Legyen Ön Is Milliomos (Dokumentáció) (félkész)

Készítette(Sajben Dániel)

A programnak a működéshez bizonyos külső fájlokat tartalmaznia kell, ilyen például a „loim” excel fájl, amiből én készítettem egy loimbe.csv-típusú excelfájlt, a könnyebb kezelhetőség érdekében.

A program , két darab „.c”-típusú source fájlt használ, a mainen kívül és egy darab header fájlt, mely az előbb említett .c fájlokat és azok struktúráját tartalmazza. Ezek a modulok a dicsőségtáblához, illetve a toplista kiíratásához kellenek.

A main.c fájlban az include résznél, az utolsó elkülönített 3 sor, azért felelős, hogy a programunk képes legyen ékezetes szöveget is beolvasni az excel fájlból és megjeleníteni azt, ennek a modulnak a másik része a main() függvény elején található. Ez csak Windows operációs rendszeren képes az ékezetek megjelenítésére, ezért, ha a programot más operációs rendszeren használjuk, ilyen téren problémákba ütközhetünk.

A main.c fájlban használt struktúrát a header fájlban definiáltuk, „Kerdessablon” néven, ami elengedhetetlen lesz számunkra a későbbiekben, hiszem minden adat, amivel dolgozni fogunk, egy ilyen típusú tömbben lesz eltárolva. Fő feladatunk pedig ennek a tömbnek a kezelése lesz.

Az első „while” ciklusunk, mely „//1.”-el van jelölve, megnézi hogy különböző nehézségű kérdésekből, hány-hány darab van. A loimbe.csv excel fájlból egy-egy sort olvas be a második sortól kezdve, addig amíg a fájl végéig el nem jut. Mindig a sor első celláját megvizsgálja és a nehézség alapján, egy tömbnek azt az elemét 1-el növeli. Ezen kívül növeljük az „n” változót, amivel azt számoljuk, hogy mekkora helyet kell majd foglalnunk a „kerdesek” tömbnek.

A következő ciklusunk, mely „//2.”-al van jelölve, tokenizál. Ez azt jelenti nekünk, hogy bekérünk egy sort a kérdések excel fájlunkból, és részekre bontjuk, az strtok() függvény segítségével, amit az elején létrehozott struktúra típusú tömbünkben eltárolunk, később ezzel fogunk végig dolgozni. Az strtok() függvény a kiválasztott karakter, helyére egy NULL pointert tesz, így tagolja a sorokat.

A „//3.”-al jelölt ciklusunk, már jóval összetettebb és több dolgot csinál. Random generál egy kérdést a kiválasztott nehézségi szintből, és ha az már volt, akkor újat generál, addig amíg jót nem kapunk. Ezt követően a kérdések kiírását valósítjuk meg a választól függően, maximum 15-öt írunk ki. Itt valósítjuk meg a „felező” és a „közönség segítsége” modult is, melynek az egyszer használhatóságáról egy-egy bool típusú változó gondoskodik. Mind a két modulban van random szám generálás, hogy mindig mások legyenek a kimenő adatok. Mind a két module úgy van megoldva, hogy segítse a felhasználót. A közönség segítségénél a legnagyobb esélyű válasz mindig jó, a felező pedig mindig tartalmazza a jó választ. A végén ellenőrizzük a válasz helyességét, és hogy hányadik kérdés volt, ezek alapján jóváírjuk a nyereményeket, és a játék végeztével leállítjuk a stoppert, melyet a nehézségi szint kiválasztásánál indítottunk el, illetve beleírjuk a játékos adatait egy .txt fájlba, amiből majd a tablasorrend() függvényünk, melyet a dicsőségtábla.c fájlban készítettünk, kiolvassa az adatokat és sorba rendezi őket, majd kiírja minden játék elején a toplista résznél, ami legfelül található..

A tablasorrend() függvényünk, a dicsosegtabla.c fájlban található. A függvény úgy működik, hogy egy .txt fájlba beleírja a játékos adatait. Ezután megszámolja, hogy hány sorból áll a .txt fájl és egy ekkora tömböt dinamikusan foglal. Ezt követően tokenizálja a sorokat, és beletölti az adatokat egy „Tabledesign” típusú tömbbe. Ha ezzel megvan, buborék rendezéssel sorba rendezi a tömb elemeit, a nyeremények alapján, azonos nyeremények esetén a jobb idővel rendelkező játékos kerül előre. Végül a függvény visszaírja a játékosok adatait a fájlba, helyes sorrendben.

Az utolsó modul, amire ki kell térnünk az kiir()-függvény, mely a kiir_dicsosegtabla.c fájlban található. Ez a függvény gondoskodik a toplista megjelenítéséről. Az argumentum két fájlt kér be, ezek általában ugyan azok, az egyik azért kell, hogy megszámoljuk, hogy hány sorból áll a fájlunk, így ha 10-nél kevesebb játékos játszott, akkor csak annyi játékost írjunk ki, ha meg több akkor csak 10-et. A második fájl az argumentumba

meg azért kell, mert abból íratjuk ki a megfelelő darab sort, nyilván ebben a fájlban a játékosok már rendezett sorrendben vannak, melyről a másik függvényünk gondoskodik.

A program, megfelelően fel is szabadítja a dinamikusan foglalt területeket, melyet a debugmalloc.h ellenőriz.