**Guião da aula 6**

Laboratório de Algoritmia I

Laboratórios de Informática II

Ano letivo 2019/20

Last Update: 03/20/2020 10:25:33

**Tarefas a executar esta semana**

As seguintes tarefas devem ser entregues (através do github) até ao dia 20 de Março às 20h00:

* Tornar o repositório Github **privado**
* Documentação do código
* Criação de um prompt
* Adicionar mais comandos ao interpretador
* Implementação da jogada
* Validação das jogadas
* Deteção do fim de jogo

**Repositório privado**

A primeira tarefa desta semana é **tornar o repositório Github privado**. Para isso deverão fazer os seguintes passos:

* Ir aos *Settings* e ir à opção make this repository private
* Convidar o utilizador **equipadocente-la1li21920**
* Mudar o nome do repositório para <UC><Turno><Nº de grupo>

Por exemplo, nomes de repositórios válidos são LA1PL1G5 ou LI2PL7G9.

**Slack**

Vamos criar um conjunto de regras para o **Slack**

* Juntar-se ao [Slack](https://join.slack.com/t/departamentod-l0u3255/shared_invite/zt-cm6rmsky-wUTf2o1el7cn8K8k8q50XA)
* Cada grupo deve criar um canal **privado** com o nome <UC>PL<Nº do Turno>g<Nº de grupo com dois algarismos>
* Cada grupo deve colocar na descrição do canal o url do repositório **Github**
* Cada grupo deve, no seu canal privado, mencionar o docente para que este receba um email quando lhe quiserem perguntar algo diretamente, como por exemplo terem colocado uma nova funcionalidade no github que querem que o docente veja.
* Colocar na descrição do canal do **Slack** o link do **Github**

Eis uma lista dos docentes:

Rui Mendes

@Rui Mendes

Vítor Alves

@Victor Alves

Filipa Ferraz

@Filipa Ferraz

Nuno Oliveira

@NunoO

Francisco Marcondes

@Francisco Marcondes

**Documentação do código**

Pretende-se que crie a documentação do código. Essa documentação poderá ser obtida através do comando Doxygen. Para obter a documentação:

1. Para a instalação:
   1. Instalar o doxygen
   2. Ir para a pasta do código e correr doxygen -g para gerar o ficheiro de configuração
   3. Editar o ficheiro de configuração
2. Correr o comando doxygen para gerar a documentação.

De cada vez que se corre o comando doxygen, são geradas duas pastas: html e latex. Entra-se na pasta html e abre-se o ficheiro index.html.

Segue-se um exemplo de um ficheiro dados.h com os comentários para obter a documentação:

/\*\*

@file dados.h

Definição do estado e das funções que o manipulam

\*/

#ifndef \_\_\_DADOS\_H\_\_\_

#define \_\_\_DADOS\_H\_\_\_

#define BUF\_SIZE 1024

/\*\*

\brief Tipo de dados para os erros

\*/

typedef enum {

OK,

COORDENADA\_INVALIDA,

JOGADA\_INVALIDA,

ERRO\_LER\_TAB,

ERRO\_ABRIR\_FICHEIRO,

} ERROS;

/\*\*

\brief Tipo de dados para a casa

\*/

typedef enum {

UM = '1',

DOIS = '2',

VAZIO = '.',

BRANCA = '\*',

PRETA = '#'

} CASA;

/\*\*

\brief Tipo de dados para as coordenadas

\*/

typedef struct {

int coluna;

int linha;

} COORDENADA;

/\*\*

\brief Tipo de dados para a jogada

\*/

typedef struct {

COORDENADA jogador1;

COORDENADA jogador2;

} JOGADA;

/\*\*

\brief Tipo de dados para as jogadas

\*/

typedef JOGADA JOGADAS[32];

/\*\*

\brief Tipo de dados para o estado

\*/

typedef struct {

/\*\* O tabuleiro \*/

CASA tab[8][8];

/\*\* As jogadas \*/

JOGADAS jogadas;

/\*\* O número das jogadas, usado no prompt \*/

int num\_jogadas;

/\*\* O jogador atual \*/

int jogador\_atual;

/\*\* O nº de comando, usado no prompt \*/

int num\_comando;

/\*\* A coordenada da última jogada \*/

COORDENADA ultima\_jogada;

} ESTADO;

/\*\*

\brief Muda o valor de uma casa

@param e Apontador para o estado

@param c A coordenada

@param V O novo valor para a casa

\*/

void set\_casa(ESTADO \*e, COORDENADA c, CASA V);

/\*\*

\brief Devolve o valor de uma casa

@param e Apontador para o estado

@param c A coordenada

@returns O valor da casa

\*/

CASA get\_casa(ESTADO \*e, COORDENADA c);

/\*\*

\brief Inicializa o valor do estado

Esta função inicializa o valor do estado. Isso implica o tabuleiro ser colocado na posição inicial e todos os campos do estado estarem com o valor por omissão.

@returns O novo estado

\*/

ESTADO \*inicializar\_estado();

#endif

**Criação de um prompt**

Um prompt deve permitir ao utilizador saber o estado do jogo, quantos comados foram introduzidos, quem é o jogador atual e qual é a jogada atual. No exemplo dado abaixo, o implementador optou por colocar

# <número de comandos> PL<1 ou 2 conforme o jogador> (<número da jogada atual>)>

Segue-se um exemplo de um extrato da interação com o jogo:

8 .......2

7 ........

6 ........

5 ....\*...

4 ........

3 ........

2 ........

1 1.......

abcdefgh

# 01 PL1 (0)> d4

8 .......2

7 ........

6 ........

5 ....#...

4 ...\*....

3 ........

2 ........

1 1.......

abcdefgh

# 02 PL2 (1)> d5

8 .......2

7 ........

6 ........

5 ...\*#...

4 ...#....

3 ........

2 ........

1 1.......

abcdefgh

# 03 PL1 (1)> c5

8 .......2

7 ........

6 ........

5 ..\*##...

4 ...#....

3 ........

2 ........

1 1.......

abcdefgh

# 04 PL2 (2)> c6

8 .......2

7 ........

6 ..\*.....

5 ..###...

4 ...#....

3 ........

2 ........

1 1.......

abcdefgh

# 05 PL1 (2)> b5

8 .......2

7 ........

6 ..#.....

5 .\*###...

4 ...#....

3 ........

2 ........

1 1.......

abcdefgh

# 06 PL2 (3)>

Não precisam de seguir exatamente o que se optou aqui mas pretende-se que o prompt seja informativo e tenha, no mímino, a indicação do jogador e do número da jogada.

**Implementação da jogada**

Pretende-se que implemente o resto das funcionalidades da jogada, nomeadamente:

* Verificar se a jogada é válida:
  + Se a casa onde se pretende jogar está vazia
  + Se é vizinha da peça branca
* Colocar uma peça negra na casa atual da peça branca
* Colocar a peça branca na casa atual
* Detetar se o jogo acabou e congratular o jogador correspondente
* O jogo acaba se:
  + A casa branca se encontra no canto inferior esquerdo: ganha **o jogador 1**
  + A casa branca se encontra no canto superior direito: ganha **o jogador 2**
  + A jogada atual colocou a branca num sítio onde o próximo jogador não consegue jogar porque **não há nenhuma casa vizinha livre**: o jogador que colocou a branca ganhou

**Adicionar comandos ao interpretador**

Pretende-se que adicione mais comandos ao intepretador:

* Comando Q
* Comando gr
* Comando ler

Tanto o comando **gr** como o **ler** recebem um argumento: o nome do ficheiro.

Os comandos **gr** e **ler** não precisam de ser implementados na totalidade. Por agora pede-se que o programa só crie o ficheiro com o tabuleiro. A informação relativa às jogadas pode ficar para depois.

Eis as system calls necessárias para criar um ficheiro:

* fopen
* fclose
* fgets
* fscanf
* fprintf
* fputc

Relembro que escrever:

printf("ola\n");

É o mesmo que escrever:

fprintf(stdout, "ola\n");

E por isso pretende-se que reutilizem a função que imprime o tabuleiro para o comando que grava o tabuleiro. Tal implica algo tão simples como substituir printf por fprintf por exemplo. Segue-se um tutorial para [ficheiros](https://www.cprogramming.com/tutorial/cfileio.html), mas existem outros.