

Ministerul Educației, Culturii și Cercetării
Universitatea Tehnică a Moldovei



Departamentul Ingineria Software și Automatică

RAPORT

Lucrarea de laborator nr. 5
la Programarea Calculatoarelor
Varianta 18

A efectuat:
st. gr. TI-206
A verificat:
Lector universitar

Cătălin Pleșu
Vitalie Mititelu

Lucrarea de laborator nr. 5

Tema : Alocarea dinamică a memoriei pentru tablourile bidimensionale. Utilizarea funcțiilor și a pointerilor

Scopul : Programarea algoritmilor de prelucrare a tablourilor bidimensionale prin utilizarea funcțiilor, pointerilor și alocarea dinamică a memoriei pentru tablou.

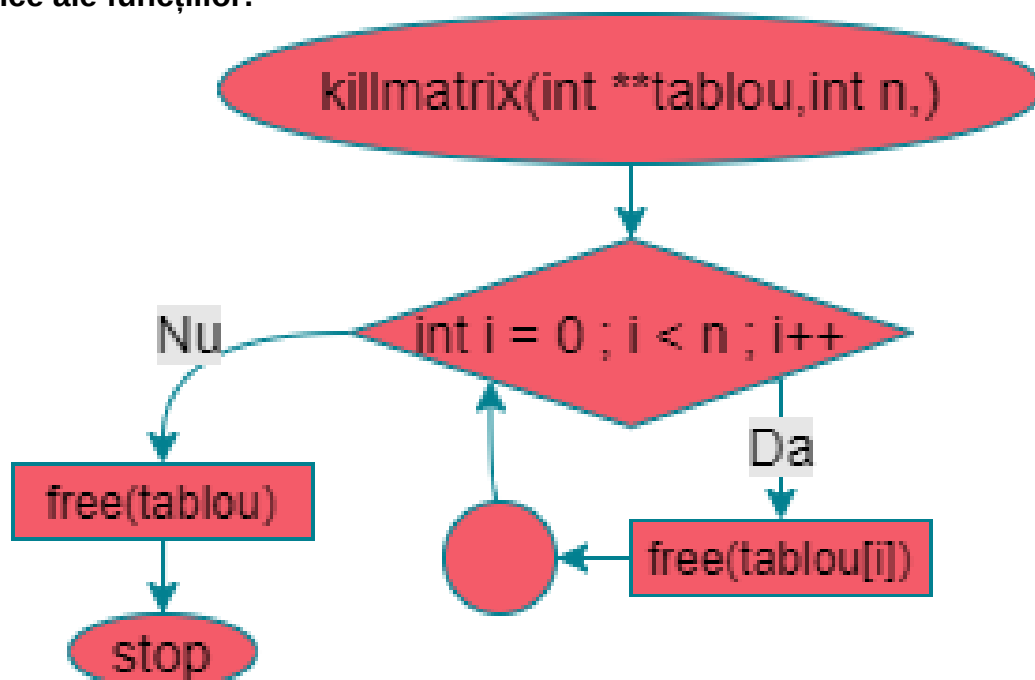
Sarcina: Scrieți un program care citește de la tastatură numărul $n > 1$ de rânduri și numărul $m > 1$ de coloane ale tabloului bidimensional (matricii), apoi citește de la tastatură aceste $n \times m$ elemente ale tabloului, efectuează calculele indicate în variantă și afișează pe ecran rezultatul:

Varianta 18. Să se determine suma elementelor pozitive din liniile pare și produsul elementelor mai mici ca 5 din coloanele impare.

Rezumat succint la temă :

- Programul trebuie să efectueze operațiile de la laboratorul precedent.
- Acest program conține subprograme adică funcții și pointeri.
- Am declarat antetul funcției înainte de funcția main iar restul funcției le-am scris sub funcția main.
- Prima funcție **menu** citește dimensiunile matricii, nu este o funcție specială.
- Am funcțiile **printm** și **scanm** care citesc și scriu matrici, ca parametri formali sunt: un dublu pointer și două variabile integer.
- Funcția **killmatrix** are ca scop eliberarea memoriei alocate pentru matrice.
- Operațiile cerute de condiția variantei 18 sunt executate în funcțiile **randparsum** și **colinparprod**.
- Matricea data este stocată în variabila **tablou** care este un dublu pointer.
- Pentru alocarea unei matrici bidimensionale am alocat memoria necesară pentru rândurile matricii, apoi cu un ciclu **for** am alocat memoria pentru coloane pe fiecare rând.
- Eliberarea memoriei are loc în funcția **killmatrix** și se execută în ordinea inversă alocării memoriei, mai întâi scăpăm de memoria de pe rânduri apoi eliminăm acest dublu pointer.
- Din proprie inițiativă am creat funcția **randm** care întreabă utilizatorul dacă dorește ca matricea să fie umplută cu elemente aleatorii cu seed-ul din timp.

Schemele logice ale funcțiilor:



randm(int **tablou,int n,int m)

doriti o matrice random 0 - da; 1 - nu

&b

!b

Dati limita de sus si jos

&max &min

int i = 0 ; i < n ; i++

Da

\n

int j = 0 ; j < m ; j++

DA

tablou[i][j] = (rand() % (max - min + 1)) + min

return b

menu(int *n,int *m)

Condiții

Nr de rânduri =

n

Nr de coloane =

m

stop

scanm(int **tablou,int n,int m)

int i = 0 ; i < n ; i++

Nu

Da

stop

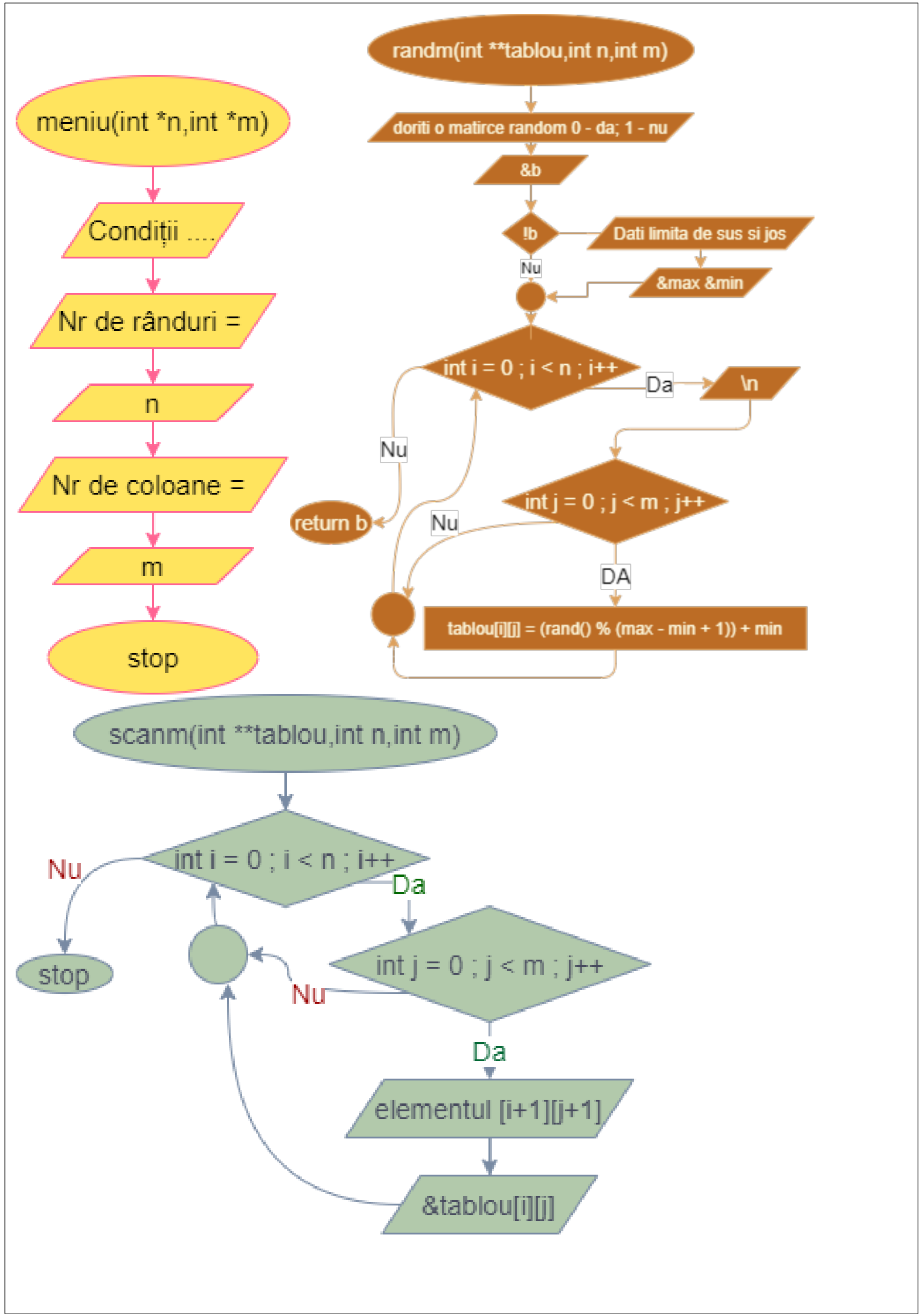
int j = 0 ; j < m ; j++

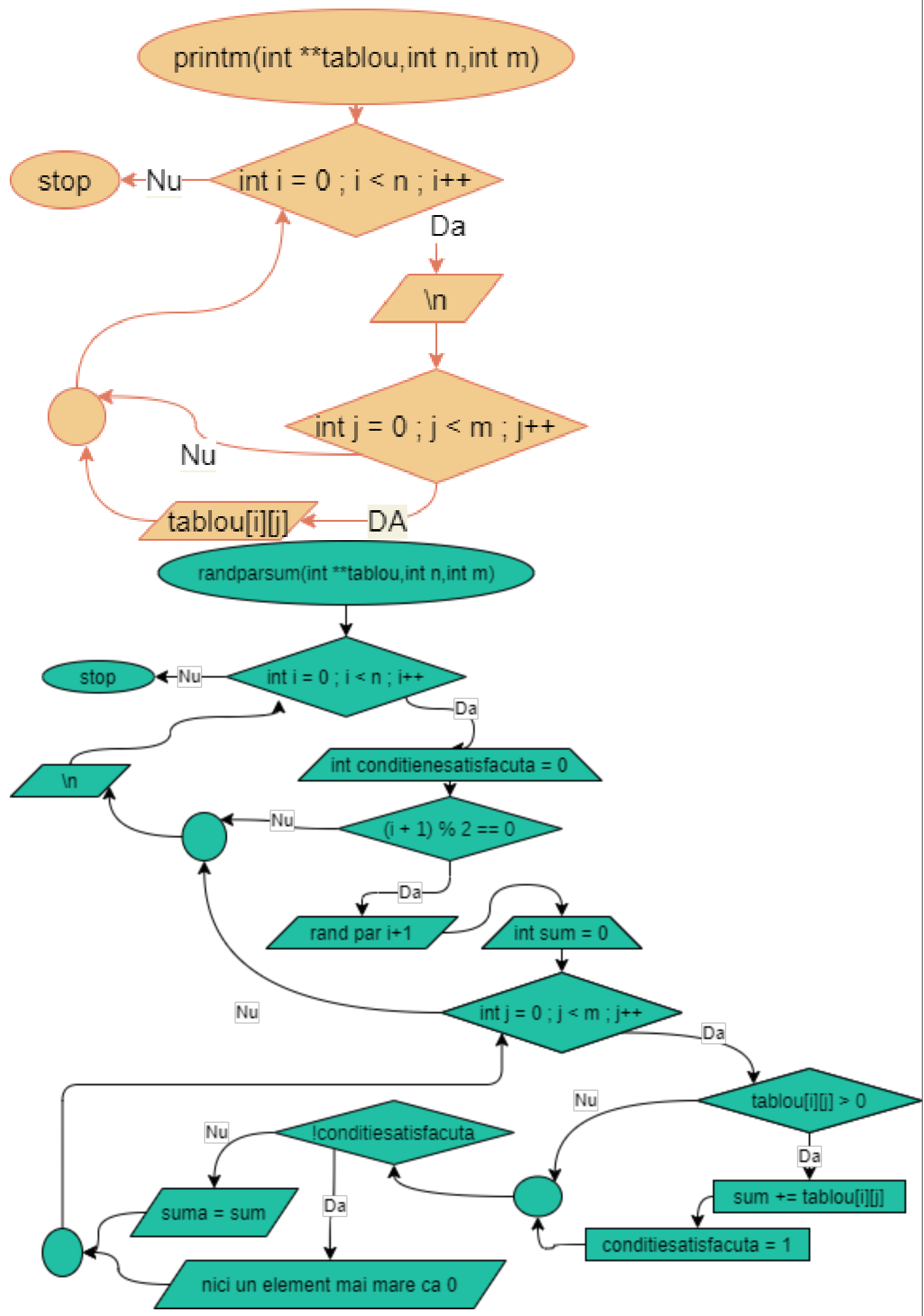
Nu

Da

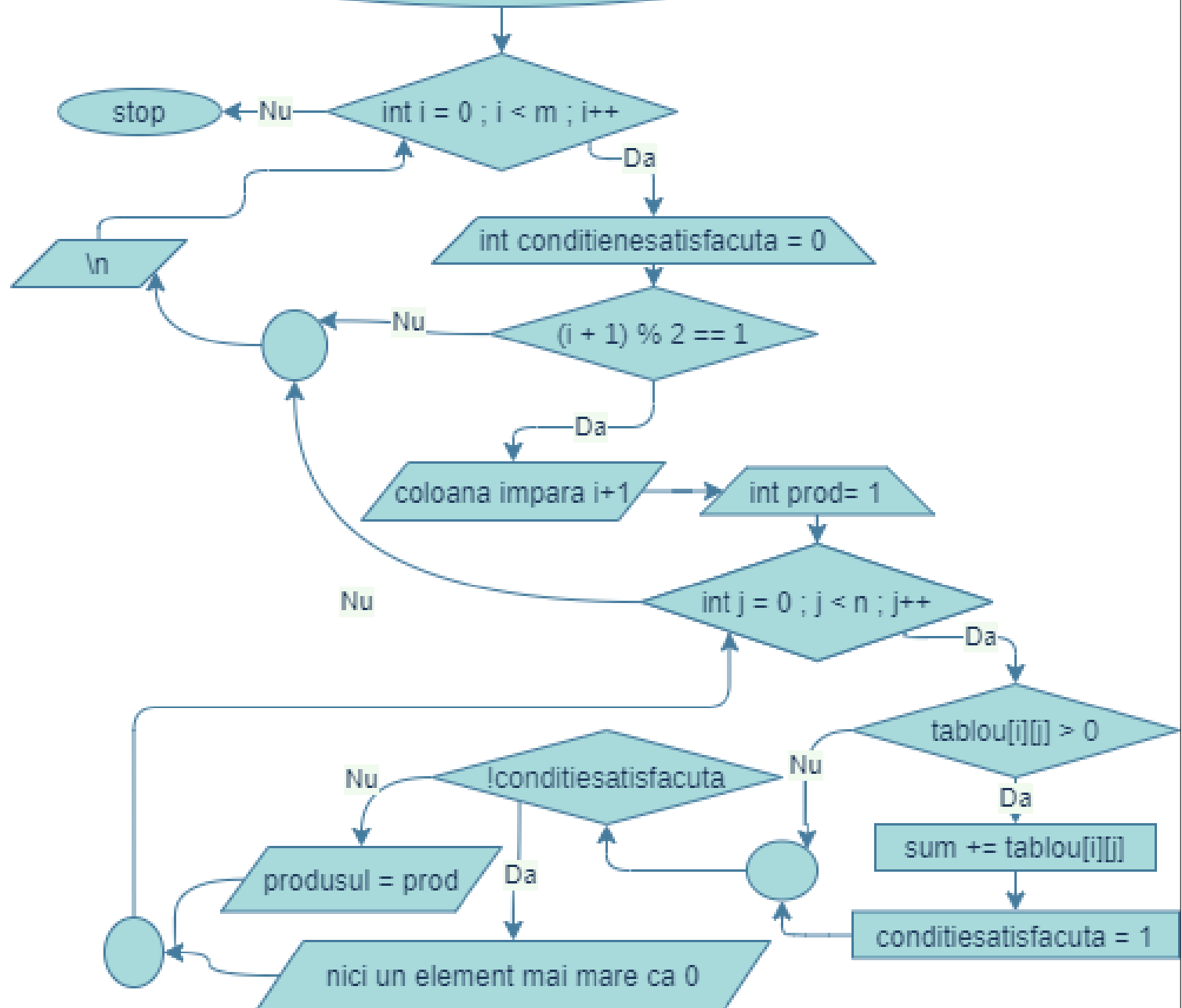
elementul [i+1][j+1]

&tablou[i][j]

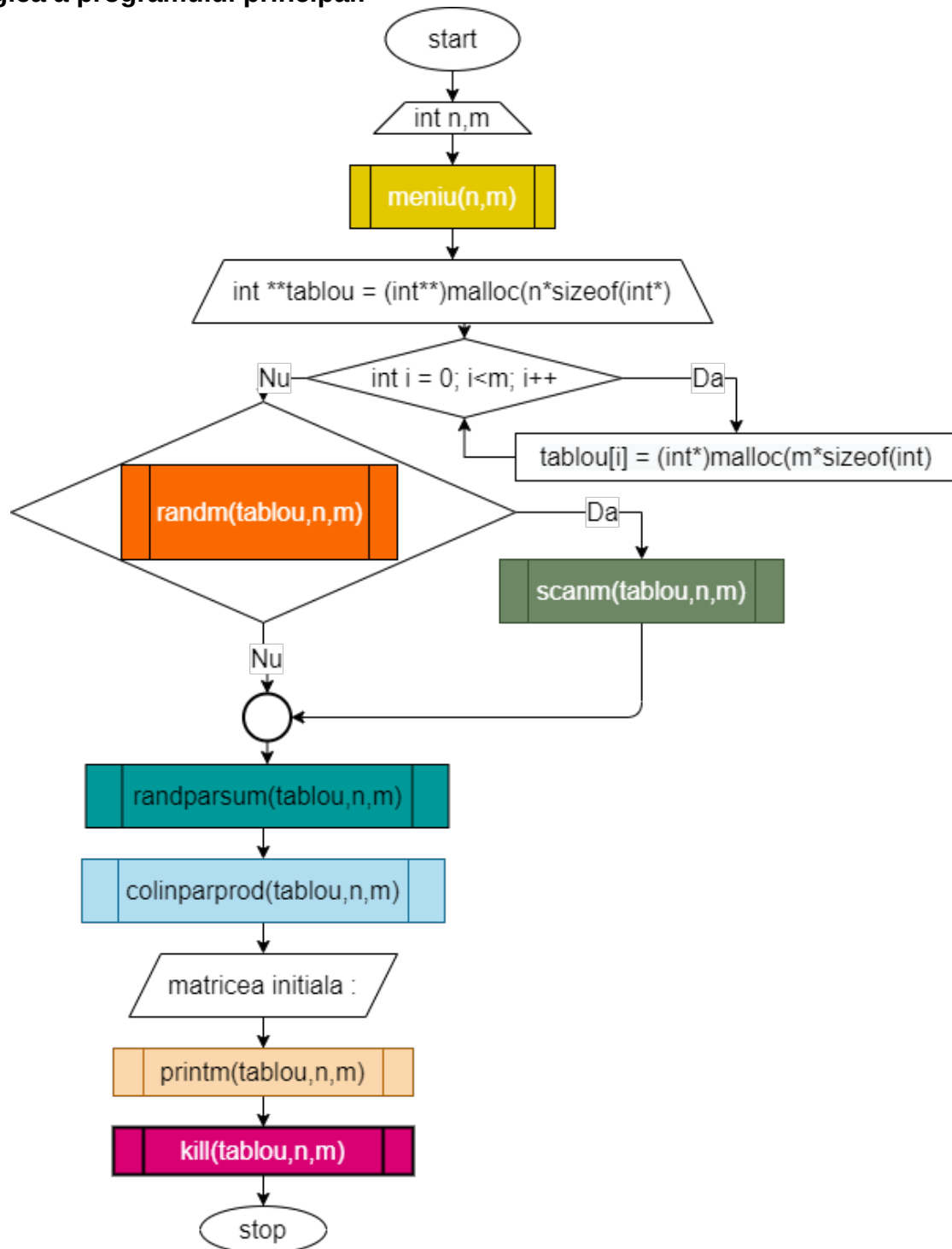




colinparprod(int **tablou,int n,int m)



Schema logică a programului principal:



Cod sursă în limbajul C :

//varianta 18 labul 5

#include <stdio.h>

#include <stdlib.h>

void meniu(int *n,int *m);

void printm(int **tablou, int n, int m);

void scanm(int **tablou, int n, int m);

void killmatrix(int **tablou, int n);

void randparsum(int **tablou, int n, int m);

void colinparprod(int **tablou, int n, int m);

int randm(int **tablou, int n, int m);

int main()

{

```

int n, m;
printf("\n\n\t\t\t");
//functia mea care scrie conditiile programului si citeste dimensiunile matricii
menu(&n,&m);
// alocarea dinamica a unui tablou de pointeri
int **tablou = (int **)malloc(n * sizeof(int *));
//tabloului de pointeri de mai sus i se alocata alt tablou
for (int i=0; i<n; i++)
    tablou[i] = (int *)malloc(m * sizeof(int));

//citeste matricea daca utilizatorului nu ii este lene
if ( randm(tablou, n, m))
{
    scanm(tablou, n, m);
}

randparsum(tablou, n, m);

colinparprod(tablou, n, m);

//printeaza tabloul / matricea
printf("matricea initiala");
printm(tablou, n, m);
//elibereaza memoria matricii
killmatrix(tablou, n);
return 0;
}

void menu(int *n,int *m){
printf("Lucrarea de laborator nr. 5 Varianta 18\n\n\n");
printf("Acest program va citi o matrice si va calcula :\nsuma elementelor pozitive din liniile par
e si\nprodusul elementelor mai mici ca 5 din coloanele impare\n");
printf("nr de randuri = ");
scanf("%d", n);
printf("nr de coloane = ");
scanf("%d", m);
}

void scanm(int **tablou, int n,int m) {

for (int i = 0; i < n; i++)
    for (int j = 0; j < m; j++)
    {
        printf("elementul [%d %d] = ", i + 1, j + 1);
        scanf("%d", &tablou[i][j]);
    }
}

void printm(int **tablou, int n,int m) {

for (int i = 0; i < n; i++)
{
    printf("\n");
    for (int j = 0; j < m; j++)

```

```

    {
        printf("%6d ", tablou[i][j]);
    }
}

void randparsum(int **tablou, int n, int m){
for (int i = 0; i < n; i++)
{
    int conditiesatisfacuta = 0;
    if ((i + 1) % 2 == 0)
    { //operatiile pentru rand par
        printf("rand par %d : ", i + 1);
        int sum = 0;
        for (int j = 0; j < m; j++)
        {
            if (tablou[i][j] > 0)
            {
                sum += tablou[i][j];
                conditiesatisfacuta = 1;
            }
        }
        if (!conditiesatisfacuta)
            printf(" nici un numare > 0");
        else
            printf(" sum = %d", sum);
    }
    printf("\n");
}
}

void colinparprod(int **tablou, int n, int m){
for (int i = 0; i < m; i++)
{
    int conditiesatisfacuta = 0;
    if ((i + 1) % 2 == 1)
    { //operatiile pentru coloana impara
        printf("\ncoloana impara %d : ", i + 1);
        int prod = 1;
        for (int j = 0; j < n; j++)
        {
            if (tablou[j][i] < 5)
            {
                prod *= tablou[j][i];
                conditiesatisfacuta = 1;
            }
        }
        if (!conditiesatisfacuta)
            printf(" conditie nesatisfacuta 0 elemente < 5");

        else
            printf(" prod = %d", prod);
    }
    printf("\n");
}
}

```


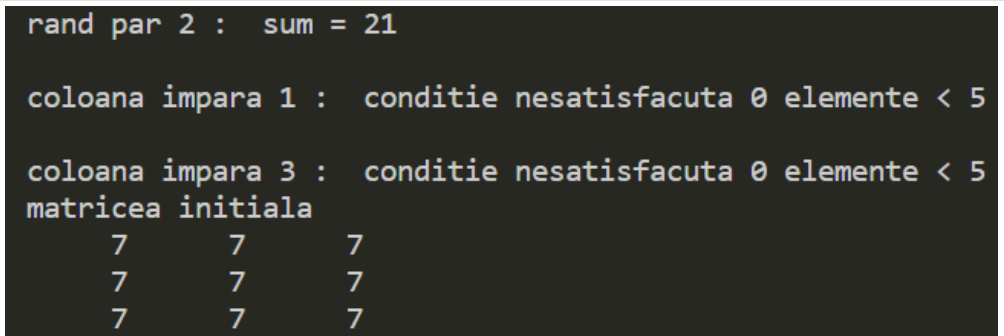
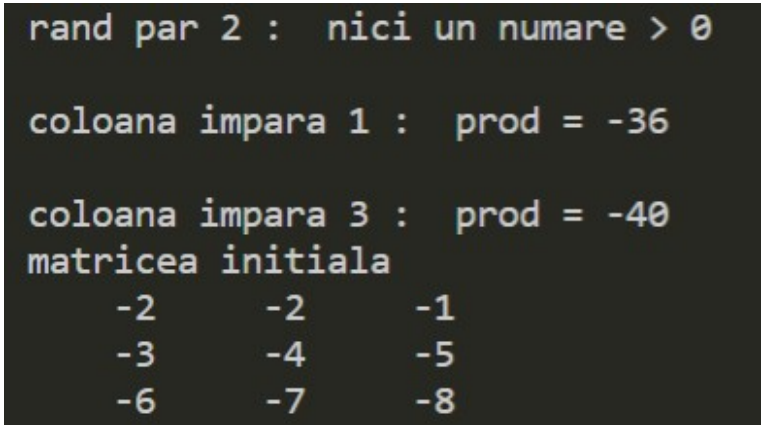


```
}
```

```
void killmatrix(int **tablou, int n) {  
    for (int i=0; i<n; i++)  
        free(tablou[i]);  
    free(tablou);  
}
```

```
int randm(int **tablou, int n, int m){  
    srand(time(NULL));  
    int b,max,min;  
    printf("Daca doriti sa introduceti matricea => tastati - 1 \ndaca doriti o matrice umpluta cu ele  
mente aleatorii => tastati - 0\noptiunea aleasa - ");  
    scanf("%d", &b);  
    if (!b){  
        printf("dati limita de sus si jos");  
        scanf("%d %d", &max, &min);  
    }  
    for (int i = 0; i < n; i++)  
    {  
        printf("\n");  
        for (int j = 0; j < m; j++){  
            tablou[i][j] =(rand() % (max - min + 1)) + min;  
        }  
    }  
    return b;  
}
```

Verificarea datelor de ieșire :

Date de intrare	Screenshot
nr de randuri = 3 nr de coloane = 3 optiunea aleasa - 1 enementul [1 1] = 1 enementul [1 2] = 2 enementul [1 3] = 3 enementul [2 1] = 4 enementul [2 2] = 5 enementul [2 3] = 6 enementul [3 1] = 7 enementul [3 2] = 8 enementul [3 3] = 9	 <pre> Lucrarea de laborator nr. 5 Varianta 18 Acest program va citi o matrice si va calcula : suma elementelor pozitive din liniile pare si produsul elementelor mai mici ca 5 din coloanele impare nr de randuri = 3 nr de coloane = 3 Daca doriti sa introduceti matricea => tastati - 1 daca doriti o matrice umpluta cu elemente aleatorii => tastati - 0 optiunea aleasa - 1 elementul [1 1] = 1 elementul [1 2] = 2 elementul [1 3] = 3 elementul [2 1] = 4 elementul [2 2] = 5 elementul [2 3] = 6 elementul [3 1] = 7 elementul [3 2] = 8 elementul [3 3] = 9 rand par 2 : sum = 15 coloana impara 1 : prod = 4 coloana impara 3 : prod = 3 matricea initiala 1 2 3 4 5 6 7 8 9 </pre>
nr de randuri = 3 nr de coloane = 3 optiunea aleasa - 1 enementul [1 1] = 7 enementul [1 2] = 7 enementul [1 3] = 7 enementul [2 1] = 7 enementul [2 2] = 7 enementul [2 3] = 7 enementul [3 1] = 7 enementul [3 2] = 7 enementul [3 3] = 7	 <pre> rand par 2 : sum = 21 coloana impara 1 : conditie nesatisfacuta 0 elemente < 5 coloana impara 3 : conditie nesatisfacuta 0 elemente < 5 matricea initiala 7 7 7 7 7 7 7 7 7 </pre>
nr de randuri = 3 nr de coloane = 3 optiunea aleasa - 1 enementul [1 1] = -2 enementul [1 2] = -2 enementul [1 3] = -1 enementul [2 1] = -3 enementul [2 2] = -4 enementul [2 3] = -5 enementul [3 1] = -6 enementul [3 2] = -7 enementul [3 3] = -8	 <pre> rand par 2 : nici un numare > 0 coloana impara 1 : prod = -36 coloana impara 3 : prod = -40 matricea initiala -2 -2 -1 -3 -4 -5 -6 -7 -8 </pre>

nr de randuri = 6
nr de coloane = 6
optiunea aleasa – 0
dati limita de sus si jos
10 -10

```
rand par 2 : sum = 16
rand par 4 : sum = 22
rand par 6 : sum = 26
coloana impara 1 : prod = 2592
coloana impara 3 : prod = 50
coloana impara 5 : prod = -504

matricea initiala
-8 -4 8 -9 -8 9
-6 -9 10 4 -9 2
-9 -6 10 -9 5 2
-3 7 9 -10 -7 6
2 -6 -5 4 6 2
10 -9 -10 5 5 6
```

Analiza datelor de ieşire :

1. Datele de ieşire sunt calculate ca în programul de la laboratorul precedent singura.
2. Singura deosebire este că la dorinţa utilizatorului programul poate genera o matrice aleatorie cu dimensiunile predefinite de utilizator şi cu limitele de sus şi jos tot setate de utilizator.

În rest:

1. Dacă elementele de pe rândurile pare sunt mai mari ca 0 şi dacă elementele de pe coloanele impare sunt mai mici ca 5 pe ecran vor fi afişate sumele şi produsurile respective.
2. Dacă toate elementele de pe coloanele impare sunt mai mari ca 5 atunci se va afişa că nu a fost nici un număr mai mic ca cinci.
3. Dacă toate elementele de pe rândurile pare sunt negative atunci se va afişa că nici un număr nu este mai mare ca zero.
4. În cazul în care pe coloane există numere mai mari ca 5, se va calcula doar produsul elementelor mai mici ca 5, dacă avem un singur element el se înmulţeşte cu 1 şi se afişează produsul.
5. În cazul când pe rând există elemente negative acestea sunt ignorate în obţinerea sumei.

Concluzii :

În limbajul C programatorul poate alocă sau realoca dinamic memoria, acest lucru este deosebit de important atunci când dorim să schimbăm dimensiunile unui tablou fie acesta bidimensional sau unidimensional. Pentru a alocă memoria dinamic trebuie să utilizăm pointeri. Pointerii către alţi pointeri sunt ceva de genul ca un tablou n dimensional. Funcţiile pot primi ca argumente valori sau pointeri, când argumentul este pointer valoarea variabilei respective va fi schimbată şi în programul principal, iar acest lucru este util când avem nevoie să se returneze mai multe valori, iar acest lucru nu este posibil în nici un limbaj de programare pe care îl cunosc (cu excepţia MATLAB care nu este chiar un limbaj de programare). Avantajul utilizării funcţiilor este că acestea pot fi copiate dintr-un program în altul fără a te îngrijora dacă în programul tău există deja aceste variabile sau ceva ar putea merge rău.