveremos um Jogo de Xadrez simplificado, usando conceitos de Estrutura de Dados como: matrizes, ponteiros, TADs, Listas, Pilhas, entre outros.

O projeto deve ser construído como uma aplicação-console e, portanto, nenhum suporte gráfico será requisitado. Embora os grupos estejam livres para usar recursos gráficos, fica esclarecido que essa decisão não será, de forma alguma, considerada na avaliação.

2. Tipos Abstratos:

O programa deve possuir, minimamente, as seguintes estruturas e funções. Os detalhes são deixados para que os grupos decidam, mas tudo o que for sugerido abaixo, deve efetivamente fazer parte do projeto.

Todas as entidades do programa devem ser implementadas como TADs, ou seja, blocos funcionais contendo a Estrutura de Dados que define a entidade, e o conjunto de funções que determina as operações da entidade. Todas as definições (estruturas e assinaturas de funções) devem ser colocadas em um arquivo *header* (.h). A implementação das funções deve estar em um arquivo .c com o mesmo nome do .h.

TAD Jogo:

Estrutura que armazena o estado do jogo em andamento.

Estrutura:

- Título ou nome do Jogo
- Referência (ponteiro) para o Tabuleiro (descrito a baixo)
- Lista de Peças
- Jogada em andamento (qual peça, para onde: critério do grupo como modelar a jogada)
- Turno: indica de qual jogador é a vez (Pretas ou Brancas)

Nome:

Turma:

Professor: Giuliano Paes Carnielli

Data:

- Mensagem: considerando que a estrutura jogo deve ser passada para quase todas as funções do programa, este atributo pode facilitar o envio de mensagens para o display. Bastaria copiar a mensagem para este atributo, e depois, a função responsável por imprimir a interface do jogo pode ler e imprimir também essa informação.

Outros elementos que pertençam ao jogo podem ser mapeados nesta estrutura, como por exemplo, os jogadores. Critério do grupo.

Funções:

i. Função criaJogo():

Função que instancia e inicializa uma nova estrutura Jogo. Deve ser responsável por criar as peças e o tabuleiro, e qualquer outro elemento pertencente ao jogo. Deve ainda inicializar o tabuleiro, colocando as peças em suas posições de partida.

ii. Função display():

Função responsável por redesenhar a interface. Pode ser a própria função main, e pode dividir-se em outras sub-funções:

- Desenhar o Tabuleiro
- Tratar entrada do usuário

Abaixo, uma sugestão de como a interface deve ser impressa:

Kasparov x Deep Blue

		---	---	---	---	---	---	---	---	
8	T	H	B	Q	K	B	H	T	---	
7	P	P	P	P	P	P	P	P	P	
6	---	---	---	---	---	---	---	---	---	
5	---	---	---	---	---	---	---	---	---	
4	---	---	---	---	---	---	---	---	---	
3	---	---	---	---	---	---	---	---	---	
2	p	p	p	p	p	p	p	p	p	
1	t	h	b	q	k	b	h	t	---	
	A	B	C	D	E	F	G	H	---	

: -

Digite uma jogada (ex. 7e 6e) ou <ENTER> para sair
Branças (minúsculas):

Nome:

Turma:

Professor: Giuliano Paes Carnielli

Data:

iii. Função executaJogada():

Função que recebe a entrada tratada do usuário e executa a movimentação apropriada. Uma jogada deve possuir a forma "2d 4d", onde "2d" indica a coordenada de origem (posição da peça a ser movimentada), e "4d" a coordenada destino. Lembrando que uma coordenada é composta do número da linha, e da letra referente à coluna.

Trata-se de uma função complexa e que deve se decompor em várias tarefas. Algumas das tarefas que devem ser tratadas estão descritas abaixo:

- Verificar se as coordenadas são válidas.
- Verificar se existe uma peça na coordenada de origem;
- Verificar se a cor da peça corresponde à vez da jogada;
- Realizar a movimentação da peça no tabuleiro;
- Verificar se existe peça na posição destino:
- Se for do adversário, registrar a captura;
- Se for do próprio jogador, impedir a jogada

TAD Tabuleiro:

Estrutura que representa o tabuleiro do Xadrez. Usado para armazenar a situação do jogo e como base para a impressão da interface.

Estrutura:

- Matriz 8x8 de Peças (ou outra representação equivalente)

Funções:**i. Função criaTabuleiro():**

Construtor que instancia e inicializa um Tabuleiro.

SUGESTÃO: tornar o Tabuleiro mais flexível, permitindo que a situação inicial das peças seja passada por parâmetro, ao invés de fixar a distribuição. Isso permitiria, por exemplo, salvar um jogo em arquivo e, depois, recarrega-lo, instanciando um novo tabuleiro com as peças nas posições salvas.

ii. Função getCasa():

Função que retorna a Peça contida na casa indicada (coordenadas x, y), ou **NULL**, caso a casa esteja vazia.

iii. Função setCasa():

Função que coloca uma Peça na casa indicada. Precisa tratar a situação da casa já estar ocupada. O grupo deve definir como esse tratamento será feito.

TAD Peça:

Peça deve representar uma peça do Xadrez.

Nome:

Turma:

Professor: Giuliano Paes Carnielli

Data:

Estrutura:

- Nome: descrição da peça (ex. Bispo da Rainha). Usado apenas para acrescentar informações à interface.
- Símbolo: indica como a peça é representada no tabuleiro. Deve ser um único caractere (char) e seguir o mapeamento abaixo:
 - Torre: 't'
 - Cavalo: 'h'
 - Bispo: 'b'
 - Rainha: 'q'
 - Rei: 'k'
 - Peão: 'p'
 - OBS: peças brancas serão representadas por caracteres minúsculos, e as peças pretas, por caracteres maiúsculos.

SUGESTÃO: um "Enum" pode ser usado para mapear a lista de símbolos.

- Lado: indica se a peça é branca ou preta. Apesar de ser possível deduzir isso a partir do símbolo, este atributo facilita esse tipo de verificação.

SUGESTÃO: usar "enum" para mapear Brancas e Pretas.

- Linha, Coluna: coordenadas da peça no tabuleiro.
- movePeca: ponteiro para função específica de movimento. Cada peça possui um tipo diferente de movimento. A sugestão aqui é criar funções de validação para cada tipo de peça (mantendo a mesma assinatura), e durante a criação da peça, fazer o ponteiro movePeca apontar a função adequada.

Grupos podem adotar outra abordagem, mas deve validar a alternativa com o professor.

Funções:

i. Função movePeca():

Cada peça possui um tipo diferente de movimento. Este método deve ser específico para cada tipo de peça e deve validar se o movimento pretendido é possível.

ATENÇÃO: os diferentes tipos de peças devem ser criados como funções independentes e relacionados a cada tipo de estrutura Peca por meio de um ponteiro para funções (ver atributo movePeca).

Nome:

Turma:

Professor: Giuliano Paes Carnielli

Data:

3. Primeira Etapa:

Na primeira etapa do projeto, válida para o primeiro bimestre, os grupos deverão entregar o projeto Jogo de Xadrez, contendo as seguintes funcionalidades:

1. Criação de um novo jogo
2. Permitir movimentar peças usando o esquema de coordenadas (e.x. 2d 4e)
3. Validar os movimentos corretamente (de acordo com as validações discutidas em ExecutaJogada).
4. Validar e executar capturas de peças
5. Validar o final do jogo, quando um Rei é capturado.

As seguintes jogadas complexas NÃO serão incluídas:

1. Roque
2. "En passant"
3. Promoção de peças que atinjam o limite oposto do tabuleiro

Nome:

Turma:

Professor: Giuliano Paes Carnielli

Data:

4. Apêndice A: regras de movimentação das peças

Cada peça no Xadrez possui um tipo de movimentação específico. Nas descrições abaixo, considere a seguinte notação para movimento de peças:

- lin_i : linha de origem de uma peça
- col_i : coluna de origem de uma peça
- lin_d : linha de destino de uma peça
- col_d : coluna de destino de uma peça

Considere que apenas o Cavalo pode “pular” outras peças. Logo, na validação do movimento, é necessário verificar se não existem obstáculos.

Torre:

Descrição: A torre se movimenta para em qualquer sentido, quantas casas quiser (sem pular nenhuma outra peça), em uma única direção: Linha ou Coluna.

Verificação:

$$(|lin_i - lin_d| == 0 \text{ OU } |col_i - col_d| == 0)$$

Explicação: o movimento é válido se o deslocamento em uma das direções (Linha ou Coluna) é zero. Note, entretanto, que se for zero nas duas direções, não houve movimento, e isso não pode ser considerado uma jogada válida.

Bispo:

Descrição: O bispo se movimenta em qualquer sentido, quantas casas quiser (sem pular nenhuma outra peça), mas sempre na diagonal.

Verificação:

$$(|lin_i - lin_d| == |col_i - col_d|)$$

Explicação: o movimento é válido se o deslocamento em uma das direções é proporcional ao movimento da outra direção (diagonal)

Cavalo:

Descrição: O cavalo tem um movimento especial que lembra a letra L. O cavalo se movimenta 2 casas para frente ou para trás e em seguida 1 casa para a direita ou para a esquerda, ou 2 casas para a direita ou para a esquerda e em seguida 1 casa para frente ou para trás. O cavalo é a única peça do xadrez que pode pular outras peças.

Verificação:

$$((|lin_i - lin_d| == 2 \text{ E } |col_i - col_d| == 1) \text{ OU } (|lin_i - lin_d| == 1 \text{ E } |col_i - col_d| == 2))$$

Nome:

Turma:

Professor: Giuliano Paes Carnielli

Data:

Explicação: o movimento é válido se o deslocamento envolve duas casas em uma direção e uma casa em outra.

Rainha:

Descrição: A Rainha, também conhecida como Dama, é a peça mais poderosa do xadrez, ela pode ir para frente ou para trás, para direita ou para a esquerda, ou na diagonal, quantas casas quiser, mas não pode pular nenhuma outra peça.

Verificação:

Movimento igual ao do Bispo OU Movimento igual ao da Torre

Explicação: a Rainha é a peça que reúne as possibilidades de movimento de um Bispo e uma Torre.

Rei:

Descrição: Move-se apenas uma casa em qualquer direção.

Observação: não pode ir para uma casa que esteja sob ataque (discutiremos se esta regra será ou não implementada no projeto).

Verificação:

$$(|lin_i - lin_d| \leq 1 \text{ E } |col_i - col_d| \leq 1)$$

Explicação: o movimento é válido se o deslocamento for no máximo 1 em qualquer direção. Note, entretanto, que se for zero nas duas direções, não houve movimento, e isso não pode ser considerado uma jogada válida.

Peão:

Descrição: O peão só se movimenta para frente, sendo a única peça que não se move para trás. No primeiro lance de cada peão ele pode avançar 1 ou 2 casas. A partir do segundo lance de cada peão ele irá movimenta-se apenas 1 casa.

O Peão pode, ainda, mover-se uma casa na diagonal, desde que seja para capturar uma peça adversária.

Verificação (complexa):

Branças:

```
(1º Movimento E
  ( ( |col_d - col_i| == 0 E lin_d - lin_i == 2 ) OU
    ( |col_d - col_i| == 1 E lin_d - lin_i == 1 E Captura ) )
) OU
(!1º Movimento E
  ( lin_d - lin_i == 1 OU ( |col_d - col_i| == 1 E Captura ) ) )
```

Pretas: trocar (lin_d - lin_i) por (lin_i - lin_d)