STRUCTURI DE DATE

Vector (Tablou)

Lector dr. Dorin IORDACHE

Agenda

01

Multime la Structura

02

Declarare tablou

03

Implemenatre

04

Operatii

DE LA MULTIME LA STRUCTURA

01

De la mulțime la structură

- Set de date: $A=\{a_1,a_2,\ldots,a_n\}$ (dacă elementele sunt distincte corespunde conceptului de mulțime din matematică)
- Structura de date: set de date + relații între elementele mulțimii
- Dpdv matematic:
  - O relație binară R este o submulțime a lui AxA; daca (a_i,a_j) aparține lui R spunem ca a_i se află în relația R cu a_j (se notează $a_i R a_j$)
  - O relație binară poate avea diferite proprietăți: reflexivitate (a_iRa_i), simetrie (a_iRa_j implică a_jRa_i), antisimetrie (a_iRa_j si a_jRa_i implica $a_i=a_j$), tranzitivitate (a_iRa_j, a_jRa_k implica a_iRa_k)
  - Cazuri particulare:
    - Relație de echivalență: reflexivă, simetrică și tranzitivă
    - Relație de ordine: reflexivă, antisimetrică și tranzitivă

De la mulțime la structură

- Set de date: $A=\{a_1,a_2,\ldots,a_n\}$
- Exemple de relatii binare
  - „succesor” (RS): a_i RS a_j daca a_i se află înaintea lui a_j
  - „predecesor” (RP): a_i RP a_j daca a_i se află după a_j
  - „succesor direct” (RSD): a_i RSD a_j daca a_i RS a_j si nu exista a_k astfel incat a_i RS a_k si a_k RS a_j
  - „predecesor direct” (RPD): a_i RPD a_j daca a_i RP a_j si nu exista a_k astfel incat a_i RP a_k si a_k RP a_j

- Structura liniară = set de date + relație de tip succesor sau predecesor (sau ambele) cu proprietatea că fiecare element are un unic succesor direct (și/sau predecesor direct)

- Cazul cel mai simplu: extinderea relatiilor „predecesor”, „succesor” corespunzătoare mulțimii indicilor asupra datelor din set

DECLARARE TABLOU

02

Tablou (unidimensional)

O listă de valori cu același tip de date care sunt stocate folosind un singur nume de grup.

Declarație generală a tabloului:

tipul de date nume-matrice[număr-de-articole];

Numărul de elemente trebuie specificat înainte de declarare:

```
const int SIZE = 100;
float arr[SIZE];
```

De la mulțime la structură

- Exemplu: M=(3,1,4,2)
- Variante de reprezentare a elementelor secvenței M (în Python)
  - folosind 5 variabile: a=1, b=2, c=3, d=4
  - folosind: x=[3,1,4,2]
- In care dintre cele două variante putem considera că este definită o structură de date?
- Este vorba de o structură liniară?

Vector (Tablou)

- **unidimensional**

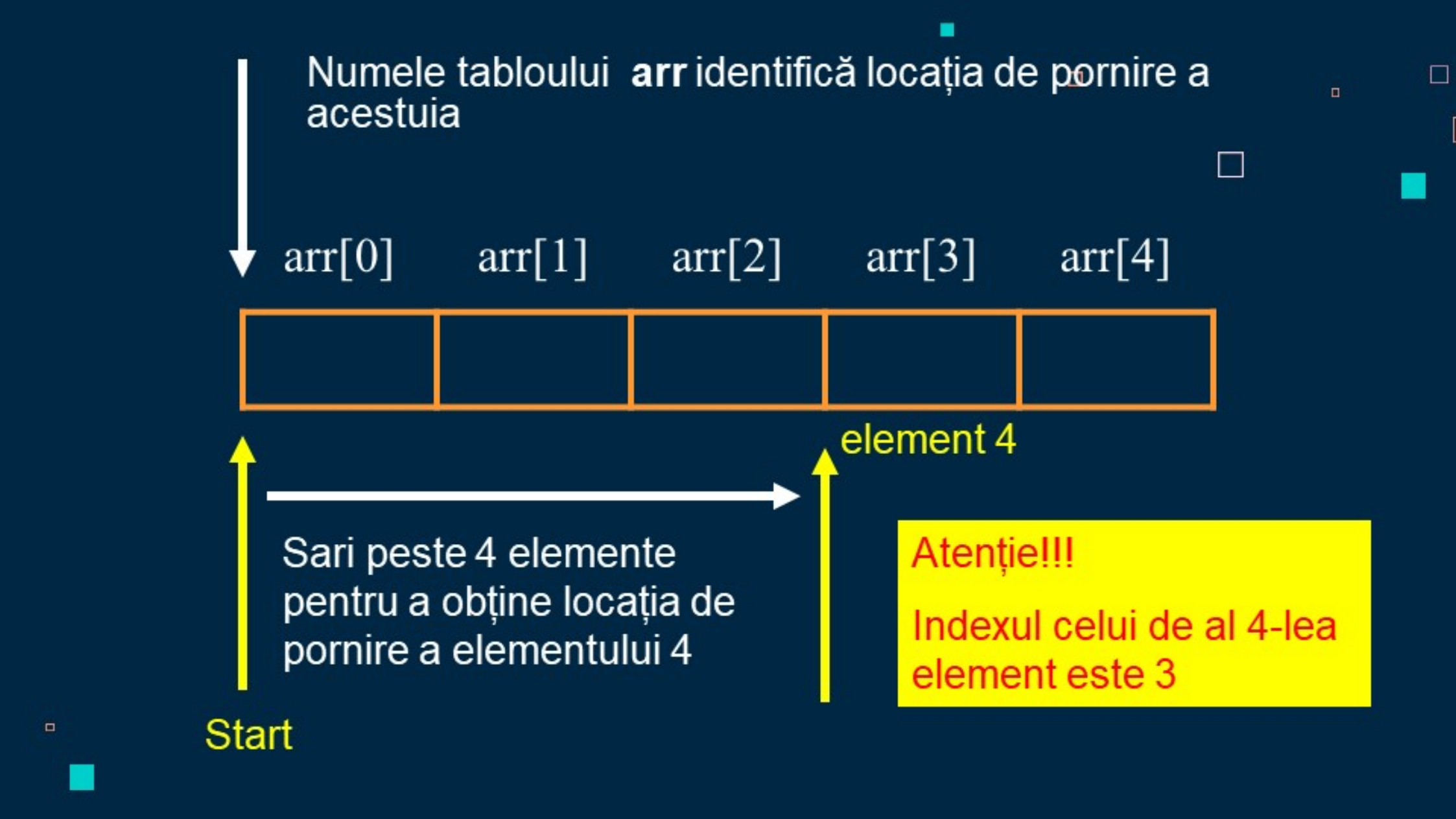
Elemente	5	8	10	1	9
Index	0	1	2	3	4

Tablou (unidimensional)

Elementele individuale ale tabloului pot fi accesate prin specificarea numelui acestuia și a indexului elementului:

arr[3]

Atenție:
indicii au valori de la 0 la numărul de elemente -1!!


Numele tabloului **arr** identifică locația de pornire a acestuia
arr[0]
arr[1]
arr[2]
arr[3]
arr[4]
element 4
Sari peste 4 elemente pentru a obține locația de pornire a elementului 4
Start
Atenție!!!
Indexul celui de al 4-lea element este 3

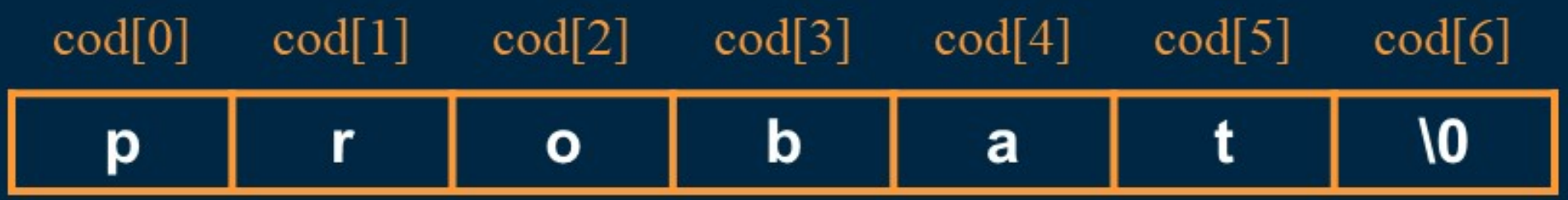
Initializarea tabloului

Tablourile pot fi inițializate în timpul declarației lor

int arr[5] = {98, 87, 92, 79, 85};
int arr[5] = {98, 87} - ce se întâmplă în acest caz??

Care este diferența dintre următoarele două declarații?

char cod [] = {‘p', ‘r', ‘o', ‘b', ‘a’,’t’};
char cod[] = “probat”;



Tablou (unidimensional)

Tablourile pot conține orice tip de valoare (fie valori primitive, fie referințe)

- Indicii (index) sunt folositi pentru a accesa valori specific

- Exemple:

counts[0] // prima variabilă în counts
counts[1] // a doua variabilă în counts
counts[9] // ultima variabilă în counts
counts[10] // eroare – se încearcă accesul // variabilă în afara numărului

Tablou de siruri de caractere

```
final String[ ] ZI = {
                    "Luni", "Marți", "Miercuri", "Joi", "Vineri", "Sâmbătă",
                    "Duminică"
                    };
```

Analizati declararea de mai sus. Este corecta?

IMPLEMENTARE

03

Implementarea structurii liniare

Variante de implementare:

- Folosind o zonă contiguă de memorie -> tablouri
- Folosind zone disparate „înlănțuite” prin specificarea unor informații de tip referință (pointer) -> liste înlănțuite

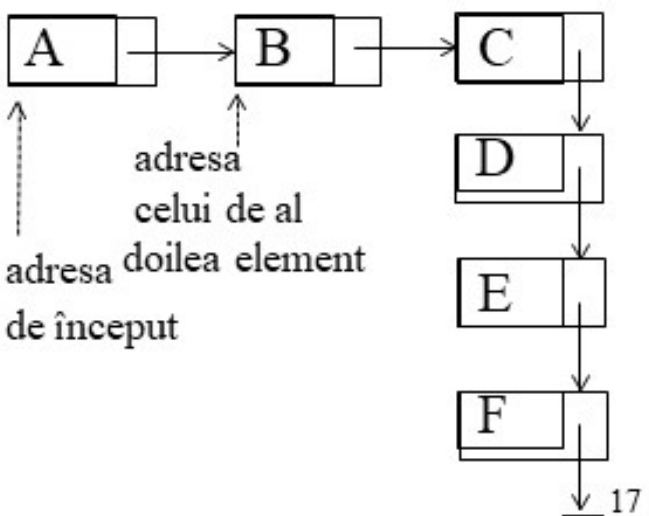
Tablou

adresa unui element =
adresa de început + deplasament

adresa
de început

Obs:
deplasament = 0 pt primul element
= 1 pt al doilea
etc.

Structură înlănțuită



Implementarea structurii liniare

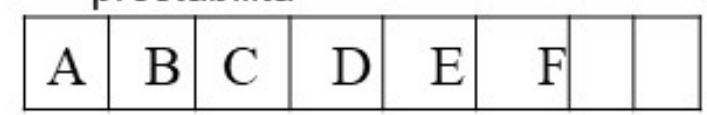
Tablou

Avantaje:

- Acces rapid pe baza indexului
- Se stochează doar valorile elementelor

Dezavantaje:

- Dimensiunea maximă este prestabilită



adresa de început

adresa unui element =
adresa de început + deplasament
(corelat cu valoarea indexului)
Obs:
deplasament = 0 pt primul element
= 1 pt al doilea
etc.

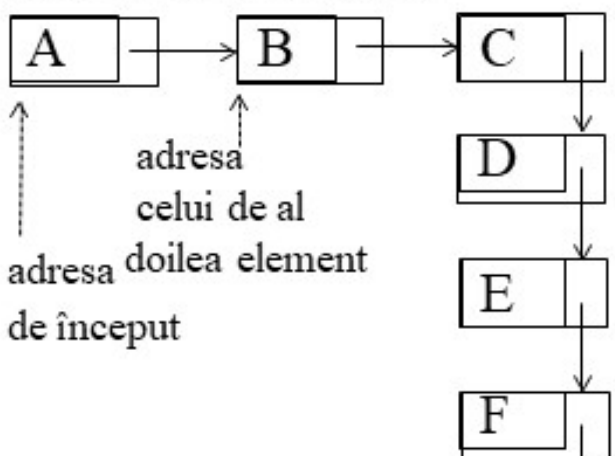
Structură (listă) înlănțuită

Avantaje:

- Dimensiune flexibilă
- Cost mic la inserare/ eliminare

Dezavantaje:

- Necesită stocarea unor informații adiționale (ex: adresa elem. următor)
- Accesul aleator nu este facil



Implementare utilizând tablouri

Tablou: x[1..nmax] - zona maximă alocată pentru stocarea structurii
n – numărul efectiv de elemente

Structura: s are două componente (câmpuri):
s.x (tabloul ce conține valorile)
s.n (număr efectiv de elemente)

Complexitatea operațiilor:

- Interogare după poziție
  - Elementul aflat pe o anumită poziție:
    s.x[i] cost:$\Theta(1)$
  - Elementul următor/ anterior unui element specificat (element curent):
    s.x[i-1] sau s.x[i+1] cost: $\Theta(1)$

Implementare utilizând tablouri

Tablou: x[1..nmax] - zona maximă alocată pentru stocarea structurii
n – numărul efectiv de elemente

Structura: s are două componente (câmpuri):
s.x (tabloul ce conține valorile)
s.n (numărul elementelor)

Complexitatea operațiilor: Interogare după valoare

- Elementul care conține o valoare specificată
  căutare secvențială cost:O(n)
- Elementul care conține cea mai mică/ mare valoare
  determinare minim/maxim cost:$\Theta(n)$
- Elementul care conține o valoare specificată prin poziția relativă în raport cu alte valori (ex: al treilea element în ordine crescătoare, elementul median etc)
  selecția celui de al k-lea element în ordine crescătoare/ descrescătoare:
  - varianta bazată pe sortare parțială prin metoda selecției: $\Theta(kn)$
  - varianta bazată pe ideea de la quicksort : O(n) în medie

Implementare utilizând tablouri

Tablou: x[1..nmax] - zona maximă alocată pentru stocarea structurii
n – numărul efectiv de elemente

Structura: s are două componente (câmpuri):
s.x (tabloul ce conține valorile)
s.n (numărul elementelor)

Inserarea unui element

- La început (cost: $\Theta(n)$)
  - Necesită deplasarea tuturor elementelor, începând de la ultimul, cu o poziție la dreapta
- La sfârșit (cost:$\Theta(1)$)
  - s.n=s.n+1; s.x[s.n]=e
- Pe poziția i (cost:O(n))
  - Necesită deplasarea elementelor cu indicii n, n-1,..., i cu o poziție la dreapta

Implementare utilizând tablouri

Tablou: x[1..nmax] - zona maximă alocată pentru stocarea structurii
n – numărul efectiv de elemente

Structura: s are două componente (câmpuri):
s.x (tabloul ce conține valorile)
s.n (numărul elementelor)

Eliminarea (ștergerea, suprimarea) unui element

- De la început (cost:$\Theta(n)$)
  - Deplasarea tuturor elementelor, începând cu primul, cu o poziție la stânga
- De la sfârșit (cost:$\Theta(1)$)
  - s.n=s.n-1
- De pe poziția i (cost:O(n))
  - Deplasarea tuturor elementelor având indicii (i+1), (i+2),..., n cu o poziție la stânga

OPERATII

04

Operatii

Parcurgerea elementelor

Fie A un tablou cu limita inferioară LB și limita superioară UB. Acest algoritm parcurge tabloul A și aplică operația PROCESS fiecărui element al acestuia.

```
1.    Repeat For I = LB to UB
2.          Apply PROCESS to A[I]
      [End of For Loop]
3.    Exit
```

Operatii

Inserare intr-un tablou neordonat

```
1.    Set I = N
2.    Repeat While (I >= LOC)
3.          Set A[I+1] = A[I]
4.          Set I = I - 1
      [End of While Loop]
5.    Set A[LOC] = ITEM
6.    Set N = N + 1
7.    Exit
```

[Initializare contor]

[Mutare element inapoi]

[Decrementare contor]

[Inserare element]

[Actualizare numar elemente in tablou]

Operatii

Inserare intr-un tablou ordonat

```
1.    Set I = N
2.    Repeat While (ITEM < A[I]) and (I >= 1)
3.          Set A[I+1] = A[I]
4.          Set I = I - 1
      [End of While Loop]
5.    Set A[I+1] = ITEM
6.    Set N = N + 1
7.    Exit
```

[Initializare contor]

[Mutare element inapoi]

[Decrementare contor]

[Inserare element]

[Actualizare numar elemente in tablou]

Operatii

Stergere element dintr-un tablou

```
1.    Set ITEM = A[LOC]
2.    Repeat For I = LOC to N
3.          Set A[I] = A[I+1]
      [End of For Loop]
4.    Set N = N - 1
5.    Exit
```

[Setare element de sters]

[Mutare element urmator inapoi]

[Actualizare numar elemente in tablou]

Operatii

Concatenare 2 tablouri neordonate

```
1.    Repeat For I = 1 to M
2.          Set C[I] = A[I]
      [End of For Loop]
3.    Set K = 1
4.    Repeat For J = M+1 to M+N
5.          Set C[J] = B[K]
6.          Set K = K + 1
      [End of For Loop]
7.    Exit
```

[Copiere elemente A in C]

[Initializare contor K]

[Copiere elemente B in C]

[Rezultatul concatenarii este C]

Intrebari?

dorin.lordache@365.univ-ovidius.ro

Mulțumesc