Programozói dokumentáció

I. Projekt felépítése

A programom egy „Legyen Ön is Milliomos” típusú quiz játék, mely C11-es szabvány alapján lett megírva CLion fejlesztőkörnyezetben, macOS Big Sur operációs rendszeren. Csak az alap könyvtárakat használja. Működéséhez 2 db *.csv* típusú fájl szükséges. Egyik a felelet-választós kérdéseket kell tartalmazza, neve: *felelet_valasztos.csv* (minimum 15 db kérdésre szükség van a megfelelő működéshez), másik pedig a sorkérdéseket, neve: *sorkerdesek.csv* (minimum 1 db kérdésre szükség van a megfelelő működéshez). A minimum követelményeken felül bármekkora méretű adatbázisból képes a program működni.

A *felelet_valasztos.csv*-ben található adatok szabványa: egy sorban egy kérdéshez tartozó minden adatnak megtalálhatónak kell lennie az alább felsorolt sorrendben, a különböző adatrészei egy adott kérdésnek a *.csv* típusú fájlok jellegeként pontosvesszővel kell elválasztva legyenek. A szöveges adatok akár ékezetes karaktereket is tartalmazhatnak. Az egy kérdéshez tartozó különböző adatok sorrendben:

- Nehézségi szint: 1-15 közötti érték számmal megadva.
- Kérdés: szöveg.
- ’A’ válaszlehetőség: szöveg.
- ’B’ válaszlehetőség: szöveg.
- ’C’ válaszlehetőség: szöveg.
- ’D’ válaszlehetőség: szöveg.
- Helyes válasz: karakter, amely A, B, C vagy D értékek lehetnek.
- Kategória: bármilyen kategória megadható, lényeg, hogy egy adatbázisban azonos stílussal legyenek megadva ezek. (pl.: BIOLÓGIA - csupa nagybetűkkel).

A *sorkerdesek.csv*-ben található adatok szabványa: alapjaiban azonos a *felelet_valasztos.csv*-nél leírtakhoz, csak mások a kérdésekhez tartozó adatok:

- Kérdés: szöveg.
- 'A' válaszlehetőség: szöveg.
- 'B' válaszlehetőség: szöveg.
- 'C' válaszlehetőség: szöveg.
- 'D' válaszlehetőség: szöveg.
- Helyes sorrend: szöveg, amely A, B, C vagy D értékek lehetnek a megfelelő sorrendben egymástól nem elválasztva, tehát egybe írva.
- Kategória: bármilyen kategória megadható, lényeg, hogy egy adatbázisban azonos stílussal legyenek megadva ezek, és a *felelet_valasztos.csv* szabványával azonos.

A program ezeken az előre megadott fájlokon kívül még létrehoz 2 másik *.txt* típusú fájlt a játék során.

Egy *save.txt*-t, amelyben egy adott játékhoz tartozó minden adat megtalálható a játék során használatos összes kérdéssel együtt. Az alábbiakban felsorolom a felépítését:

- Első sor: a verziószáma a mentésfájlnak, hogy esetleg egy újabb verziójú játék esetén, amely már más stílusú mentésfájlból dolgozik visszamenőleg egy korábbi stílusú mentésből is tudjon dolgozni. (ez a játék esetén mindig: „MILLIOMOS v1.0")
- Második sor: a *Milliomos* nevű struktúrához tartozó adatokat tartalmazza pontosvesszővel elválasztva (részletes leírását lásd később). Sorrendben az alábbiak:
    a. Van-e sorkérdés: 0 vagy 1 lehet az értéke, értelemszerűen 0 ha nincsen a mentésben sorkérdés és 1 ha van.
    b. Hány lehetősége van még sorkérdés választ tippelni: szám (0-3).
    c. Van-e felezés segítsége: 0 vagy 1 lehet az értéke.
    d. Van-e közönség szavazás segítsége: 0 vagy 1 lehet az értéke.
    e. Játékban eltelt percek száma (0-60).
    f. Játékban eltelt másodpercek száma (0-60).
    g. Elért pontszám: *long* típusú szám.
    h. Játék nehézségi szintje: szám (1-3).

- Ezeket követő sorok a kérdések, amelyek közül az első sor a sorkérdés (a már fent leírt formátumként), ha van, utána pedig a felelet-választós kérdések találhatóak, mindegyik külön sorban azzal a plusszal a fentiekkel leírtakkal, hogy az első adat a kérdések sorrendben megindexelve. A többi adat ugyanaz a formátummal van megadva, ahogy az eredeti dokumentumban van az indexelés után.

A másik fájl, amit a program létrehoz, egy *dicsoseglista.txt*, amely az eddigi elért top 10 eredményt tárolja. Felépítése:

- Első sor: a verziószáma a dicsőségtáblának, hogy esetleg egy újabb verziójú játék esetén, amely már más stílusú dicsőséglistából dolgozik visszamenőleg egy korábbi stílusú fájlból is tudjon dolgozni. (ez a játék esetén mindig: „DICSOSEGLISTA v1.0")
- Az ezt követő sorok a top 10 eredményt elért játékos adatai, játékosonként külön sorban. Nem feltétlen van elmentve 10 játékos a dicsőséglistában, az elején még csak annyi, amennyi a legutóbbi törlés óta játszott a játékkal. Az adatok itt is pontosvesszővel vannak elválasztva egymástól, sorrendjük a következő:
    a. Játék során elért pontszám: *long* típusú szám.
    b. Játékos neve (csak egy szavas lehet): szöveg.
    c. A játék nehézségi szintje: szám (1-3).
    d. Játék ideje percekben (0-60).
    e. Játék ideje másodpercekben (0-60).

A teljes játékom a *main.c*-n kívül 3 különböző modulra van felosztva, melyekhez mind megtalálható az azonos nevű header fájl is.

A *main.c* csupán a menüt kezeli, illetve új játék esetén meghívja a teljes játék legenerálásához szükséges függvényeket.

A *lista.c* modulban találhatóak a listakezelésekhez használt függvényeim, illetve a sztringeknek dinamikusan foglaló függvényem és a felelet-választós és sorkérdés struktúrák felszabadítására használt függvényeim.

A *file.c* modulban találhatóak a különböző fájloknak és azok struktúráinak a kezelésére használt függvényeim, illetve egy buborékrendező és a dicsőségtábla kiírásához használt függvény.

A *jatek.c* modulban, pedig a játék teljes lebonyolításához használt függvényeim találhatóak meg.

II. Adatszerkezetek

Az összes struktúrámat a *lista.h* fájlban tárolom, mert az a legalapvetőbb modulom, ami az összes többi működéséhez szükséges. A struktúrákban a pointer típusú adatok, azért így vannak inicializálva, hogy dinamikusan lehessen azokat tárolni.

- *FelValasz* struktúra: a felelet-választós kérdések adatait tárolja egy lista típusú struktúrában, azért, hogy utána könnyen lehessen random kiszedegetni az adatbázisból a játék során használt kérdéseket. Részei:
    a. *int id:* indexe a kérdésnek.
    b. *int lvl:* nehézségi szintje a kérdésnek (1-15).
    c. *char *kerdes:* kérdésre mutató pointer.
    d. *char *v_A:* 'A' válaszlehetőségre mutató pointer.
    e. *char *v_B:* 'B' válaszlehetőségre mutató pointer.
    f. *char *v_C:* 'C' válaszlehetőségre mutató pointer.
    g. *char *v_D:* 'D' válaszlehetőségre mutató pointer.
    h. *char valasz:* helyes válasznak a karaktere.
    i. *char *kategoria:* kategóriára mutató pointer.
    j. *struct FelValasz *next:* következő listaelemre mutató pointer.
- *Sorkerdes* struktúra: a sorkérdések adatait tárolja egy egyszerű struktúrában. Részei:
    a. *char *kerdes:* kérdésre mutató pointer.
    b. *char *v_A:* 'A' válaszlehetőségre mutató pointer.
    c. *char *v_B:* 'B' válaszlehetőségre mutató pointer.
    d. *char *v_C:* 'C' válaszlehetőségre mutató pointer.
    e. *char *v_D:* 'D' válaszlehetőségre mutató pointer.
    f. *char valasz[5]:* helyes sorrendnek a sztringje.
    g. *char *kategoria:* kategóriára mutató pointer.

- *Milliomos* struktúra: az játék adatait tárolja egy egyszerű struktúrában. Részei:
    a. *int tries:* a sorkérdésre hátralévő tippek számát tárolja (0-3).
    b. *int sec:* az eltelt másodperceket tárolja (0-60).
    c. *int min:* az eltelt perceket tárolja (0-60).
    d. *long money:* a játék során az egyes kérdések által összeszedett pénzmennyiséget tárolja.
    e. *int diff:* a játék nehézségi szintje (1-3).
    f. *FelValasz *fv:* a felelet-választós kérdések listájának első elemére mutató pointer.
    g. *Sorkerdes *sor:* a sorkérdésre mutató pointer.
    h. *bool skerdes:* van-e még hátra sorkérdés a játékban.
    i. *bool felezes:* van-e még felezési segítsége a játékosnak.
    j. *bool kozonseg:* van-e még közönség szavazási segítsége a játékosnak.
- *Nehezseg* struktúra: azt tárolja, hogy az egyes nehézségi szintek esetén, mennyi kérdésre van szüksége a játéknak a különböző felelet-választós nehézségi csoportokból. Részei:
    a. *int easy:* könnyű nehézségi csoportból összeszedendő kérdések száma.
    b. *int mid:* közepes nehézségi csoportból összeszedendő kérdések száma.
    c. *int hard:* nehéz nehézségi csoportból összeszedendő kérdések száma.
- *Score* struktúra: a dicsőséglistához használt struktúra, amely a dicsőséglustában tárolandó és kiírandó adatokat tartalmazza. Részei:
    a. *char *name:* a játékos nevére mutató pointer.
    b. *int sec:* az eltelt másodperceket tárolja (0-60).
    c. *int min:* az eltelt perceket tárolja (0-60).
    d. *long score:* a játék során szerzett pontok számát tárolja.
    e. *int diff:* a játék nehézségi szintje (1-3).

III. Függvények

A *lista.c* modul függvényei:

- *FelValasz *lista_hozzafuz (FelValasz *start, FelValasz *data):*
    a. feladat: ha van adata, amit fel tud dolgozni, akkor elkezdi az alap adatszerkezet listáját felépíteni és visszaadja az első elemre mutató pointert. Egyébként NULL pointerrel tér vissza.
    b. paraméterek: az első elemre mutató pointer (ha még nincs akkor NULL), és a listára felfűzendő struktúrára mutató pointer.
- *char *dintomb_foglal (char *str):*
    a. feladat: a paraméterként megadott sztringet egy dinamikusan lefoglalt memóriaterületre másolja és visszaadja az arra mutató pointert. NULL a visszatérési értéke, ha sikertelen a dinamikus foglalás.
    b. paraméterek: sztringre mutató pointer.
- *void lista_bejar (FelValasz* start, int id):*
    a. feladat: az adott listát bejárja a végéig és az utolsó elemét megindexeli.
    b. paraméterek: a lista kezdeti elemére mutató pointer és az index.
- *FelValasz *lista_keres (FelValasz *start, const int idx, const int min, const int max):*
    a. feladat: megkeres először egy adott indexű és utána egy adott nehézségi szintű elemet a listában, ami az adott indexű elem után helyezkedik el, visszaadja az arra mutató pointert. A nehézséget random választja ki, a maximum és a minimum értékek közöttit.
    b. paraméterek: a lista kezdeti elemére mutató pointer, a keresett indexű elem, a minimum és a maximum nehézségi szintje a keresett elemnek.
- *FelValasz *lista_torles (FelValasz *start, const int idx):*
    a. feladat: kikeres egy adott indexű listaelemet és kitörli a listából, mindig visszaadja a lista kezdeti elemét.
    b. paraméterek: a lista kezdeti elemére mutató pointer és a keresett index.

- *void lista_rendez_kategoria (FelValasz* start):*
    a. feladat: rendezi a listát az alapján, hogy egymás után nem következhet ugyanaz a kategóriájú kérdés.
    b. paraméterek: a lista kezdeti elemére mutató pointer.
- *void lista_indexel (FelValasz* start):*
    a. feladat: megindexeli a láncolt lista elemeit az elejétől a végéig.
    b. paraméterek: a lista kezdeti elemére mutató pointer.
- *void felszabadit1 (FelValasz *start):*
    a. feladat: felszabadítja a felelet-választós kérdések listájának struktúrái által dinamikusan lefoglalt memóriaterületeket.
    b. paraméterek: a lista kezdeti elemére mutató pointer.
- *void felszabadit2 (Sorkerdes *p):*
    a. feladat: felszabadítja a sorkérdés struktúra által dinamikusan lefoglalt memóriaterületet.
    b. paraméterek: a sorkérdés struktúrára mutató pointer.

A *file.c* modul függvényei:

- *FelValasz *read_felvalasz (FILE *fp):*
    a. feladat: felelet-választós kérdések minden adatának beolvasása az adott fájlból és dinamikusan foglalt memóriaterületen való eltárolása. Visszatérési értéke a beolvasott struktúrára mutató pointer, illetve NULL, ha már nincs mit beolvasni, ekkor az olvasott fájlt is bezárja vagy még akkor is NULL, ha sikertelen a dinamikus foglalás.
    b. paraméterek: a már megnyitott fájl olvasására használt pointer.
- *int counter_sorkerdes (void):* a *read_sorkerdes* segédfüggvénye.
    a. feladat: megszámolja hány soros a sorkérdéseket tartalmazó fájl. Visszatérési értéke a darabszám, illetve -1 ha nem sikerült a fájlt beolvasni.
- *void scan_sorkerdes (FILE *fp, Sorkerdes *first):* a *read_sorkerdes* segédfüggvénye.
    a. feladat: kiolvas egy sorkérdést minden adatával együtt a fájlból.
    b. paraméterek: a fájl beolvasásához használt pointer és a sorkérdés struktúrára mutató pointer.
- *Sorkerdes *read_sorkerdes (const FelValasz *start):*

   a. feladat: a sorkérdés fájlból random kiolvas egy sorkérdést és azt eltárolja egy dinamikusan foglalt memóriaterületen, ha kategóriaegyezés van közte és az első felelet-választós kérdés között akkor újra olvas. Visszaadja a sorkérdés struktúrára mutató pointert, illetve bármilyen lehetséges hiba esetén az visszatérési értéke NULL.

   b. paraméterek: a felelet-választós kérdés lista kezdeti elemére mutató pointer.

- *int save (const Milliomos game):*

   a. feladat: menti a fentiekben leírt módon a *save.txt* fájlba a játék jelenlegi helyzetét. Visszatérési értéke 0, ha sikeres a mentés és -5, ha sikertelen.

   b. paraméterek: a játék struktúra.

- *void scan_save_felvalasz (FILE *fp, Milliomos *game):* a *read_save* segédfüggvénye.

   a. feladat: a felelet-választós kérdések listáját olvassa be és építi fel a *save.txt* fájlból. Idő előtt visszatér, ha üres a fájl.

   b. paraméterek: a fájl beolvasására használt pointer és a játék struktúrára mutató pointer.

- *Milliomos read_save (void):*

   a. feladat: kiolvassa a mentett játékmenetet a *save.txt* fájlból a fentiekben leírt adatszerkezet szerint. Visszatér a beolvasott játék struktúrájával, illetve idő előtt visszatér, ha a fájl üres.

- *int hasonlit(void const *, void const *):*

   a. feladat: qsort() függvény segédfüggvénye, amely összehasonlítja az elemeket. Visszatérése -1, ha a második elem nagyobb, 1, ha az első nagyobb, amúgy meg ha egyenlőek, akkor 0.

   b. paraméterek: két összehasonlítandó tömbelem.

- *void scoreboard (const Score game):*

   a. feladat: beírja a dicsőséglistába az elért eredményt, ha elég jó ahhoz, hogy felkerüljön. Minden esetben beolvassa az egész dicsőségtáblát és menti az újat. Idő előtt visszatér, ha valami problémába ütközik a függvény.

   b. paraméterek: az játék során elért eredmény struktúrája.

- *void scoreboard_kiir (void):*

a. feladat: beolvassa a *dicsoseglista.txt* fájlból az adatokat és kiírja azokat. Idő előtt kilép, ha nem található a fájl.

A *jatek.c* modul függvényei:

- *Nehezseg valaszt (Milliomos *game):*
  a. feladat: játék nehézségi szintjének kiválasztása. Visszaadja a nehézségi szint struktúrát.
  b. paraméterek: a játék struktúrájára mutató pointer.
- *int sorkerdes (Milliomos *game):*
  a. feladat: kiválasztása annak, hogy szeretnél-e a játék során sorkérdést. Visszatérési értéke 0, ha sikeres volt a sorkérdés kiválasztás és -4 ha nem.
  b. paraméterek: a játék struktúrájára mutató pointer.
- *void trim (char *kerdes):*
  a. feladat: segédfüggvény, amely kiszedi a felesleges idézőjeleket a kérdésekből, amit a *.csv* típusú fájlok raknak bizonyos esetekbe bele.
  b. paraméterek: a kérdésre mutató pointer.
- *FelValasz *jatek_hozzafuz (FelValasz *start, FelValasz *data):*
  a. feladat: a játék felelet-választós listáját állítja össze, nehézségi sorrendbe rendezve. Visszatérési értéke a lista kezdeti eleme, illetve NULL, ha sikertelen a dinamikus foglalás.
  b. paraméterek: a játék felelet-választós lista kezdeti elemére mutató pointer és a következő listaelemre mutató pointer.
- *FelValasz *generalas (FelValasz **start, const Nehezseg diff, const int db):*
  a. feladat: kiválogatja az eredeti adathalmazból random a játék során használt felelet-választós kérdéseket a nehézségi szinttől függően. Visszatérési értéke a felelet-választós kérdések listájának kezdeti eleme.
  b. paraméterek: a lista kezdeti elemére mutató pointer pointere, a nehézségi szint struktúrája és az eredeti adathalmaz sorainak száma.

- *int min (int a, int b):*
    a. feladat: kiválasztja két szám közül a kisebbet és visszaadja.
    b. paraméterek: egyik szám és másik szám.
- *void felezes (Milliomos *game):*
    a. feladat: felezési segítség függvénye.
    b. paraméterek: a játék struktúrájára mutató pointer.
- *void kozonseg (Milliomos *game):*
    a. feladat: a közönség szavazás segítség függvénye.
    b. paraméterek: a játék struktúrájára mutató pointer.
- *void timer (Milliomos *game, long ido):*
    a. feladat: átváltja az eltelt másodperceket percekre és másodpercekre.
    b. paraméterek: a játék struktúrájára mutató pointer és a játék során eltelt másodpercek száma.
- *void feloszt (long sz):*
    a. feladat: egy adott számot 3 számjegyenként tagolva írja ki.
    b. paraméterek: a kiírandó szám.
- *bool part_sorkerdes (Milliomos *game, const long *start):*
    a. feladat: a játék sorkérdés részét lebonyolító függvény. Visszatérési értéke igaz, ha rossz a válasz vagy félbe akarja hagyni a játékos a játékot, illetve egyébként hamis.
    b. paraméterek: a játék struktúrájára mutató pointer és az idő kezdeti értékére mutató pointer.
- *int part_fv (Milliomos *game, const long *start, int *i, const long *winnings):*
    a. feladat: a játék sorkérdés részét lebonyolító függvény. Visszatérési értéke 1, ha félbe akarja hagyni a játékos a játékot és 2, ha rossz a válasz, amúgy meg 0.
    b. paraméterek: a játék struktúrájára mutató pointer, az idő kezdeti értékére mutató pointer, az indexre mutató pointer és a nyeremények tömbjének kezdeti elemére mutató pointer.

- *void play (Milliomos *game):*
    a. feladat: a játék menetét végig vivő függvény. Idő előtt kilép, ha valamelyik kérdésre rossz a válaszod.
    b. paraméterek: a játék struktúrájára mutató pointer.