Program For Finding Transpose and Determinant of Matrix

Pranav Tiwari
Department of Electronics and Communication
Indian Institute of Information Technology, Allahabad
Mumbai, Maharastra
iec2021048@iiita.ac.in

Sreevardhan
Department of Electronics and Communication
Indian Institute of Information Technology, Allahabad
Chittoor, Andra Pradesh
iec2021049@iiita.ac.in

Shriyansh Lohiya
Department of Electronics and Communication
Indian Institute of Information Technology, Allahabad
Jaipur, Rajasthan
iec2021047@iiita.ac.in

Mohd Kaif
Department of Electronics and Communication
Indian Institute of Information Technology, Allahabad
Amroha, Uttar Pradesh
iec2021046@iiita.ac.in

***Abstract*—Given a random matrix of size m*n, Write a C program which can find the Transpose of the given matrix and later determine the determinant of 2*2 and 3*3 of a given matrix.**

I. Introduction (*Heading 1*)

In order to find the transpose and determinant of the given array first we would have to write a input program followed by a program for finding transpose and at the end a program to find the determinant of the given matrix using the C language.

In order to store the values of the matrix we would make use of a 2-Dimensionnal array and pass it through 2 loops in order to take the input for the said array.

II. 2-Dimentional Arrays

An array is defined as the collection of similar type of data items stored at contiguous memory locations. Arrays are the derived data type in C programming which can store multiple elements of the one primitive data type at a time

In order to initialize an array we need to follow the following statement:-

data_type array_name[x];.

Where data_type is the primitive data type which of which the elements are stored off. Whereas the array_name is used to name and refer the array and the 'x' denotes the number of elements stored in the given array.

Now in order to make a 2-D array, we just need to modify the statement a little bit as given below:-

data_type array_name[x][y];

Here, We can imagine it as a array of an array, i.e each one of the 'x' elements of the given array can store another array of size 'y'. So in total it would store x*y number of elements in the array.

Elements in the 2-Dimensional array are commonly referred as x[i][j] where i denotes the row number of the array and j denotes the column of the array.

III. Do while Loop

A Do while loop is a repetition control structure which allows us to write a loop which runs until the condition in the while statement becomes false. In this kind of loop even when the condition is false the loop would run for one time, where as in While loop the loop would not run at all.

The basic form of declaring a Do while loop is given below:-

```
do
{
   //Statements you want to run in the loop.
} while(condition for the loop to repeat);
```

IV. For Loop

A for loop is used to make a loop executed for given number of times. Unlike Do while loop we can use this loop when we have fix conditions of initialization and some increments after each loop. The basic form of making a for loop is given below:-

```
for(initialization; ccondition ; increment)
{
   //statement you want to execute in the loop.
}
```

V. If Else Statements

If else statements are conditional statements which are used to run statements if they follow a certain condition mentioned by the if else statements. The basic form of writing a if else statement is given below :-

```
If(condition)
{
   //body of the statement.
}
Else if(condition)
{
   //body of the statement.
}
Else{
   //body of the statement.
}
```

Here if none of the given conditions are true, instead of just ending the process the program would run the else statement since it doesn't have any condition. NOTE: You can only write one else statement per if else sereies.

VI. Nesting Of For Loop

Sometimes we feel the need of adding another loop inside a existing loop in order to have 2 elements which changes on their own conditions. For example if one wants to input elements of 3x3 matrix, he/she cannot just go and make a loop for 9 elements, he/she would first have to make a loop for the rows followed by a loop of column such that each each time a row loop is running, the values of columns are the only one changing while row index is remaining constant.

The basic form of declaring a nested loop in C is given below:-

```
for(initialization; condition ; increment)
{
  for(initialization; coniditon; increment)
     {
        //the statements that you want to execute.
      }
}
```

VII. Procedure

In this C program to find the Transpose and Determinant of the given matrix, First we declared the number of rows and columns for the matrix and asked for the input by the user to input values for them in the range of [1,100].

```
   int Matrix_Rows, Matrix_Columns;
      // Code to Assign the number of Rows in the matrix.
   do
   {
      printf("Enter the number of Rows of the Matrix row \n");
      scanf("%d", &Matrix_Rows);
   } while (Matrix_Rows < 1 && Matrix_Rows > 100);

   // Code to Assign the number of Columns in the matrix.

   do
   {
      printf("Enter the number of Columns of the Matrix\n");
      scanf("%d", &Matrix_Columns);
    } while (Matrix_Columns < 1 && Matrix_Columns > 100);
```

After getting the values of rows and columns we took use of Nesting in For loop (V) and asked for user to input values for a matrix of size m*n where m is the number of rows and n is the number of columns in the given matrix.

```
// Intializing the 2D array to store the values in the matrix.
   int Matrix[100][100];

   // Taking the values of the matrix as input.

   for (int i = 0; i < Matrix_Rows; i++)
   {
      for (int j = 0; j < Matrix_Columns; j++)
      {
         printf("Enter the value for the %dth row and %dth column: ", i + 1, j + 1);
         scanf("%d", &Matrix[i][j]);
      }
   }
```

After taking the input from the user and putting it into the given matrix, We made the program to print the given matrix so that the end user can confirm that the matrix he inputed is same as the matrix stored in the system.

```
// Printing the matrix for the end user.

   printf("Given matrix is:\n");

   for (int i = 0; i < Matrix_Rows; i++)

   {

      for (int j = 0; j < Matrix_Columns; j++)

      {

         printf("%d ", Matrix[i][j]);

      }

      printf("\n");

   }
```

Now since we have the matrix, It is not that difficult to find the transpose of the given matrix. All we have to do is just change the indices of the rows and columns such that instead of printing [i][j] form it will print in [j][i] form, Since we know that a transpose of a

matrix is just switching the columns and rows of the matrix.

```
// Function for printing the Transpose of the given matrix.
   printf("Transposed matrix is:\n");
   for (int j = 0; j < Matrix_Columns; j++)
   {
      for (int i = 0; i < Matrix_Rows; i++)
      {
         printf("%d ", Matrix[i][j]);
      }
      printf("\n");
   }
```

From the above program we conclude the first part of the question of finding the transpose of the given matrix.

The next part of the questions deals with finding the determinant of the given matrix. Since the determinant can exceed the limit from int we have made use of long instead of int to make it more versatile. As of now our knowledge is limited by finding the determinant of only the 2x2 and 3x3 matrices, we have only been able to come up with the determinants of those matrices. So in the case of 2x2 matrix we just multiply the diagonal elements and subtract them by each other to find the determinant of the matrix and in the case of 3x3 we use the general formula that we learned in our 11th grade. We also added few extra statements to make the end user understand that if they provided a matrix which is bigger than 3x3 or a non-square matrix, since determinant is only defined for square matrices. The code for the following is given below:-

```
long determinant;
   if (Matrix_Columns == 2 && Matrix_Rows == 2)
   {
      determinant = Matrix[0][0] * Matrix[1][1] - Matrix[1][0] * Matrix[0][1];
      printf("\nThe value of the 2x2 Determinant is : %ld\n", determinant);
   }
   else if (Matrix_Columns == 3 && Matrix_Rows == 3)
   {
      determinant = Matrix[0][0] * ((Matrix[1][1] * Matrix[2][2]) - (Matrix[2][1] *Matrix[1][2])) - Matrix[0][1] * (Matrix[1][0] * Matrix[2][2] - Matrix[2][0] *Matrix[1][2]) + Matrix[0][2]*(Matrix[1][0]* Matrix[2][1] - Matrix[2][0] * Matrix[1][1]);
      printf("\nThe value of the 3x3, Determinant is : %ld\n", determinant);
   }
   else if (Matrix_Rows != Matrix_Columns)
   {
      printf("Since the matrix is not a square matrix, We cannot find the value of the determinant.\n");
   }
   else if (Matrix_Columns > 3 && Matrix_Rows == Matrix_Columns)
   {
      printf("The value of determinant can not be calculated by the given code.\n");
   }
```

VIII. Time Complexity

In the given code the The Time complexity of various steps are given below:-

1.In declaring the number of rows and columns the time complexity is o(1) since it depends on user and assuming that the user gives the correct inputs.

2.In the next step we have taken inputs for the matrix which involves m*n loops i.e its time complexity is o(m*n), Where m is the number of rows and n is the number of columns.

3.For printing the matrix taken as input, it would again take the same amount of operations i.e time complexity is again o(m*n)

4.In the function about printing the transpose of the matrix instead of using m*n operations we are doing n*m operations but since m*n = n*m we have again the same time complexity that is o(m*n)

5.Since we have made use of if else statements and avoided any use of recursion or loops the time complexity for finding the determinant is only o(1).

Since in the above all steps the max time complexity is o(m*n), the time complexity of the program is o(m*n).

IX. Input Taken By the Program

1.The first two lines of input is the number of rows and the number of columns for the given matrix.
2.The next m*n lines of input consist of elements which the user wants to store in the matrix. Where m is the number of rows and n is the number of columns.

X. Conclusion

The output of using the following program is given below:

```
Enter the number of Rows of the Matrix row
3
Enter the number of Columns of the Matrix
3
Enter the value for the 1th row and 1th column: 1
Enter the value for the 1th row and 2th column: 2
Enter the value for the 1th row and 3th column: 3
Enter the value for the 2th row and 1th column: 4
```

```
Enter the value for the 2th row and 2th column: 5
Enter the value for the 2th row and 3th column: 6
Enter the value for the 3th row and 1th column: 7
Enter the value for the 3th row and 2th column: 8
Enter the value for the 3th row and 3th column: 9
Given matrix is:
1 2 3
4 5 6
7 8 9
Transposed matrix is:
1 4 7
2 5 8
3 6 9

The value of the 2x2 Determinant is : 0
```

XI. Program Code

```
#include <stdio.h>
int main()
{

    int Matrix_Rows, Matrix_Columns;
    //Code to Assign the number of Rows in the matrix.
    do
    {
        printf("Enter the number of Rows of the Matrix\n");
        scanf("%d", &Matrix_Rows);
    } while (Matrix_Rows<1 && Matrix_Rows > 100);
    //Code to Assign the number of Columns in the matrix.
    do
    {

        printf("Enter the number of Columns of the Matrix\n");
        scanf("%d", &Matrix_Columns);
    } while (Matrix_Columns < 1 && Matrix_Columns > 100);

    //Intializing the 2D array to store the values in the matrix.
    int Matrix[100][100];

    //Taking the values of the matrix as input.

    for (int i = 0; i < Matrix_Rows; i++)
    {
        for (int j = 0; j < Matrix_Columns; j++)
        {
            printf("Enter the value for the %dth row and %dth column: ",i+1,j+1);
            scanf("%d", &Matrix[i][j]);
        }
    }
    //Printing the matrix for the end user.
    printf("Given matrix is:\n");
    for (int i = 0; i < Matrix_Rows; i++)
    {
        for (int j = 0; j < Matrix_Columns; j++)
        {
            printf("%d ", Matrix[i][j]);
        }
        printf("\n");
    }

    //Function for printing the Transpose of the given matrix.
    printf("Transposed matrix is:\n");
    for (int j = 0; j < Matrix_Columns; j++)
    {
        for (int i = 0; i < Matrix_Rows; i++)
        {
            printf("%d ", Matrix[i][j]);
        }
        printf("\n");
    }

    long determinant;
    if(Matrix_Columns == 2 && Matrix_Rows == 2){
        determinant = Matrix[0][0]*Matrix[1][1] - Matrix[1][0]*Matrix[0][1];
        printf("\nThe value of the 2x2 Determinant is : %d\n", determinant);
    }
    else if(Matrix_Columns == 3 && Matrix_Rows == 3){
```

```
        determinant = Matrix[0][0] *
((Matrix[1][1] * Matrix[2][2]) -
(Matrix[2][1] * Matrix[1][2])) -
Matrix[0][1] * (Matrix[1][0] *
Matrix[2][2] - Matrix[2][0] *
Matrix[1][2]) + Matrix[0][2] *
(Matrix[1][0] * Matrix[2][1] -
Matrix[2][0] * Matrix[1][1]);
        printf("\nThe value of the 3x3
Determinant is : %d\n",determinant);
    }
    else if(Matrix_Rows !=
Matrix_Columns){
        printf("Since the matrix is not
a square matrix, We cannot find the
value of the determinant.\n");
    }
    else if(Matrix_Columns > 3 &&
Matrix_Rows == Matrix_Columns){
        printf("The value of determinant
can not be calculated by the given
code.\n");
    }
    return 0;
}
```