

Egy klasszikus aknakereső játék parancssoros CLI változata.

C nyelven készült, és a grafikus felület helyett a terminálban fut.

A cél, hogy működésében hasonlítson az ismert aknakeresőre:

véletlenszerűen elhelyezett aknák, megnyitható mezők, flag funkció, időmérés és fájlba mentett eredmények.

FORDÍTÁS ÉS FUTTATÁS

Fordítás parancssorból:

```
gcc main.c board.c game.c history.c -o aknakereso
```

Futtatás Linux/Mac alatt:

```
./aknakereso
```

Windows esetén:

```
aknakereso.exe
```

FŐ FUNKCIÓK

A program egy főmenüből indul, ahonnan a következő lehetőségek érhetők el:

- Új játék kezdése
- Játékosnév beállítása
- Nehézség módosítása
- Korábbi eredmények megtekintése
- Eredmények törlése
- Kilépés

A játék menete:

A pálya egy táblázat, amely mezőkből áll.

A játékos parancsokat adhat meg a terminálban:

- m sor:oszlop → mező megnyitása (például m 5:3)
- f sor:oszlop → flag elhelyezése vagy levétele (például f 2:7)
- q → kilépés a játékból

A játék automatikusan méri az időt, és győzelem esetén elmenti az eredményt a history.csv fájlba.

A „guest” névvel játszott körök nem kerülnek mentésre.

VIZUÁLIS MEGJELENÍTÉS

A program a terminálban színes kimenetet használ.

A különböző elemek más-más színnel jelennek meg, hogy könnyebb legyen áttekinteni a pályát.

x : még nem nyitott mező

* : flag (jelölt mező)

B : bomba (piros színnel a játék végén)

szám : hány bomba van a környező mezőkben (zöld)

sor/oszlop számok : világoskék színnel

A színezés ANSI escape kódokkal történik, ami Linuxon és újabb Windows terminálok is működik.

PROGRAM FELÉPÍTÉSE ÉS FÁJLOK

board.h / board.c

A pályakezelés itt történik.

A Cella struktúra tárolja az egyes mezők adatait:

```
typedef struct {  
    int isBomb;  
    int isOpen;  
    int isFlagged;  
    int around;  
} Cella;
```

Főbb függvények:

tabla_gen(): létrehozza dinamikusan a pályát

bomba_gen(): véletlenszerűen elhelyezi az aknákat

korulotte_szamit(): kiszámítja, hány bomba van egy mező körül
free_table(): felszabadítja a foglalt memóriát játék végén
A függvények gondoskodnak arról, hogy a játék elején minden mező alapértelmezett értékekkel legyen feltöltve, majd a szükséges adatok kiszámításra kerüljenek.
game.h / game.c
A játékmenet logikáját tartalmazza.
Kezeli a pálya kirajzolását, a mezők megnyitását, a flag váltását és a képernyő frissítését.
A rekurzív nyitás miatt, ha egy üres (nullás) mezőt nyit a játékos, a körülötte lévő mezők automatikusan megnyílnak.
A kirajzolás ANSI színekkel történik, ami áttekinthetőbbé teszi a játékot.
history.h / history.c
Az eredmények mentéséért és kezeléséért felel.
A fájl neve history.csv.
A program indításkor ellenőrzi, hogy a fájl létezik-e, és ha nem, létrehozza.
A fájl formátuma:
name,time,difficulty
bela,120,144
nina,85,100
main.c
A főprogram, amely a menürendszert és a játék indítását kezeli.
A jatek() függvény a játékmenet fő logikáját valósítja meg, kezeli a felhasználói parancsokat, az időmérést, valamint a győzelem és a vereség feltételeit.
JÁTÉKSZABÁLYOK
A cél a biztonságos mezők felfedése anélkül, hogy aknára lépnénk.
Vereség akkor történik, ha bomba mezőt nyitunk meg.
Győzelem akkor következik be, ha minden nem-bomba mezőt sikerül megnyitni.
A számok megmutatják, hány bomba van a nyolc környező mezőben.
A nullás értékű mezők automatikusan továbbnyílnak.
A flag-ek a gyanús helyek jelölésére szolgálnak, így elkerülhető, hogy véletlenül aknára lépjünk.
MEMÓRIAKEZELÉS
A pálya dinamikusan kétdimenziós tömbként jön létre.
Minden sor külön malloc() hívással foglалódik, a játék végén pedig a free_table függvény szabadítja fel őket.
A felszabadítások sorrendje és típusa minden esetben helyesre van állítva.
TECHNIKAI MEGJEGYZÉSEK
A színes kimenet ANSI escape kódokkal működik.
A véletlenszerű akna-elhelyezés srand(time(NULL)) hívással történik.
Az időmérés a time() függvénnyel van megvalósítva.
A program Windows és Linux rendszeren is fordítható.
A játék rekurzív működése biztosítja, hogy a nullás mezők automatikusan továbbnyíljanak.