LUCRAREA DE LABORATOR NR. 8

Tema: Prelucrarea tablourilor bidimensionale în C/C++

Obiective:

- să definească noțiunea de tablou bidimensional;
- să aplice algoritmi fundamentali de prelucrare a tablourilor bidimensionale;
- să cunoască algoritmul de citire și afișare a tablourilor bidimensionale;
- să aplice algoritmii fundamentali de prelucrare a tablourilor bidimensionale;
- să realizeze parcurgerea matricelor pe linie, coloană sau diagonală;
- să propună algoritmi de prelucrare a tablourilor bidimensionale.

8.1. Sarcină pentru soluţionare

De creat un tablou cu dimensiunea de 9*9 (a se vedea Figura 8.1) cu completarea sectoarelor matricei: elementele amplasate mai sus şi mai jos de diagonalele principală şi secundară cu ajutorul CL, de la colțul stâng sus spre dreapta.

CL – consecutivitatea liniară numerică (1, 2, 3, ...).

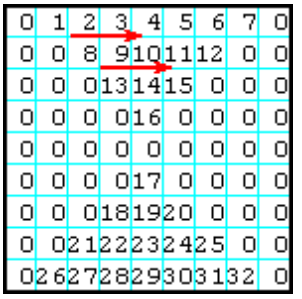

Figura 8.1. Tablou de dimensiunea 9*9

8.2. Exemplu de soluţionare a sarcinii

Elaborarea algoritmului

Fie *D* dimensiunea tabloului, *i* – numărul de linii, *j* – numărul de coloane. Este evident că în linia *i* elementele nenule în partea de sus a tabloului sunt amplasate pe coloanele cu numerele $i < j < D– i–1$, iar în partea de jos $D–i–1 < j < i$. Prin urmare, algoritmul constă în selectarea tabloului rând cu rând şi verificarea fiecărui element, dacă indicii elementului satisfac condiţiile enumerate mai sus, elementului i se atribuie următoarea valoare din CL, în caz contrar – 0.

Se introducem variabila *c* – valoarea curentă a membrului CL cu valoarea inițială 1 (blocul 2). În continuare, se organizează ciclul extern în care se revăd rândurile (blocul 3), iar în ciclul intern – coloanele tabloului (blocul 4). La fiecare iterație a ciclului intern numărul coloanei *j* se compară cu limitele valorilor descrise mai sus. Blocul 5 prevede amplasarea elementului în partea de sus, iar blocul 6 – amplasarea elementului în partea de jos a tabloului. Dacă cel puțin unul dintre cei doi operatori condiționali au valoarea adevăr (TRUE), membrului curent al tabloului i se atribuie valoarea *c*, apoi *c* se incrementează (blocul 7). Dacă nu, membrului curent i se atribuie valoarea 0 (blocul 8).

După ieşirea din ciclul extern, care s-a început în blocul 3, este organizat algoritmul de afișare a tabloului la ecran (blocurile 9-12).

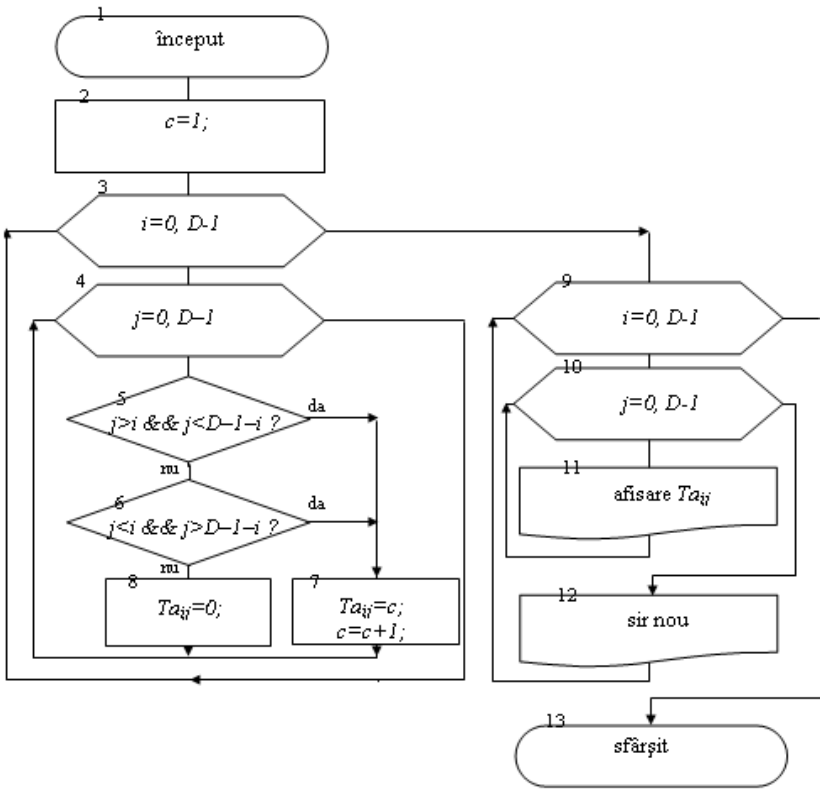

Figura 8.2. Schema bloc

Determinarea variabilelor programului

Pentru realizarea algoritmului, sunt declarate următoarele variabile:

Tabloul se prezintă în memoria statică:

```
int Ta[D][D];
```

Variabilele pentru prezentarea rândurilor *i* şi coloanelor *j*:

```
short i, j;
```

Variabila – membrul curent CL:

```
short c;
```

Elaborarea textului programului

Textul programului începe cu includerea fișierului *stdio.h* şi declararea constantei *D* – dimensiunea tabloului.

Tabloul bidimensional *Ta* este declarat în afara funcției principale, ce asigură amplasarea lui în memoria statică.

Se deschide corpul funcției principale şi se declară variabila *c* cu valoarea inițială 1. Se continuă cu ciclul extern pentru selectarea rândurilor cu modificarea variabilei *i* de la *0* până la *D-1* şi ciclul intern de selectare a coloanelor cu modificarea variabilei *j* de la *0* până la *D-1*.

Instrucțiunea condițională (blocul 5) verifică condiția pentru elementul ce se află în partea de sus a tabloului. Întrucât este necesar ca ambele condiţii să se satisfacă concomitent, ele sunt unite cu operatorul logic &&. La satisfacerea condiției, valoarea *c* se atribuie elementului tabloului cu indicii *[i][j]* şi se incrementează. În caz contrar, elementul este verificat dacă este amplasat în partea de jos al tabloului (blocul 6). La satisfacerea condiției valoarea *c* se atribuie elementului tabloului cu indicii *[i][j]* şi se incrementează. În caz contrar, elementului tabloului se atribuie 0.

Programul urmează cu deschiderea ciclului de afișare a tabloului.

Textul integral al programului este prezentat în Figura 8.3.

[*] Lab10.cpp

```
1  #include <iostream>
2  #define D 9
3  using namespace std;
4  int Ta[D][D];
5
6  int main(){
7      short i, j;
8      short c=1;          //membrul curent CL
9          for(i=0; i<D; i++)
10            for(j=0; j<D; j++)
11            if((j>i) && (j<D-i-1)) Ta[i][j]=c++;
12              else
13              if((j>D-i-1) && (j<i)) Ta[i][j]=c++;
14                 else Ta[i][j]=0;
15     //afisarea tabloului
16     for(i=0; i<D; i++) {
17        for(j=0; j<D; j++){
18         cout<<"\t"<<Ta[i][j];
19        }
20        cout<<endl;
21     }
22     return 0;
23 }
24
```

Figura 8.3. Textul programului

Ajustarea programului

Forma de afișare a rezultatelor programului este destul de vizibilă, întrucât după valorile afișate se poate de stabilit funcționarea corectă a programului sau se pot lua decizii în care ramură a algoritmului s-a comis greșeala. În cazul depistării erorilor, în ajustarea programului se pot urmări indicii curenți a elementelor. Cele mai probabile erori – definirea incorectă a momentului de trecere în sectorul de jos al tabloului.

Rezultatele derulării programului

Rezultatele lucrului programului este adus mai jos:

```
C:\Users\User\Desktop\Indrumar C\Laborator10\Lab10.exe
        0       1       2       3       4       5       6       7       0
        0       0       8       9       10      11      12      0       0
        0       0       0       13      14      15      0       0       0
        0       0       0       0       16      0       0       0       0
        0       0       0       0       0       0       0       0       0
        0       0       0       0       17      0       0       0       0
        0       0       0       18      19      20      0       0       0
        0       0       21      22      23      24      25      0       0
        0       26      27      28      29      30      31      32      0

--------------------------------
Process exited after 0.09465 seconds with return value 0
Press any key to continue . . .
```

Figura 8.4. Rezultatele furnizate de program

8.3. Subiecte de evaluare

Exerciții:

1. Să se implementeze programul din exemplul prezentat.
2. De generat o matrice de valori aliatoare reale cuprinse în intervalul – 1…8.
3. De afișat linia şi coloana pe care se află elementul maxim în matrice.
4. De verificat dacă o matrice pătrată este simetrică pe verticală.
5. De elaborat un program care verifică dacă există două linii ale unei matrice identice.
6. De elaborat un program care calculează suma şi produsul a două matrice pătrate de dimensiunea *n.*
7. Fiind dată o matrice care conține aliator doar valori de 0 şi 1, să se calculeze pe câte linii există o singură valoare nulă.
8. Fiind dată o matrice de numere întregi, să se afișeze numărul liniei cu suma elementelor mai mare.

Verificarea cunoștințelor:

1. Definiți noțiunea de tablou bidimensional.
2. Prezentați exemple de adresare către elementele tabloului bidimensional.
3. Descrieți prioritatea prezentării informaţiei în formă de tablou.
4. Descrieți modul de amplasare în memorie a tablourile bidimensionale.
5. Identificați avantajul prezentării informaţiei în formă de tablou.
6. Relatați despre stabilirea lungimii tabloului la declararea lui.

8.4. Bibliografie

1. Kris Jamsa & Lars Klander, *Totul despre C şi C++*, Editura Teora, Bucureşti, 2006.
2. Herbert Schildt, *C++ manual complet*, Editura Teora, Bucureşti, 1999. Disponibil: https://drive.google.com/file/d/1BrQtITgykcWk03xxtl3Q0-nvcRRFAsy7/view?usp=sharing
3. Tutorials C++ Language. Operators. Disponibil: https://cplusplus.com/doc/tutorial-ro/arrays/