# LEGYEN ÖN IS MILLIOMOS

## félkész program - dokumentáció

Bánszky Koppány - KK4UWP

## Program fájlszerkezete:

```
main.c
properties.h

'libs/'
    debugmalloc.c
    din_list_manager.c
    din_list_manager.h
    econio.c
    econio.h
    manage_stat_file.c
    manage_stat_file.h
    read_data.c
    read_data.h
    read_data_from_row_file.c
    read_data_from_row_file.h
    utility.c
    utility.h

'pages/'
    choose_question.c
    choose_question.h
    main_menu.c
    main_menu.h
    name_page.c
    name_page.h
    pause_page.c
    pause_oage.h
    row_question_page.c
    row_question_page.h
    settings_menu.c
    settings_menu.h
    stat_page.c
    stat_page.h
```

A program a `debugmalloc` és a econio könyvtárakra épül, így ezeknek a forrásfájljait, a fentebb mutatott módon be kell építeni a fájlszerkezetbe.

### main.c

Általános beállítások definiálása.

A program felületek kezelése egységesített felülettel a funkciók meghívására. Sablon:

```
struct Page_Func {
   struct Page_Func (*function)(void*); //Meghívandó funkció, ez minden esetben egy
```

```
megjelenített felület
   void *param; //A felületnek szükséges paraméter. Ez bármilyen pointer lehet, ahol
szükség van rá, definiálva van egy struct is a funkcióhoz.
   int key; //A lenyomott billentyű, amivel indította a felhasználó a funkciót
}
Page_Func function(void*)

//Ha megadott funkció NULL, a program véget ér és visszatér 0-val a op-rendszernek
```

### properties.h

Általános használt struct-ok definiálása

```
struct Properties {
   ...
   unsigned long int time_sec; //játékidő, másodpercben
   unsigned long int timestamp; //segéd változó, adott menet kezdeti indejét tárolja
   int used_hel; //adott játékban a felhasznált segítségeket tárolja (val 1 = Egyik se, val
2 = felezés, val 3 = szavazás, val 6 = mindkettő)
};
struct Page_Func;
```

## libs/ fájlok

### din_list_manager.c

```
    /* Felszabadítja a Choose_By_Question láncolt listát
         @param Choose_By_Questoin[] első eleme
      */
    void free_Choose_Question_list(Choose_By_Question *elso);

    /* Felszabadítja a nehézségi szint csoportosítva tárolt listát
         @param Difficulties_list *ptr[]
    */
    void free_Difficulty_list(Difficulties_list *list_for_different_difficulties[]);

    /*
      Felszabadítja a Row_Question láncolt listát
         @param Row_Question[] első eleme
    */
    void free_Row_Question_list(Row_Question *elso);

    /*
    Difficulties_list láncolt lista hosszát számolja meg
    @param Difficultes_list *első elem
    return int hossz
    */
    int din_length_diff(Difficulties_list *list);

   /*
    Row_Question láncolt lista hosszát számolja meg
    @param Row_Question *első elem
    return int hossz
   */
```

```
   int din_length_row(Row_Question *list);

   /*
    Kiválaszt egy kérdést a megadott nehézség szerint
    @param int nehézség, int kérdés sorszáma, Difficulties_list egész nehézségekre bontott
lista
    return a kiválasztott kérdés
   */
   Choose_By_Question *get_question_by_number(int difficulty, int num, Difficulties_list
*list_for_different_difficulties[]);

   /*
    Kiválaszt egy sorbarendezendő kérdést a megadott sorszám alapján
    @param int sorszám, Row_Questoin *ptr a lista első eleméhez
    return kérdés
   */
   Row_Question *get_row_question_by_number(int num, Row_Question *first);
```

manage_stat_file.c

Játékosok statisztikáit kezelő funkciók vannak a fileban

```
/*
 Beolvassa a korábban mentett statisztikákat
 A program futása alatt elég egyszer, az elején meghívni
 return Statistict* dinamikus lista első eleme, ha nincs, vagy nem tudja beolvasni a filet,
NULL
*/
Statistics *stat_file();

/*
 Megváltoztatja a statisztikáit az adott játékosnak, a beállítások alapján
 @param Regular_Data *ptr
 return{
   Ha a létezett az adott játékos és változtatta az adatait: int 1
   Ha egy játékos se volt még az adatbázisban: int -1
   Ha új játékost vitt fel a létező adatok közé: int 2
 }
 Fontos: ez még nem írja ki fájlba a változott adatokat
 */
int change_stat(void *param);

/*
 Kiírja az összes statisztikai adatot a source/rangsor.csv fájlba
 @param Statistics* a lista első eleme
 return {
   Ha nem lehet megnyitni a fájlt: int -1
   Ha üres a lista: int -1
   Egyébként: int a kiírt játékos statisztikák száma
 }
*/
int write_stat_file(Statistics *first);
```

read_data.c

Az adatok fájlból való beolvasásáért és tárolásáért felel

```
/*
 Megnézi, hogy egy adott file létezik-e?! Ha nem, bekér egy, a szükséges filere mutató
útvonalat
 @param alap útvonal, a file típusa [LOIM / SOROK]
 return új vagy régi útvonal, ami helyes
*/
int validate_file(char *path, File_type fileType);

/* Betölti a feleletválasztós kérdéseket, egy Choose_By_Question struktúrába. Az adatokat
egy láncolt listába fűzi */
/* @param útvonal a megfelelő forrásfilehoz (string), Difficulies_list nehézség szerint
bontott lista
 return értéke a láncolt lista első eleme */
Choose_By_Question *load_loim_in_dinlist(char *pathToFile, Difficulties_list
*list_for_different_difficulties[]);
```

## read_data_from_row_file.c

A sorkérdések beolvasásáért felel

```
/*
Beolvassa a megadaott sorok.csv fileból a kérdéseket és eltárolja láncolt listában
@param megadott útvonal a filehoz
return Row_Question* a lista első eleme
*/
Row_Question *create_row_question_list(char path_to_file[]);
```

## utility.c

Egyéb, segítő functionök helye, több helyen is előfordulhatnak a programban

```
/* Legyen ön is milliomos felirat megjelenítése */
void display_title();

/* adott hosszúságú üres karakterekből álló stringet tesz a első paraméterbe
 * Visszatérési értéke a sikeres üreskarakterek generálásának száma*/
int create_blank_space(char dest[], int length);

/* Megjeleníti a Menu_Item listában lévő elemeketet a képernyőn, szép elválasztással
 * Visszatérési értéke a sikeresen megjelenített menüelemek száma*/
int display_menu_items(Menu_Item item_list[], int length);

/* A Menu_Item lista alapján kezeli a billentyű lenyomásokat
 * return Page_Func{
   Page_Func következő oldal,
   void* annak szükséges paraméterei
   int lenyomott billentyű
 }
   */
Page_Func wait_menu_input(Menu_Item item_list[], int length);
```

```
/* Meghvásakor, ha van már beállítva játékosnév, elindítja a játékot, ha nincs bekér egy
nevet */
/* @param Regular_Data* a minden oldalon használt adatok, ez a main.c-ben van declarálva*/
Page_Func start_game();

/* Felszabadítja a a dinamikusan foglalt listákat
   @param void* -> Exit_Params*
   return Page_Func{
      NULL,
      NULL,
      200
   }
*/
int app_exit();

/* Feltölt egy int listát a következőképp:
1 elem: megadott stringben a karakter szám, adott (str_end) karakterig, vagy \n karakterig
*/
/* Paraméterei: string és a elválasztó karakter, return értéke az int lista. !!!FEL KELL
SZABADÍTANI!!! */
int *count_chars_between_char(char *str, char str_end);

/* A kértnek megfelelően tagol egy listában megadott szöveget
 @param char **sor, int sor listának hossza, Alignments a kívánt tagolás
 return char* a tagolt szöveg
 fel kell szabadítani!
*/
char *justify_content(char **line, int list_length, Aligments align);

/*
 A túl hosszú szöveget külön sorokra bontja az utolsó szóközök mentén
 @param string
 return char* tagolt szöveg
 fel kell szabadítani
*/
char *break_line_at_last_space(char *str);

/*
 Egy listában adott szöveget összeköt egy adott stringgel
 @param char **lista a szövegekkel, int lista hossza, char* összekötő szöveg
 return char* összekötött szöveg
 fel kell szabadítani
*/
char *join(char **list, int length, char* intersection);

/*
 Másodpercben megadott időt ad vissza "00 min 00 sec" formátumban
 @param int másodpercek
 return char*
 fel kell szabadítani
*/
char *readable_time(unsigned long int sec);
```

## pages/ fájlok

choose_question.c

```
/*
 A feleletválasztós kérdések menüje.
 @param Choose_question_Params* {
   Regular_Data*,
   int kérdés száma: Ha első -1, ha nem 0
 }
 return Page_Func
*/
Page_Func choose_question(void *param);


/*
 Kezeli és mutatja a felhasznált segítséget
 @param Show_help_Params* {
   Regular_Data*,
   int kérdés száma - lineáris
 }
 return Page_Func
*/
Page_Func show_help(void* param);
```

main_menu.c

```
/* Belépteti a játékot a főmenübe */
/* @param Main_menu_Params*
 return Page_Func
*/
Page_Func main_menu(void* param);
```

Tartozik hozzá egy header file is

name_page.c

```
/* Bekér a felhasználótól egy játékos nevet
 a nevet tároló stringet kilépéskor fel kell szabadítani
 a nevet a beállításokba menti rögtön
 @param Name_page_Params*
 return Page_Func
*/
Page_Func name_page(void *param);
```

pause_page.c

A játék különböző pontjai között egy menüt mutat, a kontextusnak megfelelően

```
/*
 @param Pause_page_Params*
 return Page_Func
```

```
*/
Page_Func pause_page(void *param);
```

### row_question_page.c

Menü a sorbarendezős kérdéseknek

```
/*
Megjeleníti a menüt
   @param Regular_Data*
   return Page_Func
*/
Page_Func row_question_page(void *param);

/*
Teszteli a választ
@param Regular_Data*
return Page_Func
*/
Page_Func test_row_answer(void *param);
```

### settings_menu.c

```
/* Megjeleníti a beállítások menüpontot
   @param Settings_menu_Param*
   return Page_Func */
Page_Func settings_menu(void *param);

/* A szavazás sikerességének esélyein lépked végig, még nem menti a végleges beállításokat.
   @param Settings_menu_Param*
   return Page_Func */
Page_Func set_vote_chance(void *param);


/* A nehézség beállításait lépteti, még nem menti véglegesen.
   @param Settings_menu_Param*
   return Page_Func */
Page_Func set_difficulty(void *param);

/* Véglegesen elmenti a beállításokat
   @param Settings_menu_Param*
   return Page_Func */
Page_Func save(void *param);


/* Visszalép a főmenübe, nem ment semmit
   @param Settings_menu_Param*
   return Page_Func */
Page_Func back(void *param);
```

### stat_page.c

A statisztikák menüjét kezeli

```
/* Megjeleníti a statisztikák rekordjait sorban
@param Statistics* a láncolt lista első eleme
minden kiírt lista elemet fel is szabadít
*/
void printTableRow(Statistics *first);

/* Kiválogatja és idő szerint növekvő sorba rendezi a statisztikákat.
@params Settings_menu_Param* a szűrés feltétele miatt, bool szűrés nélkül rendezzen (true)
return Statistics* a rendezett lista első eleme
fel kell szabadítani a listát
*/
Statistics *filteredSorted(Settings_menu_Param *params, bool unfiltered);

/*
A megadott feltételek szerint elindítja a listázást és a megjelenítést
@param Settings_menu_Param*
return Page_Func
*/
Page_Func search(void *param);

/* Visszatér a főmenübe
@param Settings_menu_Param*
return Page_Func
*/
Page_Func backToMain(void *param);

/*
Beállítja a nehézség szűrőjét
@param Settings_menu_Param*
return Page_Func
*/
Page_Func find_difficulty(void *param);

/*
Beállítja a szavazás szűrőjét
@param Settings_menu_Param*
return Page_Func
*/
Page_Func find_vote_chance(void *param);

/*
Megjeleníti a menüt
@param Settings_menu_Param*
return Page_Func
*/
Page_Func stat_menu_all(void *param);
```