Carmen Mincă Radu Boriga

INFORMATICĂ

PROFILUL REAL NEINTENSIV

Pascal - C++

Manual

(Aprobat MEC cu Ordinul nr. 4188/02.07.2004)

+

miniculegere de probleme

(nu a facut obiectul avizării)

L&S Info-mat

Cuprins

1. Algoritmi

1.1. Noțiuni generale

„Un algoritm e un șir de reguli fixe. O funcție recursivă e ceva care ne face să trecem de la 1 la 2, de la 2 la 3, și așa mai departe. Că, în fond ele sunt același lucru, aceasta este o teoremă."
(Gr.C.Moisil)

Noțiunea de *algoritm* este una dintre noțiunile centrale ale așa numitei revoluții tehnico-științifice declanșate de apariția calculatoarelor electronice, fără însă a constitui un produs al acesteia. Ea este o noțiune foarte veche, semnificația ei schimbându-se în diferitele etape de evoluție a științei și include în ea preocupările omului de a rezolva problemele într-un mod în care soluția lor să poată fi obținută mecanic, cu ajutorul calculatorului.

Primii lingviști au considerat că noțiunea de algoritm derivă din combinația cuvintelor *algiros* (trudă) și *aritmos* (numere). Alții au susținut că originea cuvântului se găsește în cuvântul *Algor*, numele unui rege al Castiliei. Unii istorici matematicieni au considerat că originea reală a cuvântului algoritm provine din numele faimosului autor de cărți arabe "Abu Ja'far Mohammed ibn Musâ al Khowârizmî" (825 e.n). Traducerea literară a numelui este: "tatăl lui Ja'far Mohammed, fiul lui Moses, născut în Khowârizmî", Khowârizmî fiind numele vechi al micului orășel Khiva din Rusia.

Un dicționar german (Lexiconul Matematic Complet - Leipzig, 1747) a definit astfel cuvântul *"algorism" : "În această denumire sunt combinate noțiunile a patru tipuri de calcule aritmetice și anume : adunarea, scăderea, înmulțirea și împărțirea".* În aceeași perioadă apare și expresia latină *"algorithmus infinitesimala"* pentru a denumi *"tipuri de calcule cu cantități infinite mici"* (Calculul infinitezimal - Leibniz).

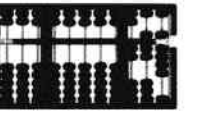
Începând cu anul 1950, cuvântul algoritm este frecvent asociat cu *"algoritmul lui Euclid"*, care reprezintă un proces de determinare a celui mai mare divizor comun pentru două numere naturale și apare în *"Elementele"* lui Euclid, cartea a III-a.

Cuvântul algoritm apare în dicționare începând cu anul 1957. Până în acel an, în dicționare era prezentă doar forma veche a cuvântului

"algorism", care are următoarea semnificație: procesul de a face aritmetică cu ajutorul numerelor arabe.

Cu timpul, forma și semnificația cuvântului "algorism", se schimbă. În dicționarul englez din Oxford, se precizează faptul că s-a realizat o "refacere eronată" a cuvântului prin confuzia literelor asemănătoare din cuvântul "aritmetică". Originea inițială a cuvântului "algorism" fiind uitată și, datorită asemănării dintre acesta și aritmetică, s-a realizat transformarea cuvântului "algorism" în "algoritm".

În Evul Mediu, lumea calculatoriștilor, a persoanelor care studiau calculele cu numere, s-a scindat în două grupări. Prima grupare (abaciștii) folosea așa numitele *"abace"*, niște mașini de calcul simple. A doua grupare (algoriștii) se foloseau în realizarea calculelor de *"algorismi"*, ce constituiau diferite metode de calcul cu utilizarea scrierii arabe a cifrelor. În jurul anului 1500, algoriștii au cucerit lumea calculelor, astfel încât, în secolul al XVIII-lea în întreaga Europă nu va mai exista nici urmă de abac. Însă, în câteva colțuri ale lumii din China, Japonia, Rusia și unele țări arabe, abacul a continuat să fie utilizat până în secolul nostru. În anii 1970-1980 tehnologiile moderne au pus la dispoziție oamenilor o unealtă foarte ieftină: calculatorul de buzunar, în care sunt contopite mai multe forme de algoritmare și de scriere a numerelor. Lumea începe să renunțe din nou la calculele algoristice, reînviind vechea dispută dintre algoriști și abaciști.



Noțiunea de algoritm constituie fundamentul construcției și programării calculatoarelor electronice, algoritmul referindu-se la posibilitatea descrierii procesului de soluționare a diverselor categorii de probleme cu ajutorul unui număr finit de operații simple.

Algoritmul este prezent în activitatea școlară frecvent. Iată câteva exemple:

- algoritmul de determinare a celui mai mare divizor comun pentru două numere naturale, datorat lui Euclid;
- algoritmul pentru determinarea numerelor prime mai mici sau egale decât un număr natural dat – ciurul lui Eratostene;
- algoritmul prin care se adună două fracții;
- algoritmul de rezolvare al ecuației de gradul al II-lea cu coeficienți reali;
- algoritmul de calculare a mediilor semestriale și generale;
- algoritmul de repartizare a elevilor în licee pe baza mediilor obținute la examenul pentru admiterea în licee.

Putem considera că ***algoritmul*** *reprezintă o metodă de soluționare a problemelor, constând dintr-o mulțime finită și bine ordonată de operații bine definite, care realizează o prelucrare a datelor de intrare (informațiile primite), oferind rezultatele dorite (datele de ieșire).*

În acest sens noțiunea de algoritm are strânse legături cu logica matematică și prelucrarea automată a datelor.

1.2. Enunțul unei probleme, date de intrare și de ieșire. Etapele rezolvării unei probleme

Se consideră o funcție $f:\mathfrak{R}\to\mathfrak{R}$, $f(x)=|2x+1|-7$. Se cere să se calculeze $f(x)$ pentru 7 valori reale ale lui x: $x_1=-0.5$, $x_2=3$, $x_3=-5$, $x_4=-20$, $x_5=10$, $x_6=5$ și $x_7=1$.

Înainte de a realiza calcularea valorilor cerute, vom explicita modulul prezent. Astfel expresia analitică a funcției devine:

$$f(x)=\begin{cases} 2x-6, & \text{dacă } x>-0.5 \\ -7, & \text{dacă } x=-0.5 \\ -2x-8, & \text{dacă } x<-0.5 \end{cases}$$

Fie $x_1=-0.5$. Observăm că $f(-0.5)=-7$. Fie $x_2=3$. Cum $x_2>-0.5$, atunci $f(3)=2\cdot3-6=0$. Fie $x_3=-5$. Cum $x_3<-0.5$, atunci $f(-5)=-2\cdot(-5)-8=2$. Se repetă calculul pentru celelalte patru valori. Observăm că algoritmul de calculare a valorilor $f(x)$ constă în a compara valoarea lui x cu -0.5. Dacă $x>-0.5$, atunci $f(x)=2x-6$. Altfel, dacă $x=-0.5$ atunci $f(x)=-7$. Altfel, dacă $x<-0.5$ atunci $f(x)=-2x-8$.

Dacă am utiliza un calculator pentru a efectua aceste calcule, calculatorul va trebui să decidă automat, prin executarea unui program, care dintre cele trei expresii ale funcției va fi utilizată în calcularea $f(x)$, în funcție de valoarea lui x (mai mare, mai mică sau egală cu -0.5). Programul corespunzător problemei va fi rezultatul codificării algoritmului de rezolvare a problemei și va conține mai multe comenzi (instrucțiuni) care vor fi executate de calculator.

În esență, procesul de rezolvare a unei probleme începe cu specificarea acesteia și se încheie cu obținerea unui program concret și corect.

În general, pentru elaborarea unui program, este necesar să parcurgem următoarele etape:

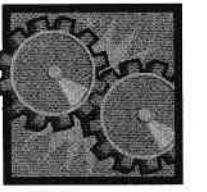
1. Specificarea problemei. Identificarea datelor de intrare și de ieșire.

În această primă etapă are loc analiza problemei. Prin această analiză se urmărește elaborarea unui enunț concret și precis al problemei, care să țină cont de condițiile concrete de realizare și execuție a programului. Enunțul trebuie să evidențieze *funcțiile programului*, adică ceea ce urmează să realizeze programul. În acest scop trebuie să identificăm informațiile de prelucrat (datele de intrare) și rezultatele cerute (datele de ieșire) ale programului. Referirea la datele de intrare și de ieșire se realizează prin intermediul variabilelor de intrare și de ieșire. Variabilele reprezintă notații simbolice pentru date.

Această primă etapă va continua cu stabilirea reprezentării și organizării datelor pe suporturile externe de informație, ținându-se cont de cerințele din enunțul problemei sau de cele impuse de utilizator.

Revenind la exemplul de mai sus, datele de intrare sunt cele 7 valori reale: x_1, x_2, x_3,...,x_7 datele de ieșire sunt $f(x_1)$, $f(x_2)$,..., $f(x_7)$. Funcția programului ce urmează a fi elaborat constă în calcularea celor 7 valori ale funcției din enunțul problemei. În ceea ce privește organizarea și reprezentarea datelor de intrare și ieșire pe suportul extern, precizăm că datele (valorile) se vor introduce de la tastatură, iar rezultatele se vor afișa pe monitor.

2. Elaborarea unui algoritm pentru rezolvarea problemei.



În această etapă are loc descrierea procesului de calcul ce conduce la soluționarea problemei, deci are loc determinarea algoritmului. În general, algoritmul specifică operațiile pe care le are de executat calculatorul pentru ca prin prelucrarea datelor de intrare să se obțină datele de ieșire așteptate.

Pentru utilizarea unui algoritm nu sunt necesare cunoștințe rafinate de matematică, este suficient să se aplice cu strictețe, în ordinea în care sunt precizate, toate operațiile cuprinse în algoritm. Însă, găsirea algoritmului constituie de cele mai multe ori cea mai dificilă etapă a procesului programării, presupunând cunoștințe din algebră, analiza matematică, discipline științifice și tehnice. Studiul algoritmilor constituie un domeniu special clar delimitat în aria largă a preocupărilor legate de știința calculatoarelor. Dezvoltarea algoritmului se realizează iterativ, prin trecerea lui gradată prin mai multe niveluri de detaliere (*proiectarea descendentă*), pornindu-se de la o reprezentare generală abstractă a rezolvării problemei, rezultând în final o reprezentare detaliată a acesteia.

Revenind la exemplul de mai sus, descrierea algoritmului de rezolvare a problemei, utilizându-se 'imbajul natural, poate f

Se parcurg următorii pași de 7 ori:

- se citește valoarea lui **x**
- dacă **x > -0.5**, atunci se calculează $f=2x-6$
  - altfel, dacă **x=-0.5** atunci valoarea funcției f este -7
  - altfel (adică **x<-0.5**) se calculează $f=-2x-8$
- se scrie valoarea lui f

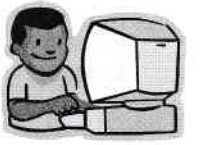
Limbajul natural nu permite o descriere suficient de riguroasă a algoritmilor. În plus, este necesar ca algoritmul creat să poată fi înțeles și utilizat și de o altă persoană. În acest scop, pentru reprezentarea algoritmilor se folosesc diverse modalități de descriere prin utilizarea limbajelor specializate convenționale. Fiecare dintre aceste limbaje trebuie să permită exprimarea cât mai naturală a raționamentului uman, să fie ușor de

înțeles și de folosit, și să reflecte caracteristicile limbajelor de programare de nivel înalt pentru a ușura codificarea algoritmilor.

Cele mai utilizate forme convenționale de reprezentare a algoritmilor sunt: schemele logice și limbajele pseudocod. Ambele variante oferă posibilitatea de a urmări cu claritate succesiunile posibile ale acțiunilor conținute de algoritm. Schemele logice sunt alcătuite din mai multe forme geometrice, fiecare simbolizând un anumit tip de acțiuni, în timp ce limbajele pseudocod utilizează cuvinte-cheie (cu sens predefinit), ce identifică operația ce se execută, și o serie de reguli speciale privind scrierea algoritmului.

În prezent, este frecvent utilizată scrierea algoritmilor cu ajutorul limbajelor pseudocod, acestea permițând transcrierea facilă a algoritmului în orice limbaj de programare. Vom utiliza în acest manual un limbaj de tip pseudocod în limba română.

3. Codificarea algoritmului într-un limbaj de programare.



Codificarea într-un limbaj de programare, înțeles și acceptat de calculator, se realizează aproape mecanic fără dificultate, obținându-se programul care va fi executat de calculator.

4. Testarea și validarea programului

Programul obținut prin codificarea algoritmului urmează a fi verificat, depistându-se eventualele erori de sintaxă (de scriere) și de semantică (de logică). Este posibil ca programul obținut să furnizeze rezultate corecte doar pentru anumite rezultate, în acest caz fiind necesar să depistăm și să eliminăm erorile de logică, lansând în execuție programul pentru mai multe seturi de date alese pe baza unor criterii specifice problemei.

1.3. Noțiunea de algoritm. Caracteristici

La începutul acestui capitol a fost prezentată mai pe larg noțiunea de algoritm. Reamintim că deși algoritmul este o realitate, el nu are o definiție riguroasă. Faptul că nu există o definiție nu constituie un impediment în utilizarea algoritmilor. Chiar și în matematică se folosesc frecvent noțiuni nedefinite fundamentale precum: mulțime, punct, spațiu.

Este necesar să precizăm din start faptul că nu există un algoritm pentru elaborarea oricărui algoritm.

Reamintim noțiunea de algoritm. Un **algoritm** reprezintă *o metodă de soluționare a problemelor, constând dintr-o mulțime finită și bine ordonată de operații bine definite, care realizează o prelucrare a datelor de intrare (informațiile primite), oferind rezultatele dorite (datele de ieșire).*

Fiecare propoziție inclusă în algoritm reprezintă o comandă ce urmează a fi executată de o persoană, o mașină sau un calculator. Fiecare comandă indică o operație (acțiune) care se aplică datelor de intrare ale algoritmului determinând prelucrarea acestora, toate comenzile specificând succesiunile posibile de transformări ale datelor ce conduc la determinarea rezultatelor - datele de ieșire.

Un algoritm este caracterizat prin următoarele atribute:

1. **Claritate**

   Prin **claritate** se înțelege formularea precisă, neinterpretabilă a operațiilor algoritmului. Algoritmul trebuie să fie bine definit, adică operațiile cerute să fie formulate riguros, fără ambiguitate.

2. **Universalitate**

   Universalitatea sau **generalitatea** algoritmilor este proprietatea acestora de a rezolva o întreagă clasă de probleme. Ea constă în posibilitatea aplicării algoritmului nu numai unor anumite date, ci unei întregi clase de date.

3. **Eficacitate**

   Eficacitatea constă în faptul că algoritmul permite obținerea unui rezultat. Considerăm următorul algoritm:

   Pas 1: Scrie un număr natural.
   Pas 2: Inversează cifrele numărului.
   Pas 3: Mergi la pasul 2.

   Este evident că acest proces de calcul este ineficace: nu conduce la nici un rezultat și, în plus, nu se termină după executarea unui număr finit de operații.

   Deducem că un algoritm este eficace dacă :

   a) Algoritmul conține un număr finit de instrucțiuni (operații, comenzi).
   b) Fiecare instrucțiune descrie o singură operație efectuabilă sau un număr finit de asemenea operații.
   c) Algoritmul este finit, în sensul că acesta conduce la un număr finit de rezultate după un număr finit de parcurgeri ale instrucțiunilor lui, deci tot acest proces are loc într-un timp finit de calcul.

Revenind la exemplul din secțiunea **1.2.**, se constată faptul că algoritmul descris este eficace deoarece permite obținerea unui rezultat. El calculează pentru oricare **7** valori reale ale lui **x**. Mai universal ar fi fost să calculăm valorile funcției pentru **n** valori reale, cu **n** număr natural nenul, citit, sau, mai general, să modificăm algoritmul astfel încât să primească ca date de intrare numărul de valori, valorile respective, precum și funcția ale cărei valori dorim să le calculăm.

1.4. Obiectele cu care lucrează algoritmii. Operații permise.

În general, obiectele supuse prelucrării în cadrul unui algoritm sunt **datele** (constante sau variabile), **expresiile** și **operațiile** care se execută în cadrul unui algoritm. În urma operațiilor indicate de algoritm, se realizează prelucrarea datelor de intrare, obținându-se la final datele de ieșire. Precizăm faptul că operațiile permise (instrucțiunile) dintr-un algoritm vor fi descrise în limbajul pseudocod, varianta în limba română.

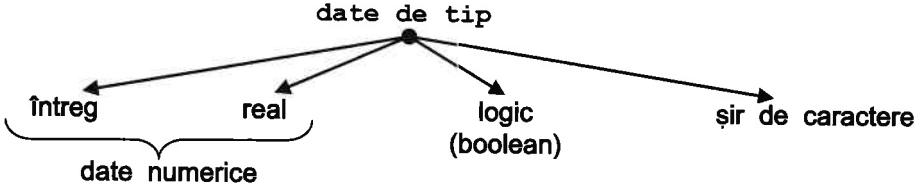
1.4.1. Date. Clasificare

Noțiunea de *date* are sensul de informații furnizate unui algoritm (program) în scopul obținerii rezultatelor corespunzătoare unei probleme. Reamintim că datele pot fi **date de intrare (informația prelucrată)** sau **date de ieșire (informația rezultată)**.

În algoritmul prezentat în secțiunea `1.2.`, datele de intrare sunt constituite din cele `7` valori reale ale lui `x`, iar datele de ieșire sunt reprezentate de cele `7` valori ale lui $f(x)$.

Datele cu care vom lucra în descrierea algoritmilor pot fi clasificate conform mai multor criterii.

a) Un prim criteriu de clasificare îl constituie ***tipul datelor.*** Conform acestui criteriu, are loc următoarea clasificare:



Observații:

- Datele de tip întreg sunt numere întregi. **Exemple:** `12345, -123, +13`.
- Datele de tip real sunt numere cu zecimale. Este necesară precizarea că în informatică mulțimea numerelor reale nu include și mulțimea numerelor iraționale ca în matematică, întrucât nici un calculator nu poate reține o infinitate de zecimale. Astfel, în informatică, numărul real are un număr finit de zecimale. Pentru scrierea unui număr real se va folosi punctul în loc de virgulă. **Exemplu:** `1.25` în loc de `1,25`.
- Datele logice pot avea valorile: `TRUE` (adevărat) sau `FALSE` (fals).
- Datele de tip șir de caractere sunt de fapt șirurile de caractere cuprinse între apostrofuri. **Exemple:** `'a'`, `'rezultat'` sau `'x='`.

b) Al doilea criteriu de clasificare îl constituie ***natura datelor***. Acest criteriu conduce la clasificarea datelor în **constante** și **variabile.**

- **Constantele** sunt date ale căror valori care nu se modifică pe parcursul aplicării algoritmului.

  Constantele întregi și reale sunt numere întregi și reale prezente în algoritm.

  Constantele logice sunt **TRUE (true)** și **FALSE (false).**

  Constantele de tip șir de caractere sunt șiruri de caractere cuprinse între apostrofuri; ele pot fi mesaje care trebuie să apară la execuția programului rezultat prin codificarea algoritmului.

  Caracteristica de bază a constantelor este faptul că ele sunt valori fixe și concrete utilizate în algoritm, fără a fi citite sau rezultate în urma efectuării unor calcule.

- **Variabilele** sunt date ale căror valori se modifică pe parcursul aplicării algoritmului.

  Variabilelor li se asociază un **nume**, care este un șir de caractere alfanumerice (litere sau cifre), în care primul caracter este obligatoriu o literă și în interiorul numelui este permisă apariția simbolului '_' (liniuța de subliniere).

  Orice algoritm lucrează cu variabile.

  Denumirea de variabilă provine din matematică. O variabilă este un nume (literă sau cuvânt, cu sau fără indici) ce semnifică orice valoare dintr-o mulțime dată. Mulțimea reprezintă domeniul variabilei respective. De exemplu, în expresia **x-5, x∈ℜ, x** este o variabilă reală al cărei domeniu este mulțimea numerelor reale.

  Numele fiecărei variabile este unic, conținutul ei (valoarea ei) se poate modifica pe parcursul algoritmului în funcție de operațiile ce se execută (de aici provine numele de variabilă).

  O variabilă poate reține un singur tip de date care poate fi întreg, real, logic sau șir de caractere.

Exemplu. Se consideră următoarea problemă pentru care vom construi un algoritm: "*Fie* **a** *și* **b** *două numere reale. Se cere să se determine soluția ecuației de gradul I:* **a·x+b=0.**"

Pentru a construi algoritmul, vom identifica mai întâi datele de intrare și datele de ieșire:

- **datele de intrare** sunt coeficienții ecuației, adică **a** și **b;**
- **datele de ieșire** sunt reprezentate doar de soluția reală **x** a ecuației, dacă ea există, altfel vom scrie un mesaj corespunzător.

Observăm că ecuația din enunț este echivalentă cu: **a·x=-b**. Vom construi un algoritm care să rezolve această ecuație:

```
citește a,b {numere reale}
┌dacă a=0
│     atunci
│           ┌dacă b=0
│           │  atunci
│           │     scrie 'Identitate. x∈ℜ'
│           │  altfel
│           │     scrie 'x∈Ø'
│           └■
│     altfel
│           scrie 'x=',-b/a
└■
```

Constantele din acest algoritm sunt: constantă numerică **0** și cele trei constante șiruri de caractere: **'Identitate. x∈ℜ', 'x∈Ø','x='.**

Pentru a respecta proprietatea de universalitate a oricărui algoritm, datele de intrare vor fi reprezentate prin intermediul variabilelor reale **a** și **b**. Analog, soluția problemei (data de ieșire) va fi dată de valoarea conținută de variabila **x** la finalul algoritmului. Ele vor realiza abstractizarea problemei, astfel încât, prin codificarea într-un limbaj de programare a algoritmului, programul va furniza soluția ecuației nu numai pentru un set de valori date ci pentru orice valori ale variabilelor **a** și **b** (de exemplu, dacă **a=3, b=9** soluția este **x=-3**; dacă modificăm valorile astfel încât **a=10, b=-5** soluția va fi **x=0.5**).

În plus, calculatorul va rezerva în memoria sa fiecărei variabile un singur spațiu de memorie, cu o dimensiune corespunzătoare memorării valorii fiecărei variabile. Într-un astfel de spațiu poate fi reținută la un moment dat o singură valoare.

Putem considera că **o variabilă** reprezintă un ansamblu de patru elemente:

- numele variabilei;
- tipul variabilei;
- valoarea ei la un moment dat;
- locul din memoria calculatorului unde este reținută valoarea variabilei (adică adresa ei în memorie).

Să ne imaginăm că o persoană își deschide un cont curent personal la o bancă. Contul va fi identificat printr-un număr unic – putem considera acest număr ca fiind adresa contului pe baza căreia persoana va avea acces la sumele depuse în cont. În același timp, contul persoanei este identificat direct prin numele persoanei, presupunând că numele sunt unice. Astfel, putem considera faptul că contul persoanei este o variabilă reală cu numele dat de numele persoanei, adresa contului fiind numărul lui de ordine, iar valoarea variabilei este suma curentă existentă în acest cont la momentul curent. Evident această sumă este un număr real care se modifică în funcție de operațiunile realizate de persoana deținătoare a contului.