



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

Partea II

13.02.2021

Probleme recursiv/iterativ

1. a) Verifica dacă un număr este cifru după regula dată: pe poziții pare sunt cifre impare și pe poziții impare sunt cifre pare.
1. b) Verifica dacă un CNP este valid.

1.a) Verifica dacă un număr este cifru după regula enunțată.

Exemplu

nr = 1234 – este cifru

nr = 23452 – este cifru

nr = 2467 - nu este cifru

nr := 2340; - False

nr := 2345700; - False

nr := 23450; - True

Varianta de rezolvare – implementare Pascal

Program ProgramCifru;

function numarDeCifreRecursiv(n: Integer): Integer;

begin

if (n < 10) then

numarDeCifreRecursiv := 1

else

numarDeCifreRecursiv := 1 + numarDeCifreRecursiv(n div 10);

end;

function numarDeCifreIterativ(n: Integer): Integer;

var nCif: Integer;

begin

nCif := 0;

while (n > 0) do

begin

nCif := nCif + 1;

n := n div 10;

end;

numarDeCifreIterativ := nCif;

end;

function esteCifru(n: Integer; p: integer): Boolean;

var cif: integer;

begin

if (n > 0) then

begin

cif := n mod 10;

if ((p mod 2 <> 0) and (cif mod 2 = 0)) or ((p mod 2 = 0) and (cif mod 2 <> 0)) then
esteCifru := esteCifru(n div 10, p + 1)

else

esteCifru := false;

end



```
    else
        esteCifru:=true;
    end;

function startEsteCifru(n:Integer):boolean;
var nCif:Integer;
begin
    nCif := numarDeCifreIterativ(n);
    if (nCif mod 2 <>0) then
        begin
            startEsteCifru := esteCifru(n, 1);
        end
    else
        startEsteCifru := esteCifru(n,0);
    end;
end;

var nr:Integer;
    // nCifre:Integer;
    eCifru:boolean;
begin
    { nCifre := numarDeCifreRecursiv(nr);
      writeln('[Recursiv] Numarul ', nr, ' are ',nCifre, ' numar de cifre. ');
      nCifre := numarDeCifreIterativ(nr);
      writeln('[Iterativ] Numarul ', nr, ' are ',nCifre, ' numar de cifre. ');
    }
    //nr := 1234; //True
    //nr := 23452; // True
    //nr := 2467; // False
    //nr := 2340; // False
    //nr := 2345700; //False
    nr := 23450; //True
    eCifru := startEsteCifru(nr);
    writeln('Este cifru ',eCifru);
    readln;
end.
```

Varianta de rezolvare – implementare C++

```
#include <iostream>
using namespace std;
int numarDeCifreRecursiv(int n){
    if (n<10)
        return 1;
    else
        return (1 + numarDeCifreRecursiv(n/10));
}
int numarDeCifreIterativ(int n){
    int nCif = 0;
    nCif = 0;
    while (n>0){

        nCif=nCif+1;
        n = n/ 10;
    }
    return nCif;
}
bool esteCifru(int n, int p){
```



```
int cif=0;
if (n>0){
    cif = n % 10;
    if (((p % 2 != 0) && (cif % 2 ==0)) || ((p % 2 == 0) && (cif % 2 !=0)))
        return esteCifru(n/10, p+1);
    else
        return false;
}
else
    return true;
}
bool startEsteCifru(int n){
int nCif=0;
nCif = numarDeCifreIterativ(n);
if (nCif % 2 !=0)
    return esteCifru(n, 1);
else
    return esteCifru(n,0);
}

int main(){
/*
int nr = 23450; //True
int nCifre = numarDeCifreRecurziv(nr);
cout<<"[Recurziv] Numarul "<<nr<<" are "<<nCifre<<" numar de cifre."<<endl;
nCifre = numarDeCifreIterativ(nr);
cout<<"[Iterativ] Numarul "<<nr<<" are "<<nCifre<<" numar de cifre."<<endl;
*/

//nr = 1234; //True
//nr = 23452; // True
//nr = 2467; // False
//nr = 2340; // False
int nr = 2345700; //False
//int nr = 23450; //True
bool eCifru = startEsteCifru(nr);
cout<<"Este cifru "<<eCifru<<endl;
return 0;
}
```

1. b) Verifica daca un CNP este valid.

Codul numeric personal este format din 13 cifre, unic pentru fiecare persoană fizică și este format din 7 componente:

Componenta **S** reprezintă sexul și secolul în care s-a născut persoana și poate avea una dintre următoarele valori:

- 1 pentru persoanele de sex masculin născute între anii 1900 - 1999
- 2 pentru persoanele de sex feminin născute între anii 1900 - 1999
- 3 pentru persoanele de sex masculin născute între anii 1800 - 1899
- 4 pentru persoanele de sex feminin născute între anii 1800 - 1899
- 5 pentru persoanele de sex masculin născute între anii 2000 - 2099
- 6 pentru persoanele de sex feminin născute între anii 2000 - 2099
- 7 pentru persoanele rezidente, de sex masculin^[6]
- 8 pentru persoanele rezidente, de sex feminin^[6]

Componenta **AA** este formată din ultimele 2 cifre ale anului nașterii

Componenta **LL** este formată din luna nașterii, cu valori între 01 și 12

Componenta **ZZ** este formată din ziua nașterii, cu valori între 01 și 28, 29, 30 sau 31, după caz



Componenta **JJ** reprezintă județul sau sectorul în care s-a născut persoana, ori în care avea domiciliul sau reședința la momentul acordării C.N.P.

Componenta **NNN** reprezintă un număr secvențial (*cuprins între 001 și 999*), repartizat pe puncte de atribuire, prin care se diferențiază persoanele de același sex, născute în același loc și cu aceeași dată de naștere

Componenta **C** este formată dintr-o cifră de control, care permite depistarea eventualelor erori de înlocuire sau inversare a cifrelor din componența C.N.P

Validarea unui C.N.P. constă în calcularea componentei **C** și compararea acesteia cu valoarea primită a aceleiași componente. Dacă acestea sunt identice, înseamnă că C.N.P. verificat este valid.

Calcularea componentei **C** se face folosind constanta "279146358279", după cum urmează:

- fiecare cifră din primele 12 cifre ale C.N.P. este înmulțită cu corespondentul său din constantă
- rezultate sunt însumate și totalul se împarte la 11
- dacă restul împărțirii este mai mic de 10, acela reprezintă valoarea componentei **C**
- dacă restul împărțirii este 10, valoarea componentei **C** este 1

CNP – generare

<https://isi.educ.ro/cnp/>

6110213125566

Varianta de rezolvare – implementare Pascal

program ValidareCNP;

function validareCNP(cnp:int64; nrConst:int64):boolean;

var s,c,rest:int64;

i:integer;

begin

s := 0;

c := cnp mod 10;

cnp := cnp div 10;

for i:=1 to 12 do

begin

s := s + (cnp mod 10)* (nrConst mod 10);

cnp := cnp div 10;

nrConst:= nrConst div 10;

end;

rest := s mod 11;

if (rest < 10) then

if (rest = c) then

validareCNP :=true

else

if (c=1) then

validareCNP := true

else

validareCNP := false;

end;

function validareCNPsuma(cnp:int64; nrConst:int64):integer;

var cCNP,cNrConst,s:int64;

begin

if (cnp<10) then

validareCNPsuma := cnp*nrConst

else

begin

cCNP := cnp mod 10;

cNrConst := nrConst mod 10;

s := cCNP * cNrConst;

validareCNPsuma := s + validareCNPsuma(cnp div 10, nrConst div 10);

end;

end;



```
function validareCNPrecursiv(cnp:int64; nrConst:int64):boolean;
var c,s, rest: int64;
begin
  c := cnp mod 10;
  s := validareCNPsuma(cnp div 10, nrConst);
  rest := s mod 11;
  if (rest < 10) then
    if (rest = c) then
      validareCNPrecursiv :=true
    else
      if (c=1) then
        validareCNPrecursiv := true
      else
        validareCNPrecursiv := false;
  end;

var cnp, nrConstanta:int64;
begin
  //cnp := 6110213125566; // True
  //cnp := 6110213125562; // False

  //cnp := 6210203017139; // True
  //cnp := 6210203017135; // False
  nrConstanta := 279146358279;
  writeln('Numarul CNP ',cnp,' este ', validareCNP(cnp, nrConstanta));
  writeln('Numarul CNP ',cnp,' este ', validareCNPrecursiv(cnp, nrConstanta));
  readln();
end.
```

Varianta de rezolvare – implementare C++

```
#include <iostream>
using namespace std;
bool validareCNP(long cnp, long nrConst){
  long s,c,rest;
  int i;
  s = 0;
  c = cnp % 10;
  cnp = cnp / 10;
  for (i=1;i<12;i++){
    s = s + (cnp % 10)* (nrConst % 10);
    cnp = cnp/10;
    nrConst = nrConst/10;
  }
  rest = s % 11;
  if (rest < 10)
    if (rest == c)
      return true;
    else
      if (c==1)
        return true;
      else
        return false;
}
int validareCNPsuma(long cnp, long nrConst){
  long cCNP,cNrConst,s,vSuma;
  if (cnp<10)
```



```
vSuma = cnp*nrConst;
else
{
    cCNP = cnp % 10;
    cNrConst = nrConst % 10;
    s = cCNP * cNrConst;
    vSuma = s + validareCNPsuma(cnp/10, nrConst/10);
}
return vSuma;
}

bool validareCNPrecursiv(long cnp, long nrConst){
long c,s, rest;
c = cnp % 10;
s = validareCNPsuma(cnp/10, nrConst);
rest = s % 11;
if (rest < 10)
    if (rest == c)
        return true;
    else
        if (c==1)
            return true;
        else
            return false;
}

long cnp, nrConstanta;
int main(){

    //cnp = 6110213125566; // True
    //cnp = 6110213125562; // False

    cnp = 6210203017139; // True
    //cnp = 6210203017135; // False
    nrConstanta = 279146358279;
    cout<<"Numarul CNP "<<cnp<<" este "<< validareCNP(cnp, nrConstanta)<<endl;
    cout<<"Numarul CNP "<<cnp<<" este "<< validareCNPrecursiv(cnp, nrConstanta)<<endl;
    return 0;
}
```

2) Problema (analiza, proiectare, implementare)

2. a) Se citesc emotii (-1=trist, 0=neutru, 1= fericit) pana la introducerea unui numar pentru oprire (2, de exemplu).

2. a.1.) Sa se determine cate perechi consecutive de acelasi fel sunt (-1,-1) si (0,0) si (1,1).

2. a.2.) Sa se determine cate triplete consecutive de forma (-1,1,0) sau (0,1,-1) (adica cu fericit la mijloc) sunt.

2. b) Se citesc numere negative/pozitive care reprezinta cheltuieli si venituri. Cifra 0 introdusa semnifica ca incepe evidenta pe luna urmatoare. Oprirea la 2 cifre de 0 consecutive introduse.

2. b.1.) Sa se afiseze pentru fiecare luna: cheltuielile, veniturile, soldul la final.

2. b.2.) Pentru cate luni s-a tinut evidenta?

2.a.1) Emotii (-1=trist, 0=neutru, 1=fericit)

Numere citite: -1, -1, -1, 0, 0, 1, 1, -1, -1, 0, 0, 1, 1, 0, -1, 1, 0, 0, 1, 1, 2



2.a.2) Triplete $(-1,1,0)$ sau $(0,1,-1)$

Numere citite : $-1,1,0,0,1,1,-1,-1,1,0,1,-1,-1,0,2$

Varianta de rezolvare – implementare Pascal

```
program Emotii_Perechi_Triplete;
var x,y, nrPerechi, z, nrTriplete:integer;
begin
    nrPerechi := 0;
    writeln('dati primul numar:');
    readln(x);
    while(x <>2) do
    begin
        writeln('dati urmatorul numar:');
        readln(y);
        if ((x = -1) and (y = -1)) or ((x=0) and (y=0)) or ((x=1) and (y=1)) then
            nrPerechi:= nrPerechi +1;
        x := y;
    end;
    writeln('Numarul de perechi la fel:', nrPerechi);
    readln();

    nrTriplete := 0;
    writeln('dati primul numar:');
    readln(x);
    writeln('dati urmatorul numar:');
    readln(y);
    while(y <>2) do
    begin
        writeln('dati urmatorul numar:');
        readln(z);
        if ((x = -1) and (y = 1) and (z=0)) or ((x=0) and (y=1) and (z=-1)) then
            nrTriplete:= nrTriplete +1;
        x := y;
        y := z;
    end;
    writeln('Numarul de triplete la fel:', nrTriplete);
    readln();
end.
```

Varianta de rezolvare – implementare C++

```
#include <iostream>
using namespace std;
int main()
{
    int x,y, nrPerechi, z, nrTriplete;
    nrPerechi = 0;
    cout<<"dati primul numar:";
    cin>>x;
    while(x !=2){
        cout<<"dati urmatorul numar:";
        cin>>y;
        if (((x == -1) && (y ==-1)) || ((x==0) && (y==0)) || ((x==1) && (y==1)))
            nrPerechi= nrPerechi +1;
        x = y;
    }
    cout<<"Numarul de perechi la fel:"<<nrPerechi;
```



```
nrTriplete = 0;
cout<<"dati primul numar:";
cin>>x;
cout<<"dati urmatorul numar:";
cin>>y;
while(y !=2){
    cout<<"dati urmatorul numar:";
    cin>>z;
    if (((x == -1) && (y ==1) && (z==0)) || ((x==0) && (y==1) && (z==1)))
        nrTriplete= nrTriplete +1;
    x = y;
    y = z;
}
cout<<"Numarul de triplete la fel:"<<nrTriplete;
return 0;
}
```

2.b.1) pentru fiecare luna: venituri, cheltuieli , sold
2.b.2) numar de luni cu evidenta

Nr citite : 20, -10, -2, 100, 20, 15, -3, 0, 40, -10, -2, 120, 20, 15, -3, 0, 20, -30, -20, 100, 20, 15, -30, 0, 0

Luna 1 : Venituri : 155, Cheltuieli : 15, Sold = 140
Luna 2 : Venituri : 175, Cheltuieli : 15, Sold = 160
Luna 3 : Venituri : 155, Cheltuieli : 80, Sold = 75
Numar de luni cu evidenta : 3

Varianta de rezolvare – implementare Pascal

```
program EvidentaSold;
var nr,nrN, venit,chelt,sold,nrL:integer;
begin
    writeln('Citeste o valoare:');
    readln(nr);

    venit:=0; chelt :=0; sold:=0;
    nrL:=0;
    nrL :=1;

    while (nr<>0) do
        begin
            if (nr>0) then
                venit := venit + nr
            else
                if (nr<0) then
                    chelt := chelt + nr;
            writeln('Citeste o valoare:');
            readln(nr);
            if (nr=0) then
                begin
                    sold := sold + venit + chelt;
                    writeln('Pentru luna ', nrL, ' venit=', venit, ', chelt=',chelt,' si sold = ', sold);
                    venit :=0; chelt:=0;
                    nrL:=nrL+1;
                end
        end
    end
```




```
writeln('Citeste o valoare:');  
readln(nr);  
end;  
end;  
  
writeln('Numarul de luni cu evidenta=',nrL-1);  
readln();  
end.
```

Varianta de rezolvare – implementare C++

```
#include <iostream>  
using namespace std;  
int nr,nrN, venit,chelt,sold,nrL;  
int main(){  
    cout<<"Citeste o valoare:";  
    cin>>nr;  
    venit=0; chelt =0; sold=0;  
    nrL =0;  
    nrL =1;  
  
    while (nr!=0){  
        if (nr>0)  
            venit = venit + nr;  
        else  
            if (nr<0)  
                chelt = chelt + nr;  
        cout<<"Citeste o valoare:";  
        cin>>nr;  
        if (nr==0){  
            sold = sold + venit + chelt;  
            cout<<"Pentru luna "<<nrL<<" venit="<<venit<<" chelt="<<chelt<<" si sold = "<<sold;  
            venit =0; chelt=0;  
            nrL=nrL+1;  
  
            cout<<"Citeste o valoare:";  
            cin>>nr;  
        }  
    }  
    cout<<"Numarul de luni cu evidenta="<<(nrL-1);  
    return 0;  
}
```

3) Probleme grila

3.a) Stabiliti care dintre variabilele intregi pozitive x si y trebuie sa aiba valoarea initiala 1, pentru ca, la sfarsitul executarii urmatoarei secvente de instructiuni, variabila z sa aiba valoarea 3 ?

```
z ← 0 ;  
Pentru i ← x la y executa  
    z ← z+i;  
SfPentru  
Daca z=1 atunci z ← 3  
    altfel z ← 5;
```



SfDaca

- a) numai x
- b) numai y
- c) atat x, cat si y
- d) nici x, nici y

3.b) Stiind ca x si y desemneaza variabile intregi, determinati valoarea initiala a variabilei x astfel incat secventa data sa afiseze exact un asterisc (*).

```
y ← x ;  
CatTimp x <= 3 executa  
    Scrie '*';  
    y ← y + 1;  
    x ← x + y;  
SfCatTimp;
```

- a) -4
- b) -2
- c) 0
- d) 1
- e) 2
- f) 4
- g) 5

3.c) Care va fi valoarea afisata de catre programul urmator daca a=b=3 ?
Variabile intregi a,b,c,z,i

```
Citeste a;  
Citeste b;  
c ← 0; z ← 1;  
Pentru i ← 1 la a executa  
    c ← c + z;  
    z ← z * b;  
SfPentru  
Scrie c
```

- a) 8
- b) 32
- c) 27
- d) 13
- e) 1

3.d) Se considera urmatoarea secventa de instructiuni in pseudocod:

```
Citeste n  
nr ← 0  
Citeste a  
Pentru i ← 2, n executa  
    Citeste b  
    Daca a <> b atunci  
        nr ← nr + 1  
SfDaca  
a ← b ;  
SfPentru  
Scrie nr
```



Ce se va afișa pe ecran dacă se citesc valorile : 8, 1, 1, 1, 2, 3, 5, 3, 3 ?

- a) 3 **b) 4** c) 7 d) 5

Se va afișa 4 (perechi de elemente consecutive diferite).

3.e) Se considera programul pseudocod alăturat:

Citeste x,y (numere naturale)

$nr \leftarrow 0$

$d \leftarrow 2$

CatTimp $d \leq x$ si $d \leq y$ executa

Daca $x \% d = 0$ si $y \% d = 0$ atunci

$nr \leftarrow nr + 1$;

$x \leftarrow [x/d]$;

$y \leftarrow [y/d]$;

altfel

$d \leftarrow d + 1$

SfDaca

SfCatTimp

Scrie nr, x, y

- a) Ce se va afișa dacă se citesc valorile $x=720$ si $y=495$?

i. nr=3, x=16, y=11

ii. nr=2, x=48, y=33

iii. nr=3, x=16, y=33

iv. nr=3, x=48, y=11

- b) Determinați toate perechile de valori de cel mult 2 cifre care se pot citi pentru x si y astfel încât să se afișeze valorile 1 7 11.

Determinați toate (cate) perechile de valori de cel mult 2 cifre care se pot citi pentru x si y astfel încât să se afișeze valorile 1 , 7, 11.

i. 4

ii. 8

iii. 2

iv. 5

- a) Descompunerea în factori primi a valorilor de intrare x si y, numărând aparițiile factorilor (divizorilor) comuni (inclusiv repetările aceluiași divizor).

Pentru $x=720$ si $y=495$ avem divizorul comun 3 apare de 2 ori, divizorul comun 5 apare o singură dată. \Rightarrow Rezultatele afișate vor fi : $nr=3, x=16, y=11$

- b) Perechile cerute au proprietatea că au un singur factor prim comun care apare o singură dată, si x: factorul comun da catul 7 iar y :factorul comun da catul 11.

- Divizor comun 2 : [14,22]

- Divizor comun 3 : [21,33]

- Divizor comun 5 : [35,55]

- Divizor comun 7 : [49,77]