MINISTERUL EDUCAȚIEI NAȚIONALE

Mariana Miloşescu

Informatică

Profilul real

Specializarea:

matematică-informatică, intensiv informatică

Manual pentru clasa a IX - a

EDITURA DIDACTICĂ ŞI PEDAGOGICĂ, R.A.

Cuprins

1. Informatica şi societatea 3
1.1. Prelucrarea informațiilor 3
1.2. Informatica 4
1.3. Etapele rezolvării unei probleme 7
1.4. Algoritmul 8
Evaluare 10

2. Datele 12
2.1. Definiția datelor 12
2.1.1. Clasificarea datelor 13
2.1.2. Tipul datelor 19
2.2. Operatorii 21
2.3. Expresiile 26
Evaluare 30

3. Algoritmii 36
3.1. Reprezentarea algoritmilor 36
3.2. Principiile programării structurate 38
3.2.1. Structura liniară 38
3.2.2. Structura alternativă 39
3.2.3. Structura repetitivă 43
3.3. Algoritmi elementari 50
3.3.1. Algoritmi pentru interschimbare 50
3.3.2. Algoritmi pentru determinarea maximului (minimului) 52
3.3.3. Algoritmi pentru prelucrarea cifrelor unui număr 54
3.3.4. Algoritmi pentru calcularea c.m.m.d.c. 60
3.3.5. Algoritmi pentru testarea unui număr prim 62
3.3.6. Algoritmi pentru prelucrarea divizorilor unui număr 65
3.3.7. Algoritmi pentru conversii între sisteme de numerație 68
3.3.8. Algoritmi pentru generarea şirurilor recurente 71
3.4. Eficiența algoritmilor 73
3.5. Aplicarea algoritmilor 78
3.5.1. Rezolvarea problemelor de matematică 78
3.5.2. Rezolvarea problemelor de fizică 82
Evaluare 85

4. Implementarea algoritmilor 89
4.1. Caracteristicile limbajului de programare 89
4.2. Structura programului 91
4.3. Instrucțiunile declarative 93
4.3.1. Tipuri de date 93
4.3.2. Constantele 95
4.3.3. Declararea variabilelor de memorie 96
4.3.4. Declararea constantelor simbolice 98

4.3.5. Declararea tipurilor de dată utilizator 99
4.4. Operațiile de citire și scriere 100
Evaluare 104
4.5. Expresia și instrucțiunea expresie 106
4.5.1. Operatorii aritmetici 107
4.5.2. Operatorul pentru conversie implicită 109
4.5.3. Operatorii pentru incrementare și decrementare 111
4.5.4. Operatorii relaționali 113
4.5.5. Operatorii logici 113
4.5.6. Operatorii logici pe biți 114
4.5.7. Operatorii de atribuire 118
4.5.8. Operatorul condițional 121
4.5.9. Operatorul virgulă 123
4.5.10. Operatorul dimensiune 125
4.5.11. Precedența și asociativitatea operatorilor 125
Evaluare 127
4.6. Instrucțiunile de control 131
4.6.1. Instrucțiunea **if...else** 131
4.6.2. Instrucțiunea **switch...case** 134
4.6.3. Instrucțiunea **while** 137
4.6.4. Instrucțiunea **for** 139
4.6.5. Instrucțiunea **do...while** 146
Evaluare 149

5. Implementarea structurilor de date 155
5.1. Structurile de date 155
5.2. Tablourile de memorie 158
5.3. Implementarea tablourilor de memorie în limbajul C++ 162
5.3.1. Tabloul cu o singură dimensiune – vectorul 162
5.3.2. Tabloul cu două dimensiuni – matricea 164
5.4. Algoritmi pentru prelucrarea tablourilor de memorie 166
5.4.1. Algoritmi pentru parcurgerea tablourilor de memorie 166
5.4.2. Algoritmi pentru căutarea unui element într-un tablou de memorie 174
5.4.3. Algoritm pentru ștergerea unui element dintr-un vector 176
5.4.4. Algoritm pentru inserarea unui element într-un vector 177
5.4.5. Algoritmi pentru sortarea unui vector 179
5.4.6. Algoritm pentru interclasarea a doi vectori 188
5.4.6. Aplicarea algoritmilor pentru prelucrarea tablourilor de memorie 190
Evaluare 193
5.5. Fișierele 201
5.6. Implementarea fișierelor text în limbajul C++ 204
5.6.1. Fluxuri de date pentru fișiere text 205
5.6.2. Citiri și scrieri cu format 210
5.6.3. Aplicații cu prelucrări de fișiere 213

Evaluare 219
Anexa 1 – Mediul de programare C++ 222
Anexa 2 – Sisteme de numerație 231
Anexa 3 – Reprezentarea internă a datelor 233
Anexa 4 – Funcții de sistem utile 236
Anexa 5 – Codul ASCII 237

2. Datele

2.1. Definiția datelor

Datele sunt obiecte prelucrate de algoritm.

> **Data** este un model de reprezentare a informației, accesibil calculatorului, cu care se poate opera pentru a obține noi informații.

Din punct de vedere logic, data este definită printr-o tripletă:

data elementară = (identificator, valoare, atribute)

Identificatorul datei

Identificatorul datei este un nume format din unul sau mai multe caractere şi este atribuit unei date de către cel care defineşte data, pentru a o putea distinge de alte date şi pentru a putea face referiri la ea în procesul de prelucrare a datelor. De exemplu *alfa*, *a11* sau *b1_1*.

Fiecare limbaj de programare are implementat diferit conceptul de identificator al datei, practicând constrângeri pentru:

- ✓ **numărul maxim de caractere din nume** (de exemplu, 10 caractere în cazul limbajului programare Visual Basic şi 64 de caractere în cazul limbajului de programare Pascal),
- ✓ **caracterele acceptate în nume** (de exemplu, majoritatea limbajelor de programare nu acceptă în nume decât cifre, litere şi caracterul linie de subliniere), şi
- ✓ **caracterul care poate fi folosit la începutul numelui** (de exemplu, marea majoritate a limbajelor de programare nu acceptă ca numele unei date să înceapă cu o cifră).

De aceea, atunci când învățați să definiți şi să manipulați date folosind un anumit limbaj de programare, trebuie să aflați ce constrângeri există pentru identificatorul datei.

Valoarea datei

Valoarea datei reprezintă conținutul zonei de memorie în care este stocată data. Se defineşte ca **domeniu de definiție al datei** mulțimea valorilor pe care le poate lua data în procesul de prelucrare.

Atributele datei

Atributele sunt proprietăți ale datelor care determină modul în care sistemul va trata datele. Cel mai important atribut este **tipul datei**.

Tipul datei definește apartenența datei la o anumită clasă de date, căreia îi corespunde un anumit model de reprezentare internă.

Indiferent de tipul de date ales, reprezentarea datei în memoria calculatorului se face printr-un șir de biți. Pentru a realiza această reprezentare, fiecare limbaj de programare are implementați **algoritmi de codificare** care asigură corespondența dintre tipul de dată și șirul de biți, atât la scrierea datelor, cât și la citirea lor.

Așadar, **orice sistem care prelucrează informația sub formă de date trebuie să aibă definit clar conceptul de dată.** Definirea conceptului de dată implică definirea următoarelor elemente:

- ✓ cum poate fi identificată data?
- ✓ cum va fi reprezentată data în memoria calculatorului?
- ✓ ce proprietăți are data?
- ✓ cum pot fi grupate datele în colecții de date?

2.1.1. Clasificarea datelor

Clasificarea datelor se poate face folosind mai multe **criterii**:

1. În funcție de momentul în care se produc în **fluxul de informație**:
   - ✓ date de intrare;
   - ✓ date intermediare;
   - ✓ date de ieșire.
2. În funcție de **valoare**:
   - ✓ date variabile;
   - ✓ date constante.
3. În funcție de **modul de compunere**:
   - ✓ date elementare;
   - ✓ structuri de date.
4. În funcție de **tip**:
   - ✓ date numerice;
   - ✓ date logice;
   - ✓ date șiruri de caractere.

Clasificarea în funcție de momentul în care se produc

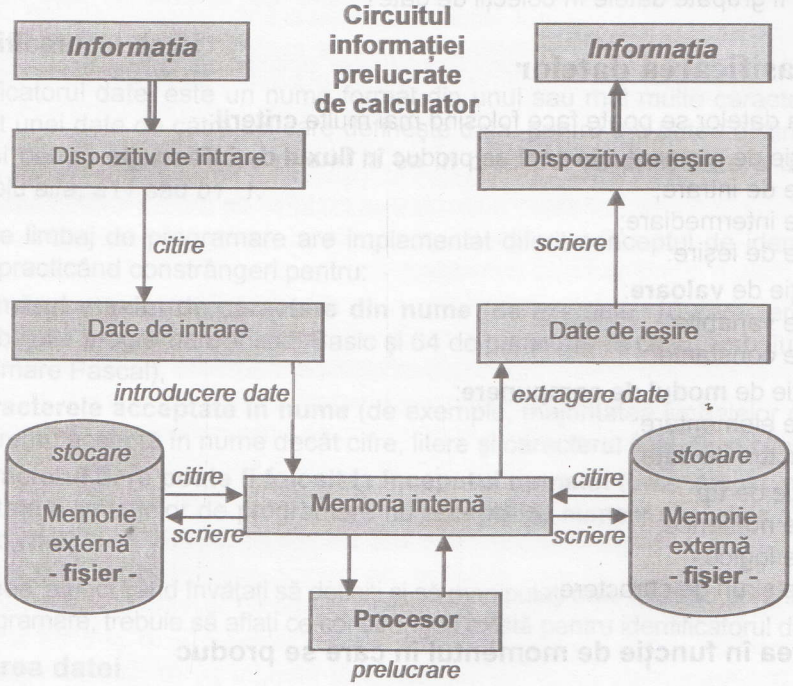
Datele se clasifică în:

- ✓ **Date de intrare**. Ele reprezintă datele care urmează să fie prelucrate în cadrul algoritmului. Sunt folosite pentru a descrie diverse evenimente și sunt informații produse în urma realizării unui eveniment, adică evaluări neprelucrate ale acelui eveniment. De exemplu, pot fi date de intrare notele unui elev. Pentru a putea fi prelucrate de procesor, datele sunt introduse în memoria internă a calculatorului. Introducerea datelor se face prin intermediul unor echipamente speci-

alizate în **citirea** informației, numite **dispozitive de intrare** (tastatură, scanner, creion optic etc.). **Dispozitivul standard de intrare este *tastatura*.**

- ✓ **Date de ieşire**. Ele sunt folosite pentru a descrie rezultatele obținute în urma prelucrărilor din cadrul algoritmului şi furnizează informațiile pentru care a fost realizat algoritmul, ca de exemplu mediile semestriale şi anuale ale elevului. Datele de ieşire sunt produse de procesor în urma operației de prelucrare şi sunt depuse în memoria internă. Pentru a fi vizualizate de om, ele sunt extrase din memoria internă prin intermediul unor echipamente specializate în **scrierea** informației, numite **dispozitive de ieşire** (ecran, imprimantă etc.). **Dispozitivul standard de ieşire este *ecranul*.**
- ✓ **Date intermediare** sau **de manevră**. Ele sunt folosite în cadrul algoritmului pentru realizarea unor prelucrări.

În vederea prelucrării, datele pot fi păstrate temporar în memoria internă sau în memoria externă (discul flexibil, hard-discul, discul compact etc.). Operația se numeşte **stocarea** datelor.



În memoria externă datele sunt păstrate în fişiere. **Un fişier este o colecție de date organizate ca o singură unitate**. Dacă datele sunt păstrate în **fişiere**, ele vor putea fi folosite ulterior ca date de intrare într-un alt algoritm.

Aşadar, orice rezolvare de problemă începe prin definirea datelor, continuă cu prelucrarea lor în conformitate cu algoritmul folosit şi se termină fie cu afişarea valorii lor, fie cu stocarea lor pe un mediu de memorare în vederea prelucrării lor ulterior.

Studiu de caz

Scop: exemplificarea tipurilor de date care pot să apară într-un algoritm.

Enunțul problemei: *Să se calculeze media aritmetică a* n *numere întregi introduse de la tastatură.*

În urma analizei problemei se obține **specificația programului**:

- ✓ **Funcția programului**. Se calculează suma numerelor introduse de la tastatură și se împarte la numărul de elemente *n*.
- ✓ **Informațiile de intrare** sunt: numărul de elemente ale mulțimii de numere și numerele care se citesc. Reprezentarea internă a informației se va face prin **datele de intrare**: ***n*** pentru numărul de numere și ***a*** pentru un număr citit curent.
- ✓ Pentru operațiile executate în cadrul algoritmului se vor folosi **datele intermediare *suma*** în care se calculează suma numerelor introduse de la tastatură, prin intermediul datei de intrare *a*, și ***i*** prin care se numără câte numere s-au introdus de la tastatură la un moment dat. Data intermediară *i* este necesară pentru a afla când se termină procesul de introducere a celor *n* numere de la tastatură, având funcția unui contor. Valoarea inițială a sumei și a contorului (înainte de a se citi primul număr) este *0*.
- ✓ **Informația de ieșire** va fi media aritmetică a celor *n* numere. Reprezentarea internă a mediei aritmetice se va face prin **data de ieșire *media*** a cărei valoare se calculează prin împărțirea sumei calculate cu ajutorul datei intermediare *suma* la numărul de elemente memorat în data de intrare *n*.

Așadar, datele folosite pentru rezolvarea acestei probleme sunt:

- ✓ **Date de intrare**: *n* și *a*.
- ✓ **Date intermediare**: *suma* și *i*.
- ✓ **Date de ieșire**: *media*.

iar algoritmul de rezolvare a problemei va prezenta un set unic de operații prin care se calculează valoarea mediei, oricare ar fi numărul de numere și valorile lor:

Pasul 1. *Început.*
Pasul 2. ***Comunică*** *valoarea pentru* ***n***.
Pasul 3. ***Atribuie*** *valorile inițiale datelor* ***suma*** *și* ***i***: ***suma=0*** *și* ***i=0***.
Pasul 4. ***Compară i≤n***. *Dacă este adevărat, execută* **Pasul 5**; *altfel, execută* **Pasul 8**.
Pasul 5. ***Comunică*** *valoarea pentru* ***a***.
Pasul 6. ***Calculează suma=suma+a*** *(adună la sumă noua valoare a lui* ***a***).
Pasul 7. ***Calculează i=i+1*** *(crește contorul* ***i*** *cu 1 deoarece s-a mai citit un număr* ***a***). *Mergi la* **Pasul 4**.
Pasul 8. ***Calculează media=suma/n***.
Pasul 9. ***Comunică*** *valoarea datei* ***media***.
Pasul 10. *Terminat.*

Observații:

1. Pașii care conțin acțiuni de comunicare folosesc numai date de intrare și de ieșire, nu și date intermediare. Datele intermediare apar numai în pași care conțin acțiuni de calcul, de atribuire sau de comparare.
2. Valoarea datelor de ieșire se calculează în cadrul algoritmului și se comunică printr-o operație de scriere. Pentru calcularea datelor de ieșire se folosesc date de intrare și/sau date intermediare. Aceste date trebuie să aibă o valoare înainte de a fi folosite în pașii care conțin acțiuni de calcul. Valoarea datelor de intrare este comunicată prin operația de citire de la tastatură. Datelor intermediare li se atribuie o valoare inițială în cadrul algoritmului.
3. **Orice nouă operație de citire executată cu o dată de intrare distruge vechea valoare memorată în dată.**

Clasificarea în funcție de valoare

Datele se clasifică în:

- ✓ **Date variabile** sau **variabile de memorie**. Pe parcursul procesului de prelucrare, valoarea acestor date se poate modifica, în limitele domeniului de definiție. În timpul execuției programului ele pot avea o valoare inițială, mai multe valori intermediare și o valoare finală. În exemplul precedent pentru calcularea mediei a *n* numere introduse de la tastatură, data *suma* are o valoare inițială *0*, mai multe valori intermediare, câte una pentru fiecare operație de citire a unui număr *a*, și o valoare finală, obținută după ce s-au citit toate cele *n* numere.
- ✓ **Date constante** sau **constante**. Pe tot parcursul procesului de prelucrare, data își va păstra aceeași valoare din domeniul de definiție al datei.

Studiu de caz

Scop: exemplificarea tipurilor de date care pot să apară într-un algoritm.

Enunțul problemei 1: *Să se calculeze aria pentru* n *cercuri, fiecare cerc având o rază precizată* r.

În urma analizei problemei se obține **specificația programului**:

- ✓ **Funcția programului**. Se calculează aria pentru *n* cercuri folosind formula matematică $aria=\pi \times r^2$, unde *r* este raza unuia dintre cele *n* cercuri.
- ✓ **Informațiile de intrare** sunt: numărul de cercuri, razele cercurilor și valoarea numărului π. Reprezentarea internă a informației se va face prin **datele de intrare**: ***n*** pentru numărul de cercuri, ***r*** pentru raza unui cerc, și ***pi*** pentru numărul π.
- ✓ Pentru operațiile executate în cadrul algoritmului se va folosi **data intermediară** *i* care reprezintă numărul cercului pentru care se calculează aria. Această dată intermediară este necesară pentru a afla când se termină procesul de calculare a ariei celor *n* cercuri și are valoarea inițială *0* (nu s-a calculat aria nici unui cerc).

- ✓ **Informațiile de ieşire** vor fi ariile celor *n* cercuri. Reprezentarea internă a ariei se va face prin **data de ieşire** *aria* a cărei valoare se calculează prin formula matematică. $aria=\pi \times r^2$.

Algoritmul de rezolvare a problemei va prezenta un set unic de operații prin care se calculează valoarea ariei, oricare ar fi numărul de cercuri şi valoarea razelor lor:

Pasul 1. *Început.*
Pasul 2. ***Comunică*** *valoarea pentru* ***n***.
Pasul 3. ***Atribuie*** *valoarea inițială contorului* ***i***: ***i=0***.
Pasul 4. ***Compară*** $i \leq n$. *Dacă este adevărat, execută* **Pasul 5**; *altfel, execută* **Pasul 9**.
Pasul 5. ***Comunică*** *valoarea pentru* ***r***.
Pasul 6. ***Calculează*** $aria=pi \times r^2$.
Pasul 7. ***Comunică*** *valoarea* ***ariei*** *pentru cercul* ***i***.
Pasul 8. ***Calculează*** $i=i+1$ *(creşte contorul* ***i*** *cu 1 deoarece s-a citit raza unui cerc* ***r***). *Mergi la* **Pasul 4**.
Pasul 9. *Terminat.*

Datele *n*, *i*, *arie* şi *r* sunt date variabile. Valoarea datei *n* se modifică pentru fiecare execuție a algoritmului (corespunde valorii introduse de la tastatură). Valoarea datelor *i*, *r* şi *aria* se modifică în funcție de numărul cercului pentru care se citeşte raza şi se calculează aria în timpul execuției algoritmului. Data *pi* este o dată constantă. Indiferent de datele de intrare cu care se execută algoritmul, valoarea ei este aceeaşi şi corespunde valorii numărului π.

Enunțul problemei 2: *Să se afişeze numerele pare care au două cifre.*

În urma analizei problemei se obține **specificația programului**:

- ✓ **Funcția programului**. Se generează numerele pare dintr-un interval $[a,b]$.
- ✓ **Informațiile de intrare** sunt limitele intervalului. Deoarece din enunțul problemei rezultă că $a=10$ (primul număr par cu două cifre) şi $b=98$ (ultimul număr par cu două cifre), reprezentarea internă a informației se va face prin **datele de intrare constante**: *10* şi *98*. Deoarece aceste date au valori constante, ele nu trebuie citite de la tastatură.
- ✓ **Informația de ieşire** vor fi numerele pare din intervalul precizat. Reprezentarea internă a unui număr par se va face prin **data de ieşire** ***n*** a cărei valoare se calculează prin incrementarea cu *2* a valorii anterioare: $n=n+2$.

Algoritmul de rezolvare a problemei va fi:

Pasul 1. *Început.*
Pasul 2. ***Atribuie*** *valoarea inițială numărului* ***n***: ***n=10***.
Pasul 3. ***Compară*** $n \leq 98$. *Dacă este adevărat execută* **Pasul 4**, *altfel execută* **Pasul 6**.
Pasul 4. ***Calculează*** $n=n+2$.
Pasul 5. ***Comunică*** *valoarea lui* ***n***. *Mergi la* **Pasul 3**.
Pasul 6. *Terminat.*

Clasificarea în funcție de modul de compunere

Datele se clasifică în:

- ✓ **Date simple** sau **date elementare**. Sunt date independente unele de altele din punct de vedere al reprezentării lor în memorie. Chiar dacă ele pot depinde din punct de vedere logic (valoarea unei date este dependentă de valoarea altei date), ele nu depind din punct de vedere fizic (localizarea unei date pe suportul de memorare nu se face în funcție de locația unei alte date pe suport). În calcularea ariei unui cerc, datele *n*, *i*, *r* şi *aria* sunt date elementare.
- ✓ **Date compuse** sau **structuri de date**. Sunt colecții de date între care există anumite relații. Fiecare componentă a structurii are o anumită poziție în cadrul structurii, iar toate componentele formează un întreg, astfel încât prelucrarea se poate face atât la nivelul structurii de date (care poate fi considerată o entitate de sine stătătoare), cât şi la nivelul fiecărei componente. Pentru fiecare tip de structură de date în limbajul de programare trebuie să fie definiți algoritmi de localizare a componentelor în cadrul structurii de date. Între componentele structurii există şi legături de conținut, adică întregul ansamblu de date din colecție poate caracteriza un obiect, o persoană, un fenomen, un proces etc. De exemplu, o colecție cu 12 elemente în care se memorează valorile lunare ale unui contor electric. Structura de date caracterizează în acest caz un proces: consumul lunar de energie electrică. Aşadar orice obiect, proces sau fenomen din lumea reală poate fi caracterizat printr-o listă de **proprietăți**. Valorile proprietăților din listă pot fi reprezentate în calculator (lumea virtuală) sub forma unei colecții de date.

Să ne închipuim că într-un algoritm trebuie reprezentată o clasă: profesorul şi cei *n* elevi. Ei reprezintă datele pe care le prelucrează algoritmul. Profesorului îi va corespunde o dată elementară, iar grupului de elevi o structură de date. Locul ocupat în clasă de fiecare dintre ei reprezintă zona de memorie alocată datei: catedra este zona de memorie alocată profesorului, iar grupul de bănci, zona de memorie alocată elevilor. Cele două zone sunt independente. În schimb, în cadrul zonei de bănci (zona de memorie a structurii de date), fiecărui element de structură (elevul) i se alocă un loc într-o bancă, poziția sa putând fi identificată după numărul băncii. Dacă pentru grupul de elevi nu s-ar folosi o structură de date, ci date elementare, fiecărei date elementare va trebui să îi dăm un nume, iar algoritmul ar fi foarte greu de scris. În primul rând nu ştim câte date elementare să folosim pentru elevi. Algoritmul trebuie să fie general, deci să funcționeze şi pentru o clasă cu 25 de elevi, dar şi pentru o clasă cu 30 de elevi. În timpul anului şcolar, poate să vină în clasă un elev nou sau poate să plece din clasă un elev. Ce se întâmplă în acest caz cu datele elementare, deoarece la o execuție a algoritmului, atunci când vine un elev nou în clasă ar trebui să apară o nouă dată elementară, iar la o altă execuție a algoritmului, atunci când pleacă un elev din clasă, trebuie să dispară o dată elementară? Soluția este de a folosi o colecție de date. Colecția va avea atâtea elemente câte bănci sunt în clasă. Se va folosi un singur nume de dată care se va atribui colecției, fiecare element identificându-se apoi după numărul băncii.

Studiu de caz

Scop: exemplificarea modului de compunere a datelor.

Enunțul problemei 1: *Să se calculeze media aritmetică a* n *numere introduse de la tastatură.*

Enunțul problemei 2: *Se introduc* n *numere de la tastatură. Să se afișeze aceste numere ordonate crescător.*

Pentru problema 1 se pot folosi numai date elementare, deoarece, după citirea unui număr prin intermediul datei *a*, el se prelucrează imediat (se adună la sumă) şi variabila de memorie va putea fi refolosită apoi pentru citirea unui alt număr.

Pentru problema 2 nu se poate folosi decât o colecție de date, deoarece, pentru aranjarea într-o anumită ordine a celor *n* numere citite de la tastatură trebuie să se păstreze în memorie toate aceste date pentru a se putea compara între ele în vederea ordonării.

2.1.2. Tipul datei

Tipul datei determină:
- ✓ **dimensiunea zonei de memorie** alocate datei (se măsoară în octeți);
- ✓ **operatorii** care pot fi aplicați pe acea dată;
- ✓ modul în care data este **reprezentată în memoria internă** (metoda de codificare în binar a valorii datei).

Tipul datei este definit prin dubletul **(V,*O*)**, unde:
V = domeniul de definiție intern al datei;
O = mulțimea operatorilor care se pot aplica pe mulțimea de valori ale datei.

Limbajele de programare acceptă următoarele **tipuri de date**:
- **tipul numeric**
- **tipul logic**
- **tipul şir de caractere**

Tipul numeric

Tipul numeric a fost implementat pentru reprezentarea numerelor întregi sau cu zecimale, pozitive sau negative, şi pentru a realiza majoritatea operațiilor matematice întâlnite în practică. Pentru tipul numeric există subtipurile **real** şi **întreg**.

Deci: **V** = **R** (mulțimea numerelor reale) sau **I** (mulțimea numerelor întregi)
$O = \mathcal{M}^1 \cup \mathcal{R}^2$

[1] Mulțimea operatorilor matematici.
[2] Mulțimea operatorilor relaționali (de comparare).

Constantele de tip numeric se reprezintă prin numere cu semn sau fără semn, folosindu-se punctul pentru separarea părții întregi de partea zecimală: 2; -0.15; 3.175; 20.0.

Tipul logic

Tipul logic sau boolean a fost implementat pentru reprezentarea datelor care nu pot lua decât două valori: adevărat (*true*), pe care o notăm cu *T*, sau fals (*false*), pe care o notăm cu *F*.

Deci: $\mathbf{V} = \mathbf{L}$ (mulțimea valorilor logice) = {*T*, *F*}
$O = \mathcal{L}$ [3]

Tipul şir de caractere

Tipul şir de caractere a fost implementat pentru reprezentarea unei mulțimi ordonate de caractere care este tratată ca un tot unitar.

Deci: $\mathbf{V} = \{\mathbf{P_C}\}$ (mulțimea părților mulțimii $\mathbf{C}$[4])
$O = \mathcal{R} \cup \mathcal{C}$ [5]

În memoria internă, fiecare caracter din şir se reprezintă prin codul său ASCII. Constantele de tip şir de caractere se specifică prin mulțimea ordonată de caractere care compun şirul, delimitată, în funcție de limbajul de programare, de anumite semne speciale: apostrofuri ('Buna ziua') sau ghilimele ("Buna ziua").

alfa ⟶ identificator de dată elementară

"alfa' 'alfa" ⟶ construcții greşite

"alfa" 'alfa' "500" '500' ⟶ constante de tip şir de caractere

500 ⟶ constantă de tip numeric

Constanta de tip numeric 500 este diferită de constanta de tip şir de caractere "500" atât din punct de vedere al modului de reprezentare în memoria internă a calculatorului, cât şi din punct de vedere al operatorilor acceptați. De exemplu, asupra constantei numerice se pot aplica operatori matematici şi relaționali, iar asupra constantei de tip şir de caractere operatori de concatenare şi relaționali. Constanta de tip numeric este reprezentată în memoria internă prin conversia în binar a numărului, iar constanta de tip şir de caractere este reprezentată prin conversia fiecărui caracter din şir în 8 cifre binare corespunzătoare codului ASCII al caracterului respectiv.

---

[3] Mulțimea operatorilor logici.
[4] Mulțimea caracterelor care este formată din litere, cifre şi semne speciale.
[5] Mulțimea operatorilor de concatenare.

2.2. Operatorii

Operatorii sunt caractere speciale (*, /, >, = etc.) sau cuvinte cheie[6] (**mod**, **and** etc.) prin intermediul cărora se reprezintă operațiile care se efectuează în cadrul unui algoritm. Fiecare limbaj de programare are implementat propriul set de operatori. În acest capitol vor fi prezentați operatorii care se folosesc în cadrul unui algoritm.

Asupra operanzilor dintr-o expresie puteți aplica următorii operatori:

- **operatorul de atribuire,**
- **operatorii matematici,**
- **operatorul de concatenare a șirurilor de caractere,**
- **operatorii relaționali,**
- **operatorii logici.**

Operatorii pot fi aplicați numai pe anumite tipuri de operanzi, producând rezultate de un anumit tip. Dacă notăm cu a și b operanzii asupra cărora se aplică operatorul, cu $\odot$ operatorul și cu c rezultatul:

$$a \odot b = c$$

atunci relația între a, b, $\odot$ și c este dată de următorul tabel:

Tipul operanzilor a și b	Tipul operatorului $\odot$	Tipul rezultatului c
numeric	matematic ($\mathcal{M}$)	numeric
numeric	relațional ($\mathcal{R}$)	logic
șir de caractere	concatenare ($\mathcal{C}$)	șir de caractere
șir de caractere	relațional ($\mathcal{R}$)	logic
logic	logic ($\mathcal{L}$)	logic

Operatorii matematici

$$\mathcal{M} = \{+, -, *, /, **|^{7}, \text{mod}, \text{div}\}$$

Se aplică pe date de tip numeric și furnizează un rezultat de tip numeric.

Operator	Semnificație	Exemplu
+ (adunare)	Adună matematic cei doi operanzi.	5+2=7
- (scădere)	Scade al doilea operand din primul operand.	7-3=4
/ (împărțire reală)	Împarte primul operand la al doilea operand.	7/2=3.5
* (înmulțire)	Înmulțește cei doi operanzi.	2*4=8
**|^ (ridicare la putere)	Ridică primul operand la puterea furnizată de cel de al doilea operand.	2**3=8

[6] Cuvinte rezervate, care nu mai pot fi folosite ca identificatori pentru date, deoarece ele au un înțeles bine stabilit pentru limbajul de programare.

[7] **Convenție de notare:** Semnul | plasat între două elemente are semnificația conjuncției *sau*, adică pentru operația de ridicare la putere pot fi folosite simbolurile ** *sau* simbolul ^.