RECAPITULARE

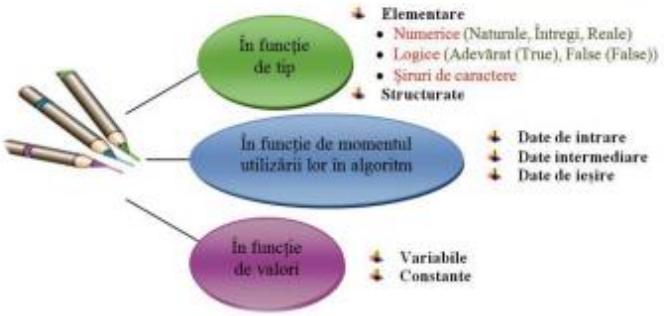
Să ne reamintim!

Algoritmul este o succesiune finită de pași (instrucțiuni), realizați într-o ordine bine definită, pentru ca, pornind de la anumite date cunoscute, numite ***date de intrare***, să obținem rezultatele dorite, numite ***date de ieșire***.

Proprietățile algoritmului sunt ***finititudinea*** (furnizarea rezultatelor după un număr finit de pași), ***claritatea*** (precis, fără ambiguități), ***generalitatea*** (acoperă o clasă generală de probleme, nu una particulară), ***corectitudinea*** (furnizează soluții corecte), ***unicitatea*** – (furnizează soluții unice pentru aceleași date de intrare, indiferent de numărul de efectuări ale algoritmului), ***eficiența*** (numărul de pași parcurși este cât se poate de mic), ***optimalitatea*** (algoritmul urmează calea directă de rezolvare), ***verificabilitatea*** (fiecare pas poate fi verificat), ***completitudinea*** (sunt tratate și cazurile particulare ale unei probleme generale).

Datele sunt obiectele cu care lucrează orice algoritm. Datele sunt caracterizate prin: ***nume***, ***tip*** și ***valoare***.

Clasificarea datelor



Expresia este un enunț alcătuit din unul sau mai mulți ***operanzi*** legați între ei prin ***operatori***.

Clasificarea operatorilor:

- ***De atribuire*** – utilizat pentru stabilirea valorii pentru o anumită variabilă sau constantă (←);
- ***Aritmetici*** – +, –, *, / (DIV – câtul împărțirii), % (MOD – restul împărțirii);
- ***Relaționali*** – <, >, <= (mai mic sau egal), >= (mai mare sau egal), =, <> (diferit);
- ***Logici*** – AND (și – conjucția), OR (sau – disjuncția), NOT (negația)

Tipuri de expresii:

- ***aritmetice*** – au ca rezultat o valoare de tip numeric;
- ***logice*** – au ca rezultat o valoare de tip logic.

În realizarea unui algoritm se utilizează următoarele ***structuri***:

Structura secvențială (liniară)
- Declararea variabilelor sau a constantelor

  Se specifică pentru datele problemei numele, tipul de date și valoarea primită
- Operația de citire

  Se preiau succesiv valori și se asociază, în ordinea preluării, variabilele specificate
- Operația de scriere

  Permite vizualizarea rezultatelor obținute în urma prelucrării datelor
- Operația de atribuire

  Presupune atribuirea valorii unei variabile din cadrul algoritmului, valoare obținută eventual în urma evaluării unei expresii.

Structura decizională (alternativă)
- Folosită în cazul în care pasul următor al unui algoritm depinde de anumite condiții
- Sintaxa (forma)

```
DACĂ condiție ATUNCI
   Instrucțiuni 1
ALTFEL
   Instrucțiuni 2
```

Structura repetitivă
- **Cu număr necunoscut de pași**

  condiționată anterior (cu test inițial)

  condiționată posterior (cu test final)
- **Cu număr cunoscut de pași (cu contor)**

ETAPE DE REZOLVARE A UNUI EXERCIȚIU ALGORITMIC

Pentru a putea realiza un algoritm, reprezentat prin schemă logică sau orice altă formă, trebuie să ținem cont de o anumită ordine a etapelor.

- **Analiza problemei**
  - Presupune determinarea datelor de intrare, de ieșire și a metodei de rezolvare.
- **Elaborarea algoritmului de rezolvare**
  - Presupune scrierea algoritmului folosind schema logică sau limbajul pseudocod.
- **Transcrierea algoritmului într-un limbaj de programare**
  - Presupune transcrierea din schemă logică sau limbaj pseudocod într-un limbaj de programare.
- **Testarea programului**
  - Presupune determinarea erorilor de sintaxă sau de algoritm, după testarea mai multor seturi de date.
- **Depanarea programului**
  - Presupune corectarea erorilor determinate în etapa anterioară.

1. Schema logică

D ***Schema logică*** este o modalitate de reprezentare a algoritmilor sub formă grafică prin utilizarea unor blocuri legate între ele prin săgeți.

O schemă reprezintă grafic toți pașii parcurși în realizarea unui algoritm.

D **Blocurile** sunt elementele grafice prin care se reprezintă fiecare operație din cadrul algoritmului. Acestea se parcurg liniar, pornind de la primul bloc și terminând cu ultimul. Blocurile se dezvoltă arborescent, fiind legate între ele prin săgeți direcționale care indică direcția de la un bloc la altul.

CLASIFICAREA BLOCURILOR

1. **Blocul START**

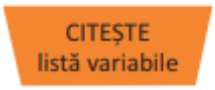
- este un bloc unic în cadrul unei scheme logice
- acesta reprezintă punctul de pornire al schemei
- se reprezintă grafic printr-o formă ovală
- în formă este trecut cuvântul **START**

2. **Blocul STOP**

- este un bloc unic în cadrul schemei logice
- el reprezintă punctul final al schemei
- este necesar ca după un număr finit de pași să se ajungă la acest bloc
- se reprezintă tot printr-o formă ovală
- în formă este trecut cuvântul **STOP**

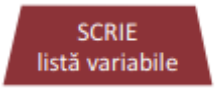
3. **Blocul de CITIRE**

- mai poartă denumirea și de bloc de **intrare**
- se utilizează pentru a reprezenta operația de citire a datelor
- se reprezintă printr-un trapez cu baza mare sus sau printr-un paralelogram
- conține textul ***CITEȘTE*** *listă variabile,* unde *listă variabile* se înlocuiește cu numele datelor citite, separate prin virgulă.
- tipuri de variabile care pot fi citite sunt: ***numerice*** (naturale, întregi, reale) și șiruri de caractere



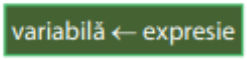
4. **Blocul de SCRIERE**

- mai poartă denumirea și de bloc de **ieșire**
- se utilizează pentru a reprezenta grafic operația de scriere a datelor
- se reprezintă printr-un trapez cu baza mare jos sau printr-un paralelogram
- operația utilizată este ***SCRIE*** *listă variabile*, unde *lista variabile* se înlocuiește cu denumirile variabilelor care vor fi afișate.



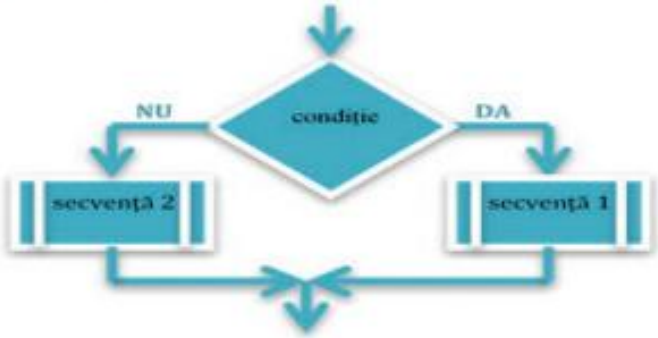
5. **Blocul de ATRIBUIRE**

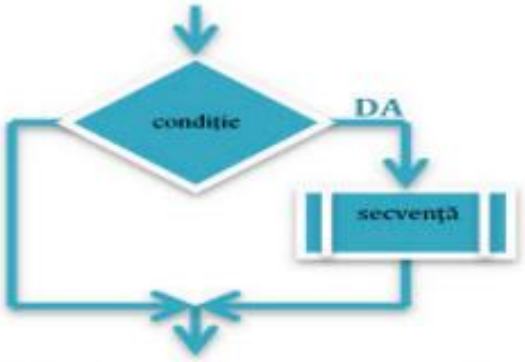
- permite **atribuirea** de valori datelor din algoritm
- se reprezintă printr-un dreptunghi care conține atribuirile propriu-zise
- valorile anterioare ale acelor variabile se vor pierde



6. **Blocul de DECIZIE**

- se utilizează pentru implementarea structurilor decizionale și repetitive (structura repetitivă se va studia în lecțiile următoare)
- se reprezintă printr-un romb
- din acest bloc pot ieși două săgeți corespunzătoare celor două ramificări ale structurii decizionale
- algoritmul se execută **doar pe una** din cele două ramuri ale sale: fie pe ramura **DA**, fie pe ramura **NU**
- **decizia** se realizează pe baza unei **condiții** aflate în cadrul rombului, iar în urma evaluării ei, se stabilește calea de urmat în cadrul schemei logice
- **condiția** este reprezentată printr-o expresie de tip logic
- există două tipuri de reprezentare a acestui bloc, deoarece în anumite cazuri una din cele două ramuri poate să lipsească





D ***Secvența*** este o notație abstractă pentru blocurile care urmează să fie executate pe acea ramură a structurii.

Conectorii sunt elemente grafice de legătură care se utilizează în diferite situații:

- Când se dorește continuarea pe aceeași pagină a schemei logice, dar în partea alăturată, se utilizează ***conectorii în pagină***, reprezentați prin cerculețe cu numere în ele, la bază în partea stângă și la început în partea dreaptă.

- Când se dorește continuarea schemei pe una sau mai multe pagini se folosesc ***conectorii între pagini***, reprezentați prin săgeți pline cu numere în ele. Acestea apar și la baza paginii și la începutul schemei din pagina următoare.

2. Limbajul pseudocod

Limbajul pseudocod este o modalitate de reprezentare a algoritmului, independent de un anumit limbaj de programare. Acesta respectă reguli de scriere și o anumită sintaxă în descrierea operațiilor ce îl compun.

În descrierea algoritmilor prin limbaj pseudocod noțiunea de ***operație*** utilizată în cadrul schemei logice, ce va fi înlocuită cu noțiunea de ***instrucțiune***. În realizarea algoritmului cu ajutorul limbajului pseudocod se utilizează aceleași structuri prezentate anterior: structura ***secvențială***, structura ***decizională*** și structura ***repetitivă***. Noțiunile *utilizate* până acum, date, variabile, constante, operatori, expresii, toate rămân valabile și pentru limbajul pseudocod.

Orice algoritm începe cu cuvântul *algoritm*, urmat de definirea acestuia în funcție de cerințe, și se termină cu notația *sfârșit algoritm*.

Exemplu:

algoritm sumă
.....
sfârșit algoritm

Structura ***secvențială*** conține următoarele instrucțiuni:

– ***Declararea variabilelor sau a constantelor*** – se realizează la începutul algoritmului și presupune specificarea pentru datele problemei a numelui, a tipului de date și a valorii primite pentru o constantă. Sintaxa instrucțiunii este:

Tipul <listă variabile>

Exemplu:
întreg *a, b, c* – s-au declarat trei variabile de tip întreg *a, b* și *c*
real *m* – s-a declarat o variabilă de tip real *m*
natural *t* – s-a declarat o variabila de tip natural *t*
caracter *x* – s-a declarat o variabilă de tip caracter *x*
text *s*←*„Azi e luni”* – s-a declarat o variabilă de tip șir de caractere *s*, care are drept conținut textul dintre ghilimele

– ***Instrucțiunea de citire*** – preia succesiv valori și se asociază, în ordinea preluării, variabilelor specificate.

citește <listă variabile>

Exemplu:
citește *a, b, c* – s-au citit valorile pentru cele trei variabile **în ordinea înșirării lor.**
citește *m* – s-a preluat o valoare pentru variabila *m*.

Se pot prelua una sau mai multe valori folosind o singură instrucțiune de citire.

– *Instrucțiunea de scriere* – permite vizualizarea rezultatelor obținute în urma prelucrării datelor

scrie <listă variabile>

Exemplu:

scrie ***a, b, c*** – se afișează conținutul celor trei variabile ***a, b*** și ***c***.

scrie ***„Media este", m*** – se afișează mesajul urmat de valoarea variabilei ***m***.

Se pot afișa atât conținuturile variabilelor, precum și anumite mesaje care se trec între ghilimele. Conținuturile și mesajele se separă prin ***virgulă***.

– *Instrucțiunea de atribuire* – presupune atribuirea valorii obținute eventual în urma evaluării unei expresii, unei variabile din cadrul algoritmului

v ← expresie

Exemplu:

b ← 13 – i se atribuie variabilei *b* valoarea 13

a ← ***b*** + 3 – i se atribuie variabilei a rezultatul expresiei *b* + 3, și anume valoarea 16

x ← (7 < ***a***) – i se atribuie expresiei logice (***7 < a***) rezultatul, și anume valoarea 1 (true)

y ← (***a*** < 3) AND (***b*** > 5) – i se atribuie expresiei logice (***a*** < 3) AND (***b*** > 5) rezultatul, și anume valoarea 0 (false)

În cadrul unei instrucțiuni de atribuire, tipul variabilei trebuie să corespundă cu tipul expresiei.

Structura ***decizională (alternativă)*** este descrisă în limbajul pseudocod cu ajutorul ***instrucțiunii de decizie*** și are sintaxa:

```
dacă (condiție) atunci
        instrucțiuni_1
altfel
        instrucțiuni_2
sf.dacă
```

Modul de execuție al instrucțiunii de decizie

1) Se evaluează ***condiția***, stabilindu-se valoarea de adevăr a acesteia (Adevărat – True, Fals – False).

2) În funcție de valoarea de adevăr are loc una din cele două seturi de instrucțiuni. Dacă este ***adevărată*** condiția, se va executa setul de ***instrucțiuni_1***, iar dacă aceasta este ***falsă***, se va executa setul de ***instrucțiuni_2***.

3) Indiferent de ramura pe care va merge, după executarea instrucțiunilor se iese din structură și se continuă algoritmul.

Nu se pot executa ambele seturi de instrucțiuni simultan, deoarece valoarea condiției nu poate fi adevărată și falsă în același timp.

Este recomandată ***indentarea*** (alinierea mai înspre interior a instrucțiunilor din cadrul structurii), prezentată mai sus, în scrierea programelor pentru a putea fi urmărite mai ușor și pentru a spori lizibilitatea întregului algoritm.

Există și o formă simplificată a acestei instrucțiuni, și anume aceea în care lipsește a doua ramură – ***altfel***. În această situație avem următoarea formă:

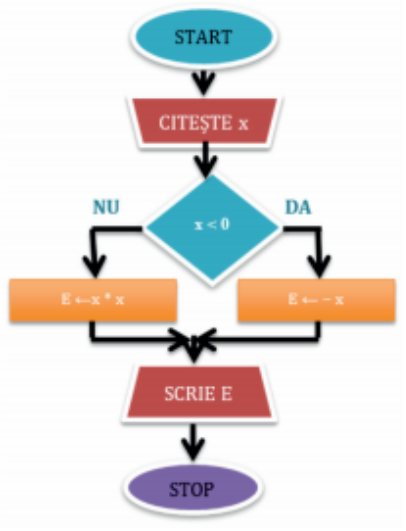
```
dacă (condiție) atunci
        instrucțiuni
sf.dacă
```

Modul de execuție

1) Se evaluează ***condiția***, stabilindu-se o valoare a acesteia (Adevărat – True, Fals – False).

2) Dacă este ***adevărată*** condiția, se va executa setul de ***instrucțiuni***, iar dacă aceasta este ***falsă***, se va ieși din structură.

Scrierea unui algoritm reprezentat prin schemă logică în limbajul pseudocod. Parcurgerea unui algoritm



Fie schema logică alăturată. Algoritmul scris în limbaj pseudocod ese următorul:

```
algoritm expresie
întreg x, E          // se declară variabilele
citește x            // se citește x
dacă (x < 0) atunci
    E← –x            // ramura DA
altfel
    E← x*x           // ramura NU
sf.dacă
scrie E              // se afișează valoarea expresiei
sfârșit algoritm
```

Pentru a verifica ce returnează algoritmul, în cazul unei valori concrete citite, acesta trebuie parcurs liniar de la prima la ultima instrucțiune, iar în cazul instrucțiunii ***dacă,*** se evaluează condiția și se alege ramura pe care se continuă. După ce se trece de instrucțiunea ***dacă***, se reia parcurgerea liniară a instrucțiunior.

Pași parcurși:

- se declară variabilele x și E;
- se citește o valoare pentru x;
- dacă valoarea citită este – 4, atunci condiția ($x < 0$) este adevărată, deci E va lua valoare – (– 4), adică 4;
- dacă valoarea citită ar fi fost 5, atunci condiția era negativă, iar E ar fi luat valoarea 25;
- se afișează valoarea lui E.