UNIVERSIDADE ESTADUAL DE MONTES CLAROS

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS – CCET

Patrick Pierre Fernandes Ferreira

Messias Ferreira Muniz

**JOGO OTHELO EM C**

Montes Claros, Minas Gerais

Março de 2011

Patrick Pierre Fernandes Ferreira

Messias Ferreira Muniz

**JOGO OTHELO EM C**

Trabalho apresentado ao professor Renato, como parte das exigências para avaliação da disciplina de Algoritmos e programação I, 1º período.

Montes Claros – MG

Março/2011

**Introdução**

Reversi foi um jogo criado na Inglaterra por volta do século XIX para dois jogadores. Consiste em um tabuleiro de 8x8, totalizando 64 casas, com peças pretas e brancas (aqui simbolizadas por 1 e 2). O jogo inicia-se com cada um dos jogadores com duas peças de sua cor no tabuleiro, e tem como objetivo obter uma maior pontuação antes que os dois jogadores fiquem sem jogadas válidas, ou que o tabuleiro seja completamente preenchido. Os jogadores alternam as jogadas, colocando sempre a sua peça em uma casa vazia e em uma posição que cerque as peças do adversário, se isso não for possível, ele perde a jogada. Sempre que uma peça ou grupo de peças seqüenciais do adversário são cercadas, o jogador que as cercou “ganha” suas posições, substituindo-as pelas peças de sua cor. Quando o jogo termina, ganha o jogador que tiver mais peças no tabuleiro.

Este trabalho tem como objetivo desenvolver um jogo reversi utilizando a linguagem de programação C. Quando o jogo é iniciado, pede como dados de entrada as posição (linha coluna) em que o jogador deseja jogar. É analisada então a possibilidade da jogada, para determinar se ela é válida ou não. Se a jogada for válida, é incrementado o numero do jogador na matriz (tab[8][8]), e nas casas entre ela e a próxima com o mesmo número. Assim sucessivamente até que ambos fiquem sem jogadas válidas ou que todas as casas sejam preenchidas.

**Implementação**

Nosso modelo de reversi foi desenvolvido a partir de testes. Primeiro a matriz (tab[8][8]) recebe “0” em todas as posições, para que não receba valores aleatório, é então dado “inicio a contagem” dos clocks do processador e após a matriz recebe nas determinadas posições os valores iniciais do jogo.

O jogo entra então em um loop, tendo como condição a falta de entradas válidas para os dois jogadores. A validade é determinada por um “for” que testa as casas do tabuleiro, verificando se existe uma casa em que possa ocorrer a inserção de elementos na tabela. Se esse não existir possibilidade de jogada para o determinado jogador, o valor da variável que indica a validade da jogada não é alterada, o que impede que ele entre na sua rodada, pulando assim para o próximo jogador, o mesmo acontece com o próximo jogador. O jogo é impedido de continuar quando os indicadores mostram os dois jogadores como não tendo mais jogadas válidas.

As entradas dos jogadores são então verificadas, se a casa estiver vazia, elas entram em um loop onde são testadas todas as possibilidades de aquela jogada ser válida, primeiro checando posição por posição em torno da casa escolhida. Se existir uma peça do jogador adversário, ele entra em um loop para chegar se, ao final do conjunto de peças do adversário, existe uma peça sua, caso premissa seja verdadeira, ele acrescenta a “val” o valor “1”. Após é verificado o valor de “val”, se ele for igual a “1”, são acrescentados na matriz o número representante do jogador entre essas casas, se não, “nval” é incrementado.

|  |  |  |
| --- | --- | --- |
| Linha+1,Coluna-1 | Linha+1,Coluna | Linha+1,Coluna+1 |
| Linha,Coluna-1 | Linha,Coluna **\*** | Linha,Coluna+1 |
| Linha-1,Coluna-1 | Linha-1,Coluna | Linha-1,Coluna+1 |

**\***posição escolhida

Os testes são realizados nas 8 direções possíveis, se a validade for diferente de zero, indica que existe um valor representado pelo adversário em alguma das direções e que essa jogada pode ser executada. Os valores são então substituídos pelo valor representantes do jogador que escolheu a posição. Se o nval for igual a 8, vai indicar que ao redor da casa escolhida não existe nenhuma possibilidade de jogada, imprimindo na tela “Jogada inválida!”, e fazendo com que o “val” continue valendo “1”, fazendo com que ele volte para o loop, pedindo para o jogador inserir a jogada de novo.

Poderíamos pegar como exemplo a entrada do primeiro jogador como sendo “2 3”, lembrando que o primeiro numero é o indicador de linha, e o segundo o de coluna:

Tabela antes da primeira jogada:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |
| 3 |  |  |  | 2 | 1 |  |  |  |
| 4 |  |  |  | 1 | 2 |  |  |  |
| 5 |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |

É verificado então a possibilidade da jogada:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |
| 2 |  |  |  | \* |  |  |  |  |
| 3 |  |  |  | 2 | 1 |  |  |  |
| 4 |  |  |  | 1 | 2 |  |  |  |
| 5 |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |

\*casa escolhida possibilidades invalidas possibilidade valida





|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |
| 2 |  |  |  | 1 |  |  |  |  |
| 3 |  |  |  | 1 | 1 |  |  |  |
| 4 |  |  |  | 1 | 2 |  |  |  |
| 5 |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |

Após o termino do jogo, é realizado uma contagem da quantidades de peças que cada jogador possui no tabuleiro, e é impresso o nome do jogador que tiver obtido maior pontuação e o tempo do jogo.

Descrição das variáveis criadas na implementação:

**val (int)**: testa todas as possibilidade de a jogada ser válida em relação a posição de peças;

**val1 e val2 (int)**: foi usada em um loop para verificar se a jogada é ou não válida, somente quando a casa colhida for válida val1 e val2 mudam de valor;

**jpoin1 e jpoint2 (int)**: contagem de pontos no final da execução;

**l e c (int):** entrada de coordenadas (l para linhas e c para colunas);

**nval (int)**: indica a não validade de cada possibilidade de jogada, se nval, ao final dos testes for igual a 8, a jogada é inválida;

**escolha (int)**:variável utilizada no menu para escolher que o jogador possa escolher se deseja iniciar uma partida ou ver as instruções do jogo;

**temp\_t (Double)**:variável criada para contar o tempo a partir do clock do processador;

**jog1[10] e jog2[10] (char)**: Variaveis criadas para receberem os nomes dos jogadores.

**Conclusões**

Encontramos grandes dificuldades em relação à conclusão deste trabalho, principalmente em relação a como fazer para que o programa acrescentasse na tabela somente nas posições possíveis de jogada. O trabalho foi desenvolvido da maneira que achamos mais prática e que provavelmente não dariam tantas linhas de código, para que o programa não ficasse tão extenso. Outra grande dificuldade encontrada foi em relação a tempo, não referindo a prazos, mas a horários, já que ambos da dupla ainda estudam e trabalham.

Nosso modelo de Reversi ainda apresenta alguns erros que não tivemos tempo para corrigir, mas a lógica e linha de raciocínio utilizado para chegar ao resultado foi a mesma descrita acima.