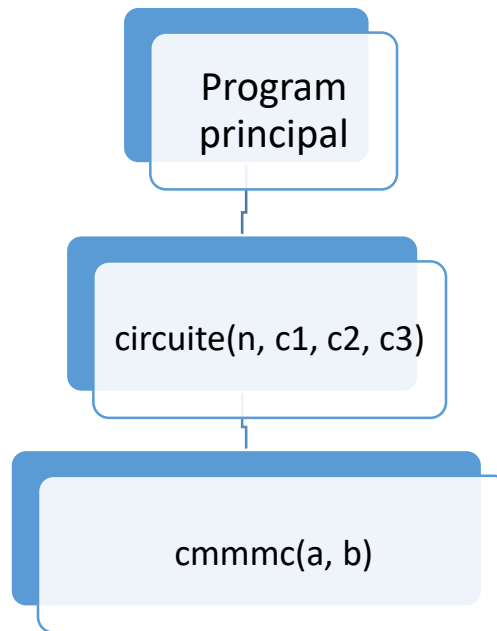


[illegible]



Specificarea subalgoritmilor

Funcția **cmmmc(a, b)**:

Descriere: Calculează cel mai mic multiplu comun a două numere naturale.

Date: a, b – numere naturale.

Rezultate: cel mai mic multiplu comun al celor două numere.

Subalgoritm **circuite(n, c1, c2, c3)**:

Descriere: Calculează numărul de circuite complete pentru fiecare soldat, până la întâlnire.

Date: n – latura domeniului.

Rezultate: c1, c2, c3 – numărul de circuite parcurse de către cei trei soldați.

Implementare

Varianta C++

```

#include <iostream>

using namespace std;

/*
Descriere: Calculeaza cel mai mic multiplu comun a doua numere naturale.
Date: a, b – numere naturale.
Rezultate: cel mai mic multiplu comun al celor doua numere.
*/
int cmmmc(int a, int b)
{
    if (a < b)

```

```

{
    int aux = a;
    a = b;
    b = aux;
}

// cel mai mare numar se aduna repetat cu el insusi
// si se verifica divizibilitatea cu celalalt
int rez = a;
while (rez % b != 0)
    rez += a;

return rez;
}

/*
Descriere: Calculeaza numarul de circuite complete pentru fiecare soldat, pana la
intalnire.
Date: n - latura domeniului.
Rezultate: c1, c2, c3 - numarul de circuite parcurse de catre cei trei soldati.
*/
void circuite(int n, int& c1, int& c2, int& c3)
{
    int l1 = 4 * (n - 1); // lungimea culoarului exterior
    int l2 = 4 * (n - 3); // lungimea culoarului din mijloc
    int l3 = 4 * (n - 5); // lungimea culoarului interior

    int intalnire = cmmmc(cmmmc(l1, l2), l3); // cmmmc al lui l1, l2, l3
    c1 = intalnire / l1;
    c2 = intalnire / l2;
    c3 = intalnire / l3;
}

int main()
{
    int n = 0;
    cout << "Introduceti latura domeniului: ";
    cin >> n;

    int c1 = 0, c2 = 0, c3 = 0;
    circuite(n, c1, c2, c3);
    cout << "Soldatul S1 va face " << c1 << " circuite pana la intalnire." << endl;
    cout << "Soldatul S2 va face " << c2 << " circuite pana la intalnire." << endl;
    cout << "Soldatul S3 va face " << c3 << " circuite pana la intalnire." << endl;

    return 0;
}

```

Varianta Pascal

```

function cmmmc(a,b:integer): integer;
    {se aduna cel mai mare repetat cu el insusi,}
    {si se verifica divizibilitatea cu celalalt}

var aux, rez: integer;
begin
    if (a < b) then

```

```

    begin
        aux := a;
        a := b;
        b := aux;
    end;

    rez := a;
    while (rez Mod b) > 0 do
        rez := rez + a;
    cmmmc := rez;
end;

procedure circuite(n:integer; var c1, c2, c3:integer);
var l1,l2,l3,intalnire: integer;      {nr. de patrutele parcurse pentru un
circuit complet de cei 3 soldati}
begin
    l1:=4*(n-1);
    l2:=4*(n-3);
    l3:=4*(n-5);
    intalnire:=cmmmc(cmmmc(l1,l2),l3);  {cel mai mic multiplu comun al lui
p1,p2,p3}

    c1:=intalnire div l1;                {nr de circuite pentru fiecare soldat}
    c2:=intalnire div l2;
    c3:=intalnire div l3;
end;

var n, c1, c2, c3: integer;
begin

write('Introduceti latura domeniului: ');
readln(n);
circuite(n, c1, c2, c3);
writeln('Soldatul S1 va face ', c1, ' circuite pana la intalnire.');
```

```

writeln('Soldatul S2 va face ', c2, ' circuite pana la intalnire.');
```

```

writeln('Soldatul S3 va face ', c3, ' circuite pana la intalnire.');
```

```

end.

```

Problema 2

Enunț

Să se realizeze câte o funcție recursivă cu un singur parametru (numărul n) pentru:

- determinarea cifrei minime a lui n , se va returna cifra minimă;
- determinarea cifrei pare maxime a lui n , se va returna cifra pară maximă sau -1, dacă n nu are cifre pare;

Analiză

Vom scrie formulele recursive pentru cele două cerințe.

a)

$$cifraMinima(n) = \begin{cases} n, & \text{dacă } n \text{ este format dintr-o singură cifră} \\ \min\{n \bmod 10, \text{cifraMinima}(\lfloor \frac{n}{10} \rfloor)\}, & \text{altfel} \end{cases}$$

O altă variantă recursivă pentru determinarea cifrei minime este:

$$cifraMinima(\overline{a_1 a_2 \dots a_{n-1} a_n}) = \begin{cases} n, & \text{dacă } n \text{ este format dintr-o singură cifră} \\ cifraMinima(\overline{a_1 a_2 \dots \min\{a_{n-1}, a_n\}}), & \text{altfel} \end{cases}$$

În această variantă apelul recursiv se face pentru numărul obținut prin înlocuirea ultimelor două cifre ale sale cu minimul dintre acestea.

b)

$$cifraParaMaxima(n) = \begin{cases} n, & \text{dacă } n \text{ este format dintr-o singură cifră pară} \\ -1, & \text{dacă } n \text{ este format dintr-o singură cifră impară} \\ cifraParaMaxima(\lfloor \frac{n}{10} \rfloor), & \text{dacă } n \bmod 10 \text{ este impară} \\ \max\{n \bmod 10, cifraParaMaxima(\lfloor \frac{n}{10} \rfloor)\}, & \text{dacă } n \bmod 10 \text{ este pară} \end{cases}$$

Specificarea funcțiilor

Funcția **cifraMinima(n)**:

Descriere: Returnează cifra minimă a unui număr dat.

Date: n – număr natural.

Rezultate: cifra minimă a numărului dat.

Funcția **cifraParaMaxima(n)**:

Descriere: Returnează cifra pară maximă a unui număr dat.

Date: n – număr natural.

Rezultate: cifra para maxima a numarului dat.

Implementare

Varianta C++

```
#include <iostream>

using namespace std;

/*
Descriere: Returneaza cifra minima a unui numar dat.
Date: n - numar natural.
Rezultate: cifra minima a numarului dat.
*/
int cifraMinima_V1(int n)
{
    if (n <= 9) // cazul in care numarul este format dintr-o singura cifra
        return n;
    int cifraMinima_restul_numarului = cifraMinima_V1(n / 10);
    return (n % 10 < cifraMinima_restul_numarului ? n % 10 :
cifraMinima_restul_numarului);
}

/*
Descriere: Returneaza cifra minima a unui numar dat.
Date: n - numar natural.
Rezultate: cifra minima a numarului dat.
*/
int cifraMinima_V2(int n)
{
    if (n <= 9) // cazul in care numarul este format dintr-o singura cifra
        return n;
    int minim_ultimele_doua_cifre = n % 10;
    int penultima_cifra = (n / 10) % 10;
    if (penultima_cifra < minim_ultimele_doua_cifre)
        minim_ultimele_doua_cifre = penultima_cifra;
    return cifraMinima_V2((n/100) * 10 + minim_ultimele_doua_cifre);
}

/*
Descriere: Returneaza cifra para maxima a unui numar dat.
Date : n - numar natural.
Rezultate : cifra para maxima a numarului dat.
*/
int cifraParaMaxima(int n)
{
    if (n <= 9) // daca n este format dintr-o singura cifra
    {
        if (n % 2 == 0) // daca este par
            return n;
        else
            return -1;
    }

    int ultima_cifra = n % 10;
    if (ultima_cifra % 2 != 0)
```

```

        return cifraParaMaxima(n / 10);
    int cifraParaMaxima_restul_numarului = cifraParaMaxima(n / 10);
    return (ultima_cifra > cifraParaMaxima_restul_numarului ? ultima_cifra :
cifraParaMaxima_restul_numarului);
}

int main()
{
    int n = 0;
    cout << "Introduceti numarul: ";
    cin >> n;

    cout << "Cifra minima a numarului " << n << " este: " << cifraMinima_V1(n) <<
endl;
    cout << "Cifra minima a numarului " << n << " este: " << cifraMinima_V2(n) <<
endl;
    cout << "Cifra para maxima a numarului " << n << " este: " << cifraParaMaxima(n)
<< endl;

    return 0;
}

```

Varianta Pascal

```

{
Descriere:  Returneaza cifra minima a unui numar dat.
Date: n - numar natural.
Rezultate: cifra minima a numarului dat.
}
function cifraMinima_V1(n: longint): integer;
var cifraMinima_restul_numarului: integer;
begin
    if (n <= 9) then // cazul in care numarul este format dintr-o singura
cifra
        cifraMinima_V1 := n
    else
    begin
        cifraMinima_restul_numarului := cifraMinima_V1(n div 10);
        if ((n mod 10) < cifraMinima_restul_numarului) then
            cifraMinima_V1 := n mod 10
        else
            cifraMinima_V1 := cifraMinima_restul_numarului;
    end;
end;

function cifraMinima_V2(n: longint): integer;
var ultima_cifra, penultima_cifra: integer;
begin
    if(n > 9) then
    begin
        ultima_cifra := n mod 10;
        penultima_cifra := (n div 10) mod 10;
        if (penultima_cifra < ultima_cifra)
            then cifraMinima_V2 := cifraMinima_V2(n div 10)
    end;
end;

```

```

        else cifraMinima_V2 := cifraMinima_V2(((n div 100) * 10) +
ultima_cifra);
    end
    else cifraMinima_V2 := n;
end;

{
Descriere: Returneaza cifra para maxima a unui numar dat.
Date : n - numar natural.
Rezultate : cifra para maxima a numarului dat.
}
function cifraParaMaxima(n: longint): integer;
var ultima_cifra, cifraParaMaxima_restul_numarului: integer;
begin
    if (n <= 9) then {daca n este format dintr-o singura cifra}
        begin
            if (n mod 2 = 0) then {daca este par}
                cifraParaMaxima := n
            else
                cifraParaMaxima := -1;
            end
        end
    else
        begin
            ultima_cifra := n mod 10;
            if (ultima_cifra mod 2 <> 0) then
                cifraParaMaxima := cifraParaMaxima(n div 10)
            else
                begin
                    cifraParaMaxima_restul_numarului := cifraParaMaxima(n div 10);
                    if (ultima_cifra > cifraParaMaxima_restul_numarului) then
                        cifraParaMaxima := ultima_cifra
                    else
                        cifraParaMaxima := cifraParaMaxima_restul_numarului;
                    end
                end
            end;
        end
    end;
end;

var n: longint;
begin
    write('Introduceti numarul: ');
    readln(n);
    writeln('Cifra minima este: ', cifraMinima_V1(n));
    writeln('Cifra minima este: ', cifraMinima_V2(n));
    writeln('Cifra para maxima este: ', cifraParaMaxima(n));

end.

```


Problema 3

Enunț

Fie i, j, k trei numere naturale. Să scrie un subprogram care returnează restul împărțirii numărului (i^j) la k , deci $(i^j) \bmod k$ (iterativ și recursiv).

Exemple:

$$(100^{100}) \bmod 7 = 2$$

$$(125^{199}) \bmod 999 = 800$$

$$(2^{10}) \bmod 9 = 7$$

Analiză

- practic se înmulțesc resturile modulo k ; nu e nevoie de calculul expresiei i^j ;
- se calculează la fiecare pas de înmulțire restul produsului (modulo k);

Aceasta deoarece se știe că:

$$(a * b) \bmod c = (a \bmod c * b \bmod c) \bmod c$$

Atunci:

$$a^x \bmod c = (a \bmod c * a^{x-1} \bmod c) \bmod c$$

Pentru varianta recursivă, formula recursivă se deduce din formula anterioară astfel:

$$rest(i, j, k) = \begin{cases} 1, & \text{daca } j = 0 \\ ((i \bmod k) * rest(i, j - 1, k)) \bmod k & \end{cases}$$

Specificarea funcției

Funcția **Rest(i, j, k)**:

Descriere: returnează restul împărțirii lui (i^j) la k .

Date: i, j, k - numere naturale

Rezultate: R un număr natural: $R \in \{0, 1, \dots, k-1\}$

Implementare

Varianta iterativă C++

```
/*
Descriere: returneaza restul impartirii lui (i^j) la k.
Date: i, j, k - numere naturale
Rezultate: R un numar natural: R in {0, 1, ..., k-1}.
*/
```

```

int Rest(int i, int j, int k)
{
    int r = i % k;
    int rest = 1;
    while (j > 0)
    {
        rest = (rest * r) % k;
        j--;
    }
    return rest;
}

```

Varianta recursivă C ++

```

/*
Descriere:  returneaza restul impartirii lui (i^j) la k.
Date: i,j,k - numere naturale
Rezultate: R un numar natural: R in {0,1,...k-1}.
*/
int Rest(int i, int j, int k)
{
    if (j == 0)
        return 1;
    return ((i % k) * Rest(i, j - 1, k)) % k;
}

```

Varianta iterativă Pascal

```

function Rest(i,j,k:integer):integer;      {functie ce determina restul}
var R,iModK:integer;
begin                                     {e suficient sa inmultim resturile}
    iModK:=i mod k;                      {expresia i Mod k e constanta in ciclu}
    R :=1;
    While(j>0) do
    begin
        R:=(R*(iModK)) mod k;           {atat lui i cat si produsului se aplica mod}
        j:=j-1;
    end;
    Rest:=R;
end;

```

Varianta recursivă Pascal

```

function Rest (i,j,k:integer):integer);
begin
    if(j>0) then Rest:= ((i mod k)*Rest(i,j-1,k)) % k
    else Rest:=1;
end;

```

Exemple

Date de intrare	Rezultate
100, 100, 7	2
125, 199, 999	800
2, 10, 9	7
8, 0, 100	1
5, 151, 5	0

Problema 4

Enunț

Fie x, y două numere naturale. Să scrie un subprogram care determină dacă două numere date sunt **asemenea**, fără a folosi tablouri. (Două numere naturale sunt **asemenea** dacă au aceleași cifre: 2131 e **asemenea** cu 32211 pentru că mulțimea cifrelor este aceeași: $\{1, 2, 3\}$).

Analiză

Problema are o rezolvare simplă dacă se folosește conceptul de vector de apariție pentru cifrele unui număr:

- se determină vectorii de apariție pentru x și y ;
- se compară apoi vectorii de apariție și se decide asemănarea.

Fără utilizarea vectorilor ar trebui să vedem dacă fiecare cifră a lui x este în y și invers.

Vom face 2 subprograme:

- Un subprogram care verifică dacă fiecare cifră a unui număr se află printre cifrele altui număr.
- Un subprogram care apelează apoi de 2 ori primul subprogram.

Specificarea subalgoritmilor

Funcția **CifreAinB(A, B)**:

Descriere: verifică dacă fiecare cifră a lui A este în mulțimea cifrelor lui B.

Date: $A, B > 0$ numere naturale

Rezultate: true, dacă mulțimea cifrelor lui A este inclusă în mulțimea cifrelor lui B;
false, în caz contrar

Funcția **Asemenea(X, Y)**:

Descriere: verifică dubla incluziune a cifrelor lui X și Y.

Date: - $X, Y > 0$ numere naturale

Rezultate: true, dacă **CifreAinB(X, Y)** este true **ȘI** **CifreXinY(Y, X)** este true;
false, în caz contrar

Implementare

Varianta C++

```
/*
Descriere: verifica daca fiecare cifra a lui A este in multimea cifrelor lui B.
Date: A, B > 0 numere naturale
Rezultate: true, daca multimea cifrelor lui A este inclusa in multimea cifrelor lui B;
           false, in caz contrar
*/
```

```

bool CifreAinB(long a, long b)
{
    int copieB = b;      // se retine o copie a lui B
    while (a > 0) // vom verifica daca fiecare cifra a lui A este in B
    {
        int ultima_cifra_A = a % 10;
        while (copieB > 0 && ultima_cifra_A != copieB % 10)
            copieB /= 10;
        if (copieB == 0) // daca s-a ajuns la 0, cifra curenta a lui A nu este
            return false;
        copieB = b;      // se reinitializeaza copia cu numarul initial B
        a /= 10;         // se trece la urmatoarea cifra a lui A
    }
    return true;
}

/*
Descriere: verifica dubla incluziune a cifrelor lui x si y.
Date: x, y > 0 numere naturale
Rezultate: true, daca CifreAinB(x, y) este true SI CifreAinB(y,x) este true;
           false, in caz contrar
*/
bool Asemenea(long x, long y)
{
    return (CifreAinB(x, y) && CifreAinB(y, x));
}

```

Varianta Pascal

```

function CifreAinB(A,B:longint):boolean;
var CopieB :longint;
    UcifA :byte;
begin
    CopieB :=B; {se retine o clona a lui B in CopieB}
    CifreAinB:=true; {presupunem ca incluzinea exista}
    while (A>0) do
    begin
        UcifA:=A mod 10; {verifica fiecare cifra a lui A daca e B}
        B :=CopieB; {la fiecare cifra a lui A se reface B}
        while (B>0) and (UcifA<>(B mod 10)) do B:=B div 10;
        if (B=0) {daca B=0 => cifra curenta lui A nu e in B}
        then
        begin
            CifreAinB:=false; {se pune numele functiei pe false si A pe 0}
            A :=0; {pentru a iesi din primul ciclu while}
        end
        else A:=A div 10; {daca B>0, cifra curenta a lui A e in B}
    end; {se trece la urmatoarea cifra a lui A}
end;

function Asemenea(X,Y:longint):boolean;
begin
    if (CifreAinB(X,Y)) and (CifreAinB(Y,X))
    then Asemenea:=true
    else Asemenea:=false;
end;

```

end;

Exemple

Date de intrare	Rezultate
1222331, 123	true
122235, 123	false
5656565, 56	true
5656565, 5	false
1, 8	false

Probleme tip grilă

1. Știind că variabilele a și i sunt întregi, stabiliți ce reprezintă valorile afișate de algoritmul de mai jos. S-au folosit notațiile $x\%y$ pentru restul împărțirii numărului întreg x la numărul întreg y , și $[x]$ pentru partea întreagă a numărului real x .

```

a ← 10
pentru i ← 1,6 execută
    scrie [a/7]
    a ← a % 7 * 10
sfârșit pentru

```

- primele 6 zecimale ale lui $1/7$
- primele 7 zecimale ale lui $1/6$
- primele 6 cifre ale lui $10/7$
- primele 7 cifre ale lui $10/6$

2. Ce va afișa algoritmul pseudocod de mai jos pentru două numere naturale nenule a și b ? S-a notat cu $x\%y$ restul împărțirii numerelor întregi x și y .

```

citește a,b
c ← 1
cât-timp a * c % b ≠ 0 execută
    c ← c + 1
sfârșit-cât-timp
scrie a*c

```

- a^b
- cel mai mic multiplu comun
- cel mai mare divizor comun
- $a*b$

3. Care este rezultatul execuției următoarei secvențe?

- Afișează 3 pentru $n=123$ și $c=3$;
- Afișează 2 pentru $n=12003$ și $c=0$;
- Afișează n , pentru $n=c=1$;
- Afișează 4, pentru $n=1277771$ și $c=7$
- Afișează 3, pentru $n=12555$ și $c=5$.

```

citește n,c {n, natural>0, c cifra}
z←0
cât timp n%10=c execută
    n←[n/10]
    z←z+1
scrie z

```

4. Care este rezultatul execuției următoarei secvențe?

- Afișează x , dacă $m \leq 0$.
- Afișează x^2 , dacă $m=2$.

```

Citește x,m {x,m întregi}
y←1
Cât timp m>0 execută
    Dacă m%2=0
        atunci m←[m/2]; x←x*x
        altfel m←m-1; y←y*x
Afișează y

```

- c. Afișează 1, pentru $m < 0$.
- d. Afișează x^k , dacă $m = 2^k$.
- e. Afișează un număr negativ dacă $x < 0$.

5. Care funcție schimbă valorile întregi ale lui a și b între ele?

A.

```
void F(int& a, int& b){
    a-=b;
    b+=a;
    a=b-a;
}
```

B.

```
void F(int& a, int& b){
    a=a+b;
    b=a-b;
    a=a-b;
}
```

C.

```
void F(int& a, int& b){
    a=a/b;
    b=a*b;
    a=b/a;
}
```

D.

```
void F(int& a, int& b){
    a=a*b;
    b=a/b;
    a=a/b;
}
```

E.

```
void F(int& a, int& b){
    b=b-a;
    b=a*b;
    a=b;
}
```

6. Care subprogram determină dacă numărul întreg n este prim (*true* dacă n e prim, *false* altfel)?

a)

```
bool Prim(int n){
    int d=2;
    while(d*d<=n){
        if(n%d==0) return false;
        d=(d==2)?3:d+2;
    }
    return true;
}
```

b)

```
bool Prim(int n){
    if(n<=1) return false;
    for (int d=2;d*d<=n;
        d=(d==2)?3:d+2)
        if(n%d==0) return false;
    return true;
}
```

c)

```
bool Prim(int n, int d){//apel
extern, d=2
    if(n<=1) return false;
    if(d*d>n) return true;
    if(n%d==0) return false;
    return Prim(n, (d==2)?3:d+2);
}
```


7. Se consideră funcția de mai jos, care primește ca parametru un număr natural.

```
int f(int a)
{
    int x = a % 10;

    if (x == a)
    if (x % 2 == 0)
        return a;
    else
        return 0;

    if (x % 2 == 0)
        return 10 * f((int)(a/10)) + x;
    return f((int)(a/10));
}
```

- I. De câte ori se apelează funcția pentru $a = 253401976$?
 - a. De 3 ori
 - b. De 9 ori
 - c. De 6 ori
 - d. De 7 ori
- II. Care este rezultatul funcției pentru $a = 253401976$?
 - a. 206
 - b. 53019
 - c. 2406
 - d. 0

8. Ce face subprogramul pentru n întreg.

```
void SP(long n){
    for(int i=2;i<=n;i++)
    {int c=0;
      while(!(n%i)){
        n/=i;
        c+=1;
      }
      if(c) cout<<i<<" "<<c<<endl;
    }
}
```

- a) Afișează perechi de numere prime ($n \geq 2$).
- b) Afișează numerele prime până la n ($n \geq 2$).
- c) Afișează factorii primi ai lui n ($n \geq 2$).
- d) Afișează factorii primi ai lui n și puterile lor ($n \geq 2$).
- e) Subprogramul nu afișează nimic, dacă $n < 2$.