

# Aknakereső Dokumentáció

Készítette : Kovács Balázs  
Neptun : BGM94Z

Ez az Aknakereső **macOS** és különböző **Linux** disztribúciók alatt fut.

## Az Aknakereső játék megvalósítása (*Konzolos alkalmazásként*)

Az akanakereső játék célja, hogy megtalálja a játékos az összes aknát, úgy hogy azokat elkerüli.

**A programnak két nézete van:**

- Főmenü
- Játékmenet

A főmenüből új játék indításával juthatunk el a játéknézetbe, a játék végén a program automatikusan kilép.

**A játékos megválaszthatja új játék indításakor a:**

- Játékmező méretét <x> <y>
- A játékmezőn található aknák számát <n>

**Az aknakeresőhöz szükséges funkciók:**

A program képes automatikusan felderíteni egy környéket, az számolni a játék kezdete óta eltelt időt, és a játékos tud megjelölni aknának gondolt mezőt.

- Egy mező környékének automatikus felderítését a **flood fill** rekurzív algoritmus végzi el, ami addig fut, ameddig nem talál olyan mezőt, amely környezetében akna található.
- Az aknakereső játékban úgy tudjuk megnézni, hogy nyert-e a játékos, hogy számoljuk a már felfedett mezők számát, és az amikor az összes felfedezhető mező számával egyenlő lesz (Összes mező - Aknák száma), akkor nyert a játékos. (Ehhez szükséges az aknák száma és egy 2d tömb, amiben számontartjuk, hogy mely mezők vannak felderítve)

## Adatszerkezetek

A játékmenethez szükséges három 2d tömböt dinamikus memóriakezeléssel hozza létre a program. Ezt a három tömböt a **minesweeper.h**-ban található **GameField** adattípusban található. A foglalt területet a játék végén felszabadítja a program.

A programban nincs memóriaszivárgás, ezt a **debugmalloc** könyvtárral ellenőrizhetjük.

```
typedef struct GameField {  
    char **field, **visible, **opened;  
    int size_X, size_Y, mine_C;  
    gameTime timer;  
} GameField;
```

## További segédadatszerkezetek:

- Koordináta adattípus, X és Y koordinátát tárol.

```
typedef struct Coordinate {  
    int x, y;  
} Coordinate;
```

- Idő adattípus, a játék kezdete óta eltelt időt tárolja. (perc, másodperc formában)

```
typedef struct gameTime {  
    int min, sec;  
} gameTime;
```

## Header File-ok

### minesweeper.h

```
typedef struct GameField {  
    char **field, **visible, **opened;  
    int size_X, size_Y, mine_C;  
    gameTime timer;  
} GameField;  
  
/* A játékmenethez szükséges változókat és segéd tömböket tartalmazó  
adattípus.  
* Három tömböt tárol mely szükséges a játékmenethez:  
* A field tömbben tároljuk az aknák helyét, és a környező aknamezők számát.  
* A visible tömb az amit a játékos láthat, ezt módosítjuk csak a játék  
során  
* Az opened tömbben rögzítjük a már felnyitott mezőket, ez a győzelemi  
eset felismeréséhez kell.  
* A size_X és size_Y változók rögzítik a pálya méretét.  
* A mine_C változó a pályán található aknák mennyiségét rögzíti.  
* A gTime adattípusban tároljuk a játék kezdete óta eltelt időt.  
*/
```

```
typedef struct Coordinate {  
    int x, y;  
} Coordinate;  
  
// Koordináta adattípus
```

```
typedef struct gameTime {
    int min, sec;
} gameTime;

// Idő adattípus
```

```
void gameLoop(GameField gf);
```

```
/* A game loop maga a játékmenet, a ciklusnak akkor van vége, amikor a
játékos
* veszít, vagy nyer. Ameddig fut a ciklus, kirajzolja a pályát és várja
* játékos lépését, amit a guessing függvény dolgoz fel.
* Paraméterek: GameField struktúra
* Visszatérési érték: void
*/
```

```
void guessing(GameField gf, Coordinate guess, int cmd);
```

```
/* A játékos lépését dolgozza fel a függvény.
* A cmd változó lehet 1 vagy 2, ha 1, akkor felfedez, ha 2, akkor
megjelöl egy mezőt.
* Ha aknamezőt próbál felfedni a játékos a gameloop leáll.
* Ha nincs akna a mezőn, derítse fel a program a környező nem akna
mezőket.
* Ha megjelölni szeretne egy mezőt a játékos, csak olyat tud ami még
nincs felderítve.
* Paraméterek: GameField struktúra, Coordinate struktúra, Lépés típusa
* Visszatérési érték: void
*/
```

```
bool checkWin(GameField gf);
```

```
/* Ellenőrzi, hogy nyert-e a játékos.
* Paraméterek: GameField gf
* Visszatérési érték: bool
*/
```

```
void endScreen(GameField gf, bool win);
```

```
/* A játék végén kiírt "Nyert" / "Veszített" szöveget írja ki, ha
veszített a játékos, felfedi az összes mezőt.
* Leállítja a gameloop-ot.
* Paraméterek: GameField struktúra, Egy bool érték (true ha nyert, false
ha veszített a játékos)
```

```
* Visszatérési érték: void
*/
```

```
char adjacentMines(GameField gf, int x, int y);

/* Megszámolja, hogy egy mező környezetében hány akna található, és azt rögzíti.
 * Paraméterek: GameField struktúra, X és Y koordináta
 * Visszatérési érték: char (Környező mezőkben lévő akna szám)
 */
```

```
bool isMine(GameField gf, Coordinate c);

/* Ellenőrzi, hogy adott koordinátán található mezőn van-e akna.
 * Paraméterek: GameField struktúra, Koordináta struktúra
 * Visszatérési érték: bool
 */
```

```
void floodFill(GameField gf, int x, int y);

/* A "Flood Fill" rekurzív algoritmus megvalósítása, amely a környező nem-akna mezők felfedezéséért felel.
 * Paraméterek: GameField struktúra, X és Y koordináta
 * Visszatérési érték: void
 */
```

## mainMenu.h

```
void mainMenu();

/* A főmenü működését vezérli.
 * Kiírja a menüpontokat, és a felhasználó választását feldolgozza.
 * Paraméterek: –
 * Visszatérési érték: void
 */
```

```
void newGame(GameField mf);

/* A függvény egy új játékmenetet indít el. Először bekéri a felhasználótól, hogy mekkora pályán és hány aknával szeretne játszani, majd legenerálja a játékmenethez
```

```
* szükséges 2D tömböket.  
* Paraméterek: GameField struktúra  
* Visszatérési érték: void  
*/
```

```
char **allocateMemory(char **array, GameField gf);
```

```
/* Lefoglalja a kért dinamikus 2D tömböt, amely méretét a második  
paraméterből kapja.
```

```
* Paraméterek: Lefoglalandó tömb, GameField struktúra  
* Visszatérési érték: ** pointer a lefoglalt 2d tömbre  
*/
```

```
void freeMemory(GameField gf);
```

```
/* Felszabadítja az összes dinamikusan foglalt tömböt, ami a játékmenet  
szüksége.
```

```
* Paraméterek: GameField struktúra  
* Visszatérési érték: void  
*/
```

## render.h

```
void render(GameField gf, bool reveal);
```

```
/* Kirajzolja a pályát, amit a játékos láthat.  
* Ha a második paraméter true, feldedve rajzolja ki a pályát.  
* Paraméterek: GameField struktúra  
* Visszatérési érték: void  
*/
```

```
void clearscreen();
```

```
// system("clear") parancs
```