Prelucrări ale cifrelor unui număr natural

1. Algoritmul general pentru prelucrările cifrelor unui număr natural

**I.1. PROBLEMĂ:** Se dă un număr nϵN. Se cere să se determine un rezultat format pe baza cifrelor lui n, cum ar fi:

* O sumă S (exemplu: suma cifrelor, suma acelor cifre care au o anume propietate dată, etc.)
* O valoare maximă/minimă Max/Min(cifra maximă/minimă, cifra maximă/minimă ce îndeplinește o anume proprietate, etc.)
* Numărul de cifre k (total sau doar cele ce au o anume proprietate)
* Un produs P (exemplu: produsul cifrelor, produsul acelor cifre care au o anume propietate, etc.)
* Prima cifră pc
* Un număr x format pe baza cifrelor lui n, cu inversarea ordinii cifrelor din n (de exemplu oglinditul/inversul lui n sau un număr ce prelucrează oglinditul/inversul lui n prin scrierea cifrelor de două ori, prin modificarea cifrelor după un algoritm dat, etc.)
* Un număr y format pe baza cifrelor lui n, cu păstrarea ordinii cifrelor din n (de exemplu complementul față de 9 al lui n sau un număr ce prelucrează cifrele din scrierea lui n prin modificarea lor după un algoritm dat, cu pastrarea ordinii, etc.)
* În n se găsește (măcar) o cifră cu o anume proprietate dată? (gasit va avea inițial valoarea 0 dacă răspunsul la întrebare este DA, respectiv gasit va avea valoarea 1 dacă raspunsul este NU)
* Toate cifrele lui n au o anume proprietate dată? (variabila toate va avea inițial valoarea 0 dacă răspunsul la întrebare este DA, respectiv toate va avea valoarea 1 dacă raspunsul este NU)
* Alte verificări sau rezultate (exemplu: câte perechi de cifre alăturate din scrierea lui n au o anume proprietate specificată – de ex. dacă suma este maxim 12; n are/nu are toate cifrele în ordine crescătoare; un grup de cifre alăturate ce sunt egale între ele sunt egale între ele sau au o anume proprietate specificată, trebuie înlocuit cu o cifră specificată în contextul problemei, etc.)

**Observație:** Faptul că n ar putea fi 0 și faptul că orice număr natural are cel puțin o cifră, face ca în algoritm să fie mai potrivită folosirea instrucțiunii REPETĂ decât CÂT TIMP. Se poate scrie și un algoritm general cu CÂT TIMP, dar atunci trebuie mereu avut în vedere cazul particular n=0.

Algoritmul conține 4 pași, pe care îi puteți exersa pe o multitudine de probleme.

**I.2. PAȘII ALGORITMULUI:**

1. CITEȘTE n

2. Inițializări utile

//[[1]](#footnote-1) s←0 ; Max←0 (sau Max←-1); Min←9 (sau Min←10); k←0; P←1; x←0; y←0; // gasit←0; toate←1

// Uneori e necesară p*ă*strarea valorii ini*ț*iale a lui n, adic*ă* e nevoie s*ă*-i // facem la început o copie, adic*ă* cn←n

3. REPETĂ

3.1. uc←n MOD 10

**3.2. Actualizarea rezultatului problemei, în funcție de valoarea uc**

3.3. n←n DIV 10 (Adică „tai”/elimin ultima cifră din n)

PÂNĂ CÂND n=0

4. Utilizarea/scrierea rezultatelor

1. OBSERVAȚII

**O1.** Inițializările Max← -1 și Min←10 conțin o informație în plus și sunt utile (la pasul 4) dacă se caută o cifră maximă/minimă cu o anume proprietate (e pară/impară/ >c, c e dat sau e valoare număr prim, etc.) și este posibil ca numărul n să nu conțină astfel de cifre.

**O2.** Chiar dacă nu e specificat în problemă, uneori este necesară calcularea unor rezultate auxiliare.

De exemplu, pentru a calcula suma cifrelor pare dintr-un număr, este necesară și calcularea numărului de cifre pare din scrierea lui n. Faptul că, la pasul 4, se obține S=0, nu e o informație suficientă pentru a ști că n are cifre pare și suma lor este 0 (exemplu n=703001) sau nu există suma cifrelor pare din n, numarul neconținând cifre pare (exemplu n=731).

Analog, pentru a determina produsul cifrelor impare dintr-un număr, este necesară și calcularea numărului de cifre impare din scrierea lui n. Faptul că, la pasul 4, se obține P=1, nu e o informație suficientă pentru a ști că n are cifre impare și produsul lor este 1 (exemplu n=601001) sau nu există produsul cifrelor impare din n, numarul neconținănd cifre impare (exemplu n=604008).

**O3.** Pentru „alipirea” la sfârșitul unui număr x a unei cifre uc se folosește instrucțiunea

x←x\*10+uc

Dacă dorim să „alipim” la sfârșitul unui număr x de două ori cifra uc, putem efectua x←x\*100+uc\*11 sau x←x\*10+uc urmat iar de x←x\*10+uc.

**O4.** Dacă cifra pe care dorim să o „alipim” la x se calculează pe baza lui uc printr-un algoritm, atunci vom calcula cifra respectivă la pasul 3.2. și apoi o vom „alipi” la x.

De exemplu, dacă dorim ca numărul n să fie codificat printr-un număr care să conțină, în loc de fiecare cifră pară, cifra 2 și în loc de fiecare cifră impară, cifra 1, cu cifrele codificate în ordine inversă decât la n, cum procedăm?

Pentru clarificarea problemei, cred că sunt utile câteva exemple.

Ex.1: Dacă n=1234, codul său va fi x=2121 (4 e par, 3 e impar, 2 e par, 1 e impar).

Ex.2: Dacă n=1579234, codul său va fi x=2121111 (4 e par, 3 e impar, 2 e par, 9 e impar, 7 e impar, 5 e impar, 1 e impar).

Ex.3: Dacă n=900034, codul său va fi x=212221 (4 e par, 3 e impar, 0 e par, 0 e par, 0 e par, 9 e impar).

Pasul 3.2. devine:

┌Dacă uc MOD 2=1 atunci uc←1

│ altfel uc←2

∟■

x←x\*10+uc

.... sau, mai șmecher, uc←2 - uc MOD 2; x←x\*10+uc

**O5.** Pentru determinarea unui număr y format pe baza cifrelor lui n, cu păstrarea ordinii cifrelor din n, un algoritm eficient va folosi o variabilă P care va reține puterile lui 10 și va fi utilă la pasul 3.2. pentru a „alipi” cifra uc (eventual o cifră obținută ca prelucrare a cifrei uc) în fața lui y. Variabila P, fiind un produs, se va inițializa cu 1 la pasul 2.

Exemplu: Dorim să calculăm y=complementul față de 9 al unui număr n, natural. Acest lucru se face înlocuind fiecare cifră c din număr cu cifra (9 – c) și păstrând ordinea cifrelor.

Ex.1 Dacă n=29935, atunci y=70064.

Ex.2 Dacă n=993985, atunci y=6014 (pentru că cei doi de 0 care apar în fața numărului y, nu se mai scriu)

Ex.3 Dacă n= 2201099000, atunci y=7798900999.

Ex.4 Dacă n=7798900999, atunci y=2201099000.

Observați că, dacă am forma un număr x alipind cifrele complemantare față de 9 și apoi am calcula oglinditul lui x (pe lângă faptul că algoritmul e mai ineficient), nu se va calcula mereu corect rezultatul cerut (vezi exemplul 4). Soluția constă în utilizarea variabilei P.

Algoritmul complet al problemei va fi următorul:

1. CITEȘTE n

2. // Inițializări utile

P←1; y←0;

3. REPETĂ

3.1. uc←n MOD 10

// 3.2. Actualizarea rezultatului problemei, în funcție de valoarea uc

// Vom alipi cifra (9-uc) în fața lui y și apoi e necesară actualizarea lui P

y←P\*(9-uc)+y

P←P\*10

3.3. n←n DIV 10 (Adică „tai”/elimin ultima cifră din n)

PÂNĂ CÂND n=0

4. SCRIE y

**O6.** Dacă se doresc verificări sau rezultate ce implică perechi de cifre alăturate din scrierea lui n ce au o anume proprietate specificată (de exemplu: verificarea dacă n are/nu are toate cifrele în ordine crescătoare/descrescătoare; numărarea perechilor de cifre alăturate din scrierea lui n care conțin una din cifre cu valaoarea dublă decat a celeilalte cifre din pereche, etc.) atunci e necesară, pe lângă uc, încă o variabilă ce va reține cealaltă cifră din pereche și trebuie avut grijă la pasul 3.2. să se actualizeze și această variabilă și posibil completat pasul 2 cu n←n DIV 10.

1. Exemple de PROBLEME REZOLVATE cu acest algoritm

**Problema 1.** Se dă nϵN. Să se calculeze suma cifrelor lui n și prima cifră a lui n.

1. CITEȘTE n

2. // Inițializări utile

S←0;

3. REPETĂ

3.1. uc←n MOD 10

// 3.2. Actualizarea rezultatului problemei, în funcție de valoarea uc

S←S+uc

3.3. n←n DIV 10 (Adică „tai”/elimin ultima cifră din n)

PÂNĂ CÂND n=0

4. SCRIE ”Suma cifrelor este”, S, ” si prima cifra este ”, uc.

**Problema 2.** Se dă n, număr natural. Să se calculeze inv=inversul (sau oglinditul) lui n și apoi să se verifice dacă n este palindrom (adică, citindu-l de la stânga la dreapta și de la dreapta la stânga, se obține aceeași valoare).

Ex1. Dacă n=1234 atunci inv=4321 și n nu e palindrom.

Ex2. Dacă n=4233324 atunci inv=4233324 și n este palindrom.

1. CITEȘTE n

2. // Inițializări utile (Observați că e necesară o copie a lui n!)

cn←n; inv←0;

3. REPETĂ

3.1. uc←n MOD 10

// 3.2. Actualizarea rezultatului problemei, în funcție de valoarea uc

inv←inv\*10+uc

3.3. n←n DIV 10 (Adică „tai”/elimin ultima cifră din n)

PÂNĂ CÂND n=0

4. SCRIE ”Inversul numarului ”, cn, ” este ”, inv

┌DACĂ inv = cn atunci SCRIE cn,” este palindrom”

│ ALTFEL SCRIE cn, ” nu este palindrom”

∟■

**Problema 3.** Se dă n, număr natural. Să se verifice dacă n conține măcar o cifră pară și să se afișeze un mesaj corespunzător.

1. CITEȘTE n

2. // Inițializări utile

gasit←0;

3. REPETĂ

3.1. uc←n MOD 10

// 3.2. Actualizarea rezultatului problemei, în funcție de valoarea uc

┌DACĂ uc MOD 2 = 0 ATUNCI

│ gasit←1

∟■

3.3. n←n DIV 10 (Adică „tai”/elimin ultima cifră din n)

PÂNĂ CÂND n=0

4. ┌DACĂ gasit = 0 atunci SCRIE ”Numarul nu are cifre pare”

│ ALTFEL SCRIE ”Numarul are (minim) o cifra pară”

∟■

OBSERVAȚIE: Algoritmul devine mai eficient dacă condiția n=0 din finalul instrucțiunii REPETĂ se înlocuiește cu (n=0 SAU gasit =1).

**Problema 4.** Se dă n, număr natural. Să se verifice dacă toate cifrele din n sunt cifre pare și să se afișeze un mesaj corespunzător.

1. CITEȘTE n

2. // Inițializări utile

toate←1;

3. REPETĂ

3.1. uc←n MOD 10

// 3.2. Actualizarea rezultatului problemei, în funcție de valoarea uc

┌DACĂ uc MOD 2 = 1 atunci

│ toate←0

∟■

3.3. n←n DIV 10 (Adică „tai”/elimin ultima cifră din n)

PÂNĂ CÂND n=0

4. ┌DACĂ toate = 1 atunci SCRIE ”Numarul are toate cifrele pare”

│ ALTFEL SCRIE ”Numarul nu are toate cifrele pare ”

∟■

OBSERVAȚIE: Algoritmul devine mai eficient dacă condiția n=0 din finalul instrucțiunii REPETĂ se înlocuiește cu (n=0 SAU toate =0).

**Problema 5.** Se dă n, număr natural. Să se determine numărul natural y1 format pe baza cifrelor lui n înlocuind cifra 5 cu cifra 7. (Varianta 2 a problemei înlocuiește cifrele pare cu 0 și determină y2.)

Exemplu: Dacă n=2550673, rezultatul va fi y1=277073.

(Exemplu pentru varianta 2 a problemei: Dacă n=2550673, rezultatul va fi y2=550073.)

1. CITEȘTE n

2. // Inițializări utile

P1←1; y1←0; (P2←1; y2←0; )

3. REPETĂ

3.1. uc←n MOD 10

// 3.2. Actualizarea rezultatului problemei, în funcție de valoarea uc

// Vom alipi cifra uc în fața lui y1 (respectiv y2, pentru varianta 2 a problemei), și apoi e necesară actualizarea lui P1 (sau P2, după caz)

┌DACĂ uc = 5 ATUNCI

│ uc←7

**∟**■

y1←P1\*uc+y1

P1←P1\*10

// ┌DACĂ uc MOD 2 = 0 ATUNCI

// │ uc←0

// **∟**■

// y2←P2\*uc+y2

// Cele 4 rânduri anterioare pot fi înlocuite cu y2←P2\*uc\*(uc MOD 2)+y2

// P2←P2\*10

3.3. n←n DIV 10 (Adică „tai”/elimin ultima cifră din n)

PÂNĂ CÂND n=0

4. SCRIE ”y1=”, y1

**//** 4. SCRIE ”y2=”, y2

**Problema 6.** Se dă n, număr natural. Să se calculeze numerele y1 și y2; y1 va conține doar cifrele impare din n (sau valoarea 0 dacă n nu are cifre impare) și y2 va conține doar cifrele pare din n (sau valoarea 0 dacă n nu are cifre pare). Rezultatele y1 și y2 vor conține cifrele preluate din n, păstrând în scrierea lor ordinea cifrelor ca și în n.

Exemplu: Dacă n=1227905349, atunci y1=179539 și y2=2204.

1. CITEȘTE n

2. // Inițializări utile

Py1←1; y1←0; Py2←1; y2←0;

3. REPETĂ

3.1. uc←n MOD 10

// 3.2. Actualizarea rezultatului problemei, în funcție de valoarea uc

// Vom alipi cifra uc în fața lui y1 sau y2, după caz, și apoi e necesară actualizarea lui Py1 sau Py2, după caz

┌DACĂ uc MOD 2 = 0 ATUNCI

│ y2←Py2\*uc+y2

│ Py2←Py2\*10

│ ALTFEL

│ y1←Py1\*uc+y1

│ Py1←Py1\*10

∟■

3.3. n←n DIV 10 (Adică „tai”/elimin ultima cifră din n)

PÂNĂ CÂND n=0

4. SCRIE ”y1=”, y1, ” y2=”, y2.

OBSERVAȚIE: Examinați plasarea instrucțiunilor care modifică valorile variabilelor Py1 și Py2 pe ramurile instrucțiunii DACĂ. Ce rezultate s-ar obține dacă acele instrucțiuni ar fi fost scrise după instrucțiunea ”DACĂ uc MOD 2 = 0” ?

1. Exemple de PROBLEME PROPUSE pentru acest algoritm

**Problema 7.** Se dă n, număr natural. Să se calculeze numarul R1 format de o furnicuță care „ocolește” numărul n pornind (pe „sus”) de la început la sfârșit (respectiv R2, pornind de la sfârșit spre început) și apoi revine (pe „jos”) de unde a plecat și ”culege” de pe traseu mai întâi cifrele pare iar la revenire culege cifrele impare.

Exemple: n=122397004 → R1=220047931 (R2=400221397)

n=137795 → R1=597731 (R2=137795)

Provocare: Se poate scrie un algoritm care să folosească o singură instrucțiune Repetă?

**Problema 8.** Se dă n, număr natural. Să se calculeze numarul R format pe baza cifrelor lui n, preluate în aceeași ordine ca în n doar că, dacă există un grup de cifre pe poziții alăturate având aceeași valoare, grupul va fi reprezentat în rezultat printr-o singură cifră.

Exemple: n=1337888809 → R=137809

n=40000055 → R=405

n=17839 → R=17839

**Problema 9.** *(Cu șiruri de caractere, pentru clasa a XI-a)*Fișierul cifric2.in conține pe primele două rânduri două numere naturale, fiecare având maxim 200 cifre (doar caractere de la ‘0’ la ‘9’). Numim șir\_cifric al unui număr natural n, șirul cifrelor rezultatului corespunzător lui R explicat la problema anterioară. Concepeți un program în C/C++ care verifică dacă numerele scrise în fișierul cifric2.in au același șir\_cifric și afișează în fișierul cifric2.out un mesaj corespunzător (”Da” sau ”Nu”).

1. Folosesc // la începutul unui rând de comentariu [↑](#footnote-ref-1)