```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 é necessario 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:
    // Número 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: número de baralhos invalido.\n");
        exit(EXIT_FAILURE);
    }

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

    fclose(config_file);

    return config;
}

/*
 *  Leitura dos parâmetros de configuração de cada jogador
 *  Recebe uma string e separa os parâmetros de configuração utlizando 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);
    }
```

```
69
70      str = strtok(NULL, "-");
71      if (strlen(str) > MAX_PLAYER_NAME) {
72          fprintf(stderr, "Erro: nome do jogador demasiado grande (Máx. 8 caract
    eres).\n");
73          exit(EXIT_FAILURE);
74      }
75      strcpy(config->player_names[count], str);
76
77      //Ultimo segmento da string
78      str = strtok(NULL, "\0");
79      sscanf(str, "%d-%d", &config->money[count], &config->bets[count]);
80      if (config->money[count] < 10 || config->money[count] > 500) {
81          fprintf(stderr, "Erro: valor inicial de dinheiro inválido.\n");
82          exit(EXIT_FAILURE);
83      }
84
85      if (config->bets[count] < 2 ||
86          config->bets[count] > config->money[count] / 4) {
87          fprintf(stderr, "Erro: valor da aposta invalido!\n");
88          exit(EXIT_FAILURE);
89      }
90
91      return config;
92  }
93
94  /* Vai buscar uma linha a stdin.
95   * Modifica o buffer por referência.
96   * O buffer fica vazio se o fgets() der overflow ou se a input for vazia.
97   * Senão, o buffer fica com a string de input, sem o \n.
98   */
99  void get_line(char buffer[MAX_PLAYER_NAME+2])
100 {
101     int newline = 0;
102     int c = 0;
103     fgets(buffer, MAX_PLAYER_NAME+2, stdin);
104
105     // localização do \n
106     newline = (int) strcspn(buffer, "\n");
107
108     // se não existir (ou seja, newline é o comprimento da string inserida),
109     // sabemos que a string de stdin é maior que o buffer pode conter.
110     if (newline == MAX_PLAYER_NAME+1) {
111         strcpy(buffer, "");
112         // Consumir o resto do buffer de stdin
113         while ((c = getchar()) != '\n' && c != EOF);
114         return;
115     }
116     // se existir, substituir por \0.
117     // neste caso se buffer estiver vazio, permanece vazio.
118     else
119         buffer[newline] = '\0';
120 }
121
122 //  Ler novo valor da aposta a partir stdin
123 void get_new_bet(List *players)
124 {
125     char buffer[MAX_PLAYER_NAME+2] = {0}; // newline + nullbyte
126     bool correct = false;
127     List *aux = players->next;
128     Player *cur_player = NULL;
129
130     printf("Insira o nome do jogador a modificar a aposta: ");
131     get_line(buffer);
132
133     if (buffer[0] == '\0') {
134         puts("Jogador não encontrado. Tente novamente primindo a tecla <b>.");
135         return;
136     }
137
```

```
138      correct = false;
139      //Verificar se o jogador existe
140      while (aux && !correct) {
141          cur_player = (Player *) aux->payload;
142          if (strcmp(buffer, cur_player->name) == 0 && !correct)
143              correct = true;
144          else
145              aux = aux->next;
146      }
147
148      if (!aux) {
149          puts("Jogador não encontrado. Tente novamente primindo a tecla <b>.");
150          return;
151      }
152
153      correct = false;
154      cur_player = (Player *) aux->payload;
155      long new_bet = 0;
156      do {
157          printf("Insira o novo valor da aposta do jogador %s: ", cur_player->na
     me);
158          get_line(buffer);
159
160          if (buffer[0] == '\0')
161              puts("Nova aposta inválida.");
162          else {
163              new_bet = strtol(buffer, NULL, 10);
164              // o dinheiro do jogador está (essencialmente) garantido
165              // de estar abaixo de INT_MAX (a não ser que se jogue mesmo muito)
166              // fazendo com bet pertencente a [1, money]
167              if (new_bet > cur_player->money || new_bet < 1)
168                  printf("Nova aposta inválida [1-%d].\n", cur_player->money);
169              else
170                  correct = true;
171          }
172      } while (!correct);
173
174      cur_player->bet = (int) new_bet;
175  }
176
177
178
179  /*
180   * Obter o valor do dinheiro, tipo, nome e aposta do jogador
181   * Pede ate obter um valor correto
182   */
183  Player *get_new_player(int pos)
184  {
185      char buffer[MAX_PLAYER_NAME+2] = {0};
186      bool correct = false;
187      Type type = HU;
188      char name[MAX_PLAYER_NAME+1] = {0};
189      int money = 0;
190      int bet = 0;
191      long money_tmp = 0;
192      long bet_tmp = 0;
193      Player *new_player = NULL;
194
195      printf("Escolheu o %dº lugar.\n", pos);
196
197      correct = false;
198      do {
199          printf("Introduza o tipo do jogador [HU ou EA]: ");
200          get_line(buffer);
201
202          if (buffer[0] == '\0')
203              puts("Tipo de jogador inválido [HU ou EA].");
204          else {
205              if (strcmp(buffer, "HU") == 0) {
206                  type = HU;
```

```
207                 correct = true;
208             }
209             else if (strcmp(buffer, "EA") == 0) {
210                 type = EA;
211                 correct = true;
212             }
213             else
214                 puts("Tipo de jogador inválido (HU ou EA).");
215         }
216     } while (!correct);
217
218     correct = false;
219     do {
220         printf("Introduza o nome do jogador [máx. 8 carac.]: ");
221         get_line(buffer);
222
223         if (buffer[0] == '\0')
224             puts("Nome do jogador inválido. Este tem no máximo 8 caracteres.")
    ;
225         else {
226             strcpy(name, buffer);
227             correct = true;
228         }
229     } while (!correct);
230
231     correct = false;
232     do {
233         printf("Introduza o dinheiro do jogador: ");
234         get_line(buffer);
235
236
237         if (buffer[0] == '\0')
238             puts("Dinheiro inválido.");
239         else {
240             money_tmp = strtol(buffer, NULL, 10);
241             if (money_tmp <= 1 || money_tmp > INT_MAX)
242                 printf("Quantidade de dinheiro inválida [de 1 a %d].\n", INT_M
    AX);
243             else {
244                 correct = true;
245                 money = (int) money_tmp;
246             }
247         }
248     } while (!correct);
249
250
251     correct = false;
252     do {
253         printf("Introduza a aposta do jogador: ");
254         get_line(buffer);
255
256         if (buffer[0] == '\0')
257             puts("Aposta inválida.");
258         else {
259             bet_tmp = strtol(buffer, NULL, 10);
260             if (bet_tmp > money_tmp || bet_tmp <= 0)
261                 printf("Aposta inválida [de 1 a %d].\n", money);
262             else {
263                 correct = true;
264                 bet = (int) bet_tmp;
265             }
266         }
267     } while (!correct);
268
269     // Alocar espaço para o jogador e escrever a configuração
270     new_player = (Player *) ecalloc((size_t) 1, sizeof(Player));
271
272     new_player->ingame = true;
273     new_player->type = type;
274     strcpy(new_player->name, name);
```

```
275      new_player->money = money;
276      new_player->bet = bet;
277
278      return new_player;
279  }
280
281
282  // Escrever o ficheiro de estatisticas
283  void write_stats(List *players, Player *house, List *old_players)
284  {
285      FILE *stats = NULL;
286      stats = efopen("stats.txt" , "w");
287
288      fprintf(stats, "Jogador\t\tTipo\tJogos\tVitorias\tEmpates\tDerrotas\tDinhe
     iro\n");
289
290      write_stats_players(stats, players);
291      write_stats_players(stats, old_players);
292
293      // O dinheiro da casa esta em modulo. E indicado se a casa perdeu ou ganho
     u
294      // dinheiro
295      if (house->money < 0)
296          fprintf(stats, "A casa perdeu: %d ¿\n", -1*house->money);
297      else if (house->money > 0)
298          fprintf(stats, "A casa ganhou: %d ¿\n", house->money);
299      else if (house->money == 0)
300          fprintf(stats, "A casa não ganhou nem perdeu dinheiro.\n");
301
302      fclose(stats);
303  }
304
305  //  Escrever as estatisticas dos jogadores
306  void write_stats_players(FILE *stats, List *players)
307  {
308      List *aux = players->next;
309      Player *cur_player = NULL;
310      while (aux) {
311          cur_player = (Player *) aux->payload;
312          if (cur_player->type == VA) {
313              aux = aux->next;
314              continue;
315          }
316          fprintf(stats, "%s\t", cur_player->name);
317          if (strlen(cur_player->name) < 8)
318              fprintf(stats, "\t");
319          if (cur_player->type == EA)
320              fprintf(stats, "EA\t");
321          else if (cur_player->type == HU)
322              fprintf(stats, "HU\t");
323          fprintf(stats, "%d\t", cur_player->wins+cur_player->losses+cur_player-
     >ties);
324          fprintf(stats, "%d\t\t", cur_player->wins);
325          fprintf(stats, "%d\t", cur_player->ties);
326          fprintf(stats, "%d\t\t", cur_player->losses);
327          fprintf(stats, "%d ¿\n", cur_player->money);
328          aux = aux->next;
329      }
330  }
```