UNIVERSIDADE FEDERAL DE SÃO CARLOS

PROGRAMAÇÃO ORIENTADA A OBJETOS

PROFª KATTI FACELI

Nomes: Jesimiel Efraim Dias
Jhonata Campos Santana

PROJETO DE XADREZ

1 - DESCRIÇÃO DA TAREFA

A ideia da fase 1 foi implementar tudo que fosse possível nas classes específicas sem considerar a relação com a classe Peca que é essencial para o funcionamento completo do nosso projeto de Xadrez.

Fazer a implementação parcial das classes Tabuleiro, Jogo e Jogador foi um pouco desafiador em vista de que as três classes citadas acima tem implementações dependentes uma das outras e da classe Peca.

2 – CLASSES PEÇA ESPECÍFICAS

- As classes peças específicas possuem os atributos:
  - bool capturada: Assume o valor falso quando não foi capturada e verdadeiro caso foi capturada.
  - const bool cor: Assume verdadeiro ou falso dependendo da cor da peça, repare que essa classe é de extrema importância, por exemplo, o peão branco só pode subir e o preto apenas descer, ou seja, ter o controle da cor é essencial até para a movimentação das peças.
- Métodos das classes peças específicas:
  - Cavalo(bool color): Esse é o nosso construtor, esse parâmetro color é quem inicializará o atributo cor.
  - bool checaMovimento(int linhaOrigem, int colunaOrigem, int linhaDestino, int colunaDestino): Esse método retorna verdadeiro caso a movimentação da peça

que se encontra em determinada posição da matriz seja válida ou falso caso o contrário.
    - void captura(): Quando uma peça é capturada temos de mudar o atributo capturada para verdadeiro.
    - bool getCapturada(): Retorna o estado atual da peça, verdade se está capturada ou falso caso contrário.

3 – CLASSE POSIÇÃO

- A classe posição possui por enquanto o atributo:
    - char peca: Como não temos o relacionamento com a classe Peca na fase 1 improvisamos um atributo char peca que recebe um carácter para a respectiva posição que pode ser tanto um carácter associado a uma peça específica quanto um espaço vazio para indicar que a posição não contém peça.
- Método da classe posição:
    - Posicao(): Construtor que inicia o objeto peca com a posição vazia, ou seja, espaço em branco.
    - void setPeca(char p): Esse método é um set que serve para trocar a peça que está na posição (até o momento mudando apenas o carácter do atributo).
    - char getPeca(): Retorna o char da peça específica associado aquela posição (também pode ser o vazio).

4 – CLASSE TABULEIRO

- A classe Tabuleiro possui por enquanto os atributos:
    - static const int tamanho: É o atributo que faz a matriz ter tamanho 8 X 8, inicialmente tentamos colocar ele como apenas constante, porém por exigência do compilador fomos obrigados a, também, deixá-lo estático.
    - Posicao tab[tamanho][tamanho]: É um atributo matriz de objeto Posicao e será esse atributo o responsável por simular um tabuleiro.
- Métodos da classe Tabuleiro:
    - Tabuleiro(): Construtor que inicia os elementos da matriz com suas peças referentes ao início do jogo do xadrez.

- void imprimirTabuleiro(): Imprimi o tabuleiro atual, por exemplo, a cada mudança nas posições das peças esse método deve ser chamado para atualizar a visualização do usuário.
- void iniciarTabuleiro(): coloca as peças no lugar do início do jogo.

5 – CLASSE JOGADOR

- A classe Jogador possui por enquanto os atributos:
  - string nome: Recebe o nome do jogador.
  - const bool cor: Cor das peças que o jogador irá jogar.
- Métodos da classe Jogador:
  - Jogador(bool color, string n): Construtor que recebe a cor e o nome do jogador.
  - Jogador(const Jogador &j): Construtor de cópia foi necessário para poder inicializar os atributos do tipo Jogador na classe Jogo.
  - string getNome(); Retorna o nome do jogador.

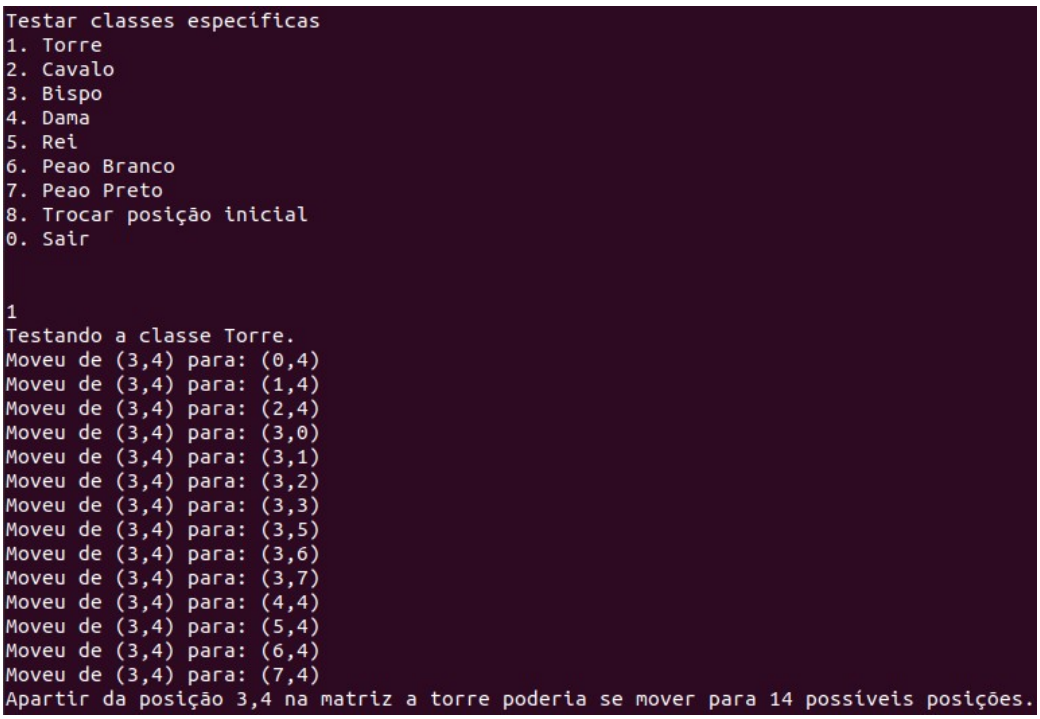
6 – CLASSE JOGO

- A classe Jogo possui por enquanto os atributos:
  - Jogador jogador_0, jogador_1: Esses atributos são os jogadores do jogo.
  - Tabuleiro tab: Esse atributo é importante, pois é com ele que teremos acesso ao nosso tabuleiro.
  - bool vez: Esse atributo assume falso se for a vez do jogador_0 e verdadeiro caso for a vez do jogador_1.
  - int estado: Esse atributo é basicamente o controle da nossa classe Jogo, ela assume, 0 se for início do jogo, 1 meio do jogo, 2 se algum jogador está em xeque e 3 se algum jogador recebeu xeque-mate.
- Métodos da classe Jogo:
  - Jogo(Jogador nome_jogador0, Jogador nome_jogador1): O construtor da classe Jogo recebe dois parâmetros do tipo Jogador que serão responsáveis por inicializar os atributos do tipo Jogador através do construtor de cópia.
  - void estadoDoJogo(): Esse método imprimi o estado: inicio do jogo, meio do jogo, também, com a combinação dos atributos estados e vez podemos saber se está e quem está em xeque e xeque-mate.
  - void mudarVezDoJogador(): Altera o atributo vez com seu valor oposto para indicar a a mudança de vez.

- void setEstado(int e): Troca o valor do atributo estado através do parâmetro.

7 – TESTE DAS CLASSES ESPECÍFICAS

- Para testar as classes específicas foi criado o mainPeca.cpp:
  - compilação: “g++ Torre.cpp Bispo.cpp Cavalo.cpp Dama.cpp Rei.cpp Peao.cpp mainPeca.cpp -o prog”
  - O mainPeca contém as variáveis: “ocorrenciaCorretas” que é responsável por saber quantas movimentações a peça pode realizar a partir de uma posição origem, “opc” que é responsável pelo controle do switch que apresenta um menu onde a partir dele é possível testar o método checaMovimento(…) de todas as classes que é o mais sensível a erros, e, linhaOrigem e colunaOrigem que possuem (pode ser mudado) respectivamente os valores 3 e 4 que são associados a posição (3,4) da matriz.
  - Aqui está o menu para testar as classes peças específicas, ao digitar, por exemplo, a opção “1. Torre” (1) iremos receber as possíveis posições que a peça torre pode ir a partir da posição (3,4) da matriz considerando as outras 63 posições livres.
  - Observe que os demais métodos (com exceção do construtor que foi testado ao declara o objeto) não foram testado no mainPeca, pois os métodos desenha, captura e getCaptura além de serem triviais e não terem utilidade nessa fase implica que testar tais métodos tão simples seriam apenas sobrecarregar o mainPeca que já está complexo o suficiente, porém, adianto dizer que testei separadamente e estão totalmente funcionais.



8 – TESTE DAS CLASSES TABULEIRO, JOGO E JOGADOR

- Para testar a classe Tabuleiro, Jogo e Jogador foi criado o mainGeral.cpp:
  - compilação: “g++ Jogo.cpp Jogador.cpp Tabuleiro.cpp Posicao.cpp mainGeral.cpp -o prog”.
  - O mainGeral.cpp contém a variável/objeto: “opc” responsável pelas opções de um menu, “jogador_0(0,"Jesimiel")” e “jogador_1(1,"Jhonata") “ objetos do tipo jogador que recebem no seu construtor a cor de suas peças e seu nome, e, o objeto Jogo que recebe como parâmetro no construtor os objetos criados anteriormente “jogo(jogador_0,jogador_1)”.
  - O menu para testar os métodos das classes Tabuleiro, Jogo e Jogador, como podemos ver, já recebemos a impressão de um tabuleiro ao executar o main onde as peças pretas são maiúsculas e cinzas e as peças brancas minúsculas com a cor padrão, como a variável estado no construtor foi inicializada com 0, quando chamamos o método estadoDoJogo() é exibido a mensagem “Inicio do jogo”, e se apertarmos, por exemplo, a opção 1 iremos para o “meio do jogo” e será exibido a mensagem “Vez do jogador Jesimiel!”, por padrão (tanto nas regras quanto no construtor) ele é o branco, ou seja, o primeiro.
  - Esse menu, também, permite testar o método mudarVezDoJogador() através da opção 4 que troca o valor do atributo vez para o seu oposto indicando a vez do outro jogador, quando feito isso, o estadoDoJogo() irá identificar que foi mudado pelo atributo vez e o estado do jogo será feito em cima do outro jogador, ou seja, o estadoDoJogo() funciona orientado pelos atributos estado e vez.