Adatmodellezés feladatok megoldása 2018
Látványos kerítés magasság- és területpontjai & Pánikszoba
Mátyás György MA informatikatanár 4. évfolyamos lev. hallgató Nkód: g0tiz4

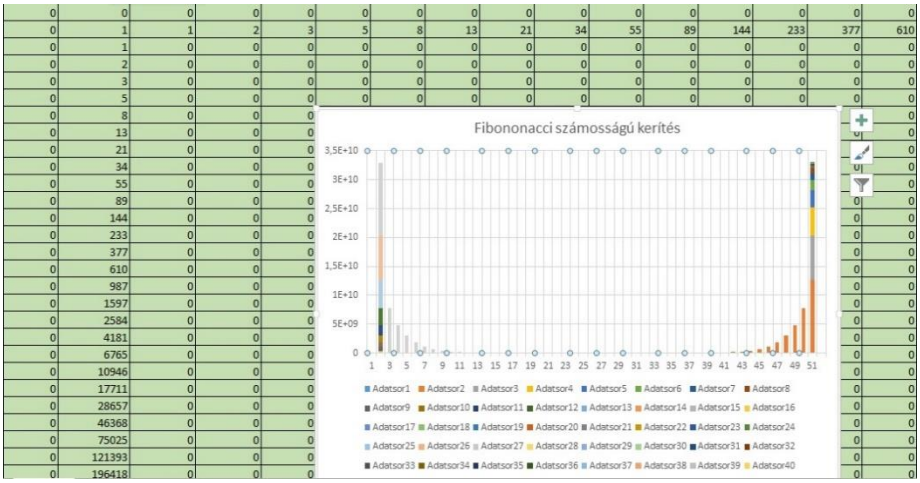
Tartalomjegyzék

A látványos kerítés magasság- és terület pontok

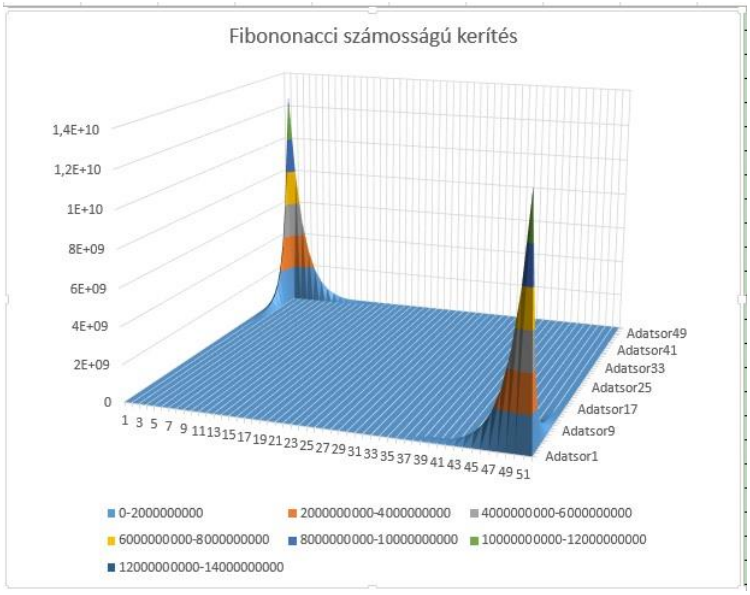
Az átlagszámítás módszer

A „Fibonacci” számok adják a kerítés magasságát és ezek határozzák meg a kerítésen belüli területpontok magasságát is, hisz azok a szomszédos pontok számtani közepe.
Az 5000 –szer újraszámolt terület magasságok mátrix összeg változása 1 egység alá esett így számomra elérte az „elég pontos” állapotot.
Az első mátrix összegváltozás nagyobb volt, mint 30 milliárd egység.
Didaktikai és szemléltetés okán a kerítést kívülről 0-k határolják.

Gyűjtőmunka az „érdekes” kerítés tervezési szakaszából

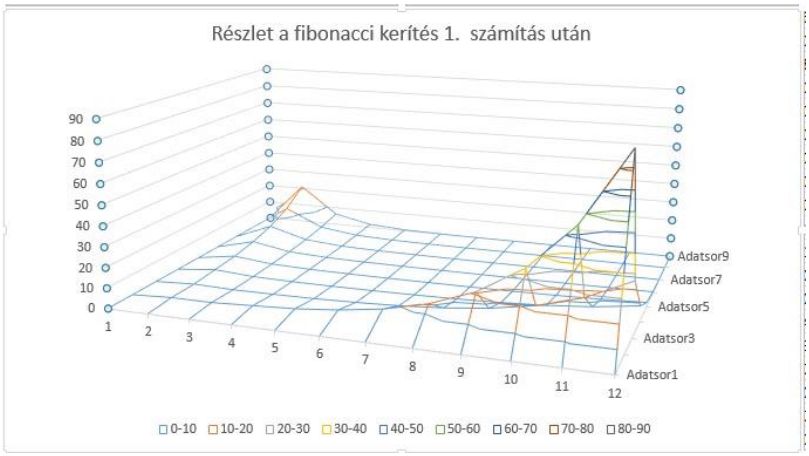


A kiválasztott kerítés minta magasság pontjainak „default” állapotáról.

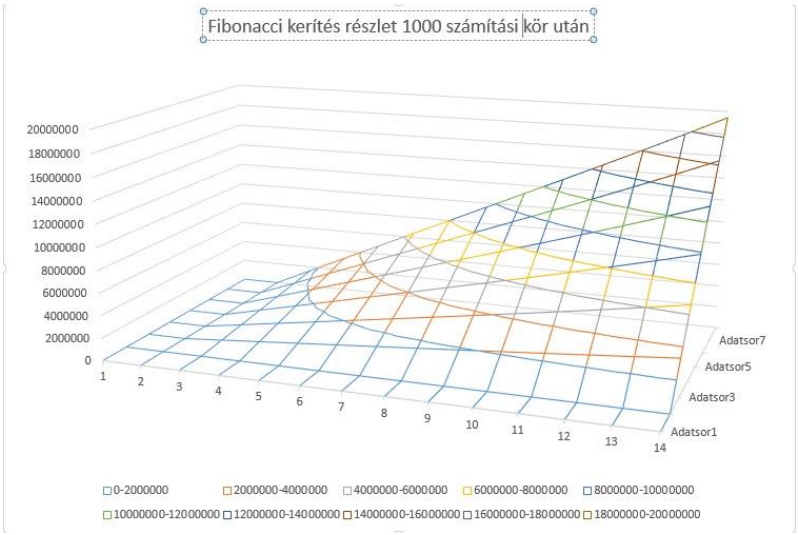


Adatmodellezés feladatok megoldása 2018
Látványos kerítés magasság- és területpontjai & Pánikszoba
Mátyás György MA informatikatanár 4. évfolyamos lev. hallgató Nkód: g0tiz4

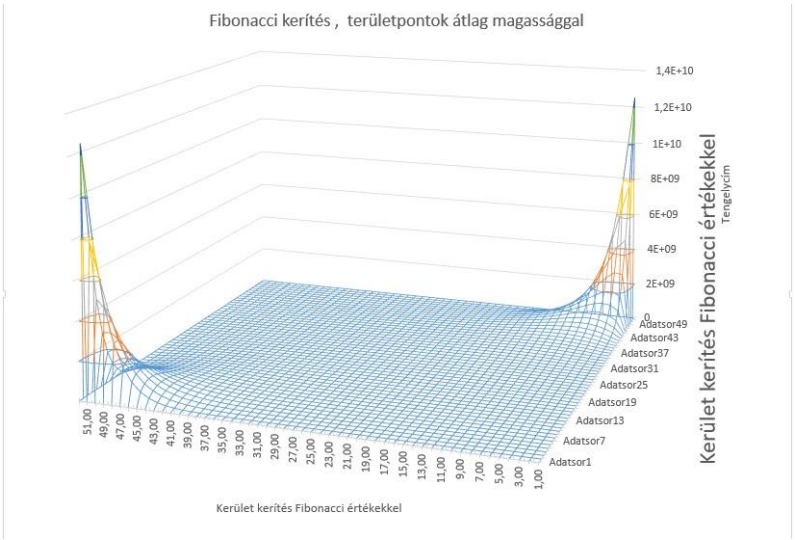
A kerület és terület pontok egy számítási kör után



A kerület és terület pontok ezer számítási kör után



A terület magasságpontok „elég pontos” kiszámítása utáni állapotról.

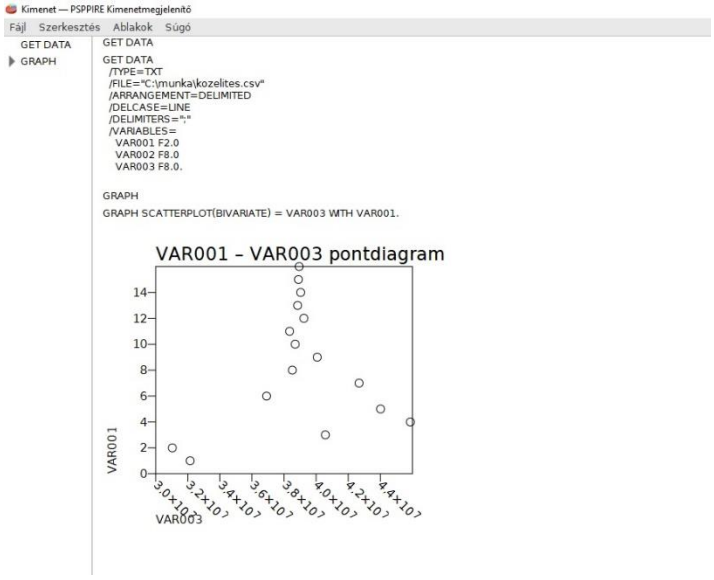


A nullához közelítő mátrix-összeg-különbség érték.

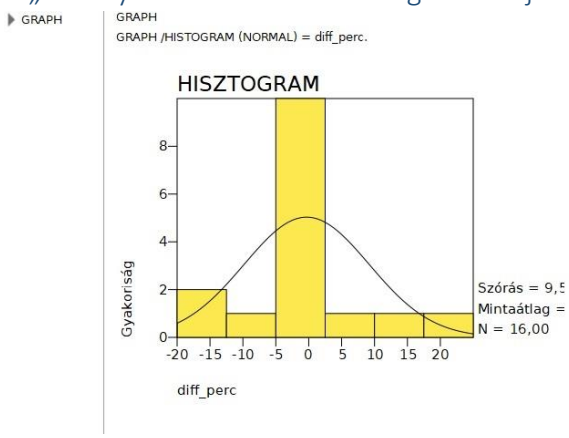
Egy területpont újraszámolása egy másik un. ” kiskutya” módszerrel.

A kiskutya több mint 112 milliószor érintette meg a kerítést az önkényesen kiválasztott xy(13,13) terület pont értékének a kiszámítása okán. Így 99,6 % fölötti egyezést mutatott a pontosnak mondható első számítással. Slussz poén, hogy a 16 db (eltérő ciklus-számú) futási mintaelemem meggyőző Gaus görbét mutat az első számítási érték irányába.

A kiskutya mintaelemeinek értéke közelít az „elég pontos” értékhez



A „kiskutya” mintaelemei Gaus görbét rajzol az „elég pontos” érték irányába.



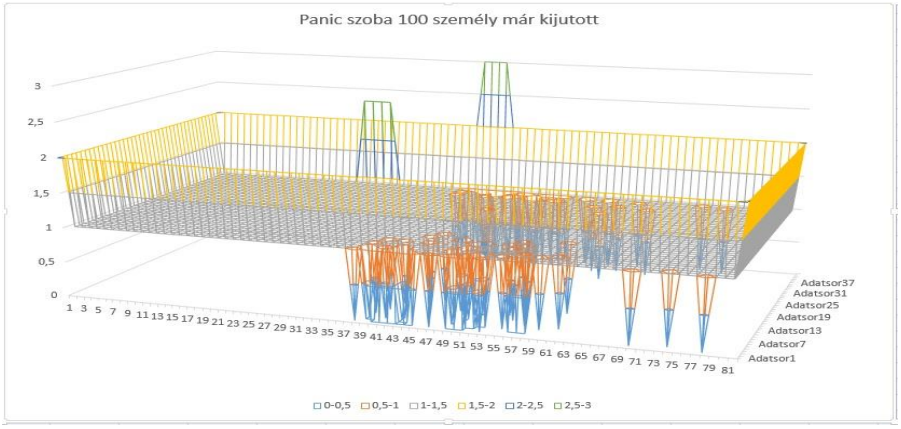
A pánik szoba feladat

```
// A szoba mátrix xy (tömb) deklarálása
// xy adott pont lehetséges értékei:
// 0=üres (nincs ott se ember se más dolog)
// 1= ember van ott,
// 2= fal van ott,
// 3= ajtó van ott (kijárat)
```

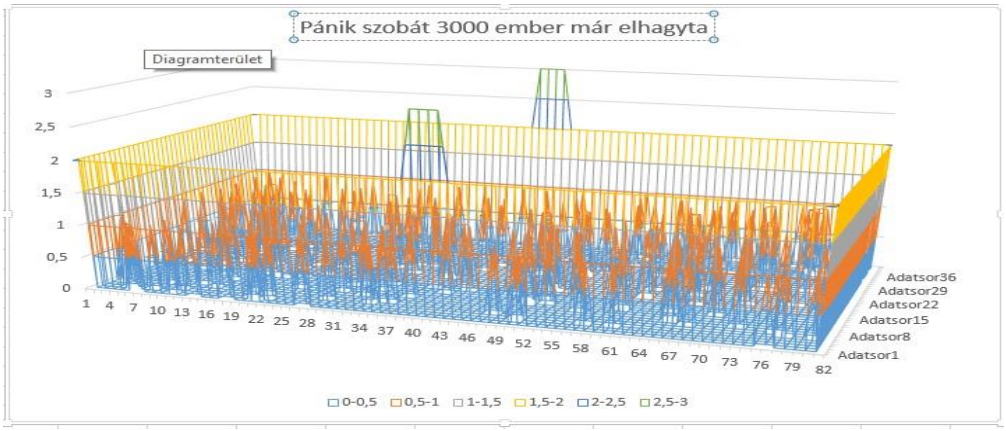
Pánik szoba hosszabbik oldalajtókkal

A pánik szoba didaktikai kezdő ábrája

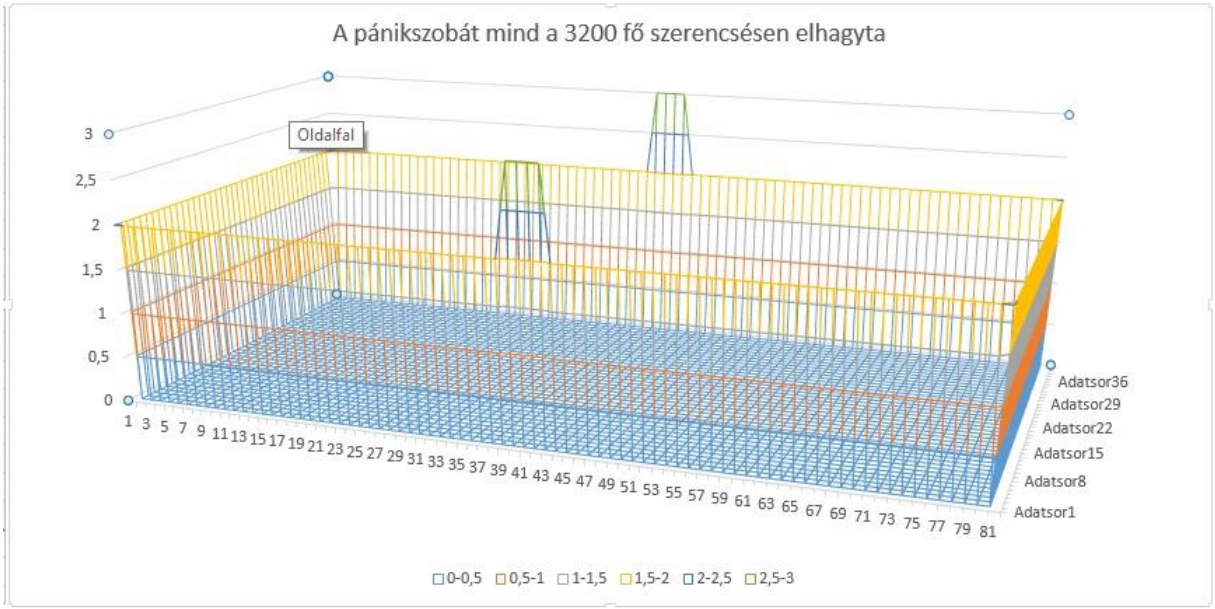
A pánik szobából már néhány ember kijutott



A pánik szobából sokan kijutottak

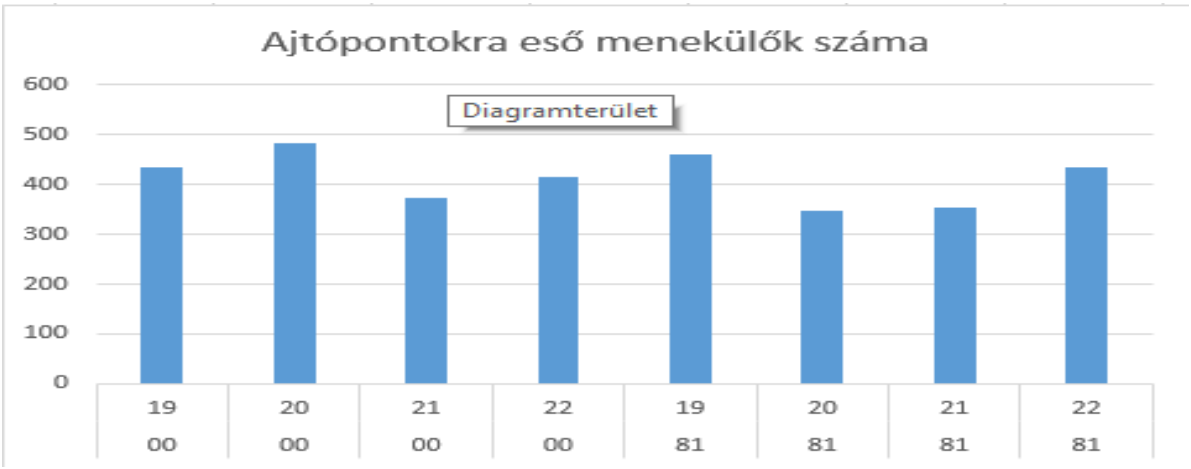


A pánik szobából mindenki kijutott

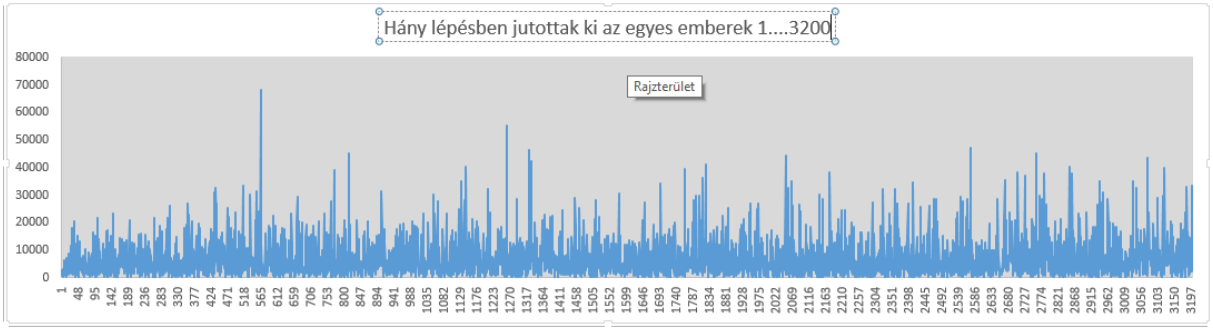


A pánik szobából történő kijutás lépés számainak alakulása egy 11 elemű minta alapján

A pánik szoba ajtópontjaira eső eloszlása a hosszabbik oldalajtó 11. mintasorozata alapján



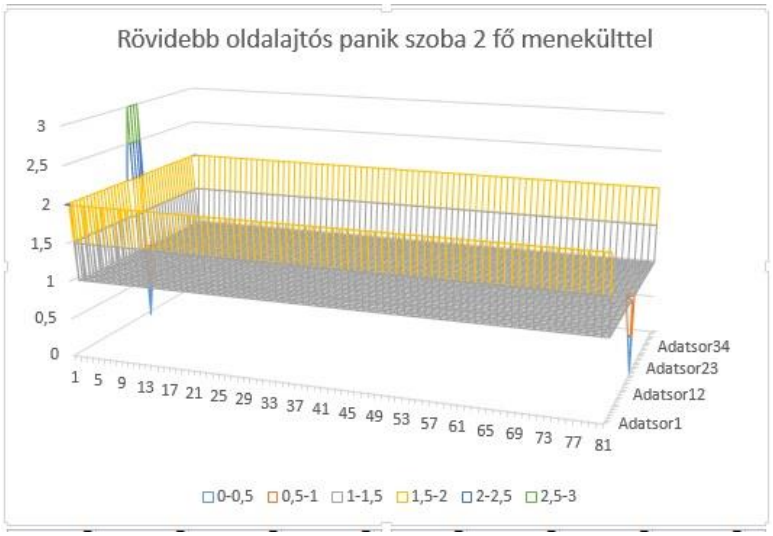
Hány lépésben jutottak ki az emberek (1…3200) ?



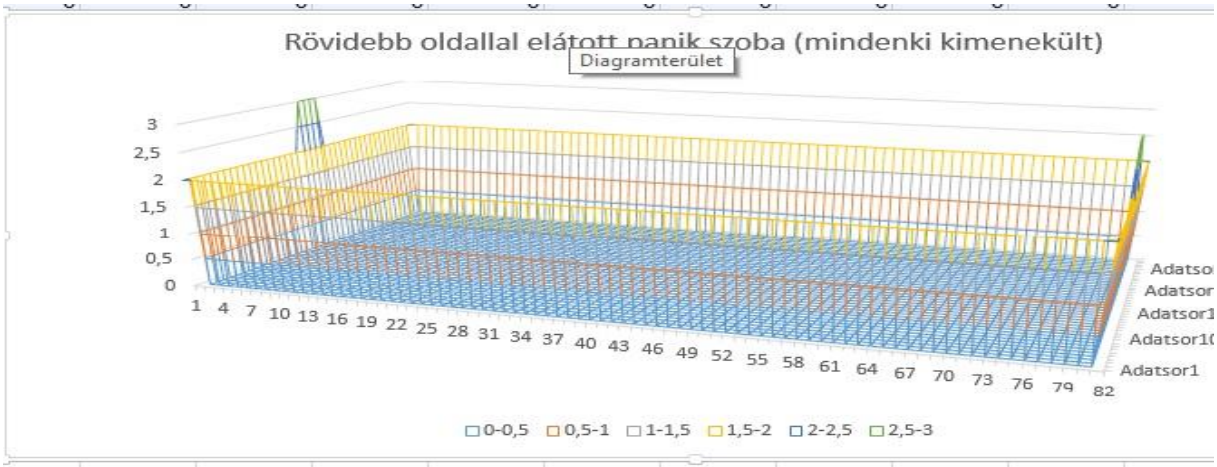
Pánik szoba rövidebbik oldalajtókkal

A pánik szobát már 2 fő elhagyta

A jobb oldali szobafalat és ajtót didaktikai okokból nem jelenítettem meg



A pánik szobát már mindenki elhagyta



Pánikszoba különböző oldalajtó variánsainak összehasonlítása

A feladatnak megfelelően először 4 egységnyi ajtót programoztam szimmetrikusan középre a 80 egységnyi oldalakon. Ezzel az oldalbeállítással 11-szer futtattam le a programot. Jellemzően 21 millió lépésszám (ciklus) fölötti eredménnyel. Az ábrán a „ hosszabb_4 oszlop” a hosszabbik oldalon elhelyezett 4 egység ajtót jelenti.

A következő beállításnál a rövidebbik oldalra kerültek a 4 egységnyi ajtók az ábrán „ rovidebb_4” oszlop jelzi ezt. Itt is 11-szer futtattam a programot és meglepetésre kisebb ciklusszámokat kaptam.

A rosszabb (hosszabb) oldal ajtóvariánst olyan változtatásnak kellett kitenni, hogy érje el a jobb eredményt felmutató (rövidebb oldal) elhelyezés eredményét. Az ajtóméret 1 egységgel történő növelését választottam. Ezt jelzi az ábrán a „hosszabb_5 oszlop” elnevezés.
Itt is 11-szer futtattam le a programot. Egy menekülés teljes ciklusa a „környezet pont”- ban meghatározott HW és SW összetevőkkel kb 40 perc alatt futott le.

A 3 féle oldalajtók összehasonlítása

MEANS

MEANS TABLES = hosszabb_4 rovidebb_4 hosszabb_5.

Esetfeldolgozási összegzés

	Esetek					
	Tartalmazza		Kihagyott		Összesen	
	N	Százalék	N	Százalék	N	Százalék
hosszabb_4:	11	100%	0	0%	11	100%
rovidebb_4:	11	100%	0	0%	11	100%
hosszabb_5:	11	100%	0	0%	11	100%

Jelentés

	Mintaátlag	N	Szórás
hosszabb_4	22828602,64	11,00	337275,95
rovidebb_4	22753726,18	11,00	278746,47
hosszabb_5	20582266,45	11,00	359480,69

Pánik szoba egyéni ötletmegvalósítás

Tanár úr érdekesnek találta a felvetésemet a véletlen-személyek megtalálásának egyéni módszeréről. Ötletem szerint úgy is meg lehet találni valakit egy mátrixba, hogy keresek egy sor és oszlop véletlen számokkal meghatározott pontot, ami:

a) vagy egy személyre utal
b) vagy az adott (random) ponttól indulva, balról-jobbra és felülről-lefelé haladva találok egy személy (1-est)
c) vagy az adott ponttól indulva, jobbról-balra és lentről-felfelé haladva találok egy személy (1-est)

Szerintem ezzel a 3 lehetőséggel teljesen be lehet járni a szoba-tömb összes elemét és akár teljesítheti is a „véletlen” fogalommal támasztott kritériumokat.

A „A pánik szobából már néhány ember kijutott” ábra ezt a megvalósított koncepciót tükrözi. Itt az látható, hogy szinte kizárólag a b) ponttal jelzett lehetőség uralja az ábrát. Ezért jobb oldali- túlsúlyos az ábra és kizárólag a jobb oldalán vannak a kijutott emberek. Valószínű,

hogy idővel kiegyenlítődött volna ez a jobbra csúszott arány, amit nem vártam meg, hanem átalakítottam a személy megtalálásának módszerét a forrásban leírt egyszerűbb módra. Ezért a többi ábra már nem tartalmaz ilyen eltolódást. A forráskódban véletlenszerűen keresek egy mátrixpontot. Ha a pont értéke=1 (ember), akkor ezzel a ponttal haladok tovább. Egyéb esetben addig ismétlem a keresést, amíg a random- pont értéke = 1 lesz.

Azt gondolom, hogy ennek a két módszernek az összehasonlítása és elemzése egy másik, egy új projekt feladata lehetne.

Hardware és software környezet

- Hardware:
  - 4 magos CPU (2,4 Ghz)
  - 8 GB RAM
- Op.Sys:,
  - Windows 10
- Fejlesztői környezet:
  - Visual Studio 2017 student verzió
  - Programnyelv: C#
- Adatmegjelenítő software:
  - Excel 2013
  - PSPP (Az IBM SPSS program free alternatívája)

A fejlesztő környezet biztosítja, hogy Linux op.sys alatt is fussanak a programjaim.

Forráskódok (c#)

A kerítés magasság és területpontjainak változásai

```
using System;
using System.IO;
namespace kerulet_terulet
{
	class Program
	{
		static void Main(string[] args)
		{
			int x,y = 0;
```

```
                    double [,]xy          =new double[52,52]; // Teület mátrix (tömb)
deklarálása
             double [] MatrixOsszeg=new double[5000]  ; // A mátrix összegének
tárolására
         // az x,y koordinátapontok 0 értékkel történő feltöltése
            for ( x = 0; x<52; x++)
                    {
                for (y = 0; y< 52; y++)
                            {
                    xy[x,y] = 0; //0-val feltölteni a magasságot
                    //Console.WriteLine("Szélessége:{0},hosszúsága:{1}, értéke:{2}",
x,y,0);
                }
                    }
            Console.WriteLine("Kész a 0-kal való feltöltés <nyom le egy
billentyűt!>");
            Console.ReadKey();
            //Kerítés szakaszok magasság pontjainak megadása Fibonacci számértékel
            ////1.kerítés szakasz y= 0 [1,1];[50,1]
            xy[1, 1] = 1; // (x - 2) - nél 0, (x - 1) - nél pedig 1 a magasság értéke
a default érték
            xy[0, 1] = 0; // (x - 2) - nél 0, (x - 1) - nél pedig 1 a magasság értéke
a default érté
            for (x = 2; x < 51; x++)
            {
                xy[x, 1] = xy[x - 2, 1] + xy[x - 1, 1]; //Fibonacci számérték= az
előző két szám összege
                Console.WriteLine("Az x=0 vonalon a kerítés y pontja:{0} és
értéke:{1}", y, xy[x, 1]);
            }
            //2.kerítés szakasz x = 51 [51, 1];[51,51]
            xy[50, 51] = 0; // (y - 2) - nél 0, (y - 1) - nél pedig 1 a magasság
értéke a default érték
            xy[50, 50] = 1; // (y - 2) - nél 0, (y - 1) - nél pedig 1 a magasság
értéke a default érték
            for (y = 49; y > 1; y--)
            {
                xy[50, y] = xy[50,y +2] + xy[50,y + 1]; //Fibonacci számérték= az
előző két szám összege
                Console.WriteLine("Az x=51 vonalon a kerítés y pontja:{0} és
értéke:{1}", y, xy[50, y]);
            }
            //3.kerítés szakasz y= 50 [x,0];[x,50]
            xy[51, 50] = 0; // (x-2)-nél 0, (x-1)-nél pedig 1 a magasság értéke
            xy[50, 50] = 1; // (x-2)-nél 0, (x-1)-nél pedig 1 a magasság értéke
            for (x =49; x > 0; x--)
            {
                xy[x, 50] = xy[x + 2, 50] + xy[x + 1, 50]; //Fibonacci számérték= az
előző két szám összege
                Console.WriteLine("Az y=50 vonalon a kerítés x pontja:{0} és
értéke:{1}", x, xy[x, 0]);
            }
            //4.kerítés szakasz x= 50 [y,0];[y,51]
            xy[0, 1] = 0; // (y-2)-nél 0, (y-1)-nél pedig 1 a magasság értéke
            xy[1, 1] = 1; // (y-2)-nél 0, (y-1)-nél pedig 1 a magasság értéke
            for (y = 2; y < 51; y++)
            {
                xy[1, y] = xy[1, y - 2] + xy[1, y - 1]; //Fibonacci számérték= az
előző két szám összege
                Console.WriteLine("Az x=10 vonalon a kerítés y pontja:{0} és
értéke:{1}", y, xy[1, y]);
            }
```

```
            Console.WriteLine("Kész a 4 kerület szakasz feltöltése <nyom le egy
billentyűt!>");
            Console.ReadKey();

            //A területpontok számtani középének számítása (kihagyva a 0 és 51
sor/oszlop kerítésadatokat)
            for (int n = 1; n <5000; n++)
            {//számítási kör
                for (x = 2; x < 50; x++)
                {//oszlopok
                    for (y = 2; y < 50; y++)
                    {//sorok
                        xy[x, y] = (xy[x,y+1]+ xy[x,y-1]+ xy[x-1,y]+ xy[x+1,y]) / 4;
//kiszámítja az adott pont számtani közepét
                        MatrixOsszeg[n] = MatrixOsszeg[n] + xy[x,y]; //összegzem a
mátrix átlagait
                    }
                }
            }
            FileStream fs = new FileStream("c:\\Munka\\matrix10241.txt",
FileMode.Create, FileAccess.Write, FileShare.None);
            StreamWriter sw = new StreamWriter(fs);
            Console.WriteLine("Press any key to write the file!");
            Console.ReadKey();
            for (y = 0; y < 52; y++)
            {
                for (x = 0; x < 52; x++)
                {
                    sw.Write(xy[x, y]);//adatkiíratás
                    sw.Write(";");//adatelválasztó
                }
                sw.Write("\n");//sorelválasztó
            }
            sw.Close();
            fs.Close();
            // mátrix összegének kiírás file-be
            FileStream fs2 = new FileStream("c:\\Munka\\matrix_100sum.txt",
FileMode.Create, FileAccess.Write, FileShare.None);
            StreamWriter sw2 = new StreamWriter(fs2);
            for (int n = 0; n < 5000; n++)
            {
                sw2.Write(MatrixOsszeg[n]);//adatkiíratás
                sw2.Write(";");//adatelválasztó
                sw2.Write("\n");//sorelválasztó
            }
            sw2.Close();
            fs2.Close();
            Console.WriteLine("operation success!");
            Console.ReadKey();
        }
    }
}
```

A kerítés magasság és egy területpontjainak számítása „kiskutya” módszerrel

```
using System;
using System.IO;
namespace kerulet_terulet
{
    class Program
```

```
    {

        static void Main(string[] args)
        {
            int x, y = 0;
            double[,] xy = new double[52, 52]; // Teület mátrix (tömb) deklarálása
            double[] MatrixOsszeg = new double[5000]; // A mátrix összegének
tárolására
                                                      // az x,y koordinátapontok 0
értékkel történő feltöltése
            int kdx = 13;  //kiskutya default  x koordinátaértéke
            int kix =  0;  //kiskutya aktuális x koordináta értéke
            int kdy = 13;  //kiskutya default  y koordinátaértéke
            int kiy =  0;  //kiskutya aktuális y koordináta értéke
            int szelrozsa = 0;//y x irányváltoztatás
            double ksum = 0; //átlaghoz gyűjtő
            double kvalue = 0; //a kiválasztott pont értéke
            Random kirany;// = new Random(); //a kiskutya véletlemszerű iránya
            int n=0;//számítás ciklus számláló
            for (x = 0; x < 52; x++)
            {
                for (y = 0; y < 52; y++)
                {
                    xy[x, y] = 0; //0-val feltölteni a magasságot
                    //Console.WriteLine("Szélessége:{0},hosszúsága:{1}, értéke:{2}",
x,y,0);
                }
            }
            Console.WriteLine("Kész a 0-kal való feltöltés <nyom le egy
billentyűt!>");
            Console.ReadKey();
            //Kerítés szakaszok magasság pontjainak megadása Fibonacci számértékel
            ////1.kerítés szakasz y= 0 [1,1];[50,1]
            xy[1, 1] = 1; // (x - 2) - nél 0, (x - 1) - nél pedig 1 a magasság értéke
a default érték
            xy[0, 1] = 0; // (x - 2) - nél 0, (x - 1) - nél pedig 1 a magasság értéke
a default érté
            for (x = 2; x < 51; x++)
            {
                xy[x, 1] = xy[x - 2, 1] + xy[x - 1, 1]; //Fibonacci számérték= az
előző két szám összege
                Console.WriteLine("Az x=0 vonalon a kerítés y pontja:{0} és
értéke:{1}", y, xy[x, 1]);
            }
            //2.kerítés szakasz x = 51 [51, 1];[51,51]
            xy[50, 51] = 0; // (y - 2) - nél 0, (y - 1) - nél pedig 1 a magasság
értéke a default érték
            xy[50, 50] = 1; // (y - 2) - nél 0, (y - 1) - nél pedig 1 a magasság
értéke a default érték
            for (y = 49; y > 1; y--)
            {
                xy[50, y] = xy[50, y + 2] + xy[50, y + 1]; //Fibonacci számérték= az
előző két szám összege
                Console.WriteLine("Az x=51 vonalon a kerítés y pontja:{0} és
értéke:{1}", y, xy[50, y]);
            }
            //3.kerítés szakasz y= 50 [x,0];[x,50]
            xy[51, 50] = 0; // (x-2)-nél 0, (x-1)-nél pedig 1 a magasság értéke
            xy[50, 50] = 1; // (x-2)-nél 0, (x-1)-nél pedig 1 a magasság értéke
            for (x = 49; x > 0; x--)
            {
```

```
                xy[x, 50] = xy[x + 2, 50] + xy[x + 1, 50]; //Fibonacci számérték= az
előző két szám összege
                Console.WriteLine("Az y=50 vonalon a kerítés x pontja:{0} és
értéke:{1}", x, xy[x, 0]);
            }
            //4.kerítés szakasz x= 50 [y,0];[y,51]
            xy[0, 1] = 0; // (y-2)-nél 0, (y-1)-nél pedig 1 a magasság értéke
            xy[1, 1] = 1; // (y-2)-nél 0, (y-1)-nél pedig 1 a magasság értéke
            for (y = 2; y < 51; y++)
            {
                xy[1, y] = xy[1, y - 2] + xy[1, y - 1]; //Fibonacci számérték= az
előző két szám összege
                Console.WriteLine("Az x=10 vonalon a kerítés y pontja:{0} és
értéke:{1}", y, xy[1, y]);
            }
            Console.WriteLine("Kész a 4 kerület szakasz feltöltése <nyom le egy
billentyűt!>");
            Console.ReadKey();
            FileStream fs = new FileStream("c:\\Munka\\kiskutya.txt", FileMode.Create,
FileAccess.Write, FileShare.None);
            StreamWriter sw2 = new StreamWriter(fs);
            Console.WriteLine("Press any key to write the file!");
            Console.ReadKey();
            //Egy db területpont számítása (kiskutya bóklászik módszerrel)
            // Az alábbi pontról indul és addig halad, amíg a kerítésbe nem ütközik 1=
x or y
            kdx = 13; //kiskutya default  x koordinátaértéke
            kix = 0;  //kiskutya aktuális x koordináta értéke
            kdy = 13; //kiskutya default  y koordinátaértéke
            kiy = 0; //kiskutya aktuális y koordináta értéke
            szelrozsa=0;//y x irányváltoztatás
            ksum = 0; //átlaghoz gyűjtő
            kvalue = 0; //a kiválasztott pont értéke
            kirany = new Random(); //a kiskutya véletlemszerű iránya
            for ( n = 1; n <99000000; n++)
            {//számítási kör
                kix = kdx;
                kiy = kdy;
                //Console.WriteLine("//számítási körben vagyok");
                //Console.ReadKey();
                while (kix != 1 && kiy != 1 && kiy != 50 && kix != 50)
                {
                    //kiskutya lépései
                    //Console.WriteLine("While ciklusban vagyok");
                    //Console.WriteLine("X:{0}",kix);
                    //Console.WriteLine("Y:{0}",kiy);
                    //Console.WriteLine("While ciklusban vagyok");
                    //Console.ReadKey();

                    szelrozsa = kirany.Next(1,5);
                        switch (szelrozsa)
                    {
                        case 1:// észak
                            kiy = kiy + 1;
                            break;
                        case 2://nyugat
                            kix = kix + 1;
                            break;
                        case 3://dél
                            kiy = kiy - 1;
                            break;
                        case 4://kelet
```

```
                        kix = kix - 1;
                        break;
                    }
                    //sw2.Write(n);
                    //sw2.Write(";");
                    //sw2.Write(kix);
                    //sw2.Write(";");
                    //sw2.Write(kiy);
                    //sw2.Write(";");
                    //sw2.Write(szelrozsa);
                    //sw2.Write(";");
                    //sw2.Write("\n");//sorelválasztó
                }
                ksum = ksum + xy[kix,kiy];//ha a kerítéshez ér a kiskutya, akkor
felveszi annak az értékét
                //sw2.Write(n);
                //sw2.Write(";");
                //sw2.Write(kix);
                //sw2.Write(";");
                //sw2.Write(kiy);
                //sw2.Write(";");
                //sw2.Write(szelrozsa);
                //sw2.Write(";");
                //sw2.Write(xy[kix,kiy]);
                //sw2.Write(";");
                //sw2.Write(ksum);
                //sw2.Write(";");
                //sw2.Write("\n");//sorelválasztó
                                //Console.WriteLine("While cikluson kívüln vagyok");
                                //Console.WriteLine("X:{0}", kix);
                                //Console.WriteLine("Y:{0}", kiy);
                                //Console.WriteLine("XY érték:{0}",xy[kix,kiy]);
                                //Console.WriteLine("Kiskutya sum:{0}", ksum);

                //Console.ReadKey();

            }
            kvalue = ksum/(n-1);//Íme az érték

            sw2.Write("A kutya default x koordináta értéke:");

            sw2.Write("\n");//sorelválasztó
            sw2.Write("A kutya defaul y koordináta értéke:");
            sw2.Write(kdy);//adatkiíratás
            sw2.Write("\n");//sorelválasztó
            sw2.Write("Az sum értéke:");
            sw2.Write(ksum);//adatkiíratás
            sw2.Write("\n");//sorelválasztó
            sw2.Write("XY pont  Kiskutya  értéke:");
            sw2.Write(kvalue);//adatkiíratás
            sw2.Write("\n");//sorelválasztó
            //sw2.Write("Az XY pont átlagszámítás értéke:");
            //sw2.Write(xy[kix,kiy]);//adatkiíratás
            sw2.Close();
            fs.Close();
            Console.WriteLine("operation success!");
            Console.ReadKey();
        }
    }
}
```

Pánikszoba elhagyása hosszabbik oldalpár ajtókkal

```
using System;
using System.IO;

namespace _20181029
{
    class Program
    {
        static void Main(string[] args)
        {
            int x, y = 0;
            double[,] xy = new double[82, 44]; // A szoba mátrix (tömb) deklarálása
            //xy adott értékei :0=üres (nincs ember), 1=ott van egy ember, 2= fal van
ott, 3=ajtó van ott (kijárat)
            int kfx = 0; //a megtalált (found) személy aktuális x koordináta értéke
            int kfy = 0; //a megtalált (found) személy aktuális y koordináta értéke
            int kix = 0; //a megtalált & irányba lépett  személy aktuális x koordináta
értéke
            int kiy = 0; //a megtalált & irányba lépett  személy aktuális y koordináta
értéke
            int szelrozsa = 0;//y x irányváltoztatás
            double mi_van_itt = 0;// xy 0,1,2,3 értékei lehetnek
            Random xkoord; //a véletlen  kiválasztáshoz szükséges x KOORDináta
            Random ykoord; //a véletlen  kiválasztáshoz szükséges y KOORDináta
            Random kirany;// = new Random(); //az  xy(kiv,kiy) kiválasztott ember
véletlemszerű iránya
            int n = 0;//kiürülés ciklus számláló Hány lépésből ürül ki a szoba?
           // int po = 0;//kilépett person emberek száma
            int pi = 3200;//bent maradt person emberek száma
                          //Egyesekkel feltölteni a szobát :)
            for (y = 0; y < 42; y++)
            {
                for (x = 0; x < 82; x++)
                {
                    xy[x, y] = 1; //Egyesekkel feltölteni a szobát :)
                }
            }
            //Console.WriteLine("Kész az egyes emberekkel való feltöltés <nyom le egy
billentyűt!>");
            //Console.ReadKey();

            //Szoba kerület-szakaszainak megadása számértékkel (2)

            //Északi szobafal = XY [1,0];[50,0]
            for (x = 0; x < 81; x++)
            {
                xy[x, 0] = 2; //Az átjárhatatlan falat 2 jelképezi
                if (x > 38 && x <= 42) //északi ajtó
                {
                    xy[x, 0] = 3;
                }
            }
            //2.Keleti szobafal = XY[51,0];[51,51]
            for (y = 0; y < 42; y++)
            {
                xy[81, y] = 2; //Az átjárhatatlan falat 2 jelképezi
            }
            //3.Déli szobafal = XY[51,0];[51,51]        z y= 50 [x,0];[x,50]
            for (x = 0; x < 81; x++)
            {
                xy[x, 41] = 2; //Az átjárhatatlan falat 2 jelképez
```

```
            if (x > 38 && x <= 42) //déli ajtó 4 egység
            {
                xy[x, 41] = 3;
            }
        }
        //4.Nyugati szobafal x= 50 [y,0];[y,51]
        for (y = 0; y < 41; y++)
        {
            xy[0, y] = 2; //Az átjárhatatlan falat 2 jelképez
        }
        //Console.WriteLine("Kész a 4 fal szakasz feltöltése <nyom le egy
billentyűt!>");
        //Console.ReadKey();

        //Egy személy véletlenszerű kiválasztása (egyes ember megpróbál kijutni az
épületből)
        // A saját pontjáról indul és egyet lép ha tud, kilép ha tud
        xkoord = new Random();  //adott személy kiválasztásához kell
        ykoord = new Random();
        szelrozsa = 0;//y x irányváltoztatás
        kirany = new Random();
        Console.WriteLine("Még nem léptünk be a while ciklus-ba:{0}", n);
        Console.ReadKey();
        FileStream fs = new FileStream("c:\\Munka\\room_of
person_2018_10_30_2.txt", FileMode.Create, FileAccess.Write, FileShare.None);
        StreamWriter sw2 = new StreamWriter(fs);
//********  Kiépült mátrix értékekkel a személyekkel=1 megrakott szobafallal=2
ajtóstól=3
        while (pi != 0)//Ez az az eset,amikor nem üres a szoba
        {
            do
            {
            kix = xkoord.Next(1, 80 + 1);//random
            kiy = ykoord.Next(1, 40 + 1);//random
            //Console.WriteLine("Random gyártás kix kiy xy[kix,kiy]:{0} {1} {2}",
kix,kiy,xy[kix,kiy]);
            //Console.ReadKey();
            }
            while (xy[kix,kiy]!=1);//addig próbálkozik, amíg 1-est (személyt) nem
talál

            szelrozsa = kirany.Next(1, 5);
            kfx = kix;
            kfy = kiy;
            switch (szelrozsa)
            {// Egyszerűen a véletlen iránynak (szélrózsa)  megfelelően poziciót
lép
                case 1:// észak
                    kiy = kiy - 1;
                    break;
                case 2://kelet
                    kix = kix + 1;
                    break;
                case 3://dél
                    kiy = kiy + 1;
                    break;
                case 4://nyugat
                    kix = kix - 1;
                    break;
            }
```

```
                mi_van_itt = xy[kix, kiy];//a tervezett lépés poziciójába milyen érték
van
                n++;// növeljük a szoba-kiürülés irányába tett lépésszámot

                switch (mi_van_itt)//kiértékeljük mi/ki van az adott pozición
                {//
                    case 0:// nincs ott ember akkor a helyére léphetünk,0 törölve az
előző hely nyomát
                        xy[kfx, kfy] = 0; //régi pozició törlése
                        xy[kix, kiy] = 1; //új pozició
                        break;
                    case 1:// ember van ott nem léphetünk a fejére
                        break;
                    case 2:// fal van ott nem ütjük bele a fejünket a falba
                        break;
                    case 3:// ajtó van ott  kisétálunk töröljük az előző helyünket
                        xy[kfx, kfy] = 0; //régi pozició törlése
                       // Console.WriteLine("***************Ajtóhoz
értünk:{0}***********", xy[kix, kiy]);
                      //  Console.ReadKey();
                        break;
                }
                Console.WriteLine("{0} {1} {2} {3} {4} {5} ", mi_van_itt, kix, kiy,
xy[kix, kiy], n, pi);
                pi = 0;
                for (y = 0; y < 41; y++)
                {
                    for (x = 0; x < 82; x++)
                    {
                        if (xy[x, y] == 1)
                        {
                            pi++;
                        }

                    }
                }
            }// üres a szoba esetén lép ki, vagyis ha pi=0

            sw2.Write(n);//4
            sw2.Write(";");
            sw2.Write(";");
            sw2.Write(pi);//6
            sw2.Write("\n");
            pi = 0;
            for (y = 0; y < 42; y++)// filebeírás
            {
                for (x = 0; x < 82; x++)
                {
                    sw2.Write(xy[x, y]); //kiírni az értéket
                    sw2.Write(";"); //kiírni az ;
                }
                sw2.Write("\n"); //kiírni az értéket
            }
            sw2.Close();
            fs.Close();
            Console.WriteLine("************Feldolgozás vége!**************");
            Console.ReadKey();

        }
    }
}
```

Pánikszoba elhagyása rövidebb oldalpár ajtókkal

Az előző programhoz képest csak a szobafalon lévő ajtók megrajzolásában van eltérés. Ezt az eltérést az alábbi programrészlet tartalmazza.

```
  //Szoba kerület-szakaszainak megadása számértékkel (2)

            //Északi szobafal = XY [1,0];[50,0]
            for (x = 0; x < 81; x++)
            {
                xy[x, 0] = 2; //Az átjárhatatlan falat 2 jelképezi

            }
            //2.Keleti szobafal = XY[51,0];[51,51]
            for (y = 0; y < 42; y++)
            {
                xy[81, y] = 2; //Az átjárhatatlan falat 2 jelképezi
                if (y > 18 && y <= 22) //északi ajtó
                {
                    xy[81,y] = 3;
                }
            }
            //3.Déli szobafal = XY[51,0];[51,51]        z y= 50 [x,0];[x,50]
            for (x = 0; x < 81; x++)
            {
                xy[x, 41] = 2; //Az átjárhatatlan falat 2 jelképez

            }
            //4.Nyugati szobafal x= 50 [y,0];[y,51]
            for (y = 0; y < 41; y++)
            {
                xy[0, y] = 2; //Az átjárhatatlan falat 2 jelképez
                if (y > 18 && y <= 22) //déli ajtó 4 egység
                {
                    xy[0,y] = 3;
                }
            }
            //Console.WriteLine("Kész a 4 fal szakasz feltöltése <nyom le egy
billentyűt!>");
            //Console.ReadKey();
```