Amőba játék

A közkedvelt amőba játékot ketten játsszák. A játékmező, egy négyzetrácsokra osztott papírlap, alakja tetszőleges lehet, többnyire téglalap vagy négyzet. A játék általánosításánál szabálytalan alakú, akár konkáv játékmező is elképzelhető, sőt nem foglalható „szigetek" elhelyezésével tehetjük változatosabbá a pályát. Követelmény viszont, hogy a játékmező összefüggő legyen, azaz bármely mezőből bármely mezőbe eljuthassunk. A játékosok egymást váltva lépnek, minden lépésben lefoglalva a játékmező egy négyzetrácsát. Foglalt négyzetrácsot nem lehet újra foglalni. Az a játékos nyer, aki egy előre megállapított mintázatnak megfelelő "térrész" (síkidom) minden mezőjét először tudja lefoglalni. Legegyszerűbb esetben a mintázat öt egymás melletti négyzetrácsot jelent, ahol a négyzetrácsok oldalaikkal, vagy csúcsaikkal érintkeznek. Ekkor a négyzetrácsok szimmetria középpontjait összekötő szakaszoknak mindkét esetben egyenest kell alkotnia.

Programnyelvi megvalósítás

Elkészíthetjük a játék Java nyelvű programnyelvi megvalósítását is, amellyel a népszerű játék számítógépes változatát játszhatjuk. A játékprogram, indítását követően, koordináta értékeket vár. A koordináták a játékosok által éppen lefoglalni kívánt mezőt egy számpárral azonosítják egy derékszögű koordináta rendszerben. Tízszer tízes négyzet alakú játékmezőt feltételezve a bemeneti értékek egy számpárokból álló diszkrét és véges halmaz elemei: $(x, y) \in N \times N$, ahol x és y tíznél kisebb nem negatív egészek (a Java nyelvben foglalt tömbök elemeinek sorszámozása nem 1-el, hanem 0-val kezdődik). Programunknak minden lépés után elemzést kell végeznie, a visszaadott érték egy a játék pillanatnyi állapotára utaló karaktersorozat lesz. A visszaadott karaktersorozat lehetséges értékei:
{*„A játékos következik", „B játékos következik", „A nyert", „B nyert", „Döntetlen"*}.
Hálózati kommunikáció esetén a játék komfortosabbá tétele indokolttá teszi a játék *Állapot* halmazának finomítását:
{*„A játékos következik", „B játékos következik", „A nyert", „B nyert", „Döntetlen", „A indíthatja a játékot", „B indíthatja a játékot", „A szünetet kért", „B szünetet kért", „A kilépett", „B kilépett", „A új játékot kért", „B új játékot kért"*}.
Megszüntethetjük a visszaadott értékekben a felesleges ismétléseket, ha az állapothalmazt két részhalmaz Descartes szorzataként állítjuk elő:
***J**átékos* {*„A", „B", null*};
***Á**llapot* {*„követezik", „nyert", „döntetlen", „indíthatja a játékot", „szünetet kért", „kilépett", „új játékot kért"*}
A *null* elem bevezetését a „döntetlen" karaktersorozat megjelenése indokolja, illetve a kommunikációt kezelő kódban elkerülhetjük a kivételkezelést.
A program formailag lehetséges összes pillanatnyi állapotának számát a program függvény elemzése alapján kaphatjuk meg.

$$\mathbf{N \times N}_{|\, x, y < 10} \rightarrow \quad \mathbf{J \times Á}$$

Figyelembe kell még vennünk, hogy a *null* elem a szabályok értelmében az *Állapot* halmaz egyetlen eleméhez, a „döntetlen"-hez társítható, így:

$$|\, N \times N_{|\, x, y < 10} \,| \; * \; [\, (\, |\, J \,| \; - 1\,) * (\, |\, Á \,| - 1\,) + 1\,] = 100 * (\, 2 * 6 + 1\,) = 1300.$$

A program egyes lefutásait egy a síkra elhelyezet kétdimenziós diszkrét ponthalmaz megfelelő elemeinek összekötése során kapott törött vonal segítségével ábrázolhatjuk. A függőleges tengelyen elhelyezzük a koordináta párok 100 különböző elemét. A vízszintes tengelyen az állapothalmaz elemeit. Minden pont, koordinátái által, a program egy-egy lehetséges pillanatnyi állapotát testesíti meg. A vízszintes tengelyen összevonhatjuk az („A”, „szünetet kért”) - („A”, „indíthatja a játékot”) állapot párt, ugyanis az elsőt mindenképpen a második követi, a program a játék kimenetele szempontjából nem kerül új állapotba. Hasonló egyszerűsítést jelenthet, ha a játékos kilépését vagy egy új játék kérését azonos eseménynek tekintünk, hiszen ez mindkét esetben a játék feladását jelenti a játékos részéről, azaz egy nem végig játszott partit.

Eltekintve a formális kommunikáció eseményterétől, a program lehetséges lefutásainak, azaz a különféleképpen lejátszható játékoknak a száma attól függ, hogy az egyes játékok során egymást követően foglalt mezők koordinátapárjai halmazának mekkora a számossága. Elméletileg a játéktáblát 100! féle sorrendben lehet kitölteni. Ha egy kitöltés során az n-edik lépés nyerő, akkor ezen kitöltés [(100-n)! - 1]-el csökkenti az elvileg lehetséges játékok számát. Ezt a játékot az egyes lépések után ugyan (n-1) esetben félbe lehet hagyni, de ez nem releváns növekedés a játékvariációk számában.

Játékmező

A játékmezőt két különböző dimenzióban is példányosítanunk kell. Egyrészt a számítógép monitorán meg kell jeleníteni a játékmezőt vizuális dimenzióban, hogy lehetősége legyen a játékosoknak az egyes mezőket kattintással lefoglalni, illetve követni saját és játékostársuk foglalásait. Másrészt a logikai dimenzióban négyzetes mátrixként, hogy elvégezhető legyen a játék gépi kiértékelése. Mindkét példányt programkóddal generáljuk le és az operatív tárban helyezzük el.

A játékmező kirajzolását végző kód (részlet):

```
//játékgombok inicializálása:
   Font font = new Font("Arial black",4,8);
   for (int i=0; i<10; i++)
      {
      for (int j=0; j<10; j++)
         {
         btn_mezo[i][j]= new JButton();
         btn_mezo[i][j].setFont(font);
         pan_gombok.add(btn_mezo[i][j]);
         btn_mezo[i][j].addActionListener(esemeny_kezelo);
         btn_mezo[i][j].setActionCommand(Integer.toString(i)+","+Integer.toString(j));
         /*akció feliratot generálunk minden egyes játékmező-gombhoz, amely
           megegyezik a gomb koordinátáival(az akció felirat nem látszik)
           ezt kérdezzük le megnyomásnál*/
         btn_mezo[i][j].setEnabled(false);
         }
      }
   //komponensek elhelyezése
   pan_gombok.setLayout(null);

   Insets insets = pan_gombok.getInsets();
   txf_inf1.setBounds(200 + insets.left, 35 + insets.top, 232, 22);
   txf_inf2.setBounds(200 + insets.left, 69 + insets.top, 232, 22);
   lbl_jatekos1.setBounds(110 + insets.left, 33 + insets.top, 102, 22);
   lbl_jatekos2.setBounds(110 + insets.left, 67 + insets.top, 102, 22);

   Dimension size = btn_ujjatek.getPreferredSize();
```

A játékmezőt szimbolizáló négyzetes mátrix példányosítása:

```
int [ ][ ] szam =new int[10][10]; //10 x 10-es amőba mezői (sor,oszlop)
//tartalmuk: 0 (nem foglalt)
//1 (1-es játékos foglalta le)
//2 (2-es játékos foglalta le)
```

Kiértékelés

A programnyelvi megvalósítás legkritikusabb része, mondhatni időben legszűkebb keresztmetszete a játékmező gépi kiértékelése. Ezt a kiértékelést minden lépés után azonnal el kell végezni, hiszen ha a játékmezőt foglaló játékosnak sikerült lefoglalnia az előre megbeszélt mintázatot, akkor nyert, a játéknak vége, és ezt tudatni kell a játékosokkal! A gépi kiértékelést négy irányban kell elvégezni: vízszintesen(–), függőlegesen(|), átlósan a jobb felső irányból a bal alsó irányba (/), illetve a bal felső irányból a jobb alsó irányba (\ ). Minden lehetséges sort, oszlopot és átlót vizsgálni kell: van-e öt egymást követő mező, amelyet az éppen lépő játékos (példánkban az 1-es játékos) foglalt le?

Az alábbi ábrán a játékmező egyes elemeit két egymás mellett álló, egyjegyű számmal jelöltük meg. Ezek a négyzetes mátrix elemeinek koordinátái, melyek egyben azonosak a vizuális játékmező megfelelő akció feliratával, melyet a kattintás eseményét kezelő *eseménykezelő* lekérdez. Az egyes mátrix elemek tartalma a játék kezdetén alapértelmezésben 0, ami azt jelzi, még nem foglalták le a mezőt, egyébként 1 vagy 2, aszerint, hogy melyik játékos foglalta a mezőt. Az első számjegy (*i index*) az adott mezőt tartalmazó sornak a sorszámát, a második számjegy (*j index*) az adott mezőt tartalmazó oszlopnak a sorszámát azonosítja. Látható, hogy a jobb felső irányból a bal alsó irányba húzódó átlók közül tizenegy különböző átlót kell megvizsgálni, ugyanis a szürkített mezőkben nem következhet öt elem egymás után.

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

Észrevehetjük, hogy a példaképpen kiválasztott nyolcadik átló elemeit azonosító indexek összege az átló mentén azonos érték (11). Ez minden vizsgálandó átlóra vonatkozóan érvényes. Tegyük fel, hogy a nyolcadik átlóban lévő színezett négyzetek az 1-es játékos foglalásait jelzik. Foglaljunk minden átlóhoz egy változót az egyes játékos számára:

```
int[ ] adb1  = new int[11];
```

Nullázzuk a nyolcadik átlóhoz tartozó változót: *adb1[7] = 0;* , majd lépkedjünk végig a mátrix elemein. Ha mátrix elemeit azonosító indexpár összege tizenegy, és a mezőt lefoglalta az egyes játékos, növeljük az adb1[7] változó tartalmát eggyel, ha nem foglalta le, akkor pedig nullázzuk. Ez ugyanis azt jelenti, hogy a korábbiaktól függetlenül megszakadt a jelölési sorozat. Közben figyelnünk kell, hogy az adb1[7] értéke eléri-e az ötöt. Azzal, hogy figyeljük az egymás után sorban, balról jobbra illetve felülről lefelé olvasott mezők koordinátáinak összegét, meg tudjuk mondani, hányadik átló mezőjét olvassuk (i+j-4). Ezt a trükköt felhasználva elegendő egyszer végigolvasni a mátrix elemeit! Ezzel lényegesen gyorsítjuk a kiértékelést, igaz a négy irány miatt 10+10+11+11=42 változó bevezetésére volt szükség, a két játékost is figyelembe véve pedig 84 változóra. Az alábbi algoritmus az átlósan jobb felső irányból a bal alsó irányba (/) elhelyezett foglalásokat értékeli ki. Lehetőség van a két átlós irány kiértékelő algoritmusának összevonására is, sőt implikálhatjuk a vízszintes és függőleges irány kiértékelését is.

```
Eljárás SzamolAtlo
    Ciklus i:=0 -tól  10-ig
        adb1[i]:=0
    Ciklus vége
    Ciklus i:=0 -tól  9-ig
        Ciklus j:=0 -tól  9-ig
            Ha szám[i][j] = 1
                adb1[i+j-4] := adb1[i+j-4] + 1
                Ha adb1[i+j-4] = 5
                    Ki:= „Nyert az egyes játékos!”
                Elágazás vége
            Egyébként
                adb1[i+j-4] := 0
            Elágazás vége
        Ciklus vége
    Ciklus vége
Eljárás vége
```

A belső ciklusmag 30-adik végrehajtásakor adb1[7]értéke 1, a 39-edikben 2, majd a 48-adik lépésben 0. Ezt követően a megfelelő lépésszám-érték számpárok a következők: 57-1, 66-2, 75-3, 84-4, 93-5. Ezután kiléphetnénk a ciklusból, hiszen a játéknak vége. Ez a minimális időnyereség azonban csak egyszer, a játék utolsó lépésében jelentkezne.