MINISTERUL EDUCAŢIEI NAŢIONALE

Emanuela Cerchez
Marinel Şerban

Informatică

Profilul real

Specializarea:
matematică–informatică, ştiinţe ale naturii

Manual pentru clasa a IX - a

EDITURA DIDACTICĂ ŞI PEDAGOGICĂ, R.A.

- **sistemul de operare** reprezintă al doilea nivel al sistemului *software* și este constituit dintr-un ansamblu de programe care coordonează toate activitățile calculatorului;
- **programele de aplicații** reprezintă nivelul superior, cel mai apropiat de utilizator, constituit din totalitatea programelor destinate rezolvării unor probleme specifice.

1.2. Informatica și societatea

Prin inventarea calculatorului electronic s-a produs o adevărată revoluție a societății contemporane, care poate fi comparată, prin impactul produs, cu Revoluția Industrială. Astăzi nu există nici un domeniu al activității umane în care să nu fie utilizat calculatorul. Calculatorul a devenit nu doar o unealtă indispensabilă activității cotidiene, ci instrumentul care permite dezvoltarea în orice domeniu de activitate a unor tehnici avansate, inițiatoare ale progresului în domeniu.

Primele calculatoare aveau putere mică de calcul, erau extrem de mari și extrem de costisitoare. Din acest motiv, au fost folosite inițial doar în institute de cercetare și universități. Dar chiar dacă, privite din perspectiva mileniului III, acele calculatoare par ridicole, încă de la început s-au dovedit a fi instrumente utile, motiv pentru care au fost investite fonduri enorme pentru dezvoltarea acestui domeniu, industria calculatoarelor devenind domeniul cu cea mai rapidă evoluție.

Dezvoltarea industriei calculatoarelor a fost însoțită de dezvoltarea unei noi industrii – cea a producătorilor de *software*. S-au înființat mari companii producătoare de programe pentru diverse domenii de activitate, din ce în ce mai mulți oameni fiind implicați în activități legate de programare, de testare a programelor, de utilizare a acestora în domeniul propriu de activitate. O analiză, chiar sumară, a ofertelor de pe piața mondială a locurilor de muncă reflectă proporția dominantă în care sunt solicitați specialiști în informatică, precum și specialiști în alte domenii, dar cu solide cunoștințe de utilizare a calculatoarelor. Practic, de la apariția calculatoarelor s-au înregistrat transformări treptate în specificul muncii umane. Omul este eliberat în mare măsură de muncile care necesită un efort fizic mare, de activitățile de rutină, devenind predominant rolul său creator.

Pentru a ilustra aceste afirmații, să urmărim modul în care calculatorul este utilizat astăzi în diverse domenii ale activității umane.

Cercetarea fundamentală în fizică, chimie, biologie s-a dezvoltat extrem de rapid în ultimele decenii, în principal datorită dezvoltării tehnologiei.

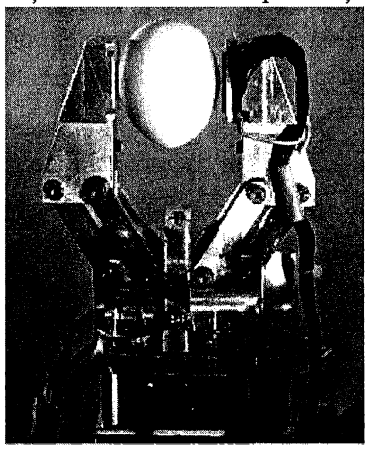
În fizică un impact deosebit au avut dezvoltarea electronicii și a calculatoarelor, aplicațiile energiei nucleare și acceleratoarele de particule de mare energie. Fizicienii au fost primii care au utilizat calculatorul, la **Los Alamos National Laboratory,** un institut de cercetări înființat de Departamentul de Energie al S.U.A. Aici, în timpul celui de al II-lea război mondial, *John von Neumann*,

creatorul arhitecturii actuale a calculatoarelor electronice, a fost cooptat consultant în proiectul bombei atomice.

În chimie, sintetizarea unor substanțe chimice, prepararea unor noi compuși precum și predicția efectelor lor se realizează cu ajutorul calculatorului și al unor programe complexe de modelare. Faptul că astăzi pot fi sintetizați hormoni, enzime, material genetic identic cu cel al ființelor vii se datorează în cea mai mare măsură puternicelor facilități de modelare oferite de calculatoarele și programele moderne. Analizele chimice și medicale de mare precizie se realizează numai cu ajutorul calculatorului. În plus, în domeniul medicinei s-au dezvoltat noi tehnici chirurgicale asistate de calculator, precum și noi tehnici de investigație medicală, cum ar fi tomografiile computerizate.

Au fost dezvoltate sistemele expert, programe complexe care au rolul de a lua decizii și de a rezolva probleme într-un domeniu specific, cum ar fi medicina sau contabilitatea, pe baza unor reguli și informații furnizate de specialiști în domeniu. Aceste programe acționează într-un mod similar experților umani și au două componente principale: o bază de cunoștințe (date și legi) specifice domeniului la care se aplică și un motor de inferență logică, ce reprezintă mecanismul prin care sistemul expert „raționează", deci formulează concluzii sau ia decizii pe baza cunoștințelor de care dispune. Totuși, pentru a fi asemenea unui expert uman, aceste programe ar trebui să aibă o calitate în plus, aceea de a învăța. Acest aspect constituie una dintre cele mai importante preocupări ale unui domeniu de vârf în informatică, Inteligența Artificială[1].

Dar cercetarea fundamentală nu este singurul domeniu în care este utilizat calculatorul, ci toate aspectele activității umane sunt influențate de acesta. În producția industrială, procesul tehnologic este controlat de calculatoare speciale, denumite calculatoare de proces. Acestea sunt calculatoare analogice, care preiau ca date de intrare rezultatele măsurătorilor unor mărimi caracteristice procesului tehnologic (temperatură, presiune etc.) și iau deciziile necesare pentru reglarea și desfășurarea procesului tehnologic în condiții optime. Calculatoarele nu sunt implicate numai în controlul producției industriale, ci chiar în producția propriu-zisă. Există mașini complexe, denumite roboți, care sunt coordonate de calculatoare și care au rolul de a executa un set de operații specifice, mult mai rapid, mai ieftin și mai precis decât un om. În plus, roboții pot lucra în zone care sunt



Acest robot este capabil să țină un ou, fără să îl spargă.

1. Inteligența Artificială este un concept care definește capacitatea unui obiect artificial de a îndeplini funcții caracteristice gândirii umane.

periculoase pentru sănătatea oamenilor, în zone toxice, în adâncul oceanelor sau în spațiul cosmic. Roboții preiau prin intermediul unor senzori informații din mediul înconjurător și sunt programați pentru a fi capabili să își adapteze activitatea în funcție de variabilele de mediu.

Activitatea de proiectare, fie că se referă la șuruburi, case, rochii, mașini sau nave spațiale, se realizează de asemenea cu ajutorul calculatorului. Există pachete de programe specializate pentru aceasta, care constituie un domeniu al informaticii denumit **CAD** (**C**omputer **A**ided **D**esign – Proiectare Asistată de Calculator).

Acestea sunt însă exemple mult prea îndepărtate, pentru unii dintre noi, de viața de zi cu zi. Calculatoarele sunt implicate în orice activitate a noastră.

Urmărim un film? Cu maximă probabilitate scenele cele mai spectaculoase din film sunt realizate pe calculator. De exemplu, scena cea mai dramatică a filmului „Titanic", scufundarea vaporului, când sute de oameni sar cu disperare de la mare înălțime, a fost realizată pe calculator, fără ca nimeni să își rișteviața. Regizorul filmului a pus actorii să acționeze în diverse situații, preluând mișcările lor prin intermediul unor senzori. Mișcările au fost transferate asupra unor creaturi artificiale, fiecare reprezentare digitală fiind coordonată în mod individual pe calculator.

Ascultăm o reclamă? Mixajul a fost cu siguranță realizat pe calculator.

Vrem să știm cum va fi vremea? Predicția vremii este posibilă numai cu ajutorul calculatorului. O parte din informațiile necesare pentru prevederea vremii, de exemplu cele preluate din straturile superioare ale atmosferei, sunt înregistrate direct de către calculatoare specializate. Modelarea datelor înregistrate presupune efectuarea rapidă a unor calcule complexe, bazate pe ecuații matematice. Interpretarea cu cât mai multă acuratețe a datelor se realizează, de asemenea, cu programe specializate, deci tot cu ajutorul calculatorului.

Vrem să plătim impozitele? La Administrația Financiară se lucrează pe calculator, existând baze de date de evidență a locuințelor, automobilelor etc.

Vrem să ne schimbăm buletinul? Începând cu fotografia care va fi preluată cu ajutorul unei camere digitale, până la înregistrarea în evidențele computerizate ale Poliției, totul se realizează pe calculator.

Ar fi mai greu să dăm un exemplu unde nu se utilizează calculatorul sau, mai exact, unde nu trebuie să se utilizeze calculatorul.

În domeniul educației, calculatorul este o adevărată provocare pentru învățământul tradițional. Informatica nu reprezintă doar o nouă disciplină de învățământ, ci produce un puternic impact asupra metodelor de învățare de la orice disciplină, din orice arie curriculară. De exemplu, accesul la Internet sau o enciclopedie multimedia reprezintă o importantă resursă educațională, oferind un spațiu informațional vast pentru toate domeniile de studiu. Calculatorul este privit ca un instrument indispensabil oricărei discipline, noile tehnologii multimedia

reprezentând calea de a comunica idei, de a sistematiza informații, de a dezvolta proiecte. Pentru orice disciplină școlară există programe educaționale, care permit individualizarea învățării. **CAI** (**C**omputer **A**ssisted **I**nstruction – Învățarea Asistată de Calculator) reprezintă o nouă metodă de învățare, prin care profesorul pune la dispoziția elevului, pe calculator, materiale de curs, programe de învățare, teme de lucru, teste. Elevii studiază individual, în ritmul propriu, materialele respective. Acum calculatorul reprezintă sursa principală de informații, nu profesorul, rolul acestuia fiind de a coordona activitatea elevilor. Studiile realizate în legătură cu învățarea asistată de calculator au demonstrat că productivitatea învățării crește, elevii dobândesc o cantitate mai mare de informații într-un timp mai scurt, se dezvoltă curiozitatea științifică, se structurează mult mai bine motivațiile și aspirațiile individuale și, nu în ultimul rând, elevilor le place să utilizeze calculatorul.

Noi probleme în era informațională

Utilizarea calculatorului în orice domeniu de activitate a produs o adevărată revoluție informațională. Așa cum era de așteptat, apar alte „puncte de fugă" în relațiile sociale, apar noi probleme, de o importanță deosebită. O astfel de problemă ar fi posibilele atentate la dreptul fiecăruia la intimitate, prin accesarea neautorizată a unor informații personale sau prin monitorizarea permanentă a spațiilor publice.

O altă problemă, de maximă importanță, este reprezentată de asigurarea drepturilor de autor asupra programelor sau asupra informațiilor de pe Internet. *Copyright*-ul (drepturile de autor) este un sistem prin care sunt protejate valorile intelectuale. *Copyright*-ul reprezintă drepturile de proprietate care derivă automat la crearea unei opere, în orice domeniu – literar, muzical, plastic, *software* etc. Durata de valabilitate a drepturilor de autor variază de la un sistem legislativ la altul, dar în general este egală cu durata vieții + 50 ani. În România, conform legii nr. 8/26.03.1996 privind dreptul de autor și drepturile conexe, drepturile de autor sunt valabile pe întreaga durată a vieții autorului, iar după moartea acestuia se transmit prin moștenire, pe o durată de 70 de ani (art.25). Capitolul IX (art. 72-81) din legea nr. 8/26.03.1996 stipulează dispoziții speciale referitoare la programele pe calculator. Conform prevederilor din acest capitol, autorul unui program (indiferent dacă este program de aplicație sau sistem de operare, cod sursă sau cod obiect) are dreptul exclusiv de a realiza sau autoriza reproducerea integrală sau parțială a programului, traducerea, adaptarea, precum și difuzarea originalului sau a copiilor acestuia, sub orice formă, inclusiv închiriere. Utilizatorii autorizați ai programului nu au dreptul de a transmite altor persoane dreptul de utilizare a programului[2]. Toate informațiile pe care un utilizator le dobândește prin studierea

2. Pentru o prezentare detaliată a tuturor aspectelor legate de drepturile de autor asupra programelor pe calculator, studiați *Legea copyright-ului pentru toți*, Editura „Soluții Informaționale", București, 1996.

și testarea funcționalității unui program nu pot fi utilizate pentru producerea sau comercializarea unui alt program, pe principii fundamental similare.

Din păcate, legea *copyright*-ului nu este întotdeauna respectată, pirateria fiind un fenomen frecvent, în special în țările slab dezvoltate. În multe țări, chiar dacă acestea au semnat convenții internaționale referitoare la respectarea drepturilor de autor, sunt difuzate, fie în formă tipărită, fie în format digital (înregistrate pe CD-uri), copii neautorizate ale unor opere de referință (literare, științifice, muzică sau film). Copierea programelor sau a unor lucrări în format digital este facilă, rapidă și practic imposibil de detectat. Deși creatorii de programe folosesc sisteme de protecție din ce în ce mai sofisticate, există specialiști în „spargerea" acestor sisteme de protecție, denumiți *cracker*-i, utilizarea *soft*-ului fără licență fiind o adevărată epidemie.

La ora actuală, legislația și contractele conțin prevederi clare referitoare la modul de utilizare a *soft*-ului, dar nu există un sistem perfecționat de control al respectării acestor prevederi. Atât în Europa, cât și în S.U.A. se derulează diverse proiecte de dezvoltare a unor sisteme electronice care să permită controlul utilizării autorizate a programelor și informațiilor în format digital.

Evoluții viitoare

Impactul utilizării calculatoarelor asupra societății s-a dovedit a fi deosebit de profund. În mod evident, informatica joacă un rol esențial în îmbunătățirea situației economice și a calității vieții oricărei națiuni și din acest motiv pentru multe guverne aceasta reprezintă o prioritate națională. Includerea informaticii ca disciplină de studiu în învățământ asigură pregătirea specialiștilor de care este nevoie pentru a aplica ceea ce ne învață istoria ultimelor decenii: soluția supraviețuirii și a succesului constă în dezvoltarea tehnologiilor informaționale moderne și aplicarea lor în toate domeniile activității umane.

Întrebări recapitulative

1. Dați exemple de domenii în care este utilizat calculatorul și descrieți modul lui de utilizare!
2. Ce este un robot? Care sunt avantajele utilizării roboților? Scrieți un scurt eseu despre evoluția roboților! Analizați modul în care calculatorul a influențat specificul muncii umane, precum și dinamica structurii ofertelor de pe piața locurilor de muncă!
3. Ce înțelegeți prin *copyright*? Este legală utilizarea unui program fără licență? Ce sisteme de protecție a produselor *soft* ați întâlnit?
4. Descrieți modul în care calculatorul influențează viața voastră personală!
5. Descrieți într-un eseu modul în care calculatorul influențează comunicarea interumană!

2. NOȚIUNEA DE ALGORITM

2.1. Ce este un algoritm?

Despre algoritmi auzim astăzi din ce în ce mai des, în contexte diferite. Conceptul de algoritm nu este nou. Termenul *algoritm* derivă din numele unui matematician persan, *Abu Ja'far Mohammed ibn Musa al Khowarizmi*[1] (cca. 825 e.n.), care a scris o carte cunoscută sub denumirea latină de *„Liber algorithmi"*[2].

Matematicienii Evului Mediu înțelegeau prin algoritm o regulă pe baza căreia se efectuau calcule aritmetice. Ulterior, termenul de algoritm a circulat într-un sens restrâns, exclusiv în domeniul matematicii. O dată cu dezvoltarea calculatoarelor cuvântul algoritm a dobândit o semnificație aparte, astfel încât astăzi gândirea algoritmică s-a transformat, dintr-un instrument specific matematicii, într-o modalitate fundamentală de abordare a problemelor în diverse domenii.

Un ***algoritm*** reprezintă o metodă de rezolvare a problemelor de un anumit tip.

A rezolva o problemă înseamnă a obține, pentru anumite *date de intrare*, rezultatul problemei (*date de ieşire*):

Algoritmul este constituit dintr-o succesiune de operații care descriu, pas cu pas, modul de obținere a datelor de ieşire, plecând de la datele de intrare.

Se pot descrie algoritmi pentru rezolvarea problemelor din orice domeniu de activitate. De exemplu, orice rețetă de bucătărie poate fi considerată un algoritm prin care, plecând de la materiile prime, obținem printr-o succesiune finită de operații produsul finit.

Exemplul 1

Presupunând că dispunem de un aragaz, o tigaie, 2 ouă, sare şi 200 ml ulei, să pregătim ochiuri.

„Date" de intrare: ouă, ulei, sare.
„Date" de ieşire: ochiuri.

---

1. În traducere exactă, *„al Khowarizmi"* înseamnă *„din oraşul Khowarizm"*. Astăzi acest oraş se numeşte Khiva şi se găseşte în Uzbekistan.
2. *Abu Ja'far Mohammed ibn Musa al Khowarizmi* este cunoscut şi datorită cărții *„Kitab al-jabr wa'l muqabala"*, o carte de exerciții matematice, din denumirea căreia provine termenul *algebră*.

Pas 1: Se pune tigaia pe foc.
Pas 2: Se toarnă uleiul în tigaie.
Pas 3: Aşteptăm până când se încinge uleiul.
Pas 4: Spargem cu îndemânare ouăle în tigaie.
Pas 5: Aşteptăm până când ouăle se rumenesc.
Pas 6: Dacă nu ținem regim, adăugăm sare.

Observați că am descris o succesiune de paşi, prin care, pentru orice „date" de intrare (ulei, sare, ouă), obținem rezultatul dorit (ochiuri). Fiecare pas constă din operații culinare specifice, care se execută în ordinea în care sunt specificate.

Exemplul 2

Să analizăm un alt exemplu, cu care sunteți de asemenea familiarizați: să se rezolve o ecuație de forma $ax+b=0$, cu $a, b \in \mathbf{R}$.

Date de intrare: $a, b \in \mathbf{R}$
Date de ieşire: $x \in \mathbf{R}$, soluția ecuației, sau un mesaj adecvat
Pas 1: Citeşte datele de intrare a şi b
Pas 2: Dacă $a \neq 0$ atunci Scrie `Soluția ecuației este x= -b/a`
altfel
Dacă $b=0$ atunci
Scrie `Ecuația este nedeterminată`
altfel
Scrie `Ecuația este imposibilă`

Această succesiune de paşi rezolvă ecuația de forma specificată, pentru orice valori ale datelor de intrare, prin urmare este un algoritm.

La matematică, ați învățat numeroşi alți algoritmi, chiar dacă nu i-ați denumit astfel: algoritmul de adunare a două numere, algoritmul de extragere a rădăcinii pătrate dintr-un număr real, algoritmul de determinare a celui mai mare divizor comun a două numere naturale etc.

Exerciții

1. Descrieți, punând în evidență succesiunea paşilor, algoritmul de tratare a gripei!
2. Descrieți, punând în evidență succesiunea paşilor, algoritmul de rezolvare a unei ecuații de forma $ax^2+bx+c=0$, cu a, b, c numere reale date.
3. Descrieți, punând în evidență succesiunea paşilor, algoritmul de calcul al mediei semestriale a unui elev la un obiect la care se susține teză.
4. Descrieți, punând în evidență succesiunea paşilor, un algoritm de copiere a unei zone de text dintr-un document *Word* într-un alt document *Word*.
5. Descrieți, punând în evidență succesiunea paşilor, algoritmul pe care l-ați aplicat la ultima experiență efectuată în laboratorul de chimie.

2.2. Proprietăți caracteristice ale algoritmilor

Exemplele precedente generează în mod firesc două întrebări:

- *Pentru orice problemă există un algoritm de rezolvare?*

Răspunsul este NU! Există probleme pentru care se poate demonstra (lucru dificil!) că nu există algoritmi de rezolvare, dar și probleme pentru care nici nu s-a demonstrat că nu admit o metodă de rezolvare algoritmică, dar nici nu s-a descoperit (încă!) soluția algoritmică.

- *Orice succesiune de pași reprezintă un algoritm?*

Din nou, răspunsul este NU! Pentru a fi un algoritm, secvența trebuie să îndeplinească trei condiții:

1. **Claritate** – la fiecare moment, operația care urmează a fi executată este unic determinată, definită și realizabilă (adică poate fi efectuată la momentul respectiv, cu mijloacele disponibile).

De exemplu, secvența *„Dacă plouă stau acasă sau merg la cinema"* nu este clară, deoarece, în cazul în care plouă, operația care se execută nu este unic determinată.

Sau să presupunem că dorim să obținem un număr natural, care se poate scrie ca sumă de pătrate. Secvența de mai jos „scrie x^2+y^2" nu este clară, deoarece nu putem calcula valoarea x^2+y^2, deoarece nu cunoaștem valorile lui x și y.

2. **Generalitate** (universalitate) – o secvență de pași reprezintă un algoritm de rezolvare a unei probleme dacă obține date de ieșire (rezultate) pentru orice date de intrare specifice problemei.

Secvența de pași prezentată în exemplul 2 este generală, deoarece conduce la rezolvarea ecuației $ax+b=0$ pentru orice valori reale ale coeficienților a și b.

Dar, dacă am fi descris o secvență de pași care să rezolve numai ecuația $x+2=0$, aceasta nu ar fi fost un algoritm!

3. **Finitudine** – rezultatele problemei se obțin după un număr finit de pași.

De exemplu, problema *„Să se determine toate zecimalele numărului π"* nu are o soluție algoritmică, deoarece π este un număr irațional, care are o infinitate de zecimale. Dar dacă am enunța problema astfel: *„Fie n un număr natural dat. Să se determine primele n zecimale ale numărului π"*, această problemă admite o soluție algoritmică, deoarece primele n zecimale se pot obține după un număr finit de pași.

În concluzie, deși nu putem defini cu rigurozitate noțiunea de algoritm, putem descrie mai detaliat această noțiune astfel:

Un ***algoritm*** este constituit dintr-o succesiune clară de operații realizabile, care au ca scop obținerea într-un timp finit a rezultatelor unei probleme, pentru orice set de date de intrare.

2.3. Etapele rezolvării unei probleme

Rezolvarea unei probleme constituie un proces complex, care comportă mai multe etape.

1. *Analiza problemei* în scopul stabilirii datelor de intrare, precum şi a rezultatelor pe care trebuie să le obținem prin rezolvarea problemei.
2. *Elaborarea unui algoritm* de rezolvare a problemei.
3. *Implementarea* algoritmului într-un limbaj de programare.
4. *Verificarea corectitudinii* algoritmului propus.

   Un prim pas constă în *testarea programului* pe diverse seturi de date de test. Seturile de date de test trebuie elaborate cu atenție, astfel încât să acopere, pe cât posibil, toate variantele de execuție a algoritmului, inclusiv situații de excepție, şi să verifice dacă fiecare subproblemă a problemei date este rezolvată corect (dacă este posibil, se va testa separat fiecare modul de program).

   Testarea poate pune în evidență, eventual, omisiuni sau erori de concepție a algoritmilor, dar nu garantează corectitudinea algoritmului. Pentru aceasta ar trebui să testăm algoritmul pe toate seturile posibile de date de intrare, ceea ce este practic imposibil. Din acest motiv, se impune utilizarea unor metode formale de demonstrare a corectitudinii algoritmului, etapă de obicei deosebit de laborioasă, necesitând un aparat matematic complex.
5. *Analiza complexității algoritmului.*

   În general, există mai mulți algoritmi de rezolvare a unei probleme date. Pentru a alege cel mai bun algoritm, trebuie să analizăm aceşti algoritmi în scopul determinării eficienței lor şi, pe cât posibil, a optimalității lor.

   Eficiența unui algoritm se evaluează din două puncte de vedere:

   1. din punctul de vedere al spațiului de memorie necesar pentru memorarea valorilor variabilelor care intervin în algoritm (complexitate spațiu);
   2. din punctul de vedere al timpului de execuție (complexitate timp).

Complexitatea spațiu o vom analiza cu precădere atunci când vom transpune algoritmul într-un limbaj de programare.

Pentru a estima *complexitatea timp* vom presupune că se lucrează pe un calculator „clasic", în sensul că o singură instrucțiune este executată la un moment dat. Astfel, timpul necesar execuției programului depinde de numărul de operații elementare efectuate de algoritm.

Observație

Elaborarea algoritmilor nu este un proces liniar, adeseori este necesar să revenim la o anumită etapă şi să o repetăm. De exemplu, după ce am demonstrat

corectitudinea algoritmului şi am analizat eficienţa sa, ne putem pune problema de a optimiza algoritmul sau numai implementarea sa, caz în care trebuie să revenim la cea de a doua etapă, de proiectare a algoritmilor şi de scriere a codului, etapă urmată în mod necesar de teste de corectitudine, eliminare a erorilor (*bug*-urilor), demonstraţii de corectitudine, teste de determinare a complexităţii, analiza teoretică a complexităţii etc.

2.4. Date

Orice algoritm lucrează cu date: *date de intrare* (datele pe care trebuie să le primească un algoritm din exterior), *date de ieşire* (datele pe care trebuie să le furnizeze algoritmul în exterior), precum şi *date de manevră* (date temporare, necesare algoritmului pentru a obţine datele de ieşire pe baza datelor de intrare).

Datele cu care lucrează algoritmii pot fi clasificate din mai multe puncte de vedere. O primă clasificare a datelor, în funcţie de posibilitatea de a-şi modifica valoarea, este:

1. **Constante** – date care nu îşi modifică valoarea; de exemplu: `10`, `3.14`, `"sir de caractere"`, `'A'`, `fals`.
2. **Variabile** – date care îşi modifică valoarea.

O variabilă poate fi referită printr-un nume (o succesiune de litere şi cifre, primul caracter fiind obligatoriu literă) şi are asociată o valoare. Numele unei variabile nu se schimbă pe parcursul algoritmului, dar valoarea acesteia se poate modifica.

De exemplu, pentru rezolvarea ecuaţiei de forma `ax+b=0`, am utilizat două variabile `a` şi `b`. Prin operaţia *„Citeşte datele de intrare a şi b"*, acestora li se asocia câte o valoare reală, introdusă de la tastatură. Utilizarea acestor două variabile era strict necesară, pentru a respecta generalitatea algoritmului. Dacă am fi utilizat două valori constante (de exemplu, `2` şi `3.5`), secvenţa de operaţii ar fi rezolvat numai ecuaţia `2x+3.5=0`, deci nu ar fi avut utilitate.

Observaţi că la începutul algoritmului am specificat (am **declarat**) faptul că `a` şi `b` sunt numere reale. Această declaraţie este necesară, pentru a cunoaşte natura valorilor care pot fi asociate celor două variabile şi, ca urmare, operaţiile permise cu acestea. Spunem că am declarat **tipul** variabilei respective. *O variabilă poate reţine numai valori de tipul declarat.*

În funcţie de valoarea lor, datele pot fi clasificate astfel:
1. **Date numerice** – au ca valori numere (naturale, întregi sau reale);
2. **Date alfabetice** – au ca valori caractere sau şiruri de caractere;
3. **Date logice** – au valoarea `adevărat` sau `fals`.

2.5. Expresii

O *expresie* este constituită dintr-o succesiune de operanzi, conectați prin operatori. Un *operand* poate fi o constantă, o variabilă, sau o expresie încadrată între paranteze rotunde. *Operatorii* desemnează operațiile care se execută asupra operanzilor. Operatorii care pot fi utilizați într-o expresie depind de tipul operanzilor (numerici întregi, numerici reali, caractere, șiruri de caractere sau logici).

Evaluarea unei expresii presupune calculul valorii expresiei, prin înlocuirea valorilor variabilelor care intervin ca operanzi în expresie și efectuarea operațiilor specificate de operatori.

Vom prezenta trei categorii de operatori.

Operatori aritmetici

Operatorii aritmetici definesc o operație aritmetică și pot fi clasificați astfel:

1. *operatori aritmetici multiplicativi*: * (înmulțire), /(împărțire), % (restul împărțirii întregi).

Operatorul de împărțire (/) are un efect diferit, în funcție de tipul operanzilor. Dacă ambii operanzi sunt întregi, se face împărțire întreagă (se obține ca rezultat un număr întreg, care este câtul împărțirii primului operand la cel de-al doilea). Mai exact, fie `a` și `b` două variabile întregi. Expresia `a/b` are ca valoare câtul împărțirii întregi a lui `a` la `b`. Dacă, de exemplu, `a` are valoarea `7` și `b` are valoarea `2`, expresia `a/b` are valoarea `3`.

Dacă cel puțin unul dintre operanzi este real, se face împărțire reală (se obține ca rezultat un număr real). De exemplu, dacă `a` și `b` sunt două variabile reale, iar `a` are valoarea `7` și `b` are valoarea `2`, expresia `a/b` are valoarea `3.5`.

Operatorul % se poate aplica numai asupra operanzilor întregi.

2. *operatori aritmetici aditivi*: + (adunare) și – (scădere).

Operatorii aritmetici aditivi și multiplicativi sunt *binari* (acționează asupra a doi operanzi). Operatorii aritmetici se pot aplica numai operanzilor numerici. Rezultatul evaluării unei expresii aritmetice este numeric (întreg sau real, în funcție de operanzi și operatori).

Operatori relaționali

Operatorii relaționali descriu relația de ordine sau de egalitate dintre cei doi operanzi: $<$ (mai mic), $>$ (mai mare), $\leq$ (mai mic sau egal), $\geq$ (mai mare sau egal), $=$ (egal), $\neq$ (diferit).

Operatorii relaționali sunt operatori binari și se pot aplica numai operanzilor numerici, logici (`fals < adevărat`) și caractere (ordinea caracterelor fiind determinată de codurile lor ASCII[3]).

Valoarea unei expresii relaționale este întotdeauna de tip logic (deci poate fi `adevărat` sau `fals`).

Operatori logici

Operatorii logici definesc o operație logică: *negație logică* – `!`; *conjuncție logică* – `și`; *disjuncție logică* – `sau`. Operatorul ! este unar, operatorii și, sau sunt operatori binari. Efectul acestor operatori este cel uzual, învățat la logică matematică. Îl reamintim în tabelul următor:

x	y	! x	x sau y	x și y
fals	fals	adevărat	fals	fals
fals	adevărat	adevărat	adevărat	fals
adevărat	fals	fals	adevărat	fals
adevărat	adevărat	fals	adevărat	adevărat

Operatorii logici se pot aplica operanzilor logici. Valoarea unei expresii logice este de tip logic.

Greșeli frecvente în scrierea expresiilor

1. O greșeală frecventă în utilizarea operatorilor relaționali este utilizarea înlănțuită a acestora.
   De exemplu, să presupunem că este necesară o expresie care să aibă valoarea `adevărat` dacă valorile a trei variabile întregi `a`, `b`, `c` sunt egale. Este greșit să scriem `a=b=c`. Corect ar fi, de exemplu, `a=b și b=c`.
2. Frecvent sunt „uitate" paranteze, de exemplu pentru a calcula media aritmetică a variabilelor `a` și `b`, este scris `a+b/2` în loc de `(a+b)/2`.
3. O greșeală des întâlnită este omiterea operatorul de înmulțire „`*`". De exemplu, este scris `a+2b` în loc de `a+2*b`.

Evaluarea unei expresii

În procesul de evaluare a unei expresii se respectă regulile de bază, învățate la matematică (în primul rând se evaluează expresiile dintre parantezele rotunde; apoi se execută operațiile în ordinea priorității lor; dacă există mai multe operații cu aceeași prioritate, se execută în ordine, în funcție de asociativitatea lor). Prioritatea operatorilor este (1 fiind considerată prioritatea maximă):

---

3. Vezi anexa 1.

Prioritate	Operatori	Simbol	Asociativitate
1	Negația logică	!	de la dreapta la stânga
2	Aritmetici multiplicativi	*, /, %	de la stânga la dreapta
3	Aritmetici aditivi	+, -	de la stânga la dreapta
4	Relaționali	<, >, ≤, ≥, =, ≠	de la stânga la dreapta
5	Conjuncție logică	şi	de la stânga la dreapta
6	Disjuncție logică	sau	de la stânga la dreapta

Exemple

1. Fie `x` un număr natural. Expresia `x%10` are ca valoare restul împărțirii numărului `x` la `10` (care este de fapt ultima cifră a lui `x`).
2. Fie `n` un număr natural. Expresia `n*(n+1)/2` reprezintă suma numerelor naturale mai mici sau egale cu `n`.
3. Să presupunem că în variabila `R` memorăm raza unui cerc, iar constanta π o aproximăm[4] la valoarea `3.14`.
   Expresia aritmetică `2*3.14*R` are ca valoare lungimea cercului de rază `R` (cu aproximație).
   Expresia aritmetică `3.14*R*R` are ca valoare aproximativă aria cercului de rază `R`.
4. Fie `x` un număr natural. Expresia `x<3` are valoarea `adevărat` dacă `x` este `0`, `1` sau `2` şi `fals`, dacă `x≥3`.
5. Să scriem o expresie logică a cărei valoare să fie `adevărat` dacă şi numai dacă numărul întreg memorat într-o variabilă `x` este un număr par.
   Un număr întreg este par dacă el este divizibil cu 2, adică restul împărțirii numărului la 2 este 0. Prin urmare, expresia `x%2=0` are valoarea `adevărat` dacă şi numai dacă `x` este par.
6. Să scriem o expresie logică a cărei valoare să fie `adevărat` dacă şi numai dacă valorile întregi ale variabilelor `x` şi `y` sunt strict pozitive: `x>0 şi y>0`.

2.6. Probleme propuse

1. Să considerăm `a`, `b` şi `c` trei variabile întregi, având valorile `4`, `6` şi respectiv `2`. Evaluați expresia `(1+2*a/b+a+2*b/3*c)/2+a`.

2. Fie `n` o variabilă întreagă cu valoarea `32735`. Evaluați expresia `n%10+n/10%10+n/100%10`.

---

4. Numerele iraționale, care au o infinitate de zecimale, nu pot fi reprezentate în memoria unui calculator decât cu aproximație (reținând un număr convenabil de zecimale exacte).

3. Să considerăm a, b şi c trei variabile întregi, având valorile 5, 7 şi respectiv 8. Evaluaţi expresia a<=b şi b<c sau c%2=1.

4. Fie a o variabilă întreagă cu valoarea 5 şi b o variabilă întreagă cu valoarea 10. Ce valoare are expresia a%2=0 şi b%2=0 sau a%2=1 şi b%2=1? Ce semnificaţie are această expresie?

5. Considerăm că a, b, c sunt variabile întregi, iar x şi y sunt variabile reale. Care sunt greşelile din următoarele expresii?

a. b*b-4ac
b. x^2-y^2
c. a<b şi c
d. a≤b≤c
e. 1+x%y
f. !x%2=0

6. Scrieţi o expresie care are valoarea adevărat dacă şi numai dacă valoarea memorată în variabila reală x este în intervalul [a,b] (a şi b variabile reale).

7. Scrieţi o expresie care are valoarea adevărat dacă şi numai dacă valoarea memorată în variabila reală x este în intervalul [a,b]∪[c,d].

8. Scrieţi o expresie care are valoarea adevărat dacă şi numai dacă numărul natural memorat în variabila x este par şi are două cifre.

9. Scrieţi o expresie logică care să aibă valoarea adevărat dacă şi numai dacă valorile variabilelor a, b şi c sunt în ordine strict crescătoare.

10. Scrieţi o expresie a cărei valoare să fie adevărat dacă şi numai dacă numărul natural memorat în variabila an reprezintă un an bisect. Un an este bisect dacă este divizibil cu 4, dar nu este divizibil cu 100 sau este divizibil cu 400. De exemplu 1600 şi 1968 sunt ani bisecţi, dar 1700 şi 1970 nu sunt ani bisecţi.

11. Scrieţi o expresie a cărei valoare să fie adevărat dacă şi numai dacă numerele întregi memorate în variabilele x şi y sunt numere consecutive.

12. Scrieţi o expresie a cărei valoare să fie adevărat dacă şi numai dacă numerele naturale memorate în variabilele x şi y sunt ambele nenule.

13. Care dintre următoarele expresii are valoarea adevărat dacă şi numai dacă variabila întreagă x are ca valoare un număr impar strict pozitiv?

a. x%2≠0 şi x>0
b. !(x%2=0 sau x≤0)
c. !x%2=0 şi !x≤0
d. x>0 sau x%2=1

3. REPREZENTAREA ALGORITMILOR

3.1. Principiile programării structurate

Creşterea complexităţii aplicaţiilor a impus la începutul anilor '70 apariţia unei noi paradigme în programare: *programarea structurată*. Scopul era de a dezvolta noi tehnici de programare, care să permită dezvoltarea unor programe fiabile, uşor de elaborat în echipă, uşor de depanat, de întreţinut şi de reutilizat.

Un prim principiu al programării structurate este *modularizarea*. Pentru proiectarea unor aplicaţii complexe, este necesară descompunerea problemei care trebuie rezolvată în subprobleme relativ independente, pentru fiecare dintre aceste subprobleme scriindu-se module de program mai simple. Fiecare modul[1] efectuează un set de prelucrări specifice şi este relativ independent de celelalte module, cu care comunică prin intermediul unui set de parametri, care constituie interfaţa. Avantajele sunt multiple. Cum la orice firmă se lucrează în echipă, modulele de program pot fi implementate de mai mulţi programatori. Modificarea unui modul nu afectează celelalte module. Fiecare modul poate fi implementat, testat, depanat, modificat, independent de celelalte.

Un alt principiu fundamental este *structurarea datelor şi a prelucrărilor.*

Programatorul are posibilitatea de a-şi grupa datele în colecţii, organizate după anumite reguli, denumite structuri de date[2].

Prelucrările asupra datelor sunt structurate separat. Conform teoremei de structură Böhm-Jacopini, orice prelucrare poate fi descrisă prin compunerea a trei structuri fundamentale: *structura liniară* (secvenţială), *structura alternativă* şi *structura repetitivă.*

3.2. Reprezentarea algoritmilor în pseudocod

Pentru ca o secvenţă de operaţii să constituie un algoritm, ea trebuie să fie clară, adică la orice moment operaţia care urmează a fi executată trebuie să fie unic determinată, definită şi realizabilă (să poată fi efectuată la momentul respectiv, cu mijloacele disponibile). Apare întrebarea: care sunt operaţiile definite, cu ajutorul cărora să putem descrie algoritmi?

---

1. Proiectarea modulară a aplicaţiilor va fi studiată în detaliu în clasa a X-a.
2. În capitolul 10 vom studia câteva structuri de date fundamentale. În clasele a X-a şi a XI-a vom studia şi alte structuri de date (de exemplu, stiva, coada, lista).

Un răspuns posibil ar fi: operațiile definite sunt instrucțiunile limbajului de programare X! Este un răspuns acceptabil pentru toți cei care cunosc acest limbaj de programare. Dar pentru ceilalți? Nu putem impune nimănui să învețe un anumit limbaj de programare, numai pentru a înțelege algoritmii pe care îi scriem noi. În plus, experiența celor 5 decenii care s-au scurs de la apariția limbajelor de programare, ne învață că nici un limbaj nu este veşnic, nici unul nu a avut supremația, nici în timp, nici ca număr de utilizatori.

Prin urmare, este nevoie de o metodă universală de reprezentare a algoritmilor, ulterior fiecare programator având posibilitatea de a implementa algoritmii în limbajul pe care îl preferă. De-a lungul timpului s-au impus două modalități de reprezentare a algoritmilor: *schemele logice* şi *limbajele de tip pseudocod.*

Schemele logice constituie o metodă de reprezentare grafică, foarte sugestivă, dar cu o serie de dezavantaje: se dă o egală importanță componentelor principale ca şi detaliului, prin urmare schemele logice devin deosebit de stufoase şi greu de urmărit; pentru aplicațiile mai complexe, când este necesară modularizarea, este practic imposibil de pus în evidență legăturile dintre module în cadrul schemei logice.

Din acest motiv, treptat s-a impus o altă metodă de reprezentare a algoritmilor: pseudocodul. Un limbaj de tip pseudocod este un ansamblu de convenții, respectate în mod sistematic, care definesc operațiile permise (denumite şi instrucțiuni) pentru reprezentarea algoritmilor.

Vom prezenta în continuare un limbaj pseudocod, cu ajutorul căruia vom reprezenta algoritmii.

3.3. Structura secvențială

Declararea datelor

```
variabila tip;
```

La începutul oricărui algoritm, vom preciza datele de intrare, datele de ieşire, eventualele date de manevră, precum şi tipul acestora. Înainte de a utiliza orice variabilă, o vom declara, precizând numele şi tipul ei. O variabilă nu poate fi declarată de mai multe ori în acelaşi algoritm.

Exemple

```
x real;
c caracter;
i întreg;
```

Operația de citire

```
Citeşte variabila1, variabila2, ..., variabilan;
```

Efect: Prin operația de citire (denumită și operație de intrare) se preiau succesiv valori de la tastatură și se asociază, în ordine, variabilelor specificate.

Operația de scriere

```
Scrie expresie1, expresie2, ..., expresien;
```

Efect: Operația de scriere (denumită și operație de ieșire) presupune evaluarea în ordine a expresiilor specificate și afișarea pe ecran a valorilor lor pe aceeași linie.

Operația de atribuire

```
variabila ← expresie;
```

Efect: se evaluează expresia, apoi se atribuie valoarea expresiei variabilei din membrul stâng.

Instrucțiune compusă

```
{
  instrucțiune_1
  instrucțiune_2
    ...
  instrucțiune_n
}
```

Efect: se efectuează în ordine instrucțiunile specificate.

Instrucțiunea compusă este utilă atunci când sintaxa permite executarea unei singure instrucțiuni, dar este necesară efectuarea mai multor operații.

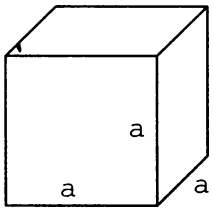
Observații

1. Orice instrucțiune se termină cu caracterul `';'`.
2. Pentru claritate, putem insera într-un algoritm comentarii, mici texte explicative. Începutul unui comentariu este marcat de succesiunea de caractere `/*`, iar sfârșitul comentariului este marcat de `*/`.

Parcurgerea instrucțiunilor în secvență, în ordinea specificării lor, reprezintă o ***structură liniară*** (secvențială).

3.4. Aplicații

Cub



Fie a un număr real, citit de la tastatură, care reprezintă lungimea laturii unui cub. Să se scrie un algoritm care să calculeze și să afișeze volumul și suprafața totală a cubului.

Soluție

```
Date de intrare:  a real;
Date de ieşire:   V real;     /*volumul cubului */
                  S real;     /*suprafaţa totală  */
Citeşte a;
V←a*a*a;
Scrie "Volumul cubului este ", V;
S←6*a*a;
Scrie "Suprafaţa totală a cubului este ", S;
```

Compus chimic

Un grup de cercetători studiază un compus chimic descoperit pe planeta Marte. În urma analizelor efectuate, au dedus că o moleculă din acest compus este formată din nC atomi de carbon, nO atomi de oxigen și nH atomi de hidrogen. Știind că masa atomului de carbon este 12, masa atomului de oxigen este 16, iar masa atomului de hidrogen este 1, să se scrie un algoritm care să calculeze și să afișeze masa moleculară a acestui compus.

Soluție

```
Date de intrare:  nC natural; /*numărul de atomi de carbon */
                  nO natural; /*numărul de atomi de oxigen */
                  nH natural; /*numărul de atomi de hidrogen */
Date de ieşire:   m natural;  /*masa moleculară a compusului */
Citeşte nC, nO, nH;
m←nc*12+nO*16+nH;
Scrie "Masa moleculară a compusului este ", m;
```

Înghețată

De ziua lui, Ionel a primit de la bunica S lei și ar vrea să invite la înghețată cât mai mulți colegi. Știind că o înghețată costă P lei, să se scrie un algoritm care să calculeze și să afișeze numărul maxim de colegi pe care Ionel îi poate invita și suma de bani care îi mai rămâne lui Ionel.

Soluție

```
Date de intrare:  S natural;     /* suma */
                  P natural;     /* prețul unei înghețate */
Date de ieșire:   nrc natural;   /* numarul maxim de invitați */
                  rest natural; /* suma rămasă */
Citește S, P;
nrc←S/(P+1);/* P+1 pentru că și Ionel mănâncă înghețată */
rest←S%(P+1);
Scrie "Numarul maxim de invitati este ", nrC;
Scrie "Suma rămasă este ", rest;
```

Triunghi

Fie x un număr natural format din 5 cifre ($x_4x_3x_2x_1x_0$). Să se afișeze un triunghi format din cifrele numărului x astfel:

- pe prima linie (în vârful triunghiului) se va afla cifra din mijloc (x_2)
- pe linia a doua se vor afla cifrele $x_3x_2x_1$
- pe a treia linie se vor afla toate cifrele lui x.

De exemplu, dacă x=15289, triunghiul va arăta astfel:

```
  2
 528
15289
```

Soluție

Problema constă în „spargerea" numărului x în cifre. În acest scop am numerotat cifrele numărului x de la dreapta la stânga începând cu 0, astfel încât numărul cifrei să corespundă puterii corespunzătoare a bazei (în cazul nostru baza 10): x_0 este cifra unităților (deci corespunde lui 10^0), x_1 este cifra zecilor (deci corespunde puterii 10^1), x_2 este cifra sutelor (corespunde lui 10^2) ș.a.m.d.

Devine astfel evident că pentru a extrage cifrele numărului x trebuie să efectuăm împărțiri la 10. Pentru a obține ultima cifră din x vom împărți pe x la 10 și vom reține restul în x_0. Eliminăm apoi ultima cifră din x (împărțind pe x la 10, x devine $x_4x_3x_2x_1$) acum x_1 a devenit cifra unităților și continuăm extragerea cifrelor numărului x în același mod.

```
Date de intrare:     x natural;
Date de manevră:     x0, x1, x2, x3 naturale;
Citește x;
x0←x%10; /* reținn cifra unităților */
x←x/10;  /*elimin cifra unităților */
x1←x%10; /* reținn cifra zecilor */
x←x/10;  /*elimin cifra zecilor */
x2←x%10; /* reținn cifra sutelor */
```

```
x←x/10;  /*elimin cifra sutelor */
x3←x%10; /* reţin cifra miilor */
x←x/10;  /*elimin cifra miilor, în x rămâne cifra zecilor
de mii */
Scrie "  ", x2; Scrie " ", x3,x2,x1; Scrie x,x3,x2,x1,x0;
```

Exerciţiu

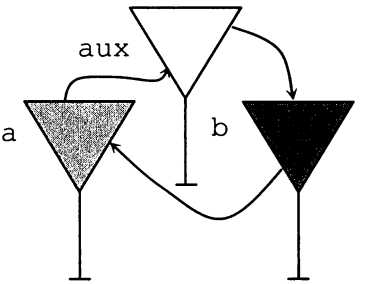
Modificaţi algoritmul precedent astfel încât să afişeze un triunghi format din cifrele unui număr de 6 cifre. De exemplu pentru `123456` algoritmul va afişa:

```
  34
 2345
123456
```

Schimb

Se introduc de la tastatură numerele reale `a` şi `b`. Să se interschimbe valorile variabilelor `a` şi `b`, apoi să se afişeze.

Soluţie



Să ne imaginăm că variabila `a` este un pahar cu vin alb, iar variabila `b` un pahar cu vin roşu. Trebuie să schimbăm conţinuturile celor două pahare. Singura soluţie fără „pierderi" este de a utiliza un pahar auxiliar (variabila `aux`). Vom turna vinul alb în paharul auxiliar (atribuim variabilei `aux` valoarea variabilei `a`). Acum paharul de vin alb este gol, turnăm în el vinul roşu (atribuim variabilei `a` valoarea variabilei `b`). Paharul de vin roşu (variabila `b`) a devenit disponibil, turnăm în el vinul alb din paharul auxiliar (atribuim variabilei `b` valoarea variabilei `aux`).

Această metodă este denumită sugestiv *„regula celor trei pahare"*, deşi analogia nu este absolut perfectă (când atribuim unei variabile valoarea altei variabile, această valoare se va găsi după atribuire în ambele, în timp ce atunci când turnăm conţinutul unui pahar de vin în alt pahar de vin ...).

```
Date de intrare/ieşire:  a real; b real;
Date de manevră: aux real;
Citeşte a, b;                  /* 1 */
aux←a;                         /* 2 */
a←b;                           /* 3 */
b←aux;                         /* 4 */
Scrie "a= ", a, "b= ", b;      /* 5 */
```

Pentru a înțelege mai bine acest algoritm să urmărim execuția lui pentru un set particular de date de intrare. În acest scop, am numerotat instrucțiunile de la 1 la 5. De exemplu, să presupunem că de la tastatură se introduc valorile 3 și 7.

Linia	a	b	aux	Explicație
1	3	7	?	Se citesc de la tastatură valorile 3 și 7 și se atribuie în ordine variabilelor a și b. Variabila aux are deocamdată o valoare necunoscută.
2	3	7	3	Se copiază valoarea variabilei a în variabila aux.
3	7	7	3	Variabilei a i se atribuie valoarea variabilei b.
4	7	3	3	Variabilei b i se atribuie valoarea variabilei aux (în care am „salvat" valoarea inițială a variabilei a).
5	7	3	3	Scriem valorile variabilelor a și b însoțite de mesaje explicative (pe ecran va apărea a= 7 b= 3).

3.5. Probleme propuse

1. Ce va afișa următorul algoritm, dacă se citesc valorile 7 și 23?

```
Date de intrare/ieșire:  a natural; b natural;
Citește a, b;
a←a+b;
b←a-b;
a←a-b;
Scrie "a= ", a, " b= ", b;
```

2. Ce va afișa următorul algoritm, dacă se citește valoarea 1234?

```
Date de intrare:   a natural;
Date de manevră:   b natural;
Date de ieșire:    c natural;
Citește a;
b←a%100;
a←a/100;
c←b*100+a;
Scrie c;
```

3. Ce valoare va avea variabila a la sfârșitul următoarei secvențe de instrucțiuni?

```
a, b întregi;
a ← 3; b ← 7;
b ← a+b/2; a ← a-b/2*a;
```

4. Fie a, b, c și d patru variabile reale. Care dintre următoarele instrucțiuni atribuie variabilei d media aritmetică a valorilor variabilelor a, b și c?

a. `d ← (a+b+c)/2;`
b. `d ← a/3+b/3+c/3;`
c. `d ← a+b+c/3;`
d. `d ←(a+b+c)/4-1;`

5. Fie `x1`, `x2`, `x3`, `x4`, `x5` cinci valori reale. Scrieți un algoritm care să folosească o singură variabilă suplimentară pentru a permuta circular valorile celor cinci variabile (adică în final `x1` să aibă valoarea inițială a variabilei `x2`, `x2` valoarea inițială a variabilei `x3`, `x3` valoarea inițială a variabilei `x4`, `x4` valoarea inițială a variabilei `x5`, iar `x5` valoarea inițială a variabilei `x1`).

6. Fie `a`, `b` și `c` trei numere reale, care reprezintă lungimile laturilor unui triunghi. Să se scrie un algoritm care să calculeze și să afișeze perimetrul și aria triunghiului.

7. O broască țestoasă parcurge o distanță de `D` kilometri în `H` ore. Să se scrie un algoritm care să calculeze și să afișeze viteza cu care se deplasează broasca țestoasă (exprimată în metri/secundă).

8. Doi colegi (Vasilică și Ionică) pleacă simultan din orașele în care locuiesc, unul către celălalt. Știind că distanța dintre cele două orașe este `D`, că Vasilică merge cu viteza `v1`, iar Ionică merge cu viteza `v2` (`D`, `v1`, `v2` numere reale), scrieți un algoritm care calculează după cât timp se întâlnesc cei doi colegi și la ce distanță de orașul în care locuiește Vasilică.

9. Fie A și B două puncte în plan, specificate prin coordonatele lor carteziene. Să se scrie un algoritm care să calculeze și să afișeze lungimea segmentului AB.

10. A fost odată un balaur cu 6 capete. Într-o zi Făt-Frumos s-a supărat și i-a tăiat un cap. Peste noapte i-au crescut alte 6 capete în loc. Pe același gât! A doua zi, Făt-Frumos i-a tăiat iar un cap, dar peste noapte balaurului i-au crescut în loc alte 6 capete ... și tot așa timp de `n` zile. În cea de-a `(n+1)`-a zi, Făt-Frumos s-a plictisit și a plecat acasă! Scrieți un algoritm care citește de la tastatură `n`, numărul de zile, și care afișează pe ecran câte capete avea balaurul după `n` zile.

    De exemplu, pentru `n=3`, algoritmul va afișa: `Dupa 3 zile balaurul are 15 capete.`

(Olimpiada Județeană pentru Gimnaziu, clasa a V-a, 2002)

3.6. Structura alternativă

```
Dacă expresie
     atunci
       instrucțiune_1
     altfel
       instrucțiune_2
```

Efect:

Se evaluează expresia.

Dacă valoarea expresiei este `adevărat`, atunci se execută `instrucțiune_1`.

Dacă valoarea expresiei este `fals`, se execută `instrucțiune_2`.

Observații

1. Atât pe ramura `atunci`, cât și pe ramura `altfel` este permisă executarea unei singure instrucțiuni. În cazul în care este necesară efectuarea mai multor operații, acestea se grupează într-o singură instrucțiune compusă.
2. Dacă pe ramura `altfel` nu este necesară efectuarea nici unei operații, această ramură poate lipsi.
3. Instrucțiunea `Dacă` permite executarea unei singure instrucțiuni, în funcție de valoarea unei expresii, deci permite selectarea condiționată a unei alternative. Această instrucțiune implementează în pseudocodul nostru structura alternativă.
4. `Instrucțiune_1` și `Instrucțiune_2` sunt subordonate instrucțiunii de decizie. În cazul în care o instrucțiune este subordonată unei alte instrucțiuni, aceasta se scrie de obicei indentat față de instrucțiunea care o subordonează. Modul de spațiere sau de scriere pe linii nu este predefinit, dar este important ca modul de redactare să facă programele ușor de urmărit și de întreținut.

Selectarea instrucțiunii ce urmează să fie executată în funcție de valoarea unei expresii reprezintă o ***structură alternativă***.

3.7. Aplicații

Modul

Se introduce de la tastatură un număr întreg x. Scrieți un algoritm care calculează și afișează modulul numărului x.

Soluție

Funcția modul este definită astfel: $|x| = \begin{cases} x, & dacă\ x \geq 0 \\ -x, & dacă\ x < 0 \end{cases}$

```
Date de intrare:  x întreg;
Date de ieşire:   m întreg;     /*modulul */
Citeşte x;
Dacă x<0 atunci m←-x;
         altfel m←x;
Scrie "Modulul este ", m;
```

Observație

Observați că nu am testat decât condiția `x<0`. Evident, este suficient. Dacă am pus întrebarea *„Moneda a căzut cu stema în sus?"* și ni s-a răspuns *„Nu!"*, este inutil să mai întrebăm dacă a căzut cu cealaltă față în sus.

Paritate

Se introduce de la tastatură un număr întreg x. Scrieți un algoritm care testează dacă x este un număr par.

Soluție

Numărul x este par dacă este divizibil cu 2 (adică restul împărțirii lui x la 2 este 0).

```
Date de intrare:  x întreg;
Date de ieşire:   Rezultatul testului
Citeşte x;
Dacă x%2=0 atunci Scrie x, ' este par";
           altfel Scrie x, ' nu este par";
```

Ecuația de gradul al II-lea

Fie o ecuație de gradul al II-lea cu coeficienți reali $ax^2+bx+c=0$ ($a\neq 0$). Scrieți un algoritm care să rezolve ecuația în mulțimea numerelor reale.

Soluție

Se calculează mai întâi discriminantul ecuației. Dacă acesta este negativ, ecuația nu are rădăcini reale. Dacă acesta este egal cu 0, ecuația are două rădăcini reale egale (confundate). Dacă discriminantul este mai mare decât 0, ecuația are două rădăcini reale distincte.

```
Date de intrare:     a, b, c reale; /* coeficienții ecuației */
Date de manevră:     d real;        /* discriminantul */
Date de ieşire:      Soluția
Citeşte a, b, c;
d←b*b-4*a*c;                        /* calculam discriminantul */
Dacă d<0 atunci
     Scrie "Ecuația nu are rădăcini reale";
     altfel
     Dacă d=0 atunci
       Scrie "x1=x2=", -b/(2*a);
       altfel                                    /*d este >0 */
       Scrie "x1=", (-b+√d)/(2*a), "x2=", (-b-√d)/(2*a)
```

Exercițiu

Modificați algoritmul pentru cazul în care ecuația are gradul cel mult II (caz în care a poate fi 0).

Funcție

Fie x un număr real. Să se calculeze valoarea funcției:
$F(x)=\max\{|2x-1|, 9-x^2\}$

Soluție

```
Date de intrare:  x real;
Date de ieşire:   f real;                    /* F(x) */
Citeşte x;
f ← 9-x*x;
Dacă 2*x-1>=0  atunci                 /* |2x-1|=2x-1 */
     {Dacă 2*x-1 > f atunci f ← 2*x-1;}
     altfel                          /* |2x-1|=1-2x */
     {Dacă 1-2*x > f atunci f ← 1- 2*x;}
Scrie f;
```

Sumă

Orice sumă de bani S (S>7) poate fi plătită numai cu monede de 3 lei şi de 5 lei. Dat fiind S>7, scrieți un algoritm care să determine o modalitate de plată a sumei S numai cu monede de 3 lei şi de 5 lei.

Soluție

Problema cere de fapt să găsim două numere naturale x şi y astfel încât S să poată fi scris sub forma S=3*x+5*y.

Vom analiza următoarele trei cazuri:

1. S%3 (restul împărțirii lui S la 3) este 0. În acest caz x=S/3, iar y=0.
2. S%3 este 1. Deoarece S>7, vom considera x=S/3-3, şi y=2, pentru că S=3*S/3+1=3*(S/3-3)+3*3+1=3*(S/3-3)+10=3*(S/3-3)+5*2.
3. S%3 este 2. În acest caz vom considera x=S/3-1 şi y=1, deoarece S=3*S/3+2=3*(S/3-1)+3+2=3*(S/3-1)+5.

```
Date de intrare:     S     natural;
Date de ieşire:      x, y naturale;
Date de manevră:     r     natural;
Citeşte S;
r ← S%3;
Dacă r=0  atunci {x ← S/3; y ← 0;}
     altfel
     Dacă r=1 atunci { x ← S/3-3; y ← 2;}
              altfel { x ← S/3-1; y ← 1;}
Scrie S,"=3*",x,"+5*",y;
```

3.8. Probleme propuse

1. Care este efectul următoarei secvențe de instrucțiuni?

```
a, b, c, x întregi;
a←3; b←5; c←7;
Dacă a-b/2<0 atunci x←1; altfel Dacă a+b-c/2<b atunci x←2;
altfel Dacă a%b+c>b atunci x←3; altfel x←4;
Scrie x;
```

2. Ce valoare inițială ar putea avea variabila x, astfel încât la sfârșitul execuției următoarei secvențe de instrucțiuni variabila y să aibă valoarea 2?

```
Dacă x>3 atunci Dacă x<7 atunci Dacă x%2=0 atunci y←1;
altfel y←2; altfel y←3;  altfel y←4.
```

3. Fie a și b două numere întregi. Scrieți un algoritm care să verifice dacă a și b sunt numere consecutive.

4. Fie ecuația cu coeficienți reali $ax^2+bx+c=0$ ($a\neq 0$). Scrieți un algoritm care, fără a calcula rădăcinile ecuației, să determine natura și semnul acestora.

5. Fie x un număr natural de trei cifre. Scrieți un algoritm care să elimine una dintre cifrele numărului astfel încât numărul de două cifre rămas să fie maxim.

6. Scrieți un algoritm care să citească 3 caractere și să determine câte caractere distincte s-au citit.

7. Fie a și b două unghiuri, ale căror măsuri sunt exprimate în grade, minute și secunde. Să se scrie un algoritm care să calculeze și să afișeze măsura sumei celor două unghiuri.

8. Se citesc de la tastatură a, b și c trei numere reale pozitive. Scrieți un algoritm care să verifice dacă numerele citite pot constitui laturile unui triunghi dreptunghic. În caz afirmativ, calculați și afișați aria triunghiului.

9. Fie x și y două numere reale, citite de la tastatură. Scrieți un algoritm care calculează și afișează valoarea funcției:

$$f(x,y)=\begin{cases}\dfrac{x+y}{2xy}, & \text{dacă } x,\ y>0\\ \max(x,y), & \text{dacă } x=0 \text{ sau } y=0\\ \left(\dfrac{1}{x}+\dfrac{1}{y}\right)\left(\dfrac{1}{x}-\dfrac{1}{y}+2xy+x^2+y^2\right), & \text{altfel}\end{cases}$$

10. Scrieți un algoritm care citește de la trei numere întregi strict pozitive a, b și c, numere cu cel mult trei cifre fiecare. Valoarea variabilei a reprezintă

distanța în km dintre orașul A și orașul B, `b` distanța în km dintre orașul B și orașul C, iar `c` reprezintă distanța în km dintre orașul C și orașul A. Știind că un călător își planifică o vizită a celor trei orașe pornind din oricare dintre orașele A sau B și ajungând în final în oricare dintre orașele B sau C cu trecere prin cel de-al treilea oraș, să se determine un traseu de lungime minimă care respectă aceste condiții. Algoritmul va afișa cele trei litere corespunzătoare celor trei orașe, în ordinea în care sunt vizitate. Se va alege o metodă cât mai eficientă din punctul de vedere al gestionării memoriei. De exemplu, pentru `a=58, b=140, c=125`, se va afișa `BAC`. (Bacalaureat 2002, sesiune specială)

11. Să se scrie un algoritm care să rezolve sistemul de două ecuații de gradul I, cu două necunoscute și coeficienți reali:

$$a_1x+b_1y=c_1$$
$$a_2x+b_2y=c_2$$

12. Un elev este declarat promovat la bacalaureat dacă la fiecare dintre cele 5 probe de examen a luat cel puțin nota 5, iar media sa generală este cel puțin 6. Date fiind cele 5 note pe care elevul le-a obținut la bacalaureat, scrieți un algoritm care să verifice dacă elevul a promovat sau nu examenul de bacalaureat.

13. Ionel are `H1` cm, Gigel are `H2` cm, iar Dănuț are `H3` cm. Scrieți un algoritm care să afișeze numele celor 3 copii în ordinea crescătoare a înălțimii.

3.9. Structura repetitivă

Instrucțiunea repetitivă condiționată anterior

```
Cât-timp expresie execută
    instrucțiune
```

Efect:

Pas 1: se evaluează expresia;
Pas 2: dacă valoarea expresiei este `fals`, se iese din instrucțiunea `cât-timp`;
dacă valoarea expresiei este `adevărat`, se execută instrucțiunea, apoi se revine la Pas 1.

Observații

1. Instrucțiunea se execută repetat, cât timp valoarea expresiei este `adevărat`. Pentru ca ciclul să nu fie infinit, este obligatoriu ca instrucțiunea care se execută să modifice cel puțin una dintre variabilele care intervin în expresie, astfel încât aceasta să poată lua valoarea `fals`.
2. Dacă expresia are de la început valoarea `fals`, instrucțiunea nu se execută nici măcar o dată.

Instrucțiunea repetitivă condiționată posterior

```
Execută
        instrucțiune
cât-timp expresie;
```

Efect:

Pas 1: se execută instrucțiunea;
Pas 2: se evaluează expresia;
Pas 3: dacă valoarea expresiei este `fals` se iese din instrucțiunea repetitivă;
dacă valoarea expresiei este `adevărat`, se revine la Pas 1.

Observații

1. Instrucțiunea se execută repetat, cât timp valoarea expresiei este `adevărat`. Pentru ca ciclul să nu fie infinit, este obligatoriu ca instrucțiunea care se execută să modifice cel puțin una dintre variabilele care intervin în expresie, astfel încât aceasta să poată lua valoarea `fals`. Deoarece evaluarea expresiei se face după execuția instrucțiunii, instrucțiunea se execută cel puțin o dată.
2. Instrucțiunea repetitivă condiționată posterior (`Execută cât-timp`) poate fi simulată cu ajutorul instrucțiunii repetitive condiționate anterior (`Cât-timp`) și reciproc.

Simularea instrucțiunii `Cât-timp` cu ajutorul instrucțiunii `Execută cât-timp`:

```
Dacă expresie atunci
    Execută
        instrucțiune
    cât-timp expresie;
```

Observați că a fost necesară testarea inițială a expresiei deoarece, spre deosebire de instrucțiunea `Cât-timp`, instrucțiunea `Execută cât-timp` efectuează o dată `instrucțiune` înainte de a testa expresia.

Simularea instrucțiunii `Execută cât-timp` cu ajutorul instrucțiunii `Cât-timp`:

```
instrucțiune;
Cât-timp expresie execută
    instrucțiune
```

Observați că în acest caz a fost necesar să executăm o dată instrucțiunea în afara ciclului.

Deducem că nu este necesară existența ambelor instrucțiuni, dar în funcție de problemă, vom alege instrucțiunea repetitivă adecvată, pentru care descrierea algoritmului este mai concisă.

Instrucțiunea repetitivă cu număr cunoscut de pași

```
Pentru contor←expresie_1, expresie_2 execută
       instrucțiune
```

Efect:

Pas 1: Se evaluează `expresie_1`.
Pas 2: Se atribuie variabilei `contor` valoarea expresiei `expresie_1`.
Pas 3. Se evaluează `expresie_2`.
Pas 4: Dacă valoarea variabilei `contor` este mai mare decât valoarea expresiei `expresie_2`, atunci se iese din instrucțiunea repetitivă. Dacă valoarea variabilei `contor` este mai mică sau egală cu valoarea expresiei `expresie_2`, atunci se execută `instrucțiune` și apoi se incrementează (se mărește cu 1) valoarea variabilei `contor`, după care se revine la Pas 3.

Observații

1. Dacă inițial valoarea expresiei `expresie_1` este mai mare decât valoarea `expresiei_2`, `instrucțiune` nu se execută nici o dată. În caz contrar, și dacă `instrucțiune` nu modifică valorile variabilelor care intervin în `expresie_2`, `instrucțiune` se execută de (`expresie_2-expresie_1+1`) ori.
2. Instrucțiunea repetitivă cu număr cunoscut de pași poate fi simulată cu ajutorul celorlalte două instrucțiuni repetitive.

Cu ajutorul instrucțiunii repetitive condiționată anterior (`Cât-timp`), instrucțiunea `Pentru` poate fi simulată astfel:

```
contor ← expresie_1
Cât_timp contor ≤ expresie_2 execută
          {
           instrucțiune;
           contor ← contor+1;
          }
```

Cu ajutorul instrucțiunii repetitive condiționată posterior (`Execută cât-timp`), instrucțiunea `Pentru` poate fi simulată astfel:

```
contor ← expresie_1
Dacă contor ≤ expresie_2 atunci
     Execută
          {
           instrucțiune;
           contor ← contor+1;
          }
     cât-timp contor ≤ expresie_2;
```

Executarea repetată a unei instrucțiuni, controlată de valoarea unei expresii, reprezintă o ***structură repetitivă***.

3.10. Aplicații

Secvență

Se consideră următorul algoritm descris în pseudocod

```
Date de intrare:  n, a, b naturale;
Date de ieşire:   nr natural;
Date de manevră:  i natural;
Citeşte n;                                   /* 1 */
Citeşte a;                                   /* 2 */
i←1; nr←0;                                   /* 3 */
Cât_timp i<n execută                         /* 4 */
         { Citeşte b;                        /* 5 */
           i←i+1;                            /* 6 */
           Dacă b%a=0 atunci nr←nr+1;        /* 7 */
           a←b;          }                   /* 8 */
Scrie nr;                                    /* 9 */
```

a. Ce se va afişa dacă se citesc valorile 6, 4, 8, 7, 21, 3, 6?
b. Introduceți o succesiune de cel puțin 3 valori astfel încât algoritmul să afişeze valoarea 0.
c. Care este efectul acestui algoritm ?

Soluție

a. Se cere să verificăm efectul algoritmului pentru un set particular de date de intrare. În acest caz vom executa instrucțiunile algoritmului pas cu pas, urmărind valorile variabilelor care intervin în algoritm. Pentru simplitate, am numerotat liniile de la 1 la 9.

Linia	n	a	b	nr	i	Explicație
1	6	?	?	?	?	Se citeşte valoarea 6 şi se atribuie variabilei n. Celelalte variabile au valori necunoscute.
2	6	4	?	?	?	Se citeşte valoarea 4 şi se atribuie variabilei a.
3	6	4	?	0	1	Se inițializează variabilele nr şi i cu valoarea 0, respectiv 1.
4	6	4	?	0	1	Se testează dacă valoarea variabilei i (1) este mai mică decât valoarea variabilei n (6). Deoarece expresia i<n are valoarea adevărat se continuă execuția algoritmului cu instrucțiunea 5.
5	6	4	8	0	1	Se citeşte valoarea 8 şi se atribuie variabilei b.

6	6	4	8	0	2	Se măreşte valoarea variabilei i cu 1.
7	6	4	8	1	2	Testăm dacă restul împărţirii variabilei b la variabila a este 0 (adică dacă a divide b). Deoarece 4 divide 8, valoarea variabilei nr se măreşte cu 1.
8	6	8	8	1	2	Variabilei a i se atribuie valoarea variabilei b, apoi se revine la instrucţiunea de pe linia 4.
4	6	8	8	1	2	Se testează dacă valoarea variabilei i (2) este mai mică decât valoarea variabilei n (6). Deoarece expresia i<n are valoarea adevărat se continuă execuţia algoritmului cu instrucţiunea de pe linia 5.
5	6	8	7	1	2	Se citeşte valoarea 7 şi se atribuie variabilei b.
6	6	8	7	1	3	Se măreşte valoarea variabilei i cu 1.
7	6	8	7	1	3	Testăm dacă restul împărţirii variabilei b la variabila a este 0 (adică dacă a divide b). Deoarece 8 nu divide 7, valoarea variabilei nr nu se măreşte cu 1.
8	6	7	7	1	3	Variabilei a i se atribuie valoarea variabilei b, apoi se revine la instrucţiunea de pe linia 4.
4	6	7	7	1	3	Se testează dacă valoarea variabilei i (3) este mai mică decât valoarea variabilei n (6). Deoarece expresia i<n are valoarea adevărat se continuă execuţia algoritmului cu instrucţiunea de pe linia 5.
5	6	7	21	1	3	Se citeşte valoarea 21 şi se atribuie variabilei b.
6	6	7	21	1	4	Se măreşte valoarea variabilei i cu 1.
7	6	7	21	2	4	Testăm dacă restul împărţirii variabilei b la variabila a este 0 (adică dacă a divide b). Deoarece 7 divide 21, valoarea variabilei nr se măreşte cu 1.
8	6	21	21	2	4	Variabilei a i se atribuie valoarea variabilei b, apoi se revine la instrucţiunea de pe linia 4.
4	6	21	21	2	4	Se testează dacă valoarea variabilei i (4) este mai mică decât valoarea variabilei n (6).

						Deoarece expresia i<n are valoarea adevărat se continuă execuţia algoritmului cu instrucţiunea de pe linia 5.
5	6	21	3	2	4	Se citeşte valoarea 3 şi se atribuie variabilei b.
6	6	21	3	2	5	Se măreşte valoarea variabilei i cu 1.
7	6	21	3	2	5	Testăm dacă restul împărţirii variabilei b la variabila a este 0 (adică dacă a divide b). Deoarece 21 nu divide 3, valoarea variabilei nr nu se măreşte cu 1.
8	6	3	3	2	5	Variabilei a i se atribuie valoarea variabilei b, apoi se revine la instrucţiunea de pe linia 4.
4	6	3	3	2	5	Se testează dacă valoarea variabilei i (5) este mai mică decât valoarea variabilei n (6). Deoarece expresia i<n are valoarea adevărat se continuă execuţia algoritmului cu instrucţiunea de pe linia 5.
5	6	3	6	2	5	Se citeşte valoarea 6 şi se atribuie variabilei b.
6	6	3	6	2	6	Se măreşte valoarea variabilei i cu 1.
7	6	3	6	3	6	Testăm dacă restul împărţirii variabilei b la variabila a este 0 (adică dacă a divide b). Deoarece 3 divide 6, valoarea variabilei nr se măreşte cu 1.
8	6	6	6	3	6	Variabilei a i se atribuie valoarea variabilei b, apoi se revine la instrucţiunea de pe linia 4.
4	6	6	6	3	6	Se testează dacă valoarea variabilei i (6) este mai mică decât valoarea variabilei n (6). Deoarece expresia i<n are valoarea fals se iese din ciclu şi se continuă execuţia algoritmului cu instrucţiunea de pe linia 9.
9	6	6	6	3	6	Se afişează valoarea variabilei nr (3).

b. Vom alege o secvenţă de exact 3 valori. Observăm că prima valoare din secvenţă este egală cu numărul de valori care urmează deci va trebui să fie 2. Pentru ca valoarea variabilei nr să rămână 0 (aşa cum este iniţial) va trebui să alegem două valori astfel încât condiţia din instrucţiunea 7 să nu fie îndeplinită (adică prima valoare să nu dividă pe cea de-a doua valoare). Prin urmare o soluţie posibilă este 2 5 7.

c. Urmărind algoritmul pas cu pas, deducem că se citeşte mai întâi o valoare n, apoi se citesc succesiv n numere naturale. Permanent avem memorate în variabilele a şi b două numere consecutiv citite dintre cele n. La fiecare pas se testează dacă primul număr (cel memorat în variabila a) divide pe cel de al doilea (cel memorat în variabila b) şi dacă da, valoarea variabilei nr este incrementată (mărită cu 1). Prin urmare, algoritmul numără în variabila nr câte perechi de elemente consecutive din secvenţa citită au proprietatea că primul element din pereche îl divide pe cel de-al doilea element din pereche.

Calcul

Se consideră următoarea secvenţă de instrucţiuni în pseudocod:

```
Date de intrare:       n natural;
Date de intrare/ieşire: x natural;
Date de manevră:       i natural;
Citeşte n, x;                                  /* 1 */
Pentru i←1,n execută                           /* 2 */
        x ← x*x;                               /* 3 */
Scrie x;                                       /* 4 */
```

a. Ce se va afişa pe ecran pentru n=3 şi x=2?
b. Scrieţi o secvenţă echivalentă, care să utilizeze structura repetitivă cât timp.
c. Care este efectul acestui algoritm?

Soluţie

a. Pentru a determina efectul algoritmului pentru n=3 şi x=2 vom urmări pas cu pas execuţia algoritmului, ilustrând în tabelul următor valorile variabilelor care intervin în acest algoritm.

Linia	n	x	i	Explicaţie
1	3	2	?	Se citeşte valoarea 3 şi se atribuie variabilei n, apoi se citeşte valoarea 2 şi se atribuie variabilei x. Valoarea variabilei i este deocamdată nedefinită.
2	3	2	1	Se iniţializează variabila i cu 1. Deoarece valoarea variabilei i ≤ valoarea variabilei n se continuă cu instrucţiunea de pe linia 3.
3	3	4	1	Se modifică valoarea variabilei x, se revine la linia 2.
2	3	4	2	Se măreşte valoarea variabilei i cu 1. Deoarece valoarea variabilei i ≤ valoarea variabilei n se continuă cu instrucţiunea de pe linia 3.
3	3	16	2	Se modifică valoarea variabilei x, se revine la linia 2.

2	3	16	3	Se măreşte valoarea variabilei i cu 1. Deoarece valoarea variabilei i ≤ valoarea variabilei n se continuă cu instrucţiunea de pe linia 3.
3	3	256	3	Se modifică valoarea variabilei x, se revine la linia 2.
2	3	256	4	Se măreşte valoarea variabilei i cu 1. Deoarece valoarea variabilei i > valoarea variabilei n se iese din ciclu şi se continuă cu instrucţiunea de pe linia 4.
4	3	256	4	Se afişează valoarea **256**.

b. Trebuie să transformăm structura repetitivă `Pentru` într-o structură repetitivă `Cât-timp`:

```
Date de intrare:        n natural;
Date de intrare/ieşire: x natural;
Date de manevră:        i natural;
Citeşte n, x;
i←1;
Cât-timp i≤n execută
     {  x ← x*x;
        i ← i+1;     }
Scrie x;
```

Observaţi că înainte de a intra în ciclu am iniţializat valoarea variabilei `i` cu `1`, iar în ciclu am inclus o instrucţiune care să incrementeze la fiecare pas valoarea variabilei `i`.

c. Pentru a determina efectul general al acestui algoritm vom aplica un raţionament inductiv. Deoarece valoarea variabilei `x` se modifică la fiecare pas, pentru a fi uşor de urmărit evoluţia acestei variabile, vom nota cu `x0` valoarea iniţială a variabilei `x`.

n	x	Explicaţie
0	$x0$	Dacă pentru `n` se citeşte valoarea `0` instrucţiunea `Pentru` nu va fi executată nici o dată, deci `x` rămâne cu valoarea sa citită iniţial (`x0`).
1	$x0^2$	Dacă pentru `n` se citeşte valoarea `1` instrucţiunea `Pentru` va fi executată o singură dată, deci `x` va avea valoarea sa $x0^2$.
2	$x0^4$	Dacă pentru `n` se citeşte valoarea `2` instrucţiunea `Pentru` va fi executată de două ori; după prima execuţie `x` va avea valoarea sa $x0^2$, iar după cea de a doua $x0^4$.
3	$x0^8$	Dacă pentru `n` se citeşte valoarea `3` instrucţiunea `Pentru` va fi executată de trei ori; după prima execuţie `x` va avea valoarea sa $x0^2$, după cea de a doua $x0^4$, iar după cea de a treia $x0^8$.

Urmărind execuţia algoritmului pentru câteva valori ale lui `n`, putem formula propoziţia inductivă:

„După n paşi, variabila x va avea valoarea $x0^{2^n}$".

Să demonstrăm propoziția: „După n+1 paşi, variabila x va avea valoarea $x0^{2^{n+1}}$"

Observăm că la pasul n+1 variabilei x i se atribuie valoarea variabilei x de la pasul n înmulțită cu ea însăşi, adică $x0^{2^n} * x0^{2^n} = x0^{2^n + 2^n} = x0^{2^{n+1}}$.

Numărare

Se consideră următoarea secvență de instrucțiuni în pseudocod:

```
Date de intrare:     a, b naturale;
Date de ieşire:      nr natural;
Date de manevră:     i natural;
Citeşte a, b;                                    /* 1 */
nr←0;                                            /* 2 */
Pentru i←a, b execută                            /* 3 */
      Dacă i%2=0 atunci nr ← nr+1;               /* 4 */
Scrie nr;                                        /* 5 */
```

a. Ce se va afişa pe ecran pentru a=5 şi b=8?
b. Scrieți o secvență echivalentă, care să utilizeze structura repetitivă Execută cât timp.
c. Care este efectul acestui algoritm?
d. Scrieți un algoritm echivalent mai eficient.

Soluție

a. Vom urmări pas cu pas execuția acestui algoritm:

Linia	a	b	nr	i	Explicație
1	5	8	?	?	Se citesc valorile 5 şi 8 şi se atribuie în ordine variabilelor a şi b. Variabilele nr şi i au deocamdată o valoare nedefinită.
2	5	8	0	?	Se inițializează variabila nr cu 0.
3	5	8	0	5	Se inițializează variabila i cu valoarea variabilei a (5). Deoarece valoarea variabilei i este ≤ valoarea variabilei b se continuă cu instrucțiunea de pe linia 4.
4	5	8	0	5	Se testează dacă restul împărțirii variabilei i la 2 este 0 (adică dacă i este număr par). Deoarece i este impar, valoarea variabilei nr rămâne nemodificată. Se revine la linia 3.

3	5	8	0	6	Se măreşte valoarea variabilei i cu 1. Deoarece valoarea variabilei i este ≤ valoarea variabilei b se continuă cu instrucţiunea de pe linia 4.
4	5	8	1	6	Se testează dacă restul împărţirii variabilei i la 2 este 0 (adică dacă i este număr par). Deoarece i este par, valoarea variabilei nr se măreşte cu 1. Se revine la linia 3.
3	5	8	1	7	Se măreşte valoarea variabilei i cu 1. Deoarece valoarea variabilei i este ≤ valoarea variabilei b se continuă cu instrucţiunea de pe linia 4.
4	5	8	1	7	Se testează dacă restul împărţirii variabilei i la 2 este 0 (adică dacă i este număr par). Deoarece i este impar, valoarea variabilei nr rămâne nemodificată. Se revine la linia 3.
3	5	8	1	8	Se măreşte valoarea variabilei i cu 1. Deoarece valoarea variabilei i este ≤ valoarea variabilei b se continuă cu instrucţiunea de pe linia 4.
4	5	8	2	8	Se testează dacă restul împărţirii variabilei i la 2 este 0 (adică dacă i este număr par). Deoarece i este par, valoarea variabilei nr se măreşte cu 1. Se revine la linia 3.
3	5	8	2	9	Se măreşte valoarea variabilei i cu 1. Deoarece valoarea variabilei i este > valoarea variabilei b se iese din instrucţiunea repetitivă şi se continuă cu instrucţiunea de pe linia 5.
3	5	8	2	9	Se afişează valoarea variabilei nr (2).

b. Va trebui să simulăm instrucţiunea Pentru cu ajutorul instrucţiunii Execută cât-timp:

```
Date de intrare:     a, b naturale;
Date de ieşire:      nr natural;
Date de manevră:     i natural;
Citeşte a, b;
nr ← 0; i ← a;
Dacă i ≤ b atunci
     Execută
       { Dacă i%2=0 atunci nr ← nr+1;
         i ← i+1;                          }
     Cât-timp (i≤b);
Scrie nr;
```

Observaţi că a fost necesar să testăm separat, înainte de a intra în instrucţiunea repetitivă Execută cât-timp, dacă i≤b. Dacă nu am fi pus acest test şi dacă iniţial am fi citit o valoare pară pentru variabila a strict mai mare decât valoarea

pentru variabila b, atunci acest algoritm ar fi afişat (incorect) valoarea 1, în loc de 0. Acest lucru se datorează faptului că instrucţiunea compusă subordonată instrucţiunii Execută cât-timp s-ar fi executat o dată şi s-ar fi mărit valoarea variabilei nr cu 1.

c. Observăm că variabila i primeşte în ordine toate valorile naturale din intervalul [a, b] (spunem că „variabila i parcurge intervalul [a, b]"). Pentru fiecare număr natural din intervalul [a, b] se testează dacă el este par şi dacă da se măreşte valoarea variabilei nr cu 1. Prin urmare, în variabila nr numărăm valorile pare din intervalul [a, b]).

d. Doi algoritmi sunt echivalenţi dacă, pentru orice set de date de intrare, produc aceleaşi date de ieşire.

Un algoritm echivalent, dar mai eficient de a număra valorile pare dintr-un interval dat [a, b] nu necesită parcurgerea tuturor numerelor din interval. În primul rând observăm că în intervalul [a, b] sunt exact L=b-a+1 numere naturale. În al doilea rând observăm că aproximativ jumătate din valorile din intervalul [a, b] sunt pare şi restul sunt impare. Am spus aproximativ jumătate deoarece trebuie să fim atenţi la paritatea capetelor intervalului. Mai exact va trebui să analizăm 4 cazuri posibile:

a	b	nr	Explicaţie
par	impar	L/2	Se formează exact L/2 perechi de numere naturale consecutive (par, impar).
impar	par	L/2	Se formează exact L/2 perechi de numere naturale consecutive (impar, par).
par	par	1+L/2	Se formează exact L/2 perechi de numere naturale consecutive (par, impar) la care se mai adaugă b care este de asemenea par.
impar	impar	L/2	Se formează exact L/2 perechi de numere naturale consecutive (impar, par), b fiind impar nu va fi numărat.

Algoritmul eficient de a număra valorile pare din intervalul [a, b] este:

```
Date de intrare:     a, b naturale;
Date de ieşire:      nr natural;
Date de manevră:     i natural;
Citeşte a, b;
nr ← 0;
Dacă a ≤ b atunci
     {nr ← b-a+1;
      Dacă a%2=0 şi b%2=0 atunci nr ← nr+1;}
Scrie nr;
```

Factorial

Fie n un număr natural. Să se calculeze n! (se citeşte „n factorial").

Soluţie

Prin definiţie, n!=1*2*...*n, pentru orice n>0, iar 0!=1. Observăm că n! se obţine înmulţind succesiv toate numerele naturale mai mici sau egale cu n. Pentru a calcula acest produs, vom utiliza o variabilă f. Iniţial variabila f va fi 1 (pentru că 1 este elementul neutru la înmulţire), apoi vom parcurge numerele naturale de la 2 la n, înmulţind succesiv variabila f cu fiecare dintre aceste numere.

```
Date de intrare:    n natural;
Date de ieşire:     f natural;
Date de manevră:    i natural;
Citeşte n;
f ← 1;
Pentru i ← 1, n execută f ← f * i;
Scrie f;
```

Exerciţii

1. Urmăriţi pas cu pas execuţia acestui algoritm pentru n=6.
2. Scrieţi un algoritm echivalent care să utilizeze instrucţiunea Cât-timp.

Putere

Fie n un număr natural şi x un număr real. Scrieţi un algoritm care calculează pe x^n.

Soluţie

$$x^n=\underbrace{x*x*...*x}_{\text{de n ori}}=(\underbrace{x*x*...*x}_{\text{de n-1 ori}})*x = x^{n-1}*x$$

Evident, $x^0=1$.

Vom utiliza o variabilă p în care vom calcula puterea. La fiecare pas i (i=1, ..., n) vom înmulţi valoarea variabilei p cu x.

```
Date de intrare: n natural; x real;
Date de ieşire: p real;
Date de manevră: i natural;
Citeşte n, x;
p ← 1;
Pentru i ← 1, n execută p ← p * x;
Scrie p;
```

Observație

Acest algoritm efectuează exact n înmulțiri.

Dobândă

Să presupunem că dorim să depunem la bancă o sumă S, cu o dobândă anuală de d%. Presupunând că timp de n ani nu intenționăm să scoatem bani din bancă și știind că dobânda se calculează și se cumulează în cont anual, elaborați un algoritm care să determine ce sumă vom avea în cont după n ani.

De exemplu, să presupunem că am depus 100 de lei, cu o dobândă de 5%. După 3 ani vom avea 115.7625 lei. Mai exact:

	Suma	Dobânda	Total
Inițial	100	0	100
După 1 an	100	5	105
După 2 ani	105	5.25	110.25
După 3 ani	110.25	5.5125	115.7625

Soluție

Vom simula „trecerea anilor", calculând după fiecare an dobânda și cumulând-o la suma existentă.

```
Date de intrare:      n natural; d real;
Date de intrare/ieşire: S real;
Date de manevră:      i natural;
Citeşte S, n, d;
Pentru i ← 1, n execută S ← S+S*d/100;
Scrie S;
```

Exercițiu

Urmăriți pas cu pas execuția acestui algoritm pentru S=1000, d=10%, n=5.

Maxim

I. Fie a și b două numere reale, citite de la tastatură. Scrieți un algoritm care determină și afișează pe ecran cea mai mare valoare citită.

Soluție

Compar valorile celor două variabile citite și o afișez pe cea mai mare:

```
Date de intrare:  a real; b real;
Date de ieşire:   maximul dintre a şi b
Citeşte a, b;
Dacă a < b atunci Scrie "max= ", b;
           altfel Scrie "max= ", a;
```

O altă soluție, care are avantajul că o vom putea generaliza pentru oricâte elemente, este de a reține maximul într-o variabilă (să o numim max). Inițializăm variabila max cu a (inițial pot considera că a este „cel mai bun"). Apoi compar maximul cu valoarea variabilei b și, dacă max este mai mic, îl schimb.

```
Date de intrare:  a real; b real;
Date de ieșire:   max real; /* maximul dintre a și b */
Citește a, b;
max←a;                    /* inițializare */
Dacă max < b atunci max←b;
Scrie "max= ", max;
```

II. Fie a, b și c trei numere reale, citite de la tastatură. Scrieți un algoritm care determină și afișează cea mai mare valoare citită.

Soluție

Ideea, utilizată în cea de a doua variantă a algoritmului precedent, o putem utiliza și pentru trei variabile:

```
Date de intrare:  a real; b real; c real;
Date de ieșire:   max real; /* maximul dintre a, b și c */
Citește a, b, c;
max←a;                          /* inițializare */
Dacă max < b atunci max←b;      /* compar cu b */
Dacă max < c atunci max←c;      /* compar cu c */
Scrie "max= ", max;
```

III. Se citește de la tastatură un număr natural nenul n, apoi se citesc succesiv n valori reale. Scrieți un algoritm care determină și afișează cea mai mare valoare reală citită.

Soluție

O primă problemă este declararea datelor de intrare: ni se dau n valori reale, numai că de data aceasta n nu este constant, ci variabil (citit de la tastatură). Cum să declarăm un număr variabil de variabile? Nu putem și nici nu este necesar, pentru că nu este nevoie să citim fiecare număr într-o variabilă separată! Vom citi numerele succesiv în aceeași variabilă (să o numim a) și imediat ce le-am citit le vom compara cu elementul maxim.

Pentru a intui mai bine algoritmul de calcul al maximului, să ne imaginăm un șir de elevi, care intră în clasă unul câte unul. Am dori ca pe măsură ce intră să reținem care este cel mai înalt. Evident, primul care intră poate fi considerat cel mai înalt (de altfel, este singurul pe care l-am văzut până acum). Când intră pe ușă un nou elev, compar înălțimea lui cu înălțimea celui pe care eu îl consider cel mai înalt de până acum. Dacă noul elev este mai înalt, reținînălțimea lui ca etalon pentru viitoarele comparații (el este cel mai înalt de până acum). Prin urmare, noi reținem permanent maximul dintre cei intrați deja.

Această idee o vom utiliza în continuare: în variabila max vom reține la fiecare moment cea mai mare valoare citită până la momentul respectiv.

```
Date de intrare:  n natural; a real;
Date de ieşire:   max real; /* maximul dintre valorile citite */
Date de manevră:  i natural;
/*în i reţinem câte valori am citit până la un moment dat*/
Citeşte n;
Citeşte max;  /* iniţializăm max cu primul element citit */
i←1;          /* i indică numărul de valori citite */
Cât-timp i<n execută
    {         /* cât timp nu am citit toate cele n valori */
      Citeşte a;          /* citesc o nouă valoare */
      i←i+1;              /* număr valoarea citită */
      Dacă max < a atunci max←a;      /* compar cu max */
    }
Scrie "max= ", max;
```

Observații

1. În instrucțiunea Cât-timp a fost necesară efectuarea mai multor operații (citirea unei valori, numărarea valorii citite şi compararea acesteia cu valoarea maximă de până acum). Prin urmare, am grupat instrucțiunile corespunzătoare într-o singură instrucțiune compusă.
2. Pentru a calcula maximul a n numere a fost necesară executarea instrucțiunii compuse subordonate instrucțiunii Cât-timp de n-1 ori.

Exerciții

1. Rescrieți algoritmul precedent utilizând o instrucțiune repetitivă cu număr cunoscut de paşi.
2. Modificați algoritmul precedent astfel încât să calculeze cel mai mic dintre elementele citite.
3. Modificați algoritmul precedent astfel încât să calculeze cele mai mari două elemente dintre cele citite.

Medie aritmetică

Se citeşte de la tastatură un număr natural nenul n, apoi se citesc succesiv n valori reale. Scrieți un algoritm care determină şi afişează media aritmetică a valorilor strict pozitive.

Soluție

Media aritmetică se calculează împărțind suma valorilor strict pozitive la numărul lor.

În mod asemănător cu problema precedentă, vom citi succesiv cele n valori întregi în aceeaşi variabilă a. După fiecare citire vom verifica dacă valoarea citită este strict pozitivă şi, în caz afirmativ, o vom număra şi o vom adăuga la sumă.

```
Date de intrare: n natural; a real;
Date de ieşire:  medie real; /* media aritmetică a numerelor */
Date de manevră: i natural; nr natural; s real;
/*în i reţinem câte valori am citit până la un moment dat;
  în nr calculăm câte valori pozitive am citit; în s
  calculăm suma valorilor strict pozitive citite */
Citeşte n;
i←0;   /* iniţial nu am citit nici o valoare */
nr←0; /* iniţial nu am citit nici o valoare pozitivă*/
s←0; /*iniţial suma valorilor strict pozitive citite e 0 */
Cât-timp i<n execută
    {         /* cât timp nu am citit toate cele n valori */
      Citeşte a;                  /* citesc o nouă valoare */
      i←i+1;                      /* număr valoarea citită */
      Dacă a > 0 atunci /* valoarea este strict pozitivă */
           {
             s←s+a;               /* o adun la sumă */
             nr←nr+1;             /* o număr */
           }
    }
Dacă nr≠0 atunci Scrie "Media aritmetică = ", s/nr;
     altfel Scrie "Media aritmetică nu se poate calcula";
```

Sumă cifre

Se citeşte de la tastatură un număr natural n. Să se calculeze suma cifrelor lui n.

Soluţie

Cel mai uşor se obţine ultima cifră din număr (cifra unităţilor): n%10 (restul împărţirii lui n la 10). După ce utilizăm ultima cifră (o adăugăm la sumă), nu mai avem nevoie de ea şi o eliminăm din număr (n←n/10). Cifra zecilor a devenit acum ultima cifră. Repetăm procedeul, cât timp numărul mai are cifre.

```
Date de intrare: n natural;
Date de ieşire:  s natural; /* suma cifrelor lui n */
Citeşte n;
s←0;                    /* iniţial suma cifrelor este 0   */
Cât-timp n≠0 execută /* cât timp mai există cifre în n */
    { s←s+n%10;      /* adun la s ultima cifră a lui n */
      n=n/10; }      /* elimin ultima cifră a lui n*/
Scrie "Suma cifrelor = ", s;
```

Observație

Inițializarea unei variabile în care trebuie să calculăm o sumă se face întotdeauna cu 0 (elementul neutru la adunare). În mod similar, inițializarea unei variabile în care trebuie să calculăm un produs se face întotdeauna cu 1 (elementul neutru la înmulțire).

Exerciții

1. Urmăriți pas cu pas execuția acestui algoritm pentru n=5672.
2. Modificați algoritmul astfel încât să determine numărul n răsturnat (scris de la dreapta la stânga). De exemplu, pentru n=3201 algoritmul va afișa 1023.
3. Modificați algoritmul astfel încât să determine numărul de cifre pare ale lui n.

Perechi

Fie n un număr natural nenul. Să se genereze toate perechile (a, b), cu proprietatea că a | b, unde a și b sunt numere naturale nenule mai mici decât n.

Soluție

În primul rând observăm că pentru ca a să dividă b trebuie ca a≤b. O primă idee ar fi ca a să parcurgă toate numerele naturale nenule mai mici decât n, apoi pentru fiecare valoare a, b să parcurgă toate numerele mai mari sau egale cu a și mai mici decât n, formând astfel toate perechile de numere (a, b) cu a≤b<n. Pentru fiecare pereche vom testa dacă a | b.

```
Date de intrare:  n natural;
Date de manevră:  a, b naturale;
Date de ieşire:   perechile (a, b) cu proprietatea cerută;
Citeşte n;
Pentru a ← 1, n-1 execută
        Pentru b ← a, n-1 execută
            Dacă b%a=0 atunci Scrie '(', a, ',', b, ')';
```

În acest algoritm instrucțiunea Dacă este executată de n*(n-1)/2 ori, deoarece:

- când a=1, a doua instrucțiune Pentru se execută de n-1 ori;
- când a=2, a doua instrucțiune Pentru se execută de n-2 ori;
- când a=3, a doua instrucțiune Pentru se execută de n-3 ori;
- ...
- când a=n-1, a doua instrucțiune Pentru se execută o dată.

În total: 1+2+3+...+n-1=n*(n-1)/2 executări ale instrucțiunii Dacă.

O idee mai bună ar fi ca, având a fixat, să nu mai parcurgem cu variabila b toate numerele naturale de la a la n-1, ci doar multiplii lui a:

```
Date de intrare:  n natural;
Date de manevră: a, b naturale;
Citeşte n;
Pentru a ← 1, n-1 execută
      { m ← 1;
        Cât-timp a*m≤n-1 execută
                {Scrie '(', a, ',', a*m, ')';
                 m ← m+1; }
      }
```

În acest mod algoritmul a devenit mai eficient, obținându-se direct perechile corecte.

Divizori

Dat fiind n un număr natural, să se determine toți divizorii naturali ai lui n.

Soluție

În primul rând observăm că orice număr natural n are ca divizori pe 1 şi pe n (aceştia se numesc divizori improprii). În afară de divizorii improprii, cel mai mic divizor ar putea fi 2, iar cel mai mare n/2. Prin urmare, pentru a afla divizorii proprii ai lui n, vom baleia toate numerele naturale din intervalul [2, n/2] şi vom testa pentru fiecare număr natural x din acest interval dacă divide sau nu pe n (adică dacă restul împărțirii lui n la x este 0).

```
Date de intrare:  n natural;
Date de manevră: x natural;
Date de ieşire:  Divizorii lui n
Citeşte n;
Scrie 1;
Pentru x←2, n/2 execută    /* parcurg intervalul [2,n/2] */
       Dacă n%x=0 atunci   /* x divide n? */
            Scrie x;       /* x este divizor, îl scriu */
Scrie n;
```

Primalitate

Fie n un număr natural, n>1, citit de la tastatură. Scrieți un program care verifică dacă numărul n este prim.

Soluție

Pentru a testa dacă numărul n este prim, vom verifica dacă are sau nu divizori proprii (divizori diferiți de 1 şi de n). Pentru aceasta vom parcurge intervalul

numerelor naturale de la 2 la $\sqrt{n}$ [3], verificând pentru fiecare număr dacă este sau nu divizor al lui n. Pentru a reține rezultatul testului vom utiliza o variabilă denumită prim cu semnificația „prim este 1 dacă n este prim și 0 dacă n nu este prim". Inițial variabilei prim îi atribuim valoarea 1 („prezumția de nevinovăție").

Pe parcursul verificării, dacă găsim un divizor, variabilei prim îi atribuim valoarea 0 („dovada" că n nu este prim a fost găsită).

Verificarea continuă cât timp avem *unde* căuta (nu am ajuns la $\sqrt{n}$) și avem *de ce* căuta (nu am găsit încă „dovada vinovăției" – un divizor al lui n).

```
Date de intrare:  n natural;
Date de manevră:  d, prim naturale;
Date de ieşire:   Rezultatul testului
Citeşte n;
prim ← 1; d ← 2;
Cât-timp d≤√n şi prim=1 execută
     Dacă n%d=0 atunci prim=0;    /* am gasit un divizor */
         altfel d ← d+1;          /* căutam mai departe */
Dacă prim=1 atunci Scrie n, " este prim";
            altfel Scrie n, " nu este prim";
```

Am putea îmbunătăți performanțele acestui algoritm observând că singurul număr prim par este 2. Vom testa în primul rând dacă n este par (caz în care, exceptându-l pe 2, nu este prim), apoi vom verifica numai numerele impare.

```
Date de intrare:  n natural;
Date de manevră:  d, prim naturale;
Date de ieşire:   Rezultatul testului;
Citeşte n;
prim ← 1;
Dacă n>2 atunci
   Dacă n%2 =0 atunci prim ← 0;
        altfel
        {d ← 3;
         Cât-timp d≤√n şi prim=1 execută
              Dacă n%d=0 atunci prim=0;
                         altfel d←d+2;
         }
Dacă prim=1 atunci Scrie n, " este prim";
            altfel Scrie n, " nu este prim";
```

3. La matematică ați demonstrat o teoremă conform căreia dacă un număr nu are nici un divizor până la radicalul său, atunci el este prim.

Exerciții

1. Urmăriți pas cu pas execuția acestui algoritm pentru n=13, n=8 și n=49.
2. Modificați algoritmul astfel încât să afișeze toate numerele naturale prime ≤ n.

Descompunere în factori primi

Fie n un număr natural (n>1), citit de la tastatură. Scrieți un program care să afișeze descompunerea numărului natural n în factori primi.

De exemplu, pentru n=720 programul va afișa 2^4*3^2*5^1, ceea ce înseamnă că divizorii primi sunt 2, 3 și 5, având respectiv multiplicitățile 4, 2 și 1.

Soluție

Vom aplica algoritmul învățat la matematică: considerăm toate numerele naturale începând cu 2. Pentru fiecare număr, verificăm dacă este divizor al lui n. În caz afirmativ, calculăm multiplicitatea acestui divizor în n, împărțind succesiv pe n la divizor și calculând numărul de împărțiri efectuate, după care afișăm atât divizorul găsit, cât și multiplicitatea sa. Procedeul se repetă cât timp n mai are divizori (n>1).

```
Date de intrare:  n natural;
Date de manevră: d, m naturale;
n, m, d naturale;
Citeşte n;
d ← 2;
Cât-timp n>1 execută
       Dacă n%d=0 atunci                    /* d divide n */
            {     /* calculez multiplicitatea lui d în n */
             m ← 0;
             Cât-timp n%d=0 execută
                  { m ← m+1; n ← n/d; }
             Scrie "divizorul ", d," multiplicitatea ", m;
            }
```

Exerciții

1. Urmăriți pas cu pas execuția acestui algoritm pentru n=2500.
2. Observați că nu am testat nicăieri dacă d este prim. Nu este necesar, deoarece, datorită faptului că atunci când găsim un divizor al lui n, îl împărțim pe n la divizor ori de câte ori este posibil, orice divizor nou găsit este prim. Demonstrați această afirmație prin reducere la absurd.
3. Modificați programul astfel încât să afișeze numai cel mai mare divizor prim.
4. Modificați programul astfel încât să calculeze numărul total de divizori ai lui n.

Cel mai mare divizor comun

Fie n şi m două numere naturale, citite de la tastatură. Scrieţi un program care să calculeze şi să afişeze c.m.m.d.c. (n, m).

Soluţie

Vom aplica algoritmul lui Euclid, învăţat la matematică. Cât timp m≠0, împărţim n la m şi calculăm restul; la pasul următor împărţitorul devine deîmpărţit, iar restul devine împărţitor. Ultimul rest ≠ 0 este c.m.m.d.c.

De exemplu, pentru a=495 şi b=720, *cel mai mare divizor comun* prin algoritmul lui Euclid se obţine astfel:

Deîmpărţit		Împărţitor			Rest	
495	=	720	·	0	+ 495	
720	=	495	·	1	+ 225	
495	=	225	·	2	+ 45	
225	=	45	·	5	+ 0	⇒ c.m.m.d.c.(495,720)=45

Date de intrare/ieşire: n, m naturale;
Date de manevră: rest natural;

```
Citeşte n, m;
Cât-timp m≠0 execută
        { rest←n%m;     /* calculez restul                    */
          n←m;          /* împărţitorul devine deîmpărţit     */
          m←rest; }     /* restul devine împărţitor           */
Scrie "cmmdc=" , n;     /* ultimul rest diferit de 0          */
```

Exerciţii

1. Urmăriţi pas cu pas execuţia acestui algoritm pentru n=121 şi m=4950.
2. Observaţi că nu am testat nicăieri dacă iniţial n>m. Nu este necesar, deoarece, dacă iniţial n ar fi mai mic decât m, după prima executare a instrucţiunii compuse subordonate instrucţiunii Cât-timp valorile variabilelor n şi m vor fi interschimbate.
3. Scrieţi un algoritm echivalent care să utilizeze instrucţiunea repetitivă Execută cât-timp.
4. Modificaţi acest algoritm astfel încât să simplifice fracţia n/m şi să afişeze fracţia ireductibilă obţinută.
5. Modificaţi acest algoritm astfel încât să verifice dacă numerele n şi m sunt prime între ele.
6. Modificaţi acest algoritm astfel încât să calculeze c.m.m.m.c. (n, m) (cel mai mic multiplu comun al numerelor n şi m).

7. Modificați acest algoritm astfel încât să determine Φ(n) (indicatorul lui Euler[4]). Prin definiție, Φ(n) este numărul de numere naturale mai mici decât n, relativ prime cu n. Două numere sunt relativ prime dacă cel mai mare divizor comun al lor este 1.

Termen Fibonacci

Fie șirul Fibonacci[5] 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

Observați că primii doi termeni sunt egali cu 1, fiecare termen următor fiind egal cu suma celor doi termeni care îl preced.

Fie n un număr natural citit de la tastatură. Scrieți un program care afișează cel de-al n-lea termen din șirul Fibonacci.

Soluție

Observăm că pentru a calcula un termen al șirului Fibonacci avem nevoie de doi termeni precedenți. Vom utiliza trei variabile f0, f1, și f2. În f0 și f1 vom reține doi termeni Fibonacci consecutivi, iar în f2 vom calcula termenul următor. Variabilele f0 și f1 vor fi inițializate cu 1 (primii doi termeni din șirul Fibonacci). Următorii termeni (de la al doilea până la al n-lea) îi vom calcula succesiv. La fiecare pas i, în f2 vom calcula al i-lea termen din șirul Fibonacci (f2←f1+f0), apoi ne pregătim pentru pasul următor (f0←f1, f1←f2):

```
Date de intrare:  n natural;
Date de manevră: i, f0, f1, f2 naturale;
f0 ← 1; f1 ← 1; f2 ← 1;
Citește n;
Pentru i←2, n execută
       { f2 ← f1+f0;          /* calculez termenul curent */
         f0 ← f1;             /* ma deplasez in sir       */
         f1 ← f2; }
Scrie "Cel de-al ", n, "-lea termen Fibonacci este: ", f2;
```

Exerciții

1. Urmăriți pas cu pas execuția acestui program pentru n=10.

---

4. Leonhard Euler (1707–1783) a fost unul dintre cei mai prolifici matematicieni din istorie. A scris 866 de lucrări și a câștigat premiul Academiei Franceze de 12 ori.
5. Acest șir a fost introdus în 1202 de către Leonardus filius Bonaccii Pisanus (Leonardo, fiul lui Bonacci Pisanul), pentru rezolvarea următoarei probleme: *Să se determine câte perechi de iepuri se vor naște într-un an, dintr-o singură pereche de iepuri, știind că fiecare pereche de iepuri dă naștere lunar la o nouă pereche de iepuri, o pereche devine fertilă în decurs de o lună, iar iepurii nu mor pe parcursul anului.* Șirul *Fibonacci* are multe aplicații, în variate domenii. De exemplu, numerele *Fibonacci* sunt utilizate în probleme de sortare și căutare sau în strategii de joc.

2. Modificați algoritmul astfel încât să afișeze cel de-al n-lea termen din șirul Lucas[6]. Primii termeni din șirul Lucas sunt 1, 3, 4, 7, 11, 18, 29, 47, ...
3. Modificați algoritmul astfel încât să calculeze suma primilor n termeni din șirul Fibonacci.
4. Modificați algoritmul astfel încât să calculeze cel mai mare termen din șirul Fibonacci, mai mic sau egal cu n.

Existență

Se citesc de la tastatură două numere naturale nenul n și p, apoi se citesc succesiv n valori întregi. Scrieți un algoritm care să verifice dacă printre cele n valori citite există multipli ai lui p.

Soluție

Vom citi succesiv cele n valori întregi în aceeași variabilă a. După fiecare citire vom verifica dacă valoarea citită este divizibilă cu p și, în caz afirmativ, vom reține faptul că există un multiplu al lui p. Rezultatul verificării îl vom reține în variabila exista. Inițial valoarea acestei variabile este 0 (deoarece încă nu am descoperit nici un multiplu al lui p). Dacă pe parcursul citirii vom descoperi un multiplu al lui p, variabila exista va primi valoarea 1.

```
Date de intrare:  n, p, a naturale;
Date de manevră:  i, exista naturale;
Date de ieșire:   Rezultatul testului;
Citește n, p;
exista ← 0;
Pentru i←1, n execută
       { Citește a;            /* citesc un nou termen      */
         Dacă a%p=0 atunci  /* am descoperit un multiplu */
         exista ← 1; }
Dacă exista=1 atunci Scrie "Exista un multiplu al lui ", p;
     altfel Scrie "Nu exista un multiplu al lui ", p;
```

Exerciții

1. Urmăriți pas cu pas execuția acestui algoritm pentru șirul de valori 6, 3, 7, 4, 1, 12, 4, 5.
2. Modificați algoritmul astfel încât să calculeze numărul de multipli ai lui p existenți în șir.

---

6. François-Edouard-Anatole Lucas (1842–1891), matematician francez cu importante contribuții în teoria numerelor.

Număr divizori primi

Se citeşte de la tastatură un număr natural nenul n, apoi se citesc succesiv n valori întregi. Scrieți un algoritm care să verifice dacă oricare număr dintre cele n citite are un număr impar de divizori primi.

Soluție

Vom citi succesiv cele n valori în aceeaşi variabilă x. Pentru fiecare valoare citită vom determina numărul de divizori primi (folosind un algoritm asemănător cu descompunerea în factori primi, numai că în loc să afişăm divizorii primi şi multiplicitățile acestora, vom număra divizorii).

Rezultatul verificării îl vom reține în variabila oricare. Inițial, această variabilă are valoarea 1 (deoarece încă nu am găsit un număr care să nu aibă un număr impar de divizori primi). Dacă, pe parcursul citirii, vom întâlni un număr cu număr par de divizori primi, variabilei oricare îi vom atribui valoarea 0.

```
Date de intrare:  n, x naturale;
Date de manevră:  i, d, nr, oricare naturale;
Date de ieşire:   Rezultatul testului;
Citeşte n;
oricare ← 1;
Pentru i←1, n execută
     {Citeşte x;              /* citesc o noua valoare     */
      nr ← 0;
     /*numărul de divizori primi ai lui x este iniţial 0 */
      d ← 2;          /*determin divizorii primi ai lui x */
      Cât-timp x>1 execută
          {Dacă x%d=0 atunci /* am descoperit un divizor */
             {nr ← nr+1;
              /* număr acest divizor prim, apoi
               îl elimin din x, prin impartiri repetate  */
              Cât-timp x%d=0 execută x=x/d;  }
           d ← d+1; }       /* caut divizori mai departe */
       Dacă nr % 2 = 0 atunci oricare ← 0;
       /* pentru că am gasit o valoare cu număr par
          de divizori primi */
      }
Dacă oricare=1 atunci Scrie "DA";
               altfel Scrie "NU";
```

Vocale

Se citeşte de la tastatură o propoziție scrisă cu litere mici, terminată cu '.'. Scrieți un algoritm care calculează şi afişează numărul de vocale din propoziție.

Soluție

Citim succesiv caracterele din care este constituită propoziția, până la sfârșit, în aceeași variabilă (să o numim c). Pentru fiecare caracter citit, verificăm dacă este vocală ('a', 'e', 'i', 'o' sau 'u') și dacă da, îl numărăm.

```
Date de intrare: c caracter;
/*în c vom citi succesiv caracterele din care este
constituită propoziția */
Date de ieșire: nr natural;                    /* numărul de vocale */
nr←0;          /*inițial numărul de vocale citite este 0 */
Execută
    {
      Citește c;                               /*citim un caracter */
      Dacă c='a' sau c='e' sau c='i' sau c='o' sau c='u'
           atunci nr←nr+1;     /* c este vocală, o număr */
    }
cât-timp c ≠ '.';
Scrie "Numărul de vocale este ", nr;
```

3.11. Probleme propuse

1. Se consideră următoarea secvență de instrucțiuni, scrisă în pseudocod:

```
Date de intrare/ieșire: n, m naturale;
Citește n, m;              /* n și m sunt diferite de 0 */
Cât-timp n≠m execută
    Dacă n>m atunci n←n-m;
             altfel m←m-n;
Scrie n;
```

   a. Ce rezultat se scrie pentru n=72 și m=180? Dar pentru n=30 și m=30?
   b. Dați exemple de valori ce ar putea fi introduse pentru n și m astfel încât algoritmul să afișeze valoarea 13.
   c. Care este efectul acestui algoritm?

2. Se consideră următorul algoritm descris în pseudocod:

```
Date de intrare: a, b naturale;
Date de ieșire: c natural;
Citește a, b;
c←0;
Cât_timp a≠0 și b≠0 execută
      {Dacă a%10>b%10  atunci c←c*10+a%10;
                       altfel  c←c*10+b%10;
        a←a/10;  b←b/10;                    }
```

   a. Ce valoare va avea variabila c după execuția acestui algoritm dacă valorile citite pentru variabilele a și b sunt 14624 și, respectiv, 8632?

b. Dacă se citeşte valoarea 7382 pentru variabila a, ce valoare am putea introduce pentru variabila b astfel încât valoarea calculată de algoritm în variabila c să fie 2938?

c. Dați exemplu de două numere naturale de cel puțin două cifre, care ar putea fi introduse pentru variabilele a şi b astfel încât valoarea calculată de algoritm în variabila c să fie 1?

3. Se consideră următorul algoritm descris în pseudocod:

```
Date de intrare: n natural;
Date de ieşire: d natural;
Citeşte n;
d←2;
Cât-timp n>1 execută
      Dacă n%d=0 atunci n←n/d;
                 altfel  d←d+1;
Scrie d;
```

a. Ce valoare va fi afişată dacă valoarea citită pentru variabila n este 720?

b. Dați exemplu de valori (cel puțin două) care ar putea fi introduse pentru variabila n, astfel încât valoarea afişată de algoritm să fie 11.

c. Care este efectul acestui algoritm ?

4. Se consideră următorul algoritm descris în pseudocod:

```
Date de intrare:   n, b naturale;
Date de ieşire:    x natural;
Date de manevră:   p natural;
Citeşte n, b;                    /* 2≤b≤9 */:
x←0; p←1;
Cât-timp n>0 execută
          { x←x + p*(n %10);
            p←p*b; n←n/10;              }
Scrie x;
```

a. Ce valoare va fi afişată dacă se citesc valorile 1032 şi 4?

b. Ce valoare trebuie să introducem pentru variabila b astfel încât pentru n=407 să se afişeze valoarea 331?

c. Care este efectul acestui algoritm ?

5. Se consideră următorul algoritm descris în pseudocod:

```
Date de intrare:   a, b naturale;
Date de ieşire:    x natural;
Date de manevră:   i natural;
Citeşte a, b;
i←a; x←0;
Cât-timp i<=b execută
          { x←x+i;
            i←i+1; }
Scrie x;
```

a. Ce valoare va fi afişată pe ecran dacă se introduc valorile 1 şi 10?

b. Ce valori ar putea fi introduse pentru a şi b astfel încât programul să afişeze valoarea 22?

c. Daţi exemplu de valori distincte care ar putea fi introduse astfel încât algoritmul să afişeze valoarea 0.

d. Scrieţi un algoritm echivalent mai eficient. Doi algoritmi sunt consideraţi echivalenţi dacă pentru orice date de intrare ei produc aceleaşi rezultate.

6. Care dintre următoarele secvenţe de instrucţiuni atribuie variabilei întregi u valoarea primei cifre a numărului natural reprezentat de variabila x ?

a.
```
u←x/10;
```

b.
```
u←x;
Cât-timp u≥10 execută u←u%10;
```

c.
```
u←x%10;
```

d.
```
Cât-timp x≥10 execută x←x/10;
u←x;
```

7. Care dintre următoarele secvenţe afişează cel mai mare divizor propriu al numărului natural neprim memorat în variabila x?

a.
```
d←2;
Cât-timp x%d=0 execută
          d←d+1;
Scrie x/d;
```

b.
```
d←2;
Execută d←d+1;
cât-timp x%d≠0;
Scrie x/d;
```

c.
```
d←2;
Cât-timp x%d≠0 execută
          d←d+1;
Scrie x/d;
```

d.
```
d←x/2;
Cât-timp x%d≠0 execută
          d←d-1;
Scrie d;
```

8. Ce valoare iniţială ar trebui să aibă variabila x astfel încât după execuţia următoarei secvenţe de instrucţiuni să se afişeze valoarea 640?

```
d←2;
Cât-timp d<50 execută
          { x←x*d; d←d*d;}
Scrie x;
```

9. Fie n un număr natural nenul şi p un număr prim. Algoritmul următor trebuie să determine multiplicitatea lui p în n. Ce greşeli conţine algoritmul?

```
n, p, m naturale;
Citeşte n, p;
Cât-timp n%p=0 execută m←m+1; n←n/p;
Scrie m;
```

10. *Număr perfect*

Un număr este perfect dacă el este egal cu suma divizorilor săi (exclusiv el însuşi). Scrieţi un algoritm care să testeze dacă un număr natural dat este perfect.

11. *Numere prietene*

Două numere naturale a şi b se numesc prietene dacă a este egal cu suma divizorilor lui b (exclusiv b), iar b este egal cu suma divizorilor lui a (exclusiv a). De exemplu a=220 şi b=284 sunt prietene. Scrieţi un program care să determine primele trei perechi de numere prietene, cu a<b.

12. *Prime*

Fie x un număr natural, x>2. Scrieţi un algoritm care să determine cel mai mare număr prim, mai mic decât x şi cel mai mic număr prim, mai mare decât x.

13. *Palindrom*

Fie n un număr natural, citit de la tastatură. Scrieţi un algoritm care testează dacă n este palindrom (citit de la stânga la dreapta şi de la dreapta la stânga este acelaşi). De exemplu 1331, 14541 sunt palindroame.

14. *Maxim şi minim*

Se citeşte de la tastatură un număr naturale nenul n, apoi se citesc succesiv n valori naturale. Scrieţi un algoritm care să determine şi cel mai mare şi cel mai mic element din cele citite.

15. *Sumă şi produs*

Se citeşte de la tastatură un număr natural nenul n, apoi se citesc succesiv n valori întregi. Scrieţi un algoritm care determină şi afişează suma valorilor pare citite şi produsul valorilor nenule.

16. *Palindrom prim*

Se citeşte de la tastatură un număr naturale nenul n, apoi se citesc succesiv n valori naturale. Să se verifice dacă printre valorile citite există un palindrom prim.

17. *Cifre nenule*

Se citeşte un număr natural nenul n, apoi se citesc succesiv n valori întregi. Să se verifice dacă oricare dintre cele n valori citite are exact 3 cifre nenule.

18. *Cifra 0*

Scrieţi un algoritm care să citească de la tastatură o succesiune de valori naturale, până la citirea valorii –1 şi care să determine de câte ori apare cifra 0 în scrierea numerelor citite.

19. *Cifra de control*

Fie n un număr natural, citit de la tastatură. Scrieţi un algoritm care calculează şi afişează cifra de control a lui n. Cifra de control a unui număr natural se obţine calculând suma cifrelor numărului, apoi suma cifrelor sumei ş.a.m.d. până la obţinerea unei singure cifre.

De exemplu, pentru n=293 calculăm suma cifrelor 2+9+3=14. Cum suma nu este formată dintr-o singură cifră, repetăm procedeul: 1+4=5. Deci 5 este cifra de control a lui 293.

20. *Pătrate perfecte*

Se citeşte de la tastatură un număr natural n. Scrieți un algoritm eficient care să afişeze toate pătratele perfecte mai mici decât n.

21. *Numere piramidale*

Numerele piramidale se definesc ca fiind sumele parțiale ale şirului pătratelor perfecte 1, 4, 9, 16, 25.... De exemplu, primele 5 numere piramidale sunt 1, 5, 14, 30, 55. Scrieți un algoritm care să afişeze primele n numere piramidale.

22. *Fracții*

Fie n un număr natural nenul. Scrieți un algoritm care să determine numărul de fracții ireductibile care au numitorul şi numărătorul din mulțimea {1, 2, ..., n}. De exemplu, pentru n=3, se pot forma fracțiile ireductibile {1/1, 1/2, 1/3, 2/1, 2/3, 3/1, 3/2}, deci algoritmul va afişa valoarea 7.

23. *Şir*

Să considerăm următorul şir definit prin recurență astfel $s_0=1$, $s_1=2$, $s_2=1$, iar $s_n=s_{n-1}+2*s_{n-2}-s_{n-3}$ (pentru $n\geq 3$). Scrieți un algoritm care să afişeze al n-lea termen din acest şir.

24. *Medie semestrială*

Se citesc n, numărul de note la obținute de un elev la informatică pe parcursul semestrului, apoi se citesc cele n note. Se citeşte apoi nota obținută de elev la teză. Scrieți un algoritm care să calculeze media semestrială a elevului.

25. *C.m.m.d.c.*

Se citeşte n, un număr natural n nenul, apoi se citesc n valori naturale. Scrieți un algoritm care să calculeze cel mai mare divizor comun al celor n numere citite.

26. *Număr*

Se citeşte de la tastatură o succesiune de caractere, terminată cu spațiu (' '). Scrieți un algoritm care să verifice dacă succesiunea de caractere citită poate fi considerată un număr natural scris în baza 10. Modificați algoritmul astfel încât să testeze dacă succesiunea de caractere citită poate fi considerată un număr real.

27. *Cuvinte*

Se citeşte de la tastatură o propoziție constituită din cuvinte separate prin unul sau mai multe spații. Sfârşitul propoziției este marcat de întâlnirea caracterului '.'. Scrieți un algoritm care determină şi afişează numărul de cuvinte din propoziție.

4. EXEMPLE DE IMPLEMENTARE

În capitolul precedent, am descris algoritmii în pseudocod, un ansamblu de convenții ușor de înțeles și de utilizat de către oameni. Din păcate, calculatoarele nu „înțeleg" (deocamdată) un astfel de limbaj. Pentru a rezolva o problemă cu ajutorul calculatorului trebuie să transpunem (să implementăm) algoritmul de rezolvare a problemei într-un limbaj de programare.

Descrierea unui algoritm într-un limbaj de programare se numește ***program***.

Pentru exemplificare, vom implementa în limbajul de programare Pascal și în limbajul de programare C++ trei algoritmi fundamentali, învățați în capitolul precedent.

Algoritmul lui Euclid

Algoritmul lui Euclid determină cel mai mare divizor comun al două numere naturale.

Limbajul Pascal	Limbajul C++

`program Euclid;`
`var n, m, r: integer;`
`begin`
`write('n= '); readln(n);`
`write('m= '); readln(m);`
`while m <> 0 do`
`begin`
`r:=n mod m;`
`n:=m;`
`m:=r;`
`end;`
`writeln('cmmdc=', n);`
`end.`
	`#include <iostream.h>`
	`int main()`
	`{`
	`int n, m, r;`
	`cout<<"n= "; cin>>n;`
	`cout<<"m= "; cin>>m;`
	`while (m)`
	`{r=n%m;`
	`n=m;`
	`m=r;}`
	`cout<<"cmmdc="<<n<<endl;`
	`return 0;`
	`}`

Șirul Fibonacci

În capitolul precedent am elaborat algoritmul de determinare a celui de-al n-lea termen din șirul *Fibonacci*. Amintim că șirul *Fibonacci* este generat după regulile:

$f_1=f_2=1;$

$f_n=f_{n-1}+f_{n-2}$, pentru orice $n>2$.

Limbajul Pascal	Limbajul C++

`program Fibonacci;`
`var n, i, f1, f2, f3: integer;`
`begin`
`write('n= '); readln(n);`
`f2:=1; f3:=1;`
`for i:=2 to n do`
`begin`
`f1:=f2;`
`f2:=f3;`
`f3:=f1+f2;`
`end;`
`writeln(f3);`
`end.`
	`#include <iostream.h>`
	`int main()`
	`{`
	`int n, i, f1, f2, f3;`
	`cout<<"n= "; cin>>n;`
	`f2=f3=1;`
	`for (i=2; i<=n; i++)`
	`{f1=f2;`
	`f2=f3;`
	`f3=f1+f2;}`
	`cout<<f3<<endl;`
	`return 0;`
	`}`

Putere

Să implementăm în limbajul de programare Pascal şi C++ algoritmul de calcul al lui x^n (unde n este un număr natural, iar x un număr real pozitiv).

Limbajul Pascal	Limbajul C++

`program Fibonacci;`
`var n, i: integer;`
`x, p: real;`
`begin`
`write('n= '); readln(n);`
`write('x= '); readln(x);`
`p:=1;`
`for i:=1 to n do`
`p:=p*x;`
`writeln(p:10:2);`
`end.`
	`#include <iostream.h>`
	`int main()`
	`{`
	`int n, i;`
	`float x, p=1;`
	`cout<<"n= "; cin>>n;`
	`cout<<"x= "; cin>>x;`
	`for (i=1; i<=n; i++)`
	`p*=x;`
	`cout<<p<<endl;`
	`return 0;`
	`}`

ANEXA 1. SISTEME DE NUMERAȚIE

Un ***sistem de numerație*** este alcătuit dintr-o mulțime finită de simboluri și un set de reguli de reprezentare a numerelor cu ajutorul simbolurilor respective. Numărul de simboluri constituie ***baza*** sistemului de numerație.

Sistemele de numerație pot fi clasificate din punctul de vedere al semnificației poziției simbolurilor în reprezentarea numerelor astfel:

- sisteme de numerație *poziționale* – în reprezentarea numerelor, poziția simbolurilor este determinantă; de exemplu, sistemul de numerație zecimal, sistemul de numerație binar, sistemul de numerație *Fibonacci* etc.
- sisteme de numerație *nepoziționale* – poziția simbolurilor nu este determinantă în ce privește valoarea acestora în cadrul reprezentării numerelor; de exemplu, sistemul de numerație roman.

Sistemul de numerație roman

Sistemul de numerație roman folosește pentru reprezentarea numerelor simbolurile: I, V, X, L, C, D, M.

Valoarea simbolurilor în sistemul de numerație zecimal este:

Simbol	I	V	X	L	C	D	M
Valoare	1	5	10	50	100	500	1000

Reguli de calcul al valorii numerelor reprezentate în sistemul de numerație roman:

1. Un număr format din unul sau mai multe simboluri identice are valoarea egală cu **suma** valorilor simbolurilor componente.

De exemplu: `II=I+I=1+1=2; XXX=X+X+X=10+10+10=30`

2. Dacă un simbol cu o valoare mai mică *precedă* un simbol cu o valoare mai mare, atunci din valoarea mai mare se **scade** valoarea mai mică.

De exemplu: `IV=V-1=5-1=4; VL=L-V=50-5=45; CM=M-C=1000-100=900`

3. Dacă un simbol cu o valoare mai mică *succedă* un simbol cu o valoare mai mare, atunci cele două valori se **adună**.

De exemplu: `VI=V+I=5+1=6;` `CX=C+X=100+10=110;`

`MLX=M+L+X=1000+50+10=1060`

Se observă că sunt folosite puține simboluri (7) și că regulile de calcul al valorilor nu sunt nici multe (3) și nici complicate. Cu toate acestea apare o problemă care face ca acest sistem de numerație să nu fie folosit; problema constă în faptul că din cele 3 reguli de mai sus **NU** rezultă **UNICITATEA** reprezentării unui număr.

Altfel spus, numărul 47 poate fi (corect) reprezentat prin: XLVII, XXXXVII, XLVXII. Acest fapt constituie una din deficiențele esențiale ale sistemului de numerație roman.

Observați, de asemenea, că sistemul de numerație roman nu este pozițional. De exemplu, în reprezentările numerele IV și VI valoarea simbolurilor I și V este aceeași (1, respectiv 5) independent de poziția lor în cadrul numărului.

Sisteme de numerație poziționale

Până acum, la matematică, am utilizat în special sisteme de numerație poziționale pentru care pozițiile simbolurilor corespund puterilor bazei sistemului de numerație. Mulțimea simbolurilor utilizate pentru reprezentarea numerelor depinde de baza sistemului de numerație. De exemplu, în sistemul de numerație *zecimal* se utilizează 10 simboluri: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9; în sistemul de numerație *binar* se utilizează două simboluri: 0, 1; în sistemul de numerație *octal* se utilizează 8 simboluri: 0, 1, 2, 3, 4, 5, 6, 7; în sistemul de numerație *hexazecimal* se utilizează 16 simboluri: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Semnificația zecimală a simbolurilor A, B, C, D, E, F este, în ordine, 10, 11, 12, 13, 14, 15.

Să notăm cu **b** baza sistemului de numerație (**b** ∈ **N**, **b**>1).

Regula de reprezentare a numerelor în baza **b**

Fiecare cifră dintr-un număr valorează de **b** ori mai mult decât cifra din dreapta sa, deci cifrele numărului, de la dreapta spre stânga, corespund poziționalputerilor bazei sistemului de numerație:

$$x_n x_{n-1} \ldots x_1 x_0 , y_1 y_2 \ldots y_{p(b)} = x_n \cdot b^n + x_{n-1} \cdot b^{n-1} + \ldots + x_1 \cdot b + x_0 + y_1 \cdot b^{-1} + \ldots + y_p \cdot b^{-p}$$

Operații de conversie

Conversia numerelor din baza 10 *în baza* b

Regula de conversie a numerelor naturale

Pentru a converti un număr natural din baza 10 într-o bază **b** (**b**∈**N**, **b**>1), vom face împărțiri succesive la **b** până când obținem câtul 0. La primul pas deîmpărțitul este numărul care se convertește; în continuare, la fiecare pas, câtul de la împărțirea precedentă devine deîmpărțit. Reprezentarea numărului în baza **b** se obține considerând resturile în ordinea inversă obținerii lor.

Exemplu

Pentru a converti $n=1263_{(10)}$ în baza 16 vom face împărțiri succesive la 16:

deîmpărțit	cât	rest
1263	=16*78	+ 15
78	=16* 4	+ 14
4	=16* 0	+ 4

Considerând resturile în ordine inversă obţinem $1263_{(10)}=4EF_{(16)}$.

Regula de conversie a numerelor reale din intervalul (0, 1)

Conversia numerelor subunitare din baza 10 într-o bază **b** (**b**∈**N**, **b**>1), se realizează prin înmulţiri succesive cu **b**, până când se obţine o perioadă sau până când partea fracţionară a rezultatului este 0. La primul pas deînmulţitul este numărul care se converteşte; în continuare, la fiecare pas, partea fracţionară a rezultatului precedent devine deînmulţit. Reprezentarea numărului în baza **b** se obţine considerând cifrele de la partea întreagă a rezultatelor în ordinea obţinerii lor.

Exemplu

Să convertim $n=0,3125_{(10)}$ în baza 2:

```
0,3125  x 2 = 0,625
0,625   x 2 = 1,25
0,25    x 2 = 0,5
0,5     x 2 = 1,0
```

Am obţinut partea fracţionară 0, deci $0,3125_{(10)}=0,0101_{(2)}$.

Regula de conversie a numerelor reale

Pentru a converti un număr real din baza 10 în baza **b** (**b**∈**N**, **b**>1), se convertesc separat partea întreagă şi partea fracţionară, conform regulilor anterioare.

De exemplu, $1263,3125_{(10)}=10011101111,0101_{(2)}$.

Conversia numerelor din baza b *în baza* 10.

Pentru a converti un număr din baza **b** (**b**∈**N**, **b**>1) în baza 10 utilizăm dezvoltarea după puterile bazei sistemului de numeraţie şi efectuăm calculele.

Exemple

1. $1AF_{(16)}=1\cdot16^2+A\cdot16+F=431_{(10)}$
2. $11001,101_{(2)}=2^4+2^3+1+2^{-1}+2^{-3}=25,625_{(10)}$.

Conversia numerelor din baza b *în baza* b'

Conversia numerelor din baza **b** în baza **b'** (**b**, **b'**∈**N**, **b**, **b'**>1) se realizează prin intermediul bazei 10: realizăm conversia din baza **b** în baza 10, apoi din baza 10 în **b'**.

Conversii între baze puteri ale lui 2

Datorită faptului că, deocamdată, calculatoarele utilizează informaţii reprezentate în formă binară, sistemul de numeraţie binar prezintă pentru informaticieni o importanţă deosebită. Calculatoarele par a se descurca foarte uşor cu şiruri de cifre binare, dar oamenii se obişnuiesc mai greu. Frecvent,

informaticienii apelează la conversia şirurilor binare în valori octale sau hexazecimale[1].

Din fericire, operaţiile de conversie între bazele 2 şi 8, respectiv 2 şi 16 se realizează mult mai simplu conversiile între două baze oarecare, datorită faptului că atât 8, cât şi 16 sunt puteri ale bazei 2.

Regula de conversie a numerelor din baza 8 (16) *în baza* 2

Pentru a converti un număr octal (hexazecimal) în binar vom înlocui fiecare cifră numărului cu reprezentarea sa binară pe 3, respectiv 4 poziţii.

Exemple

1. $168AF_{(16)} = 1\ 0110\ 1000\ 1010\ 1111\ _{(2)}$
2. $75210{,}231_{(8)} = 111\ 101\ 010\ 001\ 000,\ 010\ 011011\ _{(2)}$

Regula de conversie a numerelor din baza 2 *în baza* 8 (16)

Pentru a converti un număr binar în baza 8 (16) vom asocia cifrele binare în grupe de câte 3, respectiv 4 începând de la dreapta spre stânga pentru partea întreagă şi de la stânga spre dreapta pentru partea zecimală, completând, eventual, ultimele grupe cu zerouri nesemnificative.

Exemple

1. $110\ 1010\ 1110\ 1010\ 1000\ 0010,\ 1111\ 0000\ 0001\ 11_{(2)} = 6AE82,F01C_{(16)}$
2. $1\ 111\ 001\ 010\ 000\ 110,\ 110\ 101\ 1_{(2)} = 171206,654_{(8)}$.

Pentru aceste conversii este util tabelul următor, care conţine reprezentarea binară a valorilor {0, 1, ..., 15}:

Sistem zecimal	Sistem binar	Sistem zecimal	Sistem binar	Sistem zecimal	Sistem binar	Sistem zecimal	Sistem binar
0	0	4	100	8	1000	12	1100
1	1	5	101	9	1001	13	1101
2	10	6	110	10	1010	14	1110
3	11	7	111	11	1011	15	1111

Sistemul de numeraţie Fibonacci

Cu ajutorul termenilor şirului *Fibonacci* putem defini un sistem de numeraţie poziţional foarte interesant. Mulţimea simbolurilor care reprezintă cifrele acestui sistem de numeraţie este {0, 1}, deci baza sistemului de numeraţie este 2. Regula de reprezentare a numerelor naturale în sistemul de numeraţie *Fibonacci* este de a asocia poziţional cifrelor numărului termeni ai şirului *Fibonacci*, începând cu f_1, şi nu puteri ale bazei.

$$\overline{a_n a_{n-1} \ldots a_1} = a_n \cdot f_n + a_{n-1} \cdot f_{n-1} + \ldots + a_1 \cdot f_1$$

1. De exemplu, informaticienii exprimă uzual adresele de memorie în hexazecimal, nu ca un şir de 32 de cifre binare.

Pentru ca reprezentarea numerelor naturale în sistemul *Fibonacci* să fie unică impunem condiția ca oricare două cifre consecutive să nu fie egale cu 1. Condiția este naturală, deoarece o succesiune de forma ...011... poate fi înlocuită cu ...100... .

Exerciții

1. Să se efectueze următoarele conversii din baza 10 în baza specificată:
   a. 5412, 65536, 256, 65535 în baza 16
   b. 32768, 32767, 312, 0.3175 în baza 2
   c. 49, 720, 153 în baza 7.
2. Să se efectueze următoarele conversii din baza specificată în baza 10:
   a. A015, FFFF, 100 din baza 16
   b. 1000101, 1000000, 111111, 0.1011 din baza 2
   c. 3002, 100, 333 din baza 4.
3. Să se efectueze următoarele conversii între bazele specificate:
   a. 110101010101 din baza 2 în baza 16 și în baza 8
   b. 756201 din baza 8 în baza 2
   c. a09ce3 din baza 16 în baza 2
   d. 562 din baza 8 în baza 2, 7, 10, 16.
4. Scrieți un algoritm care să citească un număr natural b ($2 \le b \le 9$) și un număr natural x scris în baza b. Algoritmul va afișa valoarea numărului x scrisă în baza 10.
5. Scrieți un algoritm care să citească un număr natural b ($2 \le b \le 9$) și un număr natural x scris în baza 10. Algoritmul va afișa în ordine inversă cifrele numărului x scris în baza 10.
6. Scrieți un algoritm de conversie a numerelor naturale din sistemul de numerație zecimal în sistemul de numerație *Fibonacci*.
7. Scrieți un algoritm de conversie a numerelor naturale din sistemul de numerație *Fibonacci* în sistemul de numerație zecimal.

ANEXA 2. SOLUȚII ȘI INDICAȚII

4.1. Noțiunea de algoritm

Expresii

1. `11` ; 2. `15`; 3.`adevărat`; 4.`fals`; expresia testează dacă valorile memorate în variabilele `a` și `b` au aceeași paritate.
5. a. `4ac` nu poate fi numele unei variabile (numele unei variabile nu începe cu o cifră); aparent a fost omis operatorul de înmulțire;
   b. nu există operatorul `^`;
   c. operatorul `şi` se aplică operanzilor logici;
   d. operatorul relațional nu poate fi utilizat înlănțuit;
   e. operatorul `%` se aplică numai operanzilor întregi;
   f. operatorul `!` se aplică operanzilor logici.
6. `a≤x şi x≤b`
7. `a≤x şi x≤b sau c≤x şi x≤d`
8. `x%2=0 şi x>9 şi x<100`
9. `a<b şi b<c`
10. `an%4=0 şi an%100≠0 sau an%400=0`
11. `x-y=1 sau y-x=1`
12. `x*y≠0`
13. a, b.

4.2. Reprezentarea algoritmilor

Structura secvențială

1. `a=23 b=7;` 2. `3412;` 3. `-6;` 4. `b`

5.
```
Date de intrare/ieşire: x1, x2, x3, x4, x5 reale;
Date de manevră: aux real;
Citeşte x1, x2, x3, x4, x5;
aux←x1; x1←x2; x2←x3; x3←x4; x4←x5; x5←aux;
Scrie x1, ' ',x2, ' ', x3, ' ', x4, ' ', x5;
```

6.
```
Date de intrare:  a, b, c reale;
Date de ieşire:   aria, p, sp reale;
Citeşte a, b, c;
p←a+b+c; sp←p/2; aria←√(sp*(sp-a)*(sp-b)*(sp-c));
Scrie p, ' ',aria;
```

7.

```
Date de intrare:  D, H reale;
Date de ieşire:   v    real;
Citeşte D, H;
/* transform distanţa în metri şi timpul în secunde */
D←D*1000; H←H*3600;
v←D/H;
Scrie v;
```

8. Utilizăm formula vitezei v=s/t ⇒ t=s/v. Cei doi colegi se întâlnesc după acelaşi interval de timp, în care primul a parcurs distanţa d1, iar cel de al doilea distanţa D-d1. Obţinem un sistem de ecuaţii prin rezolvarea căruia determinăm d1 şi t:

```
Date de intrare: v1 real; v2 real; D real;
Date de ieşire:  d1 real; /*distanţa parcursă de primul*/
                 t real; /* timpul după care se întâlnesc */
Citeşte D, v1, v2;
d1←v1*D/(v1+v2); t←d1/v1;
Scrie "d1= ", d1, "t= ", t;
```

9.

```
Date de intrare: xA, yA, xB, yB reale;/*coordonatele punctelor*/
Date de ieşire:  L real;              /*lungimea segmentului AB*/
Citeşte xA, yA, xB, yB;
L←√(xA-xB)*(xA-xB)+(yA-yB)*(yA-yB);
Scrie "L= ", L;
```

10. Să urmărim „evoluţia" capetelor balaurului:

Moment	Nr. capete	Explicaţie
Iniţial	6	
Ziua 1	5	Avea 6 capete, Făt-Frumos i-a tăiat un cap, deci au rămas 5
Noaptea 1	11	În locul capului tăiat au crescut 6, deci în total 6+5, iar în locul capului tăiat au crescut 6, deci în total 16
Ziua 2	10	Avea 11 capete, Făt-Frumos i-a tăiat unul, deci au rămas 10
Noaptea 2	16	În locul capului tăiat au crescut 6, deci în total 6+10
Ziua 3	15	Avea 16 capete, Făt-Frumos i-a tăiat unul, deci au rămas 15

Observăm că în ziua n balaurul are 5*n capete.

```
Date de intrare: n natural;
Date de ieşire:  nr natural; /* numărul de capete */
Citeşte n;
nr←5*n;
Scrie "Dupa ", n, " zile balaurul va avea ", nr, " capete";
```

Structura alternativă

1. Valoarea afişată este 3; 2. 5

3.

```
Date de intrare: a, b întregi;
Date de ieşire: rezultatul testului
Citeşte a, b;
Dacă a-b=1 sau b-a=1 atunci Scrie "DA";
                     altfel Scrie "NU";
```

4.

```
Date de intrare: a, b, c reale;
Date de manevră: s, p, d reale;
Date de ieşire:  rezultatul testului
Citeşte a, b, c;
s←-b/a; p←c/a; d←b*b-4*a*c;
Dacă d<0 atunci Scrie "Ecuaţia nu are rădăcini reale";
     altfel
     {Scrie "Ecuaţia are rădăcini reale.";
      Dacă d=0 atunci Scrie "Rădăcinile sunt egale";
               altfel Scrie "Rădăcinile sunt distincte";
      Dacă p<0 atunci Scrie "Rădăcinile au semne contrare"
               altfel
               {Scrie "Rădăcinile sunt ";
                Dacă s>0 atunci Scrie "pozitive";
                         altfel
                         Dacă s<0 atunci Scrie "negative";
                                  altfel Scrie "nule";
               }
      }
```

5.

```
Date de intrare: x natural;
Date de manevră: c1, c2, c3 naturale;
Citeşte x;
/* extrag cele 3 cifre ale numărului */
c3←x%10; c2←x/10%10; c3←x/100;
Dacă c1<c2 Scrie c2, c3;
     altfel Dacă c2<c3 atunci Scrie c1, c3;
                       altfel Scrie c2, c3;
```

6.

```
Date de intrare: c1, c2, c3 caracter;
Date de ieşire:  nr /*numărul de cifre distincte */
Citeşte c1, c2, c3;
Dacă c1=c2 şi c2=c3 atunci nr←1;
     altfel
     Dacă c1=c3 sau c1=c2 sau c2=c3 atunci nr←2;
                                    altfel nr←3;
Scrie nr;
```

7.

Date de intrare: ag, am, as, bg, bm, bs reale;
Date de ieşire: cg, cm, cs reale;
Citeşte ag, am, as, bg, bm, bs;
cs←as+bs; cm←0; cg←0;
Dacă cs>60 **atunci** {cs←cs-60; cm←1;}
cm←cm+am+bm;
Dacă cm>60 **atunci** {cm←cm-60; cg←1;}
cg←cg+ag+bg;
Dacă cg>360 **atunci** cg←cg-360;
Scrie cg, ' grade ',cm, ' minute ', cs, ' secunde';

8.

Date de intrare: a, b, c reale;
Date de ieşire: rezultatul testului şi aria

```
Citeşte a, b, c;
Dacă a*a=b*b+c*c atunci
     Scrie "Triunghi dreptunghic cu aria ", b*c/2;
     altfel
     Dacă b*b=c*c+a*a atunci
          Scrie "Triunghi dreptunghic cu aria ", a*c/2;
          altfel
          Dacă c*c=b*b+a*a atunci
               Scrie "Triunghi dreptunghic cu aria ", b*a/2;
               altfel
               Scrie "NU formeaza un triunghi dreptunghic";
```

9.

```
Date de intrare: x, y reale;
Citeşte x, y;
Dacă x>0 şi y>0 atunci
     Scrie (x+y)/(2*x*y);
     altfel
     Dacă x=0 sau y=0 atunci
          Dacă x>y atunci Scrie x;
                    altfel Scrie y;
          altfel
          Scrie (1/x+1/y)*(1/x-1/y+2*x*y+x*x+y*y);
```

10. Conform restricţiilor problemei există doar 3 variante în care pot fi vizitate cele 3 oraşe (ABC, ACB, BAC). Vom calcula lungimea fiecărui traseu posibil şi vom afişa traseul cel mai scurt.

```
Date de intrare: a, b, c naturale;
Citeşte a, b, c;
Dacă a+b<c+b şi a+b<a+c atunci Scrie "ABC";
     altfel
     Dacă b+c<a+b şi b+c<a+c atunci Scrie "ACB";
          altfel Scrie "BAC";
```

Observaţi că nu am utilizat nici o variabilă suplimentară.

11.

```
Date de intrare: a1, b1, c1, a2, b2, c2 reale;
Date de ieşire: x, y reale;
Citeşte a1, b1, c1, a2, b2, c2;
Dacă a1=0 atunci
    Dacă b1=0 atunci
      Dacă c1=0 atunci
            Scrie "x şi y pot avea orice valoare reală";
            altfel
            Scrie "Sistemul nu are soluţii";
       altfel      /* a1=0, b1≠0 */
       Dacă a2=0 atunci
         Dacă c1*b2=c2*b1 atunci
       Scrie "y=",c1/b1," x poate avea orice valoare reală";
              altfel Scrie "Sistemul nu are soluţii";
          altfel /* a2≠0 */
          Scrie "x=",(c2*b1-c1*b2)/(a2*b1)," y=",c1/b1;
     altfel /* a1≠0 */
     Dacă a1*b2=a2*b1 atunci
          Dacă a1*c2=c1*a2 atunci
               Scrie "x şi y pot avea orice valoare reală";
               altfel
               Scrie "Sistemul nu are soluţii";
          altfel
          {y←(c2*a1-a2*c1)/(b2*a1-a2*b1);
           x←(c1-b1*y)/a1;
           Scrie "x= ", x, "y=", y;}
```

12.

```
Date de intrare: m1, m2, m3, m4, m5 reale;
Date de ieşire: mg real;
Citeşte m1, m2, m3, m4, m5;
Dacă m1<5 sau m2<5 sau m3<5 sau m4<5 sau m5<5 atunci
     Scrie "Nepromovat";
     altfel
     {mg←(m1+m2+m3+m4+m5)/5;
      Dacă mg<6 atunci Scrie "Nepromovat";
                altfel Scrie "Promovat";  }
```

13.

```
Date de intrare: h1, h2, h3 naturale;
Citeşte h1, h2, h3;
Dacă h1≤h2 atunci
    Dacă h2≤h3 atunci
         Scrie "Ionel Gigel Danut"
         altfel  /* h3<h2 */
         Dacă h1≤h3 atunci
              Scrie "Ionel Danut Gigel"
              altfel /* h3<h1 */
              Scrie "Danut Ionel Gigel"
```

```
altfel /* h2<h1 */
Dacă h1≤h3 atunci
    Scrie "Gigel Ionel Danut";
    altfel /*h2<h1, h3<h1 */
    Dacă h2≤h3 atunci
        Scrie "Gigel Danut Ionel"
        altfel /* h3<h2<h1*/
        Scrie "Danut Gigel Ionel";
```

Structura repetitivă

1. a. 36, 30; b. 169, 26 (sau oricare două numere care au cel mai mare divizor comun 13); c. se calculează c.m.m.d.c.(n, m) prin algoritmul lui Euclid, simulând împărțirea prin scăderi repetate.
2. a. 4368; b. Algoritmul extrage succesiv cifrele din numerele a şi b, până când unul dintre numere se termină. La fiecare pas compară ultima cifră din a cu ultima cifră din b, o alege pe cea mai mare şi o adaugă la sfârşitul numărului construit în variabila c, prin urmare o soluție posibilă ar fi 8291; c. o soluție posibilă ar fi a=10, b=10 (sau orice putere a lui 10).
3. a. 5; b. orice număr pentru care cel mai mare divizor prim este 11, de exemplu 11 sau 33 ; c. calculează cel mai mare divizor prim al lui n.
4. a. 78; b. 9; c. Numărul n, considerat a fi scris în baza b, este convertit în baza 10.
5. a. 55; b. a=4 şi b=7; c. orice pereche de valori, pentru care a>b;
   d. în variabila x se calculează suma numerelor din intervalul [a, b]; calculul acestei valori se poate realiza direct, astfel:

```
Dacă a≤b atunci x←b*(b+1)/2-a*(a-1)/2;
          altfel x←0;
```

6. d; 7. c, d; 8. 5.

9. Variabila m nu a fost inițializată cu 0. Ciclul este infinit deoarece se execută numai instrucțiunea m←m+1; (dacă am fi dorit să se execute şi instrucțiunea n←n/p; ar fi trebuit să scriem {m←m+1; n←n/p;}, formând astfel o instrucțiune compusă).

10.

```
Date de intrare: n natural;
Date de manevră: d, s naturale;
Citeşte n;
s←1; /*inițializăm suma divizorilor cu 1, pentru că 1 este
întotdeauna divizor */
Pentru d←2, n/2 execută
       Dacă n%d=0 atunci /* d este divizor al lui n */
            s←s+d;       /* îl adunăm la sumă */
Dacă s=n atunci Scrie "Perfect";
          altfel Scrie "Nu este perfect";
```

11.

```
Date de ieşire: a, b naturale;
Date de manevră: nr, sa, sb, d naturale;
nr←0; /*numărul de perechi de numere prietene determinate
este iniţial 0*/
a←2;
Cât-timp nr<3 execută    /* nu am obţinut încă 3 perechi */
    {sa←1;           /* calculez suma divizorilor lui a */
     Pentru d←2, a/2 executa
       Dacă a%d=0 atunci /* d este divizor al lui a */
            sa←sa+d;    /* îl adunăm la sumă */
     b←sa;
     /* b trebuie să fie egal cu suma divizorilor lui a */
     Dacă b>a atunci
         {sb←1;     /* calculez suma divizorilor lui b */
          Pentru d←2, b/2 executa
                 Dacă b%d=0 atunci  /*d divizor al lui b*/
                      sb←sb+d;    /* îl adunăm la sumă */
          Dacă a=sb atunci
              /*am obtinut o pereche de numere prietene*/
             {Scrie a, ' ', b; /* o afişez */
              nr←nr+1;}       /* si o număr */
         }
     a←a+1; /* caut în continuare */
     }
}
```

12.

```
Date de intrare: x natural;
Date de ieşire:  mare, mic naturale;
Date de manevră: d, prim naturale;
Citeşte x;
mare ← x-1;
Execută
   {prim ← 1;               /*testez dacă mare este prim */
    Dacă mare>2 atunci
        Dacă mare%2 =0 atunci prim ← 0;
            altfel
           {d ← 3;
            Cât-timp d≤√mare şi prim=1 execută
                     Dacă mare%d=0 atunci prim=0;
                          altfel d←d+2;
           }
    Dacă prim≠1 atunci /*caut mai departe */
        mare ← mare-1;
    }
cât-timp prim=0;
```

```
mic ← x+1;
Execută
   {prim ← 1;                    /*testez dacă mic este prim */
    Dacă mic>2 atunci
        Dacă mic%2 =0 atunci prim ← 0;
             altfel
            {d ← 3;
             Cât-timp d≤√mic şi prim=1 execută
                      Dacă mic%d=0 atunci prim=0;
                           altfel d←d+2;
            }
    Dacă prim≠1 atunci /*caut mai departe */
         mic ← mic+1;
    }
cât-timp prim=0;
Scrie mare, ' ', mic;
```

13. Vom inversa numărul citit de la tastatură şi vom testa dacă este egal cu numărul iniţial:

```
Date de intrare:     n natural;
Date de ieşire:      mesajul "DA" sau "NU" (după caz);
Date de manevră:     in natural; /*numărul n inversat*/
                     sn natural;
/*în sn salvăm valoarea lui n, deoarece n se distruge pe
parcursul inversării */
Citeşte n;
sn←n; in←0;
Cât-timp n>0 execută
          {in←in*10+n%10;  n←n/10; }
Dacă sn=in atunci Scrie "DA";
           altfel Scrie "NU";
```

14.

```
Date de intrare:  n natural; a real;
Date de manevră: i natural;
Date de ieşire:   max, min reale;
Citeşte n; /*citesc numărul de valori */
Citeşte  a;  /*  citesc  prima  valoare  separat,  pentru  a
iniţializa maximul şi minimul */
min←a; max←a;
Pentru i←2,n execută
    {Citeşte a;   /*citesc succesiv celelalte n-1 valori */
     /*compar valoarea citită cu maximul şi minimul */
     Dacă max<a atunci max←a;
          altfel
          Dacă min>a atunci min←a;
     }
Scrie min, ' ', max;
```

Observați că, dacă valoarea citită a a fost mai mare decât maximul, nu este necesar să o mai comparăm cu minimul.

15.

```
Date de intrare: n natural; a întreg;
Date de manevră: i natural;
Date de ieşire: s, p întregi;
Citeşte n; /*citesc numărul de valori */
s←0; /*iniţializez suma cu 0, elementul neutru la adunare */
p←1; /*iniţializez produsul cu 1, elementul neutru la înmulţire
*/
Pentru i←1,n execută
      {Citeşte a;              /*citesc succesiv valorile */
       Dacă a%2=0 atunci       /*valoarea este pară */
            s←s+a;             /*o adaug la sumă */
       Dacă a≠0 atunci         /*valoarea este nenulă */
            p←p*a;             /*o înmulţesc cu produsul */
       }
Scrie "s= ",s," p=",p;
```

16.

```
Date de intrare: n, a naturale;
Date de manevră: i, exista, prim, r, sa    naturale;
Date de ieşire:  Rezultatul testului
Citeşte n;                     /*citesc numărul de valori */
exista←0;   /*iniţial nu am găsit nici un palindrom prim */
Pentru i←1,n execută
      {Citeşte a;              /*citesc succesiv valorile */
       prim ← 1;                /*testez dacă a este prim */
       Dacă a>2 atunci
           Dacă a%2 =0 atunci prim ← 0;
                 altfel
                 { d ← 3;
                   Cât-timp d≤√a şi prim=1 execută
                       Dacă a%d=0 atunci prim=0;
                                  altfel d←d+2;
                  }
       Dacă prim=1 atunci /* testez dacă a este palindrom*/
            {
              sa←a;   /*salvez valoarea iniţială a lui a */
              r←0;    /* în r voi calcula inversul lui a */
              Cât-timp a≠0 execută
               {r←r*10+a%10;
        /*lipesc la sfârşitul lui r ultima cifră a lui a*/
                a←a/10;} /*elimin ultima cifră a lui a */
              Dacă sa=r atunci    /* a este şi palindrom */
                   exista←1;
      }
```

```
Dacă exista=1 atunci Scrie "DA";
              altfel Scrie "NU";
```

17.

```
Date de intrare: n, a naturale;
Date de manevră: i, oricare, nr    naturale;
Date de ieşire:  Rezultatul testului
Citeşte n;                    /*citesc numărul de valori */
oricare←1;
/*iniţial consider că toate valorile au proprietatea
cerută*/
Pentru i←1,n execută
     {Citeşte a;              /*citesc succesiv valorile */
 /*extrag cifrele numărului a şi le număr pe cele nenule */
      nr←0;
      Cât-timp a>0 execută
        {Dacă a%10>0 atunci /*ultima cifră din a e nenulă*/
            nr←nr+1;                          /* o număr */
         a←a/10;  }       /*elimin ultima cifră a lui a */
       Dacă nr≠3 atunci
       /*am gasit un număr care nu are 3 cifre nenule */
          oricare ←0;
       }
Dacă oricare=1 atunci Scrie "DA";
               altfel Scrie "NU";
```

18.Voi citi succesiv valorile date şi pentru fiecare valoare citită voi extrage cifrele numărând în variabila nr0 câte cifre egale cu 0 am obţinut.

```
Date de intrare: a natural;
Date de ieşire:  nr0 natural;
nr0←0;
/*iniţial numărul de zerouri existente este 0*/
Execută
  {
    Citeşte a;                        /* citesc o valoare */
     Dacă a≠-1 atunci
  /*extrag cifrele numărului a şi le număr pe cele nule */
          Execută
          {Dacă a%10=0 atunci
                        /*ultima cifră a lui a este 0 */
              nr0←nr0+1;                  /* o număr */
            a←a/10;  }/*elimin ultima cifră a lui a */
          cât-timp a>0;
   }
cât-timp a≠-1;
Scrie nr0;
```

19.Vom aplica repetat algoritmul de calcul al sumei cifrelor unui număr:

```
Date de intrare:  n natural;
```

```
Date de ieşire:    s natural      /*cifra de control*/
Citeşte n;
s←n;
Cât-timp s>9 execută
        {s←0;                  /*calculez suma cifrelor lui n*/
         Cât-timp n>0 execută
                  {s←s+n%10; n←n/10;}
          n←s; }                    /*suma calculată devine n*/
Scrie "cifra de control este ", s;
```

20. Nu vom parcurge toate numerele naturale mai mici decât n testând la fiecare pas dacă numărul curent este sau nu pătrat perfect, ci vom genera direct pătratele perfecte:

```
Date de intrare:  n naturale;
Date de manevră: i natural;
Citeşte n;
i←1;
Cât-timp i*i<n execută
       {Scrie i*i; i←i+1;}
```

21.

```
Date de intrare:  n natural;
Date de manevră: i natural;
Date de ieşire:   s natural;
Citeşte n;
s←0; /*in s calculez suma primelor i pătrate perefecte */
Pentru i←1, n execută
      {s←s+i*i; Scrie s; }
```

22. Vom construi toate fracţiile care au numitorul şi numărătorul în mulţimea {1, 2, ..., n} şi le vom număra numai pe cele ireductibile.

```
Date de intrare:  n natural;
Date de manevră: a, b, sa, sb, r naturale;
Date de ieşire:    nr natural;
a, b, sa, sb, r naturale;
Citeşte n;
nr←1;                              /* numărăm fracţia 1/1 */
Pentru a←1, n-1 execută
       Pentru b←a+1,n execută
         /*verific dacă fracţia a/b este ireductibilă */
             {sa←a; sb←b;
              Cât-timp sb≠0 execută
                  {r←sa%sb; sa←sb; sb←r;  }
              Dacă sa=1 atunci            /* cmmdc(a,b)=1 */
                   nr←nr+2; /*număr fracţiile a/b şi b/a */
            }
Scrie nr;
```

23. În mod similar generării celui de-al n-lea termen al şirului Fibonacci vom utiliza 4 variabile: s0, s1, s2 reţin 3 termeni consecutivi din şir, iar în s3 vom calcula termenul următor; după care ne pregătim pentru pasul următor, deplasându-ne în şir.

```
Date de intrare: n natural;
Date de manevră: i, s0, s1, s2 întregi;
Date de ieşire: s3 întreg;
Citeşte n;
Dacă n=0 sau n=2 atunci Scrie 1;
     altfel
     Dacă n=1 atunci Scrie 2;
          altfel
         {s0←1; s1←2; s2←1;
          Pentru i←3, n execută
                {s3←s2+2*s1-s0; s0←s1; s1←s2; s2←s3;}
          Scrie s3; }
```

24. Calculăm mai întâi MO media la oral, apoi media semestrială se obţine după formula (3*MO+Teza)/4.

```
Date de intrare: n, nota, Teza natural;
Date de manevră: i natural; MO real;
Date de ieşire: mg real;
Citeşte n;
MO←0;
Pentru i←1,n execută
       {Citeşte nota; MO←MO+nota;}
MO←MO/n;
Citeşte Teza;
mg ← (MO*3+Teza)/4;
Scrie mg;
```

25. Pentru a calcula c.m.m.d.c. al n valori ne bazăm pe faptul că operaţia c.m.m.d.c. este asociativă, adică cmmdc($a_1, \ldots, a_{n-1}, a_n$) = cmmdc(cmmdc($a_1, \ldots, a_{n-1}$), a_n)). Vom utiliza o variabilă d în care vom memora la fiecare pas c.m.m.d.c. al valorilor citite până la momentul respectiv. Variabila d va fi iniţializată cu prima valoare citită. La fiecare pas citim o nouă valoare şi calculăm c.m.m.d.c. dintre valoarea citită şi d prin algoritmul lui Euclid.

```
Date de intrare: n, a naturale;
Date de manevră: i, r natural;
Date de ieşire: d natural;
Citeşte n;
Citeşte a; /*citesc prima valoare separat pentru a
iniţializa cmmdc */
d←a;
Pentru i←2,n execută
       {Citeşte a; /*citesc o nouă valoare */
        /*calculez cmmdc între noua valoare citită şi
```

```
          cmmmdc al valorilor citite până acum */
          Cât-timp a≠0 execută
              {r←d%a; d←a; a←r; }
        }
Scrie d;
```

26.Citim succesiv caracterele în aceeaşi variabilă c, până la întâlnirea caracterului spaţiu. Vom utiliza o variabilă logică (Este) care va avea valoarea adevărat dacă secvenţa de caractere poate fi un număr natural şi fals, altfel. La întâlnirea unui caracter diferit de cifră, variabila Este va primi valoarea fals.

```
Date de intrare:     c caracter;
Date de ieşire:      mesajul "DA" sau "NU" (după caz);
Date de manevră:     Este logic;
Este ← adevărat;
Execută
     {Citeşte c;
      Dacă c≠' 'atunci
           Dacă !(c≥'0' şi c≤'9') atunci Este ←fals;
     }
cât-timp c≠' ';
Dacă Este=adevărat atunci Scrie "DA";
                  altfel Scrie "NU";
```

Singura modificare pe care trebuie să o facem pentru ca algoritmul să verifice dacă secvenţa este un număr întreg se referă la primul caracter din secvenţă (care ar putea fi şi '+' sau '-'). Pentru ca numărul să fie real acceptăm şi un singur caracter punct.

27. Citesc propoziţia caracter cu caracter, memorând succesiv caracterele citite în aceeaşi variabilă c.

```
Date de intrare:     c caracter;
Date de ieşire:      nr natural;            /*numărul de cuvinte*/
nr← 0;
Execută
  {Execută                                  /*sar peste spaţii */
    Citeşte c;
   Cât-timp c=' ';
   Dacă c≠'.' atunci                        /*începe un cuvânt */
       {nr←nr+1;                                    /*îl număr */
                                            /*sar peste cuvânt*/
        Cât-timp c≠' ' şi c≠'.' execută  Citeşte c;
       }
    }
Cât-timp c≠'.';
Scrie "nr= ", nr;
```

BIBLIOGRAFIE

1. **Andonie, Răzvan; Gârbacea, Ilie** – *Algoritmi fundamentali. O perspectivă C++*. Editura Libris, Cluj, 1995.
2. **Atanasiu, A.** – *Cum se scrie un algoritm? Simplu.* Editura Agni, Bucureşti, 1993.
3. **Atanasiu, A.; Pintea, R.** – *Culegere de probleme Pascal.* Editura Petrion, Bucureşti, 1996.
4. **Atanasiu, A.** – *Probleme distractive de logică.* Editura Petrion, Bucureşti, 2001.
5. **Bostan, Gh.** – *Culegere de probleme de informatică.* Editura Lumina, Chişinău, 1996.
6. **Cerchez, Em.; Şerban, M.** – *Informatica. Varianta Pascal.* Manual pentru clasa a X-a. Editura Polirom, Iaşi, 2000.
7. **Cerchez, Em.** – *Informatica. Culegere de probleme pentru liceu.* Editura Polirom, Iaşi, 2002.
8. **Cormen, Thomas; Leiserson, Charles; Rivest, Ronald** – *Introduction to Algorithms.* The Massachusetts Institute of Technology, 1990.
9. **Livovschi, Leon; Georgescu, Horia** – *Sinteza şi analiza algoritmilor.* Editura Ştiinţifică şi Enciclopedică, Bucureşti, 1986.
10. **Lucanu, Dorel** – *Bazele proiectării programelor şi algoritmilor.* Editura Universităţii „Al. I. Cuza", Iaşi, 1996.
11. **Mitrana, Victor** – *Provocarea algoritmilor.* Editura Agni, Bucureşti, 1994.
12. **Niculescu, St.; Cerchez, Em.; Şerban, M. ş.a.** – *Bacalaureat şi atestat în informatică.* Editura L&S Infomat, 2000.
13. **S.N.E.E.** – Subiecte bacalaureat, 2000–2002.

CUPRINS