Amőba

Kovács Donát

Neptun: BYVO9O

A program leírása

A program képes ember vs ember, ember vs számítógép és számítógép vs számítógép játékmenetek futtatására. Három különböző stratégiával játszó számítógép játékos van. Az első a támadó, melynek a célja, hogy minél gyorsabban összegyűjtsön a saját jeléből ötöt egy sorban/oszlopban/átlóban. Ennek az ellentettje a második, amely azt nézi, hogy az ellenfelének hol van a leghosszabb sora/oszlopa/átlója a jelekből és azt zárja le a saját jelével. A támadó és a védekező ugyanazt az evaluate (kiértékelő) függvényt használja, csak a támadó a saját jelét nézi, hogy hogyan tud a lehető leghosszabb sorokat kialakítani, míg a védekező az ellenfelének a jeleit nézi, hogy hogyan tud olyan helyre tenni, amivel a legnagyobb „kárt" okozza neki. A harmadik pedig egy random, amely egy olyan helyre rak véletlenszerűen, amelyik 1 sugarú környezetében bármelyik jelből már van lerakva, ezzel garantálva, hogy lehetőség szerint összefüggő maradjon a játéktér.

Bármilyen operációs rendszeren, bármilyen C fordítóval lefuttatható a program. Nem igényel a szabványos könyvtárakon kívül semmit.

A játék egésze egy struktúrában (Game) van tárolva (játékosok, opciók, pálya). A játékosok egy két elemű tömböt alkot, a játékpálya pedig egy tömbök tömbje.

A program öt modulból áll:

1. program.c – csak a main függvényt tartalmazza
2. moves.c – a játéklépésekkel kapcsolatos függvényeket tartalmazza → hozzá tartozó header file: moves.h
3. board.c – a játéktáblával kapcsolatos függvényeket tartalmazza → hozzá tartozó header file: board.h
4. menu.c – a menüket/almenüket és a játék menetét tartalmazza → hozzá tartozó header file: menu.h
5. base.c – ebben a sztring és int beolvasó függvények vannak → továbbá a hozzátartozó base.h headerben található meg a program alap struktúrája.

Függvények:

2. moves.c

`bool human_move(Game *g, int *row, int *col)` – bekéri az ember lépését, majd ha lehetséges a lépés, akkor visszaadja a sort és az oszlopot a megadott pointerekben.

Paraméterek:

g – a játék struktúrája
row – sorra mutató pointer
col – oszlopra mutató pointer

Visszatérési értéke pedig true, hogyha lehetséges a lépés és false, ha nem vagy ha a felhasználó visszalépett a menübe.

`bool computer_move_offensive(Game *g, int *row, int *col)` – a támadó számítógép stratégiájának az algoritmusa: ha elsőként lép, akkor a tábla közepének a sorát és oszlopát adja vissza, máskülönben végigmegy a játéktáblán és kiértékeli, hogy melyik cella lenne a legjobb lépés a nyertes vonal létrehozásához és akkor annak a cellának a sorát és oszlopát adja vissza.

Paraméterek:

g – a játék struktúrája
row – sorra mutató pointer
col – oszlopra mutató pointer

Visszatérési értéke pedig true, ha lépett a számítógép és false, ha a felhasználó visszalépett a menübe.

`bool computer_move_defensive(Game *g, int *row, int *col)` – a védekező számítógép stratégiájának az algoritmusa: ha elsőként lép, akkor a tábla közepének a sorát és oszlopát adja vissza, máskülönben végigmegy a játéktáblán és kiértékeli, hogy melyik cella lenne a legjobb lépés az ellenfélnek a nyertes vonal létrehozásához és akkor annak a cellának a sorát és oszlopát adja vissza.

Paraméterek:

g – a játék struktúrája
row – sorra mutató pointer
col – oszlopra mutató pointer

Visszatérési értéke pedig true, ha lépett a számítógép és false, ha a felhasználó visszalépett a menübe.

`bool computer_move_random(Game *g, int *row, int *col)` – a random lépő számítógép stratégiájának az algoritmusa: ha elsőként lép, akkor a tábla közepének a sorát és oszlopát adja vissza, máskülönben generál két olyan véletlen számot (row és col), amik által meghatározott pozíció a táblára már elhelyezett valamely jel szomszédságába esik, és ezeket a sor- és oszlopszámokat adja vissza.

Paraméterek:

g – a játék struktúrája
row – sorra mutató pointer
col – oszlopra mutató pointer

Visszatérési értéke pedig true, ha lépett a számítógép és false, ha a felhasználó visszalépett a menübe.

`bool mark_player_move(Game *g, int row, int col)` – ha érvényes a lépés az adott helyen, akkor az aktuális játékos jelét rakja be a board tömbbe.

Paraméterek:

g – a játék struktúrája
row – sor
col – oszlop

Visszatérési értéke pedig true, hogyha érvényes a lépés és false, ha nem.

`bool next_move(Game *g, int *row, int *col)` – az aktuális játékos típusának megfelelő játékos lépés függvényt hívja meg, amely függvény a bemenetben megadott sor és oszlop pointerekben adja vissza a generált lépést.

Paraméterek:

g – a játék struktúrája
row – sorra mutató pointer
col – oszlopra mutató pointer

Visszatérési értéke pedig true, hogyha lehetséges a lépés és false, ha nem vagy ha a felhasználó visszalépett a menübe

3. board.c

`void destroy_board(Game *g)` – felszabadítja a tábla által lefoglalt memóriaterületet.
Paraméterek:
g – a játék struktúrája
Nincs visszatérési értéke.

`bool create_board(Game *g)` – lefoglalja a memóriát a tábla számára.
Paraméterek:
g – a játék struktúrája
Visszatérési értéke pedig true, ha sikerült és false, ha nem.

`bool save_game(Game *g)` – a játékállást kimenti egy fájlba.
Paraméterek:
g – a játék struktúrája
Visszatérési értéke pedig true, ha sikerült és false, ha nem.

`bool load_game(Game *g)` – betölti a játékállást a kimentett fájlból.
Paraméterek:
g – a játék struktúrája
Visszatérési értéke pedig true, ha sikerült és false, ha nem.

`void board_size(Game *g)` – bekéri a felhasználótól a tábla méretét és ellenőrzi, hogy lehetséges-e az adott méret beállítása, majd beállítja azt.
Paraméterek:
g – a játék struktúrája
Nincs visszatérési értéke.

`static void rowname(int index, char *str)` – bemenetként megkapja a sor indexét, és a paraméterként kapont sztringbe belerakja az adott sor nevét A,...,Z,AA,...,AZ formátumban.
Paraméterek:
index – egy sor indexe
str – sztring pointer, melynek legalább 2 karaktert és a lezáró nullát kell tudnia fogadni
Nincs visszatérési értéke.

`bool is_board_full(Game *g)` – végigmegy a játéktáblán és visszaadja, hogy betelt-e
Paraméterek:
g – a játék struktúrája
Visszatérési értéke pedig true, ha betelt a tábla és false, ha van még legalább egy üres mező.

`bool is_board_empty(Game *g)` – végigmegy a játéktáblán és visszaadja, hogy üres-e
Paraméterek:
g – a játék struktúrája
Visszatérési értéke pedig true, ha üres a tábla és false, ha van legalább egy kitöltött mező.

`bool has_neighbour(Game *g, int row, int col)` – visszaadja, hogy van-e a bemenetre kapott pozícióval szomszédos lévő cellában (függőlegesen, vízszintesen vagy átlósan) „X” vagy „O”
Paraméterek:
g – a játék struktúrája
row – a sor száma
col – az oszlop száma

Visszatérési értéke pedig true, ha van legalább egy szomszédja és false, ha nincs.

`static int evaluate(Game *g, int row, int col, bool longest_line_mode)` – többcélú függvény, amely kiértékeli az állást az adott pontban: ha longest line módban van hívva, akkor visszaadja a leghosszabb egybefüggő sorhosszt, különben visszaad egy generált értékelési pontszámot az adott pozícióra

Paraméterek:

g – a játék struktúrája
row – a sor száma
col – az oszlop száma
longest_line_mode – a longest line mód feltétele
Visszatérési értéke pedig pedig egy egész szám, amely a **longest_line_mode**-tól függ.

`static int longest_line(Game *g, int row, int col)` – visszaadja a leghosszabb egy sorban/oszlopban/átlóban lévő egybefüggő azonos jelek számát.

Paraméterek:

g – a játék struktúrája
row – a sor száma
col – az oszlop száma
Visszatérési értéke pedig a leghosszabb egybefüggő azonos jelek száma.

`static int evaluate_move(Game *g, int row, int col)` – generál egy értékelési pontszámot, melyben a legnagyobb részt a leghosszabb vonalak száma számít, de egyben a rövid vonalakat is figyelembe veszi.

Paraméterek:

g – a játék struktúrája
row – a sor száma
col – az oszlop száma
Visszatérési értéke pedig az adott pozíció értékelési pontszáma.

`void evaluate_board(Game *g, int *row, int *col, char sign)` – megkeresi és visszaadja a következő lehetséges lépést, amely a legmagasabb értékelési pontszámmal rendelkezik.

Paraméterek:

g – a játék struktúrája
row – sorra mutató pointer
col – oszlopra mutató pointer
sign – erre a játékos jelre végzi el a pozíció értékelést
Visszatérési értéke nincsen.

`bool is_game_won(Game *g)` – megnézi, hogy nyert-e valamelyik játékos.

Paraméterek:

g – a játék struktúrája
Visszatérési értéke pedig true, ha nyert valaki és false, ha nem.

`void print_board(Game *g)` – megjeleníti a táblát, és a játék végén ha az adott cella beletartozik egy nyertes vonalba, akkor zárójelbe rakva jeleníti meg (pl.: (X)) azt.

Paraméterek:

g – a játék struktúrája
Nincs visszatérési értéke.

4. menu.c

`bool ask_yes_or_no(char *question)` – bemenetként kap egy sztringet, amelyet kiír kérdésként, majd visszaadja hogy igennel vagy nemmel válaszolt-e a felhasználó.
Paraméterek:
question – egy sztring, amely egy kérdést tartalmaz
Visszatérési értéke pedig true, ha igen a válasz és false, ha nem.

`void set_player_name(char *name)` – bekéri a játékos nevét, és elrakja a bemenetbe.
Paraméterek:
name – az egyik játékos új neve kerül bele
Nincs visszatérési értéke.

`char *get_player_type_name(int type)` – a bemenetére kapott játékos típus számnak a szöveges megnevezését adja vissza.
Paraméterek:
type – a játékos típusa
Visszatérési értéke pedig egy sztring, amely a típus megnevezése.

`void set_player_type(int player_number, int *type)` – bekéri a felhasználótól a játékos típusának a számát és eltárolja a bemenetre kapott változóba.
Paraméterek:
player_number – az aktuális játékos száma
type – ebbe lesz eltárolva a játékos új típusának a száma
Nincs visszatérési értéke.

`static void play_game(Game *g)` – elindítja a játékot, és egy ciklusban működteti lépésenként, amíg véget nem ér a játék vagy amíg a felhasználó vissza nem lép a menübe.
Paraméterek:
g – a játék struktúrája
Nincs visszatérési értéke.

`static void play_menu(Game *g)` – a play menüt megjeleníti és vezérli.
Paraméterek:
g – a játék struktúrája
Nincs visszatérési értéke.

`static void options_menu(Game *g)` – az options menüt megjeleníti és vezérli.
Paraméterek:
g – a játék struktúrája
Nincs visszatérési értéke.

`static void print_rules(void)` – kiírja a játékszabályokat.
Nincs bemenő paramétere és nincs visszatérési értéke.

`void main_menu(Game *g)` – a főmenüt megjeleníti és vezérli.
Paraméterek:
g – a játék struktúrája
Nincs visszatérési értéke.

`void init_game(Game *g)` – kezdőértéket ad a játék struktúra elemeinek.
Paraméterek:
g – a játék struktúrája
Nincs visszatérési értéke.

`void exit_game(Game *g)` – a program által lefoglalt erőforrásokat felszabadítja.
Paraméterek:
g – a játék struktúrája
Nincs visszatérési értéke.

5. base.c

`void input_str(char *str, int size)` – bekér egy sztringet és elrakja a bemenetére kapott változóba.
Paraméterek:
str – egy 'size' elemet fogadni képes sztring
size - a megadott sztring hossza
Nincs visszatérési értéke.

`void input_int(int *number` – bekér egy sztringet, átkonvertálja integerbe és elrakja a bemenetbe.
Paraméterek:
number – ebbe kerül az átkonvertált integer
Nincs visszatérési értéke.