PROGRAMOZÁS

Programozás beágyazott visszavezetéssel

Gregorics Tibor

http://people.inf.elte.hu/gt/prog

Feladat

Egy osztályba *n* diák jár, akik *m* darab tantárgyat tanulnak. Ismerjük a félévvégi osztályzataikat, és lehet olyan eset, hogy egy diák bizonyos tárgyakból nem kap egyáltalán jegyet (ezt nulla jelzi).
Ki a legjobb (átlagú) diák azok között, akik legalább egy tárgyból kaptak osztályzatot?

feltételes maximum keresés: maximális átlagú diák, feltéve, hogy kapott osztályzatot

Specifikáció

A= (napló : $\mathbb{N}^{n\times m}$, l: $\mathbb{L}$, max: $\mathbb{R}$, ind: $\mathbb{N}$)

Ef=(napló = napló')

$$Uf = \left(Ef \wedge (l,max,ind) = \max_{\substack{i=1 \\ darab(i)>0}}^{n} (összeg(i)/darab(i)) \right)$$

az i-dik diák átlaga

az i-dik diák jegyeinek összege

az i-dik diák tantárgyainak száma

ahol

összeg: $[1..n] \rightarrow \mathbb{N}$

darab: $[1..n] \rightarrow \mathbb{N}$

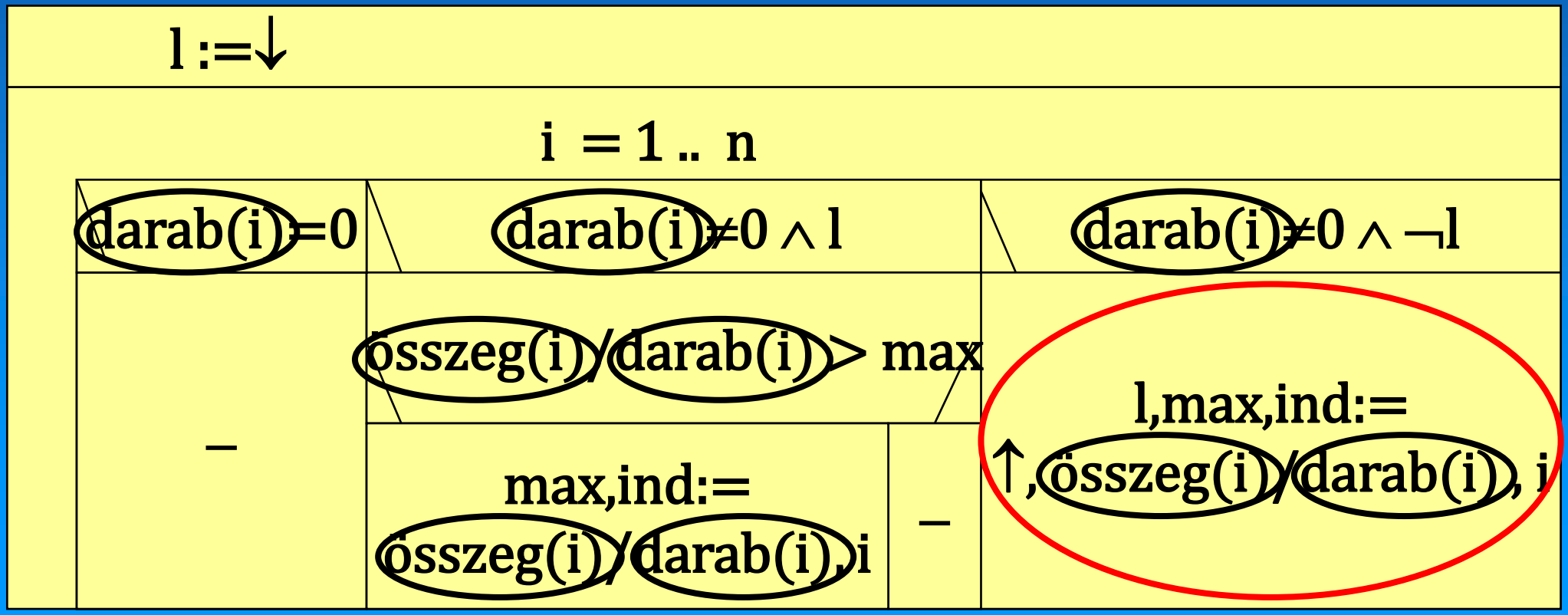
$$összeg(i) = \sum_{j=1}^{m} napló[i,j]$$

$$darab(i) = \sum_{\substack{j=1 \\ napló[i,j]\neq 0}}^{m} 1$$

Fő program

feltételes maximum keresés

m .. n	~ 1 .. n
β(i)	~ darab(i)>0
f(i)	~ összeg(i)/darab(i)
H , >	~ ℝ , >



Nem-megengedett feltétel kiemelése

Egy kifejezés ***nem-megengedett***, ha a választott implementációs környezetben nem tudjuk egy az egyben kódolni.
Ha egy feltétel ilyen, akkor ***nem-megengedett feltételről***, ha egy értékadás jobboldali kifejezése ilyen, akkor ***nem-megengedett értékadásról*** beszélünk.

l,max,ind :=
↑, összeg(i) / darab(i) , i

feltétel kiemelése önálló értékadásba
új segédváltozók bevezetésével

segéd: s:ℕ, c:ℕ

s := összeg(i)

c := darab(i)

l,max,ind := ↑, s / c , i

A ***top-down tervezés*** során kapott nem-megengedett értékadásokat megengedett részprogrammal kell helyettesíteni.
(továbbtervezés vagy ***finomítás***).

Részfeladatok

Kiolvasható a részfeladatok teljes specifikációja

s := összeg(i)

$$\text{összeg}: [1..n] \to \mathbb{N}$$

$$\text{összeg}(i) = \sum_{j=1}^{m} \text{napló}[i,j]$$

összegzés

m .. n	~ 1 .. m
f(i)	~ napló[i, j]
i	~ j

s := 0

j = 1 .. m

s := s+napló[i, j]

c := darab(i)

$$\text{darab}: [1..n] \to \mathbb{N}$$

$$\text{darab}(i) = \sum_{\substack{j=1 \\ \text{napló}[i,j]\neq 0}}^{m} 1$$

számlálás

m .. n	~ 1 .. m
β(i)	~ napló[i, j] ≠ 0
i	~ j

c := 0

j = 1 .. m

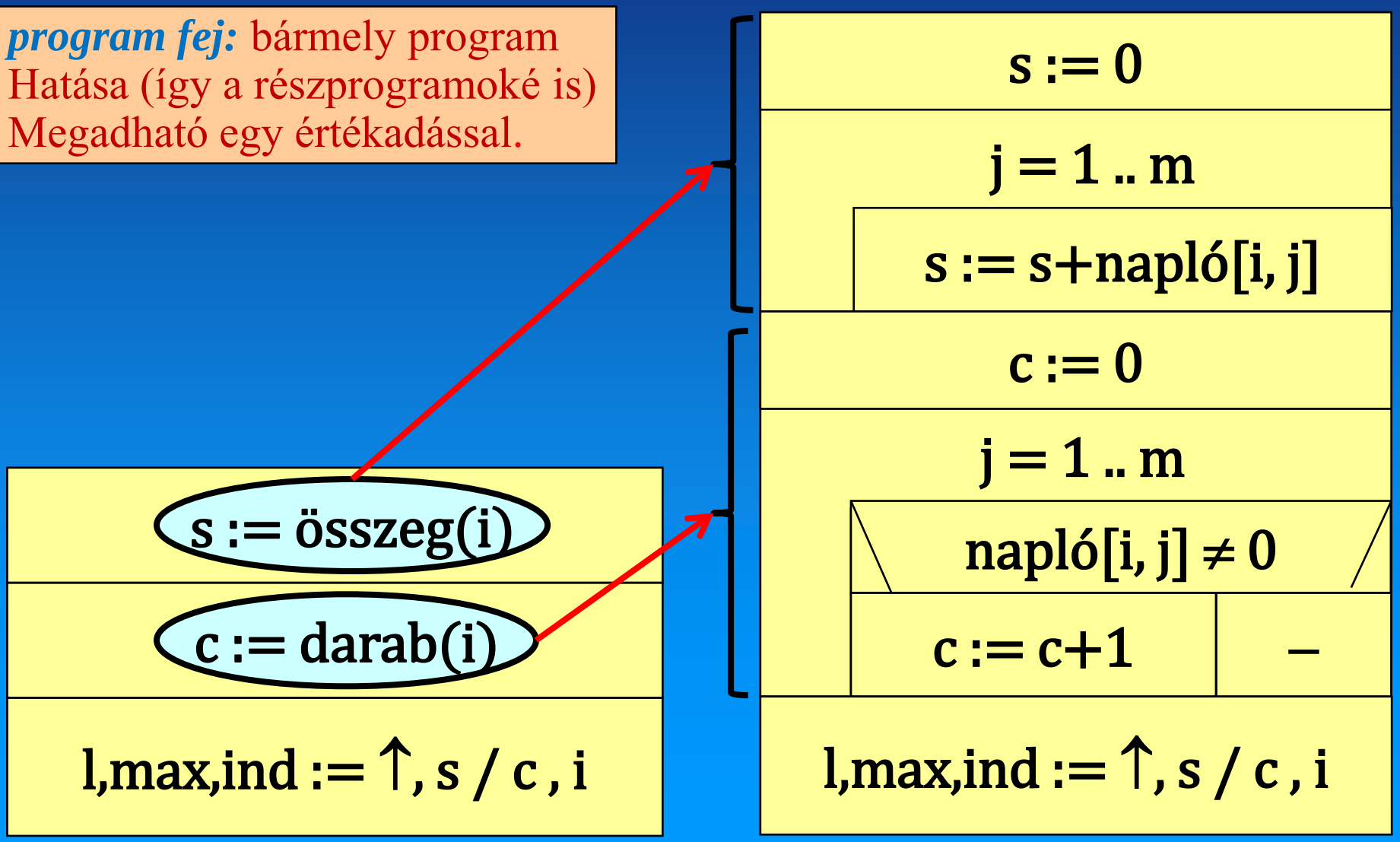
napló[i, j] ≠ 0

c := c+1 | –

Részprogram beágyazása

A ***részprogram*** egy másik programba beágyazott program.

program fej: bármely program
Hatása (így a részprogramoké is)
Megadható egy értékadással.

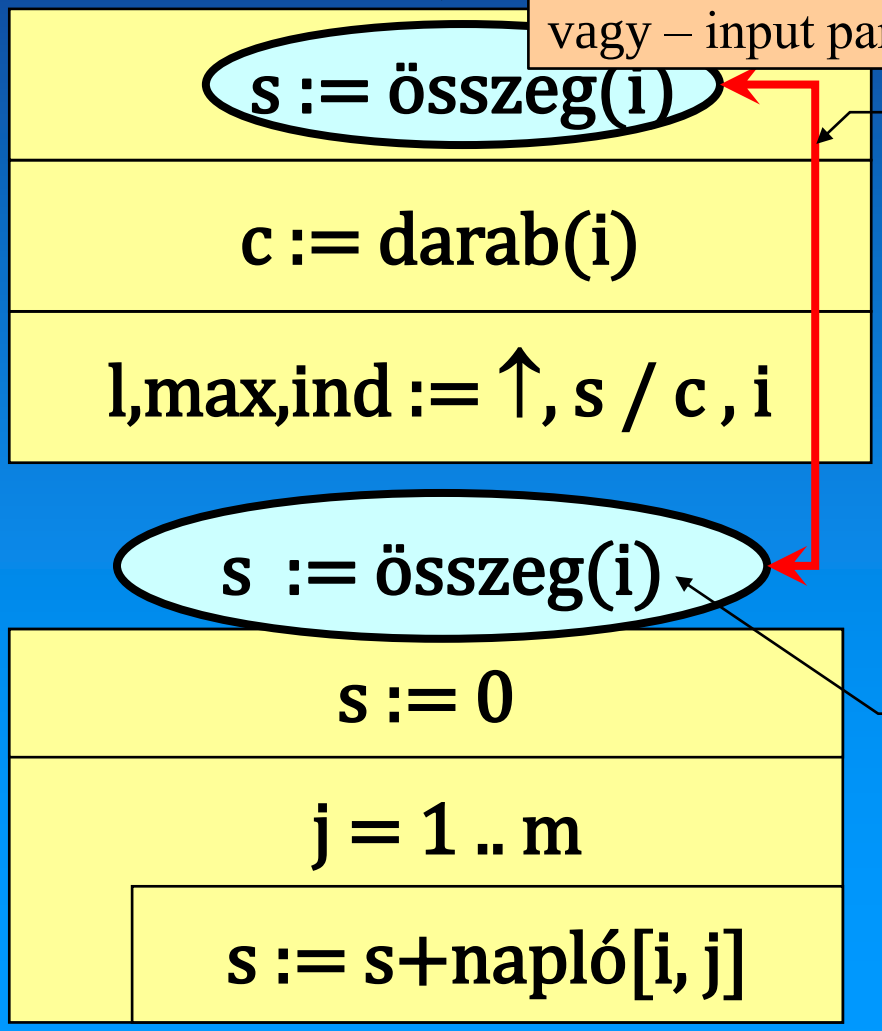


Részprogram helyett alprogram

Az ***alprogram*** egy olyan részprogram, amelyet az az alprogram fejére történő hivatkozással (hívással) aktiválhatunk.

sorrend és ***típus-megfelelés***

A ***hívásban*** a paraméterváltozók helyén azokkal kompatibilis típusú ***aktuális paraméterek*** állnak. Ezek vagy ***változók*** (elkülönülnek a paraméterváltozóktól, de esetenként velük azonos nevet viselnek), vagy – input paraméterváltozók esetében – ***kifejezések*** lehetnek.



1. a hívó program végrehajtása megszakad
2. létrejönnek az alprogram paraméterváltozói
3. az input változók megkapják a hívásban megfelelő változók vagy kifejezések értékeit
4. végrehajtódik az alprogram
5. az output változók értéke visszaadódik a hívás megfelelő változóiba
6. a hívó program folytatódik

Az ***alprogram fejének*** (ami egy értékadás) jobboldalán ***input paraméterváltozók***, baloldalán ***output paraméterváltozók*** állnak (egy változó mindkét helyen is szerepelhet). Ezek a változók ***lokálisak*** az alprogramra nézve: annak indulásakor jönnek létre és leállásakor szűnnek meg.

Különféle változók a programban

feladat:

napló:$\mathbb{N}^{n\times m}$, l:$\mathbb{L}$, max:$\mathbb{R}$, ind:$\mathbb{N}$

segéd:

i:$\mathbb{N}$, s:$\mathbb{N}$, c:$\mathbb{N}$

Az alprogram változói:
- alprogram paraméterváltozói
- alprogram segéd változói
- főprogram látható változói, de csak olvasásra használhatóak

s := összeg(i)

c := darab(i)

l,max,ind := ↑, s / c , i

napló az alprogram ***globális változója*** amit csak olvasásra használhat

paraméter:

s:$\mathbb{N}$, i:$\mathbb{N}$

s := összeg(i)

s := 0

j = 1 .. m

s := s+napló[i, j]

segéd:

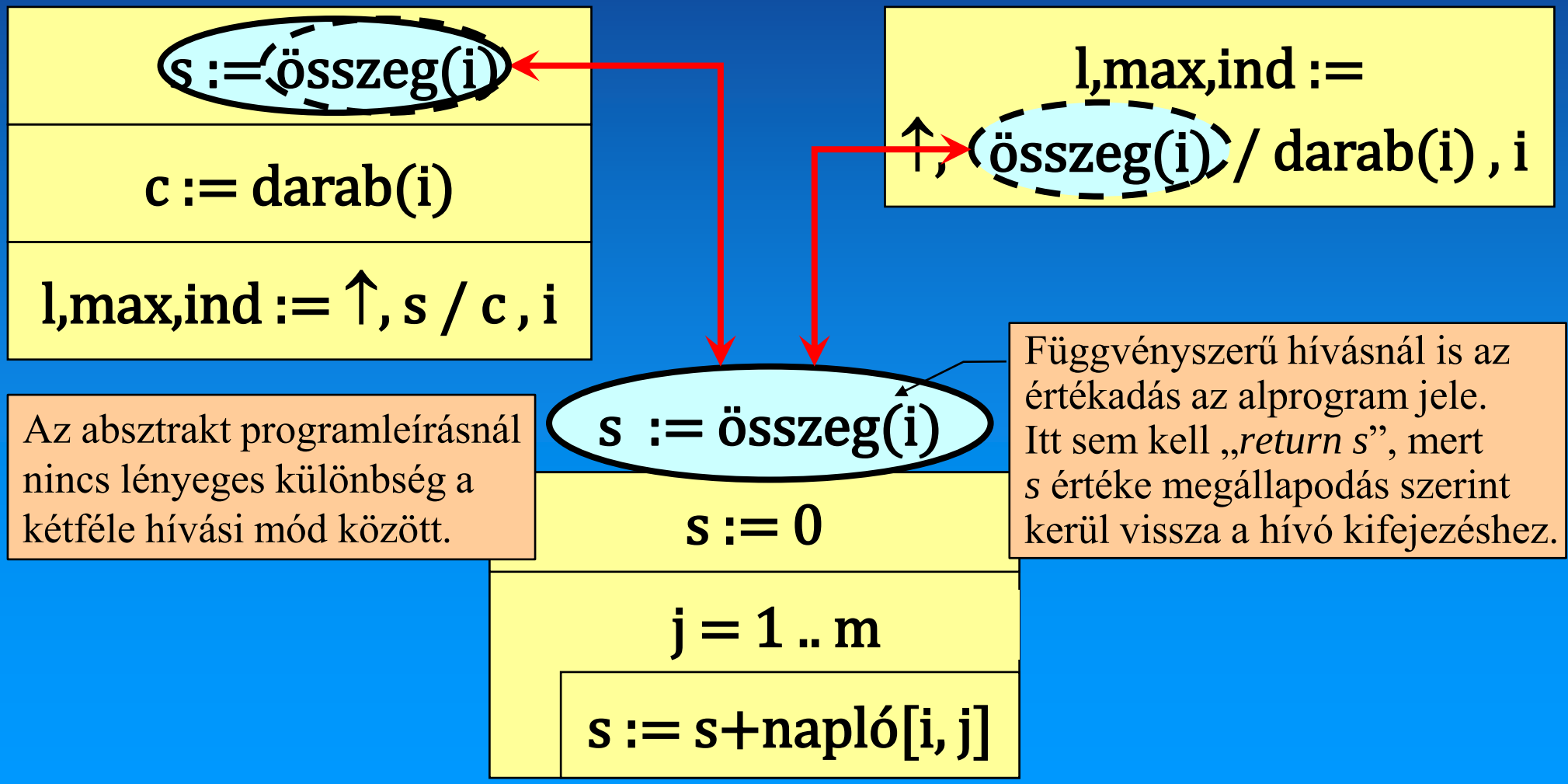
j:$\mathbb{N}$

az alprogram ***lokális változói*** közül az *s* és *i* nem azonos a főprogram *s* és *i* segédváltozóival

Alprogramok hívási módjai

Egy alprogram ***eljárásszerű hívása*** annak fejére, tehát a teljes értékadásra történő hivatkozás (***hívó utasítás***).

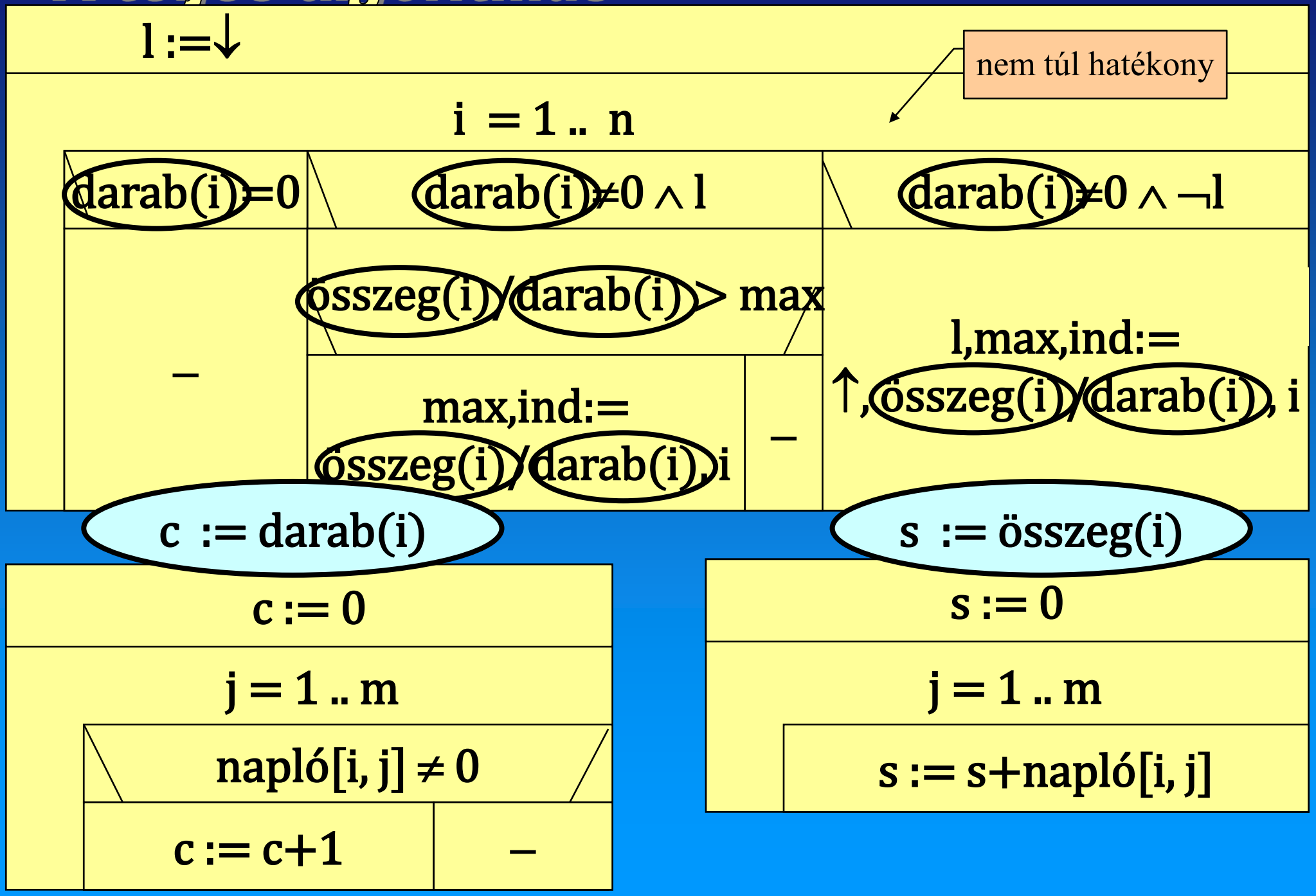
A ***függvényszerű hívás*** az alprogram fejének jobboldalára történő hivatkozás (***hívó kifejezés***). Az alprogram befejeződésekor az output paraméterváltozók együttes értéke lesz a hívó kifejezés értéke (***visszatérési érték***).



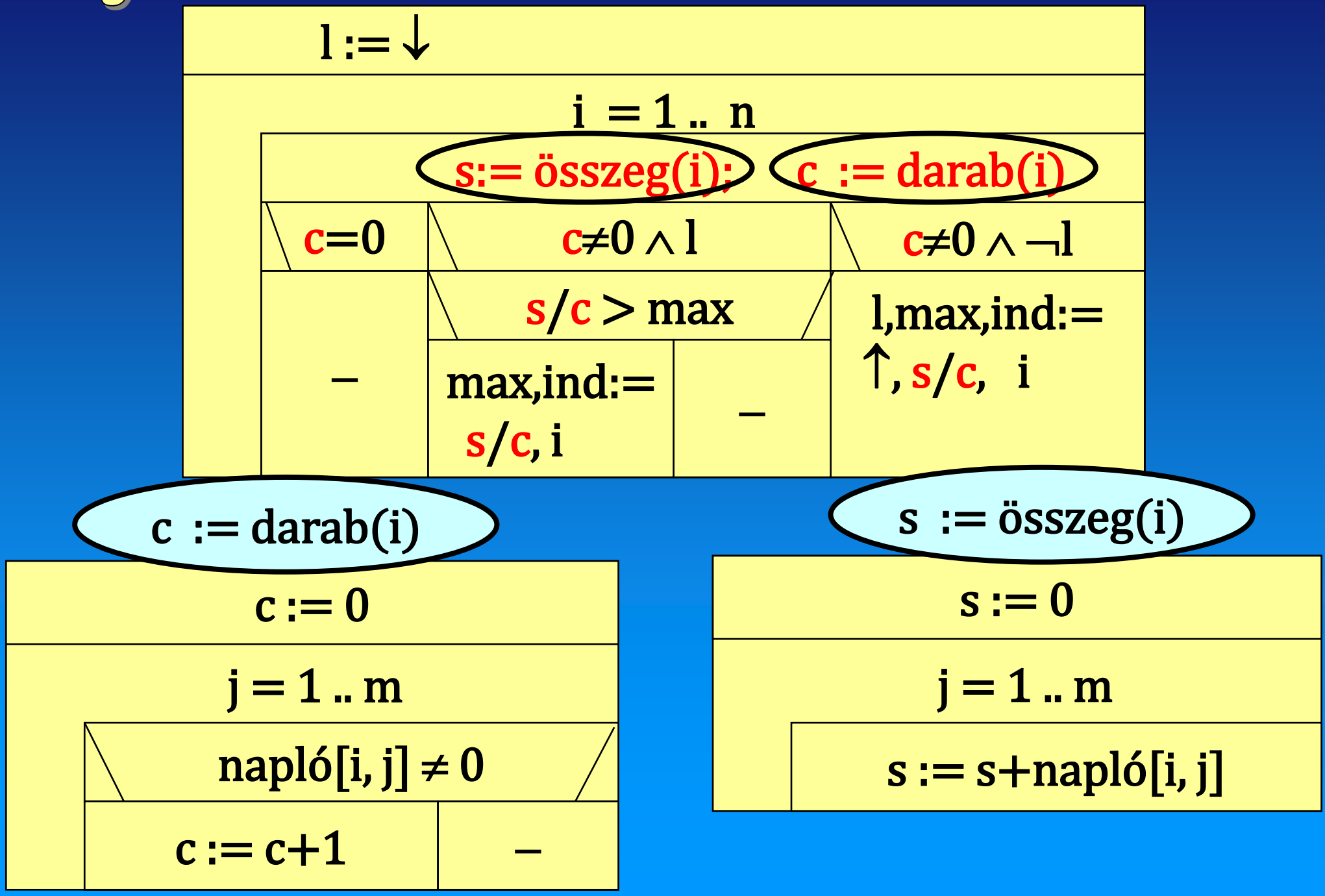
Az absztrakt programleírásnál nincs lényeges különbség a kétféle hívási mód között.

Függvényszerű hívásnál is az értékadás az alprogram jele. Itt sem kell „*return s*”, mert *s* értéke megállapodás szerint kerül vissza a hívó kifejezéshez.

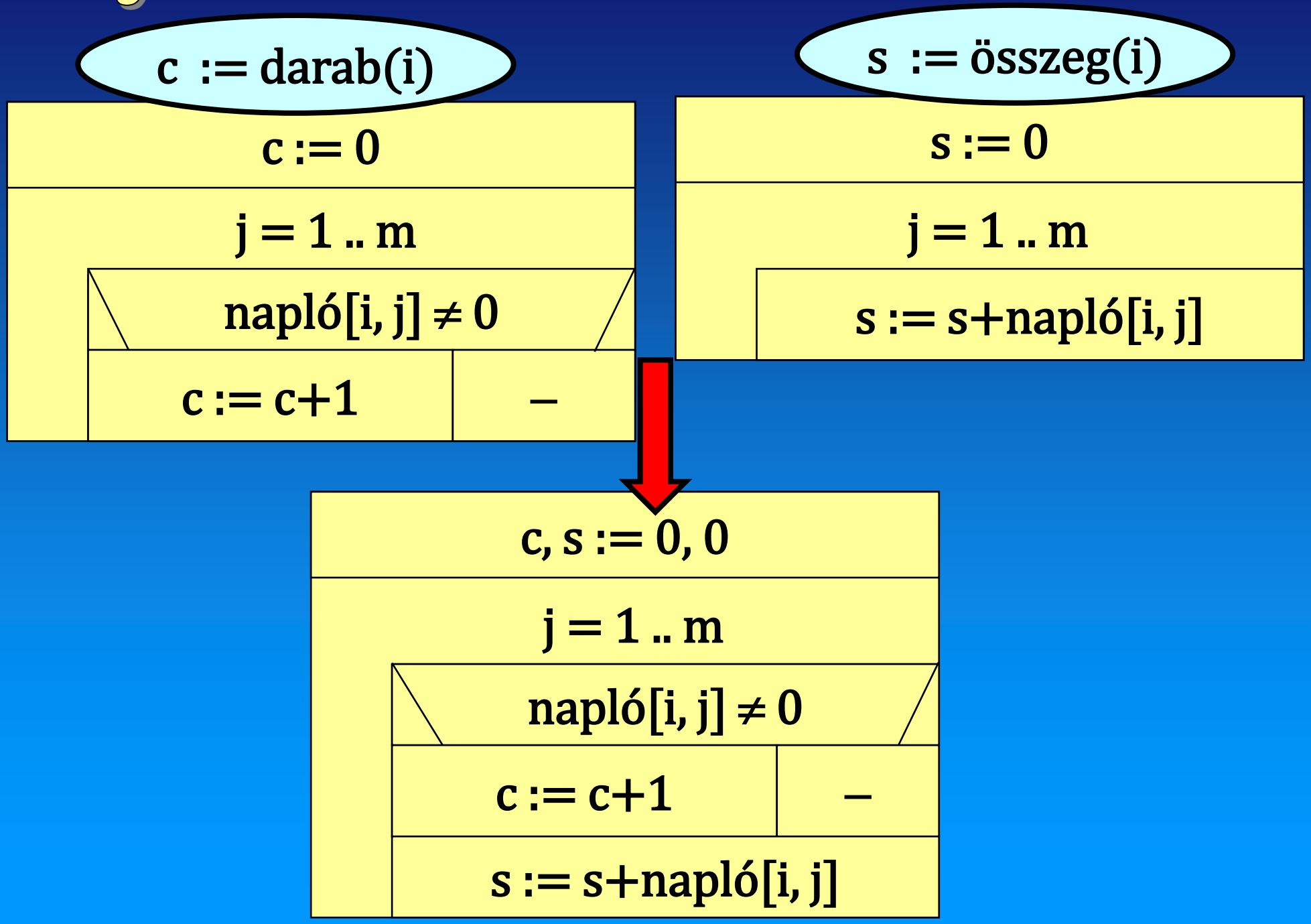
A teljes algoritmus



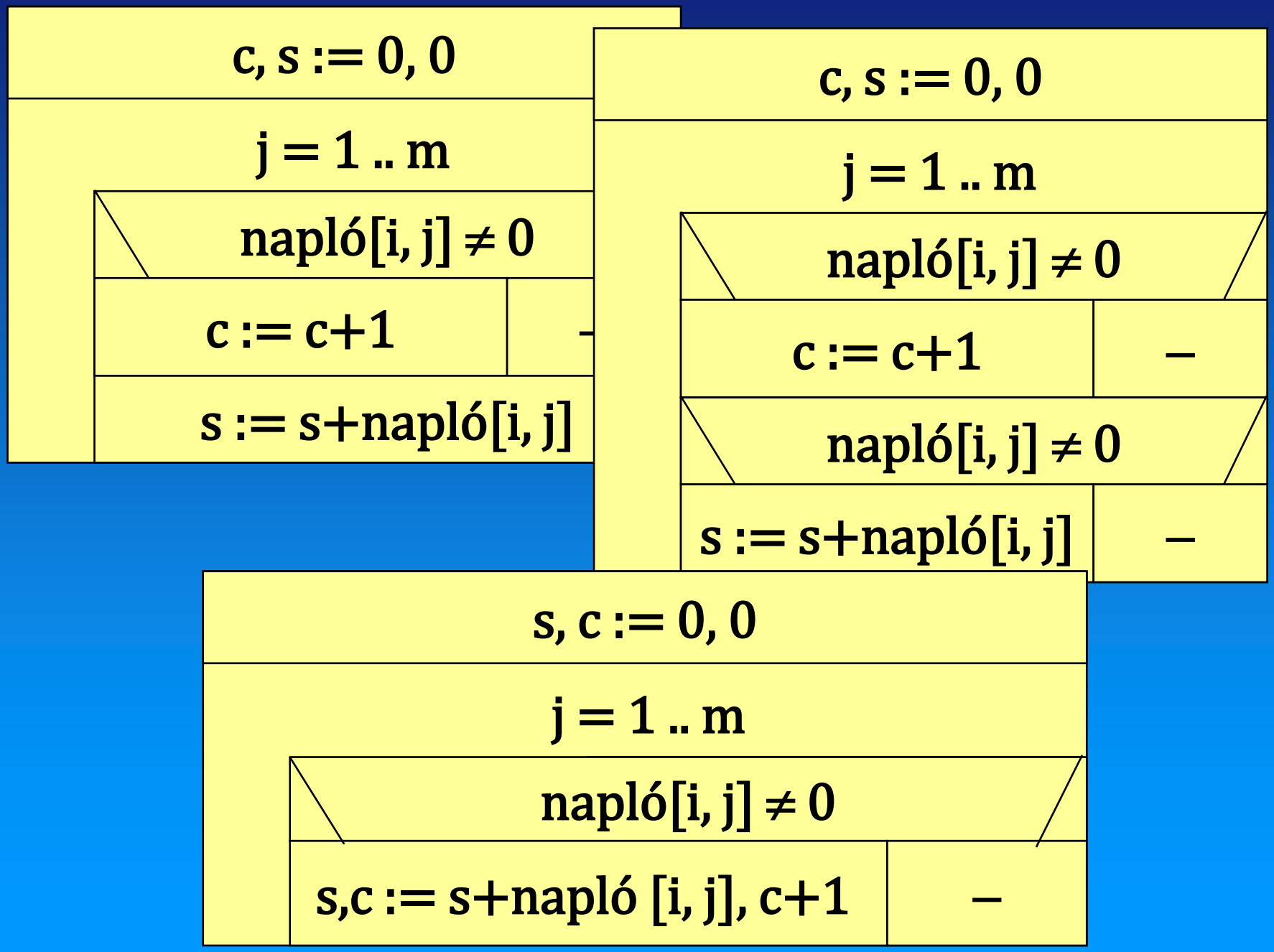
Program átalakítása : kiemelés



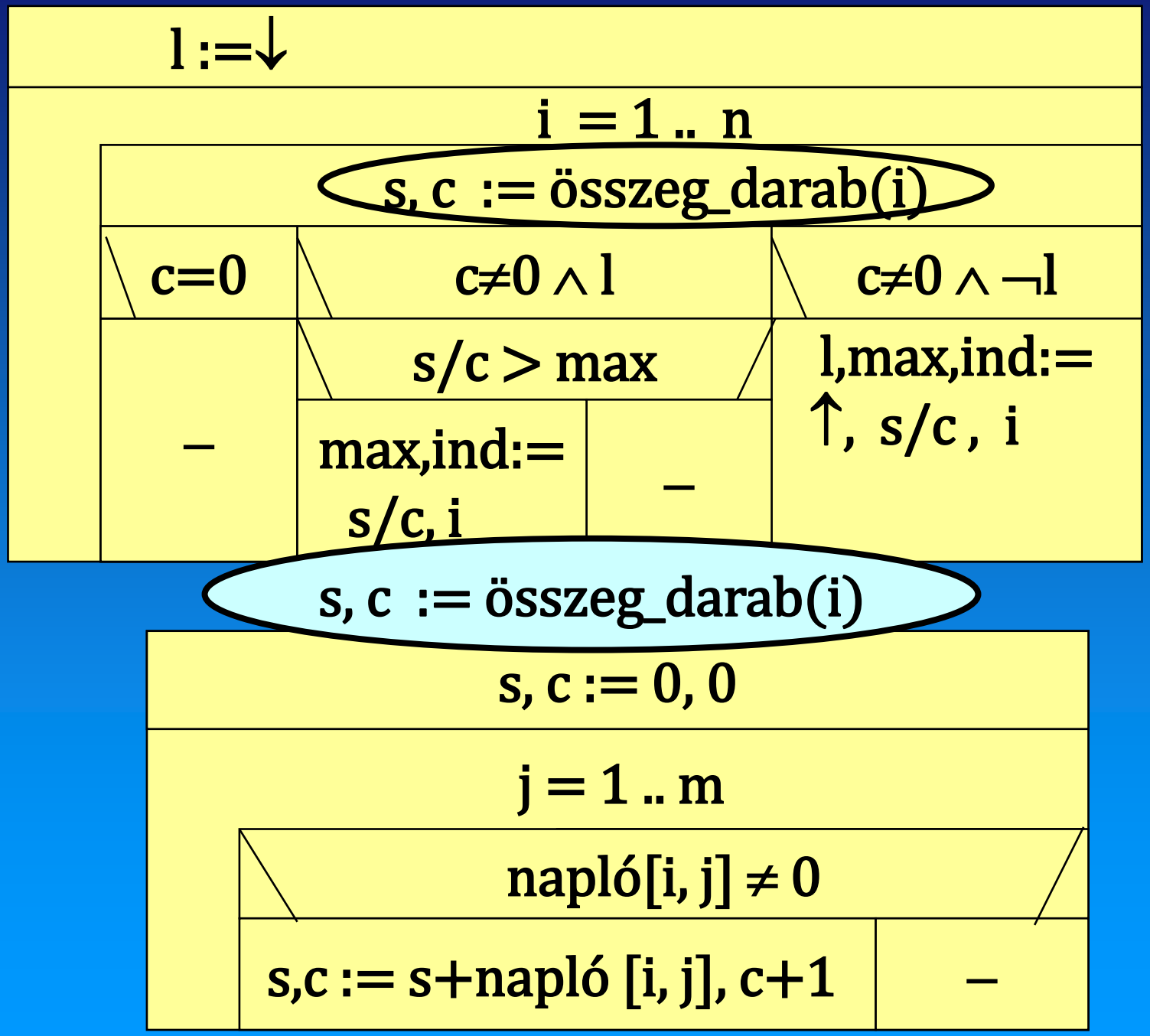
Program átalakítása: ciklusok összevonása



Program átalakítása: elágazások összevonása



Program végleges alakja



Program-átalakítások

Helyes programból helyes programot

- Hatékonyságot javító (tervezéskor)
  - Egymás után többször szereplő, ugyanazon nem-megengedett feltétel kiemelése egy értékadásba.
  - Szekvenciában álló, független, de azonos feltételű elágazások illetve azonos feltételű ciklusok összevonása
- Megvalósítást támogató (kódoláskor)
  - Szimultán értékadás megszüntetése
  - Három vagy többágú elágazás kódolása

Tesztesetek: fekete doboz/ érvényes/felt. max. ker

- intervallum hossza: diákok száma (n) szerint

nulla	n=0	⟶ l=hamis
egy	n=1, m=3, naplo = [2, 2, 5]	⟶ l,max,ind=igaz,3,1
egy	n=1, m=0, naplo = [ ]	⟶ l=hamis
több	n=3, m=3, naplo = $\begin{bmatrix} 2, 2, 5 \\ 0, 0, 0 \\ 1, 1, 4 \end{bmatrix}$	⟶l,max,ind=igaz,3,1

- intervallum eleje: első diák legyen a válasz

  n=3, m=3, lásd előző ⟶l,max,ind=igaz,3,1

- intervallum vége: utolsó diák legyen a válasz

  n=3, m=3, naplo = $\begin{bmatrix} 1, 1, 1 \\ 0, 0, 0 \\ 1, 1, 4 \end{bmatrix}$ ⟶ l,max,ind=igaz,2,3

- eredmény: nincs érvényes diák (a nincs diák esetét már néztük)

  n=2, m=3, naplo = $\begin{bmatrix} 0, 0, 0 \\ 0, 0, 0 \end{bmatrix}$ ⟶ l,max,ind=hamis,2,3

Tesztesetek: fekete doboz/ érvényes/felt. max. ker

- eredmény: folytatás

van egy érvényes diák

n=2, m=3, naplo= $\begin{bmatrix} 0, 0, 0 \\ 1, 0, 0 \end{bmatrix}$ ⟶ l,max,ind=igaz,1,2

több érvényes diák van, legjobb középen

n=3, m=3, naplo = $\begin{bmatrix} 0, 0, 0 \\ 5, 0, 0 \\ 1, 2, 1 \end{bmatrix}$ ⟶ l,max,ind=igaz,5,2

több érvényes diák van, legjobb az első …

több érvényes diák van, legjobb az utolsó …

több érvényes egyformán legjobb diák

n=3, m=3, naplo = $\begin{bmatrix} 0, 0, 0 \\ 5, 0, 0 \\ 5, 5, 0 \end{bmatrix}$ ⟶ l,max,ind=igaz,5,1

törtszám az átlag

n=2, m=3, naplo = $\begin{bmatrix} 1, 2, 4 \\ 2, 2, 4 \end{bmatrix}$ ⟶ l,max,ind=igaz,2.33,1

Tesztesetek: fekete doboz/ érvényes/össz-száml.

- intervallum hossza: tárgyak száma (m) szerint
  - nulla m=0 ⟶ l=hamis
  - egy n=1, m=2, naplo=[2, 2] ⟶ l,max,ind=igaz,2,1
  - több n=1, m=3, naplo=[2, 2, 5] ⟶ l,max,ind=igaz,3,1
- intervallum eleje: első tárgy beszámít-e
  - n=1, m=3, naplo=[0, 2, 5] ⟶ l,max,ind=igaz,2.5,1
  - naplo=[2, 2, 5] ⟶ l,max,ind=igaz,3,1
- intervallum vége: utolsó tárgy beszámít-e
  - n=1, m=3 naplo=[5, 2, 0] ⟶ l,max,ind=igaz,2.5,1
  - naplo=[5, 2, 2] ⟶ l,max,ind=igaz,3,1
- eredmény: számlálás eredménye 0, 1, több
  - n=1, m=3, naplo=[0, 0, 0] ⟶ l=hamis
  - naplo=[5, 0, 0] ⟶ l,max,ind=igaz,5,1
  - naplo=[5, 5, 0] ⟶ l,max,ind=igaz,5,1

Változók szerepköre

feladat:

napló:$\mathbb{N}^{n\times m}$, *l*:$\mathbb{L}$, *max*:$\mathbb{R}$, *ind*:$\mathbb{N}$

l :=↓

i = 1 .. n

s, c := összeg_darab(i)

c=0	c≠0 ∧ l	c≠0 ∧ ¬l
–	s/c > max	l,max,ind:= ↑, s/c, i
	max,ind:= s/c, i	–

segéd:

s:$\mathbb{N}$, *c*:$\mathbb{N}$, *i*:$\mathbb{N}$

s, c := összeg_darab(i)

paraméter:

s:$\mathbb{N}$, *c*:$\mathbb{N}$, *i*:$\mathbb{N}$

s, c := 0, 0

j = 1 .. m

napló[i,j] ≠ 0

s,c := s+ napló[i,j], c+1	–

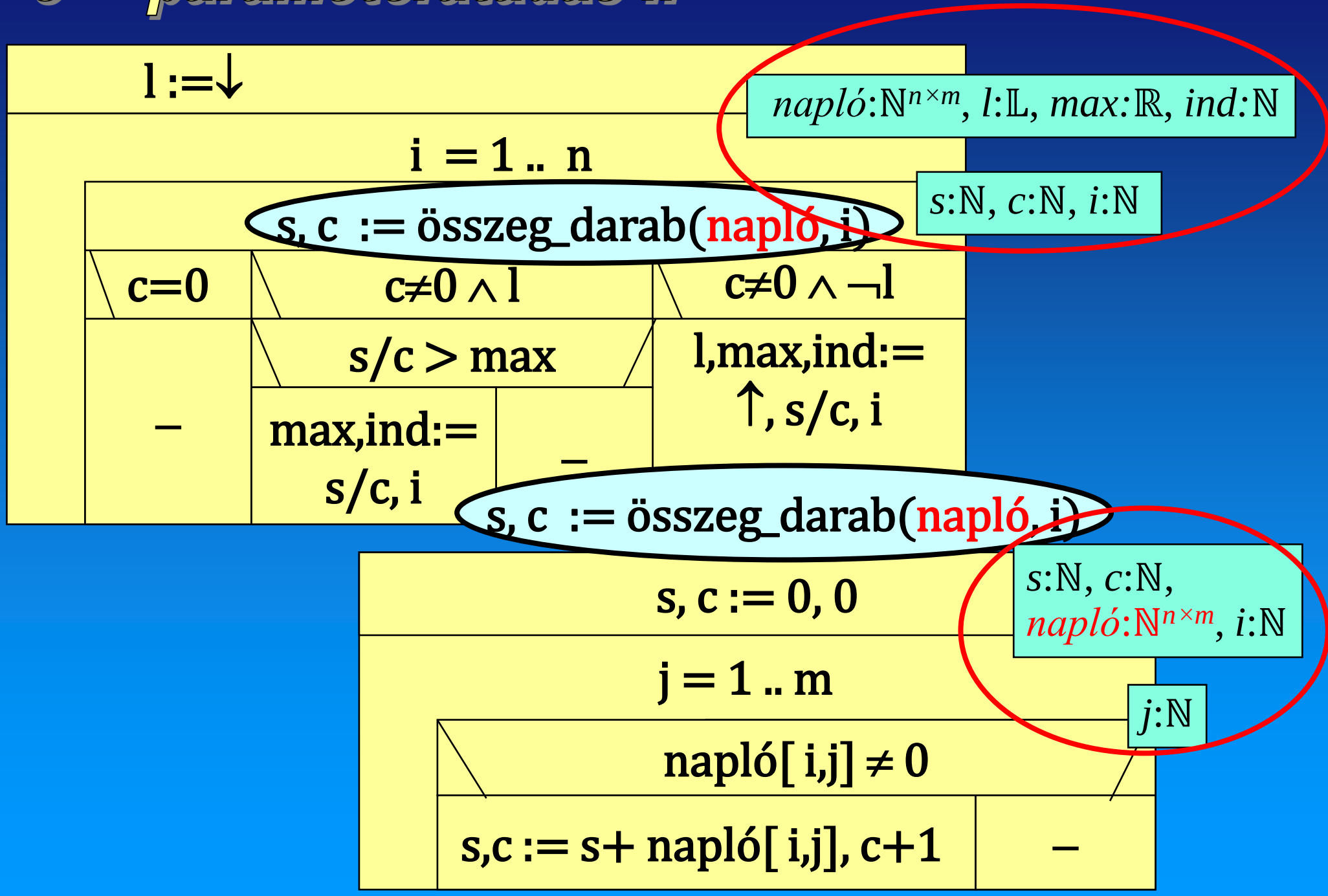
segéd:

j:$\mathbb{N}$

napló az alprogram ***globális változója*** amit csak olvasásra használ

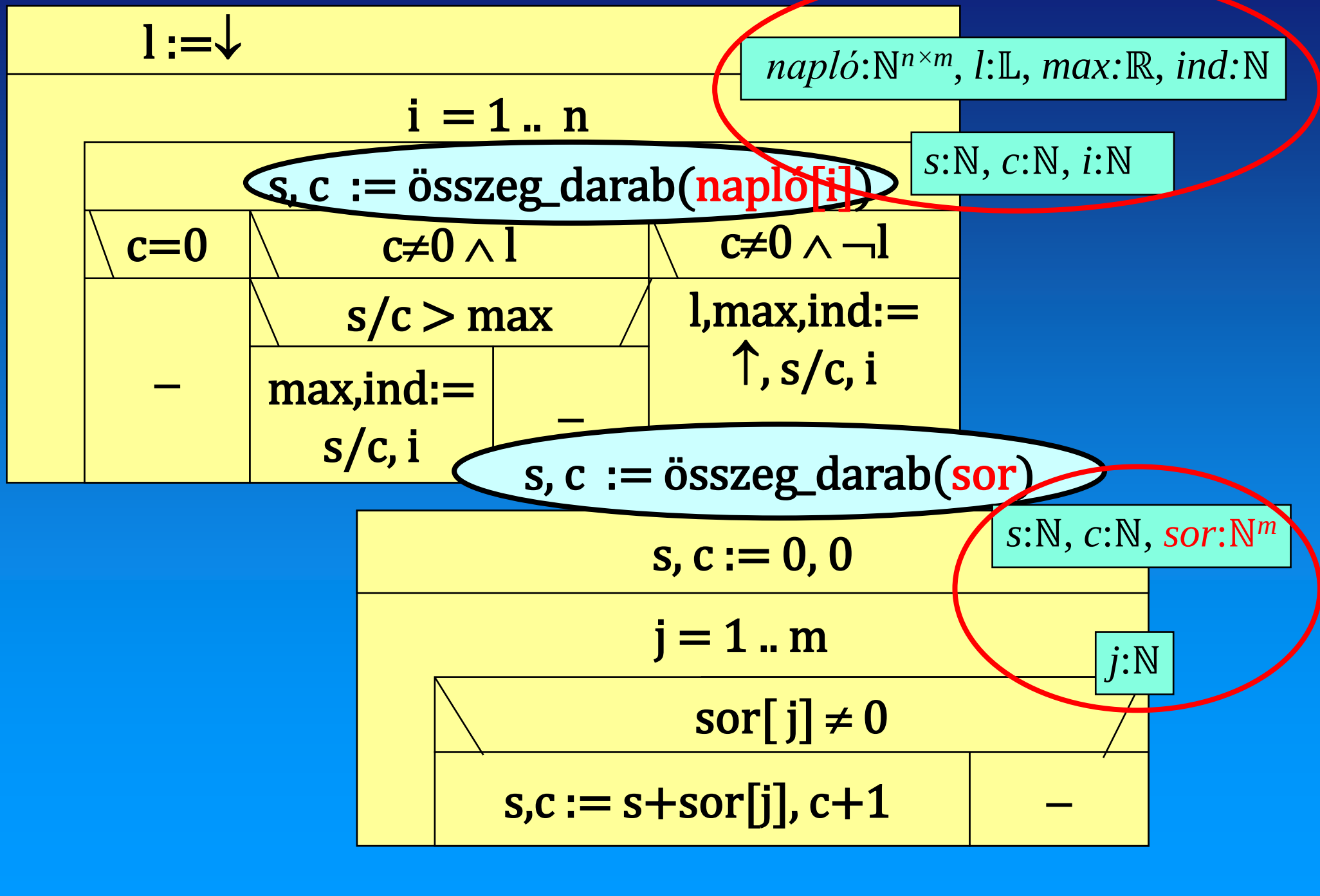
C++ paraméterátadás I.

Nem használunk globális változókat

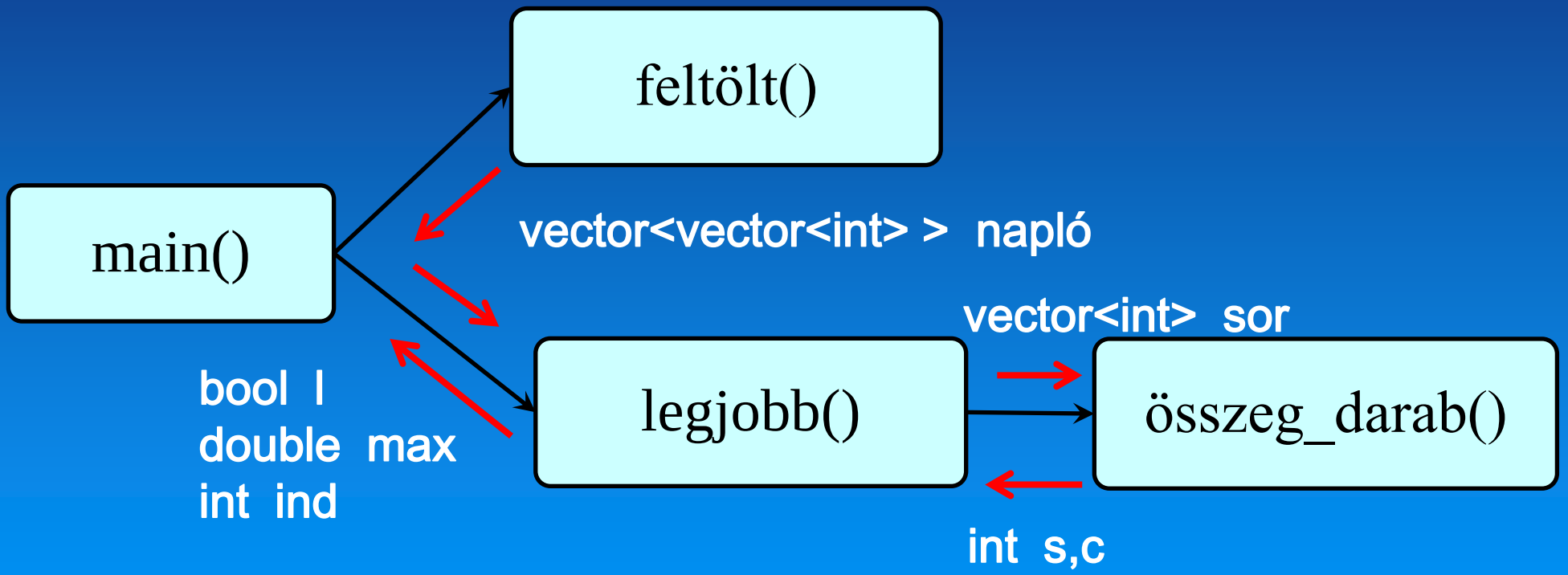


C++ paraméterátadás II.

Nem használunk globális változókat



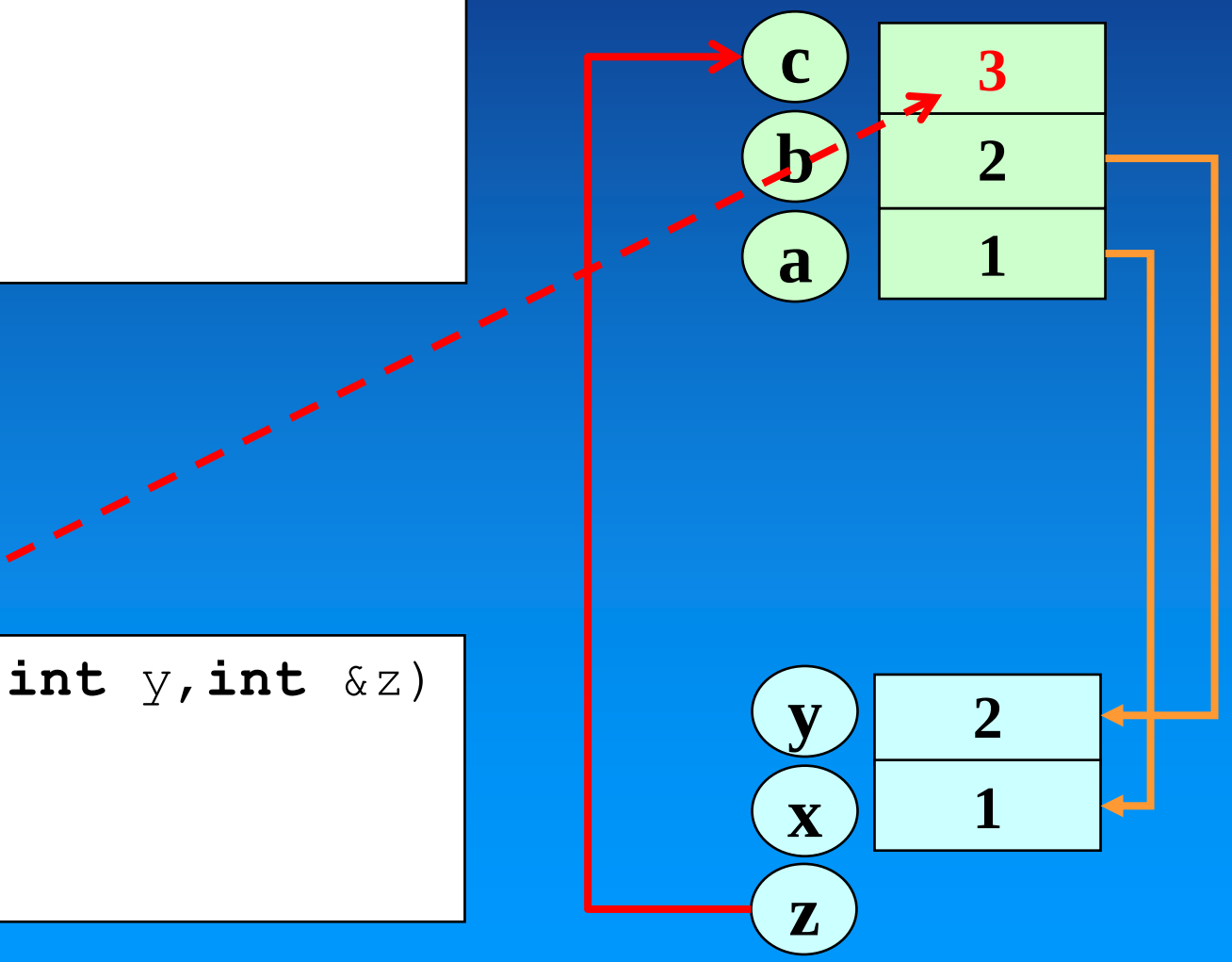
Modulszerkezet



Paraméter átadás C++ nyelvben

```
int a=1, b=2, c=0;
osszead(a,b,c);
```

```
void osszead(int x,int y,int &z)
{
     z=x+y;
}
```



Függvények közötti adatforgalom C++ nyelven

- Bemenő adat átadása
  - Érték szerint (hívás helyén állhat kifejezés)
    - `int n`
  - Konstans referencia szerinti (memória takarékos)
    - `const string &str, const vector<int> &sor`
- Eredmény visszaadása
  - Referencia szerint (hívás helyén NEM állhat kifejezés)
    - `int &n, vector<int> &sor`
  - Visszatérési értékkel (**`return`**)
    - lehet összetett (**`struct`**) is
- Globális változó
  - csak indokolt esetben

Alprogramok paraméterezése

```
void feltolt(vector<vector<int> > &naplo);
```

```
vector<vector<int> > feltolt();
```

```
void legjobb(const vector<vector<int> > &naplo,
             bool &l, double &max, int &ind)
```

```
bool legjobb(const vector<vector<int> > &naplo,
             double &max, int &ind);
```

```
void összeg_darab( const vector<int> &sor,
                   int &s, int &c);
```

```
struct Par{int s, int c};
Par összeg_darab(const vector<int> &sor);
```

```
Par e;
e.s … e.c
return e;
```

Program szerkezet

```
#include <iostream>
#include <vector>
using namespace std;

void feltolt(std::vector<std::vector<int> > &naplo);
bool legjobb(const vector<vector<int> > &naplo, double &max, int &ind);
void osszeg_darab(const vector<int> &sor, int &s,int &c);

int main()
{
    vector<vector<int> > naplo;
    int ind;
    double max;
    feltolt(naplo);
    if(legjobb(naplo,max,ind))
        cout << "Az egyik legjobb diák a " << ind+1
             << ". sorszámú, akinek az átlaga:" << max << endl;
    else cout << "Senki sem kapott osztályzatot.\n";
    return 0;
}
void feltolt(std::vector<std::vector<int> > &naplo){ … }
bool legjobb(const vector<vector<int> > &naplo,double &max,int &ind){ … }
void osszeg_darab(const vector<int> &sor,int &s,int &c){ … }
```

Csomagokra (komponensekre) bontás előnyei

- Csomagonkénti fordíthatóság

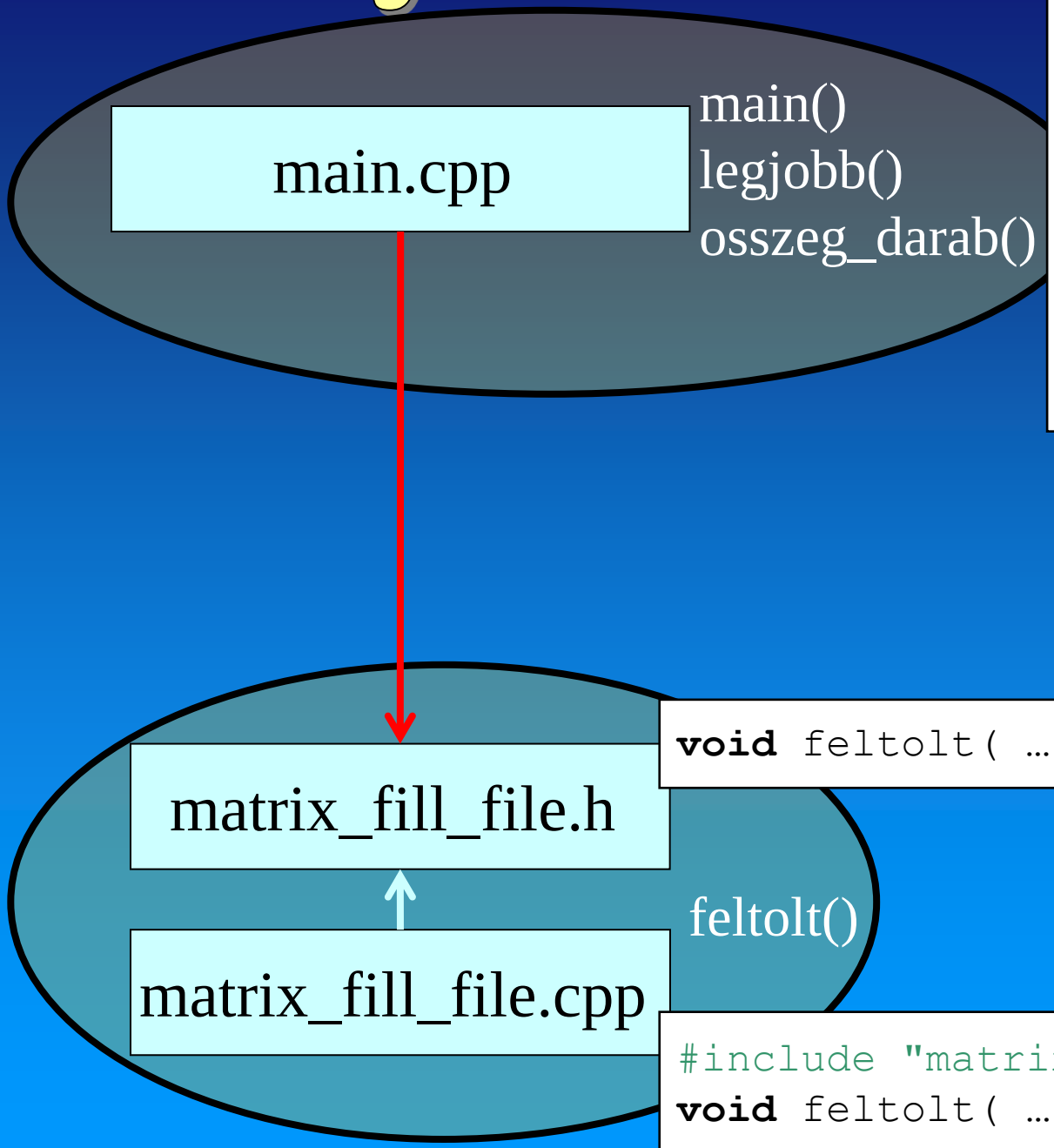
  Lehetővé teszi a csoportmunkában történő fejlesztést
- Újrafelhasználhatóság

  Más programokban közvetlenül (NEM copy/paste módon) felhasználhatóak
- Cserélhetőség

  Azonos célú csomagok közötti váltogatás
- Áttekinthetőség

  A program fejlesztése, tesztelése, javítása könnyebb

Csomag szerkezet I.




```cpp
#include <iostream>
#include <vector>
#include "matrix_fill_file.h"
using namespace std;

bool legjobb(…);
void osszeg_darab(…);

int main(){ … }
bool legjobb( … ){ … }
void osszeg_darab( … ){ … }
```

```cpp
void feltolt( … );
```

```cpp
#include "matrix_fill_file.h"
void feltolt( … ){ … }
```

Csomag szerkezet II.

main.cpp

main()
legjobb()
osszeg_daral

```
#include <iostream>
#include <vector>
#include "matrix_fill_keyboard.h"
using namespace std;

bool legjobb(…);
void osszeg_darab(…);

int main(){ … }
bool legjobb( … ){ … }
void osszeg_darab( … ){ … }
```

```
void feltolt( … );
```

matrix_fill_keyboard.h

feltolt()

```
#include "matrix_fill_keyboard.h"
#include "read_int.h"
bool jegy( … ) { … }
void feltolt( … ){ … }
```

matrix_fill_keyboard.cpp

read_int.h

```
int read_int( … );
```

read_int()

read_int.cpp

```
#include "read_int.h"
int read_int( … ){ … }
```

Csomag szerkezet III.