ALGORITMI CU CICLURI

Definiţie: Algoritmii care pot fi descrişi folosind pe lângă comenzi/instrucţiuni de citire (în pseudocod: citeste), scriere (în pseudocod: scrie) şi cicluri (în pseudocod: repeta...pana, cat_timp...repeta) se numesc *algoritmi cu cicluri*.

R1 (3-4*).

Enunţul problemei: De pe mediul de intrare se citeşte un număr întreg pozitiv *b* şi un şir de valori întregi pozitive terminate printr-o valoare negativă sau 0 (care nu face parte din şir). Să se numere câte valori sunt mai mari strict decât *b*.

De exemplu:
- pentru *b* = 5 şi pentru şirul 3, 7, 4, 10, 2, -3 avem *contor* = 2, iar
- pentru *b* = 5 şi pentru şirul 3, 1, 2, 4, 0 avem *contor* = 0.

Metoda de rezolvare: Termenii şirului vor fi păstraţi pe rând în variabila *a*, iar numărarea se va face folosind variabila *contor*.

La **probleme de contorizare**, variabila *contor* se iniţializează cu 0 (adică presupunem că sunt 0 elemente ce verifică proprietatea considerată), după care parcurgem elementele necesare şi dacă este verificată proprietatea, atunci variabila *contor* se incrementează (se măreşte cu 1 ⇔ încă un element cu proprietatea considerată).

Descrierea algoritmului în pseudocod:

```
citeşte b          *>0
contor ← 0         *deocamdata sunt 0 elem mai mari ca b
repeta
      citeste a
      daca a>b atunci              *se indeplineste conditia
        contor ← contor + 1        *contorul se mareste cu 1
pana a≤0
scrie contor
```

O descriere a algoritmului în C++(CodeBlocks):

```
#include <iostream>
using namespace std;
int a,b,contor;       //orice var globala se init. cu 0
int main()
{
 cout<<"b="; cin>>b;
 cout<<"Dati valori pozitive (la final negativ sau 0): ";
 //contor = 0; //nu ar mai fi necesar
 do
 {
  cin>>a;             //citim cate o valoare
  if (a>b) contor++;  //dc.val.curenta b>a o contorizam
 }
 while (a>0);         //cat timp valoarea citita a fost > 0
 //daca nu, se iese din "do-while"
 cout<<"Sunt "<<contor<<" valori mai mari decat "<<b<<endl;
 return 0;
}
```

R2 (3*).

Enunţul problemei: Se citeşte de la tastatură un număr întreg. Să se determine suma cifrelor sale.

Metoda de rezolvare: Cifra unităţilor se poate calcula prin considerarea restului împărţirii numărului iniţial la 10, apoi în mod similar se determină cifra zecilor, doar că în locul numărului iniţial se consideră câtul împărţirii numărului iniţial la 10, ş.a.m.d.
De exemplu, pentru n = 123:
- cifra unităţilor = restul împărţirii lui 123 la 10 = 3
- cifra zecilor = restul împărţirii lui 12 (=câtul împărţirii lui 123 la 10) la 10 = 2
- cifra sutelor = restul împărţirii lui 1 (=câtul împărţirii lui 12 la 10) la 10 = 1.

Ne oprim când avem un cât al împărţirii nul şi nu mai putem considera restul împărţirii la 10.

Descrierea algoritmului în pseudocod:

```
citeşte n   *intreg
S ← 0       *orice suma se initializeaza cu 0
cat_timp n≠0 repeta   *n are cel putin o cifra
      S ← S + n%10    *adunam ultima cifra a numarului curent
      n ← [n/10]      *mai departe se ia catul
scrie S               *(adica numarul fara ultima cifra)
```

O descriere a algoritmului în C++(CodeBlocks):

```
#include <iostream>
using namespace std;
int n,s; //automat s=0       // n = maxim 2147483648
int main()
{
 cout<<"n="; cin>>n;
 while (n) //sau n!=0 sau n>0 //n are cel putin o cifra nenula
 {
  s += n%10; //se adauga ultima cifra a numarului curent
  n /= 10;  // se sterge ultima cifra
 }
 cout<<"Suma cifrelor este: "<<s<<endl;
 return 0;
}
```

Suplimentar:
Pentru a lucra cu numere cu mai multe cifre, putem considera n de tip real, doar ca algoritmul anterior nu mai este valabil pentru că operatorul "%" nu lucrează cu operatori de tip real. Atunci, putem folosi teorema împărţirii cu rest: $a = b * \text{cât} + \text{rest}$, unde cât = $[a / b]$ şi $a, b \in \mathbf{R}$ => rest = $a - b*[a / b]$. În cazul nostru b = 10 şi $a = n$.

```
#include <iostream>
#include <cmath> //pt floor
using namespace std;
double n,s; //automat s = 0
int main()
{cout<<"n="; cin>>n;
 while (n)
 {s += (n-10*floor(n/10)); //ultima cifra a nr curent
  n = floor(n/10);
  //catul impartirii lui n la 10 => se taie ultima cifra
 }
 cout<<"Suma cifrelor este: "<<s<<endl;
 return 0;
}
```

R3 (3*).

Enunţul problemei: Se citeşte de la tastatură un număr întreg. Să se descrie un algoritm pentru determinarea răsturnatului acestuia. De exemplu răsturnatul lui 7251 este 1527.

Metoda de rezolvare: Se parcurg pe rând cifrele numărului inițial și apoi se taie, în același timp construindu-se pas cu pas răsturnatul prin creșterea rangului a ceea ce s-a construit la pasul anterior și adăugînd cifra curentă.

pas	*m*	*n*
0	0	725**1**
1	0*10+1=1	72**5**
2	1*10+5=15	7**2**
3	15*10+2=152	**7**
4	152*10+7=1527	0

Descrierea algoritmului în pseudocod:

```
citeşte n   *intreg nenul
citeste n
m ← 0                  *initializam rasturnatul cu 0
cat_timp n≠0 repeta
     m ← 10m + n%10    *crestem "in rang" numarul de pana acum
                       *si adaugam ultima cifra a nr. ramas
     n ← n/10          *taiem inca o cifra din nr. ramas
scrie m
```

O descriere a algoritmului în C++(CodeBlocks):

```
#include<iostream>
using namespace std;
int main()
{int n, m;
 cout<<"n="; cin>>n;
 m = 0;
 while (n)
 {   m = m*10 + n%10;
     n /= 10;
 }
 cout<<"Rasturnatul sau este: "<<m<<endl;
 return 0;
}
```

sau, folosind funcție

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

int Rasturnat(int n)
{//valoarea lui n va ramane aceeasi dupa incheierea functiei
    int m = 0;
    while (n)
    {
        m = m*10 + n%10; //m-ul ant.creste in rang
                         //si se adauga ultima cifra din n-ul curent
        n /= 10; //n-ului curent i se sterge ultima cifra
    }
    return m;
}
```

```
int main()
{
 int n;
 cout<<"n="; cin>>n;
 cout<<"Rasturnatul lui "<<n<<" este: "<<Rasturnat(n)<<endl;
 return 0;
}
```

R4 (4*).

Enunţul problemei: Se citeşte de la tastatură un număr natural *n* nenul. Să se descrie un algoritm pentru determinarea cifrei maxime a numărului dat. De exemplu, pentru numărul *n* = 2515323 cifra maximă este 5.

Metoda de rezolvare: Se presupune că cifra maximă este cifra unităților, apoi se parcurg și celelalte cifre și se actualizează valoarea cifrei maxime dacă cifra curentă depășește valoarea maximului de până la momentul respectiv.

O descriere a algoritmului în C++(CodeBlocks):

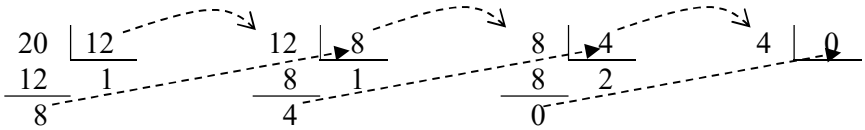
```
#include<iostream>
using namespace std;
int main()
{
 int n,ciframaxima;
 cout<<"n="; cin>>n;
 if (n<=0) return 0;
 ciframaxima = n%10; //cifra unitatilor
 n /= 10; //"taiem" cifra unitatilor
 while (n>0)
 {
     if (n%10 > ciframaxima) //cifra curenta > maximul de pana acum
        ciframaxima = n%10;  //actualizam cifra maxima
     n /= 10; //"taiem" cifra curenta
 }
 cout<<"Cifra max. a numarului dat este: "<<ciframaxima<<endl;
 return 0;
}
```

R5 (4*).

Enunţul problemei: Să se determine cel mai mare divizor comun a două numere naturale nenule.

Metoda de rezolvare: O primă metodă de determinarea celui mai mare divizor comun se bazează pe **algoritmul lui Euclid (împărţiri repetate)**.

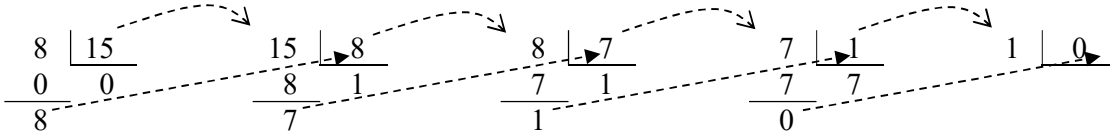
De exemplu, pentru *a* = 20 şi *b* = 12:



=> cmmdc(20,12) = 4. Verificare: 20 = 2^2 * 5, 12 = 2^2 * 3 => cmmdc = elementele comune la puterea cea mai mica = 2^2 = 4.

Se împarte întâi *a* la *b*, apoi la următoarea împărţire: deîmpărţitorul devine *b* (adică împărţitorul de la împărţirea curentă) şi împărţitorul devine restul împărţirii curente, ş.a.m.d până când se obţine rest 0; atunci cel mai mare divizor comun este ultimul rest nenul = împărţitorul de la ultima împărţire.

Pentru $a = 8$ şi $b = 15$:



Descrierea algoritmului în pseudocod:

```
citeşte a,b      *intregi nenule
repeta
      rest ← a % b
      a ← b           *pregatim urmatoarea impartire
      b ← rest
pana b = 0
scrie a
```

O descriere a algoritmului în C++(CodeBlocks):

```
//c.m.m.d.c. prin împărţiri repetate (alg. lui Eulid)
#include <iostream>
using namespace std;
int a, b, deimp, imp, rest;
int main()
{do  //controlam datelor de intrare (vrem a!=0 si b!=0)
 {cout<<"a (!=0) = "; cin>>a;
  cout<<"b (!=0) = "; cin>>b;
 } while ((a==0) || (b==0));
   //cat timp datele sunt necorespunzatoare
 deimp = a; //initial deimpartitul este a
 imp = b; //initial impartitorul este b
 //am facut copii pentru a folosi valorile initiale ale lui
 //a si b la final
 do
 {
  rest = deimp%imp;
  deimp = imp;
  imp = rest;
 }
 while (imp); //sau while(imp!=0);
 cout<<"cmmdc("<<a<<","<<b<<") = "<<deimp<<endl;
 return 0;
}
```

Observatie: Se poate evita folosirea variabilelor `deimp` şi `imp`, prin modificarea directă a variabilelor *a* şi *b*, doar că la final nu mai putem face apel la valorile iniţiale ale acestora.

O altă variantă de program C++ este aceea în care se foloseşte funcţie pentru determinarea celui mai mare divizor comun dintre două numere:

```
#include <iostream>
using namespace std;
int cmmdc(int a, int b)
{//functie pentru determinarea cmmdc dintre a si b, nenule
 int rest;
 do {
  rest = a%b;
  a = b;
  b = rest;
 } while (b); //sau while(b!=0);
 return a;
}
```

```
int main()
{
 int a,b;
 do  //fortam datele de intrare sa fie ambele nenule
 {
  cout<<"a (!=0) = "; cin>>a;
  cout<<"b (!=0) = "; cin>>b;
 } while ((a==0) || (b==0)); //sau (!(a!=0 && b!=0))
   //cat timp vreuna din date este necorespunzatoare
 cout<<"cmmdc("<<a<<","<<b<<")="<<cmmdc(a,b)<<endl;
 return 0;
}
```

O altă variantă este cea cu **scăderi repetate**:
De exemplu, pentru $a = 20$ şi $b = 12$:

$a = 20 > b = 12 \Rightarrow a \leftarrow a-b \Rightarrow a = 20-12 = 8$
$a = 8 < b = 12 \Rightarrow b \leftarrow b-a \Rightarrow b = 12-8 = 4$
$a = 8 > b = 4 \Rightarrow a \leftarrow a-b \Rightarrow a = 8-4 = 4$
$a = 4 = b = 4$. Stop. cmmdc(20,12) = 4.

Descrierea algoritmului în pseudocod:

```
citeşte a,b
cat_timp a≠b repeta
    daca a>b atunci
        a ← a-b
    altfel
        b ← b-a
scrie a
```

O descriere a algoritmului în C++(CodeBlocks):

```
 //c.m.m.d.c. prin scăderi repetate
#include <iostream>
using namespace std;
int main()
{
 int a,b;
 do
 {
  cout<<"a="; cin>>a;
  cout<<"b="; cin>>b;
 } while ((a<=0) || (b<=0));
while (a!=b)
  if (a>b) a -= b;
  else b -= a;
 cout<<"cmmdc="<<a<<endl;
 return 0;
}
```

sau folosind o funcţie pentru determinarea celui mai mare divizor comun prin scăderi repetate:

```
#include <iostream>
using namespace std;
int cmmdc(int a, int b)
{while (a!=b)
  if (a>b) a -= b;
  else b -= a;
 return a;
}
```

```
int main()
{int a,b;
 do  //controlam datelor de intrare (vrem a!=0 si b!=0)
 {cout<<"a (!=0) = "; cin>>a;
  cout<<"b (!=0) = "; cin>>b;
 } while ((a==0) || (b==0));
 //cat timp datele sunt necorespunzatoare
 cout<<"cmmdc("<<a<<","<<b<<")="<<cmmdc(a,b)<<endl;
 return 0;
}
```

R6 (3*).

Enunțul problemei: Descrieți un algoritm pentru determinarea puterii unui factor prim *d* ce apare în descompunerea unui număr *n* dat. De exemplu, pentru *n* = 20 și *d* = 2, puterea este 2 ($20 = 2^2 * 5^1$), iar pentru *n* = 20 și *d* = 3, puterea este 0.

Metoda de rezolvare: O variantă este aceea de a contoriza de cîte ori se împarte *n* la *d*.

O descriere a algoritmului în C++(CodeBlocks):

```
#include<iostream>
using namespace std;
int main()
{
 int n,d,contor;
 cout<<"n="; cin>>n;
 cout<<"d="; cin>>d;
 contor = 0;
 while (n%d==0) //de cate ori n se divide cu d
    {
        n /= d;       //impart
        contor++;     //contorizez
    }
 cout<<d<<" apare in descompunerea lui "<<n<<" la puterea "<<
contor<<endl;
 return 0;
}
```

R7 (3*).

Enunțul problemei: Descrieți un algoritm pentru determinarea cifrei de rang *k* a unui număr natural n nenul dat. De exemplu, pentru *n* = 7432, cifra de rang *k* = 0 este 2 (cifra unităților), cifra de rang *k* = 1 este 3 (cifra zecilor), cifra de rang *k* = 2 este 4 (cifra sutelor), cifra de rang *k* = 3 este 7 (cifra miilor), iar cifra de rang *k* = 6 nu există.

Metoda de rezolvare: O variantă este aceea de a "tăia" *k* cifre din numărul dat, apoi de a afișa ultima cifră a numărului dat.

O descriere a algoritmului în C++(CodeBlocks):

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

int main()
{
 int n,k,i;
 cout<<"n="; cin>>n;
 cout<<"k="; cin>>k; //rangul cifrei (0,1,2,...)
 for (i=1;i<=k;i++) //de k ori
    n /= 10; //"tai" ultima cifra
 cout<<"Cifra de rang "<<k<<" este: "<<n%10<<endl;
```

```
 return 0;
}
```

R8 (3*-suplimentar).

Enunţul problemei: Descrieți un algoritm pentru rotunjirea unui număr dat la cifra miilor. De exemplu, pentru n = 27 432, rotunjirea la cifra miilor este 27 000 pentru că cifra sutelor este 4 < 5), iar pentru n = 27 832, rotunjirea la cifra miilor este 28 000 pentru că cifra sutelor este 8 ≥ 5).

Metoda de rezolvare: O variantă este aceea de a "tăia" 2, ne uităm la ultima cifră (cifra sutelor) și dacă aceasta este mai mică decât 5 se afișează numărul rămas fără ultima cifră*1000, altfel se afișează (numărul rămas fără ultima cifră + 1)*1000.

O descriere a algoritmului în C++(CodeBlocks):

```
#include<iostream>
using namespace std;
int main()
{
 int n;
 cout<<"n="; cin>>n;
 n /= 100;
 if (n%10>=5)
    n = (n/10+1) * 1000; //(nr.pana la cifra miilor + 1)*1000
 else
    n = n/10 * 1000; //(nr.cifra miilor) * 1000
 cout<<"Rotunjirea la cifra miilor: "<<n<<endl;
 return 0;
}
```

R9 (4*-suplimentar).

Enunţul problemei: Descrieți un algoritm pentru rotunjirea unui număr nenul dat n la ordinul k (număr natural nenul). De exemplu, pentru n = 27 4**3**2 și k = 1 (rotunjire la cifra zecilor) se obține 27430 pentru că cifra unităților <5, iar pentru n = 27 **4**72 și k = 2 (rotunjire la cifra sutelor) se obține 27 500 pentru că cifra zecilor ≥ 5.

Metoda de rezolvare: O variantă este aceea de a "tăia" k-1 cifre ale numărului dat, ne uităm la ultima cifră (cifra de ordin k-1) a numărului rămas și dacă aceasta este mai mică decât 5 se afișează numărul rămas fără această ultimă cifră*10^(k+1), altfel se afișează (numărul rămas fără ultima cifră + 1)*10^(k+1).

O descriere a algoritmului în C++(CodeBlocks):

```
#include<iostream>
using namespace std;
int main()
{
 int n,k,i,p=1; //p=10^k
 cout<<"n="; cin>>n;
 cout<<"k="; cin>>k;
 for (i=1;i<k; i++)
    { n /= 10; p*=10; } //p=10^(k-1)
 p *= 10; //p=10^(k)
 if (n%10>=5)
    n = (n/10+1) * p;
 else
    n = n/10 * p;
 cout<<"Rotunjirea la cifra de ordin k: "<<n<<endl;
 return 0;
}
```

Temă:

Pentru începători / medii:

1) Se citesc numere reale până la întâlnirea numărului 0. Scrieţi în C++ algoritmul pentru determinarea mediei aritmetice a numerelor citite. Indicaţii: numerele citite (ca la R1) se vor însuma, dar şi contoriza (eventual într-o variabilă contor), la final media aritmetică fiind suma acestora împărţită la numărul lor.
2) Scrieţi în C++ algoritmul pentru determinarea produsului cifrelor unui număr natural. Indicaţii: se parcurg cifrele unui număr ca la R2, doar ca se ia o variabilă produs care se inițializează cu 1, apoi pe parcursul parcurgerii cifrelor numărului dat, se înmulțește valoarea anterioară cu cifra curentă.
3) Descrieţi un algoritm pentru determinarea sumei cifrelor pare şi produsul cifrelor impare ale unui număr natural (dacă nu există cifre pare se va afişa 0, iar dacă nu există cifre impare se va afişa 1). Indicaţii: se ia o variabila *S* inițializată cu 0 și o variabilă produs inițializată cu 1, apoi se parcurg cifrele numărului (ca la R2) și dacă cifra curentă este pară se adună cifra curentă la valoarea anterioară a lui *S*, altfel se înmulțește cifra curentă cu valoarea anterioară a lui *P*.
4) Aducerea unei fracții la formă ireductibilă (simplificarea fracției *a* / *b* prin cmmdc(*a*, *b*)).
5) Descrieți un algoritm prin care să stabiliți dacă un număr dat este palindrom (egal cu numărul citit de la dreapta al stânga) sau nu. De exemplu, 237732 este palindrom, 65756 este tot palindrom, dar 123 nu este palindrom. Idee: răsturnatul este același cu numărul dat.

Pentru avansați:

1) Descompunerea unui număr natural nenul în factori primi.
2) Scrieţi în C++ algoritmul pentru determinarea tuturor numerelor prime cu *n* (*a* şi *b* sunt prime între ele dacă c.m.m.d.c(*a*,*b*)=1) până la un număr întreg *n* dat. De exemplu, până la *n* = 12, numerele prime cu 12 sunt: 1, 5, 7, 11. Indicații: se parcurg toate numerele de la 1 la *n* și se testează dacă acestea sunt prime între ele (adică cmmdc(*i*,*n*)==1), iar în caz afirmativ, se afișează valoarea lui *i*; se folosește o funcție cmmdc.
3) Scrieţi în C++ algoritmul pentru determinarea celui mai mic multiplu dintre două numere naturale nenule. Indicații: se citesc cele 2 numere, apoi se folosește formula cmmmc(*a*, *b*) = *a***b* / cmmdc(*a*, *b*), unde cmmdc este funcție C/C++.
4) Scrieți în C++ algoritmul pentru determinarea celui mai mare divizor comun dintre 3 numere naturale nenule date. Indicaţii: cmmdc(a,b,c) = cmmdc(cmmdc(a,b) , c).
5) Descrieți un algoritm pentru adunarea a două fracţii, prin aducerea la acelați numitor comun cel mai mic multiplu comun) și cu afişarea rezultatului sub formă de fracţie ireductibilă.

   De exemplu, $\overset{3/}{\frac{1}{2}}+\frac{1}{6}=\frac{4}{6}\overset{\backslash 2}{=}\frac{2}{3}$ (cmmmc(2,6)=6, cmmdc(4,6)=2).
6) Să se determine numerele de maxim 4 cifre care sunt egale cu suma cuburilor cifrelor acestuia (un astfel de număr este 153 = 1^3 + 5^3 + 3^3).