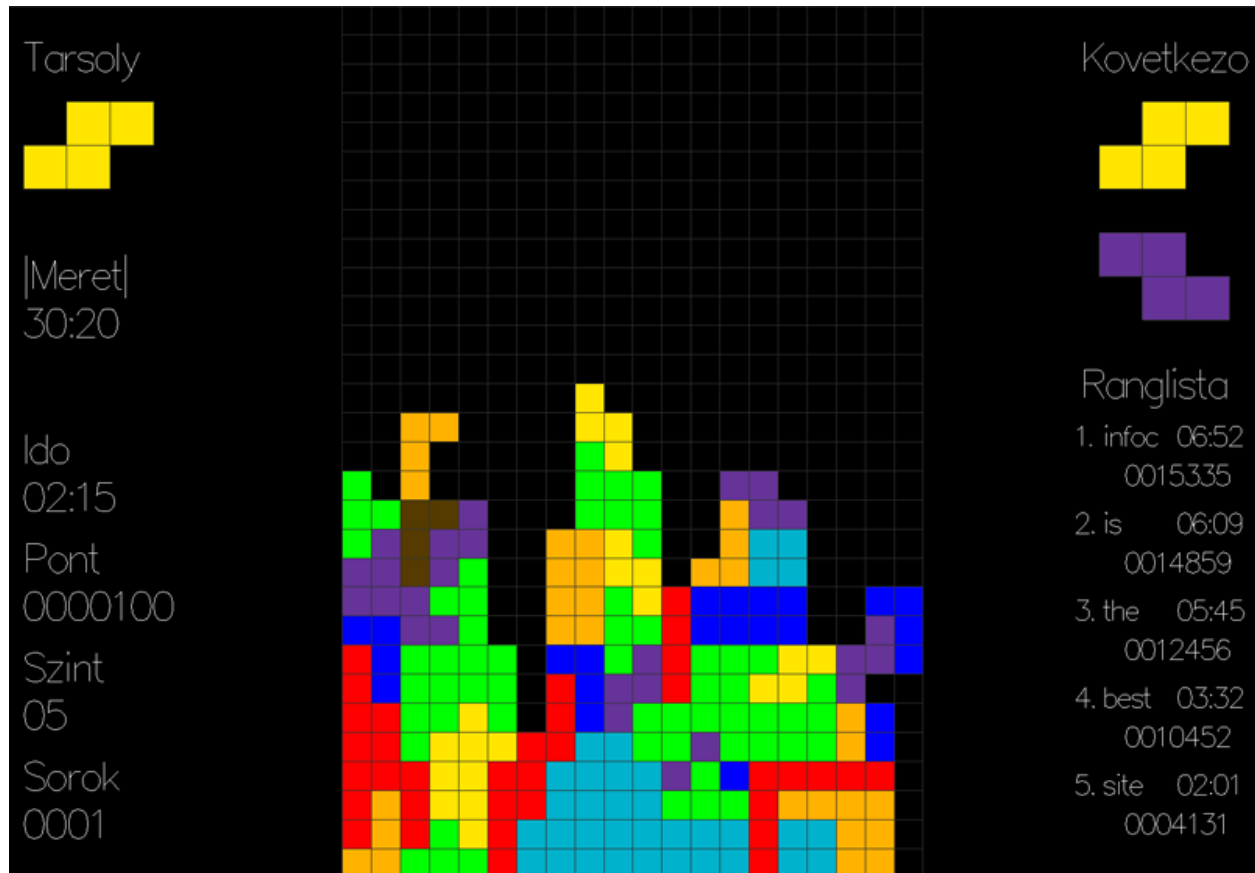


Tetris játék

Programozói dokumentáció




Tartalomjegyzék:

SZÜKSÉGES KÖRNYEZET, KÜLSŐ KÖNYVTÁRAK:	4
ADATSZERKEZETEK:	4
PÁLYA.....	4
HAND.....	4
JÁTÉK MŰKÖDÉSE, TERVEZÉSI MEGFONTOLÁSOK	5
PROJEKT FELÉPÍTÉSE	5
MAIN.....	5
<i>Belső globális változók</i>	5
<i>Függvények</i>	5
<i>Részletesebben a GLUT vezérlő része a játékban:</i>	6
MAP.....	7
<i>Az adatszerkezetek</i>	7
<i>Függvények</i>	7
HAND.....	8
<i>Az adatszerkezet</i>	8
<i>Függvények</i>	8
RANGLISTA.....	8
<i>Függvények</i>	8
MEGJELENITES	9
<i>RGBA színkód adatszerkezete</i>	9
<i>Függvények</i>	9
FÜGGVÉNYEK	10
MAIN	10
<i>void Jatek();</i>	10
<i>void display();</i>	10
<i>static void Felszabadit();</i>	10
<i>void keyboard(unsigned char key, int x, int y);</i>	10
<i>void specialKeys(unsigned char key, int x, int y);</i>	10
<i>void UjJatek(int argc, char** argv);</i>	11
MAP	11
<i>PalyaMatrix* MatrixFoglal(Palya const *vp);</i>	11
<i>void MatrixInit(Palya *vp, int const sor, int const oszlop);</i>	11
<i>void MatrixbaMasol(Palya *vp, Hand *hp);</i>	11
<i>bool Utkozes(Palya const *vp, Hand const *hp, bool const *bp, int x, int y);</i>	11
<i>int AltetrisKord(Palya const *vp, Hand const *hp);</i>	11
<i>void KovTetris(Palya *vp, Hand *hp, bool *vege);</i>	12
<i>void Eltuntet_sor(Palya *vp, int sor);</i>	12
HAND.....	12
<i>bool* TetrolVal(int melyik);</i>	12
<i>bool* HandFoglal(Hand* hp);</i>	12
<i>void HandInit(Hand* hp, int const *oszlop, int const melyik);</i>	12
<i>bool* Forgat_jobbra(Hand const *hp);</i>	13
RANGLISTA	13
<i>static void RangCsere(Ranglista *a, Ranglista *b);</i>	13

<i>void Nevhezir(Palya *vp, int c);</i>	13
<i>void NevbolTorol(Palya *vp);</i>	13
<i>void RanglistaRendez(Palya *vp);</i>	14
<i>void Ranglistament(Palya *vp);</i>	14
<i>void Ranglistabeolvas(Palya *vp);</i>	14
MEGJELENITES	14
<i>void initGL();</i>	14
<i>void Idozito(int idokoz);</i>	14
<i>void Ujrameretesz(GLsizei width, GLsizei height);</i>	15
<i>void KirajzInit(Palya *vp);</i>	15
<i>static void Szovegrajzol(char* szoveg, GLfloat x, GLfloat y, float meret);</i>	15
<i>static void RajzolNegyzet(GLfloat *Nsize, GLfloat x, GLfloat y, RGBA s, bool mode);</i>	15
<i>static void RajzolHaloba(Palya *vp, float x, float y, int sor, int oszlop, RGBA s, bool mode);</i>	15
<i>static RGBA SzinKonverter(int szin);</i>	15
<i>static void RajzolMatrix(Palya *vp);</i>	16
<i>static void RajzolTetris(Palya *vp, Hand *hp, int x, bool mode);</i>	16
<i>static void RajzolVTetris(Palya static *vp, bool* t, int size, float x, float y, int szin)</i>	16
<i>static void KirajzolIdo(Ido *t, float x, float y, float size);</i>	16
<i>static void KirajzolPont(int *pontos, float x, float y, float size);</i>	16
<i>static void KirajzolSzint(int *szints, float x, float y, float size);</i>	16
<i>static void KirajzolEltSorSzam(Palya *vp, float x, float y, float size);</i>	17
<i>static void RajzolTarsoly(Palya *vp);</i>	17
<i>static void RajzolKovi(Palya *vp);</i>	17
<i>static void RajzolRangLista(Palya *vp);</i>	17
<i>void JatekRajzol(Palya *vp, Hand *hp);</i>	17
<i>void GameOverRajzol(Palya *vp, Hand *hp);</i>	17

Szükséges környezet, külső könyvtárak:

- A játék  VisualStudio-ban íródott.
Miért? Legfőképp a praktikussága és kinézete miatt.
Sötét téma, GitHub kiegészítő és fejlesztőt segítő funkciók.
- A grafikai része OpenGL, a platformal való kommunikáció pedig GLUT.

Fordításhoz szükséges lépések:

- Ha az SDK verzióba beleköt, akkor a solutionre jobb klikkelés után „retarget solution”
- Project properties átállítása megfelelő könyvtárakhoz:
 - C/C++ fülnél a **General**-ban Additional Include Directorieshez hozzá kell adni, a játék könyvtárában lévő include mappát.
 - **Linker** fülnél a **General**-ban Additional Library Directories-hez a Tetris azon mappáját kell hozzáadni, ahol a freeglut.lib és dll van.
 - Szintén linker fülnél a **freeglut.lib**-et kell hozzáírni az Additional Dependencies-hez

Adatszerkezetek:**• Pálya**

A teljes pálya és annak adatai egy struktúrában vannak eltárolva.

Tartalmazza a **sor** és **oszlop** számot, a **sorösszeg**-eket, mely a könnyebben kigyúlt sorok kezelését könnyíti meg. Ezek mellett olyan statisztikai adatokat, mint a **szint**, az **idő**, mely egy perc és másodperc-el rendelkező struktúra, a **pontszám** az **Eltüntetett sorok száma** és a játékos **neve**, ami valójában csak a játék végén kerül bekérésre.

A megjelenítéshez még megtalálható benne a **pálya szélessége** és **magassága** és a **négyzetek mérete**.

A játékban még megtalálható a későbbre elrakás funkciója, melyet a **tarsoly** tesz lehetővé. Ez mellett pedig a **következő kettő**-t is láthatjuk. Ezek igazából csak egy egész értéket tárolnak, mely az adott tetris sorszám.

Végül pedig a nagyobb adatszerkezet a **Pálya mátrixa**, mely egy olyan dinamikus adatszerkezet, mely azt tárolja, hogy a pálya celláinak mi a színe és, hogy van-e az adott helyen.

A **ranglista** is itt kerül letárolásra, mely adatszerkezete az adott játékos nevéből, pontszámából és játékidejéből áll.

• Hand

Az egyik legegyszerűbb adatszerkezet, mivel csak annyit tárol, hogy dinamikusan a tetris hol van a pályához képest **x** és **y** kordinátában, mi a **színe**, **sorszám**, **színe** és jelenlegi **állása** egy logikai mátrixban tárolva.

Játék működése, tervezési megfontolások:

Maga a játéktér egy nagy mátrix, melyben adott mezők tartalmi vannak letárolva és a játék ezt folyamatosan adott feltételek szerint irányítja. A játéktér és a rajta mozgó tetrisketté szettem épp azért, hogy könnyebb legyen a feltételeket ellenőrizni, vagy épp az irányítást elvégezni. Ez azért volt jó, mivel így csak olyan függvényekre volt szükségem például az ütközés vizsgáláshoz, hogy adott koordinátával eltolva a tetris ütközne-e, vagy hasonló. Elforgatásnál és mozgásnál is sokkal hatékonyabbá tette a dolgokat, ugyanis eltolás csak koordináta változtatás, a forgatás pedig csak egy pár mátrix művelet, mely a függvények részénél részletesebben ki van fejtve. Mivel így csak egy pályát kell módosítani, ezért az olyan adatok, mint a statisztikák sokkal egyszerűbben nyílvántarthatók. Például ha teljesül az a feltétel, hogy megtelek egy sor, szimplán eltüntetjük a pályáról a megfelelő módon, majd adott statisztikai elemeket változtatunk. Miután megvoltak a megfelelő játék kezelő függvények a megjelenítést és a mechanikai részét kellett megvalósítani. A megjelenítés azon az elven működik, hogy meghív egy függvényt minden 10ms-onként ami szimplán belső óráként működve üzemelteti a játékot. Az irányítás szerencsére független ettől a belső órától, mivel event-ként vannak kezelve, tehát mikor megnyomunk egy gombot, akkor csak szimplán végrehajt adott függvényeket. Mivel minden része a programnak apró függvényekre van bontva, elég hatékonyan lehet továbbfejleszteni.

Projekt felépítése:

■ main

A játék függvényeit fogja össze, kezdeti inicializálásokat végzi el és a játékot működteti. Vannak itt olyan belső globális változók, amikre sajnos szükség van a glut miatt, ugyanis a működési elve azon alapul, hogy megadunk olyan függvényekre mutató pontereket, amiknek nincsenek paraméterei, egész pontosan ilyen például az, amit rendszeresen meghív az ablak újrarajzolásakor. Ezek a változók a fájlban kívül csak paraméterekként kerülnek átadásra.

Belső globális változók:

```
static Palya t;           //Pálya
static Hand h;           //Kézben lévő tetris
static int GravInterv = 0; //Gravitáció időköz
static int MpCounter = 0; //Másodperc számláló
static bool valthat = true; //Tarsolyba lehet-e tenni?
static bool vege = false; //Vége?
```

Függvények:

```
void Jatek();             //Játékot vezérlő változók módosítása
void display();           //Rendszeresen meghívódó függvény újrarajzolásra
static void Felszabadit(); //Mallocolt területek felszabadítása
void keyboard(unsigned char key, int x, int y); //Egyszerűbb gombok
void specialKeys(unsigned char key, int x, int y); //Speciális gombok
void UjJatek(int argc, char** argv); //Ujjatek inicializáció
```

Részletesebben a **GLUT** vezérlő része a játékban:

- `glutInit(&argc, argv);` // GLUT inicializáció
- `glutInitDisplayMode(GLUT_DOUBLE);` // Duplán bufferált mód
- `glutInitWindowSize(1000, 700);` // Ablak Kezdő szélesség, magasság
- `glutInitWindowPosition(` // Ablak pozíció
 `(glutGet(GLUT_SCREEN_WIDTH) - 1000) / 2,`
 `(glutGet(GLUT_SCREEN_HEIGHT) - 700) / 2);` //Középre elhelyezés
- `glutCreateWindow("Tetris játék");` // Ablak címe

- `glutDisplayFunc(display);` // Újrarajzoláskor lefuttatandó függvény beállítása
 # `void glutDisplayFunc(void (*func)(void));`

- `glutReshapeFunc(Ujrameretez);` // Ujraméretezéskor lefutó függvény
 # `void glutReshapeFunc(void (*func)(int width, int height));`
 A width és a height automatikusan átadódik

- `glutTimerFunc(0, Idozito, 10);` // Időzítő, 10-es értéket ad át az Idozito függvénynek
 # `void glutTimerFunc(unsigned int msec, void (*func)(int value), value);`
 Első érték, hogy mennyi idő múlva hívódjon meg, majd melyik függvény, és mit kapjon meg paraméterben az a függvény

- `glutKeyboardFunc(keyboard);` // Alap gombok kezelése
 # `void glutKeyboardFunc(void (*func)(unsigned char key, int x, int y));`
 key, hogy melyik gombot nyomta le, az x és y pedig, hogy hol volt az egér

- `glutSpecialFunc(specialKeys);` // Speciális gombok
 # `void glutSpecialFunc(void (*func)(int key, int x, int y));`

- `initGL();` // Extra inicializációhoz szükséges dolgok
- `glutMainLoop();` // Belépés a végtelen ciklusba

➔ Tehát a belső globális változókra szükség van.

■ map

Itt vannak a pályához kapcsolódó függvények és típusok.

Az adatszerkezetek:

```
typedef struct {          //Pálya
    bool e;                //Cellában van-e bármi?
    int c;                  //Mi a színe?
} PalyaMatrix;

typedef struct {          //Idő
    int p;                  //Perc
    int mp;                 //Másodperc
} Idő;

typedef struct Rang {     //Rang
    char *nev;              //Játékos neve
    int pont;               //Pontszáma
    Idő time;               //Ideje
} Ranglista;

typedef struct {          //Pálya
    int sor, oszlop;        //Pálya mérete
    float Nsize;            //Négyzetek mérete
    float width, height;    //Pálya szélesség, magasság
    int *sum;               //Soronkénti összeg
    int level;              //Jelenlegi szint
    Idő time;               //Eltelt idő
    int pont;               //Pontszám
    int ElSorSzam;          //Eltüntetett sorok száma
    PalyaMatrix *v;         //Pálya mátrixa
    int Tarsoly;            //Tarsolyban lévő tetris
    int KoviT[2];           //Következő Tetrisek
    char *nev;              //Jatekos neve
    Ranglista rlista[500];  //Ranglista
} Palya;
```

• Függvények:

```
//Memoria terület lefoglalása a pályának
PalyaMatrix* MatrixFoglal(Palya const *vp);
//A pálya inicializálása
void MatrixInit(Palya *vp, int const sor, int const oszlop);
//Tetris átmásolása a pályára
void MatrixbaMasol(Palya *vp, Hand *hp);
//Ütközés vizsgálat x és y eltolással bp tömbre
bool Utkozes(Palya const *vp, Hand const *hp, bool const *bp, int x, int y);
//Alsó tetris távolsága a jelenlegitől
int AltetrisKord(Palya const *vp, Hand const *hp);
//Következő tetrisre állítás
void KovTetris(Palya *vp, Hand *hp, bool *vege);
//Adott sor eltüntetése, majd fölötte lévők lejjebb húzása
void Eltuntet_sor(Palya *vp, int sor);
```








■ hand

A kézben lévő tetrishez kapcsolódó függvények és struktúra.

Az adatszerkezet:

```
typedef struct {    //Hand
    int x, y;        //Pozicioja a Maphez képest
    int color;        //Szine
    int melyik;       //Tetris tipusa
    int size;         //Mátrix mérete (size*size)
    bool* v;          //Tartalma
} Hand;
```

Belső globális változók, melyek a játékhoz kapcsolódó tetriseket tartalmazzák.

I 	J 	L 	O 	S 	T 	Z 
<pre>//A tetrisek lehetséges verziói static bool I[16] = { 0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0 }; //I static bool T[9] = { 1,1,1,0,1,0,0,0,0 }; //T static bool L[9] = { 1,1,1,1,0,0,0,0,0 }; //L static bool J[9] = { 1,1,1,0,0,1,0,0,0 }; //J static bool S[9] = { 0,1,1,1,1,0,0,0,0 }; //S static bool Z[9] = { 1,1,0,0,1,1,0,0,0 }; //Z static bool O[4] = { 1,1,1,1 }; //O</pre>						

• Függvények:

```
//Kiválasztott tetrist adja vissza (belső globális változók)
bool* TetraVal(int melyik);
//Lefoglalja a tetris mátrixát
bool* HandFoglal(Hand* hp);
//Kézben lévő tetris inicializáció
void HandInit(Hand* hp, int const *oszlop, int const melyik);
//Elforgatja a kézben lévő tetrist és visszatér annak mátrixával
bool* Forgat_jobbra(Hand const *hp);
```

■ ranglista

Ranglistával kapcsolatos függvényeket tartalmazza. A szükséges adatszerkezetek a **hand** fájlban vannak, ugyanis a pálya struktúrájába kerül letárolásra a ranglista.

• Függvények:

```
//Ranglista két elemének cserélése
static void RangCsere(Ranglista *a, Ranglista *b);
//Játékos nevéhez karakter hozzáfűzése
void Nevhezir(Palya *vp, int c);
//Játékos nevéből karakter törlése
void NevbolTorol(Palya *vp);
//Ranglista rendezése
void RanglistaRendez(Palya *vp);
//Ranglista mentése
void Ranglistament(Palya *vp);
//Ranglistához hozzáadás
void Ranglistahozaad(Palya *vp);
//Ranglista feltöltése fájlból
void Ranglistabeolvas(Palya *vp);
Benne: typedef enum FInp {nev, score, perc, masodperc, vegeell} FInp;
```


■ megjelenites

Az ablakra rajzoláshoz szükséges függvényeket tartalmazza.

• RGBA színkód adatszerkezete:

```
typedef struct {    //RGBA
    float r;        //red
    float g;        //green
    float b;        //blue
    float a;        //alpha
} RGBA;
```

• Függvények:

```
//OpenGL Grafikához szükséges inicializációk
void initGL();
//Játék belső időzítője
void Idozito(int idokoz);
// Handler - windows re-size :
    Mikor az ablakot átméretezik, az arányok nem romlanak el.
void Ujrameretesz(GLsizei width, GLsizei height);
    // Ez a függvény, az alábbi oldal példája alapján készült:
    // http://www3.ntu.edu.sg/home/ehchua/programming/opengl/cg\_introduction.html
// Kis matek a megfelelő kirajzolásokhoz
void KirajzInit(Palya *vp);
//Szöveg kirajzolása x,y koordinátára adott mérettel
static void Szovegrajzol(char* szoveg, GLfloat x, GLfloat y, float meret);
//Négyzet rajzolása Nsize mérettel x,y koordinátára s színnel
//Egyik mód a normál tetris rajzolása, másik mikor halványan kell
static void RajzolNégyzet(GLfloat *Nsize, GLfloat x, GLfloat y, RGBA s, bool mode);
//x, y koordináta szerint egy Nsize méretű négyzetrácsos hálóbba rajzolás
static void RajzolHaloba(Palya *vp, float x, float y, int sor, int oszlop,
    RGBA s, bool mode);
//Visszatér adott szám alapján a megfelelő színkóddal
static RGBA SzinKonverter(int szin);

//Pályamátrix kirajzolása
static void RajzolMatrix(Palya *vp);
//Adott kézben lévő tetris kirajzolása
static void RajzolTetris(Palya *vp, Hand *hp, int x, bool mode);
//Adott bool tömbben lévő tetris kirajzolása
static void RajzolVTetris(Palya static *vp, bool* t, int size, float x, float y,
    int szin)

//Idő kirajzolása
static void KirajzolIdo(Ido *t, float x, float y, float size);
//Pontszám kirajzolása
static void KirajzolPont(int *pontos, float x, float y, float size);
//Szint kirajzolása
static void KirajzolSzint(int *szints, float x, float y, float size);
//Eltüntetett sorok számának kirajzolása
static void KirajzolEltSorSzam(Palya *vp, float x, float y, float size);
//Tarsolyban lévő tetris kirajzolása
static void RajzolTarsoly(Palya *vp);
//Következő tetrisek kirajzolása
static void RajzolKovi(Palya *vp);
//Ranglista megjelenítés
static void RajzolRangLista(Palya *vp);
```

```
//Játék közbeni Kirajzolás
void JatekRajzol(Palya *vp, Hand *hp);
//Játék vége kirajzolás
void GameOverRajzol(Palya *vp, Hand *hp);
```

Függvények:

MAIN

```
//Játékot vezérlő változók módosítása
void Jatek();
```

Feladata: A belső játékvezérlő változók módosítása a megfelelő feltételek alapján

- Gravitációs intervallum, Időkijelző átváltások, Szint növelése

Paraméterek: Nincsenek

Visszatérés: Nincs

```
//Rendszeresen meghívódó függvény újrarajzolásra
void display();
```

Feladata: A játék megfelelő állása alapján való kirajzolási függvény kiválasztása

- Játékkirajzolás vagy Játék vége kirajzolás, Változók növelése (Mp, Grav)

Paraméterek: Nincsenek

Visszatérés: Nincs

```
//Mallocolt területek felszabadítása
static void Felszabadit();
```

Feladata: Területek felszabadítása

- Ranglistában lévő nevek, pályamátrix, hand mátrix

Paraméterek: Nincsenek

Visszatérés: Nincs

```
//Egyszerűbb gombok
void keyboard(unsigned char key, int x, int y);
```

Feladata: A billentyű gombnyomás alapján a megfelelő műveletek elvégzése

- **(27) ESC** – kilépés **(32) Szóköz** – Tetris lerakása **C** – Tarsolyba rakás
- **Játék végén: Betűk** – névhez írás **(8) Törlés** **(13) Enter** - Mentés

Paraméterek:

- **Key:** Leütött gomb száma
- **x, y:** Egér helyzete, mikor érkezett gomb nyomás

Visszatérés: Nincs

```
//Speciális gombok
void specialKeys(unsigned char key, int x, int y);
```

Feladata: A billentyű speciális gombnyomás alapján a megfelelő műveletek elvégzése

- **Fel nyíl:** Forgatás **Le nyíl:** Mozgatás lefele **Oldal nyilak:** Mozgatás oldalra

Paraméterek:

- **Key:** Leütött gomb száma
- **x, y:** Egér helyzete, mikor érkezett gomb nyomás

Visszatérés: Nincs

```
//Ujjatek inicializáció
```

```
void Ujjatek(int argc, char** argv);
```

Feladata: Kezdő inicializációk, mint pálya beállítása, vagy a megfelelő eventekhez a megfelelő függvények hozzá társítása.

Paraméterek: argc és **argv, melyet az int main-ből kap és a GLUT-nak van szüksége rá

Visszatérés: Nincs

MAP

```
//Memoria terület lefoglalasa a pályának
```

```
PalyaMatrix* MatrixFoglal(Palya const *vp);
```

Feladata: Lefoglal a pálya mérete alapján egy sor*oszlop Pályamátrix méretű területet

Paraméterek: Pálya struktúra

Visszatérés: Lefoglalt területre mutató pointer

```
//A pálya inicializálása
```

```
void MatrixInit(Palya *vp, int const sor, int const oszlop);
```

Feladata: A pálya struktúra kezdeti értékeit adja meg sor és oszlop szám alapján.

- Nulázza amit kell, beállítja a soron következő tetriseket, szintet 1-re állítja, lefoglalja a sumot és nevet, lenullázza a mátrixot.

Paraméterek: Pálya struktúra, melyet változót cím szerint kap meg. Sor és oszlop szám.

Visszatérés: Nincs

```
//Tetris átmásolása a pályára
```

```
void MatrixbaMasol(Palya *vp, Hand *hp);
```

Feladata: A pálya struktúra mátrixára bemásolja a Hand struktúra mátrixát, emellett ellenőrzi ha megtelik egy sor és arra meghívja az eltüntető függvényt.

Paraméterek: Pálya struktúra és Hand struktúra

Visszatérés: Nincs

```
//Ütközés vizsgálat x és y eltolással bp tömbre
```

```
bool Utkozes(Palya const *vp, Hand const *hp, bool const *bp, int x, int y);
```

Feladata: Ha a pályára adott méretű bool tömb adott x és y kordinátára másolunk ütközés történne-e

Paraméterek: Pálya, Hand (a méret miatt), a bool mátrixunk és az x és y kordináta ahol van.

Visszatérés: Igaz, ha van ütközés, Hamis ha nincs.

Figyelni kell, hogy a függvény meghívása előtti bool mátrixot ne felejtjük el felszabadítani ha kell.

```
//Alsó tetris távolsága a jelenlegitől
```

```
int AltetrisKord(Palya const *vp, Hand const *hp);
```

Feladata: Mi a maximum, amíg lejjebb lehet tolni a tetrist?

Paraméterek: A pálya és a Hand struktúra.

Visszatérés: Eltolás egész értékben.

```
//Következő tetrisme állítás
```

```
void KovTetris(Palya *vp, Hand *hp, bool *vege);
```

Feladata: A kézben lévő tetriskicseréli a sorban következőre és egy újat generál a sorba.

Paraméterek: A pálya mátrix, hogy tudja mik vannak a sorban, a Hand, hogy tudja hova másolni és a vége bool változó, ami azért felelős, hogy vizsgálja ha bemásoláskor már ütközés lenne, tehát nem tudja bemásolni, azaz vége a játéknak.

Visszatérés: Nincs.

```
//Adott sor eltüntetése, majd fölötte lévőket lejjebb húzása
```

```
void Eltuntet_sor(Palya *vp, int sor);
```

Feladata: A kiválasztott sor eltüntetése a pályáról

Paraméterek: A pálya és az adott sor egész értékben.

Visszatérés: Nincs.

HAND

```
//Kiválasztott tetriskiadja vissza (belső globális változók)
```

```
bool* TetraVal(int melyik);
```

Feladata: Megfelelő sorszámmű tetris-szel való visszatérés.

Paraméterek: A sorszám, amely egész.

Visszatérés: A tömbre mutató pointer.

```
//Lefoglalja a tetris mátrixát
```

```
bool* HandFoglal(Hand* hp);
```

Feladata: A kézben lévő tetris méretéhez megfelelő mátrix lefoglalása.

Paraméterek: A Hand struktúra.

Visszatérés: A lefoglalt területre mutató bool pointer.

```
//Kézben lévő tetris inicializáció
```

```
void HandInit(Hand* hp, int const *oszlop, int const melyik);
```

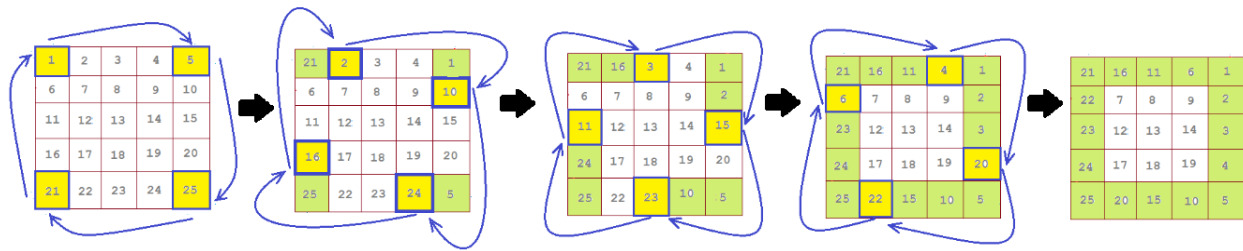
Feladata: A Hand struktúra kezdeti értékeinek beállítása. A pályához képesti pozíciója az oszlop szám alapján és a tetris kiválasztása a melyik egész alapján.

Paraméterek: Hand struktúra, az oszlopszámmra mutató pointer és a melyik egész.

Visszatérés: Nincs.

```
//Elforgatja a kézben lévő tetríst és visszatér annak mátrixával
bool* Forgat_jobbra(Hand const *hp);
```

Feladata:



Mikor 90°-al forgatunk egy $n \times n$ -es mátrixot, valójában a fenti ábra szerint pakolunk át elemeket, melyek indexét ha felírjuk, akkor megkapjuk a fenti megoldást, vagyis:

Segéd_Tömb [j] [Méret - 1 - i] = Tömb [i] [j];

Balra forgatáskor, pont ugyanez történne, csak a két a két kordináta helyet cserélne.

Segéd_Tömb [j] [i] = Tömb [i] [Méret - 1 - j];

Paraméterek: A Hand struktúra.

Visszatérés: Az elforgatott bool mátrix.

RANGLISTA

```
//Ranglista két elemének cserélése
static void RangCsere(Ranglista *a, Ranglista *b);
```

Feladata: A ranglista struktúra tömb két elemét cseréli fel. Rendezéshez van rá szükség

Paraméterek: Ranglista egyik és másik elemére mutató pointer.

Visszatérés: Nincs.

```
//Játékos nevéhez karakter hozzáfűzése
void Nevhezir(Palya *vp, int c);
```

Feladata: A játékos nevéhez egy karaktert hozzáfűz.

Paraméterek: Pálya struktúra, melyben a név van és a karakter integer értéke.

Visszatérés: Nincs.

```
//Játékos nevéből karakter törlése
void NevbolTorol(Palya *vp);
```

Feladata: A játékos nevéből törli az utolsó karaktert.

Paraméterek: A pálya struktúra, melyben a név szerepel.

Visszatérés: Nincs.

```
//Ranglista rendezése  
void RanglistaRendez(Palya *vp);
```

Feladata: Ranglista rendezése egyszerű szélső érték kereséses rendezéssel. Rendezés után automatikusan mentésre kerül a fájlba a Ranglistament függvény által.

Paraméterek: Pálya struktúra.

Visszatérés: Nincs.

```
//Ranglista mentése  
void Ranglistament(Palya *vp);
```

Feladata: A ranglista fájlba írása.

Paraméterek: Pálya struktúra

Visszatérés: Nincs

```
//Ranglistához hozzáadás  
void Ranglistahozaad(Palya *vp);
```

Feladata: Ranglista végére egy új rekord felvétele.

Paraméterek: Pálya struktúra.

Visszatérés: Nincs.

```
//Ranglista feltöltése fájlból  
void Ranglistabeolvas(Palya *vp);
```

Feladata: Ranglista beolvasása fájlból. Állapotgépes módszerrel van megvalósítva, karakterenként beolvasva a fájlból. Lehetett volna sokkal egyszerűbben is, de akartam állapotgépet is a programba.

```
typedef enum FInp {nev, score, perc, masodperc, vegeell} FInp;
```

Paraméterek: Pálya struktúra.

Visszatérés: Nincs.

A fájlnak vagy üresnek kell lennie, vagy megfelelően formázott adatokkal feltöltöttnek.

MEGJELENITES

```
// http://www3.ntu.edu.sg/home/ehchua/programming/opengl/cg\_introduction.html  
A GLUT és OpenGL megértéséhez a fenti oldal segített.  
Az ujráméretező függvény az oldalon lévő példája alapján készült.
```

```
//OpenGL Grafikához szükséges inicializációk  
void initGL();
```

Feladata: OpenGL-hez szükséges alap dolgok beállítása, ez esetben a törlő szín.

Paraméterek: Nincs.

Visszatérés: Nincs.

```
//Játék belső időzítője  
void Idozito(int idokoz);
```

Feladata: Az a függvény, melyet a glutTimerFunc hív meg rendszeresen. A függvény meghíváskor újrarajzolandónak jelöli az ablak tartalmát.

Paraméterek: idokoz, melyet megkap a TimerFunc-on keresztül. Lehetne még egy paraméter is, de arra jelen esetben nincs szükség.

Visszatérés: Nincs

```
// Handler - windows re-size :  
Mikor az ablakot átméretezik, az arányok nem romlanak el.  
void Ujramertez(GLsizei width, GLsizei height);
```

Feladata: A kirajzolt dolgok újraméretezése az ablak mérethez igazítva.

Paraméterek: Két egész típusú változó, melyek az ablak szélessége és magassága.

Visszatérés: Nincs.

```
// Kis matek a megfelelő kirajzolásokhoz  
void KirajzInit(Palya *vp);
```

Feladata: A pálya méretekkel kapcsolatos változók kiszámolása és lementése.

Paraméterek: Pálya struktúra.

Visszatérés: Nincs.

```
//Szöveg kirajzolása x,y kordinátára adott mérettel  
static void Szovegrajzol(char* szoveg, GLfloat x, GLfloat y, float meret);
```

Feladata: x és y kordinátára egy szöveg kirajzolása adott mérettel

Paraméterek: Egy szövegre mutató pointer, x és y egész kordináta és egy float méret

Visszatérés: Nincs.

```
//Négyzet rajzolása Nsize mérettel x,y kordinátára s színnel  
//Egyik mód a normál tetris rajzolása, másik mikor halványan kell  
static void RajzolNegyzet(GLfloat *Nsize, GLfloat x, GLfloat y, RGBA s, bool mode);
```

Feladata: Négyzet kirajzolása x és y kordinátára adott színnel. Két módja közt lehet váltogatni, a különbség az árnyalatban van, ugyanis a fenti és lenti tetris kirajzolása is ennek segítségével történik.

Paraméterek: A négyzetek mérete, x és y kordináta, a szín RGBA struktúrában és végül, hogy milyen módban működjön a függvény.

Visszatérés: Nincs.

```
//x, y kordináta szerint egy Nsize méretű négyzetrácsos hálóba rajzolás  
static void RajzolHaloba(Palya *vp, float x, float y, int sor, int oszlop,  
                          RGBA s, bool mode);
```

Feladata: Kirajzolja egy négyzetet adott kordinátára sor és oszlop szám mellett a hálóra.

Lényegi része, hogy így nem kell minden kordinátát kiszámolni, ugyanis elég megadni a bal felsőt és onnan kiszámolja x és y kordinátát, majd oda rajzol.

Paraméterek: A pálya mátrixa, az x és y kordináta, a sor és oszlop szám és a mode.

Visszatérés: Nincs.

```
//Visszatér adott szám alapján a megfelelő színkóddal  
static RGBA SzinKonverter(int szin);
```

Feladata: Adott számot konvertál a megfelelő színre.

Paraméterek: Egy egész szám.

Visszatérés: RGBA struktúra típusú változó.

```
//Pályamátrix kirajzolása
```

```
static void RajzolMatrix(Palya *vp);
```

Feladata: Végig fut a pályamátrixon és annak elemeit kirajzolja a RajzolHaloba segítségével.

Paraméterek: Pályamátrix.

Visszatérés: Nincs.

```
//Adott kézben lévő tetris kirajzolása
```

```
static void RajzolTetris(Palya *vp, Hand *hp, int x, bool mode);
```

Feladata: Adott eltolással kirajzol egy tetrist. Ez azért jó, mert így elég ez az egy függvény az al és normál tetris kirajzolásához.

Paraméterek: Pálya és Hand struktúra, az eltolás egész értéként és mód.

Visszatérés: Nincs.

```
//Adott bool tömbben lévő tetris kirajzolása
```

```
static void RajzolVTetris(Palya static *vp, bool* t, int size, float x, float y, int szin)
```

Feladata: Hasonló a RajzolTetrishez, de ez egy adott kordinátára rajzol ki egy bool tömbben lévő tetrist. Ez felelős azért hogy a tarsolyt és a következő tetriseket megjelenítsük.

Paraméterek: A pályamátrix, a bool mátrix benne a tetris-szel, annak mérete, x, y kordinátája és színe.

Visszatérés: Nincs.

Talán kicsit sok a paramétere, későbbi fejlesztés során akár átlehetne rendezni ezeket egy saját struktúrába.

```
//Idő kirajzolása
```

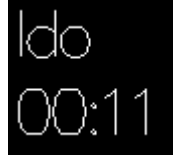
```
static void KirajzolIdo(Ido *t, float x, float y, float size);
```

Feladata: Idő megfelelő formátumban való kirajzolása x és y kordinátára.

Valójában ez az a függvény, ami a pálya méretét is kirajzolja.

Paraméterek: Az idő struktúrája, x és y kordináta és a szöveg mérete.

Visszatérés: Nincs.




```
//Pontszám kirajzolása
```

```
static void KirajzolPont(int *pontos, float x, float y, float size);
```

Feladata: Pontszám kirajzolása adott kordinátára.

Paraméterek: A pontra mutató változó, x, y kordináta és szövegméret.

Visszatérés: Nincs.



```
//Szint kirajzolása
```

```
static void KirajzolSzint(int *szints, float x, float y, float size);
```

Feladata: A szint kirajzolása, ahol jár a játékos.

Paraméterek: A szintre mutató változó, x, y kordináta és szövegméret.

Visszatérés: Nincs.



```
//Eltüntetett sorok számának kirajzolása
```

```
static void KirajzolEltSorSzam(Palya *vp, float x, float y, float size);
```

Feladata: Kirajzolja a játék során eltüntetett sorok számát.

Paraméterek: Pálya struktúra, x, y koordináta és szövegméret.

Visszatérés: Nincs.



```
//Tarsolyban lévő tetris kirajzolása
```

```
static void RajzolTarsoly(Palya *vp);
```

Feladata: A tarsoly kirajzolása.

Paraméterek: Pálya struktúra melyben benne van, hogy mi van a tarsolyban.

Visszatérés: Nincs.

```
//Következő tetrisek kirajzolása
```

```
static void RajzolKovi(Palya *vp);
```

Feladata: Következő tetrisek kirajzolása.

Paraméterek: Pálya struktúra, melyben benne vannak.

Visszatérés: Nincs.

```
//Ranglista megjelenítés
```

```
static void RajzolRangLista(Palya *vp);
```

Feladata: A ranglista első 5 elemének kirajzolása.

Paraméterek: A pálya struktúra.

Visszatérés: Nincs.

```
//Játék közbeni Kirajzolás
```

```
void JatekRajzol(Palya *vp, Hand *hp);
```

Feladata: A fenti függvényekből összeállítja a kirajzolást a megfelelő sablon szerint.

Paraméterek: A pálya és a Hand struktúra.

Visszatérés: Nincs.

```
//Játék vége kirajzolás
```

```
void GameOverRajzol(Palya *vp, Hand *hp);
```

Feladata: Ez esetben a játék végét rajzolja ki, tehát azt, mikor már csak nevet kér be a játék

Paraméterek: Pálya és Hand struktúra.

Visszatérés: Nincs.