

Probleme consultații – 15 ianuarie 2022

Algoritmi care lucrează pe numere (fără tablouri sau alte elemente structurate)

Partea 1

Lect. Dr. Coroiu Adriana

Problema 1

Se da un număr natural n și expresia $s = 1 + 2^2 + 3^3 + \dots + n^n$ ($0 < n < 100$)

Dacă se citește numărul n , alegeți care dintre variantele de cod de mai jos vor determina **doar afișarea a ultimei cifre a rezultatului expresiei s** .

Exemplu

Date de intrare
 $n=3$

Date de ieșire
2

Explicație
Suma este 32 și ultima cifră 2.

// v1 – varianta corecta	// v2 – varianta gresita	// v3 – varianta gresita	// v4 – varianta gresita
<pre>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</pre>	<pre>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</pre>	<pre>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 <-- up mod 10; end; afisare us</pre>	<pre>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 div 10; us <-- up mod 10; end; afisare us</pre>

Problema 2

Se dau secvența de cod și afirmațiile de mai jos. Analizați afirmațiile date și alegeți afirmația corectă referitoare la codul analizat.

- codul determina elementul maxim dintre valorile introduse
- codul determina perechea cu valoare maximă dintre cele n perechi introduse
- codul determina perechea cu valoare minimă dintre cele n perechi introduse
- codul determina intersecția multimilor data de cele n perechi introduse**
- codul determina reuniunea multimilor data de cele n perechi introduse

```
#include <iostream>
```

<code>using namespace std;</code>	Exemplu:
<code>int main ()</code>	
<code>{</code>	3
<code>int n, a, b, max, min, i;</code>	
<code>cout << "n=";</code>	1 4
<code>cin >> n;</code>	
<code>cout << "perechea 1:";</code>	2 5
<code>cin >> a >> b;</code>	
<code>max = a;</code>	0 7
<code>min = b;</code>	
<code>for (i = 2; i <= n; i++)</code>	Raspuns: 2 3 4
<code>{</code>	
<code>cout << "perechea " << i << ":";</code>	
<code>cin >> a;</code>	
<code>cin >> b;</code>	
<code>if (max <= a)</code>	
<code>max = a;</code>	
<code>if (min >= b)</code>	
<code>min = b;</code>	
<code>}</code>	
<code>if (max <= min)</code>	
<code>{</code>	
<code>cout << "rezultatul este:" << endl;</code>	
<code>for (i = max; i <= min; i++)</code>	
<code>cout << i << " ";</code>	
<code>}</code>	
<code>else</code>	
<code>cout << "rezultatul corect nu se poate</code>	
<code>determina";</code>	
<code>return 0;</code>	
<code>}</code>	

Problema 3

Se citește de la tastatură un număr n ($0 \leq n \leq 2147483647$). Să se afișeze numărul de biți setați din reprezentarea binară a numărului n .

Ex. $n = 27 \Rightarrow 4$
 $n = 10 \Rightarrow 2$
 $n = 2147483647 \Rightarrow 31$

Soluții:

Varianta nerecursivă (cu împărțiri la 2) – v1

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

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

Varianta recursivă (cu impartiri la 2) – v2

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

```
int numar_bit_i_recursiv(int numar) {
    if (numar == 0) return 0;
    if (numar % 2 == 1) return 1 + numar_bit_i_recursiv(numar / 2);
    return numar_bit_i_recursiv(numar / 2);
}

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

Varianta nerecursivă cu shiftare pe biti (v3)

```
#include <stdio.h>

int numar_bit_i_nerecursiv(int numar) {
    int numar_bit_i = 0;
    while (numar != 0) {
        if (numar & 1 == 1) numar_bit_i++;
        numar = numar >> 1;
    }
    return numar_bit_i;
}

int main() {
    int numar;
    scanf("%d", &numar);
    printf("%d", numar_bit_i_nerecursiv(numar));
    return 0;
}
```

Varianta recursivă cu shiftare pe biti (v4)

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

```
int numar_bit_i_recursiv(int numar) {
    if (numar == 0) return 0;
    if (numar & 1 == 1) return 1 + numar_bit_i_recursiv(numar >> 1);
}
```

```
    return numar_bit recursiv(numar >> 1);
}
int main() {
    int numar;
    scanf("%d", &numar);
    printf("Rec: %d", numar_bit recursiv(numar));
    return 0;
}
```

Problema 4

Se da un număr n ($0 \leq n \leq 2147483647$)

Sa se scrie o secvență de cod recursivă care determină valoarea numărului citit în baza 2.

Varianta recursivă:

```
#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 5.

Analizați secvența de cod de mai jos (pentru numărul n restricțiile sunt: $-32,768 \leq n \leq 32,767$).

Identificați afirmația/afirmațiile 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 inversul numarului citit
- b. **Codul determina valoarea in binar a unui nr pozitiv citit.**
- c. Codul transforma un numar pozitiv citit in numar negativ.
- d. Codul transforma un numar negativ citit in numar pozitiv.

Problema 6.

Se da un numar n in baza 10 ($0 \leq n \leq 65535$). Sa se determine valoarea numarului n in baza 16.

Exemplu:

$n = 7 \Rightarrow 7$;

$n = 10 \Rightarrow A$;

$n = 3146 \Rightarrow C4A$

```
#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 7

Scrieti o secvență de cod care realizează conversia unui număr din baza 2 sau baza 16 în baza 10.

De exemplu:

n = 1010 (în binar) => 10 în baza 10
n = 2AB (în hexazecimal) => 683 în baza 10
n = 11111100 (în binar) => 252 în 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 8

Pentru a se asigura o transmitere cat mai exactă a informațiilor pe rețea, 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 biți (un byte sau un octet).

Pentru fiecare 8 biți transmiși (adică pentru un byte) se calculează un *bit de paritate* care are valoarea 0 (dacă codul ASCII al caracterului conține un număr par de cifre binare 1) sau 1 (dacă codul ASCII al caracterului conține un număr impar de cifre binare 1).

În cazul particular al problemei noastre se transmit doar caractere ASCII standard, care au codul ASCII în intervalul [32, 127], deci codul lor ASCII are bitul 7 (primul bit din stânga) egal cu 0 (într-o configurație de 1 byte/octet bitii se numerează de la dreapta la stânga, bitul cel mai din dreapta fiind bitul 0). Pe această poziție (a bitului 7) va fi pus bitul de paritate, "economisind" astfel câte un bit pentru fiecare caracter transmis.

De exemplu, dacă mesajul care trebuie transmis conține caracterele "Paritate", succesiunea de biți transmisă va fi:

```
01010000 11100001 01110010 01101001 01110100 11100001 01110100 01100101
    P      a      r      i      t      a      t      e
```

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

Cerință

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

Date de intrare

Consultații Facultatea de Matematică și Informatică

Fișierul de intrare input.in are pe prima linie o succesiune de caractere '0' și '1' care reprezintă mesajul transmis. Între caractere nu există spații. Linia se termină cu caracterul marcaj de sfârșit de linie (newline).

Date de ieșire

Fișierul de ieșire 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 conține textul transmis în clar.

În cazul în care mesajul de pe prima linie este **NU** linia următoare va conține numerele de ordine ale caracterelor care nu au fost transmise corect, în ordine strict crescătoare, separate prin câte un spațiu.

Restricții și precizări

- Cei 8 biți ai codului ASCII a unui caracter se numerează 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' și '1' din prima linie a fișierului de intrare este multiplu de 8.
- Codurile ASCII ale caracterelor din text aparțin mulțimii {10, 32–127}, codul 10 însemnând trecerea la începutul unui rând nou.
- Nici o linie din fișierul de ieșire 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

Explicație raspuns: Toate codurile sunt corecte

input.in	rezultat.out
1101000011100001111100100110100101110100111000010111010011100101	NU 0 2 7

Explicație raspuns:

Primul caracter a fost transmis ca succesiunea de biți 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. Același lucru se verifică și pentru caracterele cu numerele de ordine 2 și 7

input.in	rezultat.out
010000011111101001101001000010100110010100001010011010100110111101101001	DA Azi e joi

Explicație 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 conține:
 - * caracterul corect transmis sau
 - * caracterul #0 în cazul în care transmisia nu s-a efectuat corect

În același timp variabila Eroare va conține ultima poziție a unui cod eronat sau 0 dacă nu sunt erori la transmiterea mesajului.

- Pentru fiecare byte/octet (deoarece știm că numărul de biți 0 sau 1 transmiși este multiplu de 8 deci nu se mai fac verificări suplimentare):
 - * se citește primul caracter separat (este bitul de paritate)
 - * se transformă în cifră 0/1
 - * se citește pe rând ceilalți 7 biți și se formează codul ASCII corect numărând în același timp biții egali cu 1
 - * dacă bitul de paritate este corect (adică dacă există un număr par de cifre 1)
 - se salvează pe poziția corespunzătoare din tablou caracterul al cărui cod se obține
 - în caz contrar, se pune pe poziția respectivă valoarea #0 și se reține în variabila Eroare poziția caracterului eronat

După încheierea acestei etape trebuie doar să se verifice variabila Eroare:

- în cazul în care are valoarea 0 (transmisie fără eroare), se va afișa 'DA' în prima linie a fișierului de ieșire, apoi se parcurge vectorul caracter cu caracter și se scrie în fișierul de ieșire, având grijă ca în cazul întâlnirii caracterului #10 (cod de linie nouă) să se treacă la o nouă linie

- în cazul în care are o valoare > 0 (transmisie cu erori), se va afișa 'NU' în prima linie a fișierului de ieșire, apoi se parcurge vectorul caracter cu caracter și, în cazul întâlnirii valorii #0 (caracter eronat) se va afișa indicele respectiv.

Soluție – 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 conține ultima poziție a unui cod eronat sau 0 dacă 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++;          //poziție caracter
```

```
        Bitparitate = c - '0'; //bitul de paritate
```

```
        Cod = 0;       //aici formează codul
```

```
        Contor = 0;    //numărul bitilor cu valoarea 1
```

```
        for (j = 1; j <= 7; j++) //citesc ceilalți 7 biți
```

```
        {
```

```
            fscanf (f, "%c", &c); //citesc bit
```

```

    Bit = c - '0';
    if (Bit == 1)
        Contor++; //daca e valoarea 1 il numar
    Cod = Cod * 2 + Bit; //formeaz 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 9

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:

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

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

Explicații:

Știind că X_1 și X_2 sunt rădăcinile ecuației, rezultă relațiile:

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

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

Dacă vom înmulți prima relație cu X_1^n și pe cea de a doua cu X_2^n vom obține:

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

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

Dacă însumăm cele două relații, vom obține:

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

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

Vom obține de aici următoarea relație de recurență pentru suma puterilor rădăcinilor unei ecuații de gradul II:

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

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.
```