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:

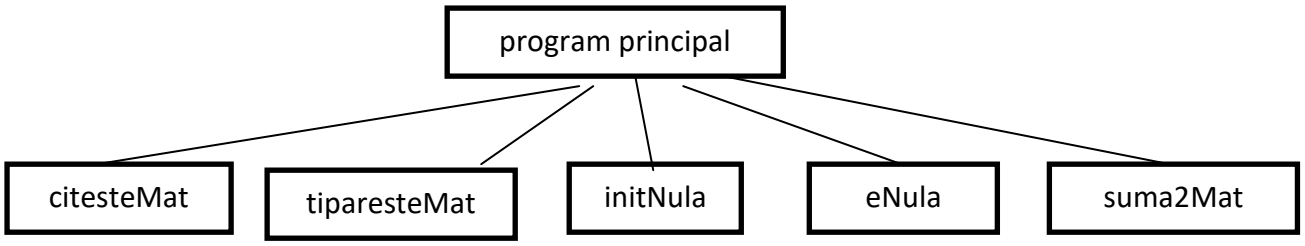
1. daca se citeste doar matricea nula rezultatul afisat va fi matricea nula
2. se presupune ca datele sunt corect introduse

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 zero

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

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

// Descr: citeste o matrice
// In: -
// Out:  a - matricea citita
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];
}

// Descr: tipareste o matrice
// In:		a - matricea
// Out:		-
//			(se tipareste matricea)
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";
	}
}

// Desc:  aduna o matrice la o matrice data
// In:		a - matrice
//			b - matrice
// Out:		a - matricea a la care s-a adunat matricea b
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];
}

// 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
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;
}

// 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
```

```
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: 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

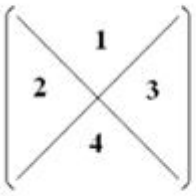
a - matrice cu n linii si n coloane

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 zero

```
// desc: verifica daca numarul nr este prim
// in: nr – numar natural
// out: ePrim = true  daca nr e prim
//              false in caz contrar
bool ePrim(int nr) {
	if (nr < 2) return false;
	int divizor = 2;
	while (divizor*divizor <= nr) {
		if (nr % divizor == 0) return false;
		divizor = divizor + 1;
	}
	return true;
}

// 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 dreptal matricei a
// Versiune care parcurge toata matricea
int nrPrime(int n, int a[MAX][MAX]){
	int nr = 0;
	for (int i = 0; i<n; i++)
		for (int j = 0; j<n; j++)
			// daca elem e in triunghiul stang al matricei si e prim
			if ((i>j) && (i + j < n - 1) && ePrim(a[i][j]))
				nr++;
			else		// daca e in triunghiul drept al matricei si e prim
				if ((i<j) && (i + j > n - 1) && ePrim(a[i][j]))
					nr++;
	return nr;
}
```

```
// O versiune a subprogramului
// care parcurge doar elementele celor doua triunghiuri
int nrPrime_V2 (int n, int a[MAX][MAX]){
	int nr = 0;
	int i, j;

	for (i = 1; i<n - 1; i++) {
		// daca elementul e in triunghiul stang al matricei merg pana la:
		// intalnirea uneia dintre cele doua diagonale
		for (j = 0; ((i>j) && (i + j < n - 1)); j++) {
			if (ePrim(a[i][j])) nr++;
		}
		// daca elementul e in triunghiul drept al matricei si e prim
		// merg de la dreapta la stanga pana la intalnirea uneia
		// dintre cele doua diagonale
		for (j = n - 1; (i<j) && (i + j>n - 1); j--) {
			if (ePrim(a[i][j]))	nr++;
		}
	}
	return nr;
}
```

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: `rezolvarea este data pentru matrici indexate de la zero`

```
// 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
void eliminaLiniiColoaneImpare(int&m, int&n, int a[MAX][MAX]){
      int i, j;
      m = m / 2;
      n = n / 2;
      for (i = 0; i<m; i++) {
            for (j = 0; j<n; j++) {
                  a[i][j] = a[2 * i][2 * j];
            }
      }
}
```

Exemple

Date de intrare	Rezultate
Matrice: 3 3 1 2 3 4 5 6 7 8 9	1 3 7 9
Matrice: 2 2 10 11 12 13	10
Matrice: 1 5 1 2 3 4 5	1 3 5

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 zero`

```
// Desc.:    initializeaza o matrice sursa cu valorile unei matrici destinatie
// In:       m,n : 1 <= n, m <= 100
//           a -matricea sursa cu m linii si n coloane
// Out:      b : b = a , b-matricea destinatie
void copiazaMat(int m, int n, int a[MAX][MAX], int b[MAX][MAX]){
      for (int i = 0; i<m; i++)
            for (int j = 0; j<n; j++)
                  b[i][j] = a[i][j];
}

const int vx[4] = { -1, 0, 1, 0 };
const int vy[4] = { 0, 1, 0, -1 };

// 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)
int nrVecini(int m, int n, int a[MAX][MAX], int i, int j){
```

```
	int k;
	int rezultat = 0;
	for (k = 0; k<4; k++)
		if ((i + vx[k] >= 0) && (i + vx[k]<m) //daca coordonatele sunt valide
			&& (j + vy[k] >= 0) && (j + vy[k]<n)
			)
			if (a[i + vx[k]][j + vy[k]] == 1)	rezultat++;
	return rezultat;
}

// 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)
bool esteImpadurita(int m, int n, int a[MAX][MAX]){
	for (int i = 0; i<m; i++)
		for (int j = 0; j<n; j++)
			if (a[i][j] == 0) return false;
	return true;
}

// 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
void impadurire(int m, int n, int a[MAX][MAX], bool & sePoate, int& nrZile){
	int b[MAX][MAX];
	int i, j;
	int nrCopaci;

	nrZile = 0;
	do {
		nrCopaci = 0;
		copiazaMat(m, n, a, b);
		for (i = 0; i<m; i++) {
			for (j = 0; j<n; j++) {
				if (a[i][j] == 0) {
					if (nrVecini(m, n, b, i, j) >= 2) {
						a[i][j] = 1;
						nrCopaci++;
					}
				}
			}
		}
		if (nrCopaci>0) nrZile++;
	} while (nrCopaci>0);
	sePoate = esteImpadurita(m, n, a);
}
```

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