# **1->(“Introdução\n”);**

A linguagem C é uma linguagem de programação estruturada, iterativa, padronizada pela ISO, criada em 1972, por Dennis Ritchie, no AT&T Bell Labs, para desenvolver o sistema operacional Unix (que foi originalmente escrito em Assembly).

C é uma das linguagens de programação mais populares e existem poucas arquiteturas para as quais não existem compiladores para C. Esta linguagem tem influenciado muitas outras linguagens de programação, mais notavelmente C++, que originalmente começou como uma extensão para C.

Neste trabalho será apresentado um pequeno jogo de tabuleiro, o jogo do comilão, que permitirá a jogabilidade alternada entre dois utilizadores e a jogabilidade entre um jogador e um jogador automático, até que um deles perca.

O programa encontra-se orientado para a simplicidade de introdução de dados e para a facilidade de manobra dos mesmos. Todas as validações de dados estão incorporadas e todas as restrições de introdução estão limitadas ao mínimo.

Este trabalho foi desenvolvido no âmbito da disciplina de Programação do 2º semestre do 1º ano de Engenharia Informática, em regime pós-laboral.

# **2->(“Manual de Instruções\n”);**

O jogo do comilão é efetuado num tabuleiro retangular dividido em células. As células são identificadas pela linha e coluna em que se encontram. No início todas as células contêm alimento comestível (\*), exceto a do canto inferior direito que contém veneno (X).

O jogo realiza-se entre 2 jogadores que efetuam movimentos alternados. Em cada jogada, os jogadores escolhem uma das células com comida. Para além da posição selecionada, todas as células para a esquerda e para cima da escolhida passam a estar vazias. As jogadas sucedem-se alternadamente, perdendo o jogador que for obrigado a ingerir o veneno.

De seguida serão descritos os principais elementos do jogo assim como o procedimento a tomar para jogar e poder usufruir de longos momentos de diversão.

## <Interface Gráfica>

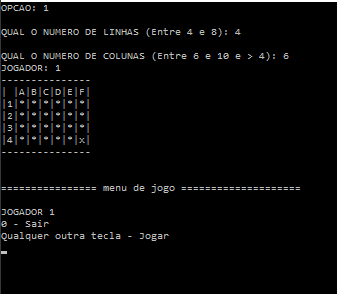


Ao iniciar o jogo é apresentado o “Menu Inicial” (Fig. 1) onde deverá selecionar a operação pretendida.

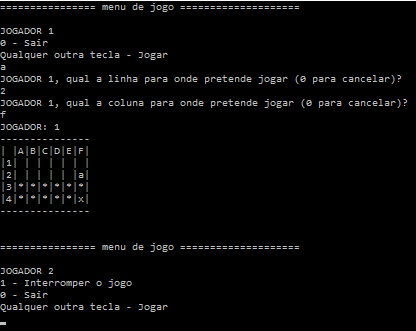
* Opção 1 para iniciar um jogo com 2 jogadores;
* Opção 2 para iniciar um novo jogo com um jogador automático como opositor;
* Opção 3 para recomeçar um jogo interrompido;
* Opção 0 para sair do jogo.

**Opção 1 para iniciar um jogo com 2 jogadores;**

Nesta opção damos início ao jogo interativo com dois jogadores (2 pessoas). Após a escolha desta opção o programa questiona o utilizador do tamanho pretendido para o tabuleiro de jogo, tal como está demonstrado na (Fig.1).

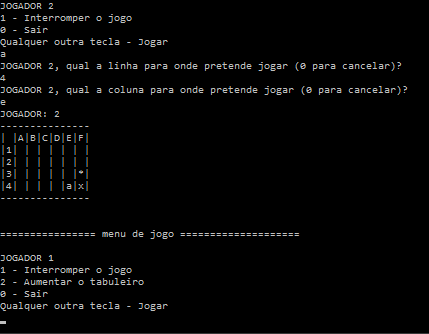


Após a escolha das dimensões do tabuleiro damos início ao jogo, mas dando hipótese de o utilizador sair se assim pretender, caso o jogador pretenda continuar, basta carregar em outra tecla.



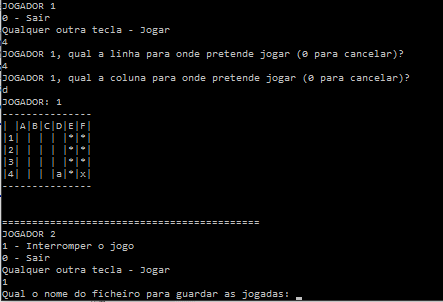
Ao clicar noutra tecla será solicitado ao jogador as coordenadas da sua jogada (nº de linha e letra de coluna). Após o utilizador efetuar a respetiva introdução das coordenadas e prosseguir, o tabuleiro será atualizado limpando os restante sectores, tal como solicitado no enunciado.

Depois do jogador 1 efetuar a jogada é a vez do jogador 2 jogar, introduzindo as coordenadas pretendidas para a sua jogada, tal como feito com o jogador 1.

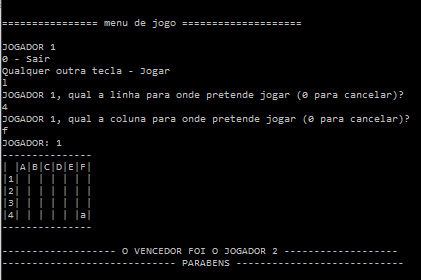


Tal como se pode observar na figura anterior, o jogador 2 efetuou uma jogada, onde levou o “a” a reposicionar-se. (entenda-se “a” como símbolo da atual posição).

Após a primeira jogada é dado ao utilizador a hipótese de aumentar as dimensões do tabuleiro, mas cada utilizador só tem direito a efetuar essa transformação uma vez.

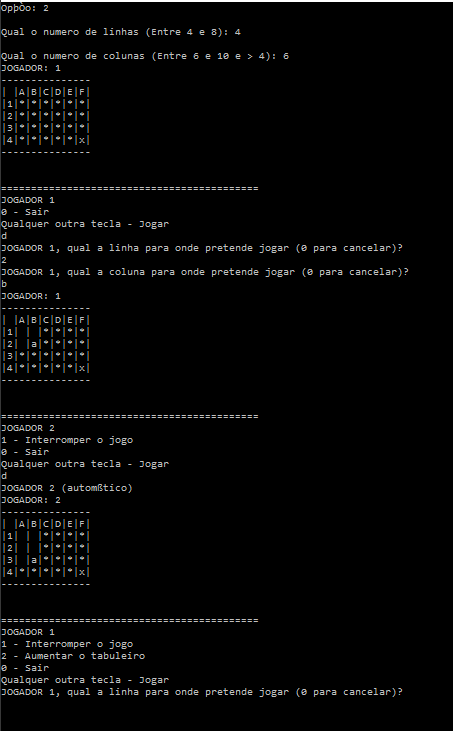


Caso um dos jogadores ganhe, é apresentado no ecrã uma mensagem com a indicação de quem foi o vencedor, tal como exemplificado na imagem seguinte.



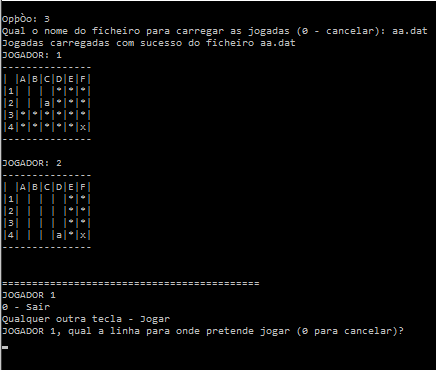
**Opção 2 para iniciar um jogo com um jogador automático como opositor;**

Aqui o princípio de jogo é o mesmo, com a exceção de só jogar uma pessoa contra o computador.



**Opção 3 para recomeçar um jogo interrompido;**

Após um dos utilizadores ter decidido interromper o jogo, gravou e atribui um nome, ao escolher esta opção, basta escrever o respetivo nome do ficheiro e logo terá acesso às jogadas efetuadas até ao momento podendo assim posteriormente continuar o jogo a partir da posição de onde efetuou a interrupção, tal como exemplificado na seguinte imagem.



**Opção 0 para sair do jogo;**

Ao escolher essa opção os jogadores optaram por sair do jogo.

# **3->(“Principais Estruturas de Dados\n”);**

Perante o desafio de criar o jogo proposto, foram traçados vários cenários para a elaboração deste programa. Esses protótipos tinham como base o enunciado e nem todos eram no ponto de vista funcional, nem intuitivos para o utilizador, nem “limpos” para o computador. Tentou-se criar um programa simples e leve que proporcionasse uma boa experiência ao utilizador, neste caso particular, jogador.

No último esboço, foram criadas duas estruturas de dados que se iriam tornar nos pilares principais do motor do jogo:

* **O TABULEIRO**, que consiste num array dinâmico (com valor mínimo e valor máximo definidos paras as linhas e para as colunas). Este é responsável por toda a apresentação “gráfica” de interação com os jogadores. Trata-se de um array bidimensional, onde cada célula da matriz tem o valor 1 (com comida) ou 0 (sem comida). É dinâmico, pois ao longo do jogo, o seu tamanho pode aumentar, pelo menos uma vez por cada jogador.
* **AS LISTAS** de jogadas. As jogadas são sequenciais e inseridas no final da lista, ficando assim desde logo ordenadas de acordo com o desenrolar do jogo. Cada objeto (elemento) da lista guarda o jogador que fez a jogada, bem como a linha e a coluna atual.

A partir daí sabe-se o estado do tabuleiro (figura 2 do enunciado) após cada jogada, pois a regra é, em cada posição da jogada, todas as células para cima e para a esquerda ficam vazias.

Quando a lista está vazia, o jogo encontra-se no estado da figura 1 do enunciado. De salientar que esta estratégia só funciona porque as jogadas são sequenciais.

Isto é: no ficheiro Prog1516\_Enunciado\_TP-1\_2, a jogada 3 só é possível desenhar o tabuleiro com aquele estado, pois antes temos a jogada 1 e a 2. Se formos buscar a jogada 3 por si só (sem as jogadas anteriores), teríamos, segunda a regra do jogo, apenas a linha 1 até à coluna F estaria vazia.

No entanto, como as jogadas são apresentadas de forma sequencial até à jogada 3, basta guardar o estado atual de cada jogada (coluna atual) no ficheiro evitando assim sobrecarregar o ficheiro com o o estado de cada célula do tabuleiro.

Exemplo de esquema de implementação da lista do jogo

|  |  |
| --- | --- |
| Jogada linha actual  coluna actual jogador |  |

|  |  |
| --- | --- |
| Jogada linha actual  coluna actual jogador |  |

|  |  |
| --- | --- |
| Jogada linha actual  coluna actual jogador | N U  L  L |

A seta indica ponteiro “prox” (…) até

# **4->(“Implementação\n”);**

## **4.1->(“Programa Principal\n”);**

Iniciando no ficheiro “**main.c**” irei descrever cada função e qual a sua tarefa que lhe está associada dentro do programa. Neste capítulo tentarei ser o mais claro possível para que a minha tomada de decisão dentro da estrutura seja entendida.

* **void inicio(int \*l, int \*c)** aqui é iniciado o jogo após o jogador ter feito essa mesma escolha. São inicializados dois ponteiros \*l (linhas) e \*c (colunas) que serão responsáveis pela dimensão do tabuleiro, de acordo com a opção tomada por parte do jogador;
* **void mostrar\_jogadas(pjogada lista, int \*\*tabuleiro, int nlin, int ncol)**, esta função é responsável por mostrar as jogadas guardadas pelo utilizador, no jogo que guardou e mostra a próxima enquanto não encontra o ponteiro lista não aponta para o NULL, ou seja, quando aponta para o final da lista;
* int **set\_jogada(int \*\*tabuleiro, int nlin, int ncol, int automatico, int jogador, int \*posactl, int \*posactc),** esta função vai perguntar ao jogador qual a jogada, e retorna nas variáveis enviadas por referência (ponteiro) a linha e coluna selecionada e enviamos o tabuleiro como ponteiro, tal como indicado numa aula teórica, como a melhor maneira de enviar um array dinâmico;
* **int menu(),** esta é a função responsável pela interação gráfica do jogo com o jogador, funcionando como interprete das escolhas dos jogador perante as opções disponíveis.
* **int main(int argc, char\*argv[]),** aqui a função controla o tamanho do tabuleiro. Este terá tamanho variável nesse sentido segui a ficha 4 - estruturas dinâmicas.  
  No ponto 3.3 do enunciado diz que podemos evoluir para um aumento do tabuleiro durante o jogo em que iremos usar o realloc[[1]](#footnote-1), *que permite alterar o número de bytes que estão presentemente associados a um bloco previamente criado utilizando a função malloc;* Usei um array em vez de outra estrutura (struct ou listas), pois em cada elemento do tabuleiro apenas temos 2 valores possíveis (que no fundo são 3)[[2]](#footnote-2):
  + 1: se a célula tem elemento comestível;
  + 0: se a célula foi comida;

Perante esta opção cada campo está devidamente comentado para melhor perceção da sua usabilidade junto do motor de jogo;

## **4.2->(“Tabuleiro\n”);**

Estando completa a descrição desta estrutura seguimos para a próximo ficheiro do jogo que é o “tabuleiro.c”. Aqui seguirei a mesma linha condutora do ficheiro anterior, descrevendo as valências de cada função nele contido.

* **void inicia\_tabuleiro(int \*\*tabuleiro, int nlin, int ncol),** reserva memória para o tabuleiro e preenche todas as células do tabuleiro com o valor "1" que significa que esses campos contêm comida que ainda não foi comida, passo a redundância;
* **int \*\*aumenta\_tabuleiro(int \*\*tabuleiro, int \*nlin, int \*ncol**), esta função é responsável pelo aumentar de tabuleiro durante o jogo, tal como solicitado no enunciado. Realoca no tabuleiro o novo valor de linhas e mantem os valores de memória que estão para trás, em termos de coluna usamos o malloc, pois estamos a fazer o malloc numa nova linha, alocando memoria onde não estava. (foi feito comentário no código para respectiva identificação);
* **void atualiza\_tabuleiro(int \*\*tabuleiro, int nlin, int ncol, int posactl, int posactc),** aplica a regra do jogo descrita na função anterior, ou seja, tudo o que está para cima e para trás da linha e coluna atual do tabuleiro, ficam com célula vazia (com comida);
* **void desenha\_tabuleiro(int \*\*tabuleiro, int nlin, int ncol, int jogador, int posactl, int posactc),** enviamos o tabuleiro como ponteiro, pois o seu valor vai ser alterado em atualiza\_tabuleiro. “l” e “c” designam o tamanho em linhas e colunas do tabuleiro. “posactl” e “posactc” designam a posição actual nas coordenadas linha e coluna atuais do tabuleiro.

## **4.3->(“Listas\n”);**

Seguimos posteriormente para o ficheiro “lista.c”. Que tem uma função também muito importante no motor do jogo, pois é esta que guarda as jogadas feitas pelos jogadores durante o jogo (passo a redundância). No tópico anterior (Principais Estruturas de Dados), inseri um esquema de como processo as listas dentro deste jogo.

* **int lista\_vazia (pjogada lista),** verifica se a lista está vazia;
* **pjogada lista\_inserir(pjogada lista, int j, int linha, int col),** percorre a lista ate ao fim (até que o elemento->proximo seja null, ou seja, até que não haja próximo, o próximo desse elemento será o novo elemento. O apontador da lista fica sempre a ser o primeiro elemento da lista, a lista é passada em parâmetro por valor e é retornado, já com o novo elemento inserido no final,
  + j -> nº do jogador atual da jogada;
  + linha e col --> coluna atual da jogada;
* **int lista\_gravar(char \*nomefich, pjogada lista, int automatico, int nlin, int ncol),** para além de gravar as jogadas, há que gravar, no início do ficheiro, o tabuleiro. Para isso basta gravar o nº de linhas e colunas do tabuleiro;
* **pjogada lista\_ler(char \*nomefich, int \*automatico, int \*nlin, int \*ncol),** “nlin” e “nco”l são variáveis passados por referência, ou seja, o que é passado por parâmetro não é o seu valor mas o seu endereço, permitindo que o conteúdo dessas variáveis passe para fora da função;
* **int lista\_tamanho(pjogada lista),** esta verifica o tamanho da lista.
* **void lista\_libertar(pjogada lista),** esta liberta a lista das jogadas efectuadas, para o jogo seguinte:

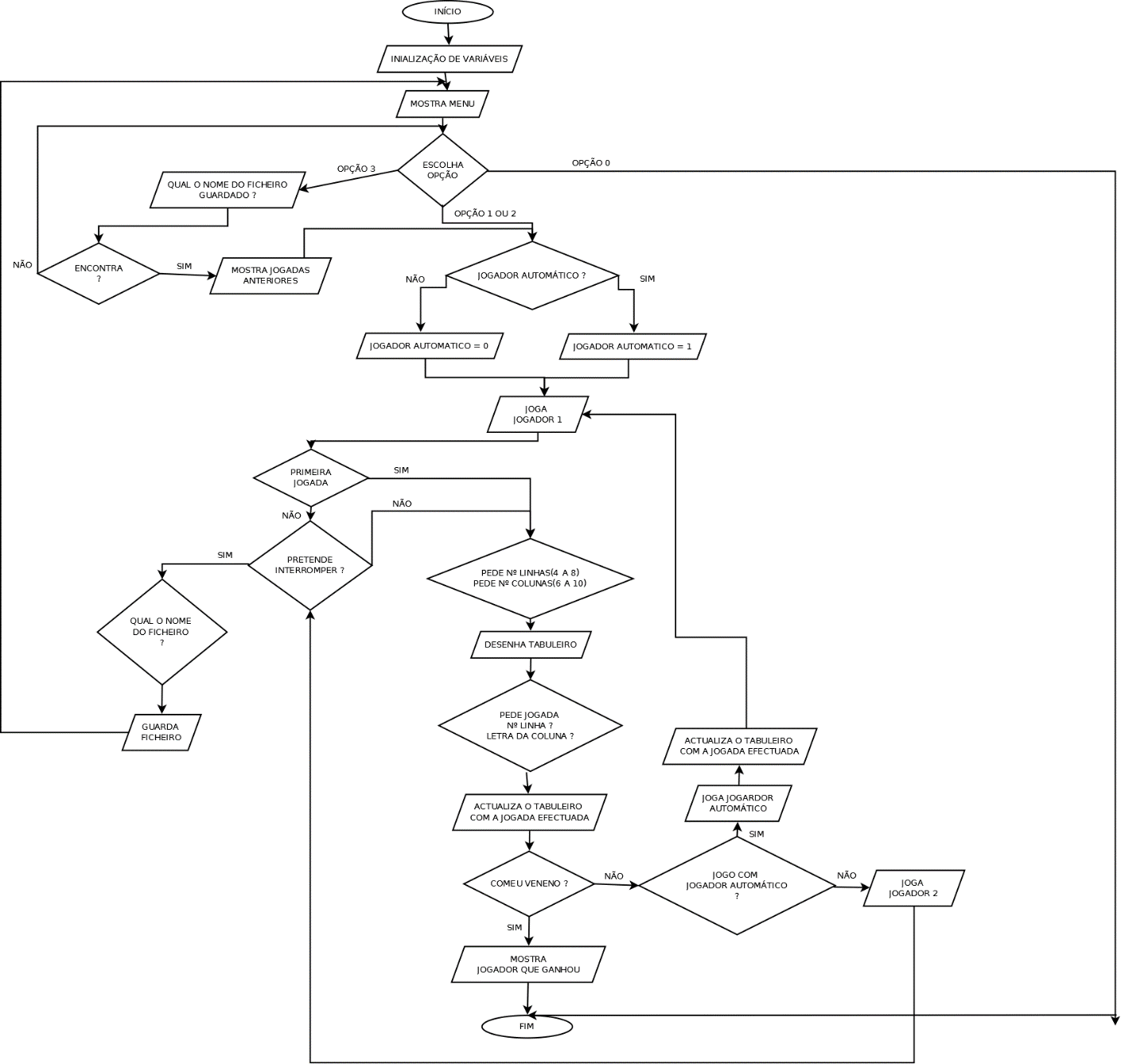
Estando os ficheiros principais totalmente descritos, resta explicar o conteúdo dos headrers files. *“Estes têm sempre a extensão* ***.h*** *pois não têm código mas apenas os cabeçalhos (headers) das funções que representam.”[[3]](#footnote-3)*

Estas já foram totalmente minuciosamente descritas, mas serão demonstradas.

O ficheiro “lista.h”, contém os cabeçalhos das funções do ficheiro “lista.c”, que posteriormente será utilizado em outros ficheiros, permitindo assim o acesso a todas as funções input e output normais. O mesmo acontece com o ficheiro “tabuleiro.h”, que contém os cabeçalhos das funções do ficheiro “tabuleiro.c”.

Assim desta forma, concluo a descrição das estruturas dinâmicas implementadas no motor do jogo.

# **5->(“Fluxograma do Programa\n”);**



# **6->(“Conclusão\n”);**

A realização do trabalho proposto permitiu o aprofundar de conhecimentos previamente adquiridos, nas aulas de Programação. A linguagem C é muito vasta e complexa, mas ao progredir no trabalho, requereu uma constante pesquisa de material e funções para satisfazer as dificuldades que iam aparecendo. Desde apontamentos tirados nas aulas, leitura dos slides e seus exemplos, à pesquisa no site stackoverflow. Contudo considero que os princípios básicos foram captados e aplicados neste programa.

Existiram algumas dificuldades no desenvolver do código, como é óbvio, mas foram ultrapassadas e assim foi possível aperfeiçoar o trabalho e o código de modo a ser o mais eficiente possível.

# **7->(“Referências\n”);**

|  |  |
| --- | --- |
| [1] | L. Damas, Linguagem C, Lisboa: Lidel, 1999. |
| [2] | S. E. Inc, “stackoverflow,” [Online]. Available: http://stackoverflow.com/questions/1970698/using-malloc-for-allocation-of-multi-dimensional-arrays-with-different-row-lengt. [Acedido em 28 Maio 2016]. |

*Norma IEEE*

1. in: Linguagem C de Luís Damas, 18ª edição [↑](#footnote-ref-1)
2. in: http://stackoverflow.com/questions/1970698/using-malloc-for-allocation-of-multi-dimensional-arrays-with-different-row-lengt [↑](#footnote-ref-2)
3. in: Linguagem C de Luís Damas, 18ª edição [↑](#footnote-ref-3)