PRELUCRAREA CIFRELOR UNUI NUMĂR

prof. dr. Daniela Marcu
prof. Ovidiu Marian Marcu

Exemple de algoritmi care prelucrează cifrele unui număr:

- construirea unui număr folosind cifrele altui număr: oglindit, palindrom
- cifra de control
- suma, produsul, factorialul cifrelor

Toți algoritmii au în comun: izolarea cifrei curente/cifrelor și utilizarea ei/lor.

Pentru a prelua ultima cifră dintr-un număr, efectuăm o împărțire la 10. Restul va fi cifra, câtul va fi numărul obținut prin eliminarea acesteia. În aceeași manieră se procedează pentru ultimele două cifre (împărțire la 100), etc. Pseudocodul de mai jos descrie preluarea ultimei cifre (cif), afișarea ei și eliminarea din numărul n.

```
citește n
cât timp n ≠ 0 execută
    cif ← n % 10  // Preluăm ultima cifră curentă
    scrie cif     // Aici poți face orice procesare (afisare, sumă etc.)
    n ← [n / 10] // Eliminăm cifra prelucrată pentru a trece la următoarea
sfârșit_cât_timp
```

Varianta cu instrucțiune repetitivă cu test final ne ”salvează” atunci când numărul de procesat este 0 inițial!

```
citește n
execută
    cif ← n % 10
    scrie cif
    n ← [n / 10]
cât timp n ≠ 0
```

Prelucrarea a câte două cifre este descrisă mai jos:

```
citește n
cât timp n ≠ 0 execută
    doua ← n % 100    // Preluăm ultimele două cifre
    scrie doua // Procesăm cifrele (ex: dacă n=12345, doua va fi 45)
```

```
    n ← [n / 100]     // Eliminăm ultimele două cifre
sfârșit_cât_timp
```

Construirea unui număr prin preluarea lor din n

1. **Plasarea cifrelor la dreapta noului număr (la sfârșit) - Oglindit (Palindrom)**

Enunt: Se citește un număr natural nenul n. Construiți un număr care sa fie format din toate cifrele lui n, de la dreapta la stânga. Un astfel de număr se numește oglindit. Exemplu: n=123 va da oglinditul 321.

Pseudocod	C++

citește n
ogl ← 0
cât timp n ≠ 0 execută
    ogl ← ogl * 10 + n % 10
// Adăugăm ultima cifră a lui n în oglindit
    n ← [n / 10]
// Eliminăm ultima cifră din n
sfârșit_cât_timp
scrie ogl
	#include <iostream>
	using namespace std;
	int main() {
	    int n;
	    cout << "n= ";
	    cin >> n;
	    int ogl = 0;
	    while (n != 0)
	    {
	        ogl = ogl * 10 + n % 10;
	        n = n / 10;
	    }
	    cout << ogl << endl;
	    return 0;
	}

Palindrom: un număr este palindrom dacă este egal cu oglinditul său.

Pseudocod	C++

citește n
copie_n ← n
ogl ← 0
cât timp n ≠ 0 execută
    ogl ← ogl * 10 + n % 10
    n ← [n / 10]
sfârșit_cât_timp
 	
dacă copie_n = ogl atunci
    scrie "Este palindrom"
altfel
	#include <iostream>
	using namespace std;
	
	int main() {
	    int n;
	    cout << "n= ";
	    cin >> n;
	    int **copie_n = n**;
	    int ogl = 0;
	
	    while (n != 0) {

```
    scrie "Nu este palindrom"
sfârșit_dacă
```

```
        ogl = ogl * 10 + n % 10;
        n = n / 10;
    }

    if (copie_n == ogl)
        cout << "Este palindrom";
    else
        cout << "Nu este palindrom";
    return 0;
}
```

2. Plasarea cifrelor la sânga noului număr (în față)

Enunt: Se citește un număr natural nenul n. Construiți un număr care sa fie format din cifrele pare ale acestuia, în ordinea apariției în n. Exemplu: n=1234 se va construi x=24.

Pseudocod

```
citește n
x ← 0
p ← 1
// p va reține puterea lui 10 pentru poziția cifrei în noul număr

cât timp n ≠ 0 execută
    cif ← n % 10
    dacă cif % 2 = 0 atunci
        x ← cif * p + x
//Adăugăm cifra în fața numărului x
        p ← p * 10
// Următoarea cifră pară va fi pusă cu o poziție mai la stânga
    sfârșit_dacă
    n ← [n / 10]
sfârșit_cât_timp

dacă p = 1 atunci
    scrie "Nu există cifre pare"
altfel
    scrie x
sfârșit_dacă
```

C++

```
#include <iostream>
using namespace std;

int main() {
    int n;
    cout << "n= ";
    cin >> n;
    int x = 0;
    int p = 1;
    while (n != 0) {
        int cif = n % 10;
        if (cif % 2 == 0) {
            x = cif * p + x;
            p = p * 10;
        }
        n = n / 10;
    }

    if (p == 1)
        cout << "Nu exista cifre pare";
    else
        cout << x;
    return 0;
}
```

Cifra de control

Cifra de control a unui număr natural este o singură cifră (între 0 și 9) obținută prin adunarea repetată a cifrelor acelui număr, până când rezultatul devine mai mic decât 10.

Exemplu: n= 458

- Pasul 1: adunăm cifrele: 4 + 5 + 8 = 17 (nu este o singură cifră)
- Pasul 2: adunăm cifrele rezultatului: 1 + 7 = 8
- Cifra de control a lui 458 este 8.

Pseudocod

```
citește n
cât timp n > 9 execută
    S ← 0
    cât timp n ≠ 0 execută
        S ← S + n % 10
        n ← [n / 10]
    sfârșit_cât_timp
    n ← S
// Noul număr devine suma cifrelor sale
sfârșit_cât_timp
scrie n
```

C++

```
#include <iostream>
using namespace std;

int main()
{
    int n;
    cout << "n= ";
    cin >> n;
    while (n > 9) {
        int S = 0;
        while (n != 0) {
            S = S + n % 10;
            n = n / 10;
        }
        n = S;
    }
    cout << n;
    return 0;
}
```

Bazându-ne pe aritmetica modulară și pe criteriul de divizibilitate cu 9, putem face următoarele aprecieri, care vor face aflarea cifrei de control mult mai rapidă:

Considerăm un număr de 3 cifre, scris în baza 10 sub forma:

$$\overline{abc} = a \cdot 100 + b \cdot 10 + c$$

Putem să rescriem puterile lui 10 (100 și 10) ca fiind (99 + 1) și (9 + 1):

$$\overline{abc} = a \cdot (99 + 1) + b \cdot (9 + 1) + c$$

Dacă desfacem parantezele, obținem:

$$\overline{abc} = (a \cdot 99 + b \cdot 9) + (a + b + c)$$

Calculăm $\overline{abc}$ mod 9 și obținem:

$$\overline{abc} \pmod 9 = (a + b + c) \pmod 9$$

De aici, rezultă o proprietate interesantă: cifra de control se obține prin simpla împărțire cu rest la 9!

Atenție la cazul particular când numărul este divizibil cu 9! Cifra de control este 9 în acest caz.

Pseudocod	C++

citește n
dacă n = 0 atunci
    cc ← 0 //cifra ce control
altfel
    dacă n % 9 = 0 atunci
        cc ← 9
    altfel
        cc ← n % 9
    sfârșit_dacă
sfârșit_dacă
scrie cc
	#include <iostream>
	using namespace std;
	
	int main()
	{
	    int n;
	    cout << "n= ";
	    cin >> n;
	    int cc;
	    if (n == 0)
	        cc = 0;
	    else
	        if (n % 9 == 0)
	            cc = 9;
	        else
	            cc = n % 9;
	    cout << cc;
	    return 0;
	}

Putem afla cifra de control și în alt mod, folosind o singură structură repetitivă!

Observație: deoarece adunăm câte o singură cifră (de la 0 la 9) la o sumă S care este deja cel mult 9, valoarea maximă pe care o poate atinge S după adunare este 9 + 9 = 18.

Orice număr între 10 și 18 este format din două cifre: cifra zecilor (care este mereu 1) și cifra unităților. Formula [S / 10] + S % 10 separă aceste două cifre și le adună.

Exemplu: Dacă S devine 13, [13 / 10] + 13 % 10 dă rezultatul 1 + 3 = 4. Suma este redusă instantaneu la o singură cifră!

Deci, pentru a folosi o singură buclă, trucul constă în a aduna cifrele direct în timp ce le extragem, iar dacă în vreun moment suma noastră parțială depășește 9 (adică devine formată din două cifre), o spargem pe ea pe loc și îi adunăm cifrele.

Pseudocod

```
citește n
S ← 0
cât timp n ≠ 0 execută
    S ← S + n % 10
    n ← [n / 10]
    dacă S > 9 atunci
        S ← [S / 10] + S % 10
// Reducem suma pe loc la o singură cifră
    sfârșit_dacă
sfârșit_cât_timp

scrie S
```

C++

```
#include <iostream>
using namespace std;

int main() {
    int n;
    cout << "n= ";
    cin >> n;
    int S = 0;
    while (n != 0)
    {
        S = S + n % 10;
        n = n / 10;
        if (S > 9)
          S = S / 10 + S % 10;
    }
    cout << S;
    return 0;
}
```

Criteriul de divizibilitate cu 11

Un număr este divizibil cu 11 dacă diferența dintre suma cifrelor aflate pe poziții alternative este 0 sau un multiplu al lui 11.

Mai simplu: adunăm alternativ cifrele, apoi scădem cele două sume. Dacă rezultatul este divizibil cu 11, atunci și numărul inițial este divizibil cu 11.

Exemplu: numărul 2728

Adunăm cifrele alternativ, de la dreapta la stânga:

$$S_1=8+7=15$$

$$S_2=2+2=4$$

Calculăm diferența:

$$D=15-4=11$$

Deoarece 11 este divizibil cu 11, numărul 2728 este divizibil cu 11.

Pseudocod

```
citește n
S1 ← 0
// Suma cifrelor de pe poziții impare
S2 ← 0
// Suma cifrelor de pe poziții pare
poz ← 1
// Variabilă pentru a urmări poziția cifrei curente
cât timp n ≠ 0 execută
    cif ← n % 10
    dacă poz % 2 ≠ 0 atunci
        S1 ← S1 + cif
    altfel
        S2 ← S2 + cif
    sfârșit_dacă
    poz ← poz + 1
    n ← [n / 10]
sfârșit_cât_timp
// Verificăm criteriul de divizibilitate
dacă (S1 - S2) % 11 = 0 atunci
    scrie " este divizibil cu 11"
altfel
    scrie " nu este divizibil cu 11"
sfârșit_dacă
```

C++

```
#include <iostream>
using namespace std;
int main()
{
    long long n;
    cin >> n;
    int S1 = 0;
    int S2 = 0;
    int poz = 1;
    while (n != 0)
    {
        int cif = n % 10;
        if (poz % 2 != 0)
            S1 = S1 + cif;
        else
            S2 = S2 + cif;
        poz = poz + 1;
        n = n / 10;
    }
    if ((S1 - S2) % 11 == 0)
        cout << "Este divizibil cu 11";
    else
        cout << "Nu este divizibil cu 11";
    return 0;
}
```

Propunem o listă cu probleme ce pot fi lucrate pentru aprofundarea noțiunilor de prelucrare a cifrelor unui număr.

Enunțul pe scurt al problemei	Ce/cum determinăm?
Suma/Produsul cifrelor	Adunăm/ Înmulțim toate cifrele
Cifra maximă	Cea mai mare cifră
Cifra minimă	Cea mai mică cifră
Numărul cifrelor pare	Câte cifre sunt divizibile cu 2
Numărul cifrelor impare	Câte cifre nu sunt divizibile cu 2
Numărul cifrelor zero	Câte zerouri apar
Prima cifră	Se împarte numărul la 10 până devine o cifră
Inversul numărului	Oglinditul
Palindrom	Numărul este egal cu inversul său
Cifre distincte	Verificăm dacă se repetă cifrele
Frecvența cifrelor	De câte ori apare fiecare cifră Se poate utiliza: vector de apariții sau 10 variabile
Cifre în ordine crescătoare	Se ordonează cifrele numărului: Se poate utiliza: vector de apariții sau 10 variabile
Cifre consecutive	Diferența dintre cifre este 1
Cifra de control	Valoare calculată după o regulă, așa cu este prezentat în acest material
Numărul cifrelor divizibile cu 3	Numărăm cifrele cu valorile 0, 3, 6, 9
Numărul cifrelor prime	Numărăm cifrele 2, 3, 5, 7