

A dark blue vertical bar runs down the left side of the page. A blue arrow points to the right from this bar, containing the date.

04-07-2016

JOGO DO COMILÃO

Programação (Trabalho Prático)

Several thin, curved lines in shades of blue and grey originate from the bottom left corner and sweep upwards and to the right.

Tiago Miguel Matrola Simões

A 21240385 / ANO LETIVO: 2015-2016

ÍNDICE

1. Introdução.....	2
1.1. Apresentação das principais estruturas de dados.....	2
1.2. Apresentação detalhada das estruturas dinâmicas implementadas.....	3
2. Algoritmo.....	5
2.1. Descrição dos ficheiros utilizados.....	5
2.2. Estrutura geral do programa, incluindo a apresentação de algoritmos de alto nível que clarifiquem as suas principais funcionalidades.....	6
2.3. Divisão do programa por ficheiros.....	8
3. Manual de utilização.....	9
3.1. Pequeno manual de utilização.....	9

1. INTRODUÇÃO

1.1. Apresentação das principais estruturas de dados

PONTEIROS PARA TRATAR FICHEIROS

FILE *f -> abre ficheiro para exportar sucessões do mapa do jogo, ficheiro texto(txt).

FILE *file_open -> abre ficheiro para guardar jogo com a finalidade de ser retomado mais tarde, ficheiro binário(bin/dat).

FILE *file_save -> abre ficheiro com a continuação do último jogo e importa valores para as variáveis.

VARIÁVEIS DO TIPO "INT"

Linhas -> número de linhas da tabela

Colunas -> número de colunas da tabela

Celulas -> linhas * colunas (total de células)

I, J -> iterações dos ciclos for

Jogada -> jogada que o jogador seleccionou (1 – Escolhe o movimento no mapa, 2 – Aumentar tamanho do mapa, 3 – Guarda jogo e sai, 4 - Mostra historial do jogo).

Linha_escolhida / coluna_escolhida -> linha/coluna escolhida pelo jogador para comer no mapa.

Indice_linha / indice_coluna -> valores das linhas/colunas no mapa, começa em 1 para ser mais confortável para os jogadores.

Jogador -> identifica quem é o jogador actual (0 – Jogador A, 1 – Jogador B, 2 – Jogador CPU (jogo automático)).

Flag -> verifica se a posição no mapa que o jogador seleccionou é válida ou inválida (0 – posição válida, 1- posição inválida).

Guardado -> verificar se o jogo foi guardado ou não (0 - não / 1 - sim).

Posicao_estrutura -> quantas estruturas já foram preenchidas nas listas-ligadas.

Resposta_auto -> verifica se o jogo é automático ou não (1 – Manual(não), 2 – Auto).

Autom -> passa à função “get_linhas_colunas” que se o jogo é automático ou não.

VARIÁVEIS DO TIPO “CHAR”

Answer -> pergunta ao jogador se este pretende continuar o jogo anterior.

VECTORES DE INTEIROS

Save_arr -> guarda valores do jogo para este ser retomado mais tarde (0 – linhas, 1 – colunas, 2 -jogador, 3 – posicao_estrutura).

fwrite_arr -> guarda valores das listas-ligadas (variável temporária).

tamanho_arr -> verifica se o jogador já aumentou o tamanho do tabuleiro (posição[0] – jogador A, posição [1] – jogador B) (0 – não, 1- já aumentou).

ESTRUTURAS

Lista -> Vector de estruturas / listas – ligadas -> pergunta ao jogador se este pretende continuar o jogo anterior.

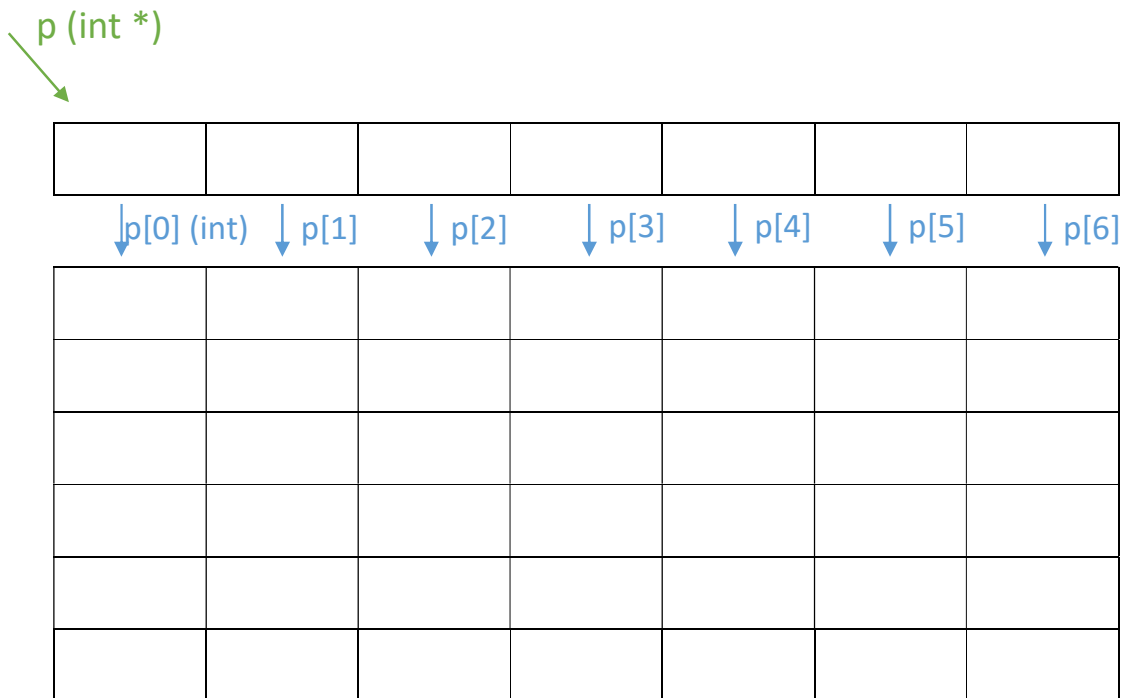
Pe -> ponteiro para as listas-ligadas.

Estrutura -> “base de dados” da situação do jogo (colunas_e, linhas_e – valores com as posições escolhidas pelos jogadores (colunas_e - abreviatura de “colunas escolhidas”), tam_linhas/tam_colunas – numero de linhas e colunas na situação actual, jogador_no_momento – o jogador no momento, *próximo – ponteiro com o endereço para a próxima lista).

Tam_alterado -> “base de dados” com a finalidade de verificar se os jogadores já aumentaram o tamanho do mapa (0 - não, 1 – sim).

1.2. Apresentação detalhada das estruturas dinâmicas implementadas

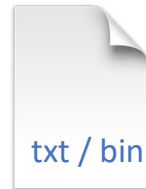
O **Tabuleiro do jogo** é alocado dinamicamente usando *malloc* e guarda o tabuleiro no ponteiro *p*. Cada célula do vetor *p* aloca memória dinâmica criando assim uma grelha.



O **Tabuleiro temporário** é alocado dinamicamente usando *malloc* e guarda o tabuleiro temporário no ponteiro *temp*. Cada célula do vetor *temp* aloca memória dinâmica criando assim uma grelha. Este tabuleiro é criado quando o jogador pretende aumentar o tamanho do tabuleiro principal. Quando esta situação ocorre o tabuleiro *temp* passa a ser o tabuleiro principal, mas antes de se tornar o tabuleiro definitivo este copia os dados do tabuleiro *p* para o tabuleiro *temp*.

2. ALGORITMO

2.1. Descrição dos ficheiros utilizados



Ficheiro **estados.txt** (FILE *f):

No final do jogo a sucessão de estados do tabuleiro é exportada para o ficheiro de texto “estados.txt”. Neste ficheiro está informação detalhada sobre como as jogadas alteraram a distribuição das peças no tabuleiro.

Ficheiro **save.bin** (FILE *file_save, *file_open):

Este ficheiro permite que o jogo seja interrompido e retomado mais tarde. O programa guarda num ficheiro binário toda a informação que permita retomar o jogo numa altura posterior. O ponteiro file_save é usado para guardar o ficheiro save.bin e o file_open para abrir o ficheiro save.bin para extrair a informação do jogo.

2.2. Estrutura geral do programa, incluindo a apresentação de algoritmos de alto nível que clarifiquem as suas principais funcionalidades

INICIO DO PROGRAMA

CRIA ficheiro de texto estados.txt, para escrita.

SE não conseguir criá-lo:

ENTÃO retorna 1.

MOSTRA mensagem introdutória (função mostra_intro()).

PERGUNTA ao jogador se este pretende continuar o jogo anterior.

SE resposta for SIM:

ENTÃO abre o ficheiro save.bin para leitura.

SE não conseguir abrir o ficheiro:

ENTÃO começa um jogo novo.

SENÃO começa um novo jogo.

PERGUNTA ao jogador se este pretende jogar com a CPU (jogo automático).

SE resposta for SIM:

ENTÃO começa um jogo automático contra a CPU.

SENÃO começa um jogo manual (2 jogadores).

ENQUANTO os jogadores não chegam a um acordo à cerca do número de células:

PERGUNTA o número de células e se estes concordam.

SE concordam

ENTÃO sai do ciclo

FIM CICLO

CRIA o tabuleiro do jogo (alocado dinamicamente).

MOSTRA manual de instruções (função mostra_manual())

ENQUANTO um dos jogadores não ganhar:

SE um dos jogadores ganhar

ENTÃO sai do ciclo, mostrando quem foi o vencedor.

SENÃO

SE o jogador escolher a opção 1

ENTÃO pede o movimento que o jogador pretende.

SE o jogador escolher a opção 2

ENTÃO verifica se esta escolha foi feita anteriormente.

SE foi

ENTÃO indica ao jogador que a jogada é inválida.

SENÃO aumenta o tabuleiro.

SE o jogador escolher a opção 3

ENTÃO guarda o jogo e termina o programa (retorna 0).

SE o jogador escolher a opção 4

ENTÃO mostra todas as jogadas efetuadas anteriormente.

SENÃO indica ao jogador que a jogada é inválida.

FIM CICLO

FIM DO PROGRAMA

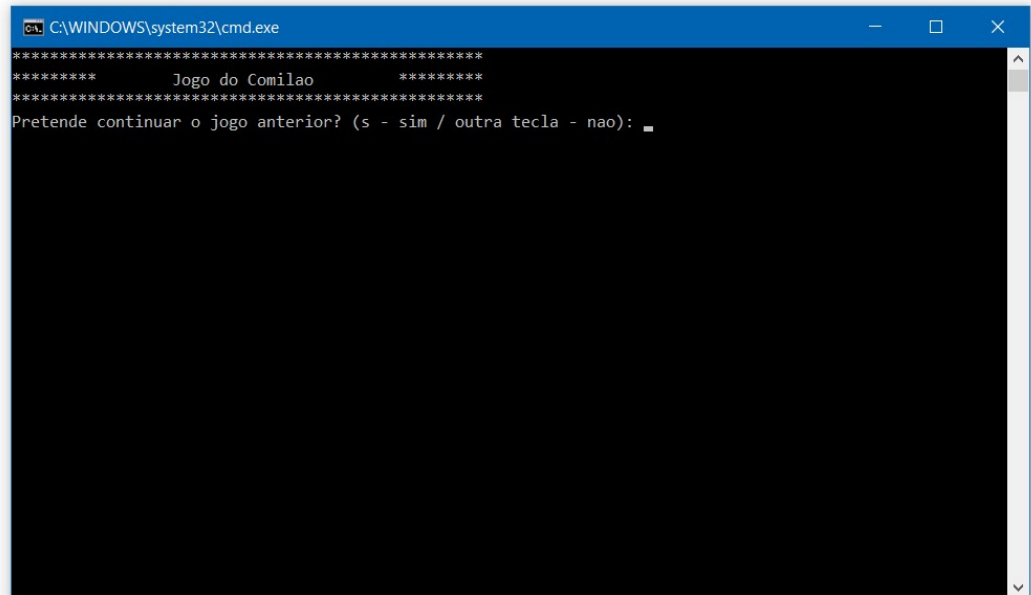
2.3. Divisão do programa por ficheiros



3. Manual de utilização

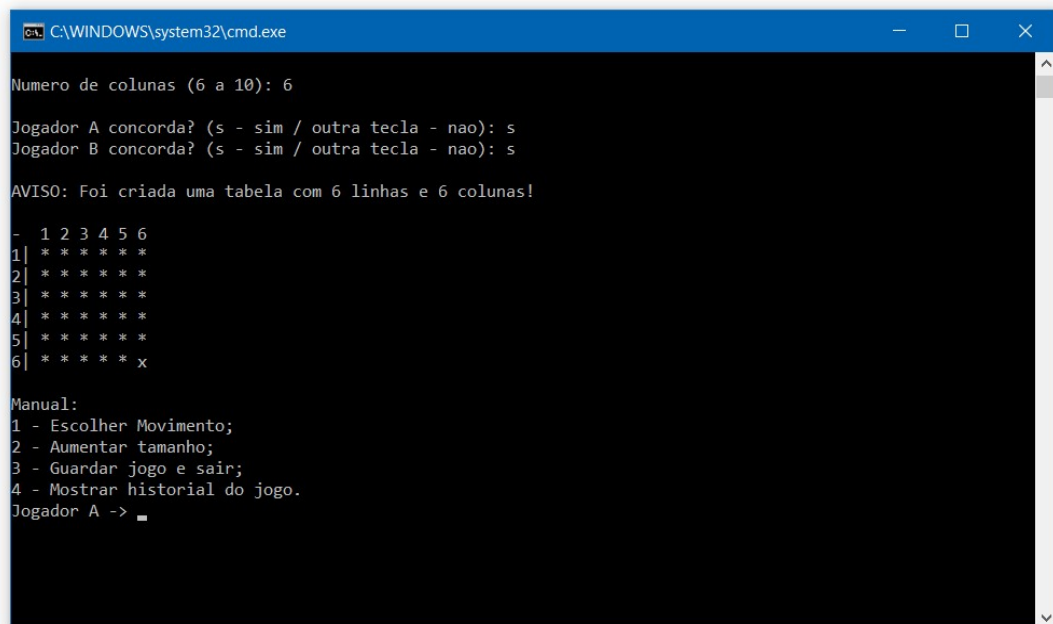
3.1. Pequeno manual de utilização.

Começo do jogo:



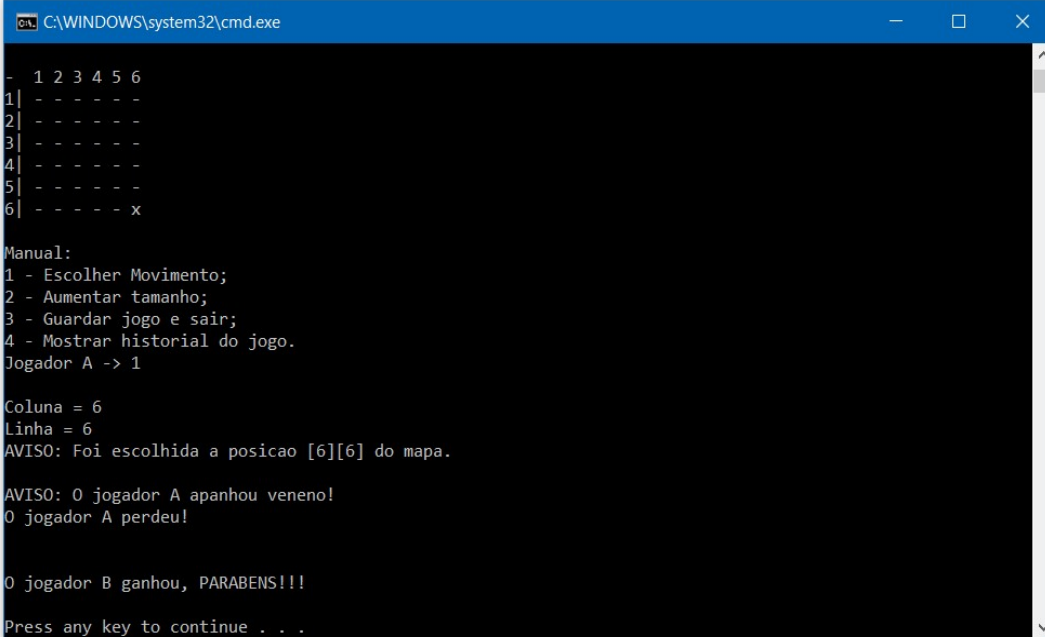
```
C:\WINDOWS\system32\cmd.exe
*****
*****  Jogo do Comilao  *****
*****
Pretende continuar o jogo anterior? (s - sim / outra tecla - nao):
```

Pergunta qual a jogada:



```
C:\WINDOWS\system32\cmd.exe
Numero de colunas (6 a 10): 6
Jogador A concorda? (s - sim / outra tecla - nao): s
Jogador B concorda? (s - sim / outra tecla - nao): s
AVISO: Foi criada uma tabela com 6 linhas e 6 colunas!
-  1 2 3 4 5 6
1| * * * * *
2| * * * * *
3| * * * * *
4| * * * * *
5| * * * * *
6| * * * * x
Manual:
1 - Escolher Movimento;
2 - Aumentar tamanho;
3 - Guardar jogo e sair;
4 - Mostrar historial do jogo.
Jogador A ->
```

Anuncia o vencedor:



```
C:\WINDOWS\system32\cmd.exe

- 1 2 3 4 5 6
1| - - - - -
2| - - - - -
3| - - - - -
4| - - - - -
5| - - - - -
6| - - - - x

Manual:
1 - Escolher Movimento;
2 - Aumentar tamanho;
3 - Guardar jogo e sair;
4 - Mostrar historial do jogo.
Jogador A -> 1

Coluna = 6
Linha = 6
AVISO: Foi escolhida a posicao [6][6] do mapa.

AVISO: O jogador A apanhou veneno!
O jogador A perdeu!

O jogador B ganhou, PARABENS!!!
Press any key to continue . . .
```