Probleme consultații – 28 ianuarie 2023
Algoritmi care lucrează pe numere
(fără tablouri sau alte elemente structurate)

Problema 1

i) Alegeti care dintre urmatoarele secvente de cod elimina in mod corect ultima cifra a unui numar n (0<=10<=35535)

A
```
#include <stdio.h>
int removeLastDigit(int num)
{
    if (num < 10)
    {
        return 0;
    }
    return (num/10);
}

int main() {
    int numar;
    printf("Introduceti n= ");
    scanf("%d", &numar);
    printf("\n Number without last digit = %d",
removeLastDigit(numar));
    return 0;
}
```

B
```
#include <stdio.h>
int removeLastDigit(int num)
{
    if (num < 10)
    {
        return 0;
    }
    return (num%10);
}

int main() {
    int numar;
    printf("Introduceti n= ");
    scanf("%d", &numar);
    printf("\n Number without last digit = %d",
removeLastDigit(numar));
    return 0;
}
```

C
```
#include <stdio.h>
int removeLastDigit(int num)
{
    if (num < 10)
    {
        return 0;
    }
    return (num/100);
}

int main() {
    int numar;
    printf("Introduceti n= ");
    scanf("%d", &numar);
    printf("\n Number without last digit = %d",
removeLastDigit(numar));
    return 0;
}
```

D
```
#include <stdio.h>
int removeLastDigit(int num)
{
    if (num < 10)
    {
        return 0;
    }
    return (num%100);
}

int main() {
    int numar;
    printf("Introduceti n= ");
    scanf("%d", &numar);
    printf("\n Number without last digit = %d",
removeLastDigit(numar));
    return 0;
}
```

ii) Care este efectul fiecarei secvente de cod asupra numarului n = 12345?

A => 1234

B => 5

C => 123

D => 45

iii) Urmatoarea secventa de cod elimina anumite valori din numarul n. Care sunt valorile eliminate?

```
#include <stdio.h>
int removeXDigits(int num) {
    if (num == 0) {
        return 0;
    }
    int digit = num % 10;
    if (digit % 2 == 1) {
        return removeXDigits(num / 10);
    }
    return digit + 10 * removeXDigits(num / 10);
}



int main() {
    int numar;
    printf("Introduceti n= ");
    scanf("%d", &numar);
    printf("\n Result = %d", removeXDigits(numar));
    return 0;
}
```

12345 => 1 3 5

iv) Care este linia in care trebuie sa modificati pentru a elimina cifrele pare?

v) Care este efectul urmatoarei secvente de cod asupra numarului 98765?

```
int YDigits(int num) {
    if (num == 0) {
        return 0;
    }
    return (num % 10 + YDigits(num / 10));
}
```

Daca se apeleaza: printf("\n Result = %d", YDigits(numar)), unde numar=98765

a. 100 b. -1 c. 0 d.100 e. 30 **f. 35** g. 40

Problema 2

Se da un număr natural n și expresia $s = 1 + 2^2 + 3^3 + \ldots + n^n$ $(0 < n < 100)$
Daca se citește numărul n, alegeti care dintre variantele de cod de mai jos vor determina **doar afișarea a ultimei cifre a rezultatului expresiei s**

Exemplu

Date de intrare	Date de ieşire	*Explicaţie*
n=3	2	Suma este 32 şi ultima cifră 2.

// v1	// v2	// v3	// v4

i,n,us,up,j : integer;
citire n
 	
us <-- 0;
for i <-- 1 **to** n **do**
begin
up <-- 1;
for j <-- 1 **to** i **do**
up <-- up*i **mod** 10;
us <-- (us+up) **mod** 10;
end;
 	
afisare us
	i,n,us,up,j : integer;
	citire n
	
	us <-- 0;
	for i <-- 2 **to** n **do**
	begin
	up <-- 2;
	for j <-- 1 **to** i **do**
	up <-- up*i **mod** 10;
	us <-- (us+up) **mod** 10;
	end;
	
	afisare us
		i,n,us,up,j : integer;
		citire n
		
		us <-- 0;
		for i <-- 1 **to** n **do**
		begin
		up <-- 1;
		for j <-- 1 **to** i **do**
		up <-- up **mod** 10;
		us <-- (us+up) **mod** 10;
		end;
		
		afisare us
			i,n,us,up,j : integer;
			citire n
			
			us <-- 0;
			for i <-- 1 **to** n **do**
			begin
			up <-- 1;
			for j <-- 1 **to** i **do**
			up <-- up*i **mod** 10;
			us <-- up **mod** 10;
			end;
			
			afisare us

Problema 3

Se dau secventa de cod și afirmatiile de mai jos. Analizati afirmatiile date si alegeti afirmatia corecta referitoare la codul analizat.

a. codul determina elementul maxim dintre valorile introduse
b. codul determina perechea cu valoare maxima dintre cele n perechi introduse
c. codul determina perechea cu valoare minima dintre cele n perechi introduse
d. codul determina intersectia multimilor data de cele n perechi introduse
e. codul determina reuniunea multimilor data de cele n perechi introduse

```
#include <iostream>
using namespace std;
int main ()
{
 int n, a, b, max, min, i;
 cout << "n =";
 cin >> n;
 cout << "perechea 1:";      3 perechi 1 9  1 5  1 4  => 1 2 3 4
 cin >> a >> b;
 max = a;
 min = b;
 for (i = 2; i <= n; i++)
  {
    cout << "perechea " << i << ":";
    cin >> a;
    cin >> b;
    if (max <= a)
   max = a;
    if (min >= b)
   min = b;
   }
 if (max <= min)
```

```
  {
    cout << "rezultatul este:" << endl;
    for (i = max; i <= min; i++)
   cout << i << " ";
   }
 else
   cout << "rezultatul corect nu se poate determina";
 return 0;
}
```

Problema 4

Se citeste de la tastatura un număr n (0 <= n <= 2147483647). Sa se afiseze numărul de biti setați din reprezentarea binara a numarului n.

Ex. n = 27 => 4
n = 10 => 2
n= 2147483647 => 31

Solutii:

Varianta nerecursivă (cu impartiri la 2) – v1

```
#include <stdio.h>
int numar_biti_nerecursiv(int numar) {
   int numar_biti = 0;
   while (numar != 0) {
      if (numar % 2 == 1) numar_biti++;
      numar = numar / 2;
   }
   return numar_biti;
}

int main() {
   int numar;
   printf("Introduceti n= ");
   scanf("%d", &numar);
   printf("\n Nerecursiv = %d", numar_biti_nerecursiv(numar));
   return 0;
}
```

Varianta recursivă (cu impartiri la 2) – v2

```
#include <stdio.h>


int numar_biti_recursiv(int numar) {
   if (numar == 0) return 0;
   if (numar % 2 == 1) return 1 + numar_biti_recursiv(numar /2);
   return numar_biti_recursiv(numar / 2);
}
int main() {
   int numar;
   printf("Introduceti n= ");
```

```
    scanf("%d", &numar);
    printf("\n Recursiv = %d", numar_biti_recursiv(numar));
    return 0;
}
```

Varianta nerecursivă cu shiftare pe biti (v3)

```
#include <stdio.h>
int numar_biti_nerecursiv(int numar) {
    int numar_biti = 0;
    while (numar != 0) {
        if (numar & 1 == 1) numar_biti++;
        numar = numar >> 1;
    }
    return numar_biti;
}
int main() {
    int numar;
    scanf("%d", &numar);
    printf("%d", numar_biti_nerecursiv(numar));
    return 0;
}
```

Varianta recursivă cu shiftare pe biti (v4)

```
#include <stdio.h>
int numar_biti_recursiv(int numar) {
    if (numar == 0) return 0;
    if (numar & 1 == 1) return 1 + numar_biti_recursiv(numar >> 1);
    return numar_biti_recursiv(numar >> 1);
}
int main() {
    int numar;
    scanf("%d", &numar);
    printf("Rec: %d", numar_biti_recursiv(numar));
    return 0;
}
```

Problema 5

Se da un numar n (0 <= n <= 2147483647)
Sa se scrie o secventă de cod recursiva care determină valoarea numarului citit in baza 2.

```
#include <stdio.h>
int conversiebaza2(int n) {
if (n == 0)return 0;
else return (n % 2 + 10 * conversiebaza2(n / 2));
}
int main() {
```

```
    int numar;
    printf("Introduceti n= ");
    scanf("%d", &numar);
    printf("\n Rez = %d", conversiebaza2(numar));
    return 0;
}
```

Problema 6

Analizati secventa de cod de mai jos (pentru numărul n restrictiile sunt: -32,768 <= n <= 32,767). Identificati afirmatia/afirmatiile corecte de mai jos.

```
#include <stdio.h>
int main (void)
{
  int i = 0;
  int j = 0;
  int b[16] = { 0 };

  printf ("Introduceti un nr in baza 10: ");
  scanf ("%d", &i);
  for (j = 15; j >= 0; j--)
    {
      b[j] = i & 0x1; // 0x1 = 1 in baza 16
      i = i >> 1;
    }
  printf ("Rezultatul este: ");
  for (j = 0; j <= 15; j++)
    printf ("%d", b[j]);

  printf ("\n");
  return 0;
}
```

a. Codul determina suma cifrelor numarului citit.

b. Codul determina valoarea in binar a unui nr natural citit.

c. Codul transforma un numar pozitiv citit in numar negativ.

d. Codul transforma un numar negativ citit in numar pozitiv.

Problema 7

Se da un numar n in baza 10 (0 <= n <= 65535). Sa se determine valoarea numarului n in baza 16.

Exemplu:

n = 7 => 7;

n=10 => A;

n= 3146 => C4A

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

// function to convert decimal to hexadecimal
void decToHexa (int n)
{
   char hexaDeciNum[100]; // char array to store hexadecimal number
   int i = 0;   // counter for hexadecimal number array
  while (n != 0)
   {
     int temp = 0;      // temporary variable to store remainder
     temp = n % 16;    // storing remainder in temp variable

     // check if temp < 10
     if (temp < 10)
   {
     hexaDeciNum[i] = temp + 48;
     i++;
   }
     else
   {
     hexaDeciNum[i] = temp + 55;
     i++;
   }
     n = n / 16;
   }

  // printing hexadecimal number array in reverse order
  for (int j = i - 1; j >= 0; j--)
    cout << hexaDeciNum[j];
}

int
main ()
{
  int n;
  cout << "n =";
  cin >> n;
  decToHexa (n);
  return 0;
}
```

Problema 8

Scrieti o secvență de cod care realizeaza conversia unui număr din baza 2 sau baza 16 in baza 10.
De exemplu:

n = 1010 (in binar) => 5 in baza 10
n = 2AB (in hexazecimal) => 683 in baza 10
n = 11111100 (in binar) => 252 in baza 10

```
#include <stdio.h>
#include <string.h>

// function to return the value of a char.
// For example, 2 is returned for '2'.
// 10 is returned for 'A', 11 for 'B', 12 for 'C', 13 for 'D', 14 for 'E', // 15 for 'F'
int val (char c)
{
  if (c >= '0' && c <= '9')
    return (int) c - '0';
  else
    return (int) c - 'A' + 10;
}

// Function to convert a number from given base 'b' to decimal
int toDecimal (char *str, int base)
{
  int len = strlen (str);
  int power = 1;       // Initialize power of base
  int num = 0;         // Initialize result
  int i;

  // Decimal equivalent is str[len-1]*1 + str[len-2]base +
  // + str[len-3](base^2) + ...
  for (i = len - 1; i >= 0; i--)
    {
      // A digit in input number must be less than number's base
      if (val (str[i]) >= base)
    {
      printf ("Invalid Number");
      return -1;
    }

      num = num + val (str[i]) * power;
      power = power * base;
    }

  return num;
}

int main ()
{
```

```
  int base;
  char str[8];
  printf ("Introduceti baza: ");
  scanf ("%d", &base);
  printf ("Introduceti numarul de converit: ");
  scanf ("%s", str);

  printf ("Decimal equivalent of %s in base %d is "
     " %d\n", str, base, toDecimal (str, base));
  return 0;
}
```

Problema 9

Pentru a se asigura o transmitere cat mai exactă a informaţiilor pe reţea, transmiterea se efectuează caracter cu caracter, fiecare caracter fiind dat prin codul său ASCII (ascii code table: https://www.asciitable.com/), adică un grup de 8 biţi (un byte sau un octet).
Pentru fiecare 8 biţi transmişi (adică pentru un byte) se calculează un *bit de paritate* care are valoarea 0 (dacă codul ASCII al caracterului conţine un număr par de cifre binare 1) sau 1 (dacă codul ASCII al caracterului conţine un număr impar de cifre binare 1).
In cazul particular al problemei noastre se transmit doar caractere ASCII standard, care au codul ASCII in intervalul [32, 127], deci codul lor ASCII are bitul 7 (primul bit din stânga) egal cu 0 (intr-o configuratie de 1 byte/octet bitii se numeroteaza de la dreapta la stanga, bitul cel mai din dreapta fiind bitul 0). Pe această poziţie (a bitului 7) va fi pus bitul de paritate, ”economisind” astfel câte un bit pentru fiecare caracter transmis.

De exemplu, dacă mesajul care trebuie trasmis conţine caracterele "Paritate", succesiunea de biţi transmisă va fi:

01010000 **1**1100001 **0**1110010 **0**1101001 **0**1110100 **1**1100001 **0**1110100 **0**1100101
P a r i t a t e

În plus, pe lângă caractere, în mesaj mai poate să apară un caracterul special, caracter care indică trecerea la începutul unui nou rând. Acest caracter are codul ASCII 10 (newline).

Cerinţă
Să se verifice dacă un text a fost sau nu transmis corect.

Date de intrare
Fişierul de intrare input.in are pe prima linie o succesiune de caractere '0' şi '1' care reprezintă mesajul transmis. Între caractere nu există spaţii. Linia se termină cu caracterul marcaj de sfârşit de linie (newline).

Date de ieşire
Fişierul de ieşire rezultat.out are pe prima linie mesajul **DA** dacă textul a fost transmis corect sau **NU** în caz contrar.
În cazul în care mesajul de pe prima linie este **DA** liniile următoare vor conţine textul transmis în clar.
În cazul în care mesajul de pe prima linie este **NU** linia următoare va conţine numerele de ordine ale caracterelor care nu au fost transmise corect, în ordine strict crescătoare, separate prin câte un spaţiu.

Restricţii şi precizări

- Cei 8 biţi ai codului ASCII a unui caracter se numerotează de la 0 la 7, de la dreapta la stânga, cel mai din stânga bit fiind bitul 7 iar cel mai din dreapta bitul 0.
- Textul transmis are cel mult 60000 caractere.
- Numărul de caractere '0' şi '1' din prima linie a fişierului de intrare este multiplu de 8.
- Codurile ASCII ale caracterelor din text aparţin mulţimii {10, 32–127}, codul 10 însemnând trecerea la începutul unui rând nou.
- Nici o linie din fişierul de ieşire nu va avea mai mult de 255 caractere.
- Caracterele din text sunt numerotate începând de la 0.

Exemple

input.in	rezultat.out
0101000011100001011100100110100101110100111000010111010001100101	DA Paritate

Explicaţie raspuns: Toate codurile sunt corecte

input.in	rezultat.out
1101000011100001111100100110100101110100111000010111010011100101	NU 0 2 7

Explicaţie raspuns:
Primul caracter a fost transmis ca succesiunea de biţi 11010000 ceea ce înseamnă că fără bitul de paritate ar fi trebuit să existe un număr impar de cifre 1, ceea ce este fals. Deci caracterul nu a fost transmis corect. Acelaşi lucru se verifică şi pentru caracterele cu numerele de ordine 2 şi 7

input.in	rezultat.out
0100000111111010011010010000101001100101000010100110101001101111011010 01	DA Azi e joi

Explicaţie raspuns:
Toate codurile sunt corecte. În text există două caractere cu cod ASCII 10

Etape pentru rezolvarea problemei:

- Se va utiliza un tablou de caractere care va contine:
  - caracterul corect transmis sau
  - caracterul #0 in cazul in care transmisia nu s-a efectual corect

In acelasi timp variabila Eroare va contine ultima pozitie a unui cod eronat sau 0 daca nu sunt erori la transmiterea mesajului.

- Pentru fiecare byte/octet (deoarece stim ca numarul de biti 0 sau 1 transmisi este multiplu de 8 deci nu se mai fac verificari suplimentare):
  - se citeste primul caracter separat (este bitul de paritate)
  - se transforma in cifra 0/1
  - se citesc pe rand ceilalti 7 biti si se formeaza codul ASCII corect numarand in acelasi timp bitii egali cu 1
  - daca bitul de paritate este corect (adica daca exista un numar par de cifre 1)
    - se salveaza pe pozitia corespunzatoare din tablou caracterul al carui cod se obtine
    - in caz contrar, se pune pe pozitia respectiva valoarea #0 si se retine in variabila Eroare pozitia caracterului eronat

Dupa incheierea acestei etape trebuie doar sa se verifice variabila Eroare:
- in cazul in care are valoarea 0 (transmisie fara eroare), se va afisa 'DA' in prima linie a fisierului de iesire, apoi se parcurge vectorul caracter cu caracter si se scrie in fisierul de iesire, avand grija ca in cazul intalnirii caracterului #10 (cod de linie noua) sa se treaca la o noua linie
- in cazul in care are o valoare > 0 (transmisie cu erori), se va afisa 'NU' in prima linie a fisierului de iesire, apoi se parcurge vectorul caracter cu caracter si, in cazul intalnirii valorii #0 (caracter eronat) se va afisa indicele respectiv.

Solutie – varianta C

```
#include <stdio.h>

#define MAX 60000

char a[MAX];          //0: eroare; Caracter: corect
char c;
long i, j;
int Bitparitate, Cod, Bit, Contor;
long Eroare;          //va contine ultima pozitie a unui cod eronat sau 0 daca nu sunt erori
FILE *f, *g;

int main ()
{
 f = fopen ("input.in", "rt");
 g = fopen ("rezultat.out", "wt");
 i = -1;
 Eroare = 0;
 c = 0;
 fscanf (f, "%c", &c);
 while (c != '\n')
  {
   i++;        //pozitie caracter
   Bitparitate = c - '0';   //bitul de paritate
   Cod = 0;        //aici formez codul
   Contor = 0;      //numarul bitilor cu valoarea 1
   for (j = 1; j <= 7; j++)  //citesc ceilalti 7 biti
  {
   fscanf (f, "%c", &c); //citesc bit
   Bit = c - '0';
   if (Bit == 1)
    Contor++;      //daca e valoarea 1 il numar
   Cod = Cod * 2 + Bit;  //formez codul
  }
   if ((Contor + Bitparitate) % 2 == 0)  //daca cod corect
  a[i] = Cod;    //pun caracterul in vector
   else         //altfel
  {
   a[i] = 1;    //pun 1
   Eroare = i;      //si retin pozitia
  }
   fscanf (f, "%c", &c);
```

```
  }
 if (Eroare == 0)     //daca nu sunt erori
  {             //scrie DA si
    fprintf (g, "DA\n");
    for (j = 0; j <= i; j++)  //afiseaza cele i+1 caractere
  if (a[j] == 10)     //avand grija la caracterul cu codul 10
    fprintf (g, "\n");
  else          //altfel
    fprintf (g, "%c", a[j]);  //scrie caracterul
  }
 else           //eroare!!!
  {
    fprintf (g, "NU\n");  //scrie NU si
    for (j = 0; j < Eroare; j++)
  if (a[j] == 1)      //cauta erorile - cod 01
    fprintf (g, "%ld ", j);   //si afiseaza pozitia lor
    fprintf (g, "%ld\n", Eroare); //afiseaza pozitia ultimei erori
  }
 fclose (g);
 return 0;
}
```

Solutie – varianta Pascal

```
Program transmisie;

Const MAX = 60000;

Var a: Array[0..MAX-1] Of Char; {#0: eroare; Caracter: corect}
   f, g: Text;
   c: Char;
   i, j: LongInt;
   Bitparitate, Cod, Bit, Contor: Byte;
   Eroare: LongInt;          {va contine ultima pozitie}
                    {a unui cod eronat sau 0 daca nu sunt erori}
Begin
 Assign(f, 'input.in'); Reset(f);
 Assign(g, 'rezultat.out'); ReWrite(g);
 i := -1; Eroare := 0;;
 While NOT(EoLn(f)) Do
  Begin
   Inc(i);                  {pozitie caracter}
   Read(f, c); Bitparitate := Ord(c)-48;   {bitul de paritate}
   Cod := 0;                  {aici formez codul}
   Contor := 0;                  {cati de 1}
   For j := 1 To 7 Do            {citesc ceilalti 7 biti}
    Begin
     Read(f, c); Bit := Ord(c)-48;{citesc bit}
     If Bit=1 Then Inc(Contor);     {daca e 1 il numar}
     Cod := Cod*2+Bit;          {formez codul}
    End;
```

```
    If (Contor+Ord(Bitparitate=1)) MOD 2=0 Then{daca cod corect}
      a[i] := Char(Cod)            {pun caracterul in vector}
    Else                         {altfel}
      Begin
        a[i] := #0;              {pun #0}
        Eroare := i              {si retin pozitia }
      End;
  End;
 If Eroare=0 Then                {daca nu sunt erori}
  Begin                          {scrie DA si}
    WriteLn(g, 'DA');
    For j := 0 To i Do           {afiseaza cele i+1 caractere}
      If a[j]=#10 Then           {avand grija la caracterul cu codul 10}
        WriteLn(g)
      Else                       {altfel}
        Write(g, a[j]);          {scrie caracterul}
{     WriteLn(g)}
  End
 Else                            {eroare!!!}
  Begin
    WriteLn(g, 'NU');            {scrie NU si}
    For j := 0 To Eroare-1 Do
      If a[j]=#0 Then            {cauta erorile - cod #0}
        Write(g, j, ' ');        {si afiseaza pozitia lor}
    WriteLn(g, Eroare)           {afiseaza pozitia ultimei erori}
  End;
 Close(g);
End.
```

Problema 10

Enunt:

Se da ecuatia de gradul 2 de forma: $X^2 - s * X + p = 0$, cu s, p apartindand 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 => rezultatul este 20 (x1 =4 si x2=2)

Daca avem ecuatia: $X^2 - 8 * X + 15 = 0$ si n=2 => rezultatul este 34 (x1 = -3 si x2= -5)

Explicatii:

Stiind ca X1 si X2 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:

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

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 ecuatii de gradul II:

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

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 11

a) Sa se scrie un subprogram care prin intermediul a trei parametri, notaţi a, b şi c, reprezentand trei valori naturale nenule, fiecare de maximum patru cifre va returna valoarea 1 dacă cele trei valori ale celor trei parametrii pot constitui laturile unui triunghi şi valoarea 0 în caz contrar.

Varianta C++

```
int verificare (int a, int b, int c)
{
  return ((a + b > c) && (a + c > b) && (b + c > a));
}
```

Varianta Pascal

```
function verificare(a,b,c:integer):integer;
begin
    if(a+b>c) and (a+c>b) and (b+c>a)
        then verificare:=1
        else verificare:=0;
end;
```

b) Sa se scrie un program care citeşte de la tastatură şase valori naturale nenule, apoi verifică, utilizând apeluri utile ale subprogramului de la subpunctul anterior dacă primele trei numere citite pot constitui laturile unui triunghi şi dacă ultimele trei numere citite pot constitui laturile unui triunghi.

In caz afirmativ, programul va afisa pe ecran mesajul **congruente** dacă cele două triunghiuri sunt congruente sau mesajul **necongruente** dacă cele două triunghiuri nu sunt congruente.

Dacă cel puţin unul dintre cele două triplete de valori nu pot constitui laturile unui triunghi, programul va afişa pe ecran mesajul **nu se poate forma un triunghi** .

Obs. Triunghiurile sunt congruente dacă există cele două triunghiuri şi dacă au laturile corespunzătoare congruente (cazul L.L.L.).

Deci, pentru a verifica vom compara lungimile laturilor în ordine corespunzătoare(cea mai mică din primul triunghi cu cea mai mică din al doilea triunghi, pana se verifica toate laturile).

Varianta C++

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

```
/*
Descriere: Returneaza valoarea 1 daca valorile parametrilor pot forma un
triunghi, 0 in caz contrar.
Date: a, b, c - numare naturale
Rezultate: 1 sau 0
*/
int a, b, c, d, e, f;
int verificare (int a, int b, int c)
{
  return ((a + b > c) && (a + c > b) && (b + c > a));
}

/*
Descriere: Se interschimba valorile celor doua variabile transmise ca si
parametrii
Date: x, y - numare intregi
Rezultate: valorile interschimbate ale parametrilor initiali
*/
void interschimb (int &x, int &y)
{
  int aux = x;
  x = y;
  y = aux;
}

/*
Descriere: Sortarea a trei valori transmise prin parametrii
Date: x, y, z - numare intregi
Rezultate: valorile parametrilor ordonate
*/
void ordonare (int &x, int &y, int &z)
{
  if (x > y) interschimb (x, y);
  if (x > z) interschimb (x, z);
  if (y > z) interschimb (y, z);
}

int main ()
{
  cin >> a >> b >> c >> d >> e >> f;
  if (verificare (a, b, c) && verificare (d, e, f))
    {
      ordonare (a, b, c);
      ordonare (d, e, f);
      if (a == d && b == e && c == f)
    cout << "triunghiuri formate congruente";
      else
    cout << "triunghiuri formate necongruente";
    }
  else
    cout << "NU se poate forma un triunghi";
}
```

Varianta Pascal

```
var a,b,c,d,e,f:integer;
{
```

```
Descriere: Returneaza valoarea 1 daca valorile parametrilor pot forma un
triunghi, 0 in caz contrar.
Date: a, b, c - numare naturale
Rezultate: 1 sau 0
}
function verificare(a,b,c:integer):integer;
begin
    if(a+b>c) and (a+c>b) and (b+c>a)
        then verificare:=1
        else verificare:=0;
end;

{
Descriere: Se interschimba valorile celor doua variabile transmise ca si
parametrii
Date: x, y - numare intregi
Rezultate: valorile interschimbate ale parametrilor initiali
}
procedure interschimb(var x,y:integer);
var aux:integer;
begin
    aux:=x;
    x:=y;
    y:=aux;
end;

{
Descriere: Sortarea a trei valori transmise prin parametrii
Date: x, y, z - numare intregi
Rezultate: valorile parametrilor ordonate
}
procedure ordonare(var x,y,z:integer);
begin
    if(x>y) then interschimb(x,y);
    if(x>z) then interschimb(x,z);
    if(y>z) then interschimb(y,z);
end;

begin
    readln(a,b,c,d,e,f);
    if(verificare(a,b,c)=1) and (verificare(d,e,f)=1)
        then
        begin
            ordonare(a,b,c);
            ordonare(d,e,f);
            if(a=d) and (b=e) and (c=f)
                then
                writeln('triunghiuri formate congruente')
                else
                writeln('triunghiuri formate necongruente');
        end
        else
        writeln('NU se poate forma un triunghi');
end.
```