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 şi al algoritmilor (de manipulare a acestor structuri) se interconectează.
- Aspecte de bază legate de algoritmi - eficienţa algoritmilor
  - cantitatea de resurse utilizate
    - timp
    - spaţiu
  - măsurarea eficienţei:
    - analiză asimptotică (complexitate timp şi spaţiu)
    - analiza empirică

Limbajul Pseudocod

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

```
Daca expresie logica atunci
  instructiuni
altfel
  instructiuni
SfDaca
```

unde secţiunea `altfel` poate lipsi.

- Structura repetitivă cu număr cunoscut de paşi:

```
Pentru  contor = li, lf, pas  executa
  instructiuni
SfPentru
```

unde contorul ia valori de la valoarea iniţială 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 paşi condiţionată anterior (test iniţial):

```
CatTimp expresie_logica executa
  instructiuni
SfCatTimp
```

- Structura repetitivă cu număr necunoscut de paşi condiţionată posterior (test final):

```
Repeta
  instructiuni
PanaCand expresie logica
```

- Definirea unui subalgoritm se va face folosind propoziţia standard:

```
Subalgoritm nume(...)
  instructiuni
SfSubalgoritm
```

- Definirea unei funcţii se va face folosind propoziţia standard:

```
Functia nume(...)
  instructiuni
SfFunctia
```

- Pentru a specifica rezultatul întors de o funcţie vom folosi numele funcţiei.

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 funcţii se face scriind într-o expresie numele funcţiei urmat de lista parametrilor actuali (ex: $m \leftarrow minim(2, 3)$).

Extensii şi convenţii

- Dacă vrem să declarăm o variabilă i de tip $Intreg$, atunci vom folosi notaţia $i : Intreg$.
- În cazul în care dorim să declarăm un tablou unidimensional t cu elemente de tip $TElement$ vom folosi notaţia $t : TElement[]$.
  - Dacă se doreşte precizarea exactă a limitelor de variaţie a unui indice, vom folosi notaţia care se bazează pe tipul subdomeniu:
    $TElement[MIN..MAX]$
- O înregistrare (un vector având lungimea n şi 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:

$$p : \uparrow Intreg$$

  Conţinutul locaţiei referite de pointerul p îl vom nota $[p]$.
- Pointerul nul (care nu referă nimic) îl vom nota prin NIL.
- Operaţiile de alocare, respectiv dealocare a pointerilor le vom nota:
  - $aloca(p)$
  - $dealoca(p)$

Convenţii folosite în specificaţii.

- Specificarea unei operatii se va face prin:

  1 **pre:** - date şi precondiţii
  2 **post:** - rezultate şi postcondiţii

[3] @ - (opţional) excepţii aruncate

- În specificarea operaţiilor prin precondiţii şi postcondiţii, când folosim numele unei variabile ne referim la valoarea acesteia.
- Având o variabilă i de tip Tip ($i : Tip$), notaţia $i \in Tip$ (exemplu: $i \in Intreg$), va fi folosită pentru a evindenţia faptul că valoarea variabilei aparţine domeniului de definiţie a tipului Tip ($Intreg$).
- Datorită faptului că valorile variabilelor pot fi modificate în urma executării unei operaţii, este necesară delimitarea dintre valoarea variabilei înainte de efectuarea operaţiei şi cea de după execuţia ei. Vom conveni să folosim caracterul $'$ (apostrof) pentru a specifica valoarea variabilei după aplicarea operaţiei.
  - De exemplu, având o operaţie dec care decrementează valoarea unei variabile x ($x : Intreg$), specificaţia operaţiei 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 interfaţă minimală formată din următoarele operaţii:

- atribuire
  atribuie(x, y) - notaţie $x \leftarrow y$
  $pre : x, y \in TElement$
  $post : x = y$

- testarea egalităţii
  egal(x, y) - notaţie $x = y$
  $pre : x, y \in TElement$
  $post : egal = \begin{cases} adevarat, & x = y \\ fals, & x \neq y \end{cases}$

Pentru simplitate, vom folosi notaţiile

- "x=y" în locul apelului funcţiei "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 operaţia de atribuire.

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

- 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 notaţiile "x<y", "x $\leq$ y", "x>y", "x $\geq$ y" pentru a ilustra relaţiile corespunzătoare între elemente de tip ***TComparabil***.