

## 1. 2-D Integer Array

### Aim:

To write a program that reads the elements of a 2D array input by the user and prints the array.

### Algorithm:

Step 1: Start

Step 2: Read a 2D array using a nested for loop with suitable number of rows and columns.

Step 3: Run the nested loop from the beginning to the end and print each element in the 2D array one by one so that the output is in the form of a matrix.

Step 4: End

### Code:

```
//to read and print a 2-D array
#include <stdio.h>

int main(){
    int disp[2][3];
    for (int i=0; i<2; i++){
        for (int j=0; j<3; j++){
            printf("Enter value for disp[%d][%d]: ",i,j);
            scanf("%d",&disp[i][j]);
        }
    }
    for (int i=0; i<2; i++){
        for (int j=0; j<3; j++){
            printf("%d ",disp[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

### Output:

```
> gcc -o q1.o q1.c
> ./q1.o
Enter value for disp[0][0]: 1
Enter value for disp[0][1]: 2
Enter value for disp[0][2]: 3
Enter value for disp[1][0]: 4
Enter value for disp[1][1]: 5
Enter value for disp[1][2]: 6
1 2 3
4 5 6
```

**Result:**

A program for reading and printing a 2D integer array is written and executed.

## 2. Temperatures of Two Cities

### Aim:

To write a program that stores the temperatures of two cities for a week and displays them.

### Algorithm:

Step 1: Start

Step 2: Read the temperature of each city during each day of a week using a nested for loop, for storing the temperatures for two cities during all 7 days of the week in a 2D array.

Step 3: Run the nested loop from the beginning to the end and print each element in the 2D array one by one so that the output displays the daily temperature of each city in the form of a matrix.

Step 4: End

### Code:

```
//to store the temperatures of two cities for a week and display them
#include <stdio.h>

int main(){
    int temp[2][7];
    for (int i=0; i<2; i++){
        for (int j=0; j<7; j++){
            printf("City %d, Day %d: ",i+1,j+1);
            scanf("%d",&temp[i][j]);
        }
    }

    printf("Displaying values:\n");
    for (int i=0; i<2; i++){
        for (int j=0; j<7; j++){
            printf("City %d, Day %d: %d\n",i+1,j+1,temp[i][j]);
        }
    }

    return 0;
}
```

**Output:**

```
> gcc -o q2.o q2.c
> ./q2.o
City 1, Day 1: 33
City 1, Day 2: 34
City 1, Day 3: 35
City 1, Day 4: 33
City 1, Day 5: 32
City 1, Day 6: 31
City 1, Day 7: 30
City 2, Day 1: 23
City 2, Day 2: 22
City 2, Day 3: 21
City 2, Day 4: 24
City 2, Day 5: 22
City 2, Day 6: 25
City 2, Day 7: 26
```

```
Displaying values:
City 1, Day 1: 33
City 1, Day 2: 34
City 1, Day 3: 35
City 1, Day 4: 33
City 1, Day 5: 32
City 1, Day 6: 31
City 1, Day 7: 30
City 2, Day 1: 23
City 2, Day 2: 22
City 2, Day 3: 21
City 2, Day 4: 24
City 2, Day 5: 22
City 2, Day 6: 25
City 2, Day 7: 26
```

**Result:**

A program for storing the temperatures of two cities for a week is written and executed.

### 3. Matrix Sum

**Aim:**

To write a program that reads two matrices and prints their sum.

**Algorithm:**

Step 1: Start

Step 2: Read two 2D arrays using a nested for loop with suitable number of rows and columns. The numbers in the array are to be entered as input from the user.

Step 3: Run a nested loop with the appropriate number of rows and columns and print **the sum of the elements in the same positions** in the two 2D arrays one by one so that the output is in the form of a matrix.

Step 4: End

**Code:**

```
//to find the sum of two matrices of order 2*2
#include <stdio.h>

int main(){
    float a[2][2], b[2][2];
    printf("Enter elements of first matrix.\n");
    for (int i=0; i<2; i++){
        for (int j=0; j<2; j++){
            printf("Enter a%d%d: ",i+1,j+1);
            scanf("%f",&a[i][j]);
        }
    }
    printf("Enter elements of second matrix.\n");
    for (int i=0; i<2; i++){
        for (int j=0; j<2; j++){
            printf("Enter b%d%d: ",i+1,j+1);
            scanf("%f",&b[i][j]);
        }
    }

    printf("Sum of Matrices:\n");
    for (int i=0; i<2; i++){
        for (int j=0; j<2; j++){
            printf("%.2f ",a[i][j]+b[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

**Output:**

```
> gcc -o q3.o q3.c
> ./q3.o
Enter elements of first matrix.
Enter a11: 2
Enter a12: 0.5
Enter a21: -1.1
Enter a22: 2
Enter elements of second matrix.
Enter b11: 0.2
Enter b12: 0
Enter b21: 0.23
Enter b22: 23
Sum of Matrices:
2.20 0.50
-0.87 25.00
```

**Result:**

A program for finding the sum of two matrices is written and executed.

## 4. Sum of Matrix Elements

### Aim:

To write a program that reads a 2D array and prints the sum of the matrix elements.

### Algorithm:

Step 1: Start

Step 2: Define a function *sum* to calculate the sum of the array elements. The function takes a 2D array as the parameter and does the following:

- Initialise a variable calculating the sum to 0.
- Run a nested for loop through the elements of the 2D array and add each element to the sum.
- After the loop is terminated, return the sum.

Step 3: Read a 2D array using a nested for loop with suitable number of rows and columns.

Step 4: Call the function *sum* with the array as the argument and print the return value.

Step 5: End

### Code:

```
//to calculate the sum of array elements by passing the array as argument to a function
#include <stdio.h>
```

```
float sum(float arr[2][2]){
    float s=0;
    for (int i=0; i<2; i++){
        for (int j=0; j<2; j++){
            s+=arr[i][j];
        }
    }
    return s;
}

int main(){
    float num[2][2];
    for (int i=0; i<2; i++){
        for (int j=0; j<2; j++){
            printf("Enter num%d%d: ",i+1,j+1);
            scanf("%f",&num[i][j]);
        }
    }

    printf("Sum of matrix elements: %.2f\n",sum(num));

    return 0;
}
```

**Output:**

```
> gcc -o q4.o q4.c
> ./q4.o
Enter num11: 0.2
Enter num12: 3.5
Enter num21: 31
Enter num22: 12.8
Sum of matrix elements: 47.50
```

**Result:**

A program to find the sum of matrix elements is written and executed.



## 5. Passing a 2D Array to a Function

### Aim:

To write a program to read a 2D array and pass it as an argument to a function before printing it.

### Algorithm:

Step 1: Start

Step 2: Define a function *print* that prints the 2D array. The function should take a 2D array as its parameter and do the following:

- Run a nested for loop through the elements of the 2D array and print each element one by one.
- Go to a new line after every row of elements is printed.

Step 3: Read a 2D array using a nested for loop with suitable number of rows and columns.

Step 4: Call the function *print* passing the 2D array as the argument to the function.

Step 5: End

### Code:

```
//to read a 2D array and pass it to a function, then print it
#include <stdio.h>
```

```
void print(float arr[2][2]){
    printf("Displaying values:\n");
    for (int i=0; i<2; i++){
        for (int j=0; j<2; j++){
            printf("num%d%d: %.2f\n",i+1,j+1,arr[i][j]);
        }
    }
}
```

```
int main(){
    float num[2][2];
    for (int i=0; i<2; i++){
        for (int j=0; j<2; j++){
            printf("Enter num%d%d: ",i+1,j+1);
            scanf("%f",&num[i][j]);
        }
    }

    print(num);

    return 0;
}
```

**Output:**

```
> gcc -o q5.o q5.c
> ./q5.o
Enter num11: 1.2
Enter num12: 2.3
Enter num21: 3.4
Enter num22: 4.5
Displaying values:
num11: 1.20
num12: 2.30
num21: 3.40
num22: 4.50
```

**Result:**

A program for passing a 2D array to a function and printing it is written and executed.

## 6. Matrix Multiplication

### Aim:

To write a program that performs matrix multiplication for two arbitrary matrices.

### Algorithm:

Step 1: Start

Step 2: Read the number of rows (m) and number of columns (n) of each of the two matrices as input from the user.

Step 3: If n1 is not equal to m2, then state that the multiplication is not possible and quit the program. Else, proceed.

Step 4: Read two matrices using a nested for loop with suitable number of rows and columns. The elements of the matrices are to be entered as input from the user.

Step 5: Read a third matrix (product matrix)

Step 6: Run a nested for loop following m1 number of rows (outer i iterations) and n2 number of columns (inner j iterations).

Step 7: For each iteration of the inner loop, do the following:

- Set the value of the corresponding element of the product matrix to 0.
- Run a for loop n1 times (k=0 to k=n1) and add the **product of the k<sup>th</sup> element in the i<sup>th</sup> row of the first matrix to and the k<sup>th</sup> element in the j<sup>th</sup> column of the second matrix.**

Step 8: Run the nested loop (m1 rows and n2 columns) from the beginning to the end and print each element in the product matrix one by one so that the output is in the form of a matrix.

Step 9: End

### Code:

```
//matrix multiplication
#include <stdio.h>

int main(){
    int m1,n1,m2,n2;
    float a[m1][n1],b[m2][n2],c[m1][n2];

    printf("Enter number of rows and cols in matrix 1: ");
    scanf("%d %d", &m1,&n1);
    printf("Enter number of rows and cols in matrix 2: ");
    scanf("%d %d", &m2,&n2);

    if (n1==m2){
        printf("Enter elements of first matrix.\n");
        for (int i=0; i<m1; i++){
            for (int j=0; j<n1; j++){
                printf("Enter a%d%d: ",i+1,j+1);
                scanf("%f",&a[i][j]);
            }
        }
        printf("Enter elements of second matrix.\n");
        for (int i=0; i<m2; i++){
```

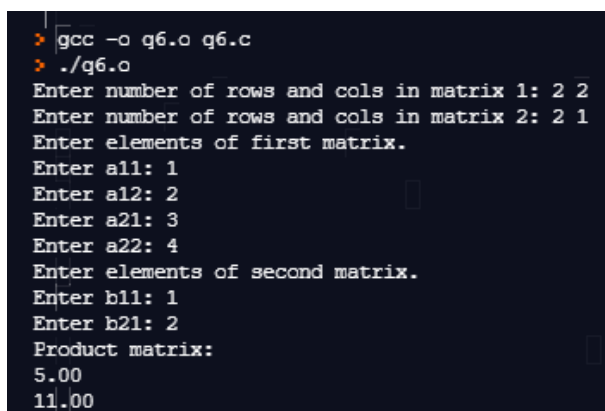
```
for (int j=0; j<n2; j++){
    printf("Enter b%d%d: ",i+1,j+1);
    scanf("%f",&b[i][j]);
}
}

printf("Product matrix:\n");
for (int i=0; i<m1; i++){
    for (int j=0; j<n2; j++){
        c[i][j]=0;
        for (int k=0; k<n1; k++){
            c[i][j]+=a[i][k]*b[k][j];
        }
    }
}

for (int i=0; i<m1; i++){
    for (int j=0; j<n2; j++){
        printf("%.2f ",c[i][j]);
    }
    printf("\n");
}

else{
    printf("Multiplication is not possible.");
}

return 0;
}
```

**Output:**

```
> gcc -o q6.o q6.c
> ./q6.o
Enter number of rows and cols in matrix 1: 2 2
Enter number of rows and cols in matrix 2: 2 1
Enter elements of first matrix.
Enter a11: 1
Enter a12: 2
Enter a21: 3
Enter a22: 4
Enter elements of second matrix.
Enter b11: 1
Enter b21: 2
Product matrix:
5.00
11.00
```

**Result:**

A program for finding the product of two matrices is written and executed.