Tablouri bidimensionale (matrici)

Problema: suma matrice

Enunt

Sa se scrie un program care citeste de la tastatura un sir de matrici cu m linii si n coloane (1 <= n, m <= 100) cu elemente numere intregi, pana la citirea matricei nule. Programul va afisa suma matricelor citite.

Observatii:

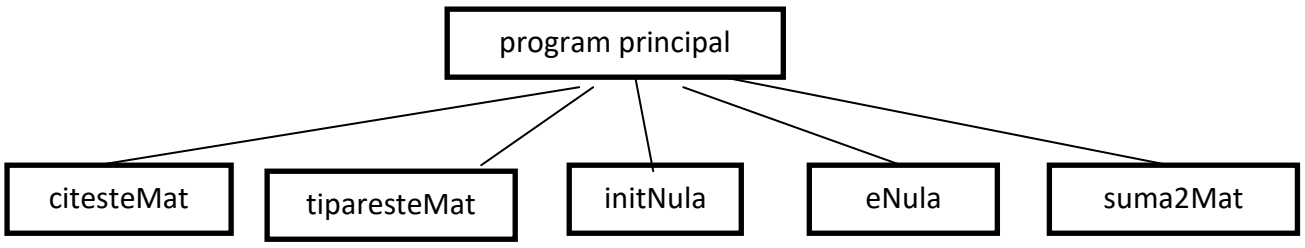
1. daca se citeste doar matricea nula rezultatul afisat va fi matricea nula
2. se presupune ca datele sunt corect introduse

Se cere să se utilizeze subprograme care să comunice între ele şi cu programul principal prin parametri. Fiecare subprogram trebuie specificat.

Pasii algorimului principal

Algoritm sumaMatrici
    @ citeste o matrice mat
    @ initializeaza suma cu matricea nula
    @ Cat Timp mat nu este nula executa
        @aduna mat la suma
        @ citeste inca o matrice in mat
    @sf.CatTimp
    @ tipareste suma
Sf.Algoritm

Identificarea subalgoritmilor



Programul

```
{       Nu s-a cerut ca indexarea sa se faca de la 0 sau de la 1
        Rezolvarea este data pentru matrici indexate de la unu     }
Program SumaMatrici;
const MAX = 100;
type Matrice=record
              m,n:integer;
              elem:array[1..MAX,1..MAX] of Integer;
       end;

{Descr: citeste o matrice
 In: -
 Out:  a - matricea citita}
```

```
Procedure citesteMat(Var a:Matrice );
Var i,j:integer;
Begin
   writeln('dati matricea: ');
   readln(a.m,a.n);
   for i:=1 to a.m do
      for j:=1 to a.n do
           read(a.elem[i,j]);
End;

{Descr: tipareste o matrice
 In:    a - matricea
 Out:  -
                    (se tipareste matricea) }
Procedure tiparesteMat(a:Matrice);
Var i,j:integer;
Begin
   for i:=1 to a.m do begin
      for j:=1 to a.n do
           write(a.elem[i,j]:5);
      writeln
   end
End;

{Desc:  aduna o matrice la o matrice data
 In:  a - matrice
             b - matrice
 Out: a - matricea suma}
Procedure suma2Mat(Var a:Matrice; b:Matrice);
Var i,j:integer;
Begin
   for i:=1 to a.m do
      for j:=1 to a.n do
           a.elem[i,j]:=a.elem[i,j]+b.elem[i,j];
End;

{Desc: verifica daca o matrice are toate elementele cu valoarea 0
 In:  a - matrice
 Out:  true daca toate elementele lui a au valoarea 0
       false in caz contrar                                          }
Function eNula(a:Matrice):Boolean;
Var i,j:integer;
    gasitValoareNenula:Boolean;
Begin
   gasitValoareNenula := false;
   i:=1;
   while ((i<=a.m) and (not gasitValoareNenula)) do begin
      j:=1;
         while ((j<=a.n) and (not gasitValoareNenula)) do begin
            if (a.elem[i,j]<>0) then gasitValoareNenula := true
              else j:=j+1;
         end;
         i:=i+1;
   end;
   eNula:= not gasitValoareNenula;
End;

{ alta versiune a functiei eNula care foloseste exit
Function eNula(a:Matrice):Boolean;
Var i,j:integer;
Begin
       eNula := true;
       for i:= 1 to a.m do
             for j:=1 to a.n do
                   if a.elem[i,j]<>0 then begin
```

```
                        eNula:=false;
                        exit;
                    end
End;
}

{Desc: initializeaza matricea nula
 In:         m - numarul de linii
             n - numarul de coloane
 Out:  a - matrice cu m linii si n coloane cu toate elementele 0  }
Procedure initNula(m:integer; n:integer; Var a:Matrice);
Var i,j:integer;
Begin
       a.m:=m;
       a.n:=n;
       for i:=1 to a.m do
              for j := 1 to a.n do
                     a.elem[i,j]:=0;
End;

{Program principal}
Var mat, suma: Matrice;

Begin
   citesteMat(mat);
   initNula(mat.m,mat.n,suma);
   while(not eNula(mat)) do begin
      suma2Mat(suma,mat);
      citesteMat(mat);
   end;
   tiparesteMat(suma);
End.
```

Exemple

Date de intrare	Rezultate
dati matricea: 2 2 1 2 3 4 dati matricea: 2 2 4 5 6 7 dati matricea: 2 2 0 0 0 0	5 7 9 11
dati matricea: 3 3 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0
dati matricea: 2 2 1 2 3 4 dati matricea: 2 2 -1 -2 -3 -4 dati matricea: 2 2 0 0 0 0	0 0 0 0

Problema: triunghi matrice

Enunt

Scrieti un subprogram care determina cate elemente numere prime se afla in triunghiul stang si cel drept al unei matrici patratice cu elemente numere naturale.

Date de intrare

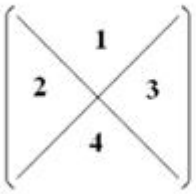
n : 3 <= n <= 100

a - matrice cu n linii si n coloane

Date de ieşire

numarul numerelor prime care se afla in triunghiul stang si cel drept al matricei a

Notă. Triunghiul stâng al matricei este cel marcat cu 2,
iar triunghiul drept este cel marcat cu 3 în figura de mai jos.



Nu se iau în considerare elementele de pe cele două diagonale.

Subprogramul

Obs: subprogramul apeleaza subprogramul ePrim
rezolvarea este data pentru matrici indexate de la unu

```
{ desc: verifica daca numarul nr este prim
  in: nr
   out: ePrim = true  daca nr e prim
                false in caz contrar }
Function ePrim(nr:integer):boolean;
Var divizor:integer;
Begin
      if nr<2 then
            ePrim:=false
      else begin
            ePrim := true;
            divizor:=2;
            while (divizor*divizor <= nr) do begin
                  if nr mod divizor = 0 then begin
                        ePrim:=false;
                        exit;
                  end;
                  divizor:=divizor+1
            end;
      end;
End;

{Desc: functia determina si returneaza cate elemente numere prime
                  se afla in triunghiul stang si cel drept
                  al unei matrici patratice cu elemente numere naturale.
 In:   n : 3 <= n <= 100
       a - matrice cu n linii si n coloane
 Out:  nrPrime -  numarul numerelor prime care se afla in triunghiul stang
                  si cel drept al matricei a                                  }
Function nrPrime(n:integer;a:Matrice):integer;
Var i,j:integer;
    nr:integer;
```

```
Begin
	nr:=0;
	for i:= 1 to n do
		for j:=1 to n do begin
			// daca ne aflam in triunghiul stang
			if (i>j) and (i+j<n+1) then
				if ePrim(a[i,j]) then nr:=nr+1;
			// daca ne aflam in triunghiul drept
			if (i<j) and (i+j>n+1) then
				if ePrim(a[i,j]) then nr:= nr+1;
		end;
	nrPrime:=nr;
End;
```

Exemple

Date de intrare	Rezultate
Matrice: 3 1 2 3 3 4 5 5 6 7	2
Matrice: 4 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7	2
Matrice: 3 1 2 3 4 5 6 7 8 9	0

Problema: elimina linii coloane impare

Enunt

Fie a o matrice cu m linii si n coloane (1 <= n, m <= 100) cu elemente numere intregi.
Scrieți un subprogram care elimina din matrice liniile si coloanele impare.
Matricea a si valorile lui m si n sunt atât parametri de intrare, cât și de ieșire pentru subalgoritm.

Subprogramul

Obs: Nu s-a cerut ca indexarea sa se faca de la 0 sau de la 1
rezolvarea este data pentru matrici indexate de la unu
rezultatele sunt diferite fata de o implementare care foloseste indexare de la 0

```
{ Desc: elimina din matrice liniile si coloanele impare
  IN:  m,n dimensiunile matricei
            a - matricea
  Out:      a contine numai liniile si coloanele pare din matricea originala
            m,n – noile dimensiuni ale matricei                              }
Procedure eliminaLiniiColoaneImpare(VAR m,n:integer; VAR a:Matrice);
Var i,j:integer;
Begin
      m:=m div 2;
      n:=n div 2;
      for i:=1 to m do
            for j:=1 to n do
                  a[i,j]:=a[2*i,2*j]
End;
```

Exemple

Date de intrare	Rezultate
Matrice: 3 3 1 2 3 4 5 6 7 8 9	5
Matrice: 2 2 10 11 12 13	13
Matrice: 1 5 1 2 3 4 5	(nu au ramas elemente dupa eliminare)

Problema: Sahara

Enunt

Datorită faptului că deşertul Sahara se extinde tot mai mult în fiecare an, statele lumii au hotărât să reducă acest proces. Chiar mai mult, specialiştii au demonstrat că la câţiva metri sub stratul de nisip se ascund zăcăminte importante de apă şi au hotărât împădurirea acestui deşert.
Se mai ştie că datorită condiţiilor climaterice extreme un copac nou plantat nu ar rezista decât dacă ar fi învecinat cu cel puţin alţi doi copaci existenţi. Deci, în procesul de împădurire, în fiecare zi se plantează copaci care pot rezista.

Pentru simplificare, să considerăm reprezentarea hărţii deşertului ca fiind o matrice cu m linii şi n coloane.
În fiecare căsuţă a matricei poate exista doar un singur copac.
Două poziţii se numesc vecine dacă ele sunt învecinate pe orizontală sau pe verticală.

Cerinţă
Scrieţi un subprogram care să determine dacă deşertul poate fi complet împădurit sau nu si in cate zile se termina procesul.

Date de intrare
m,n : 1 <= n, m <= 100
a - matricea cu m linii si n coloane
Date de ieşire
sePoate , nrZile
Daca se poate face impadurirea
sePoate – are valoarea true
si nrZile are valoarea egala cu numărul minim de zile în care se poate realiza împădurirea
Daca nu se poate face impadurirea
sePoate – are valoarea false
si nrZile are valoarea egala cu numărul minim de zile la care se opreşte împădurirea

Subprogramul

Obs: subprogramul apeleaza alte subprograme (pe care le-am implementat)
`rezolvarea este data pentru matrici indexate de la unu`

```
{nu e nevoie de subprogram pentru copiaza matrice daca lucram cu tipul Matrice
 copierea se poate face prin operatia de atribuire
 o varianta de implementare diferita de cea data in C++    }
{Desc.: determina numarul de vecini cu valoarea 1 ai elementului de pe o pozitie data
 In:     m,n : 1 <= n, m <= 100
                   matricea a cu m linii si n coloane
                   i,j : (i,j) - pozitie valida in matricea a
 Out:    nrVecini = nr. de vecini "ocupati de copaci" (cu valoarea 1) ai pozitiei (i,j)}
Function nrVecini(m,n:integer;a:matrice;i,j:integer):integer;
Var nr:integer;
Begin
   nr:=0;
   if (i-1>=1) then  {sus: nord}
      if (a[i-1,j]=1) then nr := nr + 1;
   if (j+1<=n) then  {dreapta: est}
      if (a[i,j+1]=1) then nr := nr + 1;
   if (i+1<=m) then  {jos: sud}
      if (a[i+1,j]=1) then nr := nr + 1;
   if (j-1>=1) then  {stanga: vest}
      if (a[i,j-1]=1) then nr := nr + 1;
   nrVecini := nr;
End;
```

```
{Desc.: verifica daca matricea contine doar elemente egale cu 1
             ( "desertul"este impadurit sau nu)
 In:     m,n : 1 <= n, m <= 100
         matricea a cu m linii si n coloane
 Out:    esteImpadurita = true daca "pe toate pozitiile se afla copaci" (toate elementele
                                  din matrice au valoarea 1)
                          false (in caz contrar)}
Function esteImpadurita(m,n:integer; a:matrice):boolean;
Var i,j:integer;
    rezultat:boolean;
begin
   rezultat := true;
   i:=1;
   while rezultat and (i<=m) do begin
      j:=1;
      while rezultat and (j<=n) do begin
         if a[i,j]=0 then rezultat:= false;
         j:=j+1;
      end;
      i:=i+1;
   end;
   esteImpadurita:=rezultat;
end;

{ o varianta de implementare care foloseste exit
Function esteImpadurita(m,n:integer; a:matrice):boolean;
Var i,j:integer;
begin
   for i:= 1 to m do begin
      for j:=1 to n do begin
         if a[i,j]=0 then begin exit(false) end;
      end;
   end;
   esteImpadurita:=true;
end;
}

{Desc: verifica daca desertul Sahara poate fi complet împadurit sau nu
 si determina nr de zile in care se poate face impadurirea
 Datele de intrare si de iesire sunt cele specificate in enunt}
Procedure impadurire(m,n:integer;a:matrice; Var sePoate:boolean; Var nrZile:integer);
Var i,j:integer;
    b:matrice;
    nrCopaci:integer;
Begin
   nrZile:=0;
   repeat
      nrCopaci:=0;
      b:=a;
      for i:=1 to m do
         for j:=1 to n do
            if (a[i,j]=0) and (nrVecini(m,n,b,i,j)>=2) then begin
               a[i,j]:= 1;
               nrCopaci := nrCopaci+1;
            end;
       nrZile := nrZile + 1;
   until (nrCopaci=0);
   nrZile:= nrZile -1;
   sePoate := esteImpadurita(m,n,a);
end;
```

Exemple:

Date de intrare	Rezultate
Dati matricea: 3 3 1 0 1 0 0 0 1 1 0	sePoate = true nrZile = 4
Dati matricea: 3 3 1 0 0 0 0 0 1 1 0	sePoate = false nrZile = 3