Programozás alapjai 1. Nagyházifeladat dokumentáció

Készítette: Harsányi Zsolt

Modulok Listája:

main.c

fajlkezeles_lista.c

menu.c

jatekmenet.c

main.c:

A main.c a kiindulási pont, itt hívódik meg a menu.c és még tartalmaz egy segéd függvényt ahhoz, hogy a random értékek mindig változzanak a program futása alatt.

menu.c

A játék tulajdonképpen itt kezdi működését, meghívva a menu() eljárást. Értelemszerűen választanunk kell a menüelemek közül, mit szeretnénk tenni. Előtte azonban a nev_bekeres függvény eltárolja a nevünket ami majd a dicsőséglista íráshoz kell. Ezután kezdhetünk új játékot. Először egy nehézséget kell választanunk ami szerint olvassa be a játék a kérdéseket a csv fájlból. Nehézségi szintek:

Könnyű: 1 – 5 nehézségű kérdések

Közepes: 5 – 10 nehézségű kérdések

Nehéz: 10 – 15 nehézségű kérdések

Ezután nehézség szerint beolvassa az adatokat a program. Ha sikeres volt a beolvasás, akkor meghívásra kerül a jatek_menet() függvény, ami a helyesen eltalált kérdések számát adja vissza. Ha többet talált el a játékos, mint 0, akkor be kerül a dicsőséglistába, és természetesen a segédváltozókat nullázza a játék.

Ha a játékos csak szeretné lekérni a jelenlegi dicsőséglistát, akkor a 2-es opciót választja, ami először végig jár a dicsőséglistán, megszámolva rekordjainak számát és azt visszaadja. Utána, ha a rekordok száma nagyobb, mint 0, akkor kiírja a dicsőséglistát sorba rendezve.

Ha a játékos ki akar lépni, a 3-as opciót választja, amivel meghívja a felszabadító eljárásokat, tehát nem hagy memóriaszivárgást maga után és leáll a menürendszert működtető ciklus(értéke 0 lesz).

fajlkezeles_lista.c

Mivel a további modulok több függvényből állnak, mostmár függvényekre lesz bontva a leírás.

A fajlkezeles_lista.c dolga a fájlokkal és láncolt listával való műveletek megvalósítása. Először is a láncolt lista struktúrája kerül definiálásra, majd a fő lista kerül létrehozásra NULL értékkel.

Kerdes *letrehoz(Kerdes *eleje,char* beolvasott_sor):

Célja egy fájlból beolvasott sor hozzáfűzése a listához. Paraméterei maga a lista, amihez hozzáfűz és a sor amit hozzá fog adni. Visszatérési értéke mindig a keletkezett lista.

void felszabadit(Kerdes *kerdesek):

Az eljárás a lista memóriájának felszabadításáért felelős. Amíg a lista pointere nem NULL-ra mutat meghívja a free() függvényt a lista egy sorára és a listaelemre.

void beolvas(int valasztott_nehezseg):

Az adatok beolvasásáért felelős eljárás. Egy FILE pointer segítségével a választott nehézség szerint beolvassa a szöveges fájl sorait és azokra egyesével meghívja a letrehoz() függvényt, mindig hozzáfűzve a listához a következő sort. Három lehetséges módon olvashat be, ami ugye a választott nehézségtől függ. Mivel minden sor a nehézségi szintjével kezdődik, elég azt vizsgálni, hogy melyik intervallumba esik. A végén lezárja a fájl olvasást.

void fajlba_iras(char* nev,int nehezseg,int megvalaszolt):

Neve elárulja feladatát, A dicsoseg_lista.txt nevű fájlba fogja kiírni a szerzett pontot és a játékos nevét, aki szerezte. A menu.c-nél említve van,hogy csak akkor hívódik meg, ha a játékosnak sikerült legalább egy kérdésre helyesen válaszolnia. A beolvas()-nál kimaradt, de itt is van egy kivételkezelés a txt-t illetűen, hogy létezik-e. Paraméterként a játékos

nevét, a választott nehézségi szintet és a sikeresen megválaszolt kérdések számát kapja. Utóbbi kettőből számolja ki a pontot, amit szerzett a játékos.

void rekordok_rendez(int *t,int n):

Egy egyszerű törpe rendezés csökkenő sorrendben. Természetesen a tömböt és a hosszát kapja paraméterként.

int dicsoseg_lista():

Megnyitja a dicsőséglistát és megszámolja mennyi adata van. Ezt fogja visszaadni. Ha nincs, akkor nullát.

void kiir(int rekordok_szama):

Célja a dicsőséglista kiírása csökkenő sorrendben. Ha a dicsőséglistában nincs 10 elem, akkor azt fogja kiírni, ha viszont több van akkor csak az első 10-et. Ehhez egy dinamikus tömbben eltárolja a szerzett pontokat, egy listában pedig a dicsőséglista sorait. A szerzett pontokra meghívódik a rekordok_rendez() függvény és ezután összehasonlítja a szerzett pontoka mostmár csökkenő sorrendben a sorokkal és így fognak a kimenetre kerülni az adatok. A végén a segéd listát és segéd tömböt is felszabadítja.

jatekmenet.c

A játék szíve tulajdonképpen. Többnyire bármi, amivel kapcsolatban a játékos dönt, vagy a játékos befolyásol, az itt valósul meg.

void nev_bekeres(char nev):

Eljárás, mely a játékos nevét kéri el, lefoglalva azt a paraméterként kapott sztringben.

void nev_felszabadit(char *nev):

A kapott név felszabadításáért felelős eljárás.

int valasztas():

A nehézségi szint választásáért felelős függvény. Visszatérési értéke a játékos által választott nehézségi szint.

int random_kerdes_szam(int kerdesek_szumma):

Paraméterként a lista összes beolvasott sorának számát kapja, ami segítségével visszaad egy random sort, amit majd feltesz a játékosnak.

int kerdes_kiir(char* sor):

Egy sort kap, amiből kiírja a kérdést.Ez a sor felépítése szerint az első ';' után helyezkedik el. Visszaadja a kérdés utáni pozíciót, mint segédváltozó később, hogy a többi függvénynek ne kelljen újra addig bejárni a sort.

int valaszok_kiir(char *sor,int folytatas):

A kimenetre írja a válaszokat. Paraméterként az aktuális sort, és azt a számot kapja, ahonnan folytatnia kell a sorban az olvasást. Először háromszor a ';' karakterig olvas a sorban, megkeresve a válaszok kezdő indexei előtt lévő ';' karaktert. Utána létrehoz egy segéd tömböt, benne a válaszok betűivel és egyet a válaszok kezdő indexeivel. Ezután 4-szer beolvassa a választ, iterációnként változtatva a válasz betűjét, és az indexet, ahonnan kezdi az olvasást. Visszaad egy segéd indexet, ahol a helyes válasz betűjele fog kezdődni.

char valaszadas():

Kéri a játékostól a választ, arra hogy melyik válasz helyes a kérdésre. Vagyis egy betűt A-tól D-ig. A függvény képes kisbetűs karaktert is fogadni. Visszaadja a megadott karaktert.

int helyes_valasz(char *sor,int folytatas,char valasz):

Összehasonlítja a megadott választ a helyes válasszal a sorban. A helyes válasz indexét a folytatas segítségével találja meg. 1-et ad vissza, ha a válasz helyes volt, 0-át ha nem.

void segitsegek(int segitsegek,int kozonseg):

Lényege, hogy azokat a segítségeket írja ki, amik még elérhetőek. Ha a paraméterként kapott segítségek bármelyike 0 értékű, akkor már a következő kérdésnél nem fogja a kimenetre írni.

void helyes_kiir(int helyes_valasz,int *valaszok,char helyes,char *sor):

A helyes válasz kiírásáért felel. Az index egy valaszok nevű a válaszok indexeit tartalmazó helyes_valasz helyén lesz, amit a kezdete változónak ad át. A sorból fog olvasni, a helyes betű pedig a válasz elé fog kerülni.

void helyes_melle_masik_kiir(int *valaszok,char* valasz_betuk,char* sor,char helyes):

A helyes válasz mellett egy másik random kérdés kiírásáért felel. Először random választ egy másik kérdést, utána kiírja azt.

void kerdes_felezes(char *sor,int seged,int valasz_helye):

Az eljárás lényege, hogy a 4 válaszlehetőségből ki vegyen kettőt, benne hagyva a jó választ és mellette még random egyet. Először is a paraméterként kapott valasz_helye indexxel megkeresi a sorban a helyes válasz betűjelét. Ezután hasonlóan mint a valaszok_kiir() függvény megkeresi a ’;’ karakterek indexeit a sorban. Ezek egy valaszok nevű tömbbe kerülnek bele és emellett még egy ABCD-t tartalmazó segéd sztring is keletkezik. E kettőt kapja paraméterként a helyes_kiir() eljárás és a helyes_melle_masik_kiir() eljárás. Hogy ne mindig a helyes válasz legyen felül és ne legyen kiszámítható, ezért egy random szám segítségével a program eldönti, hogy milyen sorrendben írja ki a helyes és a másik válasz lehetőséget.

void szavazat_stat(int szavazat):

A paraméterként kapott szavazatot osztja kettővel és annyiszor fog ’+’ jelet írni egy sorba.

void kerdes_kozonseg(char *sor,int valasz_helye):

A közönség szavazatait reprezentálja. Először is megkeresi a helyes válasz betűjelét a valasz_helye index segítségével a sorban, utána létrehoz egy valaszok nevű segéd sztringet és egy 4 elemű szavazásokat tartalmazó egész típusú tömböt. Ezután egy ciklus segítségével, ami addig fog végrehajtódni, amíg az összes szavazat összege nem pont százzal lesz egyenlő. Ezután történik egy maximum érték és egy maximum hely keresés. Ezek után pedig kicseréli a szavazasok tömbben a legnagyobb helyének indexét a helyes válasz indexével. Természetesen a program feltételezi, hogy a sok néző többségében a jó válaszra fog szavazni. Ezt követően pedig egyszerűen kiírja a válaszokat ABCD sorrendben, meghívva a szavazat_stat() függvényt a szavazatokra.

void segitseg_keres(char *sor,int *felezes,int *kozonseg,int seged,int valasz_helye):

Az eljárás célja, hogy minden kérdésnél segítséget ajánljon fel a játékosnak, feltéve ha maradt hátra segítségkérési lehetősége. Ezek az int *felezes és int *kozonseg változók értékei, amik 1 vagy 0 lehetnek, attól függően hogy a játékos elhasználta-e őket. Ha a játékos szeretne segítséget kérni(Az 'I'-t választja), akkor választania kell a fent maradó segítségek közül. Értelemszerűen ha a felezést választja akkor a kerdes_felezes(), ha a szavazást akkor a kerdes_szavazas() hívódik meg és a hozzájuk tartozó int kozonseg, vagy szavazas ertek nullára áll. Ha a nemet választja akkor pedig a program a válaszát fogja várni, ahelyett hogy segíteni.

int jatek_menet(Kerdes *lista):

A legfontosabb függvény a modulban, többnyire az összes eddigi eljárást és függvényt öleli egybe, létrehozva magát a játékmenetet. A függvény az elején feltölt egy 15 elemű tömböt random indexekkel a listából, amik a kérdések lesznek, amiket feltesz a játékosnak. Azért 15, mert maximum ennyi kérdésre fog válaszolni a játékos. Mindeszt úgy teszi, hogy ne ismételjék egymást a kérdések. Ezután elkezdi feldobálni a random kérdéseket egy ciklus segítségével. Ezt addig teszi amíg a kérdések száma meg nem haladja a 15-öt vagy rosszul válaszol a játékos. Visszatérési értéke a helyesen megválaszolt kérdések száma.