**Distrify**

**Relatório Final**



Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

**Grupo 02: Distrify**

André Rodrigues Barros – up201303567  
Edgar Duarte Ramos – up201305973

Faculdade de Engenharia da Universidade do Porto

Rua Roberto Frias, sn, 4200-465 Porto, Portugal

8 De Novembro de 2015

# Resumo

Foi proposto ao grupo desenvolver um jogo em PROLOG, denominado de Distrify, um jogo de tabuleiro cujo objetivo é, como na maioria dos jogos de tabuleiro, tentar vencer o nosso oponente.

O projeto teve como objetivo aplicarmos os conhecimentos transmitidos na unidade curricular, sendo de salientar que PROLOG é uma linguagem de programação diferente do que estamos habituados e por isso foi algo complicado realizar de forma eficiente o jogo, sendo que envolveu muita pesquisa para que conseguíssemos encontrar solução para os problemas que foram aparecendo, aquando do desenvolvimento do projeto. Recorrendo a diversos predicados já existentes, e também criando alguns novos, a solução aos problemas foi encontrada.

Como resultado final do projeto, temos um jogo simples mas apelativo e muito divertido de se jogar, sendo um bom jogo para amantes de jogos de tabuleiro. É, também, um jogo eficiente visto que foi desenvolvido para execução na linha de comandos.

Em suma, é de fácil perceção que o nosso conhecimento da linguagem PROLOG aumentou, sendo possível a consolidação dos conceitos aprendidos nas aulas e aplicados nas aulas práticas, sendo que, a nosso ver, o jogo apresentado é um motivo para um sentimento de realização para o grupo.

# **Índice**

[Resumo 2](#_Toc434785599)

[Índice 3](#_Toc434785600)

[1 Introdução 4](#_Toc434785601)

[2 O Jogo Distrify 5](#_Toc434785602)

[2.1 Detalhes do Jogo 5](#_Toc434785603)

[2.2 Objetivo 5](#_Toc434785604)

[2.3 Jogada 5](#_Toc434785605)

[2.4 Regras de introdução de peças 5](#_Toc434785606)

[3 Lógica do Jogo 6](#_Toc434785607)

[3.1 Visualização do Tabuleiro 6](#_Toc434785608)

[3.2 Execução de Jogadas 6](#_Toc434785609)

[3.3 Final do Jogo 7](#_Toc434785610)

[3.4 Jogada do Computador 7](#_Toc434785611)

[4 Interface Gráfica 8](#_Toc434785612)

[5 Conclusão 13](#_Toc434785613)

[6 Anexos 14](#_Toc434785614)

# Introdução

No âmbito da unidade curricular de Programação em Lógica, do curso Mestrado Integrado em Engenharia Informática e Computação, foi-nos sugerido o desenvolvimento de um jogo na linguagem PROLOG. Dentro de um leque de várias opções que nos foram disponibilizados pelos docentes da unidade curricular, achamos que o Distrify era o jogo com o qual nos identificamos mais, e daí ser essa a nossa escolha.

A escolha em torno de Distrify baseou-se no facto de ser um jogo simples mas divertido de se jogar, porém embora seja simples envolve toda uma estratégia e inteligência que nos apelou e nos levou a escolher o mesmo.

O jogo permite também uma boa prática mental e estratégica com o desenrolar de uma partida entre dois elementos, sendo que o nosso estilo de jogo está dependente das ações do nosso adversário.

O objetivo deste trabalho foi a aplicação dos primeiros conceitos interiorizados na unidade curricular. Este método de avaliação torna-se importante pois permite-nos avaliar os conhecimentos que adquirimos até então. Por outro lado, também permite enriquecer ainda mais o nosso conhecimento, e favorece o trabalho autónomo por parte dos alunos, estudando em simultâneo se os alunos são capazes ou não de produzir algo que é funcional e eficiente a partir destes conhecimentos.

Assim, este relatório encontra-se dividido em várias secções:

* as regras e as jogadas possíveis do jogo
* a lógica implementada no jogo, descrição do projeto e da sua implementação em PROLOG
* o modo de interação entre o utilizador e o programa, e entre a jogabilidade e a visualização
* uma conclusão final do grupo em relação ao jogo e a tudo o que o envolveu

# O Jogo Distrify

## Detalhes do Jogo

Distrify é um jogo de conexão jogado por dois jogadores, constituído por peças pretas e peças brancas, num tabuleiro cujas dimensões vão desde 9x9 até 19x19.

O jogador que controla as peças pretas joga primeiro, colocando uma peça preta em qualquer local do tabuleiro. As jogadas seguintes são alternadas entre os jogadores e cada jogada consiste na colocação de uma ou duas peças em qualquer espaço livre.

## Objetivo

O objetivo do jogo é realizar uma linha de peças que una os limites superior e inferior do tabuleiro, se for o jogador que controla as peças pretas. Por outro lado, se for o jogador que controla as peças brancas, o objetivo é construir um caminho que una os limites laterais do tabuleiro.

## Jogada

Em cada jogada, um jogador pode introduzir uma ou duas peças (sendo que pode ter como objetivo uma de duas opções: bloquear o caminho do adversário ou criar o seu caminho).

## Regras de introdução de peças

No caso em que sejam jogadas duas peças:

a) só podem ser jogadas na vertical ou horizontal

b) não podem resultar na formação de um triplet ou crosscut.

Um triplet refere-se a três peças da mesma cor seguidas em qualquer direção.

Um crosscut refere-se a um bloco de 2x2 em que cantos opostos são controlados por jogadores diferentes

# Lógica do Jogo

## Visualização do Tabuleiro

Os predicados responsáveis pela visualização do tabuleiro na linha de comandos encontram-se disponíveis no ficheiro *distrify.pl*. Os predicados são quase todos recursivos e foram desenvolvidos por camadas.

Para imprimir o tabuleiro basta chamar o predicado ***show\_Board(Board)***, onde ***Board*** é um tabuleiro num certo momento do jogo. Por outro lado, este predicado vai fazendo uso de outros predicados sendo que cada um desses predicados vai sendo mais simples e específico, imprimindo partes mais específicas do tabuleiro, tentando sempre que seja prevalecida a formatação do tabuleiro. Assim, os predicados chamados por ***show\_Board(Board)*** são ***show\_lines(1,Board)*** que mostra as diferentes linhas do tabuleiro, ***printColumnId,*** que imprime o cabeçalho do tabuleiro, e também ***printInitialSeparator,*** que imprime o separador que estabelece o limite entre o cabeçalho e o tabuleiro.

## Execução de Jogadas

Em cada jogada, é pedido ao jogador que tem a vez de jogar as coordenadas do ponto onde quer colocar a peça (ou as peças). Depois, após o programa verificar se as coordenadas são ou não válidas (verificar se não causa situações de triplet e crosscut, e se não existe uma peça no ponto escolhido).

O predicado ***check\_three\_in-row*** é o predicado que se encarrega de verificar se as peças introduzidas causam ou não uma linha de 3 peças seguidas. Caso falhe, o programa retrocede e pede ao jogador para inserir outras coordenadas.

Por outro lado, o predicado ***check\_crosscut*** é o predicado que verifica a existência de crosscut aquando da colocação da nova peça. Este predicado vai chamando outros predicados que vão tratando de situações mais simples até formarem situações mais abrangentes (ex. ***check\_crosscut\_left\_top, check\_crosscu\_left\_bottom,***…).

Seguidamente, o predicado ***choose\_coords*** lê as coordenadas do input do utilizador e verifica se estas são válidas (usando também os predicados anteriores).

Uma vez validadas as coordenadas da peça a colocar, é pedido ao jogador que insira o destino da segunda peça caso este tenha selecionado para colocar 2 peças.

Finalmente, depois da jogada completa ter sido efetuada alterna-se os jogadores e passa a ser a vez do jogador seguinte construir a sua jogada, e tentar combater a jogada do adversário ou então criar caminho para a sua vitória.

## Final do Jogo

Para verificação da existência de um caminho completo, fazemos uso do predicado ***checkCompletePath.*** Este predicado, usando também os predicados ***path***, que por sua vez faz uso do predicado ***neighbor*** (que verifica quais as adjacências de um ponto), vai tentando criar caminhos até encontrar um caminho que seja válido para haver condição de vitória.

Assim sendo, se o predicado ***checkCompletePath*** se verificar o jogo termina e devolve o vencedor.

## Jogada do Computador

As jogadas realizadas pelo computador envolvem toda uma inteligência artificial que tentamos implementar em certos predicados que passamos agora a analisar. Por conseguinte, temos um predicado que se denomina ***simulate\_play*** em que é simulada uma jogada para ver se essa jogada é válida. De outra perspetiva, o predicado ***computer\_set*** coloca efetivamente as peças que vão constituir a jogada do computador.

Posteriormente, existem predicados específicos para cada um dos jogadores para que o computador consiga decidir a melhor jogada para o levar á vitória. O predicado ***decide\_white\_play*** decide que jogada vai ser efetuada por parte do computador (que está a incorporar o jogador portador das peças brancas), tendo em conta também as condições observadas nos predicados analisados anteriormente. No caso do computador que possui as peças pretas, o predicado que decide a sua jogada é o predicado ***decide\_black\_play***. Por outro lado, se nestes dois predicados se adicionar a extensão “\_rand” (por ex. ***decide\_white\_play\_rand***) obtemos um predicado que decide uma jogada aleatória, desde que válida, sendo que não se enquadra numa jogada com vista á vitória por parte do computador.

# Interface Gráfica

A interface da linha de comandos foi feita de forma a proporcionar uma experiência simples ao utilizador, sendo que se apresenta de forma bastante simplificada mas algo cuidada do ponto de vista gráfico. Os menus estão devidamente identificados e para navegar nestes basta clicar no identificador do mesmo e em seguida pressionar *Enter*. Nestes menus não é necessário acrescentar o ponto após o identificador do menu, ou seja para escolher o menu “Play” devemos colocar “1” e depois carregar *Enter*, em detrimento de “1.” e depois *Enter*.

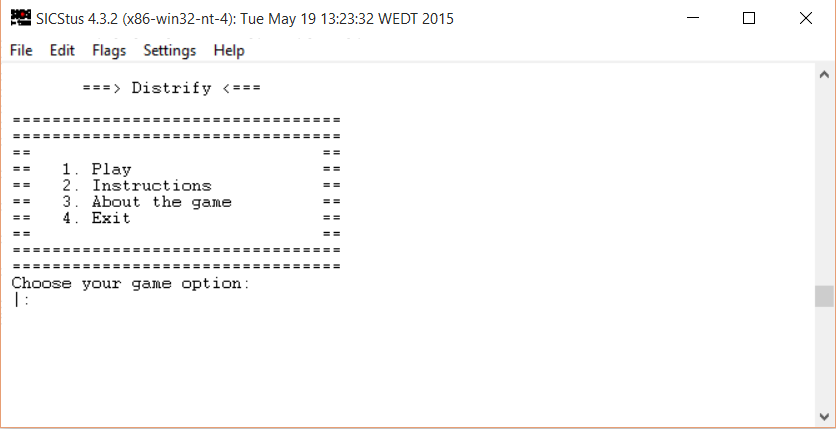


Figura - Menu principal.

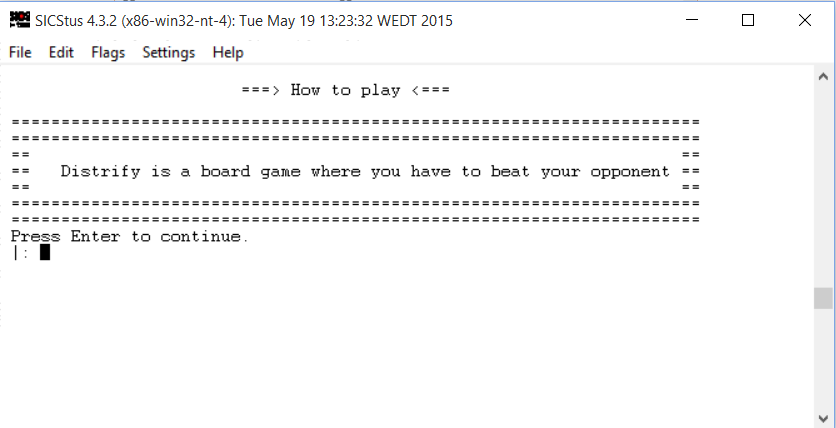


Figura - Instruções do jogo.

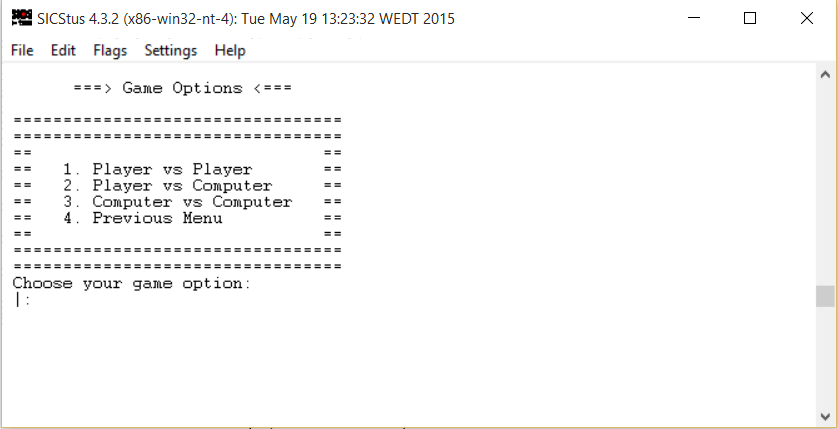


Figura - Menu de escolha do modo de jogo.

Em todos estes menus, é de reparar que a consola vai sendo limpa ao passo de cada escolha de um novo menu. Isto é feito pelo predicado ***clearScreen*** definido no ficheiro ***utils.pl***.

O estado atual do tabuleiro é exibido, bem como uma mensagem do jogador que tem a vez de jogar. De seguida, é solicitado ao jogador que insira as coordenadas da peça a colocar.

Para inserir coordenadas, basta inserir a linha e a coluna da peça escolhida e confirmar com *Enter*.

Quando um jogador tenta fazer um movimento que não é permitido, o motor do jogo exprime uma mensagem de erro e volta até ao ponto onde é permitido ao utilizador introduzir novamente outras coordenadas para serem validadas.

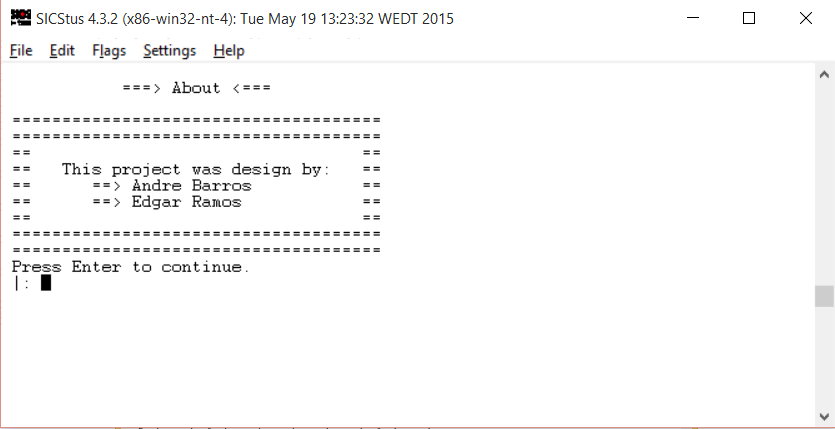


Figura – Menu de informações do projeto.

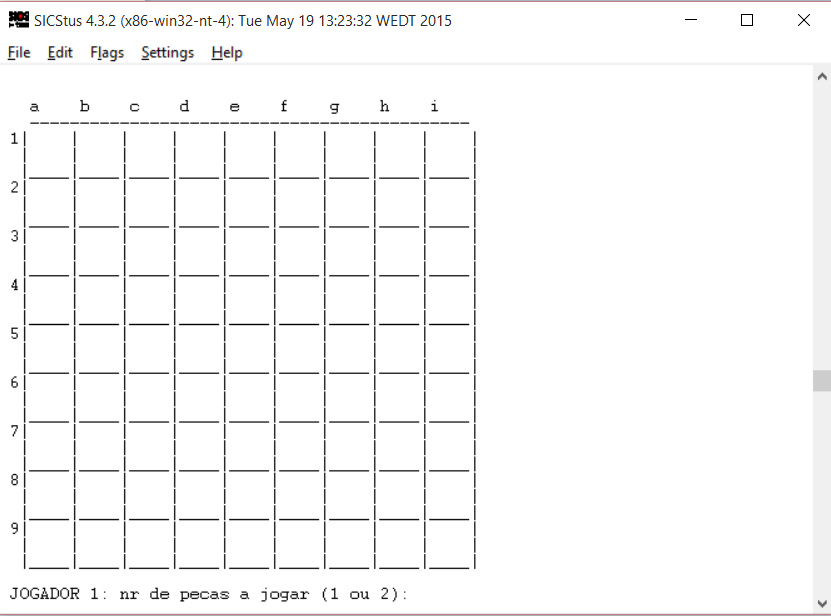


Figura - Estado inicial de um jogo.

A partir do menu de jogo, passa a ser necessária a introdução de ponto a seguir ao número da linha e da coluna, e também do número de peças a jogar, ou seja, no número de peças a jogar dizemos, por exemplo, “1.” Se quisermos jogar uma peça, ou “2.” Se quisermos jogar 2 peças. Caso haja a falta do ponto o programa fica a espera que este seja posto e não avança (como é habitual em prolog).

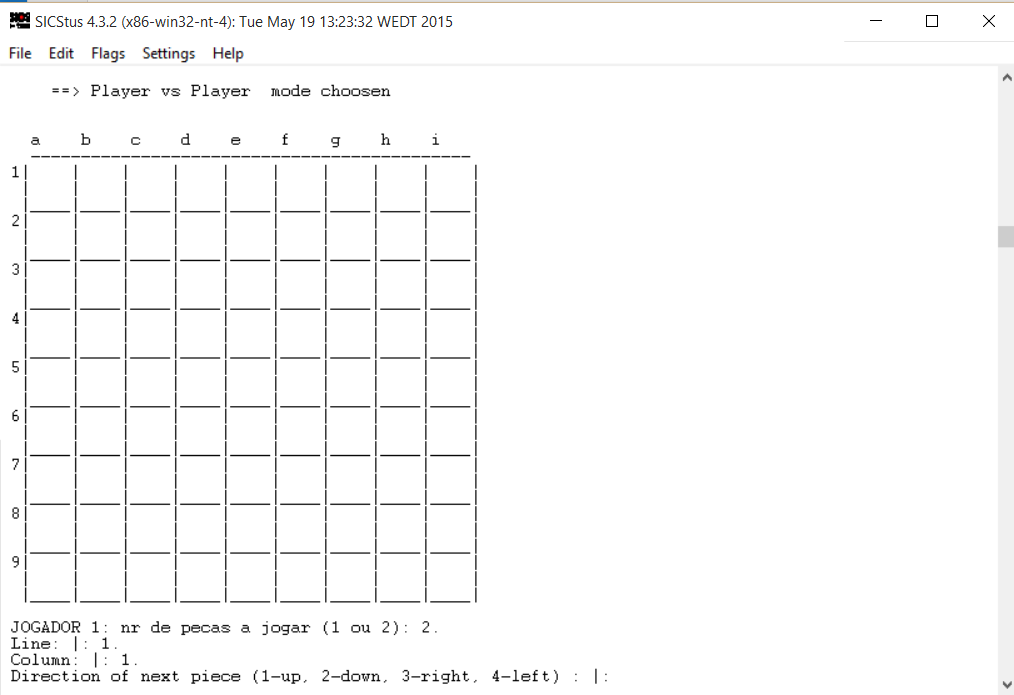


Figura - Exemplo do pedido de informações para onde o utilizador quer colocar a peça.

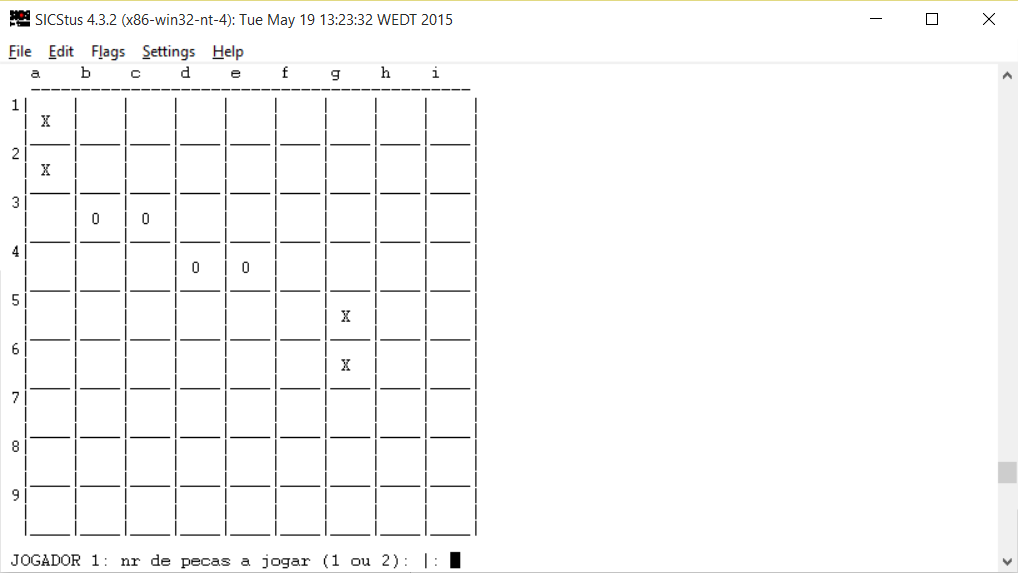


Figura - Exemplo de jogo após 2 jogadas

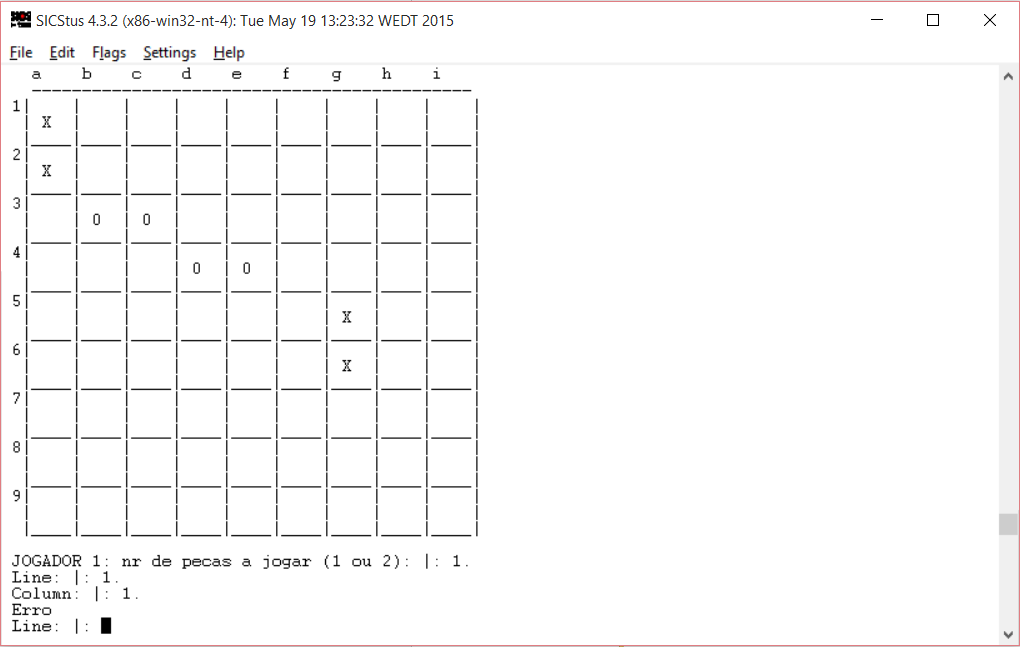


Figura - Exemplo de seleção de uma coordenada onde já se encontra uma peça.

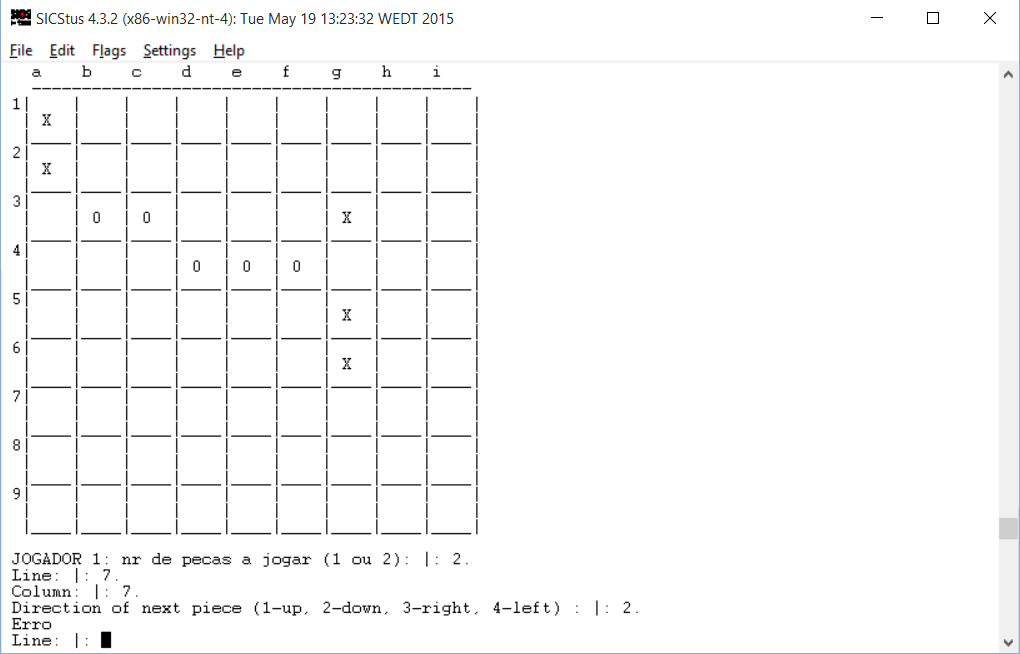


Figura - Exemplo de seleção de uma coordenada de destino inválida.

# Conclusão

O jogo Distrify exigiu imenso tempo ao grupo para a sua implementação. Como já frisamos anteriormente, Prolog é uma linguagem diferente pelo que se tornou algo difícil de perceber como seria a forma mais eficiente de verificarmos algumas das condições, principalmente relativas à inteligência do computador nas jogadas que efetua, visando a vitória no jogo.

O grupo vê o resultado final como positivo. Também os conhecimentos adquiridos durante o desenvolvimento do projeto são algo de realçar pois é algo que tem influência tanto no decorrer da unidade curricular como no percurso académico.

Distrify, sendo um jogo bastante lógico, mostrou-se um desafio que, com esforço e dedicação, e também bastante entreajuda, se tornou um jogo muito apelativo e simples que proporciona ao jogador um bom passatempo.

As dificuldades encontradas foram superadas, porém o trabalho poderia ser melhorado nalguns pontos, sendo que com a existência de alguns trabalhos de outras unidades curriculares a sobreporem-se em termos de horários e prazos se tornou complicado.

Ao contrário do que estamos habituados, este tipo de linguagem requer um pensamento lógico em cada predicado desenvolvido. Por isso e em jeito de conclusão, podemos afirmar que o grupo gostou da experiência de desenvolvimento de um jogo na linguagem PROLOG.

# Anexos

O código fonte do projeto encontra-se na pasta *src* anexada junto deste relatório.