Az Ítélet Labirintusa

Ez a program nem multiplatformos, vagyis csak egy operációs rendszeren futtatható, ami a windows. A program számára szükséges a „szovegek.txt” fájl, aminek a formátuma az egy szám és tőle egy tabulátorjellel elválasztva egy szöveg, egy „valtozasok.txt” fájl, ami 19 adatot tartalmaz, amik tabulátorjellel vannak eltárolva és egy „alap.txt” fájl, amiben a történet bevezetése van leírva. Ezek a szövegesfájlok Windows-1250-ben kell legyenek kódolva, hogy a program helyesen írja ki a konzolra a kiírandó dolgokat.

Struktúrák

- Mezo

Típus	Név	Magyarázat
egész	**id**	**A mező azonosítója, ez alapján hivatkozik a program az adott mezőre.**
karakter	**type**	**Az adott mező típusa (n-normál, s-szerencse, h-harc, v-vége).**
egész	**a**	**Normál mező esetében: az első választási lehetőség, ha a felhasználó ezt választja, akkor azon a mezőn folytatódik a történet, aminek ez az azonosítója, szerencse mezőnél, ha a játékosnak szerencséje van akkor ezen a mezőn folytatódik a történet, harcmezőn erre folytatódik a történet, ha a játékos legyőzi az ellenséget.**
egész	**b**	**Normál mező esetében: az második választási lehetőség, ha a felhasználó ezt választja, akkor azon a mezőn folytatódik a történet, aminek ez az azonosítója, szerencse mezőnél, ha a játékosnak balszerencséje van akkor ezen a mezőn folytatódik a történet.**
egész	**c**	**Csak normál mezőnél lehet, ez a harmadik választási lehetőség. Ha a felhasználó ezt választja, akkor azon a mezőn folytatódik a történet, aminek ez az azonosítója.**
egész	**d**	**Csak normál mezőnél lehet, ez a negyedik választási lehetőség. Ha a felhasználó ezt választja, akkor azon a mezőn folytatódik a történet, aminek ez az azonosítója.**
sztring	**asz**	**Az első lehetőség leírása, ha az elsőt akarja választani a felhasználó akkor ez fog történni vele.**
sztring	**bsz**	**A második lehetőség leírása, ha a másodikat akarja választani a felhasználó akkor ez fog történni vele.**
sztring	**csz**	**A harmadik lehetőség leírása, ha a harmadikat akarja választani a felhasználó akkor ez fog történni vele.**
sztring	**dsz**	**A negyedik lehetőség leírása, ha a negyediket akarja választani a felhasználó akkor ez fog történni vele.**
egész	**e**	**Ennyivel változik a felhasználó életerő pontja.**
egész	**u**	**Ennyivel változik a felhasználó ügyesség pontja.**
egész	**sz**	**Ennyivel változik a felhasználó szerencse pontja.**
sztring	**itemp**	**Ezt a tárgyat szerzi a játékos az adott mezőn.**
sztring	**itemm**	**Ezt a tárgyat elveszíti a játékos az adott mezőn.**
sztring	**itemr**	**Ez a tárgy szükséges az adott mezőre lépéshez.**
sztring	**sznev**	**Az adott mezőn az ellenség neve, ha van ellenség.**
egész	**szu**	**Az adott mezőn az ellenség ügyesség pontja, ha van ellenség.**
egész	**sze**	**Az adott mezőn az ellenség életerő pontja, ha van ellenség.**
sztring	**szoveg**	**Az adott mező története.**

- Playerdata

Típus	Név	Magyarázat
egész	e	Játékos életerő pontja
egész	u	Játékos ügyesség pontja
egész	sz	Játékos szerencse pontja
egész	emax	Játékos életerő pontjának maximuma, egyenlő a kezdeti értékkel
egész	umax	Játékos ügyesség pontjának maximuma, egyenlő a kezdeti értékkel
egész	szmax	Játékos szerencse pontjának maximuma, egyenlő a kezdeti értékkel

- Item

Típus	Név	Magyarázat
egész	db	Az adott tárgy darabszáma
sztring	nev	Az adott tárgy neve
karakter	felhaszn	Ha a felhasználó fel tudja használni az eszköztár megnyitásakor akkor „i”
karakter	valt	A felhasználó melyik adatát változtatja meg
egész	ertek	A felhasználónak a valt értékét ennyivel változtatja meg

- Itemek

Típus	Név	Magyarázat
item	targy	Egy tárgy adatai
itemek*	next	A következő itemek típusú változó memóriahelyére mutató pointer

Modulok

- beolvas.c
  - void beolvas(mezo* mezok)
    - Paraméterek: Mezo típusú mezok nevű tömb elejére mutató pointer.
    - Feladat: A „valtozasok.txt” -ből és a „szovegek.txt” -ből az adatokat beolvassa és eltárolja őket a mezok tömbben.
    - Működése: Egyesével olvassa be szövegfájlokat és azoknak a sorait külön-külön olvassa be és tárolja be egy sztringben, amit sscanf segítségével tárolja el az értékeket, a mezok tömbben.
  - void befejez(mezo *m)
    - Paraméterek: Mezo típusú m nevű pointer, ami az aktuális mezőre mutat.
    - Feladat: Az m változó béli string típusú változók (asz, bsz, csz, dsz, sznev, itemm, itemp, itemr) végére a befejező nulla beillesztése.
    - Működése: Megnézi, hogy milyen hosszú az adott string és a végére odaírja a befejező nullát.
- kiiratas.c
  - void kiiratas(mezo m,playerdata p)
    - Paraméterek: mezo típusú m nevű változó és playerdata típusú p nevű változó.
    - Feladat: Kiírja a képernyőre a játékos jelenlegi életerő pontját, ügyesség pontját, szerencse pontját, azt, hogy az adott mezőn mi történik, ha van akkor a választási lehetőségeket.
    - Működése: Minden adatot kiír a képernyőre, és közben számolja, hogy hányadik sorban tart a képernyőre jelenleg kiírt adat, mert a bont függvényben az econio_gotoxy függvénynek az egyik paramétere az y koordináta, ami a sorok számát jelenti.
  - void bont(char *str,bool opcio,int *sor)

        - Paraméterek: Sztring elejére mutató str nevű pointer, egy bool típusú opcio nevű változó, ami true, ha a kiírandó szöveg az egy választási lehetőség, és a sor nevű változóra mutató pointer.
        - Feladat: Az str stringet kiírni, úgy, hogy egy sorban maximum 100 karakter legyen, de a csak szóköz helyére illeszthet be új sor karaktert.
        - Működése: Egy ciklussal végig megy a teljes sztringen, és kiírja az összes karaktert egyesével, egy hossz változóban tárolódik a jelenlegi sornak a hossza, ami, ha eléri a nagyobb vagy egyenlő mint 100, akkor az adott sor 100. karakterétől visszafelé elkezd haladni, és megkeresi az első szóköz karaktert, és a helyére beszúr egy új sor karaktert, és a jelenlegi sorban az ez utáni összes karaktert kicseréli szóközre, és ezeket a karaktereket a következő sor elejére kiírja, és lenullázza a hossz változót, vagy ha az opcio az true akkor 4re veszi(mert ha van választási lehetőség, akkor az így néz ki pl.:[1] és utána egy tabulátorjel, ami 4 karakter), és így a választási lehetőségek első karakterei egy oszlopban lesznek.

- fuggvenyek.c
    - void vonal()
        - Feladat: Egy 100 karakter hosszúságú vonal létrehozása
        - Működése: Kiír 1db „+” karakter, utána 98db „-” karakter, majd 1db „+” karakterrel befejezi a sort, és utána új sort kezd, fontos, hogy használata előtt új sort kell kezdeni.
    - void mentes(mezo m,playerdata p,int *hely)
        - Paraméterek: Mezo típusú m nevű változó, playerdata típusú p nevű változó, egész típusú hely pointer, ami a main függvény most nevű változójára mutat.
        - Feladat: Elmenti a játékos jelenlegi adatait, és azt, hogy jelenleg melyik mező volt kiírva.
        - Működése: Megnyitja a „mentes.txt” nevű szöveges fájlt (vagy létrehozza, ha nem létezik), és a jelenlegi adatokat eltárolja benne, az egyes adatok között tabulátort használva elválasztásnak.
    - bool szerencse(playerdata *p)
        - Paraméterek: Playerdata típusú p nevű pointer, ami a játékos adataira mutat.
        - Visszatérési értékek: true, ha szerencséje van a játékosnak, false, ha balszerencséje van.
        - Feladat: Eldönti, hogy a felhasználónak szerencséje van-e.
        - Működése: Generál egy véletlenszerű számot 2 és 12 között, majd, ha a felhasználónak legalább egy szerencse pontja van, akkor levon belőle egyet, és ha a szerencse pontjainak száma több vagy egyenlő, mint a véletlen szám, akkor szerencséje van a felhasználónak, egyébként balszerencséje van.
    - int rosszvalasz(int db,char *str)
        - Paraméterek: egész típusú db nevű változó, ami a választási lehetőségek darabszámát jelöli (az eszköztár megnyitása és a mentés kilépés nincs beleszámolva), a kiírandó szöveg.
        - Visszatérési értékek: A választás száma egész szám formában. Ha az eszköztár megnyitását választotta a felhasználó, akkor -1, ha a kilépést, akkor -2, egyébként az, hogy hányadik lehetőséget választotta.
        - Feladat: Addig kérdezgetni a felhasználótól azt, hogy mit választ, ameddig alkalmas választ nem ad.
        - Működése: Először kiírja, az str sztringet, majd a választ eltárolja egy dinamikusan foglalt sztringben. Megnézi, hogyha a második karaktere a válasznak a befejező nulla, akkor, ha az első karaktere „i” vagy „I”, akkor visszatér nullával, ugyanezt megteszi „q” és „Q” -val. A választ átalakítja egész számmá és megnézi, hogy érvényes-e, ha igen akkor azzal tér vissza, ha nem, akkor elölről kezdődik a ciklus.
    - int szornysebez(playerdata *p,int *szerencsevolt,int *valasz)
        - Paraméterek: Playerdata típusú p nevű pointer, ami a játékos adataira mutat, egész típusú szerencsevolt nevű változó, ami következő harc fordulóban lesz felhasználva.
        - Visszatérési értéke: -1, ami a harcm nevű függvényben lesz felhasználva.

        - Feladat: A játékos életerejét csökkenteni, és kiírni a választási lehetőségeket, hogy a felhasználó mit szeretne csinálni.
        - Működése: Kiírja azt, hogy az ellenség megsebezte, utána a választási lehetőségeket, majd a rosszvalasz függvény kideríti a választását a felhasználónak. Végül visszatér -1-gyel.
    - int playersebez(playerdata *p,int * szerencsevolt,mezo *m,int *valasz)
        - Paraméterek: Playerdata típusú p nevű pointer, ami a játékos adataira mutat, egész típusú szerencsevolt nevű változó, ami következő harc fordulóban lesz felhasználva és a mezo típusú m nevű pointer, ami az adott mezőre mutat.
        - Visszatérési értéke: 1, ami a harcm nevű függvényben lesz felhasználva.
        - Feladat: Az ellenség életerejét csökkenteni, és kiírni a választási lehetőségeket, hogy a felhasználó mit szeretne csinálni.
        - Működése: Kiírja azt, hogy a megsebezte az ellenséget, utána a választási lehetőségeket, majd a rosszvalasz függvény kideríti a választását a felhasználónak. Végül visszatér -1-gyel.
    - void dontetlen(int *szerencsevolt,int *valasz)
        - Paraméterek: egész típusú szerencsevolt nevű változó és a karakter típusú valasz nevű pointer, ami a harcm függvény valasz nevű változójára mutat
        - Feladat: Kiírja a választási lehetőségeket, majd beolvassa a választ.
        - Működése: Kiírja a választási lehetőségeket, majd a rosszvalasz függvény segítségével beolvassa a felhasználó döntését.
    - void normalm(mezo m,int *hely,playerdata *p,itemek *fej)
        - Paraméterek: Mezo típusú m nevű változó, egész típusú hely pointer, ami a main függvény most nevű változójára mutat, playerdata típusú p nevű pointer, ami a játékos adataira mutat, itemek típusú láncolt lista elejére mutató pointer.
        - Feladat: Felajánlja a felhasználónak a lehetőséget, hogy megnyissa az eszköztárát (még nincs leprogramozva), és azt, hogy kilépjen, és elmentse a jelenlegi állást, és még 1-4 választási lehetőséget, és arra a mezőre kerül, amit kiválaszt.
        - Működése: Először megnézi, hogy hány választási lehetőség van az adott mezőn, és a rosszvalasz függvényt annyival futtatja le, majd az eredményt eltárolja egy valasz nevű karakter típusú változóban. Mivel csak érvényes válasz kerülhet a valasz nevű változóba, ezért ezután már azt nem kell leellenőrizni. A hely változó értékét arra a mezőre változtatja, amit a felhasználó kiválasztott, vagy ha az eszköztár megnyitását választotta, akkor az fog megtörténni, ha a mentés kilépést, akkor pedig az.
    - void szerencsem(mezo m,int *hely,playerdata *p,itemek *fej)
        - Paraméterek: Mezo típusú m nevű változó, egész típusú hely pointer, ami a main függvény most nevű változójára mutat, playerdata típusú p nevű pointer, ami a játékos adataira mutat, itemek típusú láncolt lista elejére mutató pointer.
        - Feladat: Felajánlja a felhasználónak a lehetőséget, hogy megnyissa az eszköztárát (még nincs leprogramozva), és azt, hogy kilépjen, és elmentse a jelenlegi állást, és azt, hogy folytassa a kalandját. A játékos szerencsepontjától függően a játék kisorsolja, hogy a játékos a kettő lehetőség közül melyik irányba fogja folytatni a kalandját.
        - Működése: Először eldönti, hogy mit választ a felhasználó. Ha az eszköztár megnyitását választja, vagy a mentés kilépést, akkor azok fognak megtörténni, ha azt választja, hogy folytatja, akkor a szerencse függvényt felhasználva kisorsolja a játék, hogy melyik mező azonosítója kerül a hely változóba.
    - void harcm(mezo m,int *hely,playerdata *p,itemek *fej)
        - Paraméterek: Mezo típusú m nevű változó, egész típusú hely pointer, ami a main függvény most nevű változójára mutat, playerdata típusú p nevű pointer, ami a játékos adataira mutat, itemek típusú láncolt lista elejére mutató pointer.
        - Feladat: Felajánlja a felhasználónak a lehetőséget, hogy megnyissa az eszköztárát (még nincs leprogramozva), és azt, hogy kilépjen, és elmentse a jelenlegi állást, és azt, hogy elkezdje a harcot. Ha az utolsót választja, akkor elkezdődik a harc, ami több körből áll, és a véletlenen

alapszik, hogy ki sebez kit, és addig megy, ameddig vagy felhasználónak, vagy az ellenségnek el nem fogy az életerő pontja.
        - Működése: Először eldönti, hogy mit választ a felhasználó. Ha az eszköztár megnyitását választja, vagy a mentés kilépést, akkor azok fognak megtörténni, ha azt választja, hogy folytatja, akkor egy ciklus addig fog menni ameddig vagy a játékos, vagy az ellenség életerő pontja 0 nem lesz, vagy a játékos ki nem lép a „q" vagy „Q" betűvel, az utóbbi esetben a harc közben a játékos adataival történt változások megmaradnak, viszont az ellenség adatai visszaállnak a kezdeti értékre. Minden kör úgy kezdődik, hogy az ügyességpontokból kiszámolja a támadó pontokat, úgy, hogy hozzájuk ad 2 és 12 közötti véletlen számot. Ezután megnézi, hogy az előző körben a játékos használta-e a szerencséjét és hogy ki sebzett kit, és kiírja, ha használt szerencsét, hogy szerencséje volt-e és hogy kinek az életerő pontja hogyan változott, ezután megnézi, hogy kinek volt nagyobb a támadóereje, és az alapján foggja használni a szornysebez, playersebez, dontetlen függvényeket. A ciklus legvégén leelenőrzi, hogy valakinek lecsökkent-e 0 vagy az alá az életerő pontja, és az alapján változtatja a hely értékét.
    - void vegem(int *hely)
        - Paraméterek: egész típusú hely nevű pointer, ami a main függvény most változójára mutat.
        - Feladat: Kiírja, hogy a játéknak vége, és törli a mentést.
        - Működése: Kiírja, hogy vége van a játéknak, és megnyitja (vagy ha nem létezik, akkor létrehozza a) a „mentes.txt" nevű szövegesfájlt.

- inventory.c
    - itemek *ujelem(item adat)
        - Paraméterek: Item típusú adat nevű változó
        - Feladat: Az új tárgynak lefoglalni egy memóriaterületet.
        - Működése: Lefoglal egy új memóriaterületet, és az adatokat beletölti, és visszatér az erre mutató pointerrel.
    - itemek *keres(itemek *fej,char *nev)
        - Paraméterek: A láncolt lista első elemére mutató pointer és a keresendő tárgy neve.
        - Feladat: Megkeresi a láncolt listában az adott tárgynak a nevét.
        - Működése: Végigmegy a listán a függvény és ha megatálja a keresendő tárgyat, akkor visszatér annak a pointerével, egyébként null pointert ad vissza.
    - itemek *hozzaad(itemek*fej,item ujitem)
        - Paraméterek: A láncolt lista első elemére mutató pointer, és az új tárgy adatai.
        - Feladat: Ha még része a listának ez a tárgy, akkor hoozzáadja, ha már része, akkor a darabszámát megnöveli.
        - Működése: Lefoglalja a memóriaterületet, az ujelem függvénnyel, ha nem sikerül akkor hibaüzenetet ír ki, ha a lista eddig üres volt, akkor ennek az itemnek a pointerével tér vissza, ha már volt elem a listában, akkor beszúrja a lista elejére ezt az új tárgyat.
    - itemek *listatorol(itemek *fej)
        - Paraméterek: A láncolt lista első elemére mutató pointer.
        - Feladat: Az eszköztár összes elemét törölni.
        - Működése: Végigmegy a listán, és minden lefoglalt memóriaterületet felszabadít, az előadáson tanult módszerrel.
    - itemek *torolelso(itemek *fej)
        - Paraméterek: A láncolt lista első elemére mutató pointer.
        - Feladat: Az eszköztár legelső elemének a törlése.
        - Működése: Ha nem volt a listában egy tárgy se, akkor nincs mit csinálni, visszatér a második elemre mutató pointerrel, ami a NULL pointer. Ha volt benne, akkor visszatér a második elemre mutató pointerrel, és felszabadítja az első helyre mutató pointert.
    - itemek *torles(itemek *fej,item torlendo)

        - Paraméterek: A láncolt lista első elemére mutató pointer, a törlendő tárgy adatai.
        - Feladat: Végigmegy a láncolt listán, és megkeresi a keresendő tárgyat, és törli azt.
        - Működése: Ha az első tárgy volt az, amit törölni kellett, akkor a torolelso függvény segítségével törli azt. Ha nem, akkor az előadáson tanult módszerrel törli a keresendő tárgyat.
    - void invkiir(itemek *fej,playerdata *p)
        - Paraméterek: A láncolt lista első elemére mutató pointer és a játékos adataira mutató pointer.
        - Feladat: Kiírja a játékos adatait, az eszköztára tartalmát, és felajánlja a lehetőséget, hogy felhasználja valamelyiket, amit, ha nem tehet meg, akkor kiírja, hogy nem felhasználható, ha felhasználható, akkor pedig egyel csökkenti a darabszámát, és a felhasználó adatait megváltoztatja.
        - Működése: Először kiírja a felhasználó adatait majd az összes tárgyat és a hozzájuk tartozó darabszámot. Utána a rosszvalasz függvény segítségével kiválasztóik az, hogy a játékos melyik tárgyat szeretné felhasználni. Ezután a program végigmegy a láncolt listán annyiadik elemig, amit a felhasználó választott, majd le ellenőrzi, hogy ezt a felhasználó fel tudja-e használni, és, ha igen, akkor megnézi, hogy hány darab van belőle, ha egy akkor a torles függvénnyel törli, ha nem, akkor egyel csökkenti a darabszámát. Ezután megnézi, hogy a játékos mely adatát változtatja meg, és mennyivel, és meg is teszi.
    - void invmentes(itemek *fej)
        - Paraméterek: A láncolt lista első elemére mutató pointer.
        - Feladat: Létrehozni egy inventory.txt nevű fájlt, és eltárolni abban a játékos eszköztárát.
        - Működése: Létrehozza a fájlt, vagy ha már létezik, akkor felülírja. Ezután egyesével beleírja minden tárgynak az adatát, majd a végén a listatorol függvénnyel felszabadítja a lefoglalt memóriát.

- kulonlegesmezok.c

  Ez a modul azért létezik, mert a könyvben vannak olyan mezők, amiket máshogyan kell kezelni, mint a többit.
    - void husz(mezo m,int *hely, playerdata *p, itemek *fej)
        - Paraméterek: Mezo típusú m nevű változó, egész típusú hely pointer, ami a main függvény most nevű változójára mutat, playerdata típusú p nevű pointer, ami a játékos adataira mutat, itemek típusú láncolt lista elejére mutató pointer.
        - Feladat: A 20-as indexű mező helyes kiírása.
        - Működés: Mivel ez a mező alapból olyan, mint egy normál mező, ezért a választási lehetőségek ugyanúgy lesznek kiírva, mintha ez egy normál mező lenne. De ha a felhasználó azt választja, hogy tudja a kódot, akkor egyesével beírja azokat, és ha akármikor elrontja, akkor a program úgy kezelmi, hogy nem tudja, és az történik, mintha azt a lehetőséget választotta volna alapból.
    - void nk(mezo m,int *hely, playerdata *p, itemek *fej)
        - Paraméterek: Mezo típusú m nevű változó, egész típusú hely pointer, ami a main függvény most nevű változójára mutat, playerdata típusú p nevű pointer, ami a játékos adataira mutat, itemek típusú láncolt lista elejére mutató pointer.
        - Feladat: A 49-es indexű mező helyes kiírása.
        - Működés: Ez a mező is csak egy sima normálmező, csak ez megváltoztatja a szerencsepontok minimumát.
    - void hh(mezo m,int *hely, playerdata *p, itemek *fej)
        - Paraméterek: Mezo típusú m nevű változó, egész típusú hely pointer, ami a main függvény most nevű változójára mutat, playerdata típusú p nevű pointer, ami a játékos adataira mutat, itemek típusú láncolt lista elejére mutató pointer.

        - Feladat: A 66-os indexű mező helyes kiírása.
        - Működés: Ebben a mezőben a felhasználó elveszít kettő tárgyat, és ezeket ő választja ki, hogy melyikek legyenek. Ez úgy történik meg, hogy először kiírja a mezőt, mintha egy normál mező lenne, majd miután a felhasználó azt választotta, hogy folytatja a játékot, akkor megjelenik a teljes eszköztár, és ki kell választania összesen kettő tárgyat, amiket elveszít, és többet nem használ fel a játék során.
    - void szt(mezo m,int *hely, playerdata *p, itemek *fej)
        - Paraméterek: Mezo típusú m nevű változó, egész típusú hely pointer, ami a main függvény most nevű változójára mutat, playerdata típusú p nevű pointer, ami a játékos adataira mutat, itemek típusú láncolt lista elejére mutató pointer.
        - Feladat: A 110-es indexű mező helyes kiírása.
        - Működése: Ez a mező olyan, mint egy harcmező, csak az ellenség életerő pontja és ügyessége a játékos jelenlegi adataitól függ, amiket nem lehet a fájlban alapból megadni, mert véletlenszerű és a játék közben változhatnak. Megváltoztatja az adottt mezőnek ezen adatait, majd átadja a harcm nevű függvények.
    - void sznny(mezo m,int *hely, playerdata *p, itemek *fej)
        - Paraméterek: Mezo típusú m nevű változó, egész típusú hely pointer, ami a main függvény most nevű változójára mutat, playerdata típusú p nevű pointer, ami a játékos adataira mutat, itemek típusú láncolt lista elejére mutató pointer.
        - Feladat: A 144-es indexű mező helyes kiírása.
        - Működése: Ez a mező a szerencse mezőre hasonlít, csak itt nem a felhasználó szerencséjétől függ, hogy melyik mezőn folytatja tovább a kalandját, hanem az ügyességpontjain. A működése teljesen ugyanolyan, mint a szerencse mezőé, csak még előtte átálítja a mező típusát s-re, hogy a kiiratas nevű függvény helyesen írja ki.
    - void szho(mezo m,int *hely, playerdata *p, itemek *fej)
        - Paraméterek: Mezo típusú m nevű változó, egész típusú hely pointer, ami a main függvény most nevű változójára mutat, playerdata típusú p nevű pointer, ami a játékos adataira mutat, itemek típusú láncolt lista elejére mutató pointer.
        - Feladat: A 165-ös és 8-as indexű mező helyes kiírása.
        - Működése: Ez a két mező is olyan, mint egy normálmező, csak ezekben a felhasználó életerőpontja 1 és 6 közötti véletlen számmal csökken. Ez megtörténik, majd teljesen ugyanúgy kiíródik a mező, mintha normálmező lenne.
    - void kulonlegesm(mezo m,int *hely,playerdata *p,itemek *fej)
        - Paraméterek: Mezo típusú m nevű változó, egész típusú hely pointer, ami a main függvény most nevű változójára mutat, playerdata típusú p nevű pointer, ami a játékos adataira mutat, itemek típusú láncolt lista elejére mutató pointer.
        - Feladat: Azoknak a mezőknek a kiírása, amelyeknek a type értéke a „k”.
        - Működés: Egy switchel megnézi, hogy az adott mezőnek az azonosítója mennyi, és az alapján lefuttatja az egyik alfüggvényt.

- econio.c
    - Econio_clrscr: Törli a konzolon jelenleg megjelenő adatokat.
    - Econio_gotoxy: A konzol egy adott koordinátájára teszi a kurzort (a bal felső sarok a 0,0).
    - Econio_set_title: Megváltoztatja a konzol címét.

- main.c

A legelején beállítja a konzol címét „Az Ítélet Labilintusa” -ra majd létrehozza a változókat. Ezután felajánlja a választási lehetőséget, hogy új játék kezdődjön, vagy a mentésből töltse be az adatokat. Ha mentésből betöltést választotta, akkor megnézi, hogy létezik-e a „mentes.txt” és ha igen, akkor megnézi, hogy mi az első karaktere (ez az elmentett mező azonosítója), ha ez nem nulla,

akkor betölti onnan az adatokat. Ha a játékos azt választotta, hogy az elejéről szeretné kezdeni, vagy, ha nem létezik a fájl, vagy az első karakter 0, akkor a játék az elejéről kezdődik. Ebben az esetben Először felajánl egy választási lehetőséget, hogy melyik Italt szeretné a játékos majd a játék folyamán felhasználni és a kiválasztottat eltárolja a láncolt listában, és ezen kívül még öt darab Élelmet, amik egyenként visszatöltenek négy Életerő pontot.. Legenerálja a játékos életerő pontját, ami egy véletlen szám 14 és 24 között, és ez lesz a maximum értéke is az életerőnek, ugyanígy generálódik a játékos többi adata is, de azok 7 és 12 közötti véletlen számok. És a játékos szerencse pontjának a minimumát beállítja nullára. Ezután beolvassa az „alap.txt” fájlt és kiírja a képernyőre, a bont függvény segítségével. Utána van egy választási lehetősége, hogy tovább, ha nem megfelelő számot ír be a felhasználó, akkor a rosszvalasz függvény segítségével addig kérdezi újra, ameddig nem írja be az 1-et. Ezután elindul a ciklus, ami addig megy ameddig vége nincs a játéknak, ami akkor következik be, ha egy függvény a most változót (ami a jelenlegi mező azonosítójánál egyel kisebb, a mezo típusú mezok tömb indexe) -1-re változtatja. Ez akkor következhet be, ha a mező típusa az vége mező vagy a játékosnak az életerő pontja eléri a nullát, vagy ha egy nem létező mező indexét veszi fel, aminek nincsen típusa (mindegyiknek kell, hogy legyen, különben nem tud rendesen kiíródni és beolvasódni). A ciklus úgy kezdődik, hogy a játékos adatait megváltoztatja annyival, amennyi az adott mezőben meg van adva. Ezután a kiiratas függvénnyel kiírja az adott mezőt, és a mező típusa alpján eldönti, hogy a normalm, szerencsem, harcm, vegem, kulonlegesm függvények közül melyiket használja.