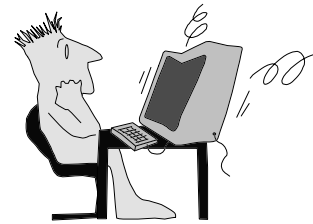


# ELEMI ALKALMAZÁSOK FEJLESZTÉSE I.

## Beágyazott visszavezetés

Készítette: Gregorics Tibor

## Tartalom



- Beágyazott visszavezetéssel készült (programozási tétel a programozási tételben) megoldás kódolása
- Mátrixok használata
- Véletlen szám generátor

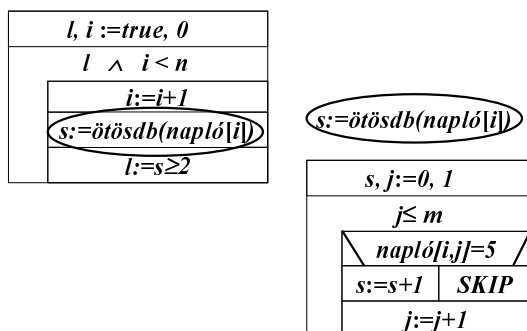
## 1. Feladat

Egy osztályba  $n$  diák jár, akik  $m$  darab tantárgyat tanulnak. Ismerjük a félévvégi osztályzataikat. Igaz-e, hogy minden diáknak van legalább két ötöse?

## Specifikáció

$A = \text{mátrix}(N) \times L$        $\text{mátrix}(N) = N^{n \times m}$   
 $\text{napló} \quad l$   
 $B = \text{mátrix}(N)$   
 $\text{napló}'$   
 $Q = (\text{napló} = \text{napló}')$   
 $R = (Q \wedge l = \forall i \in [1..n] : \text{ötösdb}(\text{napló}[i]) \geq 2)$   
 ahol  $\text{ötösdb}(\text{napló}[i]) = \sum_{j=1}^m \chi(\text{napló}[i,j]=5)$

## Absztrakt program



## Statikus mátrix

Statikus mátrix definíciója  
 elemitípus x[maxsor][maxoszlop];  
 int n,m;

Sor ill.   
 Statikus mátrix feltöltése fájlból  
 cin>>n>>m;  
 while (n>maxsor || m>maxoszlop){  
 cout<<"Hibás adat";  
 cin>>n>>m;  
 }  
 for (int i=0; i<n; i++){  
 for (int j=0; j<m; j++){  
 cin>>x[i][j];  
 }  
 }

## Dinamikus mátrix

### Dinamikus mátrix deklarációja

```
elemittipus** x;  
int n,m;
```

### Dinamikus mátrix definíciója és feltöltése

```
cin>>n>>m;  
x=new int*[n];  
for (int i=0;i<n;i++){  
    x[i]=new int[m];  
    for (int j=0;j<m;j++){  
        cin>>x[i][j];  
    }  
}
```

### Dinamikus mátrix megszüntetése

```
for (int i=0;i<n;i++){  
    delete[] x[i];  
    delete[] x;  
}
```

## Kódolás

$s := \text{ötösdb}(\text{napló}[i])$

$s, j := 0, 1$

$j \leq m$

$\text{napló}[i, j] = 5$

$s := s + 1$  SKIP

$j := j + 1$

```
s=0; j=0;  
while (j<=m-1){  
    if (naplo[i][j]==5)  
        s++;  
    j++;  
}
```

```
s=0;  
for (j=0; j<m; j++) {  
    if (v[j]==5) s++;  
}
```

## Kódolás

$s := \text{ötösdb}(\text{napló}[i])$

$$s = \sum_{j=1}^m \chi(\text{napló}[i][j]=5)$$

bemenő adat:  $\text{napló}[i]$  ami egy  $v \in \mathbb{N}^m$

kimenő adat:  $s \in \mathbb{N}$

segéd adat:  $j \in \mathbb{N}$

```
int otosdb(int* v, int m)  
{  
    int s=0;  
    for (int j=0; j<m; j++) {  
        if (v[j]==5) s++;  
    }  
    return s;  
}
```

## Kódolás

$l = \forall i \in [1..n] : \text{ötösdb}(\text{napló}[i]) \geq 2$

$l, i := \text{true}, 0$

$l \wedge i < n$

$i := i + 1$

$s := \text{ötösdb}(\text{napló}[i])$

$l := s \geq 2$

```
l=true; i=-1;  
while (l && i<n-1){  
    i++;  
    s=ototodb(naplo[i])  
    l= s>=2;  
}
```

## Kódolás

$l = \forall i \in [1..n] : \text{ötösdb}(\text{napló}[i]) \geq 2$

bemenő adat:  $\text{napló} \in \mathbb{N}^{n \times m}$

kimenő adat:  $l \in \mathbb{L}$

segéd adat:  $i \in \mathbb{N}$

```
bool mindenkinek_ket_otos(int** naplo,  
                           int n, int m)  
{  
    bool l=true;  
    int i=-1;  
    while (l && i<n-1) {  
        l=ototodb(naplo[++i],m)>=2;  
    }  
    return l;  
}
```

## C++ program

```
// Fordítási direktívák  
  
// Függvény deklarációk  
  
int main( )  
{  
    // Adatok beolvasása  
  
    // Kiértékelés és kiírás  
  
    // Lezárás  
}  
  
// Függvény definíciók
```

## Programtörzs

```
// Adatok beolvasása
beolvas(naplo,n,m,tanulo,targy);

// Kiértékelés és kiiratás
if (mindenkinek_ket_otos(naplo,n,m)) {
    cout<<"Mindenkinek legalább két ötöse van!";
}
else {
    cout<<"Van, akinek nincs két ötöse!";
}

// Lezárás
lezar(naplo, n, m);
```

naplo.cpp

```
#include <iostream>
#include <string>

void beolvas(int** &naplo, int &n, int &m,
             string* &tanulo, string* &targy);
bool mindenkinek_ket_otos(int** naplo, int n, int m);
int otosdb(int* v, int m);
void lezar(int** &naplo, int n, int m);

int main()
{
    int**      naplo;
    int        n,m;
    string*    tanulo;
    string*    targy;
```

naplo.cpp

```
// Adatok beolvasása
beolvas(naplo,n,m,tanulo,targy);

// Kiértékelés és kiiratás
if (mindenkinek_ket_otos(naplo,n,m)) {
    cout<<"Mindenkinek legalább két ötöse van! ";
}
else {
    cout<<"Van, akinek nincs két ötöse! ";
}

// Lezárás
lezar(naplo, n, m);
cout<<endl;
char kar;
cin>>kar;
return 0;
}
```

naplo.cpp

```
// Függvény definíciók

void beolvas(int** &naplo, int &n, int &m,
             string* &tanulo, string* &targy)
{
    cout<< "Hány tanuló van? ";
    cin>>n;

    tanulo=new string[n];
    cout<< "Adja meg a tanulók neveit: "<<endl;
    for(int i=0; i<n; i++) {
        cout<< "Tanuló neve: ";
        cin>>tanulo[i];
    }
    cout<<endl;
```

naplo.cpp

```
cout<<"Hány tantárgy van? ";
cin>>m;

targy=new string[m];
cout<<"Adja meg a tantárgyakat: "<<endl;
for(int j=0; j<m; j++) {
    cout<<"Tantárgy neve: ";
    cin>>targy[j];
}
cout<<endl;
```

naplo.cpp

```
naplo=new int*[n];
cout<<"Adja meg az osztályzatokat!"<<endl;
for(int i=0; i<n; i++) {
    naplo[i]=new int[m];
    cout<<tanulo[i]<<" tanuló"<<endl;
    for(int j=0; j<m; j++) {
        cout<<"\t"<<targy[j]<<":";
        cin>>naplo[i][j];
    }
}
cout<<endl;
}
```

```

naplo.cpp

void lezar(int** &naplo, int n, int m)
{
    for (int i=0; i<n; i++) {
        delete[] naplo[i];
    }
    delete[] naplo;
}

```

```

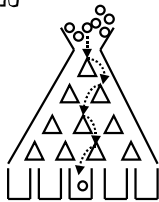
naplo.cpp

bool mindenkinek_ket_otos(int** naplo, int n, int m)
{
    bool l=true;
    int i=-1;
    while (l && i<n-1) {
        l=otosdb(naplo[++i],m)>=2;
    }
    return l;
}

int otosdb(int* v, int m)
{
    int s=0;
    for (int j=0; j<m; j++) {
        if (v[j]==5) s++;
    }
    return s;
}

```

## 2. Feladat



Szimuláljuk Galton deszkájának működését!

## Szimuláció

$$A = \begin{matrix} N & \times & N & \times & \text{vector}(N) \\ \text{szint} & \text{db} & & v \end{matrix} \quad \text{vector}(N) = N_0^{\text{szint}-1}$$

Kezdetben:  $\forall j \in [0..\text{szint}-1]$ -dik zsákra :  $v[j] = 0$

$\forall i \in [1..db]$  golyóra:  
 ha  $\text{jobbbradb}(i) \neq j$  akkor  $v[j]$ -t eggyel növeljük  
 vagy másképpen  
 $v[\text{jobbbradb}(i)]$ -t eggyel növeljük

ahol  $\text{jobbbradb}(i) = \sum_{k=1}^{\text{szint}} \chi(\text{véletlen}(0,1)=1)$

## Specifikáció

$$A = \begin{matrix} N & \times & N & \times & \text{vector}(N) \\ \text{szint} & \text{db} & & v \end{matrix} \quad \text{vector}(N) = N_0^{\text{szint}-1}$$

$$B = \begin{matrix} N & \times & N \\ \text{szint}' & \text{db}' \end{matrix}$$

$$Q = (\text{szint} = \text{szint}' \wedge \text{db} = \text{db}')$$

$$R = (Q \wedge \forall i \in [1..db] : \text{hova}[i] = \text{jobbbradb}(i) \wedge \forall j \in [0..\text{szint}-1] : v[j] = \sum_{i=1}^{db} \chi(\text{hova}[i]=j))$$

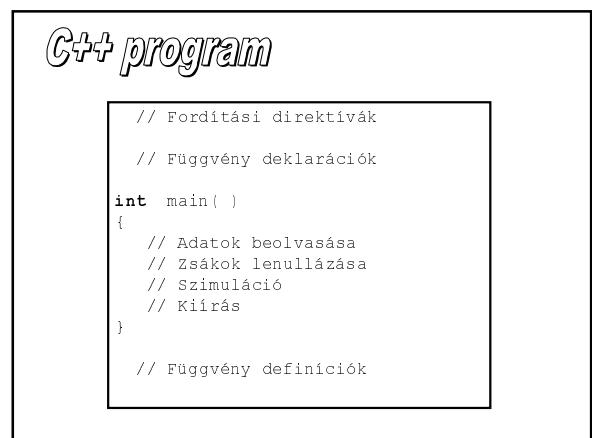
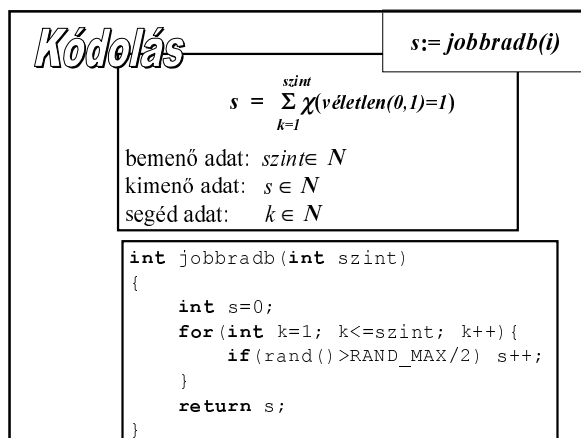
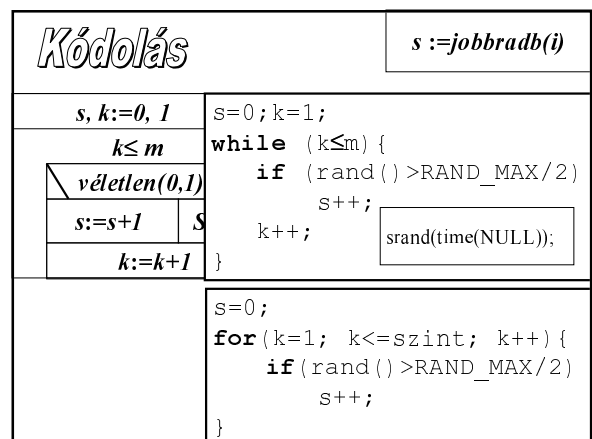
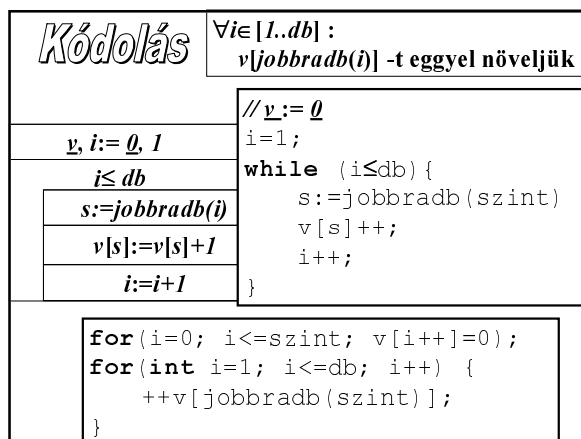
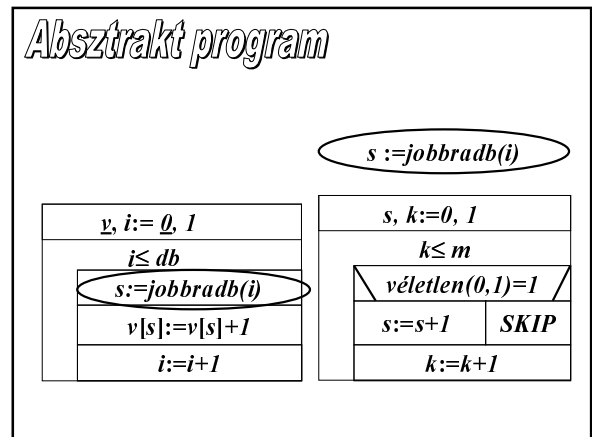
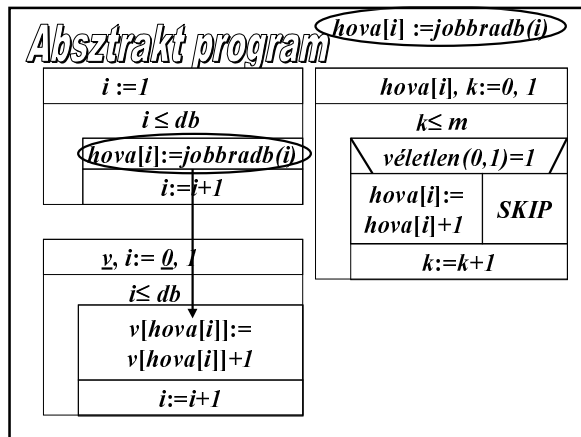
ahol  $\text{jobbbradb}(i) = \sum_{k=1}^{\text{szint}} \chi(\text{véletlen}(0,1)=1)$

## Absztrakt program

$\text{hova}[i] := \text{jobbbradb}(i)$

$i := 1$	$\text{hova}[i], k := 0, 1$
$i \leq db$	$k \leq m$
$\text{hova}[i] := \text{jobbbradb}(i)$	$\text{véletlen}(0,1)=1$
$i := i+1$	$\text{hova}[i] := \text{hova}[i]+1$ <b>SKIP</b>
$\forall j \in [0..\text{szint}-1] :$	$k := k+1$
$v[j], i := 0, 1$	
$i \leq db$	
$\text{hova}[i]=j$	
$v[j] := v[j]+1$ <b>SKIP</b>	
$i := i+1$	

Bármelyik  $i$ -re  
 a  $v$  vektornak csak  
 a  $\text{hova}[i]$ -edik eleme változik



```
Galton.cpp

#include <iostream>
#include <iomanip>
#include <stdlib.h>
#include <time.h>

int hany_jobbra(int szint);
void kiir(int* v, int szint);

int main()
{
    int* v;
    int szint, db;

    // Adatok beolvasása
    cout<< "Hány szintű legyen a Galton deszka: ";
    cin>>szint;
    cout<<"Hány golyót engedjünk le: ";
    cin>>db;
}
```

```
Galton.cpp

// Zsákok lenullázása
v=new int[szint+1];
for(int i=0; i<=szint; v[i++]=0);

// Szimuláció
srand(time(NULL));
for(int i=1; i<=db; i++) {
    ++v[hany_jobbra(szint)];
}

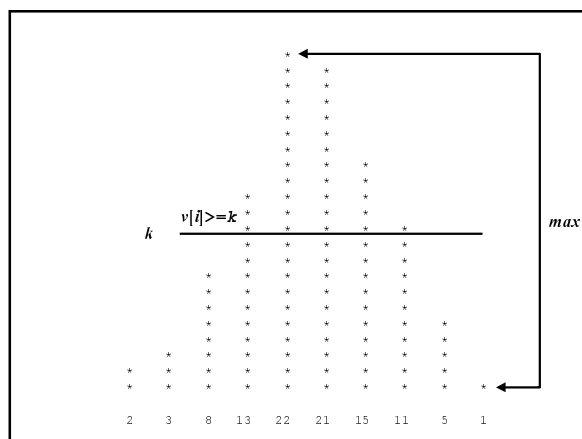
// Kiírás
kiir(v,szint);

delete[] v;
return 0;
}
```

```
Galton.cpp

// Függvény definíciók

int hany_jobbra(int szint)
{
    int s=0;
    for(int k=1; k<=szint; k++)
    {
        if(rand()>RAND_MAX/2) s++;
    }
    return s;
}
```



```
Galton.cpp

void kiir(int* v, int szint)
{
    int max=v[0];
    for(int i=1; i<=szint; i++)
    {
        if (max<v[i]) max=v[i];
    }

    cout<<endl;
    for(int k=max; k>=0; k--)
    {
        for(int i=0; i<=szint; i++)
        {
            cout<<( v[i]>=k ? " * " : "   ");
        }
        cout<<endl;
    }
}
```

```
Galton.cpp

cout<<endl<<setw(3);
for(int i=0; i<=szint; i++)
{
    cout<<v[i]<<setw(5);
}
cout<<endl;

char kar;
cin>>kar;
}
```