MINISTERUL EDUCAŢIEI NAŢIONALE

Mariana Miloşescu

Informatică

Profilul real

Specializarea:
matematică-informatică, ştiinţe ale naturii

Manual pentru clasa a IX - a

EDITURA DIDACTICĂ ŞI PEDAGOGICĂ, R.A.

Cuprins

1. Informatica şi societatea

1.1. Prelucrarea informaţiilor

Calculatorul a fost inventat de om pentru a prelucra informaţia. El îl ajută să prelucreze *foarte uşor*, într-un timp *extrem de scurt*, cu *foarte mare acurateţe*, *o mare cantitate* de informaţie *foarte complexă*.

Prelucrarea informaţiei este veche de când lumea. **Prelucrarea voluntară a informaţiei** s-a făcut însă abia atunci când babilonienii au scris primele semne cuneiforme pe tăbliţele de lut. Aşadar, prima manifestare a prelucrării informaţiei a fost *scrisul*. Încă din antichitate, pot fi puse în evidenţă două tipuri de prelucrări de informaţii:

- ✓ **prelucrarea textelor = scrisul;**
- ✓ **prelucrarea numerelor = calculul numeric.**

Prelucrarea automată a informaţiei a fost posibilă o dată cu apariţia calculatoarelor electronice. Aşadar, scopul utilizării unui calculator este de a prelucra **informaţia**. Informaţia prelucrată poate fi formată din texte, numere, imagini sau sunete şi este păstrată pe diferite medii de memorare, în diferite formate, sub formă de **date**.

Transformarea datelor în informaţii nu este un atribut exclusiv al calculatorului. Acest fenomen a apărut o dată cu omul. De la primele reprezentări ale unor cantităţi cu ajutorul degetelor, al pietricelelor sau al beţişoarelor, şi de la manipularea manuală a acestor obiecte pentru a afla câte zile mai sunt până la un anumit eveniment sau câte animale au fost vânate sau câţi războinici are tribul vecin, putem spune că are loc un proces de transformare a datelor în informaţii. Degetele, pietricelele şi beţele reprezintă datele, iar ceea ce se obţine prin manipularea lor (numărul de animale vânate, numărul de zile, numărul de războinici) reprezintă informaţia. Cele ce deosebesc un astfel de proces de o prelucrare cu ajutorul calculatorului sunt viteza de obţinere a informaţiilor şi modul de reprezentare a informaţiilor sub formă de date.

Calculatorul nu ştie să prelucreze decât şiruri de cifre binare, care pot fi modelate fizic prin impulsuri de curent, cu două niveluri de tensiune, ce corespund celor două cifre binare: 0 şi 1. Prin urmare, datele vor fi codificări binare ale informaţiei existente în exteriorul calculatorului. Dacă, într-o prelucrare manuală, datele sunt reprezentate de obiecte care pot fi manipulate de om (beţişoare, pietricele sau degete), în cazul unei prelucrări automate datele vor fi reprezentate prin obiecte pe care le poate manipula calculatorul, adică şiruri de biţi.

Aşadar, din punct de vedere al unei prelucrări automate a informaţiei, diferenţa dintre dată şi informaţie este:

- ✓ **Informaţia** este un mesaj care înlătură necunoaşterea unui anumit eveniment şi are caracter de noutate. Informaţiile sunt interpretate de oameni.
- ✓ **Data** este reprezentarea informaţiei în interiorul calculatorului. Calculatorul nu înţelege conţinutul acestor date, el numai le prelucrează, prin operaţii specifice fiecărui tip de dată. În urma prelucrării datelor, calculatorul poate furniza omului informaţii.

Pentru a rezolva o anumită sarcină, trebuie să cunoaştem modul în care se poate face acest lucru. Aşadar, trebuie să găsim o anumită **metodă**, adică un set de paşi pe care trebuie să-i executăm ca să realizăm sarcina. Acest set de paşi formează **algoritmul** pentru rezolvarea problemei respective. Iniţial, studiul algoritmilor a fost o disciplină a matematicii, prin care se căuta să se găsească un set unic de instrucţiuni prin care să se descrie rezolvarea oricărei probleme dintr-o anumită categorie. Aţi învăţat deja o parte din aceşti algoritmi: algoritmul lui Euclid pentru găsirea celui mai mare divizor comun dintre două numere, algoritmul împărţirii unui număr, algoritmul extragerii rădăcinii pătrate dintr-un număr, algoritmul conversiei unui număr reprezentat în baza zece într-un număr reprezentat într-o altă bază de numeraţie etc. Cu timpul, descrierea metodei de rezolvare a unei probleme cu ajutorul algoritmilor s-a extins şi în alte domenii de activitate.

La fel ca şi omul, şi un calculator, pentru a putea rezolva o anumită sarcină, trebuie „să aibă cunoştinţe" despre modul în care se poate face aceasta, adică să cunoască algoritmul de rezolvare a problemei. Această informaţie i se transmite calculatorului prin intermediul unui **program**. Deoarece **limbajul natural** (limbajul prin care comunică oamenii) nu este înţeles de calculator, care este construit astfel încât să poată prelucra numai cifre binare, programul prin care i se comunică algoritmul este scris într-un limbaj de programare. **Limbajul de programare** este un **limbaj artificial** care, prin exprimări simbolice (**instrucţiuni**), descrie operaţiile de prelucrare pe care trebuie să le execute calculatorul. El îi permite omului să comunice cu calculatorul (să îi dea comenzi pe care să le execute), deoarece fiecare instrucţiune din limbajul de programare va fi tradusă într-un grup de **instrucţiuni maşină** (instrucţiuni în **limbaj maşină**), adică un şir de biţi care au o anumită semnificaţie pentru calculator. Acest limbaj se numeşte limbaj maşină deoarece este propriu fiecărui tip de maşină (calculator), fiind implementat – cu ajutorul circuitelor electronice – în procesor.

Aşadar, o sarcină se poate rezolva cu ajutorul calculatorului numai dacă modul în care se rezolvă poate fi descompus în paşi, pentru a putea fi descris cu ajutorul unui algoritm, deoarece calculatorul este o **maşină algoritmică**.

Dezvoltarea prelucrării automate a informaţiilor cu ajutorul calculatorului s-a făcut în două direcţii:

- ✓ **dezvoltarea echipamentelor**, astfel încât acestea să fie capabile să stocheze cât mai multă informaţie, pe care să o prelucreze cu viteză cât mai mare, folosind algoritmi cât mai complecşi;
- ✓ **găsirea de noi algoritmi**, cât mai performanţi, pentru rezolvarea problemelor complexe şi **îmbunătăţirea tehnicilor de reprezentare şi comunicare** a lor.

1.2. Informatica

Folosirea calculatorului a dus la apariţia unei noi ştiinţe şi a unui nou domeniu de activitate: **informatica**.

> **Informatica** reprezintă un complex de discipline prin care se asigură prelucrarea raţională a informaţiilor prin intermediul maşinilor automate.

Primul calculator electronic a apărut în anul 1946, ca urmare a unei cereri precise din partea armatei americane, care a fost capabilă să finanțeze un proiect atât de costisitor. Apoi, administrația americană a cumpărat primul calculator non-militar în 1951, pentru recensământul populației. Doi ani mai târziu a fost construit primul calculator destinat unei firme particulare, când General Electric a cumpărat un calculator pentru uzina sa din Louisville. Începând din 1953, firma IBM a început să pătrundă și ea pe piața de calculatoare, prezentându-și în mediile științifice propriile calculatoare. Astfel, calculatoarele au început să pătrundă și în mediile universitare. Dezvoltarea continuă a echipamentelor electronice de calcul a făcut ca, **din 1965, informatica să nu mai fie doar o activitate anexă, ci să se transforme ea însăși într-o industrie**. În contextul unei creșteri puternice a pieței, calculatorul a devenit o unealtă folosită în toate domeniile de activitate.

La primele calculatoare electronice, programele erau scrise în cod mașină (binar) sau erau cablate sub formă de circuite electronice. Modificarea unui program și introducerea unuia nou erau foarte complicate, deoarece însemna introducerea programului bit cu bit. Din necesitatea rezolvării acestei probleme, au apărut primele sisteme de operare și primele limbaje de programare numite limbaje de nivel înalt: în 1956, limbajul **Fortran** orientat pe calcule tehnico-științifice, și în 1960 limbajul **Cobol** orientat pe aplicații economice care folosesc puține operații de calcul, dar care manipulează un volum mare de date. Limbajele de programare s-au dezvoltat continuu, pentru a se adapta la noile echipamente hardware, la noile sisteme de operare și la noile cerințe ale utilizatorilor – care însemnau de fapt noi sarcini pe care trebuia să le rezolve calculatorul, adică noi algoritmi, orientați pe rezolvarea anumitor probleme. În 1971 a fost creat în universitățile elvețiene limbajul **Pascal**, primul **limbaj structurat** (fiecare prelucrare elementară este considerată ca un bloc, iar blocurile pot fi închise – încapsulate – unele în altele). O dată cu apariția microcalculatoarelor, acest limbaj s-a răspândit foarte mult. Limbajul **Basic** a fost creat în Statele Unite, în 1975, ca un **limbaj interactiv**, și nu putea fi folosit decât pe microcalculatoare. El permitea abordarea programării și de către persoane care nu erau specialiste în informatică. În 1971 a fost creat de firma Bell-Telephone limbajul **C**, pentru a permite realizarea sistemului de operare Unix. Este un limbaj foarte performant, care posedă atât conceptele limbajelor structurate de nivel înalt, cât și conceptele limbajelor de nivel scăzut, care îi permit accesul la hardware. Programele scrise în limbajele apărute recent au crescut productivitatea programatorilor. Limbajele de nivel înalt au pus bazele **ingineriei programării**.

La începutul anilor '60, în mediile universitare au început să se formeze departamente pentru cercetarea și studierea calculatoarelor. Cu timpul, a apărut o bogată literatură de specialitate, iar cursurile din domeniul informaticii au început să fie orientate pe subdomenii și să fie gradate pe niveluri de dificultate. Astăzi, informatica este divizată în următoarele nouă **subdomenii**.

1. **Algoritmi și structuri de date.** Studiază metodele prin care se pot obține aplicații care să prelucreze diferite clase de informații, modul în care vor fi reprezentate informațiile care vor fi prelucrate și metodele de optimizare a pașilor necesari pentru realizarea aplicațiilor. Scopul acestui subdomeniu este de a identifica problemele care pot fi descrise cu ajutorul algoritmilor, de a găsi modul în care trebuie procedat pentru a descoperi algoritmul și metodele de analiză și comparare a caracteristicilor algoritmilor pentru a obține algoritmi cât mai eficienți.

2. **Limbaje de programare.** Studiază notațiile (limbajele) prin care vor fi reprezentați algoritmii şi structurile de date, astfel încât aplicația să poată fi prelucrată. Aceste limbaje sunt apropiate de limbajul natural şi pot fi uşor traduse în secvențe de comenzi pe care să le înțeleagă calculatorul. Scopul acestui subdomeniu este de a găsi noi tehnici de reprezentare şi comunicare a algoritmilor.
3. **Arhitectura calculatoarelor.** Studiază modul în care sunt organizate diferite componente hardware ale calculatorului şi modul în care sunt conectate, pentru a putea obține un sistem eficient, sigur şi util. Scopul acestui subdomeniu este de a realiza maşini algoritmice cât mai bune folosind cunoştințele despre algoritmi deja dobândite şi tehnologia existentă.
4. **Sisteme de operare.** Studiază felul în care trebuie să fie organizate programele care controlează şi coordonează toate operațiile din sistemul de calcul. Scopul acestui subdomeniu este de a face un calculator să rezolve în acelaşi timp mai multe sarcini, fără ca paşii algoritmilor care descriu rezolvarea acestor sarcini să interfereze unii cu alții, iar atunci când este cazul să se poată realiza comunicarea între diverşi algoritmi.
5. **Ingineria programării.** Studiază metodele prin care poate fi automatizată activitatea de proiectare a aplicațiilor şi de prelucrare a informațiilor, astfel încât să se obțină programe corecte, eficiente, fără erori şi uşor de exploatat.
6. **Calcule numerice şi simbolice.** Studiază descrierea fenomenelor din lumea reală prin intermediul formulelor matematice, care pot fi manipulate algebric astfel încât să se obțină modele matematice uşor de descris prin algoritmi. Scopul acestui subdomeniu este de a găsi modele matematice care să permită descrierea şi reprezentarea în calculator a fenomenelor complexe, cum sunt: zborul avioanelor, curenții marini, traiectoria sateliților şi a planetelor, mişcarea particulelor etc.
7. **Sisteme de gestiune a bazelor de date.** Studiază modul în care pot fi organizate cantități mari de date ce nu necesită, în prelucrare, calcule matematice complexe. Este cazul informațiilor prelucrate în procesele economico-sociale, în întreprinderi şi în administrație. Prelucrarea acestor date trebuie să se facă eficient, fără erori, cu asigurarea securității lor.
8. **Inteligența artificială.** Studiază modul în care percepe şi raționează mintea umană, cu scopul de a putea fi automatizate aplicații pe care omul le realizează prin metode „inteligente", care sunt dificil de descris cu ajutorul algoritmilor, ca de exemplu înțelegerea unui limbaj, crearea de noi teorii matematice, compunerea muzicii, crearea operelor de artă, luarea deciziilor în urma evaluării unor situații complexe (stabilirea unui diagnostic în medicină, mutarea pieselor la jocul de şah etc.).
9. **Animație şi robotică.** Studiază metodele prin care pot fi generate şi prelucrate imaginile şi modul în care se poate răspunde unei situații din exterior prin acționarea unui robot.

1.3. Etapele rezolvării unei probleme

Orice prelucrare automată a informațiilor presupune definirea următorului lanț:

Din această cauză, pentru orice rezolvare a unei probleme cu ajutorul calculatorului, trebuie parcurse următoarelor **etape**:

1. **analiza problemei;**
2. **elaborarea modului de rezolvare a problemei;**
3. **codificarea într-un limbaj de programare a modului de rezolvare a problemei;**
4. **testarea programului şi corectarea erorilor.**

1. Analiza problemei. Această etapă constă în formularea enunțului problemei din care vor rezulta **specificațiile** complete şi precise ale programului care va rezolva problema. Aceste specificații trebuie să țină cont de condițiile concrete de realizare a programului. Specificațiile sunt:

- ✓ **Funcția programului**. Prin ea, se determină ceea ce urmează să realizeze programul.
- ✓ **Identificarea fluxului de informații**. Aceasta presupune identificarea **informațiilor de intrare** şi, respectiv, a **informațiilor de ieşire** care vor fi descrise cu ajutorul **datelor**: **date de intrare** şi, respectiv, **date de ieşire**.

Fiecărui tip de informație îi corespunde un anumit mod de stocare în mediul de memorare, adică un anumit tip de dată. Între datele prelucrate de un program există diferite relații. Modul în care vor fi aranjate aceste date în mediul de memorare depinde de legătura dintre ele.

2. Elaborarea modului de rezolvare a problemei. Această etapă constă în găsirea metodei prin care să se poată rezolva problema. Ea presupune identificarea **prelucrărilor** care se fac asupra datelor de intrare pentru a obține datele de ieşire. Descrierea acestor prelucrări se face cu ajutorul **algoritmului** de rezolvare a problemei. Această fază este cea mai importantă şi cea mai grea, deoarece presupune definirea logică a unei secvențe de operații pe care să le poată executa calculatorul, astfel încât să se obțină rezultatele dorite.

3. Codificarea într-un limbaj de programare a modului de rezolvare a problemei. Algoritmul de rezolvare a problemei este transpus într-un limbaj de programare ales în conformitate cu specificul problemei care trebuie să fie rezolvată, pentru a fi comunicat calculatorului.

4. Testarea programului şi corectarea erorilor. Pentru testarea programului se va folosi o mulțime de seturi de date de intrare, care trebuie să prevadă toate situațiile care pot să apară în exploatarea curentă a programului. Testarea constă în executarea repetată a programului, pentru fiecare set de date de intrare. Dacă această mulțime de seturi de date nu este aleasă corect, programul nu va fi testat pe toate traseele algoritmului şi în etapa de exploatare pot apărea erori. În această etapă se pun în evidență erorile de sintaxă, erorile de logică şi dacă reprezentarea externă a rezultatelor are aspectul grafic dorit. Erorile de sintaxă apar din scrierea incorectă a instrucțiunilor şi ele vor fi corectate în program. Erorile de logică apar din cauza metodei de rezolvare alese şi ele vor trebui identificate în cadrul algoritmului şi corectate în program.

Aşadar, pentru ca un calculator **să poată produce informații**, trebuie ca, la rândul său, **să primească două categorii de informații**:

- ✓ Descrierea modului în care realizează sarcina, adică **algoritmul**, care i se comunică sub forma unui **program**.

✓ Informaţiile de care are nevoie algoritmul ca să realizeze acea sarcină care i se comunică sub formă de **date de intrare**.

Studiu de caz

Scop: exemplificarea etapelor de rezolvare a unei probleme.

Enunţul problemei: *Fiind date două numere reale a şi b, să se rezolve ecuaţia de gradul întâi cu aceşti coeficienţi: ax + b = 0.*

În urma analizei problemei, se obţine **specificaţia programului**:

- ✓ **Funcţia programului**. Dacă pentru ecuaţia de gradul întâi $ax + b = 0$ există o soluţie reală, se calculează; în caz contrar, se afişează un mesaj.
- ✓ **Informaţiile de intrare** sunt coeficienţii ecuaţiei, iar suportul extern prin care se vor introduce este tastatura. Reprezentarea internă a informaţiei se va face prin **datele de intrare** *a* şi *b*.
- ✓ **Informaţia de ieşire** va fi soluţia ecuaţiei, dacă există, iar dacă nu există, un mesaj. Suportul extern pe care va fi reprezentată informaţia de ieşire este ecranul monitorului. Reprezentarea internă a soluţiei ecuaţiei se va face prin **data de ieşire** *x*.

Metoda folosită pentru rezolvarea problemei va fi algoritmul matematic de rezolvare a ecuaţiei de gradul întâi.

Pentru testarea programului, se va considera că un set de date de intrare este format de perechea de coeficienţi (a; b), iar o mulţime completă de seturi de date de intrare poate fi {(0; 0), (0; 1.5), (2.5; 1.5)}.

1.4. Algoritmul

Datele de intrare sunt supuse unui proces de prelucrare, pentru a se obţine datele de ieşire. În funcţie de rezultatele care se doresc, prelucrarea datelor este realizată după un anumit algoritm.

Algoritmul reprezintă o **mulţime ordonată** şi **finită de paşi executabili** prin care se defineşte fără echivoc modul în care se poate realiza o anumită sarcină.

Între datele de intrare şi datele de ieşire ale algoritmului există o relaţie bine determinată de însăşi construcţia algoritmului.

În activităţile zilnice întâlnim la tot pasul algoritmi: algoritmul de utilizare a maşinii de spălat rufe sau vase (exprimat prin setul de instrucţiuni din cartea tehnică a maşinii sau de pe capacul maşinii de spălat), algoritmul de înregistrare pe o casetă video (exprimat prin setul de instrucţiuni din cartea tehnică a videorecorderului), algoritmul de interpretare a muzicii (exprimat prin partitură), algoritmul de construire a unui model de avion sau de navă (exprimat prin setul de instrucţiuni care însoţesc piesele care compun modelul), algoritmul de rezolvare a unei probleme matematice (exprimat printr-un set unic de operaţii prin care se descrie modul de rezolvare a oricărei probleme dintr-o categorie de probleme). De fapt, aproape toa-

te acțiunile noastre se desfăşoară după un algoritm bine definit. Un exemplu de algoritm al activităților zilnice este o convorbire telefonică:

Pasul 1. *Început.*
Pasul 2. *Mergi la telefon.*
Pasul 3. *Ridică microreceptorul telefonului.*
Pasul 4. *Dacă* ***are ton****, formează numărul de telefon; altfel, pleacă la vecin şi mergi la* **Pasul 10**.
Pasul 5. *Dacă* ***telefonul este ocupat****, închide telefonul şi mergi la* **Pasul 11**; *altfel, aşteaptă să răspundă.*
Pasul 6. *Dacă* ***nu răspunde****, pune microreceptorul în furcă şi mergi la* **Pasul 12**; *altfel, începi discuția cu persoana care a răspuns.*
Pasul 7. *Dacă* ***a răspuns persoana căutată****, mergi la* **Pasul 9**; *altfel, cere să vină la telefon persoana căutată.*
Pasul 8. *Dacă* ***persoana căutată nu poate să vină la telefon****, mergi la* **Pasul 13**; *altfel, aşteaptă să vină la telefon.*
Pasul 9. *Discută la telefon cu persoana căutată şi mergi la* **Pasul 13**.
Pasul 10. *Anunță la serviciul „Deranjamente telefoane" că ai telefonul defect şi mergi la* **Pasul 14**.
Pasul 11. *Aşteaptă 15 minute şi mergi la* **Pasul 2**.
Pasul 12. *Aşteaptă 1 oră şi mergi la* **Pasul 2**.
Pasul 13. *Închide telefonul.*
Pasul 14. *Terminat.*

Un exemplu de algoritm matematic este rezolvarea ecuației de gradul întâi:

$$a \times z + b = 0$$

unde **a** şi **b** sunt coeficienții ecuației şi pot lua orice valori din domeniul numerelor reale, iar **z** reprezintă un număr care se calculează şi care poate lua şi el orice valoare reală, astfel încât să fie îndeplinită relația definită prin ecuație. Algoritmul de rezolvare a ecuației va prezenta un set unic de operații, prin care se calculează valoarea lui **z**, oricare ar fi valorile pentru **a** şi **b**:

Pasul 1. *Început.*
Pasul 2. ***Comunică*** *valorile pentru* ***a*** *şi* ***b****.*
Pasul 3. ***Compară a = 0****. Dacă este adevărat, execută* **Pasul 4**; *altfel, execută* **Pasul 7**.
Pasul 4. ***Compară b = 0****. Dacă este adevărat, execută* **Pasul 5**; *altfel, execută* **Pasul 6**.
Pasul 5. ***Comunică*** *mesajul "Ecuația are o infinitate de soluții". Mergi la* **Pasul 9**.
Pasul 6. ***Comunică*** *mesajul "Ecuația nu are soluții". Mergi la* **Pasul 9**.
Pasul 7. ***Calculează z = –b/a****.*
Pasul 8. ***Comunică*** *valoarea lui* ***z****.*
Pasul 9. *Terminat.*

Numărul de paşi este finit (9 paşi). Toți paşii reprezintă acțiuni care se pot executa: compară, calculează, comunică. O dată definit acest algoritm, paşii lui se vor executa pentru orice valori ale lui **a** şi **b**, deci algoritmul descrie rezolvarea unei probleme generale. La fiecare executare a algoritmului care descrie o problemă generală va fi tratat un caz particular, adică se rezolvă ecuația de gradul întâi pentru va-

lori precizate ale lui **a** şi **b**, ca de exemplu **2×z – 4 = 0** (**a = 2**, **b = –4**) sau **0×z – 4 = 0** (**a = 0**, **b = –4**) sau **0×z – 0 = 0** (**a = 0**, **b = 0**).

Aşadar, algoritmii au următoarele **proprietăţi**:

- ✓ **Claritatea**. Orice algoritm trebuie să fie precis definit, să prezinte clar toate etapele care trebuie parcurse până la obţinerea soluţiei, fără să formuleze nimic ambiguu.
- ✓ **Finitatea**. Algoritmul trebuie să fie format dintr-un număr finit de paşi, prin executarea cărora să se ajungă la rezolvarea problemei şi obţinerea rezultatelor.
- ✓ **Succesiunea determinată a paşilor**. Paşii care compun algoritmul trebuie executaţi într-o ordine bine determinată. De obicei, ei se execută în ordine secvenţială (ordinea în care au fost scrişi). În cazul în care apare necesitatea schimbării acestei ordini, trebuie să se precizeze clar pasul care urmează să fie executat.
- ✓ **Universalitatea**. Algoritmul trebuie să permită rezolvarea unei clase de probleme, care sunt de acelaşi tip şi care diferă între ele numai prin datele de intrare. El trebuie să ofere posibilitatea de a rezolva orice problemă din acea clasă de probleme.
- ✓ **Realizabilitatea**. Paşii care compun algoritmul trebuie să reprezinte operaţii care se pot executa cu resursele disponibile.
- ✓ **Eficienţa**. Operaţiile care compun algoritmul trebuie alese astfel încât soluţia problemei să fie obţinută după un număr minim de paşi, cu precizia prestabilită sau cu o precizie satisfăcătoare.

Evaluare

Răspundeţi:

1. Ce este un algoritm? Ce sunt paşii algoritmului?
2. Determinaţi algoritmul pentru prepararea unui ceai. Identificaţi proprietăţile unui algoritm, în cazul acestui algoritm.
3. Citiţi o reţetă din cartea de bucate. Determinaţi algoritmul pentru prepararea respectivului produs culinar.
4. Daţi patru exemple de probleme a căror rezolvare nu poate fi descrisă cu ajutorul algoritmului şi patru exemple de probleme a căror rezolvare poate fi descrisă cu ajutorul algoritmului.

2. Datele

2.1. Definiția datelor

Datele sunt obiectele 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 de 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);
- ✓ **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** defineşte 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 specializate în **citirea** informației, numite **dispozitive de intrare** (tastatură, scanner, creion optic etc.). **Dispozitivul standard de intrare este *tastatura*.**
- ✓ **Date de ieșire**. Ele sunt folosite pentru a descrie rezultatele obținute în urma prelucrărilor din cadrul algoritmului și furnizează informațiile pentru care a fost realizat algoritmul, ca de exemplu mediile semestriale și anuale ale elevului. Datele de ieșire sunt produse de procesor în urma operației de prelucrare și sunt depuse în memoria internă. Pentru a fi vizualizate de om, ele sunt extrase din memoria internă prin intermediul unor echipamente specializate în **scrierea** infor-