Introduction to Programming

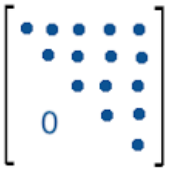
Assignment 1

Department of Electronics and Communication
Indian Institute of Technology, Allahabad

IEC2021042- Shubhasish Malick
IEC2021043- Ritee Sharma
IEC2021044- Ritam Das
IEC2021045- Aryan Anand

Abstract **- In this paper, we have discussed about upper and lower triangular matrices and how to figure out whether the matrix inputted is upper triangular or lower triangular. We have coded our solution using C language.**

Keywords - Upper triangular matrix, Lower triangular matrix, Main diagonal, loop, array, Switch Case.



Upper Triangular Matrix

Lower Triangular Matrix

I. INTRODUCTION

Here we will be discussing how to print upper and lower triangular Matrices. So, first let us know what is a matrix. A matrix is a 2-Dimensional (2-D) array of number arranged in rows and columns. The C programs in this section are based on input and output method. Basic addition and subtraction are involved. An upper triangular matrix is a matrix in which all elements are below the main diagonal is zero. Similarly, in a lower triangular matrix, the elements above the main diagonal are zero.

II. DESCRIPTION

The logic used in the code is fairly simple. In a lower triangular matrix, we will check columns and rows respectively. If the row position is lower than the column position, we will check for the elements which are null and simply count them. For the upper triangular matrix, we will check column and row respectively. If the row position is greater than the column position, we will check for the elements which are null and count them likewise. Now let us understand the algorithm and procedure.

A. *ALGORITHM :*

• Take input from the user for the size of double dimension (2-D) array.
• Define an integer array arr[n][n] and take input from console for the elements of the array arr[n][n].
• Calculate the minimum number of zeros that should be present in any triangular matrix of order “n” by the formula n(n-1)/2 and store it in any variable(number).
• Define a switch case with 2 cases for checking upper triangularity and lower triangularity.
• For checking upper triangularity set conditions to check arr[i][j] ≠ 0,for all (i=j) and arr[i][j]=0,for all (i>j) and set a flag variable (f=1) and counter (c++) variable respectively.
• For checking lower triangularity set conditions to check arr[i][j] ≠ 0,for all (i=j) and arr[i][j]=0,for all (i<j) and set a variable flag(f=1) and counter (C++).
• If the value of counter variable (c)= number and f ≠ 1 then the given matrix is triangular otherwise it is not a triangular matrix.

B. ***PROCEDURE :***

When the compilation begins at first the compiler asks for the order of the 2-D array from the user.

• Enter the size of array: 3 (from console)

• Thus the array, arr[3][3] is now declared.

• Now the compiler asks from the console to enter the elements in array and the elements are entered row-wise : 1 2 3

0 4 8

0 0 6

• Now **p** will store **n-1** i.e, 3-1 = 2.

• Number = nxp/2=3x2/2=3[minimum number of zeros in a triangular matrix].

• Further compiler will ask console to input 1 or 2 for checking upper triangularity or lower triangularity of the matrix respectively.

• “Enter 1 for checking upper triangularity” or “enter 2 for checking lower triangularity” : 1(by console)

• Further compiler will execute case no : 1 of switch case.

• For value of i=1, j will take values from j=0 to j=2 and simultaneously arr[0][0], arr[0][1], arr[0][2] will for be check for value 0 for i=j or i>j accordingly and thus value of “c” and “f ” will be updated for each iteration of inner loop.

• Now if f=1, then the compiler will print “not a triangular matrix” and the compiler will come out of outer loop else i will be incremented to 1 and 2 and thus the other positions like arr[1][0], arr[1][1], arr[1][2], arr[2][0], arr[2][1], arr[2][2] will be checked similarly and value of “c ” and “f ” will be updated.

• Finally it will check that value of c=number condition ,if found true the it will print “ it is a triangular matrix” else it will print “not a triangular matrix”.

C. ***CODE :***

traingularity.c

```c

#include<stdio.h>
int main()
{
    int i,j,option;
    int p;
    int c=0;
    int f = 0;
    int k =0;
    int number; // a variable to count minimum number of zeros in any triangular matrix
    printf("enter the size of the array: ");
    int n;
    scanf("%d",&n);
    int arr[n][n];
    printf("enter the elements in the array \n ");
    for(i=0;i<n;i++)
    {   for(j=0;j<n;j++)
        {
            scanf("%d",&arr[i][j]);
        }
    } p = n-1; // using formula
    number = n*p/2;  // we all know that the minimum number of zero in any triangular matrix of order n is n(n-1)/2

printf("enter 1 for checking upper triangularity of the matrix ");
printf(" \n enter 2 for checking lower triangularity of the matrix ");
scanf("%d",&option);
switch(option)
{
    case 1 : for(i=0;i<n;i++)
             {
                for (j=0;j<n;j++)
                {   if(i==j){
                    if(arr[i][j]==0)
                            f=1;

                }   else if (i>j){
                    if(arr[i][j]==0)
                            c++;
                } else{
                    if(arr[i][j]==0)
                        k++;




                }
                }
                if (f==1){ // to check whether there is a non zero element in main diagonal of the given matrix
                printf("not a triangular matrix");
                 break;
                 }

            } if (c>=number && k!=c){  // check that the counter variable is atleast greater than the minimum number of zeros
                printf("this is a upper triangular matrix");
            } else {
                printf("\n this is not a upper triangular matrix");
            }

        break;
    case 2 : for(i=0;i<n;i++)
             {
                for (j=0;j<n;j++)
                {   if(i==j){
                    if(arr[i][j]==0)
                            f=1;

                }   else if (i<j){
                    if(arr[i][j]==0)
                            c++;
                } else {
                    if(arr[i][j]==0)
                        k++;

                }
                }
                if (f==1){
                printf("not a triangular matrix");
                 break;
                }

            }if (c>=number && k!=c){
                printf("this is a lower triangular matrix");
            } else {
                printf(" \n this is not a lower triangular matrix");
            }
        break;
    default : printf("incorrect option choosen");
}
return 0;
}


```

III. TIME COMPLEXITY

To our program there can be six completely different cases, i.e., there can be 6 different kinds of matrices that can be inputted to which may get the same of different results.

- Upper Triangular matrix of order 3x3: Let the time taken to execute case 1 be t1.
  We take the matrix

$$\begin{vmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{vmatrix}$$

  To check upper triangularity we input 1. The program runs and gives us the result that this is an upper triangular matrix. Time taken to execute this case comes out to be **0.006138s**.

- Lower Triangularity Matrix of order 3x3: Let the time taken to execute case 2 be t2.
  We take the matrix

$$\begin{vmatrix} 5 & 0 & 0 \\ 4 & 4 & 0 \\ 5 & 2 & 3 \end{vmatrix}$$

  To check lower triangularity we input 2. The program runs and gives us the result that this is a lower triangular matrix.
  Time taken to execute this case comes out to be **0.004718s**.

- A Matrix having 0 as its diagonal elements: Let the time taken to execute case 3 be t3.
  We take a matrix:

$$\begin{vmatrix} 0 & 3 & 4 \\ 1 & 0 & 6 \\ 2 & 7 & 0 \end{vmatrix}$$

  Suppose now we check for upper triangularity by inputting 1. The result comes out to be that this is not a triangular matrix.
  Time taken to execute this case is **0.009538s**.

- A 3x3 Non-Triangular Matrix: Let the time taken to execute case 4 be t4.
  We take a matrix:

$$\begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix}$$

  Suppose now we check for lower triangularity by inputting 2. The result shows that this is not a triangular matrix.
  Time taken to execute case 4 comes out to be **0.003591s**.

- A 3x3 Diagonal Matrix: Let the time taken to execute case 5 be t5.
  We take a matrix:

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

  We input 1 to check upper triangularity of this matrix. The result clearly shows us that this is not an upper triangular matrix. The time taken to execute this case comes out to be **0.00338s**.

- A 3x3 matrix having a single non-zero term in its non-diagonal elements: Let the time taken to execute this case be t6.

$$\begin{vmatrix} 1 & 2 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{vmatrix}$$

  We input 1 to check upper triangularity of the matrix. The result tells us that this is an upper triangular matrix.
  The time taken to execute case 6 comes out to be **0.006472s**.

To calculate the time complexity, we take sum of time (corresponding to each case) and divide it by the number of cases.

Hence time complexity = $\frac{t_1+t_2+t_3+t_4+t_5+t_6}{6}$

The time complexity hence comes out to be **0.0056395 seconds**.

CONCLUSION

▪ Our program helps to determine whether the matrix inputted by the user is an upper or lower triangular matrix. Moreover, it is also capable of identifying if the matrix is not a triangular matrix.

▪ It is a very user friendly and menu driven program a gives freedom to the user to check for upper and lower triangularity.

REFERENCES

[1] https://www.codechef.com/ide- For checking the run time of program
[2] https://ncert.nic.in/textbook.php NCERT Class 12 for basic knowledge of matrices and determinants.
[3] Let us C – Yashavant Kanetkar for basic C language theory