

NB Healthcare Technologies Pvt Ltd

Day 13 Morning Assignment (09 – Feb- 2022)

By

Vamsi Krishna Mandapati

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

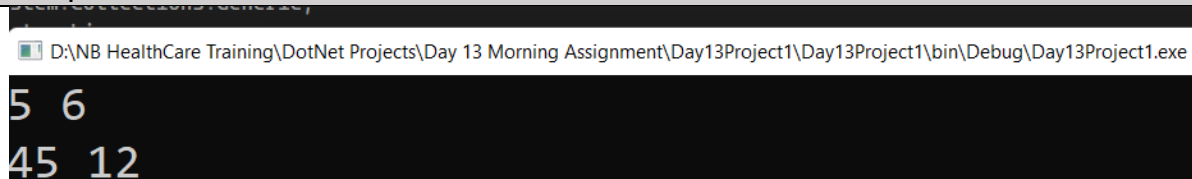
Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day13Project1
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int[,] data = new int[2, 2];
            data[0, 0] = 5;
            data[0, 1] = 6;
            data[1, 0] = 45;
            data[1, 1] = 12;

            for(int i = 0; i<2; i++)
            {
                for(int j=0;j<2; j++)
                {
                    Console.Write(data[i,j]+ " ");
                }
                Console.WriteLine("\n");
            }
            Console.ReadLine();
        }
    }
}
```

Output:



```
D:\NB HealthCare Training\DotNet Projects\Day 13 Morning Assignment\Day13Project1\Day13Project1\bin\Debug\Day13Project1.exe
5 6
45 12
```

2. 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

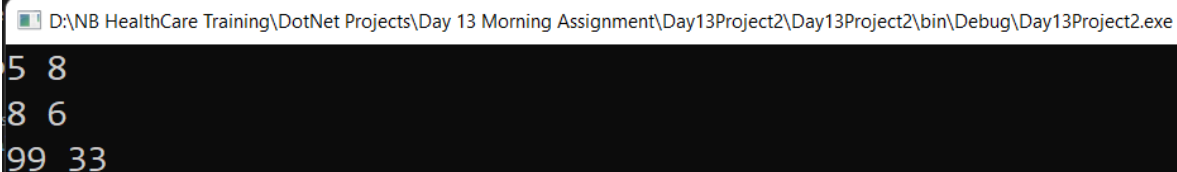
Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day13Project2
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int[,] data = new int[,] { { 5, 8 }, { 8, 6 }, { 99, 33 } };

            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 2; j++)
                {
                    Console.Write(data[i, j] + " ");
                }
                Console.WriteLine("\n");
            }
            Console.ReadLine();
        }
    }
}
```

Output:



```
D:\NB HealthCare Training\DotNet Projects\Day 13 Morning Assignment\Day13Project2\Day13Project2\bin\Debug\Day13Project2.exe
5 8
8 6
99 33
```

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

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day13Project3
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int[,] data = new int[,] { { 55,23,4}, { 58,36,56}, { 89,47,23} };
        }
    }
}
```

```

        int sum = 0;

        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                if (i == j)
                {
                    sum = sum + data[i, j];
                }
            }
        }
        Console.WriteLine("Sum :" + sum);
        Console.ReadLine();
    }
}

```

Output:

D:\NB HealthCare Training\DotNet Projects\Day 13 Morning Assignment\Day13Project3\Day13Project3\bin\Debug\Day13Project3.exe

Sum :114

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

Code:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day13Project4
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int[,] data = new int[2, 2];

            //Read from user
            for(int i =0; i < 2; i++)
            {
                for(int j=0; j < 2; j++)
                {
                    Console.WriteLine("Enter array value at {0},{1}:",i,j);
                    data[i, j] = Convert.ToInt32(Console.ReadLine());
                }
            }

            //Print the values
            for (int i = 0; i < 2; i++)

```

```

        {
            for (int j = 0; j < 2; j++)
            {
                Console.Write(data[i,j] + " "); //Console.Write(); will print in
same line.

            }
            Console.WriteLine();
            Console.ReadLine();

            //Console.WriteLine(); will print in next line
            //or
            //Console.WriteLine("\n"); will print in next line
        }
    }
}

```

Output:

D:\NB HealthCare Training\DotNet Projects\Day 13 Morning Assignment\Day13Project4\Day13Project4\bin\Debug\Day13Project4.exe

```

Enter array value at 0,0:
56
Enter array value at 0,1:
36
Enter array value at 1,0:
89
Enter array value at 1,1:
47
56 36

89 47

```

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

Code:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day3Project5
{
    internal class Program
    {
        static void Main(string[] args)
        {

```

```

int[,] a = new int[2, 2];

//Read from user
for (int i = 0; i < 2; i++)
{
    for (int j = 0; j < 2; j++)
    {
        Console.WriteLine("Enter first array value at {0},{1}:", i, j);
        a[i, j] = Convert.ToInt32(Console.ReadLine());
    }
}

int[,] b = new int[2, 2];

//Read from user
for (int i = 0; i < 2; i++)
{
    for (int j = 0; j < 2; j++)
    {
        Console.WriteLine("Enter second array value at {0},{1}:", i, j);
        b[i, j] = Convert.ToInt32(Console.ReadLine());
    }
}

Console.WriteLine("Sum of Two 2D Arrays is :");

int[,] result = new int[2, 2];

//Print the values
for (int i = 0; i < 2; i++)
{
    for (int j = 0; j < 2; j++)
    {
        result[i, j] = a[i, j] + b[i, j];
        Console.Write(result[i, j] + " ");
    }
    Console.WriteLine();
}

Console.ReadLine();
}
}
}

```

Output:

D:\NB HealthCare Training\DotNet Projects\Day 13 Morning Assignment\Day3Project5\Day3Project5\bin\Debug\Day3Project5.exe

```
Enter first array value at 0,0:
56
Enter first array value at 0,1:
32
Enter first array value at 1,0:
69
Enter first array value at 1,1:
41
Enter second array value at 0,0:
3
Enter second array value at 0,1:
23
Enter second array value at 1,0:
23
Enter second array value at 1,1:
12
Sum of Two 2D Arrays is :
59 55
92 53
```

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

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day3Project6
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int[,] a = new int[2, 2];

            //Read from user
            for (int i = 0; i < 2; i++)
            {
                for (int j = 0; j < 2; j++)
                {
                    Console.WriteLine("Enter first array value at {0},{1}:", i, j);
                    a[i, j] = Convert.ToInt32(Console.ReadLine());
                }
            }
        }
    }
}
```

```

    }
}

int[,] b = new int[2, 2];

//Read from user
for (int i = 0; i < 2; i++)
{
    for (int j = 0; j < 2; j++)
    {
        Console.WriteLine("Enter second array value at {0},{1}:", i, j);
        b[i, j] = Convert.ToInt32(Console.ReadLine());
    }
}

Console.WriteLine("Product of Two 2D Arrays is :");

int[,] result = new int[2, 2];

//Print the values
for (int i = 0; i < 2; i++)
{
    for (int j = 0; j < 2; j++)
    {
        result[i, j] = a[i, j] * b[i, j];

        Console.Write(result[i, j] + " ");

    }
    Console.WriteLine();
}

Console.ReadLine();
}
}
}

```

Output:

D:\NB HealthCare Training\DotNet Projects\Day 13 Morning Assignment\Day3Project6\Day3Project6\bin\Debug\Day3Project6.exe

```
Enter first array value at 0,0:
58
Enter first array value at 0,1:
63
Enter first array value at 1,0:
12
Enter first array value at 1,1:
45
Enter second array value at 0,0:
63
Enter second array value at 0,1:
98
Enter second array value at 1,0:
23
Enter second array value at 1,1:
47
Product of Two 2D Arrays is :
3654 6174
276 2115
```

7. What is a jagged array

What is the benefit of jagged array

Jagged Array:

- A jagged array is two dimensional array, which will have different sizes for different rows.
- The Way you declare jagged array is different when compared to normal array.
- In jagged array we don't mention the size of the column.

Benefit of the Jagged array:

- The Benefit of the jagged array is, it saves the Memory.

Example:

Normal 2D Array Declaration: `char[,] names = new char[3, 8];`

Jagged 2D Array Declaration:

```
char[][] names = new char[3][];  
names[0] = new char[] { 'v', 'a', 'm', 's',  
'i', 'k', 'r', 'i', 's', 'h', 'n', 'a' };  
names[1] = new char[] { 'p', 'a', 'v', 'a', 'n' };  
names[2] = new char[] { 'm', 'a', 'n', 'o', 'j', 'k' };
```

8. WACP to declare a jagged array and print values

Code:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace Day13Project7  
{  
    internal class Program  
    {  
        static void Main(string[] args)  
        {  
            char[][] names = new char[3][];  
  
            names[0] = new char[] { 'v', 'a', 'm',  
's', 'i', 'k', 'r', 'i', 's', 'h', 'n', 'a' };  
            names[1] = new char[] { 'p', 'a', 'v', 'a', 'n' };  
            names[2] = new char[] { 'm', 'a', 'n', 'o', 'j', 'k' };  
  
            for(int i =0; i < 3; i++)  
            {  
                for(int j = 0; j < names[i].Length; j++)  
                {  
                    Console.Write(names[i][j]);  
                }  
                Console.WriteLine();  
            }  
  
            Console.ReadLine();  
        }  
    }  
}
```

Output:

D:\NB HealthCare Training\DotNet Projects\Day 13 Morning Assignment\Day13Project7\Day13Project7\bin\Debug\Day13Project7.exe

```
vamsikrishna  
pavan  
manojk  
-
```

9. What is Recursion

Recursion:

- A function calling itself repeatedly until a specified condition is satisfied.
- The recursive function or method is a very strong functionality in C#.
- A recursive method is **a method which calls itself again and again on basis of few statements which need to be true.**
- Similarly, when a function calls itself again and again it is known as a recursive function

10. WACP to illustrate usage of Recursion.

What are the benefits of recursion

Benefits of Recursion:

- Recursion **adds clarity and (sometimes) reduces the time needed to write and debug code** (but doesn't necessarily reduce space requirements or speed of execution).
- Reduces time complexity.
- Performs better in solving problems based on tree structures

Code:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace Day13Project8  
{  
  
    internal class Program
```

```

{

    static int Factorial(int n)
    {
        if(n == 0)
        {
            return 1;
        }
        else
        {
            return n * Factorial(n - 1);
        }
    }

    static void Main(string[] args)
    {
        int n;
        Console.WriteLine("Enter any Number:");
        n = Convert.ToInt32(Console.ReadLine());

        Console.WriteLine("Factorial of {0} is {1}:", n, Factorial(n));

        Console.ReadLine();
    }
}

```

Output:

D:\NB HealthCare Training\DotNet Projects\Day 13 Morning Assignment\Day13Project8\Day13Project8\bin\Debug\Day13Project8.exe

```

Enter any Number:
8
Factorial of 8 is 40320:

```

11. WACP to illustrate usage of Stack<>

Write couple of points about Stack

Stack:

- `Stack` is a special type of collection that stores elements in LIFO style (**Last In First Out**).
- C# includes the generic `Stack<T>` and non-generic `Stack` collection classes.
- It is recommended to use the generic `Stack<T>` collection.
- Stack is useful to store temporary data in LIFO style, and you might want to delete an element after retrieving its value.

➤ Example : `Stack<int> data = new Stack<int>();`

Stack<T> Characteristics:

- `Stack<T>` is Last In First Out collection.
- It comes under `System.Collections.Generic` namespace.
- `Stack<T>` can contain elements of the specified type. It provides compile-time type checking and doesn't perform boxing-unboxing because it is generic.
- Elements can be added using the `Push()` method. Cannot use collection-initializer syntax.
- Elements can be retrieved using the `Pop()` and the `Peek()` methods. It does not support an indexer.

Stack<T> Properties and Methods:

Property	Usage
Count	Returns the total count of elements in the Stack.
Method	Usage
Push(T)	Inserts an item at the top of the stack.
Peek()	Returns the top item from the stack.
Pop()	Removes and returns items from the top of the stack.
Contains(T)	Checks whether an item exists in the stack or not.
Clear()	Removes all items from the stack.

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day13Project9
{
    internal class Program
    {
        static void Main(string[] args)
        {
        }
```

```

Stack<int> data = new Stack<int>();

data.Push(23);
data.Push(28);
data.Push(32);

Console.WriteLine("Count:" + data.Count);
Console.WriteLine("Peek:" + data.Peek());
Console.WriteLine("Count:" + data.Count);
Console.WriteLine("Pop:" + data.Pop());
Console.WriteLine("Count:" + data.Count);

Console.ReadLine();
    }
}

```

Output:

D:\NB HealthCare Training\DotNet Projects\Day 13 Morning Assignment\Day13Project9\Day13Project9\bin\Debug\Day13Project9.exe

```

Count:3
Peek:32
Count:3
Pop:32
Count:2

```

12. WACP to illustrate usage of Queue<>

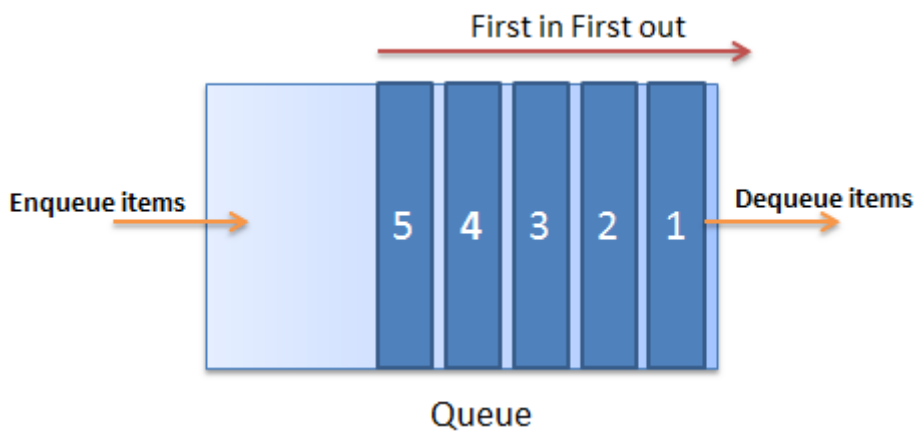
Write couple of points about Queue

Queue:

- Queue is a special type of collection that stores the elements in FIFO style (First In First Out), exactly opposite of the Stack<T> collection.
- It contains the elements in the order they were added.
- C# includes generic Queue<T> and non-generic Queue collection.
- It is recommended to use the generic Queue<T> collection.
- **Example:** Queue<int> data = new Queue<int>();

Queue<T> Characteristics:

- `Queue<T>` is FIFO (First In First Out) collection.
- It comes under `System.Collection.Generic` namespace.
- `Queue<T>` can contain elements of the specified type. It provides compile-time type checking and doesn't perform boxing-unboxing because it is generic.
- Elements can be added using the `Enqueue()` method. Cannot use collection-initializer syntax.
- Elements can be retrieved using the `Dequeue()` and the `Peek()` methods. It does not support an indexer.



Queue<T> Properties and Methods

Property	Usage
Count	Returns the total count of elements in the Queue.

Method	Usage
Enqueue(T)	Adds an item into the queue.
Dequeue	Returns an item from the beginning of the queue and removes it from the queue.
Peek(T)	Returns an first item from the queue without removing it.
Contains(T)	Checks whether an item is in the queue or not
Clear()	Removes all the items from the queue.

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day13Project10
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Queue<int> data = new Queue<int>();

            data.Enqueue(23);
            data.Enqueue(28);
            data.Enqueue(32);

            Console.WriteLine("Count:" + data.Count);
            Console.WriteLine("Peek:" + data.Peek());
            Console.WriteLine("Count:" + data.Count);
            Console.WriteLine("Pop:" + data.Dequeue());
            Console.WriteLine("Count:" + data.Count);

            Console.ReadLine();
        }
    }
}
```

Output:

D:\NB HealthCare Training\DotNet Projects\Day 13 Morning Assignment\Day13Project10\Day13Project10\bin\Debug\Day13Project10.exe

```
Count:3
Peek:23
Count:3
Pop:23
Count:2
_
```