```c
/*
 * Leitura e escrita de ficheiros
 * Lê o ficheiro de configuração dos jogadores
 * Escreve o ficheiro de estatisticas
 * Lê a os parametros de um jogador quando énecessario inserir um jogador
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>
#include "file.h"
#include "logic.h"
#include "error.h"

/* Lê o ficheiro de configuração dos jogadores
 * Numero de baralhos e jogadores
 */
Config *read_config(char *filename)
{
   char buffer[MAX_LINE_LEN];

   Config *config = NULL;
   config = (Config *) ecalloc((size_t) 1, sizeof(Config));

   FILE *config_file = efopen(filename, "r");

    //Parametros gerais de configuração:
    //Numero de jogadores e numero de baralhos
   fgets(buffer, MAX_LINE_LEN, config_file);
   sscanf(buffer, "%d-%d", &(config->num_decks), &(config->num_players));

   if (config->num_decks > 8 || config->num_decks < 4){
      fprintf(stderr, "Erro: nmero de baralhos invalido.\n");
      exit(EXIT_FAILURE);
   }

   if (config->num_players > 4 || config->num_players < 1) {
      fprintf(stderr, "Erro: nmero de jogadores invalido.\n");
      exit(EXIT_FAILURE);
   }
    //Leitura dos paramtros de configuracão de cada jogador
   for (int i=0; fgets(buffer, MAX_LINE_LEN, config_file) != NULL && i <
       config->num_players; i++)
      config = read_player(buffer, config, i);

   fclose(config_file);

   return config;
}
```

```c
/*
 *  Leitura dos parametros de configuração de cada jogador
 *  Recebe uma string e separa os parametros de configuração com strtok
 */
Config *read_player(char *line, Config *config, int count)
{
    // strtok separa o buffer no caracter '-'
   char *str = strtok(line, "-");

    //Leitura do tipo do jogador
   if (strcmp(str, "HU") == 0)
      config->player_type[count] = HU;
   else if (strcmp(str, "EA") == 0)
      config->player_type[count] = EA;
   else {
      fprintf(stderr, "Erro: tipo de jogador inválido.\n");
      exit(EXIT_FAILURE);
   }

   str = strtok(NULL, "-");
   if (strlen(str) > MAX_PLAYER_NAME) {
      fprintf(stderr, "Erro: nome do jogador demasiado grande (Máx. 8 caracteres).\n");
      exit(EXIT_FAILURE);
   }
   strcpy(config->player_names[count], str);

    //Ultimo segmento da string
    str = strtok(NULL, "\0");
   sscanf(str, "%d-%d", &config->money[count], &config->bets[count]);
   if (config->money[count] < 10 || config->money[count] > 500) {
      fprintf(stderr, "Erro: valor inicial de dinheiro inválido.\n");
      exit(EXIT_FAILURE);
   }

   if (config->bets[count] < 2 ||
      config->bets[count] > config->money[count] / 4) {
      fprintf(stderr, "Erro: valor da aposta invalido!\n");
      exit(EXIT_FAILURE);
   }

   return config;
}

/* Vai buscar uma linha a stdin.
 * Modifica o buffer por referência.
 * O buffer fica vazio se o fgets() der overflow ou se a input for vazia.
 * Senão, o buffer fica com a string de input, sem o \n.
 */
void get_line(char buffer[MAX_PLAYER_NAME+2])
{
   int newline = 0;
```

```
   int c = 0;
   fgets(buffer, MAX_PLAYER_NAME+2, stdin);

   // localização do \n
   newline = (int) strcspn(buffer, "\n");

   // se não existir (ou seja, newline éo comprimento da string inserida),
   // sabemos que a string de stdin émaior que o buffer pode conter.
   if (newline == MAX_PLAYER_NAME+1) {
      strcpy(buffer, "");
      // Consumir o resto do buffer de stdin
      while ((c = getchar()) != '\n' && c != EOF);
      return;
   }
   // se existir, substituir por \0.
   // neste caso se buffer estiver vazio, permanece vazio.
   else
      buffer[newline] = '\0';
}

//  Ler novo valor da aposta a partir stdin
void get_new_bet(List *players)
{
   char buffer[MAX_PLAYER_NAME+2] = {0}; // newline + nullbyte
   bool correct = false;
   List *aux = players->next;
   Player *cur_player = NULL;

   printf("Insira o nome do jogador a modificar a aposta: ");
   get_line(buffer);

   if (buffer[0] == '\0') {
      puts("Jogador não encontrado. Tente novamente primindo a tecla <b>.");
      return;
   }

   correct = false;
    //Verificar se o jogador existe
   while (aux && !correct) {
      cur_player = (Player *) aux->payload;
      if (strcmp(buffer, cur_player->name) == 0 && !correct)
         correct = true;
      else
         aux = aux->next;
   }

   if (!aux) {
      puts("Jogador não encontrado. Tente novamente primindo a tecla <b>.");
      return;
   }
```

```
   correct = false;
   cur_player = (Player *) aux->payload;
   long new_bet = 0;
   do {
      printf("Insira o novo valor da aposta do jogador %s: ", cur_player->name);
      get_line(buffer);

      if (buffer[0] == '\0')
         puts("Nova aposta inválida.");
      else {
         new_bet = strtol(buffer, NULL, 10);
         // o dinheiro do jogador está (essencialmente) garantido
         // de estar abaixo de INT_MAX (a não ser que se jogue mesmo muito)
         // fazendo com bet pertencente a [1, money]
         if (new_bet > cur_player->money || new_bet < 1)
            printf("Nova aposta inválida [1-%d].\n", cur_player->money);
         else
            correct = true;
      }
   } while (!correct);

   cur_player->bet = (int) new_bet;
}



/*
 * Obter o valor do dinheiro, tipo, nome e aposta do jogador
 * Pede ate obter um valor correto
 */
Player *get_new_player(int pos)
{
   char buffer[MAX_PLAYER_NAME+2] = {0};
   bool correct = false;
   Type type = HU;
   char name[MAX_PLAYER_NAME+1] = {0};
   int money = 0;
   int bet = 0;
   long money_tmp = 0;
   long bet_tmp = 0;
   Player *new_player = NULL;

   printf("Escolheu o %d lugar.\n", pos);

    correct = false;
   do {
      printf("Introduza o tipo do jogador [HU ou EA]: ");
      get_line(buffer);

      if (buffer[0] == '\0')
         puts("Tipo de jogador inválido [HU ou EA].");
```

```
      else {
         if (strcmp(buffer, "HU") == 0) {
            type = HU;
            correct = true;
         }
         else if (strcmp(buffer, "EA") == 0) {
            type = EA;
            correct = true;
         }
         else
            puts("Tipo de jogador inválido (HU ou EA).");
      }
   } while (!correct);

   correct = false;
   do {
      printf("Introduza o nome do jogador [máx. 8 carac.]: ");
      get_line(buffer);

      if (buffer[0] == '\0')
         puts("Nome do jogador inválido. Este tem no máximo 8 caracteres.");
      else {
         strcpy(name, buffer);
         correct = true;
      }
   } while (!correct);

   correct = false;
   do {
      printf("Introduza o dinheiro do jogador: ");
      get_line(buffer);


      if (buffer[0] == '\0')
         puts("Dinheiro inválido.");
      else {
         money_tmp = strtol(buffer, NULL, 10);
         if (money_tmp <= 1 || money_tmp > INT_MAX)
            printf("Quantidade de dinheiro inválida [de 1 a %d].\n", INT_MAX);
         else {
            correct = true;
            money = (int) money_tmp;
         }
      }
   } while (!correct);


   correct = false;
   do {
      printf("Introduza a aposta do jogador: ");
      get_line(buffer);
```

```
      if (buffer[0] == '\0')
         puts("Aposta inválida.");
      else {
         bet_tmp = strtol(buffer, NULL, 10);
         if (bet_tmp > money_tmp || bet_tmp <= 0)
            printf("Aposta inválida [de 1 a %d].\n", money);
         else {
            correct = true;
            bet = (int) bet_tmp;
         }
      }
   } while (!correct);

    // Alocar espaço para o jogador e escrever a configuração
   new_player = (Player *) ecalloc((size_t) 1, sizeof(Player));

   new_player->ingame = true;
   new_player->type = type;
   strcpy(new_player->name, name);
   new_player->money = money;
   new_player->bet = bet;

   return new_player;
}


// Escrever o ficheiro de estatisticas
void write_stats(List *players, Player *house, List *old_players)
{
    FILE *stats = NULL;
    stats = efopen("stats.txt" , "w");

    fprintf(stats, "Jogador\t\tTipo\tJogos\tVitorias\tEmpates\tDerrotas\tDinheiro\n");

   write_stats_players(stats, players);
   write_stats_players(stats, old_players);

    // O dinheiro da casa esta em modulo. E indicado se a casa perdeu ou ganhou
    // dinheiro
   if (house->money < 0)
      fprintf(stats, "A casa perdeu: %d \n", -1*house->money);
   else if (house->money > 0)
      fprintf(stats, "A casa ganhou: %d \n", house->money);
   else if (house->money == 0)
      fprintf(stats, "A casa não ganhou nem perdeu dinheiro.\n");

   fclose(stats);
}

//  Escrever as estatisticas dos jogadores
```

```
void write_stats_players(FILE *stats, List *players)
{
   List *aux = players->next;
   Player *cur_player = NULL;
    while (aux) {
        cur_player = (Player *) aux->payload;
        if (cur_player->type == VA) {
            aux = aux->next;
            continue;
        }
        fprintf(stats, "%s\t", cur_player->name);
        if (strlen(cur_player->name) < 8)
            fprintf(stats, "\t");
        if (cur_player->type == EA)
            fprintf(stats, "EA\t");
        else if (cur_player->type == HU)
            fprintf(stats, "HU\t");
        fprintf(stats, "%d\t", cur_player->wins+cur_player->losses+cur_player->ties);
        fprintf(stats, "%d\t\t", cur_player->wins);
        fprintf(stats, "%d\t", cur_player->ties);
        fprintf(stats, "%d\t\t", cur_player->losses);
        fprintf(stats, "%d  \n", cur_player->money);
        aux = aux->next;
    }
}
```