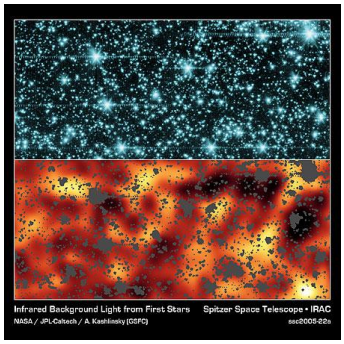
Fém /H arány	0,03	0,02	0,01	0,01	0,001	~ 0
Kor (év)	$< \cdot 10^8$	10^8 -$\cdot 10^9$	10^9 - 10^{10}	1-1,2$\cdot 10^{10}$	>1,2$\cdot 10^{10}$	>1,3 10^{10}
Összesített tömeg (Mo)	2-3$\cdot 10^9$	5$\cdot 10^9$	4,7$\cdot 10^{10}$	4,7$\cdot 10^{10}$	1,6$\cdot 10^{10}$	?
Legfényesebb csillagok	- 8 mg	- 8 mg	- 3 mg	- 3 mg	- 3 mg	?
Főbb képviselők, „markerek”	spirálkarok fiatal csillagai, asszociációk	A típ. csillagok, dMe	galaktikus mag csill., RR Lyr (P<0,4 d)	Runaway csillagok (v_z>30 km/s)	gömbhalmazok, RR Lyr (P>0,4 d)	3,6μ IR fénylés?



8. képmelléklet
A Spitzer űrtávcsővel 3,6 mm-en készített felvétel a Draco csillagkép 6×12 ívperces darabjáról (felső kép) – és a csillagok, galaxisok levonása után maradt derengés (alsó) a sejtések szerint az egykori primordiális (a keresett III. populációs) csillagok fénylése

Hegedüs Tibor

Dinamikus mátrixok. Dinamikus többdimenziós tömbök

1. feladat

Valósítsunk meg dinamikusan egy $n \times m$-elemű kétdimenziós tömböt (mátrixot)! Olvassuk be a tömb elemeit, majd írjuk ki a képernyőre.

1. megoldás

A feladatot visszavezethetjük egydimenziós dinamikus tömbökre.
Legyen a következő a mátrixunk: n=3, m=4.

1	2	3	4
5	6	7	8
9	10	11	12

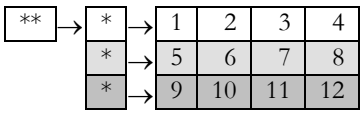
Ez a mátrix a fentiek alapján ekvivalens a következő tömbbel:

1	2	3	4	5	6	7	8	9	10	11	12

És dinamikusan helyet foglalunk az *n*×*m* nagyságú tömbnek. Ebben az esetben nyilván nem használhatjuk a kettős `[i][j]` indexelést, hanem a tömb egy elemét a `[j+i*m]` index segítségével érhetjük el.

2. megoldás

A mátrix sorait külön-külön lefoglaljuk. Az egyes sorokhoz tartozó pointereket egy pointertömbbe tesszük (*n* elemű), amelyet dinamikusan foglalunk le, majd minden pointer egy *m* elemű tömbre mutat. Így a lefoglalt tömböt a megszokott módon használhatjuk a kettős `[i][j]` indexeléssel. Az első indexelés (`i`) a pointerek tömbjéből kiválaszt egy pointert, a második indexelés pedig már a pointerrel mutatott tömbön történik. Előbb a pointerek tömbjét kell lefoglalni, majd utána a sorokat. A felszabadításnál ugyanez visszafelé történik: először a sorokat szabadítjuk fel, majd a pointerek tömbjét. A módszer hátránya az, hogy sok `malloc()` hívás kell hozzá, ami lassabb, mintha csak egy vagy kettő lenne. A módszer nagy előnye viszont az, hogy a soroknak nem feltétlenül kell ugyanolyan hosszúaknak lenniük!




```
#include<stdio.h>
#include<stdlib.h>
int main()
{
  int **t;
  int n, m;
  int i, j;
  scanf("%i", &n);
  scanf("%i", &m);
  // lefoglalás
  t = (int**)calloc(n, sizeof(int*));
  for(i=0; i<n; ++i)
    t[i] = (int*)calloc(m, sizeof(int));
  // beolvasás
  for(i=0; i<n; ++i)
  for(j=0; j<m; ++j)
    scanf("%i", &t[i][j]);
  // kiírás
  for(i=0; i<n; ++i)
  {
    for(j=0; j<m; ++j)
      printf("%3i", t[i][j]);
    printf("\n");
  }
  // felszabadítás
  for(i=0; i<n; ++i)
    free(t[i]);
  free(t);
  return 0;
}
```

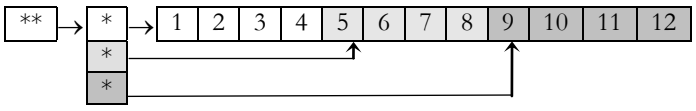
Vagy ha C++-ban programozunk:

```
#include<iostream>
#include<stdlib.h>

using namespace std;

int main()
{
  int** t;
  int n, m;
  int i, j;
  cin>>n;
  cin>>m;
  t = new int*[n];
  for(i=0; i<n; ++i)
  t[i] = new int[m];
  for(i=0; i<n; ++i)
    for(j=0; j<m; ++j)
      cin>>t[i][j];
  for(i=0; i<n; ++i)
  {
    for(j=0; j<m; ++j)
      cout<<t[i][j]<<'\t';
    cout<<endl;
  }
  for(i=0; i<n; ++i)
    delete [] t[i];
  delete [] t;
  return 0;
}
```

3. megoldás

A fenti két módszert keverhetjük is: az sorfolytonos, linearizált tömböt egyetlen `calloc()` hívással lefoglaljuk, mint az első megoldásnál. Deklarálunk egy másik dinamikus tömböt, amely pointerekből áll, mint a második megoldásnál, és ezek a pointerek a sorfolytonos tömb belsejébe mutatnak, mégpedig oda, ahol a kétdimenziós tömb leképezett sorainak elejei vannak. Így, ha indexeljük a pointerekből álló tömböt, egy pointert kapunk, amely a sorfolytonos tömb belsejébe mutat, majd azt is indexelve megkapjuk a keresett elemet.

Ez a módszer gyorsabb foglalást eredményez mint az előző (tehát kiküszöböli az előző megoldás hátrányát), mert mindössze két darab `calloc()` hívásra van hozzá szükség. Először lefoglaljuk a pointertömböt, majd az elemek tömbjét, végül pedig a sorfolytonos tömb belsejébe mutató pointereket számítjuk ki.



```c
#include<stdio.h>
#include<stdlib.h>

int main()
{
  int** t;
  int n, m;
  int i, j;
  scanf("%i", &n);
  scanf("%i", &m);
  // lefoglalás
  t=(int**)calloc(n, sizeof(int*));
  t[0]=(int*)calloc(n*m, sizeof(int));
  for(i=1; i<n; ++i)
    t[i]=t[0]+i*m;
  // beolvasás
  for(i=0; i<n; ++i)
    for(j=0; j<m; ++j)
      scanf("%i", &t[i][j]);
  // kiírás
  for(i=0; i<n; ++i)
  {
    for(j=0; j<m; ++j)
      printf("%3i", t[i][j]);
    printf("\n");
  }
  // felszabadítás
  free(t[0]);
  free(t);
  return 0;
}
```

2. feladat

Valósítsunk meg dinamikusan egy háromszög alakú kétdimenziós tömböt (mátrixot)!

Megoldás

1	2	3	4
5	6	7	
8	9		
10			

```c
#include<stdio.h>
#include<stdlib.h>

int main()
{
  int **t;
  int n;
```

```
    int i, j;
    scanf("%i", &n);
    // lefoglalás
    t = (int**)calloc(n, sizeof(int*));
    for(i=0;i<n;++i)
    t[i] = (int*)calloc(n-i, sizeof(int));
    // beolvasás
    for(i=0; i<n; ++i)
    for(j=0; j<n-i; ++j)
    scanf("%i", &t[i][j]);
    // kiírás
    for(i=0; i<n; ++i)
    {
    for(j=0; j<n-i; ++j)
    printf("%3i", t[i][j]);
    printf("\n");
    }
    // felszabadítás
    for(i=0;i<n;++i)
    free(t[i]);
    free(t);
    return 0;
}
```

Házi feladat

Valósítsuk meg dinamikusan a következő alakú tömböt:

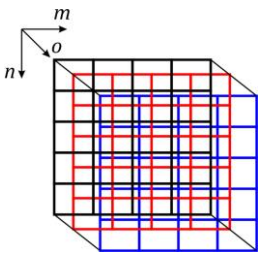
1	2	3		
4				
5	6	7	8	9
10				
11	12	13		

3. feladat

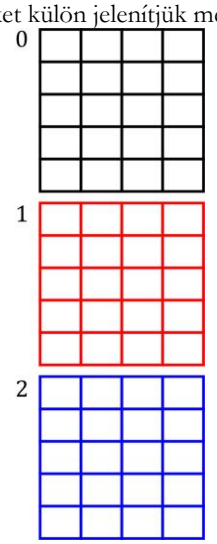
Valósítsunk meg dinamikusan egy $n \times m \times o$-elemű háromdimenziós tömböt! Olvassuk be a tömb elemeit, majd írjuk ki a képernyőre.

Megoldás

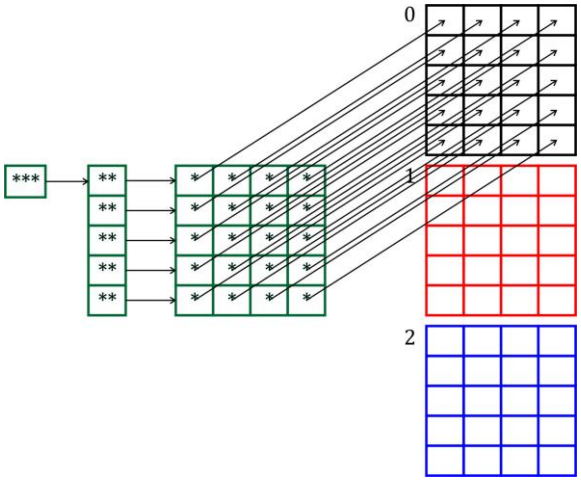
A háromdimenziós tömböt úgy foghatjuk fel, mint egy téglatestet. Például, ha n=5, m=4, o=3, a következő téglatestet kapjuk, és ez egy 5×4×3-as tömb:



Vagy, ha az egyes rétegeket külön jelenítjük meg:



Tehát a következő szerkezetet kell hogy megvalósítsuk pointerekkel:



A program:

```
#include<stdio.h>
#include<stdlib.h>

int main()
{
  int*** t;
  int n, m, o;
  int i, j, k;
  scanf("%i", &n);
  scanf("%i", &m);
  scanf("%i", &o);
  // lefoglalás
  t=(int***)calloc(n, sizeof(int**));
  for(i=0; i<n; ++i)
  {
    t[i]=(int**)calloc(m, sizeof(int*));
    for(j=0; j<m; ++j)
```

```
        t[i][j]=(int*)calloc(o, sizeof(int));
    }
    // beolvasás
    for(k=0; k<o; ++k)
      for(i=0; i<n; ++i)
        for(j=0; j<m; ++j)
          scanf("%i", &t[i][j][k]);
    // kiírás
    for(k=0; k<o; ++k)
    {
      printf("%i:\n", k);
      for(i=0; i<n; ++i)
      {
        for(j=0; j<m; ++j)
          printf("%3i", t[i][j][k]);
        printf("\n");
      }
      printf("\n\n");
    }
    // felszabadítás
    for(i=0; i<n; ++i)
    {
      for (j=0; j<m; ++j)
        free(t[i][j]);
      free(t[i]);
    }
    free(t);
    return 0;
}
```

Megjegyzés

- A fenti megoldás általánosítható tetszőleges dimenziójú tömbökre is.

Kovács Lehel

A szilícium és szilíciumtartalmú ásványok

I. rész

A szilícium nevű kémiai elem a periódusos rendszer 14. csoportjának (IV. főcsoport) második eleme: $_{14}$**Si**. Előfordulási gyakorisága szerint a világmindenségben a hetedik, a földkéregben a második leggyakoribb elem, mivel a földkéreg kőzetei és ezek mállástermékei (talajok, agyagok, homok) majdnem teljesen, megközelítőleg 95%-ban szilikátásványokból és szilícium-dioxidból állnak. Az ezekben előforduló, természetes szilíciumnak három stabil izotópja van: ^{28}Si (92,23%), ^{29}Si (4,67%) és ^{30}Si (3,1%). Földi körülmények között a szilícium soha nem fordul elő szabadon, elemi állapotban, mindig csak oxigénhez kapcsolódva (a SiO_4-egységnek megfelelő négyes koordinációban). Az