Tablouri bidimensionale (matrici)

Probleme de tip grila

1. Se da urmatoarea secventa de cod:

```
for (i = 0; i < n; i++){
      for (j = 0; j < n; j++)
            if (i > j)
                  cout << a[i][j] <<"  ";
      cout << endl;
}
```

Pentru o matrice a de dimensiuni n x n efectul secventei este:

a) De a afisa elementele din matrice de deasupra diagonalei principale
b) De a afisa elementele din matrice de sub diagonala principala (CORECT)
c) De a afisa elementele din matrice de pe diagonala principala
d) De a afisa elementele din matrice de sub diagonala secundara

2. Se da urmatoarea secventa de cod:

```
for (i = 0; i < n; i++){
          for (j = 0; j < n; j++)
                  if (j == n-i-1)
                    cout << a[i][j] << "   ";
          cout << endl;
        }
```

Pentru o matrice a de dimensiuni n x n efectul secventei este:

a) De a afisa elementele din matrice de deasupra diagonalei principale
b) De a afisa elementele din matrice de pe diagonala secundara (CORECT)
c) De a afisa elementele din matrice de pe diagonala principala
d) De a afisa elementele din matrice de sub diagonala secundara

3. Se da urmatoarea secventa de cod:

```
for (i = 0; i < n; i++)
            for (j =  ○ ; j < n; j++)
                  cout << a[i][j] << " ";
```

Numerotarea la matrice incepe de la 0, iar dimensiunea matricii este n x n. Cu ce se poate inlocui ○ astfel incat secventa de cod afiseze elementele din matrice de sub diagonala secundara:

a) n-i (CORECT)
b) n-i+1
c) i-n
d) i+1

4. Fie o matrice *a* patrata de dimensiune *n x n*. Se da urmatoare secventa de cod:

```
for(i = 0; i < n; i++){
      do{
       gasit = 0;
       for(j = 0; j < n - 1; j++)
         if(a[j][i] > a[j+1][i]){
            aux = a[j][i];
            a[j][i] = a[j+1][i];
            a[j+1][i] = aux;
            gasit = 1;
          }
      }
      while(gasit == 1);
    }
```

Precizati care ar fi efectul secventei de cod date asupra matricii:

a) Sorteaza intreaga multime de valori din matrice si le reordoneaza pe linii si coloane
b) Sorteaza doar liniile matricii
c) Sorteaza crescator doar coloanele matricii (CORECT)

Numere prime in spirala

Enunt

Sa se scrie un program care citeste de la tastatura dimensiunile n si m (n,m<=100) ale unei matrici (n x m). Programul va forma matricea din primele n x m numere prime consecutive aranjate in spirala: incepand de la stanga la dreapta, apoi de sus in jos, apoi de la dreapta la stanga si inapoi de jos in sus. La final programul va afisa matricea formata.

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

Exemplu

Date de intrare
n = 5
m = 7

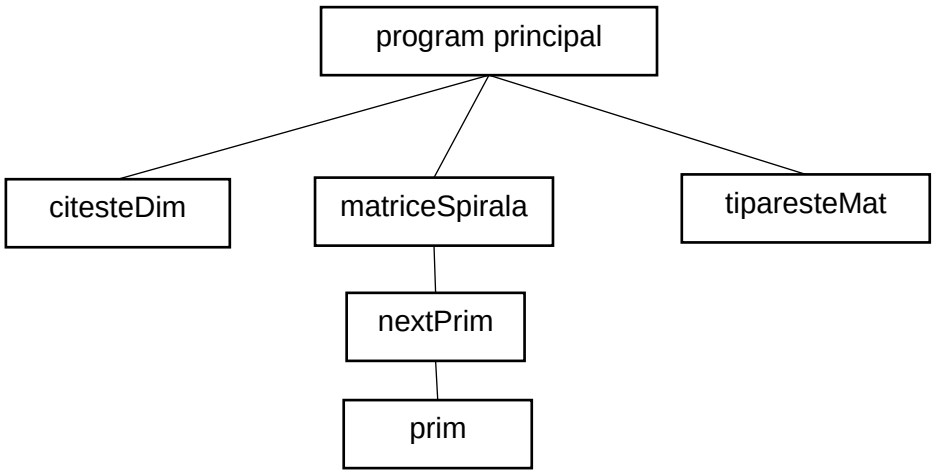
Date de iesire

```
 2   3   5   7  11  13  17
71  73  79  83  89  97  19
67 131 157 151 149 101  23
61 127 113 109 107 103  29
59  53  47  43  41  37  31
```

Pasii algorimului principal

Algoritm matriceSpirala
    @ citeste dimensiuni matrice
    @ formeaza matrice in spirala
    @ afiseaza matrice
Sf.Algoritm

Identificarea subalgoritmilor



Implementare C++

```
// Rezolvarea nu este optimizata pentru viteza de executie
// Rezolvarea exemplifica o abordare a problemei bazata pe descompunerea in subprobleme
// Programul a fost compilat cu Visual Studio Community 2015
#include <iostream>
#include "Matrice.h"
using namespace std;

void citesteDim(Matrice& a) {
	cout << "Introduceti dimensiunile matricii"<<endl;
	cout << "Linii=";
	cin >> a.n;
	cout << "Coloane=";
	cin >> a.m;
}

//verifica daca nr este prim
bool prim(int nr) {
	int i;
	if (nr == 2)
		return true;
	for (i = 2; i <= nr / 2; i++)
		if (nr%i == 0)
			return false;
	return true;
}

//returneaza cel mai mic numar prim strict mai mare ca nr
int nextPrim(int nr) {
	nr++;
	while (!prim(nr))
		nr++;
	return nr;
}

//formam in spirala matricea de numere prime consecutive
void matriceSpirala(Matrice& a) {
	int i, j;
	int nrPrim = 1;
	for (i = 0; i < (a.n / 2) + (a.n % 2); i++) {
		//parcurgem de la stanga la dreapta
		for (j = i; j < a.m - i; j++) {
			nrPrim = nextPrim(nrPrim);
			a.elem[i][j] = nrPrim;
		}

		//parcurgem de sus in jos
		for (j = 1 + i; j < a.n - i; j++) {
			nrPrim = nextPrim(nrPrim);
			a.elem[j][a.m - i - 1] = nrPrim;
		}

		//parcurgem de la dreapta la stanga
		for (j = a.m - i - 2; j >= i; j--) {
			nrPrim = nextPrim(nrPrim);
			a.elem[a.n - i - 1][j] = nrPrim;
		}
		//parcurgem de jos in sus
		for (j = a.n - i - 2; j >= i + 1; j--) {
			nrPrim = nextPrim(nrPrim);
			a.elem[j][i] = nrPrim;
		}
	}
}
```

```
int main() {
	Matrice a;
	citesteDim(a);
	matriceSpirala(a);
	afisare(a);
	return 0;
}
```

Implementare Pascal

```
// Rezolvarea nu este optimizata pentru viteza de executie
// Rezolvarea exemplifica o abordare a problemei bazata pe descompunerea in subprobleme
// Programul a fost compilat cu https://www.tutorialspoint.com/compile_pascal_online.php

Program MatriceSpirala;
const MAX = 100;

// Tipul de data matrice
type Matrice=record
	n,m:integer;
	elem:array[0..MAX,0..MAX] of Integer;
end;

Procedure citesteDim(Var a:Matrice);
begin
	writeln('introduceti dimensiunile matricii');
	readln(a.n);
	readln(a.m);
end;

//verifica daca nr este prim
Function prim(nr:Integer):Boolean;
Var i:Integer;
begin
	if nr = 2 then
	  begin
	       prim:=true;
	       exit;
	  end;
	for i:=2 to nr div 2 do
	      if nr mod i = 0 then
	             begin
	                    prim:=false;
	                    exit;
	             end;
	prim:=true;
end;
//returneaza cel mai mic numar prim strict mai mare ca nr
Function nextPrim(nr:Integer): Integer;
begin
	nr:= nr+1;
	while(prim(nr) = false) do
	       nr:= nr+1;
	  nextPrim:=nr;
end;
```

```
//formam in spirala matricea de numere prime consecutive
Procedure matriceSpirala(Var a:Matrice);
Var i,j,nrPrim:Integer;
begin
    nrPrim := 1;
       for i:=0 to (a.n div 2)+(a.n mod 2)-1 do
       begin
              //parcurgem de la stanga la dreapta
              for j:=i to a.m-i-1 do
              begin
                     nrPrim := nextPrim(nrPrim);
                     a.elem[i,j] := nrPrim;
              end;

              //parcurgem de sus in jos
              for j:=1+i to a.n-i-1 do
              begin
                     nrPrim := nextPrim(nrPrim);
                     a.elem[j,a.m-i-1] := nrPrim;
              end;

              //parcurgem de la dreapta la stanga
              for j:=a.m-i-2 downto i do
              begin
                     nrPrim := nextPrim(nrPrim);
                     a.elem[a.n-i-1,j] := nrPrim;
              end;
              //parcurgem de jos in sus
              for j:=a.n-i-2 downto i+1 do
              begin
                     nrPrim := nextPrim(nrPrim);
                     a.elem[j,i] := nrPrim;
              end;
       end;
end;

Procedure tiparesteMat(a:Matrice);
Var i,j:Integer;
begin
     for i:=0 to a.n-1 do
        begin
         for j:=0 to a.m-1 do
               write(a.elem[i,j]:5);
         writeln;
         end;
end;

{Program principal}
Var a:Matrice;
begin
      citesteDim(a);
      matriceSpirala(a);
      tiparesteMat (a);
end.
```

Sumă Matrici Rare

Enunț

O matrice A(n,m) cu elemente întregi se numește rară dacă majoritatea elementelor sale sunt egale cu 0. O matrice rară A(n,m) având k elemente nenule poate fi memorată folosind un șir X conținând k triplete de forma (linie, coloană, valoare) corespunzătoare valorilor nenule ale matricei – fără a folosi un tablou bidimensional.

Să se scrie un program care citește de la tastatură valorile n,m și două matrice rare A(n,m) și B(n,m), calculează sub forma unei matrice rare suma C(n,m) a celor două matrice A și B și afișează sub forma unui tablou bidimensional matricea C(n,m).
Citirea unei matrice se va face prin citirea numărului de linii (n), numărului de coloane (m) și citirea repetată a unor triplete (linie, coloană, valoare) – corespunzătoare valorilor nenule din matrice, până la citirea tripletului (-1,-1,-1). În cazul citirii mai multor triplete cu aceeași linie și coloană, se ia în considerare doar primul triplet citit.

Exemplu

De exemplu, pentru n=m=3, matricea A

0	**5**	**2**
0	**2**	**0**
2	**0**	**3**

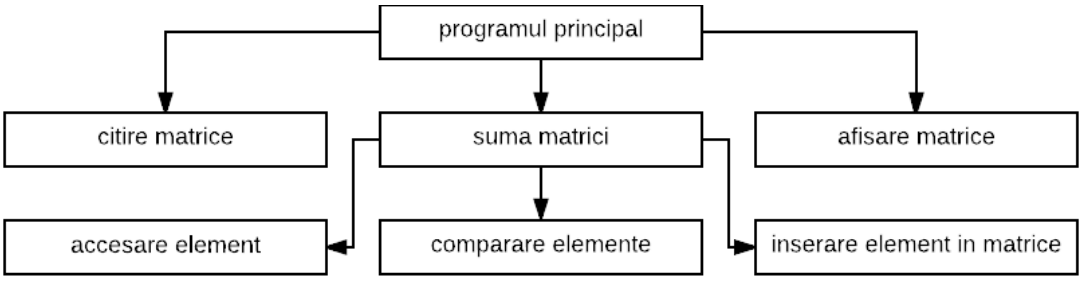
Figura 1 - Exemplu matrice rara

se va memora sub forma șirului X=((1,2,5), (1,3,2), (2,2,2), (3,1,2), (3,3,3))

Pasii algorimului principal

Algoritm Suma Matrici
    @ citire matrici
    @ determinare suma
    @ afisare matrice suma
Sf.Algoritm

Identificarea subalgoritmilor



Implementare C++

```
// Rezolvarea nu este optimizata pentru viteza de executie
// Rezolvarea exemplifica o abordare a problemei bazata pe descompunerea in subprobleme
// Programul a fost compilat cu Visual Studio Community 2015
#include <iostream>
#include <iomanip>
#include <stdio.h>

using namespace std;

#define MAX_N 100

struct Triplet {
	int linie, coloana, valoare;
};

struct MatriceRara {
	int nrLinii, nrColoane;
	int nrElemente;
	Triplet elemente[MAX_N];
};

int compara(Triplet t1, Triplet t2){
	if (t1.linie < t2.linie)
		return 1;
	if (t1.linie == t2.linie && t1.coloana < t2.coloana)
		return 1;
	return 0;
}

// Inserarea unui nou triplet in matricea rara
void inserare(MatriceRara &m, Triplet t){
	// daca matricea nu are elemente, noul triplet este primul
	if (m.nrElemente == 0)
	{
		m.elemente[0] = t;
		m.nrElemente = 1;
		return;
	}

	int i = m.nrElemente;
	while (i > 0 && compara(t, m.elemente[i - 1])) {
		m.elemente[i] = m.elemente[i - 1];
		i--;
	}
	// se insereaza tripletul
	m.elemente[i] = t;
	m.nrElemente++;
}

int element(MatriceRara &m, int linie, int coloana){
	for (int k = 0; k < m.nrElemente; k++)
		if (m.elemente[k].linie == linie && m.elemente[k].coloana == coloana)
			return m.elemente[k].valoare;
	// elementul implicit al unei matrici rare este 0
	return 0;
}
```

```
// se vor citi doar valorile nenule ale matricii
// subprogramul se terminla la citirea tripletului (-1,-1,-1)
void citire(MatriceRara &m){
	// presupunem datele valide
	std::cout << "Numarul de linii = ";
	std::cin >> m.nrLinii;
	std::cout << "Numarul de coloane = ";
	std::cin >> m.nrColoane;
	m.nrElemente = 0;

	Triplet t;
	std::cin >> t.linie >> t.coloana >> t.valoare;

	while (t.linie != -1 || t.coloana != -1 || t.valoare != -1) {
		if (element(m, t.linie, t.coloana) == 0)
			inserare(m, t);
		std::cin >> t.linie >> t.coloana >> t.valoare;
	}
}

// la calcularea sumei utilizam faptul ca sirul de triplete este ordonat 'lexicografic'
MatriceRara suma(MatriceRara& a, MatriceRara& b){
	MatriceRara c;
	// presupunem ca numarul de linii si coloane ale matricilor coincide
	c.nrColoane = a.nrColoane;
	c.nrLinii = a.nrLinii;
	c.nrElemente = 0;

	//indicii pentru tripleta curenta in cadrul matricilor a si b
	int i = 0;
	int j = 0;
	while (i < a.nrElemente && j < b.nrElemente) {
		if (compara(a.elemente[i], b.elemente[j]))
			c.elemente[c.nrElemente++] = a.elemente[i++];
		else
			if (compara(b.elemente[j], a.elemente[i]))
				c.elemente[c.nrElemente++] = b.elemente[j++];
			else {
				int s = a.elemente[i].valoare + b.elemente[j].valoare;
				//adaugam doar valorile nenule
				//este posibil ca sumarea sa rezulte intr-o valoare nula
				if (s != 0) {
					c.elemente[c.nrElemente] = a.elemente[i];
					c.elemente[c.nrElemente++].valoare = s;
				}
				i++;
				j++;
			}
	}
	// se copiaza tripletele ramase
	while (i < a.nrElemente)
		c.elemente[c.nrElemente++] = a.elemente[i++];

	while (j < b.nrElemente)
		c.elemente[c.nrElemente++] = b.elemente[j++];

	return c;
}
```

```
void tiparire(MatriceRara &m){
	cout << "Linii=" << m.nrLinii << ", Coloane=" << m.nrColoane << endl;
	for (int i = 1; i <= m.nrLinii; i++) {
		for (int j = 1; j <= m.nrColoane; j++)
			cout << setw(4) << element(m, i, j) << " ";
		cout << endl;
	}
}

void main(){
	MatriceRara a, b;
	citire(a);
	citire(b);
	tiparire(a);
	tiparire(b);
	MatriceRara c = suma(a, b);
	tiparire(c);
}
```

Implementare Pascal

```
type triplet = record
  linie, coloana, valoare : integer;
end;

type matricerara = record
  nrLinii,nrColoane,nrElemente : integer;
  elemente : array[0..100] of triplet;
end;

function compara(t1,t2:triplet) : integer;
begin
 if (t1.linie < t2.linie) then
   compara := 1
 else if (t1.linie = t2.linie) and (t1.coloana<t2.coloana) then
  compara := 1
 else compara := 0;
end;

// inserarea unui nou triplet in matricea rara
procedure inserare(m:matricerara;t:triplet);
var i:integer;
begin               // daca matricea nu are elemente, noul triplet este primul
if (m.nrElemente = 0) then
  begin
    m.elemente[0]:=t;
    m.nrElemente:=1;
  end;
  i := m.nrElemente;
  while (i>0) and (compara(t,m.elemente[i-1])=1) do
  begin
    m.elemente[i]:=m.elemente[i-1];
    dec(i);
  end;
  m.elemente[i] := t;   // se insereaza tripletul
  inc(m.nrElemente);
end;
```

```
function element(m:matricerara;linie,coloana:integer) : integer;
var aux,k:integer;
begin
  aux:=0;
  for k:=0 to m.nrElemente-1 do
    if (m.elemente[k].linie = linie) and (m.elemente[k].coloana=coloana) then
      aux := m.elemente[k].valoare;
  element := aux;
end;

procedure citire(var m:matricerara);
var t:triplet;
begin
  // presupunem datele de intrare valide
  write('Numarul de linii=');
  readln(m.nrLinii);
  write('Numarul de coloane=');
  readln(m.nrColoane);
  m.nrElemente := 0;

  write('triplet>');
  readln(t.linie,t.coloana,t.valoare);
  while (t.linie<>-1) or (t.coloana<>-1) or (t.valoare<>-1) do
  begin
    if element(m,t.linie,t.coloana) = 0 then
      inserare(m,t);

  write('triplet>');
  readln(t.linie,t.coloana,t.valoare);
  end;
end;

// la calcularea sumei utilizam faptul ca sirul de triplete
// este ordonat 'lexicografic'
function suma(a,b:matricerara) : matricerara;
var c:matricerara; s,i,j:integer;
begin
  // presupunem ca numarul de linii si coloane al matricilor coincide
  c.nrColoane := a.nrColoane;
  c.nrLinii := a.nrLinii;
  c.nrElemente := 0;

  // indicii pentru tripleta curenta in cadrul matricilor a si b
  i := 0;
  j := 0;
  while (i<a.nrElemente) and (j<b.nrElemente) do
  begin
    if compara(a.elemente[i],b.elemente[j])<>0 then
      begin
      c.elemente[c.nrElemente] := a.elemente[i];
      inc(c.nrElemente); inc (i);
      end else
      if compara(b.elemente[j],a.elemente[i])<>0 then
      begin
```

```
      c.elemente[c.nrElemente] := b.elemente[j];
      inc(c.nrElemente); inc(j);
      end else
        begin
        s := a.elemente[i].valoare + b.elemente[j].valoare;
        // adaugam doar valorile nenule
        if s<>0 then
          begin
            c.elemente[c.nrElemente] := a.elemente[i];
            c.elemente[c.nrElemente].valoare := s;
            inc(c.nrElemente);
          end;
        inc(i); inc(j);
        end;
  end;

  // se copiaza valorile ramase
  while i<a.nrElemente do
    begin
    c.elemente[c.nrElemente] := a.elemente[i];
    inc(c.nrElemente); inc(i);
    end;
  while j<b.nrElemente do
    begin
    c.elemente[c.nrElemente] := b.elemente[j];
    inc(c.nrElemente); inc(j);
    end;
suma := c;
end;

procedure tiparire(m:matricerara);
var i,j:integer;
begin
  writeln('Linii=',m.nrLinii,', Coloane=',m.nrColoane);
  for i:=1 to m.nrLinii-1 do
  begin
    for j:=1 to m.nrColoane-1 do
      write(element(m,i,j),' ');
  writeln();
  end;
end;

var a,b,c : matricerara;
begin
  citire(a);
  citire(b);
  tiparire(a);
  tiparire(b);
  c := suma(a,b);
  tiparire(c);
end.
```

Planul casei

Enunt

Părinții Corinei au cumparat o casă nouă și la cumpărare au primit planul casei. Corina și-a propus ca, înainte să vadă casa, să ghicească din plan care este cea mai mare încăpere din casă.

Scrieți un program care determină aria maximă a unei încăperi din casă.

Date de intrare

n, m: 1<= n,m <= 100

a – matricea cu n linii si m coloane reprezentând planul casei astfel:

- valoarea 0 pentru pereți
- valoarea -1 pentru spațiu gol (unde nu e perete)

Date de ieşire

Aria maximă a unei încăperi din casă. Prin încăpere înțelegem spațiu gol înconjurat de perete (delimitat de valori 0).

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

Intrebare suplimentara*:

Se schimba complexitatea daca adaugam in plus restrictia ca toate camerele sa fie convexe? Modificati corespunzator algoritmul.

Exemplu

Date de intrare

n = 6, m = 7

-1 -1 0 -1 -1 0 -1
-1 -1 0 -1 -1 0 -1
-1 -1 0 -1 -1 -1 -1
0 0 0 0 0 0 0
-1 -1 0 -1 -1 -1 -1
-1 -1 0 -1 -1 -1 -1

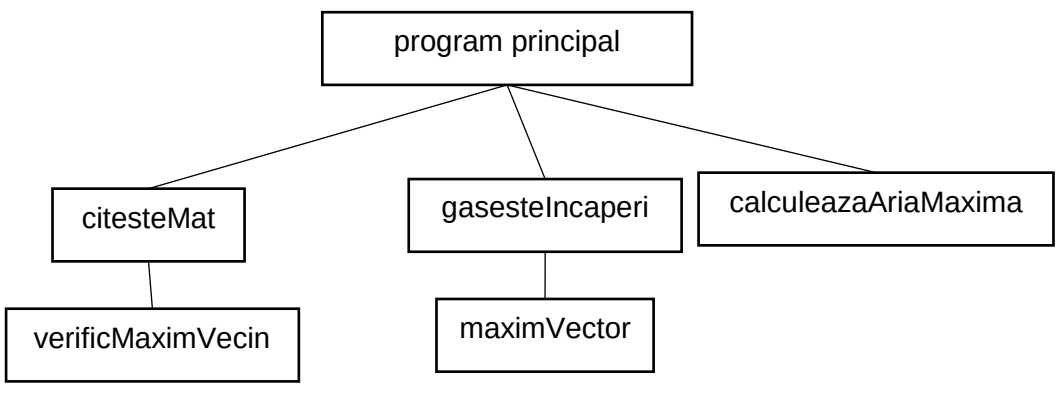
Date de iesire

10 (aria maximă a încăperii din colțul dreapta sus)

Pasii algorimului principal

Algoritm matriceSpirala
@ citeste matrice
@ identifica incaperi
@ calculeaza arii pentru incaperi
@ determina aria maxima
@ afiseaza aria maxima
Sf.Algoritm

Identificarea subalgoritmilor



Implementare nerecursivă C++

```
// Rezolvarea nu este optimizata pentru viteza de executie
// Rezolvarea exemplifica o abordare a problemei bazata pe descompunerea in subprobleme
// Programul a fost compilat cu Visual Studio Community 2015
#include <iostream>
#include "Matrice.h"
using namespace std;

//verific daca valoare vreunui vecin este >0 si o returnez.
//Inseamna ca e o camera deja detectata.
int verificMaximVecin(Matrice a, int i, int j) {
	int max = -1;

	//daca am un vecin in directia respectiva si nu e perete
	if (i > 0 && a.elem[i - 1][j] != 0)
		max = a.elem[i - 1][j];

	if (i < a.n - 1 && a.elem[i + 1][j] != 0)
		if (a.elem[i + 1][j] > max)
			max = a.elem[i + 1][j];

	if (j > 0 && a.elem[i][j - 1] != 0)
		if (a.elem[i][j - 1] > max)
			max = a.elem[i][j - 1];

	if (j < a.m - 1 && a.elem[i][j + 1] != 0)
		if (a.elem[i][j + 1] > max)
			max = a.elem[i][j + 1];

	return max;
}

//returneaza true daca au mai fost schimbari
bool gasesteIncaperi(Matrice& a, int& contorIncaperi) {
	int i, j;

	int max;
	bool schimbari = false;

	for (i = 0; i < a.n; i++)
		for (j = 0; j < a.m; j++) {
			if (a.elem[i][j] != 0) {
				max = verificMaximVecin(a, i, j);
				//daca minimul e -1, atunci e o incapere inca nedescoperita
				if (max == -1) {
					contorIncaperi++;
					a.elem[i][j] = contorIncaperi;
					schimbari = true;
				}
				//altfel, e o camera detectata deja si completez cu numarul ei
				//iar daca cumva are mai multe numere, il aleg pe cel mai mare
				else
					if (a.elem[i][j] != max) {
						a.elem[i][j] = max;
						schimbari = true;
					}
			}
		}
	return schimbari;
}
```

```
//returneaza maximul de pe primele l pozitii din vectorul v
int maximVector(int v[], int l) {
	int max = 0;
	for (int i = 0; i < l; i++)
		if (v[i] > max)
			max = v[i];
	return max;
}

int calculeazaAriaMaxima(Matrice a, int contorIncaperi) {
	int ariiCamere[200];
	int i, j;

	//initializez toate ariile cu 0;
	for (i = 0; i < contorIncaperi; i++)
		ariiCamere[i] = 0;

	for (i = 0; i < a.n; i++)
		for (j = 0; j < a.m; j++) {
			int idCamera = a.elem[i][j];
			//daca e Camera si nu perete ii cresc cu 1 aria
			if (idCamera > 0)
				ariiCamere[idCamera - 1]++;
		}
	return maximVector(ariiCamere, contorIncaperi);
}

int main() {
	Matrice a = citire("3.in");
	afisare(a);

	bool schimbari = true;
	int contorIncaperi = 0;

	//Cat timp mai sunt schimbari nu putem fi siguri ca o camera e umpluta cu acelasi
	//numar, se poate sa nu fi fost detectata din prima parcurgere ca o singura incapere.
	//De aceea parcurgem de mai multe ori si daca detectam numere diferite in aceeasi
	//incapere le suprascriem cu cel mai mare dintre cele intalnite
	while (schimbari)
		schimbari = gasesteIncaperi(a, contorIncaperi);

	int aria = calculeazaAriaMaxima(a, contorIncaperi);
	cout << "Aria maxima a unei incaperi este: " << aria << endl;
	return 0;
}
```

Implementare nerecursivă Pascal

```
// Rezolvarea nu este optimizata pentru viteza de executie
// Rezolvarea exemplifica o abordare a problemei bazata pe descompunerea in subprobleme
// Programul a fost compilat cu https://www.tutorialspoint.com/compile_pascal_online.php

Program PlanCasa;
const MAX = 100;

//tipul de data matrice
type Matrice=record
	n,m:integer;
	elem:array[0..MAX,0..MAX] of Integer;
end;

Type
     vector = array [0..MAX] of integer;
```

```
Procedure citireMatrice(Var a:Matrice);
Var i,j:Integer;
begin
    write('n=');
    readln(a.n);
    write('m=');
    readln(a.m);
    for i:=0 to a.n-1 do
    begin
          for j:=0 to a.m-1 do
          begin
              write('a[',i,'][',j,']=');
              readln(a.elem[i,j]);
           end;
    end;
end;

Procedure afisareMatrice(a : Matrice);
Var i,j:Integer;
begin
    for i:=0 to a.n-1 do
    begin
       for j:=0 to a.m-1 do
            write(a.elem[i,j],' ');
        writeln;
    end;
end;

//verific daca valoare vreunui vecin este >0 si o returnez.
//Inseamna ca e o camera deja detectata.
Function verificMaximVecin(a:Matrice; i:Integer; j:Integer) : Integer;
Var max:Integer;
begin
       max :=-1 ;

    //daca am un vecin in directia respectiva si nu e perete
    if(i>0) and (a.elem[i-1,j]<>0) then
       max := a.elem[i-1,j];

    if(i<a.n-1) and (a.elem[i+1,j]<>0) then
       if(a.elem[i+1,j]>max) then
            max := a.elem[i+1,j];

       if(j>0) and (a.elem[i,j-1]<>0) then
       if(a.elem[i,j-1]>max) then
            max := a.elem[i,j-1];

    if(j<a.m-1) and (a.elem[i,j+1]<>0) then
       if(a.elem[i,j+1]>max) then
            max := a.elem[i,j+1];

    verificMaximVecin := max;
end;
```

```
//returneaza true daca au mai fost schimbari
Function gasesteIncaperi(Var a:Matrice; Var contorIncaperi:Integer) : Boolean;
Var i, j, max:Integer;
Var schimbari:Boolean;
begin
       schimbari := false;

       for i:=0 to a.n-1 do
          for j:=0 to a.m-1 do
          begin
             if(a.elem[i,j] <> 0) then
              begin
                  max := verificMaximVecin(a,i,j);
                  //daca minimul e -1, atunci e o incapere inca nedescoperita
                  if(max = -1) then
                  begin
                      contorIncaperi := contorIncaperi+1;
                      a.elem[i,j] := contorIncaperi;
                     schimbari := true;
                  end
                  //altfel, e o camera detectata deja si completez cu numarul ei
                  //iar daca cumva are mai multe numere, il aleg pe cel mai mare
                  else
                      if(a.elem[i,j] <> max) then
                      begin
                                          a.elem[i,j] := max;
                          schimbari := true;
                      end;
              end;
          end;
    gasesteIncaperi := schimbari;
end;


//returneaza maximul de pe primele l pozitii din vectorul v
Function maximVector( v:Vector; l:Integer): Integer;
Var max,i:Integer;
begin
    max := 0;
    for i := 0 to l-1 do
       if(v[i]>max) then
           max := v[i];
    maximVector := max;
end;

Function calculeazaAriaMaxima(a : Matrice; contorIncaperi:Integer) :Integer;
Var ariiCamere: array[0..MAX] of Integer;
Var i,j,idCamera:Integer;
begin

    //initializez toate ariile cu 0;
    for i:=0 to contorIncaperi-1 do
        ariiCamere[i]:=0;

    for i:=0 to a.n-1 do
        for j:=0 to a.m-1 do
        begin
           idCamera := a.elem[i,j];
           //daca e Camera si nu perete ii cresc cu 1 aria
           if(idCamera > 0) then
              ariiCamere[idCamera-1] := ariiCamere[idCamera-1] +1;
        end;

    calculeazaAriaMaxima := maximVector(ariiCamere, contorIncaperi);
end;
```

```
{Program principal}
Var a:Matrice;
Var schimbari:Boolean;
Var contorIncaperi, aria:Integer;
begin
    citireMatrice(a);
    schimbari := true;
    contorIncaperi := 0;
    //Cat timp mai sunt schimbari nu putem fi siguri ca o camera e umpluta cu acelasi
    //numar, se poate sa nu fi fost detectata din prima parcurgere ca o singura incapere.
    //De aceea parcurgem de mai multe ori si daca detectam numere diferite in aceeasi
    //incapere le suprascriem cu cel mai mare dintre cele intalnite
    while(schimbari = true) do
       schimbari :=  gasesteIncaperi(a, contorIncaperi);

    aria := calculeazaAriaMaxima(a,contorIncaperi);
    writeln;
    writeln('Aria maxima a unei incaperi este: ',aria);
end.
```

Implementare recursivă C++

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

const int MAX = 200;

struct Matrice {
	int m;
	int n;
	int elem[MAX][MAX];
};

void afisare(Matrice m);
Matrice citire(char*);

void afisare(Matrice m) {
	cout << "linii=" << m.n << ", coloane=" << m.m << endl;
	for (int i = 0; i < m.n; i++) {
		for (int j = 0; j < m.m; j++) {
			cout << setw(4) << m.elem[i][j];
		}
		cout << endl;
	}
}

Matrice citire(char* fisier) {
	FILE *fin;
	Matrice m;

	fopen_s(&fin, fisier, "r");
	fscanf_s(fin, "%d", &m.n);
	fscanf_s(fin, "%d", &m.m);

	int v;
	for (int i = 0; i < m.n; i++) {
		for (int j = 0; j < m.m; j++) {
			fscanf_s(fin, "%d ", &m.elem[i][j]);
		}
	}
	return m;
}
```

```
// Umplem casutele legate ale matricii m cu valoarea '1', incepand cu pozitia (l,c)
int umplere(Matrice& m, int l, int c) {
	// am iesit din matrice
	if (l < 0 || l >= m.n|| c < 0 || c >= m.m) return 0;

	// am dat de un perete, sau o camera deja detectata
	if (m.elem[l][c] != -1) {
		return 0;
	}

	// marchez locatia, apoi verific recursiv vecinii
	m.elem[l][c] = 1;
	return 1 + umplere(m, l - 1, c) + umplere(m, l + 1, c) + umplere(m, l, c - 1) +
umplere(m, l, c + 1);
}

int cameraMaxima(Matrice casa) {
	int cameraMaxima = 0;
	for (int l=0;l<casa.n;l++)
		for (int c = 0; c < casa.m; c++) {
			int v = umplere(casa, l, c);
			if (v > cameraMaxima) {
				cameraMaxima = v;
			}
		}
	return cameraMaxima;
}

void main() {
	Matrice casa = citire("3a.in");
	cout << "Dimensiunea camerei maxime: " << cameraMaxima(casa);
}
```

Implementare recursivă Pascal

```
// definim tipul de date matrice
type
  matrice = record
  elem:array[0..100,0..100] of integer;
  n,m : integer;
end;

// citim datele de intrare
function readfile(s : string) : matrice;
var f:text; i,j,val:integer;
m : matrice;
begin
   assign(f,'plancasa.in');
   reset(f);
   read(f,m.n);
   read(f,m.m);
   for i:=0 to m.n-1 do
     for j:=0 to m.m-1 do
       read(f,m.elem[i][j]);
   close(f);
   readFile := m;
end;
```

```
// functia recursiva de umplere
function umplere(var m : matrice; l,c:integer) : integer;
begin
  if (l<0) or (l>=m.n ) or (c<0) or (c>=m.m) then
    umplere := 0
    else begin
    if m.elem[l][c] <> -1 then
      umplere :=0
      else begin
      m.elem[l][c] := 1;
      umplere := 1 + umplere(m,l-1,c) + umplere(m,l+1,c) + umplere(m,l,c-1)
      + umplere(m,l,c+1);
      end;
    end;
end;

// functia unde determinam dimensiunea camerei maxime
function cameraMaxima(casa:matrice) : integer;
var l,c,v,cameraMax : integer;
begin
  cameraMax := 0;
  for l:=0 to casa.n-1 do
    for c:=0 to casa.m-1 do
    begin
     v := umplere(casa,l,c);
     if v>cameraMax then cameraMax := v;
    end;
  cameraMaxima := cameraMax;
end;

var m : matrice;
begin
  m:=readfile('plancasa.in');
  writeln('Camera cea mai mare are dimensiunea ',cameraMaxima(m));
end.
```