



## Algoritmi care lucrează cu tablouri unidimensionale

### Partea I

28.11.2020

1. Fie șirul  $x=(5, 3, 2, 1, 1, 1)$ . Ce va realiza următorul algoritm?

<pre><b>{Pascal}</b> for i:=1 to n do begin     c:=x[i]     x[i]:= x[n-i+1];     x[n-i+1]:=c; end; for i:=1 to n do     Write (x[i],',','');</pre>	<pre>//C for (i=1;i&lt;=n;i++) {     c=x[i];     x[i]=x[n-i+1];     x[n-i+1]=c; } for (i=1;i&lt;=n;i++)     printf("%d,", x[i]);</pre>
--	--

- A. 1,1,2,1,3,5
- B. 1,1,1,2,3,5
- C. 5,3,2,1,1,1
- D. Nici una dintre variantele anterioare nu este corecta.

2. Se consideră subalgoritmul  $h(n)$ , unde  $n$  este un număr natural ( $1 \leq n \leq 10000$ ) și  $A=(A_1, A_2, \dots, A_n)$  un șir de numere întregi.

```
Functia h(A, n):
    Dacă n=0 atunci
        returnează 0;
    altfel
        Dacă n mod 2=1 atunci
            returnează h(A,n-1)+A[n];
        altfel
            returnează h(A,n-1)-A[n];
sfDaca
sfDaca
SfFunctie
```

**Algoritmul calculează:**

- a. Diferența dintre suma elementelor de pe poziții pare și suma elementelor de pe pozițiile impare din vectorul A
- b. Diferența elementelor de pe pozițiile impare din vectorul A
- c. Diferența dintre suma elementelor pare din vector și suma elementelor impare din vectorul A
- d. Diferența dintre suma elementelor de pe poziții impare și suma elementelor de pe pozițiile pare din vectorul A



3. Fiind dat un polinom  $P(X)=a_nX^n+a_{n-1}X^{n-1}+\dots+a_1X^1+a_0$  cu coeficienti reali ( $a_i \in \mathfrak{R}, \forall i \in \{0, \dots, n\}, a_n \neq 0$ ). Scrieti un subalgoritm care determina  $P(q)$ , valoarea polinomului  $P$  in punctul  $q$ .

$$P(X)=3X^4+6X^3-5X^2-3X^1+2 \text{ si } q=0 \Rightarrow P(0)=3 \cdot 0^4+6 \cdot 0^3-5 \cdot 0^2-3 \cdot 0+2=2$$

$$P(X)=X^5-X^3+2X^1 \text{ si } q=2 \Rightarrow P(2)=2^5-2^3+2 \cdot 2=28$$

$$P(X)=X^5-X^3+2X^1 \text{ si } q=-2 \Rightarrow P(-2)=(-2)^5-(-2)^3+2 \cdot (-2)=-28$$

Notatii:

$n$ - gradul polinomului  $P$

$a$  - sirul coeficientilor  $a=(a_0, a_1, \dots, a_n)$

$q$ - punctul in care dorim sa determinam valoarea

Functia ValPol( $a,n,q$ )

Descriere: Calculeaza valoarea polinomului  $a_nX^n+a_{n-1}X^{n-1}+\dots+a_1X^1+a_0$  in punctul  $q$ ,

$$\text{val}=a_nq^n+a_{n-1}q^{n-1}+\dots+a_1q^1+a_0$$

Date de intrare:  $a, n, q$

Rezultate: val

#### Varianta 1 - nerecursiva

<p>Functia ValPol(<math>a,n,q</math>) este:</p> <pre> val:=0; t:=1; Pentru i:=0 , n, +1, executa:     val:=val+a[i]*t;     t:=t*q; sfPentru returneaza val; sfFunctia;</pre>	<p>Functia ValPol(<math>a,n,q</math>) este:</p> <pre> val:=a[0]; t:=1; Pentru i:=1 , n, +1, executa:     t:=t*q;     val:=val+a[i]*t; sfPentru returneaza val; sfFunctia;</pre>
--	---

#### Varianta 1 - recursiva a)

Functia ValPolRec( $a,n,q, t, i$  ) este:

Daca  $i > n$  atunci

returneaza 0;

altfel

returneaza  $a[i] \cdot t + \text{ValPolRec}(a,n,q,t \cdot q, i+1)$ ;

sfDaca

sfFunctia

**Observatie** Functia se va apela astfel:  $\text{ValPolRec}(a,n,q, 1,0)$  !

#### Varianta 1 - recursiva b)



Funcția  $\text{ValPolRec2}(a, n, q, t)$  este:

```
Daca n=0 sau q=0 atunci
    returneaza a[0];
altfel
    returneaza a[n]*t+ValPolRec2(a, n-1, q, t/q);
sfDaca
SfFuncție
```

**Observatie** Funcția se va apela astfel:  $\text{ValPolRec2}(a, n, q, q^n)$

### Varianta 2 - nerecursiva

Scriem polinomul  $P(X)=P(X)=a_nX^n+a_{n-1}X^{n-1}+\dots+a_1X^1+a_0$  astfel:

$$P(X) = (\dots(((a_n * X + a_{n-1}) * X + a_{n-2}) * X + a_{n-3}) * X + \dots a_1) * X + a_0 \quad (\text{Horner})$$

Funcția  $\text{ValPolHorner}(a, n, q)$  este:

```
val:=a[n];
Pentru i:=n-1, 0, -1 este:
    val:=val*q+a[i];
sfPentru
returneaza val;
sfFuncție
```

### Varianta 2 - recursiva a)

Funcție  $\text{ValPolHornerRecA}(a, n, q, i)$  este:

```
Daca (n=i) atunci
    returneaza a[n];
altfel
    returneaza ValPolHornerRecA(a, n, q, i+1)*q+a[i];
sfDaca
SfFuncție
```

**Observatie** Funcția se va apela astfel:  $\text{ValPolHornerRecA}(a, n, q, 0)$

### Varianta 2 - recursiva b)



Funcție ValPolHornerRecB(a,n,q,i) este:

```
Daca (i=0) atunci
    returneaza a[n-i];
altfel
    returneaza ValPolHornerRecB(a,n,q,i-1)*q+a[n-i];
sfDaca
SfFuncție
```

**Observatie** Funcția se va apela astfel: ValPolHornerRecB(a,n,q,n)

4. Fiind dat un polinom  $P(X)=a_nX^n+a_{n-1}X^{n-1}+\dots+a_1X^1+a_0$  cu coeficienți întregi ( $a_i \in \mathbb{Z}, \forall i \in \{0, n\}, a_n \neq 0$  și  $n \in \mathbb{N} \setminus \{0\}$ ), scrieți un subalgoritm care determină rădăcinile întregi distincte ale polinomului  $P$ .

(specificare, impartire subalgoritmi)

Exemple:

$P_1(X)=X^2+2x+1$ , rădăcinile întregi distincte sunt:  $x_1=-1$

$P_2(x)=X^3+2x^2-x-2$ , rădăcinile întregi distincte ale polinomului sunt:  $x_1=1, x_2=-2, x_3=-1$

$P_3(X)=X^5-X^3$ , rădăcinile întregi distincte ale polinomului sunt:  $x_1=0, x_2=-1, x_3=1$

Explicatie:

1. Dacă  $a_0 \neq 0$ , se caută rădăcinile întregi ale polinomului printre divizorii coeficientului  $a_0$ .
2. Dacă  $a_0 = 0$  atunci 0 este o rădăcină întreagă a polinomului. Pentru a determina alte rădăcini întregi, întâi se determină, începând de la poziția 0, primul coeficient nenul din polinomul  $P$ , iar apoi se caută posibile rădăcini întregi printre divizorii acestui coeficient.

Exemplu:

Fie polinomul  $P_3(X)=X^5-X^3$ . Sirul coeficienților asociat este  $a=(a_0, a_1, a_2, a_3, a_4, a_5)=(0, 0, 0, -1, 0, 1)$

Valoarea coeficientului  $a_0$  este 0, deci 0 este una dintre rădăcinile întregi ale polinomului  $P_3$ . Primul coeficient nenul, începând de la poziția 0, este  $a_3=-1$ . Dacă polinomul  $P_3$  mai are alte rădăcini întregi, atunci acestea sunt divizori ai coeficientului  $a_3$ . Numerele 1 și -1 sunt ambele rădăcini ale polinomului  $P_3$ .

Date: n, a

Rezultate: rad=(rad<sub>1</sub>, rad<sub>2</sub>, ..., rad<sub>nrRad</sub>), nrRad

Descriere: Determină rădăcinile întregi distincte ale polinomului  $P(X)=a_nX^n+a_{n-1}X^{n-1}+\dots+a_1X^1+a_0$  cu coeficienți întregi,  $P(\text{rad}_i)=0, \forall i \in \{1, \dots, nrRad\}$



Subalgoritmul RadaciniPolinom (n,a, rad, nrRad) este:

```
nrRad:=0; {la inceput numarul radacinilor intregi este 0}
i:=0;
    {cautam primul coeficient nenul}
Cattimp (( i<=n) AND (a[i]=0)) executa:
    i:=i+1;
sfCattimp
    {Daca a0=0, atunci i>0}
Daca i>0 atunci
    adaugaElem(rad, nrRad, 0); {adaugam 0 la sirul radacinilor}
SfDaca
    {Determinam divizorii coeficientului nenul ai}
Pentru d:=1,abs(a[i]),+1 executa
    Daca (a[i] mod d =0) atunci
        {Verificam daca divizorii sunt radacini ale polinomului}
        Daca ValPol(a,n,d)=0 atunci adaugaElem(rad, nrRad, d);
        Daca ValPol(a,n,-d)=0 atunci adaugaElem(rad, nrRad,-d);
    sfDaca
SfPentru
SfSubalgoritm
```

Date: a,n,e

Rezultate: a,n

Descriere: adauga elementul e, la sfarsitul vectorului a, cu n elemente

Subalgoritmul adaugaElem(a,n,e) este:

```
n:=n+1;
a[n]=e;
sfSubalgoritm
```

```
//Implementare program in C++
#include <iostream>
#include <cmath>
using namespace std;
typedef int sir[100];
void citestePolinom(sir a, int& n){
    cout<<" Gradul coeficientului ";
    cin>>n;
    cout<<" Coeficientii polinomului de la n la 0"<<endl;
    for(int i=n; i>=0;i--)
        cin>>a[i];
}
void tiparirePolinom(sir a, int n){
    cout<<"Sirul coeficientilor de la n la 0 este: "<<endl;
    for(int i=n;i>=0;i--)
        cout<<a[i]<<" ";
    cout<<endl;
}
```



```
}  
  
int ValPol(sir a, int n, int q){  
    int val=0;  
    int t=1;  
    for(int i=0;i<=n;i++){  
        val+=a[i]*t;  
        t=t*q;  
    }  
    return val;  
}  
  
int power(int x, int j){  
    int val=1;  
    for (int i=1;i<=j;i++)  
        val=val*x;  
    return val;  
}  
  
int ValPolRec(sir a, int n, int q, int t, int i){  
    if (i>n)  
        return 0;  
    else  
        return a[i]*t+ValPolRec(a,n,q,t*q,i+1);  
}  
  
int ValPolRec2(sir a, int n, int q, int t){  
    if ((q==0) || (n==0))  
        return a[0];  
    else  
        return a[n]*t+ValPolRec2(a,n-1,q, t/q);  
}  
  
int ValPolHorner(sir a, int n, int q){  
    int val=a[n];  
    for(int i=n-1; i>=0;i--)  
        val=val*q+a[i];  
    return val;  
}  
  
int ValPolHornerRecA(sir a, int n, int q, int i){  
    if (n==i)  
        return a[n];  
    else  
        return ValPolHornerRecA(a,n,q,i+1)*q+a[i];  
}  
  
int ValPolHornerRecB(sir a, int n, int q, int i){  
    if (i==0)  
        return a[n-i];  
    else  
        return ValPolHornerRecB(a,n,q,i-1)*q+a[n-i];  
}  
  
void adaugaElem(sir a, int &n, int e){  
    n=n+1;  
    a[n]=e;  
}
```



```
void radaciniPolinom(sir a, int n, sir rad, int& nrRad){
    int i=0;
    nrRad=0;
    while ((i<=n) && (a[i]==0))
        i++;
    if (i>0)
        adaugaElem(rad, nrRad, 0);
    int div=1;
    for(int div=1;div<=abs(a[i]); div++){
        if (ValPol(a,n,div)==0)
            adaugaElem(rad,nrRad,div);
        if (ValPol(a,n,div*(-1))==0)
            adaugaElem(rad,nrRad,-div);
    }
}

void printSir(sir a,int n){
    for(int i=1;i<=n;i++)
        cout<<a[i]<<" ";
    cout<<endl;
}

int main(){
    sir a, rad;
    int n,q,nrRad;
    citestePolinom(a,n);
    tiparirePolinom(a,n);
    cout<<"Introduceti valoarea q=";
    cin>>q;
    cout<<"Valoarea polinomului in punctul "<<q<<" folosind:"<<endl;
    cout<<"Varianta 1 - nerecursiva    -> P("<<q<<")=<<ValPol(a,n,q)<<endl;
    cout<<"Varianta 1 - recursiva a     -> P("<<q<<")=<<ValPolRec(a,n,q, 1, 0)<<endl;
    cout<<"Varianta 1 - recursiva b      -> P("<<q<<")=<<ValPolRec2(a,n,q,power(q,n))<<endl;
    cout<<"Varianta Horner - nerecursiva -> P("<<q<<")=<<ValPolHorner(a,n,q)<<endl;
    cout<<"Varianta Horner - recursiva a -> P("<<q<<")=<<ValPolHornerRecA(a,n,q,0)<<endl;
    cout<<"Varianta Horner - recursiva b -> P("<<q<<")=<<ValPolHornerRecB(a,n,q,n)<<endl;
    radaciniPolinom(a,n,rad,nrRad);
    if (nrRad==0)
        cout<<"Polinomul nu are radacini intregi"<<endl;
    else{
        cout<<"Radacinile intregi distincte sunt: ";
        printSir(rad,nrRad);
    }
    return 0;
}
```

{Implementare program in Pascal}



```
program Polinoame;
type vector=array[0..100] of integer;

procedure citestePolinom(var a:vector; var n:word);
var i:word;
begin
  write('Introduceti gradul polinomului ');
  readln(n);
  writeln('Introduceti coeficientii n->0');
  for i:=n downto 0 do
    readln(a[i]);
end;

procedure tiparirePolinom(a:vector; n:word);
var i:word;
begin
  writeln('Sirul coeficientilor de la n la 0 este:');
  for i:=n downto 0 do
    write(a[i], ' ');
  writeln;
end;

function ValPol(a:vector;n:word;q:integer):integer;
var i:word;
    val, t:integer;
begin
  val:=0;
  t:=1;
  for i:=0 to n do
    begin
      val:=val+a[i]*t;
      t:=t*q;
    end;
  ValPol:=val;
end;

function power(x:integer; n:word):integer;
var i:word;
    val:integer;
begin
  val:=1;
  for i:=1 to n do
    val:=val*x;
  power:=val;
end;

function ValPolRec(a:vector;n:word;q:integer;t:integer;i:word):integer;
begin
```





```
if (i>n) then
    ValPolRec:=0
else
    ValPolRec:=a[i]*t+ValPolRec(a,n,q,t*q,i+1);
end;

function ValPolRec2(a:vector;n:word;q:integer;t:integer):integer;
begin
    if (q=0) or (n=0) then
        ValPolRec2:=a[0]
    else
        ValPolRec2:=a[n]*t+ValPolRec2(a,n-1,q,t div q);
end;

function ValPolHorner(a:vector;n:word; q:integer):integer;
var val:integer;
    i:word;
begin
    val:=a[n];
    if (n>0) then
        for i:=n-1 downto 0 do
            val:=val*q+a[i];
        ValPolHorner:=val;
end;

function ValPolHornerRecA(a:vector;n:word; q:integer;i:word):integer;
begin
    if (n=i) then
        ValPolHornerRecA:=a[n]
    else
        ValPolHornerRecA:=ValPolHornerRecA(a,n,q,i+1)*q+a[i];
end;

function ValPolHornerRecB(a:vector;n:word; q:integer;i:word):integer;
begin
    if (i=0) then
        ValPolHornerRecB:=a[n-i]
    else
        ValPolHornerRecB:=ValPolHornerRecB(a,n,q,i-1)*q+a[n-i];
end;

procedure adaugaElem(var a:vector;var n:word; e:integer);
begin
    n:=n+1;
    a[n]:=e;
end;

procedure radaciniPolinom(a:vector; n:word; var rad:vector; var nrRad:word );
var i,d:integer;
```



```
begin
  i:=0;
  nrRad:=0;
  while(i<n) and (a[i]=0) do
    i:=i+1;
  if (i>0) then adaugaElem(rad,nrRad,0);
  for d:=1 to abs(a[i]) do
    begin
      if (a[i] mod d =0 ) then
        begin
          if ValPol(a,n,d)=0 then adaugaElem(rad,nrRad,d);
          if ValPol(a,n,-d)=0 then adaugaElem(rad,nrRad,-d);
        end;
      end;
    end;
end;

procedure tiparireSir(a:vector;n:word);
var i:word;
begin
  if n=0 then
    writeln('Sirul vid')
  else
    begin
      for i:=1 to n do
        write(a[i],' ');
      writeln;
    end;
end;

var a, rad:vector;
    q:integer;
    n,nrRad:word;
begin
  citestePolinom(a,n);
  tiparirePolinom(a,n);
  write('Introduceti valoarea q='); readln(q);
  writeln('Valoarea polinomul in punctul ',q,' folosind:');
  writeln('Varianta 1 - nerecursiva      -> P(',q,')=',ValPol(a,n,q));
  writeln('Varianta 1 - recursiva a     -> P(',q,')=',ValPolRec(a,n,q,1,0));
  writeln('Varianta 1 - recursiva b     -> P(',q,')=',ValPolRec2(a,n,q,power(q,n)));
  writeln('Varianta Horner - nerecursiva -> P(',q,')=',ValPolHorner(a,n,q));
  writeln('Varianta Horner - recursiva a -> P(',q,')=',ValPolHornerRecA(a,n,q,0));
  writeln('Varianta Horner - recursiva b -> P(',q,')=',ValPolHornerRecB(a,n,q,n));
  radaciniPolinom(a,n,rad,nrRad);
  if (nrRad=0) then
    writeln('Polinomul nu are radacini intregi.')
  else
    begin
      write('Radacinile intregi distincte sunt: ');
      tiparireSir(rad,nrRad);
    end;
end;
```



end.

5. Se consideră subalgoritmul  $f(n)$ , unde  $n$  este un număr natural ( $1 \leq n \leq 1000$ ) și  $A=(A_0, A_1, A_2, \dots, A_n)$  un șir de numere întregi:

```
Subalgoritmul  $f(A, n, q, i)$ :  
  Dacă  $n=i$  atunci  
    returnează  $A[n]$ ;  
  altfel  
    returnează  $f(A, n, q, i+1)*q+A[i]$ ;  
SfDaca  
SfSubalgoritm
```

Dacă  $n=3$ ,  $A=(-2, -1, 2, 1)$  și se apelează subalgoritmul  $f(A, n, q, 0)$  pentru ce valori ale lui  $q$ , subalgoritmul returnează valoarea 0?:

- a. {0, 2, -3}
- b. {1, -1, -2}
- c. {3, 1, -1, -2}
- d. {4, 0, 1, 5}