

Wednesday, 09 February 2022

Day-13 Assignment 13

C# 2D-ARRAY, STACK, QUEUE.
NALLI PRUDHVI

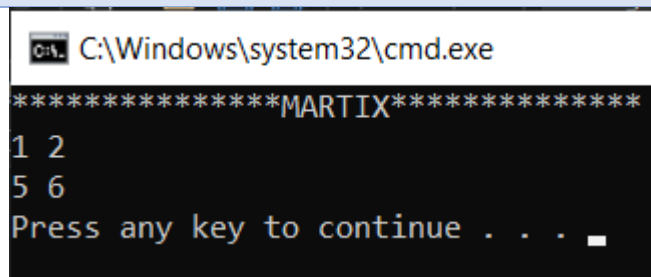
Q. Declare a 2 dimensional array of size (2,2) and initialize using indexes and print the values using nested for loop

CODE:

```
internal class Program
{
    static void Main(string[] args)
    {
        int[,] frame = new int[,] { { 1, 2 }, { 5, 6 } };

        for (int i = 0; i < frame.GetLength(0); i++)
        {
            for (int j = 0; j < frame.GetLength(1); j++)
            {
                Console.Write(frame[i, j] + " ");
            }
            Console.WriteLine();
        }
    }
}
```

OUTPUT



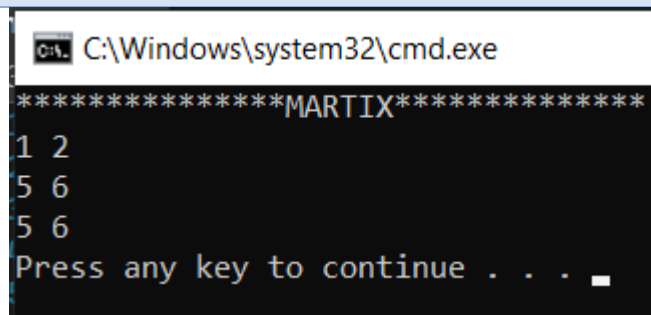
```
C:\Windows\system32\cmd.exe
*****MARTIX*****
1 2
5 6
Press any key to continue . . .
```

Q. Declare a 2-D array of size (3,2) and initialize in the same line while declaring and print the values using nested for loop

CODE

```
static void Main(string[] args)
{
    int[,] frame = new int[3,2]{ { 1, 2 }, { 5, 6 }, { 5, 6 } };
    Console.WriteLine("*****MARTIX*****");
    for (int i = 0; i < frame.GetLength(0); i++)
    {
        for (int j = 0; j < frame.GetLength(1); j++)
        {
            Console.Write(frame[i, j] + " ");
        }
        Console.WriteLine();
    }
}
```

OUTPUT



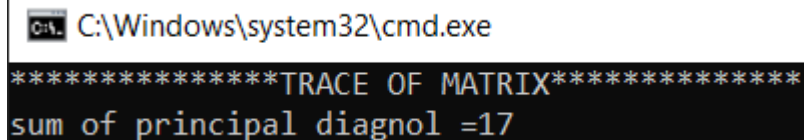
```
C:\Windows\system32\cmd.exe
*****MARTIX*****
1 2
5 6
5 6
Press any key to continue . . .
```

Q. Declare a 2-D array of size (3,3) and print trace of the array

CODE

```
static void Main(string[] args)
{
    int[,] frame = new int[,] { { 1, 2, 3 }, { 5, 6, 7 }, { 8, 9, 10 } };
    Console.WriteLine("*****TRACE OF MATRIX*****");
    int SUM = 0;
    for (int i = 0; i < frame.GetLength(0); i++)
    {
        for (int j = 0; j < frame.GetLength(1); j++)
        {
            if (i == j)
            {
                SUM += frame[i, j];
            }
        }
    }
    Console.WriteLine($"sum of principal diagonl = {SUM}");
    Console.ReadLine();
}
```

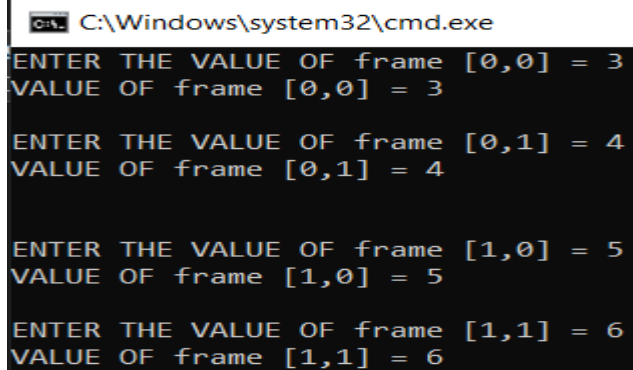
OUTPUT



Q. Declare a 2-D array of size (2,2) and read values from user and print the array values.

CODE

OUTPUT



Q. Declare TWO 2-D arrays of size (2,2) and read values from user and print the sum of the two matrices.

CODE

```
static void Main(string[] args)
{
    int[,] frame = new int[2, 2];
    int[,] frame_1 = new int[2, 2];
    int[,] frame_2 = new int[2, 2];

    for (int i = 0; i < frame.GetLength(0); i++)
    {
        for (int j = 0; j < frame.GetLength(1); j++)
        {
            Console.Write($"ENTER THE VALUE OF frame [{i},{j}] = ");
            frame[i, j] = Convert.ToInt32(Console.ReadLine());
            Console.Write($"VALUE OF frame [{i},{j}] = {frame[i, j]}");
            Console.WriteLine();
        }
    }
}
```

```

    }
    Console.WriteLine();
}
for (int i = 0; i < frame_1.GetLength(0); i++)
{
    for (int j = 0; j < frame_1.GetLength(1); j++)
    {
        Console.Write($"ENTER THE VALUE OF frame_1 [{i},{j}] = ");
        frame_1[i, j] = Convert.ToInt32(Console.ReadLine());
        Console.Write($"VALUE OF frame_1 [{i},{j}] = {frame_1[i, j]}");
        Console.WriteLine("\n\n");
    }
    Console.WriteLine();
}
for (int i = 0; i < frame_2.GetLength(0); i++)
{
    for (int j = 0; j < frame_2.GetLength(1); j++)
    {
        int v = frame[i, j] + frame_1[i, j];
        frame_2[i, j] = v;
    }
    Console.WriteLine();
}
Console.WriteLine("*****ADDITION OF MATRIX*****");
for (int i = 0; i < frame_2.GetLength(0); i++)
{
    for (int j = 0; j < frame_2.GetLength(1); j++)
    {
        Console.Write(frame_2[i, j] + " ");
    }
    Console.WriteLine();
}

Console.ReadLine();
}

```

OUTPUT

C:\Windows\system32\cmd.exe

ENTER THE VALUE OF frame [0,0] = 10

VALUE OF frame [0,0] = 10

ENTER THE VALUE OF frame [0,1] = 20

VALUE OF frame [0,1] = 20

ENTER THE VALUE OF frame [1,0] = 20

VALUE OF frame [1,0] = 20

ENTER THE VALUE OF frame [1,1] = 40

VALUE OF frame [1,1] = 40

ENTER THE VALUE OF frame_1 [0,0] = 50

VALUE OF frame_1 [0,0] = 50

ENTER THE VALUE OF frame_1 [0,1] = 60

VALUE OF frame_1 [0,1] = 60

ENTER THE VALUE OF frame_1 [1,0] = 70

VALUE OF frame_1 [1,0] = 70

ENTER THE VALUE OF frame_1 [1,1] = 80

VALUE OF frame_1 [1,1] = 80

*****ADDITION OF MATRIX*****

60 80

90 120

Q. Declare TWO 2-D arrays of size (2,2) and read values from user and print the product of the two matrices.

CODE

```
static void Main(string[] args)
{
    int[,] frame = new int[2, 2];
    int[,] frame_1 = new int[2, 2];
    int[,] frame_2 = new int[2, 2];
    //*****DECLARING MY FIRST MATRIX*****
    int i = 0, j = 0, FR1_L = frame.GetLength(0), FR2_L = frame.GetLength(1);
    while (i < FR1_L)
    {
        while (j < FR2_L)
        {
            Console.Write($"ENTER THE VALUE OF frame [{i},{j}] = ");
            frame[i, j] = Convert.ToInt32(Console.ReadLine());
            Console.Write($"VALUE OF frame [{i},{j}] = {frame[i, j]}");
```

```

        Console.WriteLine("\n\n");
        j++;
    }
    j -= j;
    Console.WriteLine();
    i++;
}
//*****DECLARING MY SECOND MATRIX*****
int k = 0, l = 0, SR1_L = frame_1.GetLength(0), SR2_L = frame_1.GetLength(1);
while (k < SR1_L)
{
    while (l < SR2_L)
    {
        Console.WriteLine($"ENTER THE VALUE OF frame_1 [{k},{l}] = ");
        frame_1[k, l] = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine($"VALUE OF frame [{k},{l}] = {frame_1[k, l]}");
        Console.WriteLine("\n\n");
        l++;
    }
    l -= l;
    Console.WriteLine();
    k++;
}
//*****VAR_FOR_PRODUCT_OF_MATRIX.*****
int u = 0, v = 0, w = 0, TR1_L = frame_2.GetLength(0), TR2_L =
frame_2.GetLength(1);
//*****MATRIX_PRODUCT_CONDITION.*****
int NEIF = frame.Length, CFM = NEIF / FR1_L;
if (NEIF == SR2_L)
{
    Console.WriteLine("Scalar matrix can't be multiplied.");
}
//*****MATRIX_PRODUCT.*****
else
{
    while (u < TR1_L)
    {
        while (v < TR2_L)
        {
            frame_2[u, v] = 0;
            while (w < TR2_L)
            {
                frame_2[u, v] += frame[u, w] * frame_1[w, v];
                w++;
            }
            w -= w;
            v++;
        }
        v -= v;
        u++;
    }
}
Console.WriteLine("*****MULTIPLICATION OF MATRIX*****");
for (int o = 0; o < frame_2.GetLength(0); o++)
{
    for (int p = 0; p < frame_2.GetLength(1); p++)
    {
        Console.Write(frame_2[o, p] + " ");
    }
    Console.WriteLine();
}
Console.ReadLine();
}

```

OUTPUT

```
C:\Windows\system32\cmd.exe
ENTER THE VALUE OF frame [0,0] = 2
VALUE OF frame [0,0] = 2

ENTER THE VALUE OF frame [0,1] = 2
VALUE OF frame [0,1] = 2

ENTER THE VALUE OF frame [1,0] = 2
VALUE OF frame [1,0] = 2

ENTER THE VALUE OF frame [1,1] = 2
VALUE OF frame [1,1] = 2

ENTER THE VALUE OF frame_1 [0,0] = 2
VALUE OF frame [0,0] = 2

ENTER THE VALUE OF frame_1 [0,1] = 2
VALUE OF frame [0,1] = 2

ENTER THE VALUE OF frame_1 [1,0] = 2
VALUE OF frame [1,0] = 2

ENTER THE VALUE OF frame_1 [1,1] = 2
VALUE OF frame [1,1] = 2

*****MULTIPLICATION OF MATRIX*****
8 8
8 8
```

Q What is a jagged array, What is the benefit of jagged array.

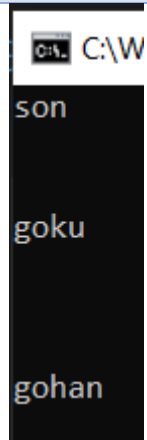
- A.** The elements of a jagged array in C# are arrays of different dimensions and sizes. A jagged array is sometimes called an "array of arrays." A special type of array is introduced in C#.
- In C# make use of jagged arrays, which are faster. A jagged array stores unequal data more efficiently. Arrays with jagged edges. Data arrives in a variety of forms. Sometimes the shape is uneven. With jagged arrays, we can store efficiently many rows of varying lengths. Any type of data, reference or value, can be used.

Q. WACP to declare a jagged array and print values

CODE

```
static void Main(string[] args)
{
    string[][] arr = new string[3][];
    arr[0] = new string[3];
    arr[1] = new string[4];
    arr[2] = new string[5];
    arr[0][0] = "son";
    arr[1][0] = "goku";
    arr[2][0] = "gohan";
    for(int i = 0; i < 3; i++)
    {
        for (int j = 0; j < arr[i].Length; j++)
        {
            Console.WriteLine(arr[i][j]);
        }
    }
    Console.ReadLine();
}
```

OUTPUT



```
C:\W
son
goku
gohan
```

Q. What is Recursion? What are the benefits of recursion?

- A. When a recursive solution makes the code simpler and easier to follow, recursion is the ideal choice. Iteration is best employed when a recursive solution does not significantly simplify the programme or when a recursive solution is horribly wasteful. A binary search for a binary tree is a nice example of recursion.

When a recursive solution makes the code simpler and easier to read, recursion is the ideal option. Iteration is best employed when a recursive solution does not significantly simplify the programme or when a recursive solution is extremely inefficient. A binary search for a binary tree is an excellent example of recursion.

Q. WACP to illustrate usage of Recursion.

CODE

```
class Program
{
    public static int fact(int n)
    {
        if (n == 0 || n==1)
            return 1;
        else
            return n*fact(n-1);
    }
}
```

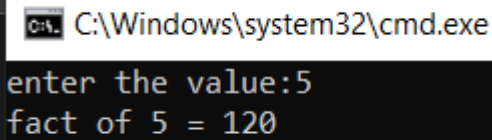


```

    }
    static void Main(string[] args)
    {
        Console.Write("enter the value:");
        int a = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine($"fact of {a} = {Program.fact(a)}");
        Console.ReadLine();
    }
}

```

OUTPUT



```

C:\Windows\system32\cmd.exe
enter the value:5
fact of 5 = 120

```

Q. WACP to illustrate usage of Stack<>

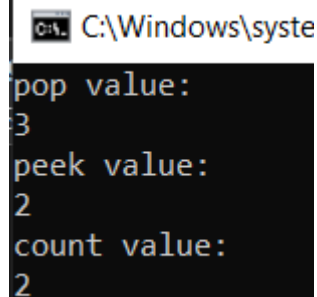
CODE

```

static void Main(string[] args)
{
    Stack<int> data = new Stack<int>();
    data.Push(1);
    data.Push(2);
    data.Push(3);
    Console.WriteLine("pop value:");
    Console.WriteLine(data.Pop());
    Console.WriteLine("peek value:");
    Console.WriteLine(data.Peek());
    Console.WriteLine("count value:");
    Console.WriteLine(data.Count());
    Console.ReadLine();
}

```

OUTPUT



```

C:\Windows\system32\cmd.exe
pop value:
3
peek value:
2
count value:
2

```

Q. Write couple of points about Stack.

- A. The Generic Stack in C# is a collection class which works on the principle of Last in First out (LIFO) and this class is present in System. Collections. Generic namespace. This Stack collection class is analogous to a stack of plates.

Q. WACP to illustrate usage of Queue.

CODE

```
static void Main(string[] args)
{
    Queue<int> data = new Queue<int>();
    data.Enqueue(1);
    data.Enqueue(2);
    data.Enqueue(3);

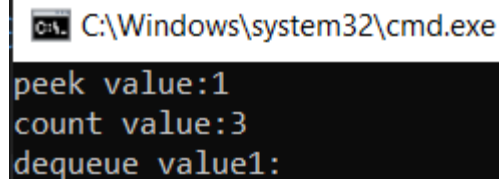
    int a = data.Peek();
    Console.WriteLine($"peek value:{a}");

    int b = data.Count();
    Console.WriteLine($"count value:{b}");

    int c = data.Dequeue();
    Console.WriteLine($"dequeue value{c}:");

    Console.ReadLine();
}
```

OUTPUT



```
C:\Windows\system32\cmd.exe
peek value:1
count value:3
dequeue value1:
_
```

Q. Write couple of points about Queue.

- A. In C#, The Generic Queue is a collection class which works on the principle of First in First out (FIFO) and this class is present in System.Collections. Namespace that is generic. The Queue collection class is analogous to a queue at the ATM machine to withdraw money.

- *Thank you*