Aknakereső:

Tartalom

Aknakereső: Felhasználói dokumentáció

Cél:

Az aknakereső egy logikai játék, ahol a játékosnak fel kell fednie minden mezőt, ami nem tartalmaz aknát, hogy nyerjen.
Egy akna felfedése vereséghez vezet.

Játékmenet:

Minden mező, ami nem tartalmaz aknát jelzi, hogy körülötte pontosan mennyi akna van, ha egy mező körül nincs akna akkor felfedi a körülötte lévő mezőket (a játék kényelmesebbé tételéért).
Egy mezőt meg lehet „zászlózni”, ezt akkor érdemes használni mikor a játékos meg van győződve, hogy egy mezőn akna van, ilyenkor a mezőre kerül egy zászló és nem lehet (véletlenül se) felfedni.
Egy megnyert menet után lehetősége a van a győztesnek, hogy beírja magát a ranglistába,

Bemenetek:

A játék irányítása a billentyűzeten keresztül történik.

Kimenetek:

A játék a képernyőn keresztül kerül megjelenítésre,
továbbá a játék a saját könyvtárában tartalmazza a ranglistákat (ezek a játékon belül is elérhetőek).

Irányítás:

A játék egy menüvel nyílik meg, ebben a menürendszerben lehet a kiválasztott nehézségű játékot elkezdeni, valamint a különböző rekordokat megnézni.
A navigáció a játék folyamán végig ki van írva a képernyőre.
Menünavigáció: Az **ESC** billentyű egy menüponttal feljebb lép, fel le, illetve a jobbra, balra **nyíl** segítségével lehet választani majd **ENTER** lenyomásával lehet továbblépni.
A játék közben a **nyilakkal** lehet a mezők közt navigálni, **ENTER** fedi fel a kiválasztott mezőt, a **SPACE** (szóköz) segítségével lehet zászlót tenni egy mezőre. **ESC**-el lehet a jelenlegi játékból kilépni (megerősítés után).

Aknakereső: Programozói dokumentáció

Projekt felépítése:

A fordításhoz szükséges fájlok:

- main.c: meghívja a különböző menüket
- menuvezerles.h/c: Menüpontok közt lépkedés, kiírás
- jatek.h/c: Játéktábla létrehozása, ebben van a játék tényleges futása
- ranglista.h/c: a ranglista írása/olvasása fájlból, kiírása/beolvasása a felhasználótól
- econio.h/c: MIT licensz, Windows konzoljának koordináta szerinti (színezéses) írása
- typedefs.h: saját enumok, structok, makrók

Szükséges környezet:

A program a Windows konzolos megjelenítésére lett tervezve, Windows 1250 (Közép-Európai) karakterkészletet használ és erre is állítja be a konzol kiírásának a kódolását.
Továbbá a program 3db .txt kiterjesztésű fájlt hoz létre a futtatható fájl könyvtárában, ezekbe írja és olvassa a ranglisták eredményeit. Az eredmények formázottak és idő szerinti növekvő sorrendben vannak, ennek írása a programon kívül kerülendő.

Adatszerkezetek:

Játékhoz tartozó:

Egy **mező** tulajdonságai: van-e rajta **akna**, a **körülötte lévő aknák** száma, **látható**-e és van-e rajta **zászló**.

```
typedef struct Mezo {
    bool akna;
    int kozelbenAknak;
    bool lathato;
    bool zaszlo;
} Mezo;
```

A **játéktábla** egy **sor x oszlop** dinamikusan lefoglalt 2Dimenziós **mezők**ből álló mátrixba kerül.
Példa:

```
if (tabla[kurzor.x][kurzor.y].akna)
    aknak_robbantasa(tabla, sorszam, oszlopszam);
```

A játékban nyilakkal lehet lépkedni, ezért egy **Koordináta (x, y)** típusú **kurzor** kerül mozgatásra, a programban **x** jelöli a **sor**ok, **y** az **oszlop**ok számát (azért lett létrehozva, hogy könnyebb legyen a konzol koordinátáit bejárni az econio segítségével)

```
typedef struct Koordinata {
    int x;
    int y;
} Koordinata;
```

Ranglistához tartozó:

A ranglisták:

```
#define konnyuLista "dicsoseglistaKonnyu.txt"
#define kozepesLista "dicsoseglistaKozepes.txt"
#define nehezLista "dicsoseglistaNehez.txt"
```

Ezek a typdefs.h-ban találhatóak, tetszés szerint át lehet nevezni fájlokat (habár az eddig létrehozottokat nem fogja tudni tovább kezelni a program, érdemes első futás előtt átírni, ha igény van rá).

Ezekben **formázott**an lesznek tárolva az adatok, egy **max 20 betű**ből álló név **;** -vel elválasztva a megoldáshoz szükséges **idő**től (az idő egy másodpercben értendő szám), majd **sorvége jel**lel elválasztva a következő adattól.
Példa:
„*Hosszú név1*;120
Hosszú név2;150”
A program csak helyes típusú adatokat fogad el új eredmények mentésekor.
Mivel .txt formátumban ment, így az eredmények könnyebben olvashatóak a program elindítása nélkül is, azonban a program nem arra lett tervezve hogy rossz formátumú fájlt is tudjon kezelni, így a manuálisan való beleírás kerülendő.

A lista egy eleme:

```
typedef struct Eredmeny {
    char *nev;
    int ido;
    struct Eredmeny *kov;
} Eredmeny;
```

Menühöz tartozó:

A különböző menükben való navigálás, illetve a játék kezdése állapotgéppel történik.

```
Menu menu = fomenu;
while (menu != kilepes) {
    switch (menu) {
        case fomenu:
            menu = menu_fomenu_vezerles(kurzor);
            break;
        case kilepomenu:
            menu = menu_kilepes_megerosites_vezerles(kurzor);
              break;
```

És így tovább…
A meghívott függvények visszaadják, hogy melyik menüpont, melyik nehézségű játék legyen elindítva, melyik ranglista legyen kiírva.

Függvények:

main.c

Konzolt Közép-Európaira állítja a konzolt (ékezetek megjelenítése).
A menüket kezeli.

ranglista.c

- `void uj_eredmeny_beszuras(Eredmeny **, char *, int);`
  A kapott név és idő szerint új elemet szúr az eredmények listájába, megfelelő helyre. (Az eredmények listája idő szerint növekvő sorrendbe van rendezve.)
  Paraméterek: A lista első elemére mutató mutatónak a mutatója (Azért, mert ha a lista első eleme NULL akkor is be lehessen szúrni)
  Eredményhez tartozó név és idő (másodpercben).

- `Eredmeny *eredmenyek_listazasa(char *);`
  Láncolt listává alakít formázott szöveges dokumentumot.
  Paraméter a szöveges dokumentum elérése *(pl.:„Valami.txt”)*. Elérésének útja relatív a .exe fájlhoz.
  Visszaadja az első elemre mutató mutatót.
  Ha nem létezik a fájl akkor NULL-al tér vissza.

- `void free_eredmeny_lista(Eredmeny *);`
  Felszabadítja a kapott láncolt listát.
  Paraméter a lista első elemére mutató mutató.

- `Menu ranglista_kiiras(char *);`
  Kiírja a kapott dokumentumból az adatokat.
  Paraméter: a dokumentum elérése.
  Visszatér: Főmenü

  - `void eredmenyek_kiirasa_kepernyore(Eredmeny *);`
    Végigmegy a láncolt listán és kiírja *„Hosszú név: perc másodperc”* formátumban
    Paraméter: Eredmények láncolt listája

- `void eredmeny_beolvasas(Nehezseg, int);`
  A felhasználótól megerősítés után kér egy megfelelő nevet, majd elmenti
  Paraméter: nehézség, idő(másodpercben)

- `bool eredmeny_beolvasas_megerosites();`
  Megkérdi a felhasználót, hogy el szeretné-e menteni az eredményét.
  Visszatérés: True ha igen.

- `bool vanbenneamineknemkene(char *);`
  Ellenőrzi, hogy megfelelő nevet adott a felhasználó.
  Paraméter: név
  Visszatérés: True ha nem.

- `void eredmenylista_elmentese(Eredmeny *, Nehezseg);`
  Elmenti a megadott helyre az eredmények listáját.
  Paraméterek: Lista eleje, Nehézség típusú nehézség

- `void eredmenyek_kiirasa_fajlba(Eredmeny *, FILE *);`
  Az eredmények listáját formázza és beírja fájlba.
  Paraméterek: Lista eleje, fájl

menüvezerles.c

- `void gotoxy(Koordinata *, int, int);`
  x,y koordinátára mozgatja a kurzort és a képernyőn is odamegy
  Paraméterek: kurzorra mutató pointer (hogy lehessen változtatni az értékeit), x és y koordináta

- `void menu_navigacio_kiiras();`
  menük navigációját kiírja

- `Menu menu_fomenu_vezerles(Koordinata);`
  Főmenü
  Paraméter: kurzor
  Visszatérés: Játékválasztómenü vagy Ranglistamenü vagy Kilépőmenü

- `void menu_fomenu_kiiras();`
  főmenü kiírása

- `Menu menu_kilepes_megerosites_vezerles(Koordinata);`
  Kilépőmenü
  Paraméter: kurzor
  Visszatérés: Kilépés (ezután vége is a programnak) vagy Főmenü

  - `void menu_kilepes_megerosites_kiiras();`
    Kilépőmenüt kiírja

- `Menu menu_uj_jatek_menu_vezerles(Koordinata);`
  Új játék menü
  Paraméter: Kurzor
  Visszatérés: Könnyű/Közepes/Nehéz Játék vagy Főmenü

  - `void menu_uj_jatek_menu_kiiras();`
    Új játék menüjét kiírja

- `Menu menu_ranglista_vezerles(Koordinata);`
  Ranglista menü
  Paraméter: kurzor
  Visszatérés: Könnyű/Közepes/Nehéz Ranglista vagy főmenü

  - `void menu_ranglista_kiiras();`
    Ranglistamenüt kiírja

jatek.c

- `Menu jatek_kezdes(int, int, int, Nehezseg);`
  Elindítja a táblafoglalást, inicializációt, táblavezérlést.
  Visszatérés előtt elindítja a táblát felszabadító függvényt.
  Paraméterek: Sorszám, Oszlopszám, Aknák száma, Nehézség
  Visszatérés: Főmenü

  - `Mezo **tabla_foglalas(int, int);`
    Dinamikusan lefoglalja a táblát (sorszám x oszlopszám mátrixba).
    Paraméterek: Sorszám, Oszlopszám
    Visszatérés: A létrehozott táblára mutató mutató

  - `void tabla_felszabaditas(Mezo **, int);`
    Felszabadítja a lefoglalt táblát.
    Paraméterek: Sorszám

  - `void tabla_inicializacio(Mezo **, int, int);`
    Mezők zászló, akna és láthatóságának paramétereit False-ra állítja.
    Paraméterek: Táblára mutató mutató, sorszám, oszlopszám

  - `Menu tabla_vezerles(Mezo **, int, int, int, Nehezseg);`
    Táblán navigál és bemenetekre megfelelő függvényeket indít el.
    „***Szóköz***”: Zászló lerakás és felszedés.
    „***Enter***”: Mező felfedés (ha nincs rajta zászló).
    Ha ez volt a legelső felfedett mező akkor szétszórja az aknákat a többi mezőn. (Hogy az első lépés ne okozza a játékos vesztét.)
    Ha akna volt akkor felrobbantja az összeset (megmutatja) és a játék vége menüt hozza fel.
    Ha ez volt az utolsó nem aknamező akkor felhozza a győzelmi menüt.
    „***Nyíl billentyűk***”: Navigálás mezők közt (a kijelölt mezőt jelezve).
    „***Esc***”: Játékból való kilépés megerősítő menüjét előhozza.
    Paraméterek: Táblára mutató mutató, sorszám, oszlopszám, aknaszám, nehézség
    Visszatérés: főmenü

- `void tabla_akna_szetszoras(Mezo **, int, int, int, Koordinata);`
  A tábla aknaszám számú mezőjére aknát tesz, kihagyva a kurzor koordinátáját.
  Paraméterek: Táblára mutató mutató, sorszám, oszlopszám, aknaszám, kurzor

- `void tabla_mezok_szamozasa(Mezo **, int, int);`
  A tábla nem akna mezőinek a körülötte lévő aknák számát beállítja.
  Paraméterek: Táblára mutató mutató, sorszám, oszlopszám

  - `int aknae(Mezo);`
    Megnézi az adott mezőről hogy van-e akna rajta.
    Paraméter: Mező
    Visszatérés: 1, ha van akna rajta, 0, ha nincs

- `void tabla_rajzolas(Mezo **, int, int);`
  Kirajzolja a táblának a mezőit.
  Paraméterek: Táblára mutató mutató, sorszám, oszlopszám

- `void kurzor_kiir(Koordinata, int);`
  Kiírja a kurzor helyét a tábla mellé.
  Paraméterek: kurzor, oszlopszám

- `void mezo_rajzolas(Mezo **, Koordinata, bool, int, int);`
  Kirajzol egy mezőt, ha ki van jelölve akkor más színnel (Így a kurzor jelenlegi állapotát lehet látni).
  Paraméterek: Táblára mutató mutató, jelölve, kurzor, sorszám, oszlopszám

- `void mezo_felfedes(Mezo **, Koordinata, bool, int, int, int *);`
  A mezőt amin a kurzor van felfedi (hacsak nincs már felfedve).
  Ha felfedte akkor a felfedett mezők számlálót növeli (ahhoz kell hogy tudjuk mikor nyer a játékos).
  Ha nem volt körülötte akna akkor elindítja a környező mezők felfedését.
  Paraméterek: Táblára mutató mutató, jelölve, kurzor, sorszám, oszlopszám, felfedett mezők számára mutató pointer

- `void kornyezo_mezo_megjelenites(Mezo **, Koordinata, int, int, int *);`
  Felfedi a kapott koordinátájú mező körüli mezőket. (Így rekurzióval felfedi az egymás mellett lévő „üres” mezőket a kurzortó indulva).
  Paraméterek: Táblára mutató mutató, kurzor, sorszám, oszlopszám, felfedett mezők számára mutató pointer

- `void aknak_robbantasa(Mezo **, int, int);`
  Kirajzolja az aknákat (piros háttérrel, jelezve hogy felrobbantak).
  Paraméterek: Táblára mutató mutató, sorszám, oszlopszám

- `void tabla_iranyitas_kiiras(int);`
  Kiírja a tábla alá hogyan kell navigálni.
  Paraméterek: sorszám

- `Menu jatekos_nyert(int, Nehezseg, int);`
  Gratulál a játékosnak (A tábla még látható), majd meghívja az eredmény beolvasást.
  Paraméterek: oszlopszám, nehézség, idő(másodpercben)
  Visszatér: Főmenü

- `Menu jatekos_vesztett(int, int);`
  Kiírja a játékosnak hogy vesztett, valamint hogy mennyi ideig tartott a játék.
  Paraméter: oszlopszám, idő
  Visszatérés: Főmenü