UNIVERSIDADE DE ÉVORA

CURSO DE ENGENHARIA INFORMÁTICA

Relatório Prático de Programação I

João Fernandes nº32409
João Silveirinha nº42575
Janeiro 2022

Índice

Introdução

Para o âmbito da disciplina de Programação I, foi-nos pedido para desenvolver um programa, em linguagem C, que nos permitisse jogar Othello com o computador, respeitando as respetivas regras do jogo.

O Othello é um jogo de tabuleiro de estratégia, onde o objetivo é acabar o jogo com maior número de peças no tabuleiro que o adversário.

O tabuleiro tem início com 4 peças centrais, duas brancas “O” e duas pretas “X”. As restantes casa do tabuleiro devem conter o caracter ‘.’. As peças pretas começam sempre primeiro.

A aplicação deve permitir jogar quer, com um tabuleiro inicial, quer com um tabuleiro obtido através de um ficheiro de texto que contem as respetivas jogadas.

No início, no modo interativo, é mostrado o tabuleiro apenas com as peças iniciais, depois durante o jogo, após cada jogada é mostrado um novo tabuleiro, já com as jogadas implementadas e as peças do adversário “viradas”.

O ficheiro tem uma jogada por linha, com um número e uma letra, que correspondem, à linha e à coluna, respetivamente.

No final j do jogo é mostrada a pontuação final.

Funções usadas

Inicialmente foi criada uma variável global indicadora do tamanho do tabuleiro: “SZ”, com o valor 8.

Para a elaboração deste programa foram criadas as seguintes funções:

- void init_board(char board[SZ][SZ]):

- Esta função tem como objetivo de inicializar o tabuleiro com as respetivas peças inicias “O” e “X” e as casas vazias “.”.

- As variáveis “i” e “j” servem, respetivamente, para denominar a linha e a coluna.

- Os dois ciclos “for” são utilizados para percorrer o tabuleiro (matriz), e a igualdade lá inserida para colocar ‘.’ em todas as posições (casas).

- Posteriormente são inseridas as peças iniciais no centro do tabuleiro.

- void print_board(char board[SZ][SZ]):

- Tem como objetivo, para além de mostrar o tabuleiro dito como normal no terminal, mostrar também as linhas e colunas numeradas de 1 a 8 e de “a” a “h”, respetivamente.

- As variáveis “i” e “j” servem, respetivamente, para denominar a linha e a coluna.

- É feito um ciclo “for” que vai incrementando a variável “letra_da_coluna” inicializada a 0, para ser somada à variável “coluna”, inicializada com ‘a’, com o objetivo de mostrar as letras de ‘a’ a ‘h’ (letra_da_coluna vai de 0 a SZ-1).

- Posteriormente são criados dois ciclos “for”, onde no primeiro é mostrado o valor de “i” em casa uma das iterações do ciclo, para criar a coluna nos números de 1 a 8. No segundo ciclo, correspondente às colunas, inserido dentro do primeiro ciclo (para criar a matriz), é feito o print do tabuleiro.

- void jogadas_a_zero(char board [SZ][SZ], int moves[][SZ]):

- Função auxiliar para colocar/inicializar o array “moves” a zero, para mais tarde, caso haja uma jogada válida, este “marcá-la” como válida, passando a 1.

- As variáveis “line” e “col”, correspondem à linha e coluna, respetivamente.

- int oponente(char color):

- Função auxiliar referente ao oponente.

- Função que indica quem é a jogar. Se o jogador atual for “O”, o seu oponente é “X”, caso contrário o oponente é “O”.

- Função utilizada o operador ternário “?”, com a seguinte estrutura: **“**Condição ? verdadeiro : falso”, ou seja, para o programa em questão: o jogador atual é “O”?, se sim, o seu oponente é “X”, se não, “O”.

- int find_squares_not_valid(char board[SZ][SZ], int line, int col):

- Uma das funções auxiliares para verificar se a jogada é válida.

- Esta função, ao percorrer o tabuleiro com os ciclos “for”, verifica se a posições em questão são posições (casas), já ocupadas por alguma peça.

- Se sim, continua a “verificação” até encontrar uma posição com o caracter ‘.’ (casa vazia).

- Não referente às diagonais.

- As variáveis “line” e “col”, são correspondentes à linha e coluna, respetivamente.

- int flanked(char board[SZ][SZ], int moves[][SZ], char color):

- Uma das funções auxiliares para verificar se a jogada é válida.

- Função que completa a função acima descrita, que serve, não só para verificar as posições que não são validas por já estarem ocupadas (ao fazer uma chamada da mesma).

- Verifica também as diagonais de cada uma das posições com os “for” que fazem as respetivas variáveis referentes a estas mesmas diagonais variarem entre -1 e 1.

- Por exemplo, quando estas variáveis tomam o valor 1, referem-se a “baixo-direita” (se a peça estiver em [linha] [coluna], vai verificar [linha +1 e coluna + 1].

- De seguida, verifica, se a pesquisa por posições validas não ultrapassa os limites do tabuleiro e garante que se trata sempre da peça em questão.

- Se a verificação se confirmar e encontrar uma peça do adversário, move-se em delta direção, flanqueando (sem virar, isto é, fazendo uma espécie de ponte) (a função play vai ser responsável pela mudança da peça) a peça do adversário até encontrar uma casa vazia. Se não encontrar, procura noutra direção (não há jogada possível nessa direção).

- Se tudo se concretizar existe uma jogada válida e o array “moves” passa a 1, aumentando assim o contador “n_movimentos”.

- As variáveis “line” e “col”, são referentes à linha e coluna respetivamente. As variáveis “delta_line” e delta_col” são responsáveis pelas diagonais, “i” e “j” são variáveis responsáveis por guardar e mudar a posição da peça durante a procura por posições válidas para a jogada. A variável “n_movimentos”, inicializada a 0, serve como um contador para guardar o número de jogadas válidas

- int jogadas_validas(char board[SZ][SZ], int moves[][SZ], char color):

- Função que faz a chamada de todas as funções auxiliares à verificação de uma jogada válida.

- Chama também a função referente ao oponente.

- int play(char board[SZ][SZ], int line, int col, char color):

- Função com um raciocínio idêntico às funções que validam as jogadas, fazendo todo o raciocínio de igual forma até ser colocada a peça.

- Quando a peça é colocada, verifica todas as peças que flanqueou nessa jogada e vira-as, fazendo um decremento em “delta_line” e delta_col”.

- Esta função usa as mesmas variáveis das funções que validam as jogadas.

- int count_pieces(char board[SZ][SZ], char color):

- O computador começa o jogo com 100 pecas (valor ridículo).

- Ao percorrer o tabuleiro, com os “for”, o número de peças do computador diminui, e as peças do jogador aumentam.

- No final a função retorna o número de peças.

- As variáveis “line” e “col” são referentes às linhas e colunas, respetivamente. A variável “pecas” serve para armazenar o número de peças.

- int temp_board(char board[SZ][SZ]):

- Função auxiliar à função que verifica a melhor jogada e à função determinada para fazer a jogada do computador.

- Esta função cria um tabuleiro temporário.

- As variáveis “i” e “i” dizem respeito às linhas e colunas respetivamente. A criação da nova variável “new_board[][]”, serve para guardar o tabuleiro original num tabuleiro temporário.

- int not_arrayMoves(int line, int col, int moves[][SZ]):

- Função auxiliar à função que verifica a melhor jogada.

- Verifica todas as jogadas válidas para encontrar a melhor, se a jogada não for válida, continua a verificar.

- A utilização do array “moves” com a negação permite saber se se trata de uma jogada que seja válida.

- As variáveis “line” e “col” são referentes às linhas e colunas, respetivamente.

- int jogada_melhor_pontuada(char board[SZ][SZ], int moves[][SZ], char color):

- Nesta função chamam-se todas as funções auxiliares acima descritas, onde já foi criado um tabuleiro temporário e já foram feitas as verificações.

- É chamada a função play(), para a jogada ser feita, nesse momento, no tabuleiro temporário.

- Verifica os pecas com que fica nessa jogada.

- Se ficar com mais, atualiza o número de peças e atualiza a função count_pieces.

- A variável “atualizar_pecas” foi criada com o intuito de guardar o novo número de peças após a jogada ser efetuada.

- void jogada_pc(char board[SZ][SZ], int moves[][SZ], char color):

- Esta função vê todas as jogadas válidas, volta a ser feito um tabuleiro temporário e faz a respetiva jogada nele.

- Depois é chamada a função que verifica a validade das jogadas, para que o computador faça uma jogada válida.

- Posteriormente, vê o número de peças para a melhor jogada e se o adversário (color), ficar com menos peças no tabuleiro do que aquelas que tinha antes da jogada do computador, guarda esse número de peças e guarda essa jogada em [linha_com_mais_turns][coluna_com_mais_turns].

- Por fim, faz a jogada, nessas coordenadas.

- Para isso foi necessário a criação de uma variável temporária para as jogadas, para ir procurando qual é aquela que "retira" mais peças do tabuleiro ao adversário.

- int print_num_pieces(char board[SZ][SZ]):

- Função criada para printar o número final de peças do computador e do jogador.

- As variáveis "pc_pieces" e "jogador_pieces" são inicializadas a 0 e são somadas ao tabuleiro final de forma a obter o número final de peças de cada um.

- void input_output(char board[SZ][SZ]):

- Função responsável pelo input do jogador e pelo output dado pelo computador.

-Se for a vez do jogador, são pedidas as coordenadas, e no caso de a jogada ser inválida, serem pedidas novas coordenadas. No caso de não ser possível jogar, é passada a vez.

- Se o input for correto chama a função play() e faz a jogada.

- Para a leitura do input são criadas 2 variaveis: "x" e"y". "y" é lido como um char e através da função "tolower(y) - 'a' ", é transformado no respetivo no inteiro através de código ASCII.

- No caso de ser o vez do computador, é chamada a função "jogada_pc", que faz a jogada do computador.

- Por fim, é chamada a função que diz o número de peças final de cada um.

- int main(int argc, char*argv[]):

- Função onde são declaradas as variáveis que já foram descritas anteriormente e as variáveis necessárias para a leitura do ficheiro de texto, tais como, "linhas_ficheiro[64]", "colunas_ficheiro[64]", "file".

- É declarada a variável com o nome que se deu ao ficheiro "FILE *ficheiro".

- Consoante o input no terminal, é escolhida a forma como se joga, ou interactivamente, ou através do ficheiro de texto.

- Se for escolhido jogar com o ficheiro de texto, este é aberto.

- Enquanto o ficheiro não chega ao fim são lidas as coordenadas.

Problemas encontrados

Com o decorrer do desenvolvimento do programa foram encontrados alguns problemas que não permitiram o total funcionamento do programa consoante o que foi pedido no enunciado.

No que ao modo interativo diz respeito, o programa mostra o tabuleiro, lê as coordenadas inseridas e faz a respetiva jogada, quer do jogador, quer do computador.

No final, tal como é pedido, mostra o número de peças total do jogador e do computador.

O problema aqui encontrado diz respeito à forma como o programa foi estruturado, o que não nos permitiu encontrar uma solução para o primeiro jogador ser escolhido de forma aleatória. Apenas conseguimos fazer de forma a ser, escolhido à priori, o primeiro jogador, pois quando tentámos utilizar uma função que fizesse a escolha de forma aleatória (srand), o programa deixava de funcionar. Pelo que decidimos e estruturámos o programa de forma que o computador jogue sempre primeiro.

No que toca à leitura do ficheiro, conseguimos abrir e ler o conteúdo do ficheiro, mas não conseguimos fazer com que o computador as interpretasse de forma a utilizá-las como jogadas.

Conclusão

Inicialmente, quando nos foi dado o enunciado do trabalho, este parecia um pouco menos complexo do que aquilo que realmente, para nós foi, pois com a implementação das funções, surgiram pequenos problemas, seja na forma como estas foram implementadas, seja em termos de raciocínio. Nem sempre aquilo que tínhamos em mente era fácil passar para a linguagem de programação.

Em suma, embora tenham sido encontrados alguns problemas na realização do trabalho e isso não nos tenha permitido fazer algumas das coisas que o enunciado pedia, o trabalho está funcional no que à verificação da validade das jogadas, às jogadas, ao input e output diz respeito.