7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

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 = 0; j < n; j++)
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
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)

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

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:

- daca se citeste doar matricea nula rezultatul afisat va fi matricea nula
- 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.

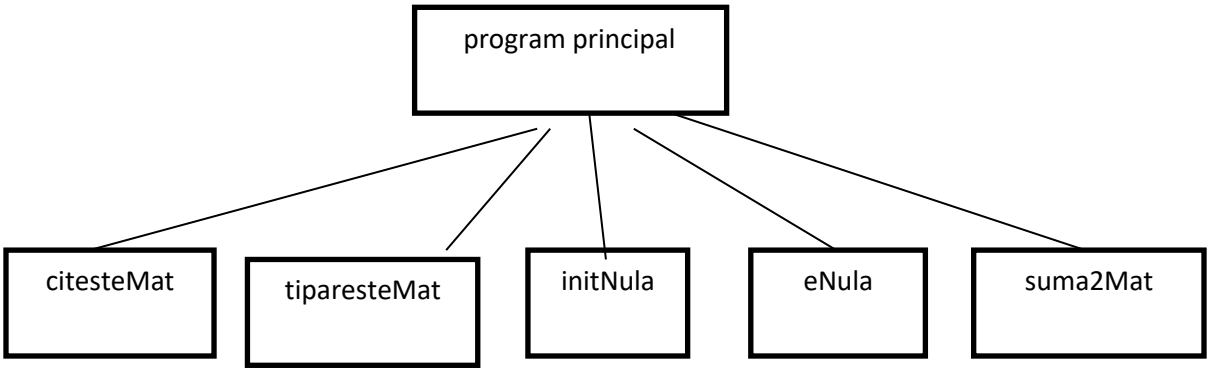
Pasii algoritmului principal

```
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

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar



Programul

```
// Rezolvarea este data pentru matrici indexate de la zero
// Programul a fost compilat cu Visual Studio Community 2017

#include <iostream>
using namespace std;
const int MAX = 100;

// Tipul de data matrice
struct Matrice {
	int m;
	int n;
	int elem[MAX][MAX];
};

// Citire matrice
void citesteMat(Matrice& a) {
	int i, j;
	cout << "Dati matricea:\n";
	cin >> a.m >> a.n;
	for (i = 0; i < a.m; i++)
		for (j = 0; j < a.n; j++)
			cin >> a.elem[i][j];
}

// Afisare matrice
void tiparesteMat(Matrice a) {
	for (int i = 0; i < a.m; i++) {
		for (int j = 0; j < a.n; j++)
			cout << a.elem[i][j] << " ";
		cout << "\n";
	}
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
}

// Aduna o matrice la o matrice data
void suma2Mat(Matrice & a, Matrice b) {
	for (int i = 0; i < a.m; i++)
		for (int j = 0; j < a.n; j++)
			a.elem[i][j] = a.elem[i][j] + b.elem[i][j];
}

// Verifica daca o matrice are toate elementele cu valoarea 0
bool eNula(Matrice a) {
	for (int i = 0; i < a.m; i++)
		for (int j = 0; j < a.n; j++)
			if (a.elem[i][j] != 0) return false;
	return true;
}

// Initializeaza matricea nula
void initNula(int m, int n, Matrice& a) {
	a.m = m;
	a.n = n;
	int i, j;
	for (i = 0; i < a.m; i++)
		for (j = 0; j < a.n; j++)
			a.elem[i][j] = 0;
}

int main() {
	Matrice mat, suma;

	citesteMat(mat);
	initNula(mat.m, mat.n, suma);

	while (!eNula(mat)) {
		suma2Mat(suma, mat);
		citesteMat(mat);
	}

	tiparesteMat(suma);

	return 0;
}
```

Exemple

Date de intrare	Rezultate
Dati matricea: 2 2 1 2 3 4 Dati matricea:	 5 7 9 11

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

2 2 4 5 6 7 Dati matricea: 2 2 0 0 0 0	
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

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

Sah[1]

Enunt

Se considera o tabla de sah cu `n+1` linii si `2n+1` coloane. Pe prima linie patratul din mijloc contine 1 gram de fan, iar celelalte patrate de pe prima linie nu contin nimic. Incepand cu linia a doua fiecare patrat contine o cantitate de fan obtinuta prin adunarea cantitatilor de fan din cele 3 patrate ale liniei anterioare cu care se invecineaza (pe verticala si diagonala).

De exemplu, daca `n=3` tabla are 4 linii, 7 coloane si urmatoarea configuratie.

			1			
		1	**1**	**1**		
	1	**2**	**3**	**2**	**1**	
1	**3**	**6**	**7**	**6**	**3**	**1**

Un cal pleaca de pe prima linie, de pe o coloana `k<=n`, sare din orice pozitie `(i,j)` in pozitia `(i+1,j+2)` atat timp cat este posibil si mananca tot fanul din patratele prin care trece. De exemplu, pentru `n=3` și `k=1`, patratele prin care trece calul sunt (0, 1), (1, 3) si (2, 5) iar cantitatea totala de fan este 0 + 1 + 1 = 2.

Cerinte

1. Determinati continutul matricii daca se citeste valoarea lui n.
2. Calculati cantitatea totala de fan de pe linia k a tablei de sah (se cunosc valorile lui n si k).
3. Calculati cate grame de fan mananca un cal care pleaca de pe prima linie, coloana k (se cunosc valorile lui n si k).

Date de intrare

Se citesc valorile n si k, 0<= n <=100, 0 <= k <= 200

Date de ieşire

- Matricea de dimensiuni n+1, 2n+1
- Cantitatea totala de fan de pe linia k

[1] Enunt adaptat pornind de la OJI 2015

7 decembrie 2019 Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

- Cantitatea de fan consumata de un cal care pleaca din linia 0, coloana k

Exemplu

Date de intrare	Date de iesire
n = 3	Matricea: 0 0 0 1 0 0 0 0 0 1 1 1 0 0 0 1 2 3 2 1 0 1 3 6 7 6 3 1
k = 3	Cantitate linia k : 27
k = 0	Cantitate consumata de cal: 4

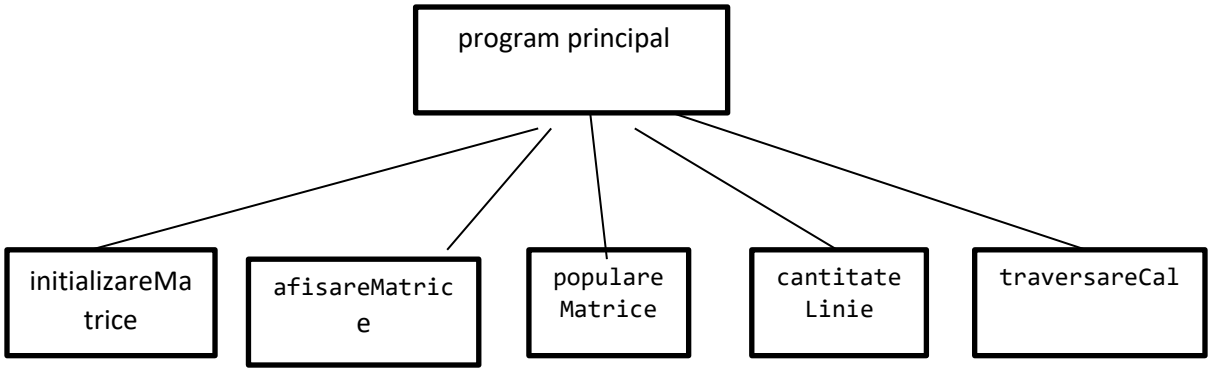
Pasii algoritmului principal

Algoritm tablaSah
    @ citeste un numar n
    @ genereaza matricea
    @ tipareste matricea
    @ citeste un numar k
    @ calculeaza cantitatea de pe linia k
    @ tipareste cantitatea
    @ citeste un numar k
    @ determina deplasarea calului si calculeaza cantitatea consumata
    @ tipareste cantitatea
Sf.Algoritm

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

Identificarea subalgoritmilor



Programul

```
// 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 2017

#include <iostream>
using namespace std;

typedef struct {
	int n;
	int elem[101][201];
} Matrice;

// Initial, toate elementele sunt 0
void initializareMatrice(Matrice &a) {
	for (int i = 0; i < a.n + 1; i++) {
		for (int j = 0; j < 2 * a.n + 1; j++) {
			a.elem[i][j] = 0;
		}
	}
}

// Tiparire matrice pe ecran
void afisareMatrice(Matrice &a) {
	for (int i = 0; i < a.n + 1; i++) {
		for (int j = 0; j < 2*a.n + 1; j++)
			cout << a.elem[i][j] << " ";
		cout << endl;
	}
	cout << endl;
}
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
// Populatea matricii cu valori conform regulii date
void populareMatrice(Matrice &a) {
	a.elem[0][a.n] = 1;
	for (int i = 1; i < a.n + 1; i++) {
		// primul si ultimul element de pe linie trebuie tratate separat
		a.elem[i][0] = a.elem[i - 1][0] + a.elem[i - 1][1];
		a.elem[i][2 * a.n] = a.elem[i - 1][2 * a.n - 1] + a.elem[i - 1][2 * a.n];
		// calcul valoare elemente pe baza insumarii a 3 elemente de pe linia
precedenta
		for (int j = 1; j < 2 * a.n; j++) {
		  a.elem[i][j] = a.elem[i - 1][j - 1] + a.elem[i - 1][j] + a.elem[i - 1][j
+ 1];
		}
	}
}

// Calcul suma elementelor de pe linia k pentru matricea populata
int cantitateLinie_v1(Matrice &a, int k) {
	int cantitate = 0;
	for (int j = 0; j < 2 * a.n + 1; j++) {
		cantitate += a.elem[k][j];
	}
	return cantitate;
}

// Returneaza 3^k
// Aceasta functie este necesara pentru a nu folosi functii din biblioteci suplimentare
(lucru // cerut in conditii de examen); inlocuieste asadar apelul functiei pow(3, k).
int putere(int k)
{
    int p = 1;
    for (int i = 0; i < k; i++)
        p = p * 3;
    return p;
}

// Calcul suma elementelor tinand cont de faptul ca suma de pe o linie este tripla fata
de suma liniei anterioare.
// Suma primei linii este 1, deducem ca suma liniei k este 3*3*...*3 (de k ori), unde k =
0 pentru prima linie.
int cantitateLinie_v2(Matrice &a, int k) {
	return putere(k);
}

// Calcul cantitate totala acumulata prin traversarea matricii pornind din linia 0,
coloana k
// conform regulii calului pe o tabla de sah
int traversareCal(Matrice &a, int k) {
	int suma = 0;
	int i = 0;
	int j = k;
	while (i < a.n + 1 && j < 2 * a.n + 1) {
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
            suma += a.elem[i][j];
            i = i + 1;
            j = j + 2;
        }
        return suma;
}

void main() {
        Matrice a;
        cout << "n = ";
        cin >> a.n;

        initializareMatrice(a);

        populareMatrice(a);
        afisareMatrice(a);

        int k = -1;
        cout << "Introduceti k = ";
        cin >> k;
        cout << "Cantitate linia  " << k << " este (v1) " << cantitateLinie_v1(a, k) <<
endl;
        cout << "Cantitate linia  " << k << " este (v2) " << cantitateLinie_v2(a, k) <<
endl;

        cout << "Introduceti coloana initiala cal = ";
        cin >> k;
        cout << "Cantitatea adunata de cal este " << traversareCal(a, k);
}
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

Numere prime

Enunt

Sa se scrie un program care citeste de la tastatura dimensiunea n a unei matrici (n x n). Programul va umple apoi toate pozițiile din matrice cu numere prime consecutive. Programul va forma din matrice un șir după următoarele regulă: prima linie, apoi prima coloană urmate de a doua linie și coloană șamd. Elementele vor apărea numai o dată în șirul format. La final programul va afisa matricea initiala impreuna cu șirul format.

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	Date de iesire
n = 5	3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 3 5 7 11 13 17 37 59 79 19 23 29 31 41 61 83 43 47 53 67 89 71 73 97 101

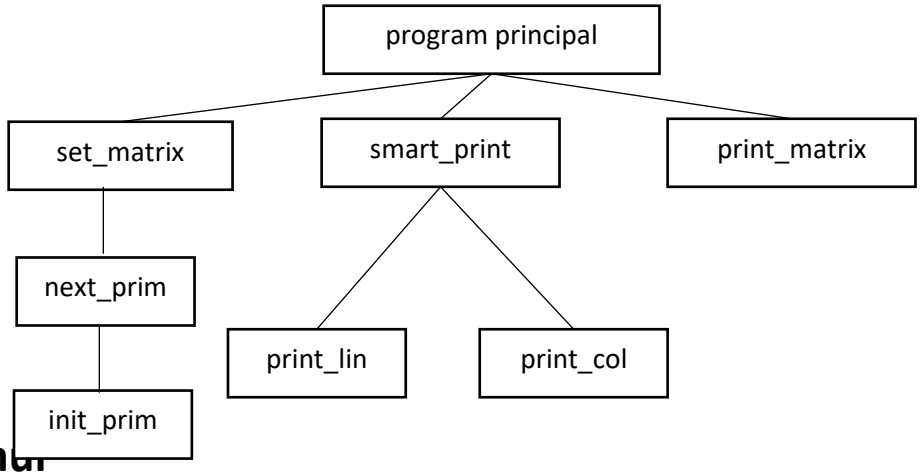
Pasii algoritmului principal

Algoritm matriceSpirala
    @ citeste dimensiuni matrice
    @ formeaza matricea cu numere prime
    @ afiseaza matricea
    @ afiseaza matricea dupa linie si coloana
Sf.Algoritm

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

Identificarea subalgoritmilor



Programul

```
// 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 2017

#include <iostream>

using namespace std;

typedef struct {
  int n;
  int elem[100][100];
} Matrix;

bool is_prim(int nr) {
  if (nr == 2)
    return true;

  for (int i = 2; i <= nr / 2; i++)
    if (nr%i == 0)
      return false;
  return true;
}

int next_prim(int nr) {
  nr++;
  while (!is_prim(nr)) nr++;
  return nr;
}

void init_matrix (Matrix &matrix) {
  int value = 2;
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
  for (int i = 0; i < matrix.n; i++) {
    for (int j = 0; j < matrix.n; j++) {
      int np = next_prim(value);
      matrix.elem[i][j] = np;
      value = np;
    }
  }
}

void print_matrix (Matrix &matrix) {
  for (int i = 0; i < matrix.n; i++) {
    for (int j = 0; j < matrix.n; j++)
      cout << matrix.elem[i][j] << " ";
    cout << endl;
  }
  cout << endl;
}

void print_lin (Matrix &matrix, int poz, int lin) {
  for (int i = poz; i < matrix.n; i++) {
    cout << matrix.elem[lin][i] << " ";
  }
}

void print_col (Matrix &matrix, int poz, int col) {
  for (int i = poz + 1; i < matrix.n; i++) {
    cout << matrix.elem[i][col] << " ";
  }
}

void smart_print (Matrix &matrix) {
  int diag_pos = 0;
  while(diag_pos < matrix.n) {
    print_lin(matrix, diag_pos, diag_pos);
    print_col(matrix, diag_pos, diag_pos);
    diag_pos++;
  }
}

int main() {
  int size = 0;
  cin >> size;

  Matrix matrix;
  matrix.n = size;

  init_matrix(matrix);
  print_matrix(matrix);
  smart_print(matrix);
}
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

Spiderman

Enunt

Omul păianjen (Spiderman) sare de pe o clădire pe alta, aflată în imediata vecinătate, în nord, est, sud sau vest. Clădirile din cartierul omului păianjen au o înălţime exprimată în numere naturale şi sunt aşezate pe `m` rânduri, câte `n` pe fiecare rând. Spiderman va alege să sară pe una dintre clădirile vecine, care are înălţimea mai mică sau egală, iar diferenţa de înălţime este minimă. Dacă există mai multe clădiri vecine de aceeaşi înălţime, omul păianjen aplică ordinea preferenţială nord, est, sud, vest, dar nu sare încă o dată pe o clădire pe care a mai sărit. Scopul omului păianjen este acela de a reuşi să facă un număr maxim de sărituri succesive.

Cerinţă

Scrieţi un program care determină numărul maxim de sărituri succesive, pe care îl poate efectua, pornind de la oricare dintre clădiri, precum şi coordonatele cladirii care reprezinta punctul de start pentru drumul maxim.

Date de intrare

n, m: 1<= n,m <= 100
a – matricea cu n linii si m coloane reprezentând înălţimile clădirilor
înălţimile clădirilor (valorile matricii) sunt numere naturale din intervalul [1,10.000]

Date de ieşire

numărul maxim de sărituri, coordonatele (i,j) punctului de start

Exemplu

Date de intrare	Date de iesire
n = 5 , m = 5 35 38 42 40 50 34 38 30 75 50 70 78 88 86 30 39 90 88 23 25 35 80 89 90 34	8 (numarul maxim de sarituri) linia = 4, coloana = 3 – pentru numerotarea de la 0 SAU linia = 5, coloana = 4 – pentru numerotarea de la 1

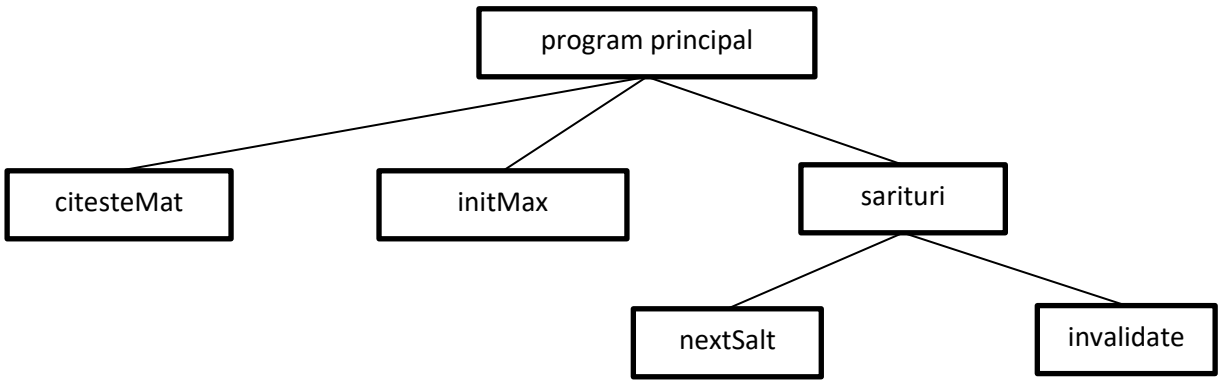
7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

Pasii algorimului principal

Algoritm matriceSpirala
    @ se initialieaza un numar max de sarituri si un punct de plecare pentru maximul respectiv
    @ Pentru fiecare punct din matrice
        @ se alege punctul de start ca si punctul curent
        @ se calculeaza numarul de sarituri pentru punctul de start respectiv
        @ Daca numar sarituri > numar max sarituri
            @se suprascriu numar max de sarituri si punctul de plecare cu noile valori
        @ Sf.Daca
    @ Sf.Pentru
    @ se tipareste numar maxim sarituri si punctul de start
Sf.Algoritm

Identificarea subalgoritmilor



Programul

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 2017
#include <iostream>
using namespace std;

// Tipul de data matrice
typedef struct {
	int n, m;
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
	int elem[100][100];
} Matrice;

typedef struct {
	int i;
	int j;
} Punct;

//Date de intrare: -
//Date de iesire: a matrice cu dimensiunile n,m –dimensiunile matricii, n,m din N,
1<=n,m<=100 si elemente intre 1 si 10000
Matrice citesteMat() {
	Matrice a;
	cout << "Introduceti dimensiunile matricii" << endl;
	cout << "Linii=";
	do {
		cin >> a.n;
		if (!(a.n >= 1 && a.n <= 100))
			cout << "Va rog sa introduceti un numar intre 1 si 100";
	} while (!(a.n >= 1 && a.n <= 100));
	cout << "Coloane=";
	do {
		cin >> a.m;
		if (!(a.m >= 1 && a.n <= 100))
			cout << "Va rog sa introduceti un numar intre 1 si 100";
	} while (!(a.m >= 1 && a.m <= 100));
	cout << "Introduceti elementele matricii linie cu linie" << endl;
	for (int i = 0;i < a.n;i++)
		for (int j = 0;j < a.m;j++)
			do {
				cin >> a.elem[i][j];
				if (!(a.elem[i][j] >= 1 && a.elem[i][j] <= 10000))
					cout << "Va rog sa introduceti un numar intre 1 si
100";
			} while (!(a.elem[i][j] >= 1 && a.elem[i][j] <= 10000));
	return a;
}

//Date de intrare: Matricea a si punctul de pornire p
//Date de iesire: punctul pe care va sari Spiderman. Acest punct va avea coordonatele -
1,-1 daca Spiderman nu mai are unde sa sara
Punct nextSalt(Matrice a, Punct p) {
	Punct next;
	next.i = -1;
	next.j = -1;
	int diferentaMinima = -1;
	int diferenta;
	//verific daca pot sari la nord
	//daca valoarea e -1 inseamna ca am fost deja pe cladirea respectiva si nu mai pot
sari acolo

	if (p.i > 0 && a.elem[p.i - 1][p.j] != -1 && a.elem[p.i - 1][p.j] <=
a.elem[p.i][p.j]) {
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
			next.i = p.i - 1;
			next.j = p.j;
			diferentaMinima = a.elem[p.i][p.j] - a.elem[p.i - 1][p.j];
		}

		//verific daca pot sari la est
		if (p.j < a.m - 1 && a.elem[p.i][p.j + 1] != -1 && a.elem[p.i][p.j + 1] <=
a.elem[p.i][p.j]) {
			diferenta = a.elem[p.i][p.j] - a.elem[p.i][p.j + 1];
			if (diferentaMinima == -1 || diferenta < diferentaMinima) {
				next.i = p.i;
				next.j = p.j + 1;
				diferentaMinima = diferenta;
			}
		}

		//verific daca pot sari la sud
		if (p.i < a.n - 1 && a.elem[p.i + 1][p.j] != -1 && a.elem[p.i + 1][p.j] <=
a.elem[p.i][p.j]) {
			diferenta = a.elem[p.i][p.j] - a.elem[p.i + 1][p.j];
			if (diferentaMinima == -1 || diferenta < diferentaMinima) {
				next.i = p.i + 1;
				next.j = p.j;
				diferentaMinima = diferenta;
			}
		}

		//verific daca pot sari la vest
		if (p.j > 0 && a.elem[p.i][p.j - 1] != -1 && a.elem[p.i][p.j - 1] <=
a.elem[p.i][p.j]) {
			diferenta = a.elem[p.i][p.j] - a.elem[p.i][p.j - 1];
			if (diferentaMinima == -1 || diferenta < diferentaMinima) {
				next.i = p.i;
				next.j = p.j - 1;
				diferentaMinima = diferenta;
			}
		}

		return next;
}

//Date de intrare: Matricea cladirilor a, Punctul de start punctStart
//Date de iesire: Matricea cladirilor a, in care s-a marcat cu -1 cladirea de pe care a
plecat Spiderman, pentru a nu mai reveni pe ea
void invalidate(Matrice& a, Punct punctStart) {
		a.elem[punctStart.i][punctStart.j] = -1;
}

//Date de intrare: Matricea cladirilor a, Punctul de start punctStart
//Date de iesire: Numarul total de sarituri pe care le poate efectua Spiderman pornind
din punctul de Start punctStart
int sarituri(Matrice a, Punct punctStart) {
		int contor = 0;
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
	Punct next = nextSalt(a, punctStart);
	while (next.i != -1) {
		contor++;
		invalidate(a, punctStart);
		punctStart = next;
		next = nextSalt(a, punctStart);
	}
	return contor;
}

//Date de intrare:-
//Date de iesire:punctul maxStartPunct se initializeaza cu coordonatele -1, -1 si numarul maxim de sarituri efectuate pana in acest moment, max, se initializeaza cu -1
void initMax(int& max, Punct& maxStartPunct) {
	max = -1;
	maxStartPunct.i = -1;
	maxStartPunct.j = -1;
}

int main() {
	Matrice a = citesteMat();

	int i, j, nr, max;
	Punct start;
	Punct maxStartPunct;

	initMax(max, maxStartPunct);

	for (i = 0; i < a.n; i++)
		for (j = 0; j < a.m; j++) {
			start.i = i;
			start.j = j;

			nr = sarituri(a, start);
			if (nr > max) {
				max = nr;
				maxStartPunct.i = start.i;
				maxStartPunct.j = start.j;
			}
		}
	cout << "Maxim: din punctul (" << maxStartPunct.i << "," << maxStartPunct.j << ")";
	cout << " a facut " << max << " sarituri" << endl;

	return 0;
}
```

Implementare Pascal

```
{ Rezolvarea nu este optimizata pentru viteza de executie
Rezolvarea exemplifica o abordare a problemei bazata pe descompunerea in subprobleme
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
Programul a fost compilat cu Turbo Pascal 7 + Dos Box}
{ Tipul de data matrice}
Type
  Matrice = Record
    n, m: Integer;
    elem: Array[1..10,1..10] Of Integer;
 {la Turbo Pascal 7 i se umple stiva de lucru la transmiterea prin parametu de tip
valoare a Array[1..100,1..100]}
  End;
Type
  Punct = Record
    i,j: Integer;
  End;

{Date de intrare: -
Date de iesire: a matrice cu dimensiunile n,m - dimensiunile matricii, n,m din N,
1<=n,m<=100 si elemente intre 1 si 10000}

Procedure citesteMat(Var a : Matrice);
Var
  i,j: Integer;
Begin
  Writeln('Introduceti dimensiunile matricii');
  Writeln('Linii:=');
  Repeat
    Readln(a.n);
    If Not ((a.n >= 1) And (a.n <= 100)) Then
      Writeln('Va rog sa introduceti un numar intre 1 si 100');
  Until (a.n >= 1) And (a.n <= 100);
  Writeln('Coloane:=');
  Repeat
    Readln(a.m);
    If Not((a.m >= 1) And (a.n <= 100)) Then
      Writeln('Va rog sa introduceti un numar intre 1 si 100');
  Until (a.m >= 1) And (a.m <= 100);
  Writeln('Introduceti elementele matricii linie cu linie');
  For i := 1 To a.n Do
    For j := 1 To a.m Do
      Repeat
        Readln(a.elem[i,j]);
        If Not(a.elem[i,j] >= 1) And (a.elem[i,j] <= 10000) Then
          Writeln('Va rog sa introduceti un numar intre 1 si 10000');
      Until (a.elem[i,j] >= 1) And (a.elem[i,j] <= 10000);
End;

{Date de intrare: Matricea a si punctul de pornire p
Date de iesire: punctul pe care va sari Spiderman. Acest punct va avea coordonatele -
1,-1 daca
Spiderman nu mai are unde sa sara}

Procedure nextSalt(a:Matrice; p:Punct;Var next:Punct);
Var
  diferentaMinima,diferenta: Integer;
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
Begin
  next.i := -1;
  next.j := -1;
  diferentaMinima := -1;

{verific daca pot sari la nord
daca valoarea e -1 inseamna ca am fost deja pe cladirea respectiva si nu mai pot sari
acolo}
  If (p.i > 1) And (a.elem[p.i - 1,p.j] <> -1) And (a.elem[p.i - 1,p.j] <=
a.elem[p.i,p.j]) Then
    Begin
      next.i := p.i - 1;
      next.j := p.j;
      diferentaMinima := a.elem[p.i,p.j] - a.elem[p.i - 1,p.j];
    End;
 {verific daca pot sari la est}
  If (p.j < a.m) And (a.elem[p.i,p.j + 1] <> -1) And (a.elem[p.i,p.j + 1] <=
a.elem[p.i,p.j]) Then
    Begin
      diferenta := a.elem[p.i,p.j] - a.elem[p.i,p.j + 1];
      If (diferentaMinima = -1) Or (diferenta < diferentaMinima) Then
        Begin
          next.i := p.i;
          next.j := p.j + 1;
          diferentaMinima := diferenta;
        End;
    End;
 {verific daca pot sari la sud}
  If (p.i < a.n) And (a.elem[p.i + 1,p.j] <> -1) And (a.elem[p.i + 1,p.j] <=
a.elem[p.i,p.j]) Then
    Begin
      diferenta := a.elem[p.i,p.j] - a.elem[p.i + 1,p.j];
      If (diferentaMinima = -1) Or (diferenta < diferentaMinima) Then
        Begin
          next.i := p.i + 1;
          next.j := p.j;
          diferentaMinima := diferenta;
        End;
    End;
 {verific daca pot sari la vest}
  If (p.j - 1 > 0) And (a.elem[p.i,p.j - 1] <> -1) And (a.elem[p.i,p.j - 1] <=
a.elem[p.i,p.j]) Then
    Begin
      diferenta := a.elem[p.i,p.j] - a.elem[p.i,p.j - 1];
      If (diferentaMinima = -1) Or (diferenta < diferentaMinima) Then
        Begin
          next.i := p.i;
          next.j := p.j - 1;
          diferentaMinima := diferenta;
        End;
    End;
End;
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
{Date de intrare: Matricea cladirilor a, Punctul de start punctStart
Date de iesire: Matricea cladirilor a, in care s-a marcat cu -1 cladirea de pe care a
plecat Spiderman,
pentru a nu mai reveni pe ea}

Procedure invalidate(Var a:Matrice; punctStart:Punct);
Begin
  a.elem[punctStart.i,punctStart.j] := -1;
End;

{Date de intrare: Matricea cladirilor a, Punctul de start punctStart
{Date de iesire: Numarul total de sarituri pe care le poate efectua Spiderman pornind
din punctul de Start punctStart}

Function sarituri(a:Matrice; punctStart:Punct): Integer;
Var
  contor: Integer;
  next: Punct;
Begin
  contor := 0;
  nextSalt(a, punctStart,next);
  While (next.i <> -1) Do
    Begin
      contor := contor+1;
      invalidate(a, punctStart);
      punctStart := next;
      nextSalt(a, punctStart,next);
    End;
  sarituri := contor;
End;

{Date de intrare:-
Date de iesire:punctul maxStartPunct se initializeaza cu coordonatele -1, -1 si
numarul maxim de sarituri efectuate
pana in acest moment, max, se initializeaza cu -1}

Procedure initMax(Var max:Integer; maxStartPunct:Punct);
Begin
  max := -1;
  maxStartPunct.i := -1;
  maxStartPunct.j := -1;
End;
Var
  a: Matrice;
  i, j, nr, max: Integer;
  start,maxStartPunct: Punct;
Begin
  citesteMat(a);
  initMax(max, maxStartPunct);
  For i := 1 To a.n Do
    For j := 1 To a.m Do
      Begin
        start.i := i;
```

7 decembrie 2019 Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
        start.j := j;
        nr := sarituri(a, start);
        If nr > max Then
          Begin
            max := nr;
            maxStartPunct.i := start.i;
            maxStartPunct.j := start.j;
          End;
      End;
  Writeln('Maxim: din punctul (' , maxStartPunct.i , ',' , maxStartPunct.j , ')');
  Writeln(' a facut ' , max , ' sarituri');
  Readln;
End.
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

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.

Cerință

Scrieţi 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 şi cu programul principal prin parametri. Fiecare subprogram trebuie specificat.

Exemplu

Date de intrare	Date de iesire
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	10 (aria maximă a încăperii din colțul dreapta sus)

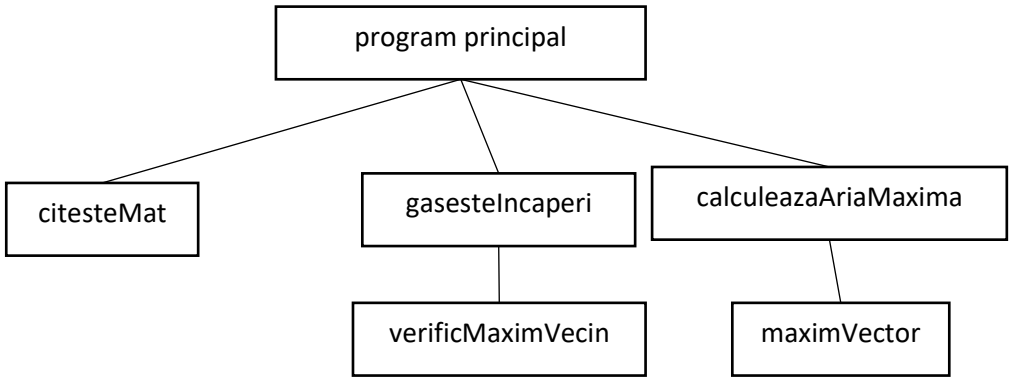
7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

Pasii algorimului principal

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

Identificarea subalgoritmilor



Programul

Implementare Iterativă 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 2017
#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)
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
			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 maximul 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) {
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
	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("casa.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;
}
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

Implementare recursivă C++

-- matrice.h --

```
const int MAX = 200;
struct Matrice {
	int m;
	int n;
	int elem[MAX][MAX];
};

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

-- matrice.cpp –

```
#include "Matrice.h"
#include <iostream>
#include <iomanip>
using namespace std;

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;
}
```

```
#include <iostream>
#include "Matrice.h"
using namespace std;
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
// 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("casa.in");
	cout << "Dimensiunea camerei maxime: " << cameraMaxima(casa);
}
```

Implementare iterativă 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 Turbo Pascal 7 + Dos Box}
{ Tipul de data matrice}
Type
  Matrice = Record
    n, m: Integer;
    elem: Array[1..10,1..10] Of Integer;
 {la Turbo Pascal 7 i se umple stiva de lucru la transmiterea prin parametu de tip
valoare a Array[1..100,1..100]}
  End;
Type Vector = Array[1..100] of Integer;
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
{Date de intrare: -
Date de iesire: a matrice cu dimensiunile n,m - dimensiunile matricii, n,m din N,
1<=n,m<=100 si elemente intre 1 si 10000}
Procedure citesteMat(Var a : Matrice);
Var
  i,j: Integer;
Begin
  Writeln('Introduceti dimensiunile matricii');
  Writeln('Linii:=');
  Repeat
    Readln(a.n);
    If Not ((a.n >= 1) And (a.n <= 100)) Then
      Writeln('Va rog sa introduceti un numar intre 1 si 100');
  Until (a.n >= 1) And (a.n <= 100);
  Writeln('Coloane:=');
  Repeat
    Readln(a.m);
    If Not((a.m >= 1) And (a.n <= 100)) Then
      Writeln('Va rog sa introduceti un numar intre 1 si 100');
  Until (a.m >= 1) And (a.m <= 100);
  Writeln('Introduceti elementele matricii linie cu linie');
  For i := 1 To a.n Do
    For j := 1 To a.m Do
      Repeat
        Readln(a.elem[i,j]);
        If (a.elem[i,j] <> -1) And (a.elem[i,j] <> 0) Then
          Writeln('Va rog sa introduceti -1 (nu e perete) sau 0 (e perete)');
      Until (a.elem[i,j] = -1) Or (a.elem[i,j] = 0);
End;

{verific daca valoarea vreunui vecin este >0 si o returnez.
Inseamna ca e o camera deja detectata.
Date de intrare: Matricea cu planul casei a si i,j coordonatele punctului de analizat
Date de iesire: valoarea celui mai mare vecin al punctului analizat}
Function verificMaximVecin(a:Matrice;i,j:Integer):Integer;
Var max:integer;
begin
      max := -1;

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

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

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

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

	{daca am un vecin in directia respectiva (dreapta) si nu e perete}
	if (j < a.m) 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
Date de intrare: Matricea cu planul casei a, cu elemente: -1 (zona nedetectata), 0 -
perete, k din [1,contor incaperi]
- care indica ca punctul curent apartine de camera k
si contorIncaperi := nr. de incaperi identificate deja
Date de iesire: false daca nu s-au mai modificat incaperile detectate,
true daca s-au mai modificat incaperile detectate, a - planul actualizat si
contorIncaperi actualizat}
Function gasesteIncaperi(Var a:Matrice; Var contorIncaperi:Integer):Boolean;
Var i,j,max:integer;
	schimbari:Boolean;
begin
	schimbari := false;

	for i := 1 to a.n do
		for j := 1 to a.m Do
		 begin
			if a.elem[i,j] <> 0 then
			 begin
				max := verificMaximVecin(a, i, j);
				{daca minimul e -1 si a.elem[i,j]=-1, atunci e o incapere
				inca nedescoperita si se va marca cu un numar nou}
				if (a.elem[i,j]=-1) and (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
				si punctul analizat se va lipi de camera cu cel mai mare
				indice
```

7 decembrie 2019 Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
                        astfel, incet, incet, unele camere sa fie absorbite de
                        altele cu indice mai mare}
                        else
                              if max>a.elem[i,j] 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
Date de intrare: v vector de nr. intregi, l- nr. de elemente ale vectorului
Date de iesire: cea mai mare valoare din vector}
Function maximVector(v:Vector; l:integer):Integer;
Var i,max:integer;
begin
      max := 0;
      for i := 1 to l do
            if v[i] > max then
                  max := v[i];
      maximVector:=max;
end;
{Date de intrare: Matricea cu planul casei a cu elemente: 0 -perete, k din [1,contor
incaperi]
- care indica ca punctul curent apartine de camera k
si contorIncaperi := cel mai amre indice al unei incaperi din casa
Date de iesire: cea mai mare arie a unei incaperi din plan}
Function calculeazaAriaMaxima(a:Matrice; contorIncaperi:Integer): Integer;
Var i,j,idCamera:integer;
      ariiCamere:Vector;
begin
      {initializez toate ariile cu 0;}
      for i := 1 to contorIncaperi do
            ariiCamere[i] := 0;

      for i := 1 to a.n do
            for j := 1 to a.m 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;
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
Var schimbari:Boolean;
      contorIncaperi,aria:Integer;
      a: Matrice;
begin
      citesteMat(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('Aria maxima a unei incaperi este: ',aria);
end.
```

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
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
    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] <> 0 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.
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

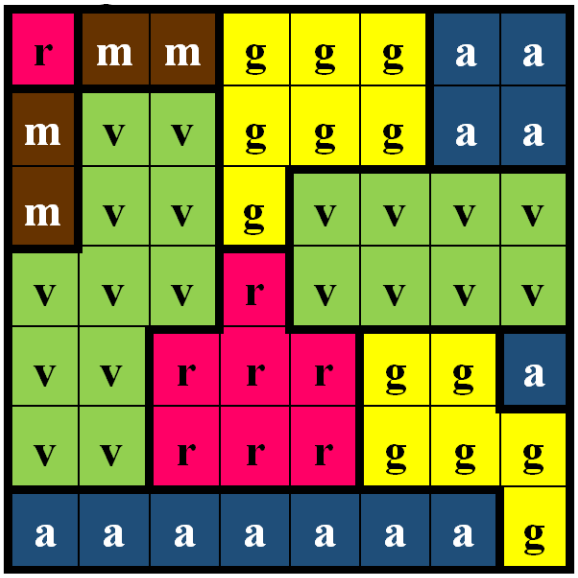
7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

Ferma[2]

Enunț

Un fermier deține o fermă de formă dreptunghiulară cu lungimea **m** metri și lățimea **n** metri. Respectând principiul rotației culturilor, fermierul și-a realizat un plan pentru semănarea culturilor în noul an. Astfel ,el a desenat un dreptunghi pe care l-a împărțit în **m** * **n** celule, fiecare corespunzând unui metru pătrat, și a colorat în culori diferite zonele care corespund unor culturi diferite. O cultură poate fi semănată pe mai multe parcele. Două celule care au o latură comună aparțin aceleiași parcele dacă au aceeași culoare (sunt însămânțate cu aceeași cultură). Fermierul are posibilitatea să irige o sigură parcelă și dorește să aleagă parcela cu cea mai mare suprafață. Nefiind mulțumit de suprafața rezultată, s-a întrebat dacă ar putea schimba cultura de pe o singură celulă, astfel încât să obțină o parcelă de suprafață mai mare.



Exemplu culturi ferma

Cerință

Dându-se dimensiunile fermei și pentru fiecare celulă culoarea corespunzătoare culturii semănate, determinați dimensiunea maximă a parcelei ce poate fi obținută prin schimbarea tipului de cultură într-o singură parcelă.

Date de intrare

Fișierul de intrare **ferma.in** va conține:

[2] Enunț adaptat pornind de la OJI 2014.

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

- pe prima linie două numere naturale **m** şi **n** separate printr-un spațiu, cu semnificația din enunț;
- pe fiecare dintre următoarele **m** linii se găsesc câte **n** caractere (litere mici), reprezentând codurile culturilor ce vor fi semănate pe cele **n** celule corespunzătoare fiecărei linii.

Date de ieşire

Dimensiunea parcelei maxime care se poate obține prin semănarea altei culturi

Restricţii şi precizări

- $2 \leq m \leq 400$
- $2 \leq n \leq 400$
- Numărul de culturi distincte este cel puţin 2 şi cel mult 26.

Exemplu

ferma.in	**Explicații**
7 8 rmmgggaa mvvgggaa mvvgvvvv vvvrvvvv vvrrrgga vvrrrggg aaaaaaag	Schimbând în **verde** (v) culoarea celulei de pe linia **3** şi coloana **4,** se obține o parcelă cu suprafața 11+8+1=20 (se unesc parcelele cu numărul 6 respectiv 8).

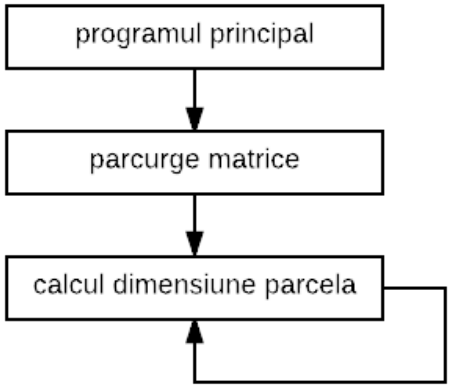
Pasii algorimului principal

Algoritm Ferma
    @ citeste matrice
    @ parcurge matricea
    @ schimbare cultura pentru fiecare casuta
    @ calculeaza aria obtinuta
    @ afiseaza aria maxima
Sf.Algoritm

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

Identificarea subalgoritmilor



Programul

```
// 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 2017
#include <iostream>

struct Parcela {
	int linie;
	int coloana;
};

struct Stiva {
	int varf;
	Parcela parcele[15000];
};

struct Ferma {
	int linii;
	int coloane;
	char celule[402][402];
};

bool egal(Parcela& p1, Parcela& p2) {
	return p1.coloana == p2.coloana && p1.linie == p2.linie;
}

// verificam daca stiva data contine parcela
bool contine(Stiva& stiva, Parcela& p) {
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
	for (int i = 0; i < stiva.varf; i++)
		if (egal(p, stiva.parcele[i]) == true)
			return true;
	return false;
}

// adaugarea unei noi parcele in stiva
void push(Stiva& stiva, Parcela& p) {
	stiva.parcele[stiva.varf++] = p;
}

// citirea datelor despre ferma
Ferma citire(char* fisier) {
	FILE *fin;
	Ferma m;
	fopen_s(&fin, fisier, "r");
	fscanf_s(fin, "%d %d\n", &m.linii, &m.coloane);
	for (int i = 0; i < m.linii; i++)
		fgets(m.celule[i], 400, fin);
	return m;
}

// apelul recursiv pentru calculul dimensiunii parcelei, incepand cu pozitia (l,c)
// valoarea 'v' retine cultura pe care o cautam
// celulele memorate le pastram intr-o stiva
int dimParcelaRec(Ferma& f, int l, int c, char v, Stiva& stiva) {
	if (l < 0 || l >= f.linii || c < 0 || c >= f.coloane) return 0;
	if (f.celule[l][c] != v) {
		return 0;
	}

	Parcela p;
	p.linie = l;
	p.coloana = c;

	// daca stiva memoreaza parcela curenta, nu o mai numaram
	if (contine(stiva, p))
		return 0;
	push(stiva, p);

	// 1 pentru celula curenta + valoarea apelului recursiv pentru celulele
adiacente
	return 1 + dimParcelaRec(f, l - 1, c, v, stiva) + dimParcelaRec(f, l + 1, c,
v, stiva) + dimParcelaRec(f, l, c - 1, v, stiva) + dimParcelaRec(f, l, c + 1, v,
stiva);
}

// calculam dimensiunea maxima a parcelei de pe pozitia (l,c)
int dimParcela(Ferma& f, int l, int c) {
	Stiva s;
	s.varf = 0;
```

7 decembrie 2019

Conf. dr. Chira Camelia
Lect. dr. Arthur Molnar

```
	return dimParcelaRec(f, l, c, f.celule[l][c], s);
}

int parcurgere(Ferma& ferma) {
	int max = -1;
	//parcurgem fiecare celula a fermei
	for (int i = 0; i < ferma.linii; i++)
		for (int j = 0; j < ferma.coloane; j++)
		{
			//retinem cultura originala a celulei curente
			char original = ferma.celule[i][j];
			for (char c = 'a'; c <= 'z'; c++) {
				//incercam sa inlocuim cultura existenta cu alta
				//de fiecare data calculam dimensiunea parcelei obtinute
				ferma.celule[i][j] = c;
				int parcela = dimParcela(ferma, i, j);
				if (parcela > max) {
					max = parcela;
				}
			}
			ferma.celule[i][j] = original;
		}
	return max;
}

void main() {
	Ferma ferma = citire("ferma.in");
	std::cout << "Dimensiunea parcelei maxime care se poate obtine: " <<
parcurgere(ferma) << std::endl;
}
```