

Urmărim o prezentare generală a structurilor de date, independentă de un limbaj de programare sau altul - pentru descrierea *algoritmilor* vom folosi limbajul *Pseudocod*.

Algoritmi

- Domeniile SD și al algoritmilor (de manipulare a acestor structuri) se interconectează.
- Aspecte de bază legate de algoritmi - eficiența algoritmilor
 - cantitatea de resurse utilizate
 - * timp
 - * spațiu
 - măsurarea eficienței:
 - * analiză asimptotică (complexitate timp și spațiu)
 - * analiza empirică

Limbajul Pseudocod

- Vom folosi două tipuri de propoziții pseudocod:
 1. propoziții standard, fiecare având sintaxa și semantica sa;
 2. propoziții nestandard (texte care descriu părți ale algoritmului încă incomplet elaborate). Aceste propoziții convenim să înceapă cu semnul '@'.
- Comentariile vor fi cuprinse între acolade.
- citirea datelor se face folosind propoziția standard:
citeste *lista*
- tipărirea rezultatelor se face folosind propoziția standard:
tiparește *lista*
- Atribuirea se va simboliza prin \leftarrow .
- Instrucțiunea alternativă va avea forma:

```

Daca expresie logica atunci
    instructiuni
altfel
    instructiuni
SfDaca

```

unde secțiunea **altfel** poate lipsi.

- Structura repetitivă cu număr cunoscut de pași:

```

Pentru contor = li, lf, pas executa
    instructiuni
SfPentru

```

unde contorul ia valori de la valoarea inițială *li*, la valoarea finală *lf*, la fiecare pas adăugându-se valoarea *pas*. Pasul poate lipsi fiind implicit egal cu 1.

- Structura repetitivă cu număr necunoscut de pași condiționată anterior (test inițial):

```

CatTimp expresie logica executa
    instructiuni
SfCatTimp

```

- Structura repetitivă cu număr necunoscut de pași condiționată posterior (test final):

```

Repeta
    instructiuni
PanaCand expresie logica

```

- Definirea unui subalgoritm se va face folosind propoziția standard:

```

Subalgoritm nume(...)
    instructiuni
SfSubalgoritm

```

- Definirea unei funcții se va face folosind propoziția standard:

```

Functia nume(...)
    instructiuni
SfFunctia

```

- Pentru a specifica rezultatul întors de o funcție vom folosi numele funcției.

Exemplu:

```

Functia minim(a,b)
    min ← a;
    Daca a < b atunci
        min ← b;
    SfDaca
    minim ← min;
SfFunctia

```

- Apelul unei proceduri se face folosind:

```

nume(< lista_parametri_actuali >)

```

- apelul unei funcții se face scriind într-o expresie numele funcției urmat de lista parametrilor actuali (ex: $m \leftarrow \text{minim}(2, 3)$).

Extensii și convenții

- Dacă vrem să declarăm o variabilă i de tip *Intreg*, atunci vom folosi notația $i : \text{Intreg}$.
- În cazul în care dorim să declarăm un tablou unidimensional t cu elemente de tip *TElement* vom folosi notația $t : \text{TElement}[]$.
 - Dacă se dorește precizarea exactă a limitelor de variație a unui indice, vom folosi notația care se bazează pe tipul subdomeniu:
 $\text{TElement}[\text{MIN}..\text{MAX}]$
- O înregistrare (un vector având lungimea n și elementele de tip *TElement*) o vom reprezenta sub forma

```

Vector
  n: Intreg
  e: TE[]

```

- **Accesul** la elementele unei înregistrări îl vom face folosind caracterul “.”
 - * Dacă ne referim la o variabilă v de tip *Vector*, atunci prin:
 - $v.n$ - ne vom referi la numărul de elemente ale vectorului;
 - $v.e[i]$ - ne vom referi la al i -lea element din vector.
- Pentru a indica pointeri (adrese ale unor zone de memorie), vom folosi caracterul \uparrow , cu alte cuvinte dacă vrem să declarăm un pointer p care referă un număr întreg, acest lucru îl vom scrie în următoarea manieră:

$p : \uparrow \text{Intreg}$

Conținutul locației referite de pointerul p îl vom nota $[p]$.

- Pointerul nul (care nu referă nimic) îl vom nota prin NIL.
- Operațiile de alocare, respectiv dealocare a pointerilor le vom nota:
 - $\text{aloca}(p)$
 - $\text{dealoca}(p)$

Convenții folosite în specificații.

- Specificarea unei operații se va face prin:
 - 1 **pre:** - date și precondiții
 - 2 **post:** - rezultate și postcondiții

[3] @ - (opțional) excepții aruncate

- În specificarea operațiilor prin precondiții și postcondiții, când folosim numele unei variabile ne referim la valoarea acesteia.
- Având o variabilă i de tip $Tip(i : Tip)$, notația $i \in Tip$ (exemplu: $i \in Intreg$), va fi folosită pentru a evidenția faptul că valoarea variabilei aparține domeniului de definiție a tipului $Tip(Intreg)$.
- Datorită faptului că valorile variabilelor pot fi modificate în urma executării unei operații, este necesară delimitarea dintre valoarea variabilei înainte de efectuarea operației și cea de după execuția ei. Vom conveni să folosim caracterul ' (apostrof) pentru a specifica valoarea variabilei după aplicarea operației.
 - De exemplu, având o operație **dec** care decrementează valoarea unei variabile x ($x : Intreg$), specificația operației va fi:

$dec(x)$
 $pre : x \in Integer$
 $post : x' = x - 1$

Tip de date generic

Pentru generalitate, vom considera că elementele unui TAD sunt de un tip de date generic **TElement** cu o interfață minimală formată din următoarele operații:

- atribuire
 $atribuie(x, y)$ - notație $x \leftarrow y$
 $pre : x, y \in TElement$
 $post : x = y$
- testarea egalității
 $egal(x, y)$ - notație $x = y$
 $pre : x, y \in TElement$
 $post : egal = \begin{cases} adevarat, & x = y \\ fals, & x \neq y \end{cases}$

Pentru simplitate, vom folosi notațiile

- “ $x=y$ ” în locul apelului funcției “ $egal(x,y)$ ” ’ pentru a ilustra egalitatea a două elemente de tip **TElement**.
- “ $x \leftarrow y$ ” în locul apelului subalgoritmului “ $atribuie(x,y)$ ” ’ pentru a ilustra operația de atribuire.

Dacă pe domeniul valorilor unui tip de date se poate defini o relație de ordine ($'\leq'$), vom defini și tipul generic ***TComparabil***, care derivă din tipul *TElement*; pe lângă interfața acestuia, *TComparabil* admite și următoarea operații:

- compararea a două elemente

compară(x, y)

pre : $x, y \in TComparabil$

$$post : compara = \begin{cases} -1, & \text{dacă } x < y \\ 0, & \text{dacă } x = y \\ 1, & \text{dacă } y > x \end{cases}$$

Pentru simplitate, vom folosi notațiile “ $x < y$ ”, “ $x \leq y$ ”, “ $x > y$ ”, “ $x \geq y$ ” pentru a ilustra relațiile corespunzătoare între elemente de tip ***TComparabil***.