

## Probleme consultații – 12 ianuarie 2019

### Algoritmi care lucrează pe numere

#### Problema 1

##### Enunț:

Scrieti un program care sa determine al n-lea numar din sirul lui Fibonacci.

##### Exemple:

Pentru n=3 => se va afisa 2

Pentru n=5 => se va afisa 5

Pentru n=7 => se va afisa 13

Pentru n=9 => se va afisa 34

Pentru n=10 => se va afisa 55

Pentru n=11 => se va afisa 89

Varianta 1 de rezolvare – *varianta recursiva – implementare C++*

```
#include <iostream>

/* crearea functiei pentru determinarea celui al n-lea numar din sirul lui
Fibonacci
    Date de intrare: numarul n
    Date de iesire: numarul din sir corespunzator
*/

int functiel(int n)
{
    if (n <= 1)
        return n;
    return functiel(n-1) + functiel(n-2);
}

int main()
{
    int n;
    std::cout <<"introduceti numarul n=";
    std::cin >>n;

    std::cout <<"Rezultat= "<<functiel(n)<<std::endl;
    return 0;
}
```

Varianta 1 de rezolvare – *varianta recursiva – implementare Pascal*

```
program problema1;
var numar:integer;
{ crearea functiei pentru determinarea celui al n-lea numar din sirul lui
Fibonacci
    Date de intrare: numarul n
    Date de iesire: numarul din sir corespunzator
}

function functie1(n: integer ):integer;
begin
    if (n <= 1) then functie1 := n
    else functie1 := functie1(n-1) + functie1(n-2);

end;
begin
write('Introduceti numarul n=');
readln(numar);
writeln( 'Rezultatul este: ', functie1(numar));
end.
```

Varianta 2 de rezolvare – *varianta iterativa – implementare C++*

```
#include <iostream>

/* crearea functiei pentru determinarea celui al n-lea numar din sirul lui
Fibonacci
    Date de intrare: numarul n
    Date de iesire: numarul din sir corespunzator
*/

int functie2(int n)
{
    int a = 0, b = 1, c, i;
    if( n == 0)
        return a;
    for (i = 2; i <= n; i++)
    {
        c = a + b;
        a = b;
        b = c;
    }
    return b;
}
```

```
int main()
{
    int n;
    std::cout <<"introduceti numarul n=";
    std::cin >>n;

    std::cout <<"Rezultat= "<<functie2(n)<<std::endl;
    return 0;
}
```

Varianta 2 de rezolvare – *varianta iterativa – implementare Pascal*

```
program problemaliterativ;
var numar:integer;

{ crearea functiei pentru determinarea celui al n-lea numar din sirul lui
Fibonacci
    Date de intrare: numarul n
    Date de iesire: numarul din sir corespunzator
}

function functie2(n: integer ):integer;
var a,b,c,i:integer;
begin
    a:=0;
    b:=1;

    if( n = 0) then functie2 :=a
    else

        for i:= 2 to n do
            begin
                c := a + b;
                a := b;
                b := c;
            end;
        functie2:=b;
    end;

begin
write('Introduceti numarul n=');
readln(numar);
writeln( 'Rezultatul este: ', functie2(numar));
end.
```

## Problema 2

### Enunț:

Sa se determine toate numerele palindroame mai mici decat un numar n dat.

### Exemple:

Pentru n = 12 => se va afisa: 1, 2, 3, 4, 5, 6, 7, 8, 9, 11

Pentru n = 104 => se va afisa: 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 22, 33, 44, 55, 66, 77, 88, 99, 101

Rezolvare – Implementare C++

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

// functie pentru crearea unui palindrom
int crearePalindrom(int input, int b, bool esteImpar)
{
    int n = input;
    int palindrom = input;

    // verificam daca numar cifrelor este par sau impar
    // daca numarul cifrelor este impar, atunci
    // nu tinem cont de ultima cifra de intrare in
    // identificarea inversului

    if (esteImpar)
        n /= b;

    // crearea efectiva a palindromului pe baza formulei
    while (n > 0)
    {
        palindrom = palindrom * b + (n % b);
        n /= b;
    }
    return palindrom;
}
```

```
// functie pentru generarea tuturor palindroamelor si afisarea lor
int generarePalindroame(int n)
{
    int number;

    // de doua ori pentru palindroame de lungime para si impara
    for (int j = 0; j < 2; j++)
    {
        int i = 1;
        while ((number = crearePalindrom(i, 10, j % 2)) < n)
        {
            cout << number << " ";
            i++;
        }
    }
    return 0;
}

int main()
{
    int n;
    std::cout << "introduceti numarul n=";
    std::cin >> n;

    std::cout << generarePalindroame(n) << std::endl;
    return 0;
}
```

Rezolvare – Implementare Pascal

```
{functie pentru a testa daca oglinditul/inversul unui numar este egal cu
numarul}
function test_palindrom(numar:longint):boolean;
begin
    if (numar = oglindit(numar)) then test_palindrom := true
    else test_palindrom := false;
end;

{programul principal al problemei}
begin
write('Introduceti numarul n=');
readln(n);
    for i := 0 to n do
        if (test_palindrom(i)) then
            write(i, ' ');
end.
```

```
program palindrom;
var n,i:integer;
{functie pentru crearea oglinditului/inversului unui numar}
function oglindit(numar_original:longint):longint;
var aux, numar_oglindit: longint;
begin
  numar_oglindit := 0;
  while (numar_original <> 0) do
  begin
    numar_oglindit := numar_oglindit * 10 + numar_original mod 10;
    numar_original := numar_original div 10;
  end;
  oglindit := numar_oglindit;
end;
```

### Problema 3

#### Enunt:

Se da ecuatia de gradul 2 de forma:  $X^2 - s * X + p = 0$ , cu  $s, p$  apartinand lui  $R$  si  $n$  apartinand lui  $N$ .

Sa se calculeze, in mod recursiv suma puterilor radacinilor  $X_1^n + X_2^n$ , fara a se calcula radacinile ecuatiei  $X_1$  si  $X_2$ .

#### Exemple:

Daca avem ecuatia:  $X^2 - 6 * X + 8 = 0$  si  $n=2 \Rightarrow$  rezultatul este 20 ( $x_1=4$  si  $x_2=2$ )

Daca avem ecuatia:  $X^2 - 8 * X + 15 = 0$  si  $n=2 \Rightarrow$  rezultatul este 34 ( $x_1=-3$  si  $x_2=-5$ )

#### Explicatii:

Stiind ca  $X_1$  si  $X_2$  sunt radacinile ecuatiei, rezulta relatiile:

$$X_1^2 - s * X_1 + p = 0$$

$$X_2^2 - s * X_2 + p = 0$$

Daca vom inmulti prima relatie cu  $X_1^n$  si pe cea de a doua cu  $X_2^n$  vom obtine:

$$X_1^{n+2} - s * X_1^{n+1} + p * X_1^n = 0$$

$$X_2^{n+2} - s * X_2^{n+1} + p * X_2^n = 0$$

Daca insumam cele doua relatii, vom obtine:

$$(X_1^{n+2} + X_2^{n+2}) - s (X_1^{n+1} + X_2^{n+1}) + p (X_1^n + X_2^n) = 0$$

Ceea ce am putea scrie in mod echivalent:  $S_{n+2} - s * S_{n+1} + p * S_n = 0$  rezulta deci:  $S_{n+2} = s * S_{n+1} - p * S_n$

Vom obtine de aici urmatoarea relatie de recurenta pentru suma puyerilor radacinilor unei ecuatiei de gradul II:

$$Sum(n) = \begin{cases} 2, & \text{daca } n = 0 \\ s, & \text{daca } n = 1 \\ s * Sum(n-1) - p * Sum(n-2), & \text{daca } n > 1 \end{cases}$$

Rezolvare – implementare C++

```
#include <iostream>
float s,p;

/*crearea functiei pentru calculul sumei conform formulei de recurenta
   Date de intrare: numarul n
   Date de iesire: rezultatul sumei conform formulei de recurenta
*/

float Sum (int n)
{
    if (!n) return 2;
    if (n==1) return s;
    return s*Sum(n-1)-p*Sum(n-2);
}
int main()
{
    int n;
    std::cout <<"introduceti cei doi coeficienti: ";
    std::cin >>s>>p;
    std::cout << "n=";
    std::cin >>n;
    std::cout <<"Rezultatul este: "<<Sum(n)<<std::endl;
    return 0;
}
```

Rezolvare – implementare Pascal

```
program problema3;
var numar,s,p:integer;

{crearea functiei pentru calculul sumei conform formulei de recurenta
   Date de intrare: numarul n
   Date de iesire: rezultatul sumei conform formulei de recurenta
}
function suma(n: integer ):integer;
begin
    if n<=0 then suma:=2
    else
        if n=1 then suma:=s
        else suma:=s*suma(n-1)-p*suma(n-2);
end;
```

## begin

```
write('Introduceti numarul n=');  
readln(numar);  
  
write('Introduceti coeficientul s=');  
readln(s);  
  
write('Introduceti coeficientul p=');  
readln(p);  
  
writeln('Rezultatul este: ', suma(numar));  
end.
```

## Problema 4

### Enunt:

Fie  $M$  o multime nevida si  $*$  :  $M \times M \rightarrow M$  o operatie binara asociativa definita pe multimea  $M$ .

Sa se scrie o functie care sa determine numarul de modalitati in care se poate evalua expresia  $x_1 * x_2 * \dots * x_n$ , unde  $x_1, x_2, \dots, x_n$  apartin multimii  $M$  si  $n > 0$ .

### Exemple:

Pentru  $n=3$ , numarul de modalitati in care se poate evalua expresia  $x_1 * x_2 * x_3$  este 2:  $x_1 * (x_2 * x_3)$  si  $(x_1 * x_2) * x_3$

Pentru  $n=4$ , numarul de modalitati de a evalua expresia este 5:  $((x_1 * x_2) * x_3) * x_4$  sau  $(x_1 * x_2) * (x_3 * x_4)$  sau  $(x_1 * (x_2 * x_3)) * x_4$  sau  $x_1 * ((x_2 * x_3) * x_4)$  sau  $x_1 * (x_2 * (x_3 * x_4))$

### Explicatii:

In primul rand, pentru  $n=1$  sau  $n=2$  nu avem decat o singura posibilitate (cazurile elementare ale problemei).

Daca  $n > 2$ , pentru a evalua produsul  $x_1 * x_2 * \dots * x_n$  trebuie sa aplicam asociativitatea si sa impartim termenii in grupandu-i in paranteze  $(x_1 * x_2 * \dots * x_k) * (x_{k+1}, x_{k+2}, \dots, x_n)$ .

Vom urma apoi evaluarea parantezelor, urmand sa efectuam operatia  $*$  intre rezultatele obtinute in urma evaluarii parantezelor.

Notam cu  $P(n)$  numarul de posibilitati de a evalua o expresie cu  $n$  termeni.

Daca impartim termenii la pozitia  $k$  ( $1 \leq k < n$ ) atunci deducem ca:  $P(n) = P(k) \times P(n-k)$  (pentru orice modalitate de evaluare a primei grupe de  $k$  termeni se poate asocia o modalitate de evaluare a celei de a doua grupe de  $n-k$  termeni).

Impartirea se poate face in orice pozitie  $k$ , incepand cu pozitia 1 pana la pozitia  $n-1$ , se poate deduce urmatoarea formula recursiva pentru calculul  $P(n)$ .



$$P(n) = \begin{cases} 1, & \text{daca } n \leq 2 \\ \sum_{k=1}^{n-1} P(k) * P(n-k), & \text{daca } n > 2 \end{cases}$$

Rezolvare – implementare C++

```
#include <iostream>
using namespace std;
/* functie pentru efectuarea calcului conform formulei de recurenta de mai
sus
Date de intrare: numarul n
Date de iesire: numarul total de posibilitati
*/

unsigned long calcul (int n)
{
    long s=0;
    int k;
    if (n<=2) return 1;
    for (k=1; k<n; k++)
        s+= calcul(k) * calcul(n-k);
return s;
}

int main()
{
    int n;
    std::cout <<"introduceti numarul n=";
    std::cin >>n;

    std::cout<<"Rezultatul este:"<< calcul(n)<<std::endl;
    return 0;
}
```

Rezolvare - implementarea in Pascal:

```
program problema4;
var numar:integer;
{functie pentru efectuarea calcului conform formulei de recurenta de mai
sus
Date de intrare: numarul n
Date de iesire: numarul total de posibilitati
}
```

```
function calcul (n:integer):integer;
var s,k:integer;
begin
  s:=0;
  if n<=2 then calcul:=1
  else
    for k:=1 to n-1 do
      calcul := (s + calcul(k) * calcul(n-k));
    end;

begin
write('Introduceti numarul n=');
readln(numar);
writeln('Rezultatul este: ', calcul(numar));
end.
```

## Problema 5

### Enunț

Semafoarele din trei intersecții diferite din centrul orașului își schimbă culoarea după fiecare 48, 72 și 108 secunde. Dacă cele trei semafoare își schimbă simultan culoarea la ora 8 dimineața, la ce altă oră își vor mai schimba simultan culoarea?

### Analiză

- cel mai mic multiplu comun

### Rezolvare C++

```
/*
Problema a fost compilata cu Microsoft Visual Studio varianta Community 2017
*/
#include <iostream>

/*
Descriere:   calculeaza cel mai mic multiplu comun a doua numere naturale
Date:       a, b - numere naturale
Rezultat:   cel mai mic multiplu comun al celor doua numere
*/
```

```
int cmmmc(int a, int b)
{
    int r;
    if (a < b)
    {
        r = a;
        a = b;
        b = r;
    }
    r = a;
    while ((r%b) != 0)
    {
        r += a;
    }
    return r;
}

int main()
{
    int a = 48;
    int b = 72;
    int c = 108;
    std::cout << "numarul de secunde dupa ora 8 la care semafoarele se vor sincroniza:
" << cmmmc(cmmmc(a, b), c) << std::endl;
    return 0;
}
```

### Rezolvare problema – implementare Pascal

```
program semafoare;
var a,b,c:integer;

{functie care calculeaza cel mai mic multiplu comun al doua numere;
 date de intrare: a, b
 date de iesire: cmmmc (a,b)
}

function cmmmc (a:integer;b:integer):integer;
var x,y,r:integer;
begin
x:=a;
y:=b;
repeat
r:=a mod b;
a:=b;
b:=r
until r=0;
cmmmc:=(x*y) div a;
end;
```

```
begin
write('Introduceti numarul a=');
readln(a);
write('Introduceti numarul b=');
readln(b);
write('Introduceti numarul c=');
readln(c);

writeln( 'Rezultatul este: ', cmmmc (a,cmmmc(b,c)));
end.
```

## Problema 6

### Enunț

Alin are în față  $M$  mingi de tenis roșii și  $N$  mingi de tenis galbene. El dorește să formeze mai multe grupuri, astfel încât toate grupurile să conțină același număr de mingi, toate mingile dintr-un grup să aibă aceeași culoare și numărul de mingi dintr-un grup să fie cât mai mare posibil. După ce a reușit să facă împărțea, Alin a primit alt set de mingi roșii și galbene. Imediat a început să le împartă în grămezi de  $M$  mingi roșii și  $N$  mingi galbene. Fiecare grămadă conține mingi de aceeași culoare. El dorește să aleagă un număr de  $X$  grămezi cu mingi roșii și  $Y$  grămezi cu mingi galbene astfel încât diferența dintre numărul total de mingi roșii și numărul total de mingi galbene să fie egală cu numărul de mingi dintr-unul din grupurile construite la început.

Să se citească de la tastatură numerele  $M$  și  $N$ . Se vor afișa pe ecran numărul de mingi dintr-o grupă construită, numărul de grămezi de mingi roșii și galbene.

### Exemple

| Date de intrare | Rezultat |
|-----------------|----------|
| 4 6             | 2 1 1    |
| 10 5            | 5 0 1    |
| 24 63           | 3 8 3    |

### Analiză

- trebuie determinat cel mai mare divizor comun a două numere  $M$  și  $N$
- trebuie să se determine două numere  $X$  și  $Y$  astfel încât  $|MX - NY| = \text{cmmdc}(M,N)$
- algoritmul extins al lui Euclid

## Rezolvare C++

```
/*
Problema a fost compilata cu Microsoft Visual Studio varianta Community 2017
*/
#include <iostream>

/*
Descriere:   algoritmul lui Euclid in varianta extinsa
Date:       a, b - numere naturale, x, y
Rezultat:   cmmdc pentru cele doua numere naturale
*/
int cmmdc_extins(int a, int b, int& x, int& y)
{
    x = 0;
    y = 1;
    int u = 1;
    int v = 0;
    int q, r, m, n;
    while (a != 0)
    {
        q = b / a;
        r = b % a;
        m = x + u * q;
        n = y + v * q;
        b = a;
        a = r;
        x = u;
        y = v;
        u = m;
        v = n;
    }
    return b;
}

int main()
{
    int x = 0, y = 0, cmmdc = 0;
    cmmdc = cmmdc_extins(4, 6, x, y);
    std::cout << "cmmdc extins: " << cmmdc << ", x = " << x << ", y = " << y <<
std::endl;
    cmmdc = cmmdc_extins(10, 5, x, y);
    std::cout << "cmmdc : " << cmmdc << ", x = " << x << ", y = " << y << std::endl;
    cmmdc = cmmdc_extins(24, 63, x, y);
    std::cout << "cmmdc : " << cmmdc << ", x = " << x << ", y = " << y << std::endl;
    return 0;
}
```

## Rezolvare Pascal

```
{  
Descriere:   algoritmul lui Euclid in varianta extinsa  
Date:       a, b - numere naturale  
Rezultat:   cmmdc pentru cele doua numere naturale  
}
```

```
program cmmdc;  
function cmmdc_extins(a:integer; b:integer):integer;  
var x,y,u,v,q,r,m,n:integer;  
begin  
    x := 0;  
    y := 1;  
    u := 1;  
    v := 0;  
    while (a <> 0) do  
        begin  
            q := b div a;  
            r := b mod a;  
            m := x + u * q;  
            n := y + v * q;  
            b := a;  
            a := r;  
            x := u;  
            y := v;  
            u := m;  
            v := n;  
        end;  
    writeln('x = ',x,' , y = ', y);  
    cmmdc_extins:=b;  
end;  
begin  
  
    writeln( 'cmmdc extins: ',cmmdc_extins(4, 6));  
    writeln( 'cmmdc extins: ',cmmdc_extins(10, 5));  
    writeln( 'cmmdc extins: ',cmmdc_extins(24, 63));  
end.
```

## Problema 7

### Enunț

O companie nou înființată de taxi are următorul model de taxare pentru șoferii săi: un șofer începe săptămâna cu 0 unități monetare, în timpul săptămânii fiecare șofer trebuie să predea companiei tot câștigul realizat în momentul în care are asupra sa N unități monetare, orice câștig suplimentar rămâne în posesia șoferului. Compania are angajați K șoferi, pentru fiecare se cunoaște numărul curselor efectuate și câștigul obținut. În nici un moment un șofer nu poate avea asupra sa mai mult de N-1 unități monetare (decât imediat după o cursă) deoarece imediat ce are mai mult de N unități monetare trebuie să le predea companiei. Mai mult, dacă după ce predă N unități monetare, numărul său de unități monetare este din nou mai mare decât N, el va preda din nou N unități monetare până când va avea un câștig mai mic decât N. Determinați câștigul de la sfârșitul săptămânii pentru un șofer.

Să se citească de la tastatură valoarea N, numărul curselor pentru un șofer și câștigul pentru fiecare cursă. Programul va afișa câștigul obținut de șofer la sfârșitul săptămânii.

### Exemple

| Date de intrare     | Rezultat |
|---------------------|----------|
| 50 4 1 2 3 4        | 10       |
| 50 5 10 20 30 40 50 | 0        |
| 50 3 874 9735 835   | 44       |

### Analiză

- aritmetică modulară (însușiri modulo)
- pe baza modului de efectuare a operațiilor în aritmetica modular, următoarea relație se respect:
 
$$(a + b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$$

### Rezolvare C++

```

/*
Problema a fost compilata cu Microsoft Visual Studio varianta Community 2017
*/
#include <iostream>

```

```
/*
Descriere:   calculeaza suma modulo n pentru doua numere naturale
Date:       a, b, n - numere naturale,
Rezultat:   (a + b) modulo n
*/
int adunare_modulo(int a, int b, int n)
{
    return (a + b) % n;
}

/*
Descriere:   detrmna prin sume modulo castigul unui sofer
Date:       n - numar natural
Rezultat:   castigul unui sofer la sfarsitul saptamanii
*/
int determinaCastigSofer(int n)
{
    int curse;
    int castig = 0;
    int castig_cursa;
    std::cin >> curse;
    while (curse)
    {
        std::cin >> castig_cursa;
        castig = adunare_modulo(castig, castig_cursa, n);
        curse--;
    }
    return castig;
}

int main()
{
    int n = 1;

    while (n)
    {
        std::cin >> n;
        std::cout << "castigul soferului este: " << determinaCastigSofer(n) <<
std::endl;
    }

    return 0;
}
```



## Rezolvare Pascal

```
program castig;
var n:integer;
{
Descriere:    calculeaza suma modulo n pentru doua numere naturale
Date:        a, b, n - numere naturale,
Rezultat:    (a + b) modulo n
}
function adunare_modulo(a:integer; b:integer; n:integer):integer;
begin
    adunare_modulo:= (a + b) mod n;
end;

{
Descriere:    detrmina prin sume modulo castigul unui sofer
Date:        n - numar natural
Rezultat:    castigul unui sofer la sfarsitul saptamanii
}
function determinaCastigSofer(n:integer):integer;
var curse, castig, castig_cursa:integer;
begin
    castig := 0;
    readln(curse);
    while (curse<>0) do
    begin
        readln(castig_cursa);
        castig := adunare_modulo(castig, castig_cursa, n);
        dec(curse);
    end;
    determinaCastigSofer:=castig;
end;
begin
    n := 1;

    while (n<>0) do
    begin
        readln(n);
        if (n<>0) then writeln('castigul soferului este: ', determinaCastigSofer(n));
    end;
end.
```

## Problema 8

### Enunț

Fie un număr  $n$ , se definește gradul numărului  $n$  ca fiind numărul de secvențe "101" din reprezentarea binară a acestuia. De exemplu, numărul 21 are gradul 2 (reprezentarea binară a lui 21 este 10101). Să se citească de la tastatură gradul  $k$  și mai multe numere (citirea se termină cu numărul 0), să se afișeze dacă un număr are gradul  $k$ .

### Exemple

| Date de intrare          | Rezultat                               |
|--------------------------|--|
| 1 7 5 13 21 29 25 0      | numere de grad cel puțin 1: 5 13 21 29 |
| 2 13 5 21 56 45 360 37 0 | numere de grad cel puțin 2: 21 45 360  |

### Analiză

- operații la nivel de bit, izolarea celor mai puțin semnificativ trei biți din reprezentarea binară și verificarea dacă sunt o secvență 101, după care se deplasează la stânga cu una sau două poziții (în funcție de rezultatul comparației cu  $(101)_2 = (5)_{10}$ ).

### Rezolvare C++

```

/*
Problema a fost compilata cu Microsoft Visual Studio varianta Community 2017
*/
#include <iostream>
using namespace std;

const int NRBITI = 32;
int BINAR[NRBITI] = { 0 };

/*
Descriere:   toate elementele vectorului BINAR = 0
*/
void resetBinar()
{
    for (int i = 0; i < NRBITI; ++i)
    {
        BINAR[i] = 0;
    }
}

```

```
/*
 * Descriere:   reprezint un numar in baza doi
 * Date:       n - numarul natural de reprezentat in baza doi
 * Rezultat:   numarul de cifre din reprezentarea binara
 */
int calculReprezBinar(int n)
{
    resetBinar();
    int i;
    for (i = 0; n > 0; i++)
    {
        BINAR[i] = n % 2;
        n = n / 2;
    }
    return i + 1;
}

/*
Descriere:   reprezint un numar in baza doi, versiunea II folosind
operatii pe biti
Date:       n - numarul natural de reprezentat in baza doi
Rezultat:   numarul de cifre din reprezentarea binara
*/
int calculReprezBinar2(int n)
{
    resetBinar();
    int i = 0;
    /*
    se izoleaza fiecare cifra din reprezentarea binara a numarului
    si se stocheaza in vectorul BINAR pe pozitia ei
    */
    while (n)
    {
        BINAR[i] = n & 1;
        n >>= 1;
        i++;
    }
    return i;
}
```

```

/*
Descriere:   calcul determina gradul unui numar (numarul de secvente 101
             din reprezentarea binara)
Date:       n - numarul natural pentru care trebuie determinat gradul
Rezultat:   gradul numarului
*/
int determinaGrad(int n)
{
    int grad = 0;
    int nrCifre = 0;
    nrCifre = calculReprezBinar2(n);
    for (int i = nrCifre; i >= 2; i--)
    {
        if (BINAR[i] == 1 && BINAR[i - 1] == 0 && BINAR[i - 2] == 1)
            grad++;
    }

    return grad;
}

int main()
{
    int k, numar = 1;
    // se citeste gradul cautat
    cin >> k;
    /* sirul s va stoca numerele de grad cel puțin k */
    int s[100] = { 0 };
    int i = 0;
    /* citirea numerelor se termina cu citirea lui 0 */
    while (numar)
    {
        cin >> numar;
        if (k <= determinaGrad(numar))
        {
            s[i] = numar;
            i++;
        }
    }
    cout << "numere de grad cel puțin " << k << " :" << endl;
    for (int y = 0; y < i; ++y)
    {
        cout << s[y] << " ";
    }
    cout << endl;
    return 0;
}

```

O abordare mai eficientă poate fi imaginată prin izolarea celor mai puțin semnificativ trei biți din reprezentarea binară și verificarea dacă sunt o secvență 101, după care se deplasează la stânga cu una sau două poziții (în funcție de rezultatul comparației cu  $(101)_2 = (5)_{10}$ ).

```

/*
Problema a fost compilata cu Microsoft Visual Studio varianta Community 2017
*/
#include <iostream>
using namespace std;

/*
Descriere:   calcul determina gradul unui numar (numarul de secvente 101
             din reprezentarea binara)
Date:       n - numarul natural pentru care trebuie determinat gradul
Rezultat:   gradul numarului
*/
int determinaGrad2(int n)
{
    int grad = 0;
    while (n)
    {
        /*
        se izoleaza cei mai putin semnificativ 3 biti si se verifica
        daca sunt o secventa 101
        */
        if ((n & 7) == 5)
        {
            grad++;
            n >>= 2;
        }
        else
        {
            n >>= 1;
        }
    }
    return grad;
}

int main()
{
    int k, numar = 1;
    // se citeste gradul cautat
    cin >> k;
    /* citirea numerelor se termina cu citirea lui 0 */
    while (numar)
    {
        cin >> numar;
        if (k == determinaGrad2(numar))
        {
            cout << "numarul " << numar << " are gradul " << k << endl;
        }
    }
    return 0;
}

```

Exemple:

|                             |
|-----------------------------|
| Date de intrare si rezultat |
| 2                           |

21

numarul 21 are gradul 2

7

5

29

45

numarul 45 are gradul 2

360

numarul 360 are gradul 2

0

3

362

numarul 362 are gradul 3

360

85

numarul 85 are gradul 3

0

## Rezolvare Pascal

```
program HelloWorld;

const NRBITI = 32;
var  BINAR:array [1..32] of integer;
var  k, numar,i,y:integer;
var  s:array [1..100] of integer;

procedure resetBinar();
var i:integer;
begin
for i:=1 to NRBITI do
begin
BINAR[i] := 0;
end;
end;

function calculReprezBinar(n: integer ):integer;
var i:integer;
begin
resetBinar();
i := 1;
while (n>0) do
begin
BINAR[i] := n mod 2;
n := n div 2;
end;
calculReprezBinar:=i+1;
end;
```

```
function calculReprezBinar2(n: integer):integer;
var i:integer;
begin
  resetBinar();
  i:= 1;
  while (n>0) do
  begin
    BINAR[i]:=n and 1;
    n:=n>>1;
    inc(i);
  end;
  calculReprezBinar2:=i;
end;

function determinaGrad2(n: integer ):integer;
var grad:integer;
begin
  grad:= 0;
  while (n>0) do
  begin
    if ((n and 7)=5) then
    begin
      inc(grad);
      n:= n>>2;
    end
    else
      n:= n>>1;
    end;
  determinaGrad2:=grad;
end;

function determinaGrad(n: integer ):integer;
var grad, nrCifre,i:integer;
begin
  grad:= 0;
  nrCifre:= 0;
  nrCifre := calculReprezBinar2(n);
  for i := nrCifre downto 3 do
    if ((BINAR[i]=1) and (BINAR[i-1]=0) and (BINAR[i-2]=1)) then
      inc(grad);

  determinaGrad:= grad;
end;

begin
  numar := 1;
  // se citeste gradul cautat
  readln(k);
  // varainta 1 fara siruri
  // se citesc elementele sirului, citirea se termina la introducerea numarului 0
  while(numar>0)do
  begin
    readln(numar);
    if (k = determinaGrad(numar)) then
    begin
      writeln('numarul ',numar,' are gradul= ',k);
    end;
  end;
```

```
end;
// varianta 2 cu siruri
i := 1;
numar:=1;
while (numar>0) do
begin
  readln(numar);
  if (k <= determinaGrad2(numar)) then
  begin
    s[i] := numar;
    inc(i);
  end;
end;
writeln( 'numere de grad ', k, ' :');
for y := 1 to i-1 do
  write(s[y], ' ');
end.
```

## Problema 9

### Enunț

Un număr  $n$  este "zâmbăreț" dacă duce la 1 după o secvență de pași unde în fiecare pas numărul este înlocuit cu suma pătratelor cifrelor ce formează numărul. Sa se scrie un program care citește mai multe numere până la citirea numărului 0 și determină câte numere "zâmbărețe" s-au citit.

### Exemple

numarul 19 este "zâmbăreț"

pas\_1:  $1+9^2=82$

pas\_2:  $64+4=68$

pas\_3:  $36+64=100$

pas\_4:  $1+0+0=1$

### Rezolvare C++

```
/*
Problema a fost compilata cu Microsoft Visual Studio varianta Community 2017
*/
#include <iostream>
using namespace std;
```



```
/*
Descriere:   calcul suma patrate cifre numar
Date:       n - numar natural
Rezultat:   suma patrate numere
*/
int sumPatrate(int n)
{
    int suma = 0;
    while (n)
    {
        suma += (n % 10) * (n % 10);
        n /= 10;
    }
    return suma;
}

/*
Descriere:   determin daca un numar este zambaret
Date:       n - numar natural
Rezultat:   True - numarul e zambaret
            False - numarul nu este zambaret
*/
bool eNumarZambaret(int n)
{
    int nr1, nr2;
    // initializare numere cu n
    nr1 = nr2 = n;
    /*
    un numar nu este zambaret daca pe parcursul iteratiilor
    atinge aceeasi valoare succesiv
    */
    do
    {
        nr1 = sumPatrate(nr1);
        nr2 = sumPatrate(sumPatrate(nr2));
    } while (nr1 != nr2);

    // daca ambele numere sunt 1, return true
    return (nr1 == 1);
}
```

```
int main()
{
    int contor = 0;
    int numar = 1;
    while (numar)
    {
        cin >> numar;
        if (eNumarZambaret(numar))
        {
            cout << "numarul " << numar << " este zambaret" << endl;
            contor++;
        }
    }
    cout << "Am citit " << contor << " numere zambarete" << endl;
    return 0;
}
```

## Rezolvare Pascal

```
function sumPatrate(n:integer):integer;
var suma:integer;
begin
    suma := 0;
    while (n>0) do
    begin
        suma := suma+ (n mod 10) * (n mod 10);
        n :=n div 10;
    end;
    sumPatrate :=suma;
end;
function eNumarZambaret(n:integer):boolean;
var nr1, nr2:integer;
begin
    // initializare numere cu n
    nr1 := n;
    nr2 := n;

    // un numar nu este zambaret daca pe parcursul iteratiilor
    // atinge aceeasi valoare succesiv

    repeat
        nr1 := sumPatrate(nr1);
        nr2 := sumPatrate(sumPatrate(nr2));
    until (nr1 = nr2);

    // daca ambele numere sunt 1, return true
    eNumarZambaret:= (nr1 = 1);
end;
var contor, numar:integer;
begin
    contor := 0;
    numar := 1;
    while (numar>0) do
    begin
```

```

readln(numar);
if (eNumarZambaret(numar)) then
begin
    writeln(numar, ' este zambaret');
    inc(contor);
end;
end;
writeln( 'Am citit ',contor,' numere zambarete');
end.

```

Exemple:

| Date de intrare | Rezultat                    |
|-----------------|-----------------------------|
| 49              | 23 este zambaret            |
| 19              | 49 este zambaret            |
| 100             | 19 este zambaret            |
| 11              | 100 este zambaret           |
| 0               | Am citit 4 numere zambarete |

## Probleme tip grilă

1. Ce face secvența:

```

int f(char* str)
{
    if (*str == '\0')
        return 0;
    else
        return 1 + f(str + 1);
}

```

Raspuns: calculeaza lungimea unui șir de caractere

2. Ce face secvența:

```

int f(int n) {
    if (n == 0)
        return 0;
    else
        return (n % 2 + 10 * f(n / 2));
}

```

Raspuns: transformă un numar în baza doi