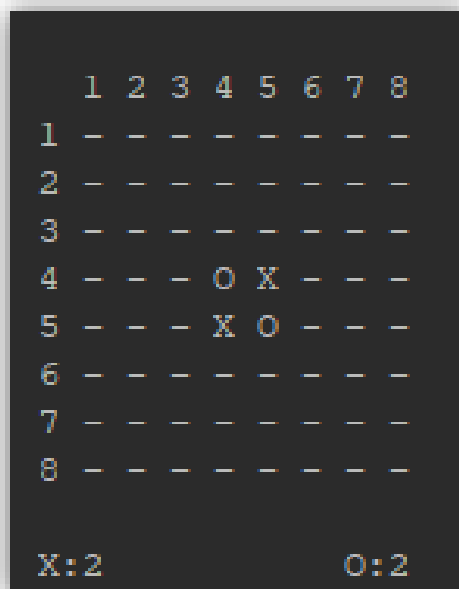




Universidade do Minho
Escola de Engenharia

Relatório do Projeto Prático de LabII e Lab Algoritmos 1º Ano de MIEI e de LCC O Jogo do Reversi



Projeto realizado por:

Ema Isabel Quintãos Dias nº 89518

Leonardo de Freitas Marreiros nº 89537

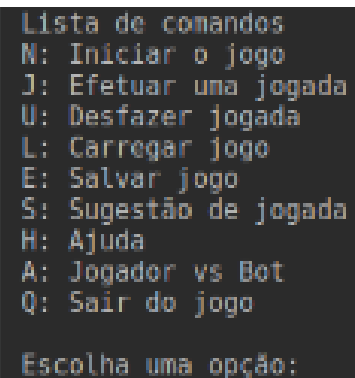
Raquel Sofia Miranda da Costa nº 89464

Grupo nº 34, PL8

No âmbito da cadeira de Laboratórios de Informática II, foi desenvolvido um projeto prático: O Jogo Reversi, na linguagem C.

Para tal, foi efetuada uma linha de comandos, para que o jogador possa selecionar o que pretende realizar. Estes comandos estão na main e são:

- ✓ N - que inicia o jogo e , para o mesmo, é usada a função auxiliar *inicia*;
- ✓ J - que efetua uma jogada, através da função auxiliar *joga*, mas percorrendo também diversas condições, que levam a outras funções auxiliares, de forma a verificar se é possível jogar, e posteriormente, diferenciam-se os modos para proceder ao próximo passo que a que uma jogada obriga;
- ✓ U - função que anula a última jogada, com recurso à auxiliar *undo*;
- ✓ E - que salva um jogo, com um nome à escolha do jogador, a função auxiliar é a *save*;
- ✓ L - que abre um jogo, com um dado nome que foi escolhido pelo jogador, através da função auxiliar *load*;
- ✓ S - dá as sugestões de jogadas, com recurso à função *sugestao*, imprime um * nessas posições de jogada;
- ✓ H - mostra a melhor jogada que o jogador tem disponível, através da auxiliar *help*, imprime um ? nessa posição de jogada;
- ✓ A - que permite que o jogador jogue contra um determinado bot, havendo 3 disponíveis, um bot nível 1, bot mais fácil, nível 2, bot médio e nível 3, bot difícil. A função auxiliar é a *bot*;
- ✓ Q - que termina o jogo;



```
Lista de comandos
N: Iniciar o jogo
J: Efetuar uma jogada
U: Desfazer jogada
L: Carregar jogo
E: Salvar jogo
S: Sugestão de jogada
H: Ajuda
A: Jogador vs Bot
Q: Sair do jogo

Escolha uma opção:
```

Todas as funções auxiliares foram produzidas no estado.c, de maneira a estruturar de forma mais clara o código.

Funções cruciais do programa

As duas funções do código mais importantes são a *valida* e a *validar*, já que são implementadas na grande maioria das funções. A função *valida* é usada na função *joga* e é aquela que permite ver se é

permitido jogar nessa posição e, se sim, efetua a alteração das peças que foram influenciadas pela tal jogada. Já a função validar, apenas indica se é permitido jogar em tal posição, e portanto, é crucial para avaliar o tabuleiro para verificar se ainda há jogadas, de maneira a passar a mesma caso não haja ou verificar se acabou o jogo.

Estas funções foram estruturadas com todas as direções: Norte, Sul, Este, Oeste, Nordeste, Sudeste, Noroeste, Sudoeste.

Estas verificam se naquelas direções existe uma peça igual à sua, em que entre a posição da jogada e essa peça são todas diferentes de vazia e opostas à sua.

Bots

A função bot faz uma distribuição de acordo com os diferentes tipos de modos para as funções *fácil* que remete para o nível um, *médio* para o nível dois e *difícil* para o nível 3.

Inicia o jogo sempre a peça X.

Esta função verifica se será o bot a começar ou o jogador, se o comando A contiver um X então, essa é a peça do bot e inicia o bot, se tiver um O, então começa o jogador e vai para a função *jogador*.

A função *jogador* recorre ao comando J, e no final do mesmo volta para o bot, de acordo com o nível.

```
ESTADO bot (ESTADO e, char c2, int c3) {  
  
    if (c3==1) {e.modos=1;  
        e=facil(e,c2);}   
    else if (c3==2) {e.modos=2;  
        e=medio(e,c2); }   
    else if (c3==3) {e.modos=3;  
        e=dificil(e,c2); }   
    else printf("Nível Inválido\n");  
  
    return e;  
}
```

Bot fácil – nível 1

O bot fácil utiliza um raciocínio bastante elementar, apenas percorre todas as posições e a primeira que encontra onde é válido jogar, com recurso à função *validar*, efetua a jogada, através da função *joga*. No final, regressa para a função *jogador*, já que a jogada seguinte é do jogador.

Bot médio – nível 2

O bot médio tem como principal objetivo jogar na posição onde irá trocar o maior número de peças possível.

Desta forma, é criado um estado temporário onde se joga cada uma das jogadas válidas e se guarda o valor da diferença entre o número de peças do estado temporário e o número de peças do estado

inicial, isto é, o número de peças que seriam trocadas caso se jogasse nessa posição.

Quando o valor atual é maior do que o valor máximo, o valor máximo é atualizado assim como salvaguardadas as posições dessa potencial jogada. Depois de passar por todas as jogadas válidas o bot joga na posição onde se encontrou o valor máximo.

Bot difícil – nível 3

O bot difícil tem como principal objetivo jogar na posição onde irá criar um pior tabuleiro para o adversário. Para isto, foi criada uma função (*avaliaTab*) que avalia um tabuleiro com base nas posições vazias e tendo em conta que certas posições são mais “valiosas” que outras e que certas posições inclusive beneficiam o adversário.

Assim sendo, os cantos, por exemplo, valem muitos pontos enquanto que as posições adjacentes aos cantos retiram pontos. De volta à função do bot difícil, nesta é criado um estado temporário onde se vai jogar cada uma das posições válidas. É, então, aplicada a função *avaliaTab* que guarda o score do tabuleiro do estado temporário. Isto vai permitir inferir acerca de onde é possível efetuar a melhor jogada, pois um tabuleiro com menor pontuação equivale a uma jogada mais eficiente (por exemplo, considerando um tabuleiro com score=500, caso seja possível jogar na posição 1 1, o score novo irá ser 401, enquanto que uma jogada na posição 2 2 devolve um tabuleiro com score=524).

Em cada iteração do ciclo atualizamos o valor do tabuleiro mínimo, se for caso disso, assim como a posição da jogada que causa esse tabuleiro. Depois de passar por todas as jogadas válidas, o bot joga na posição correspondente ao tabuleiro com score mínimo.

Vencedor

O vencedor é dado quando acaba o jogo e é selecionada através de um contador de peças que indicará quem tem mais peças. Sabe-se que o jogo acabou, através da função *acabou*.

```
int contadorX (ESTADO e) {
    int x=0;
    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 8; j++) {
            if (e.grelha[i][j]==VALOR_X) {
                x++;
            }
        }
    }
    return x;
}

int contadorO (ESTADO e) {
    int x=0;
    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 8; j++) {
            if (e.grelha[i][j]==VALOR_O) {
                x++;
            }
        }
    }
    return x;
}
```