|  |
| --- |
| Day13 Morning Assignment    By  Anusha Bellala  13-2-2022 |

|  |
| --- |
| 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] = 56;  data[0, 1] = 63;  data[1, 0] = 45;  data[1, 1] = 23;  for(int i = 0; i <2; i++)  {  for(int j = 0; j <2; j++)  {  Console.Write(data[i,j]+" ");    }  Console.Write("\n");  }  Console.ReadLine();  }  }  } |
| Output: |

|  |
| --- |
| 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[,] { { 23, 46 }, { 15, 76 }, { 47, 80 } };  Console.WriteLine("\n Enter 2-D Array Representation for declaring in same line \n");  for (int i = 0; i < 2; i++)  {  for (int j = 0; j < 2; j++)  {  Console.Write(data[i, j] + " ");  }  Console.Write("\n");  Console.ReadLine();  }  }  }  } |
| Output: |

|  |
| --- |
| 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[,] { { 67, 52, 13 }, { 23, 42, 78 }, { 32, 12, 89 } };  int sum = 0;  Console.WriteLine("\n Given Array is : \n");  for (int i = 0; i < 3; i++)  {  for (int j = 0; j < 3; j++)  {  Console.Write("\t" + data[i, j] + " ");  }  Console.Write("\n");  }  for (int i = 0; i < 3; i++)  {  for (int j = 0; j < 3; j++)  {  if (i == j)  {  sum = sum + data[i, j];  }  }  }  Console.WriteLine("\n The Trace of a given array is : {0}", sum);  Console.ReadLine();  }  }  } |
| Ouput: |

|  |
| --- |
| 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];  for (int i = 0; i < 2; i++)  {  for (int j = 0; j < 2; j++)  {  Console.Write($"\n Enter the array item at ({i},{j}) : ");  data[i, j] = Convert.ToInt32(Console.ReadLine());  }  }  Console.WriteLine("\n Given Array is : \n");  for (int i = 0; i < 2; i++)  {  for (int j = 0; j < 2; j++)  {  Console.Write("\t" + data[i, j] + " ");  }  Console.WriteLine("\n");  }  Console.WriteLine("\n");  Console.ReadLine();  }  }  } |
| Output: |

|  |
| --- |
| 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 Day13Project5  {  internal class Program  {  static void Main(string[] args)  {  Program obj = new Program();  obj.SumOfTwoArrays();  Console.ReadLine();  }  void SumOfTwoArrays()  {  Console.Write("Enter Number to Define Rows & Column:- ");  int arrayLength = Convert.ToInt32(Console.ReadLine());  int[,] array = new int[arrayLength, arrayLength];  int[,] arraySecond = new int[arrayLength, arrayLength];  int[,] arraySum = new int[arrayLength, arrayLength];  for (int i = 0; i < arrayLength; i++)  {  for (int j = 0; j < arrayLength; j++)  {  Console.Write("Array Index [{0}][{1}]:- ", i, j);  array[i, j] = Convert.ToInt32(Console.ReadLine());  }  }  Console.WriteLine("This is Your First Array:-");  for (int i = 0; i < arrayLength; i++)  {  for (int j = 0; j < arrayLength; j++)  {  if (j == 0)  {  Console.Write(array[i, j]);  }  else  {  Console.Write(" " + array[i, j]);  }  }  Console.WriteLine();  }  Console.WriteLine("Now Enter Your Second Array");  for (int i = 0; i < arrayLength; i++)  {  for (int j = 0; j < arrayLength; j++)  {  Console.Write("Array Index [{0}][{1}]:- ", i, j);  arraySecond[i, j] = Convert.ToInt32(Console.ReadLine());  }  }  Console.WriteLine("This is Your Second Array:-");  for (int i = 0; i < arrayLength; i++)  {  for (int j = 0; j < arrayLength; j++)  {  if (j == 0)  {  Console.Write(arraySecond[i, j]);  }  else  {  Console.Write(" " + arraySecond[i, j]);  }  }  Console.WriteLine();  }  Console.WriteLine("Do you want to add this arrays:- (Y/N)");  string userInput = Convert.ToString(Console.ReadLine());  if (userInput.ToUpper() == "Y")  {  for (int i = 0; i < arrayLength; i++)  {  for (int j = 0; j < arrayLength; j++)  {  arraySum[i, j] = array[i, j] + arraySecond[i, j];  }  }  Console.WriteLine("Array is Added Successfully Here is your Result");  for (int i = 0; i < arrayLength; i++)  {  for (int j = 0; j < arrayLength; j++)  {  if (j == 0)  {  Console.Write(arraySum[i, j]);  }  else  {  Console.Write(" " + arraySum[i, j]);  }  }  Console.WriteLine();  }  }  else  {  Console.WriteLine("Program Terminate Press Enter To Exit");  Console.ReadLine();  }  }  }  } |
| Ouput: |

|  |
| --- |
| 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 Day13Project6  {  internal class Program  {  static void Main(string[] args)  {  int i, j, k, r1, c1, r2, c2, sum = 0;  int[,] arr1 = new int[50, 50];  int[,] brr1 = new int[50, 50];  int[,] crr1 = new int[50, 50];  Console.Write("\n\n\t Multiplication of two Matrices");  Console.Write("\n Input the number of rows and columns of the first matrix:\n");  Console.Write("Rows : ");  r1 = Convert.ToInt32(Console.ReadLine());  Console.Write("Columns : ");  c1 = Convert.ToInt32(Console.ReadLine());  Console.Write("\nInput the number of rows of the second matrix :\n");  Console.Write("Rows : ");  r2 = Convert.ToInt32(Console.ReadLine());  Console.Write("Columns : ");  c2 = Convert.ToInt32(Console.ReadLine());  if (c1 != r2)  {  Console.Write("Multiplication of Matrix is not possible.");  Console.Write("\n Column of first matrix and row of second matrix must be same.");  }  else  {  Console.Write("Enter the Input elements in the first matrix :\n");  for (i = 0; i < r1; i++)  {  for (j = 0; j < c1; j++)  {  Console.Write($"element - [{i}],[{j}] : ");  arr1[i, j] = Convert.ToInt32(Console.ReadLine());  }  }  Console.Write("\nThe First matrix is :\n");  for (i = 0; i < r1; i++)  {  Console.Write("\n");  for (j = 0; j < c1; j++)  Console.Write("{0}\t", arr1[i, j]);  }  Console.Write("\n\n Enter the Input elements in the second matrix:\n\n");  for (i = 0; i < r2; i++)  {  for (j = 0; j < c2; j++)  {  Console.Write("element - [{0}],[{1}] : ", i, j);  brr1[i, j] = Convert.ToInt32(Console.ReadLine());  }  }  Console.Write("\nThe Second matrix is :\n");  for (i = 0; i < r2; i++)  {  Console.Write("\n");  for (j = 0; j < c2; j++)  Console.Write("{0}\t", brr1[i, j]);  }  Console.Write("\n");  for (i = 0; i < r1; i++)  for (j = 0; j < c2; j++)  crr1[i, j] = 0;  for (i = 0; i < r1; i++)  {  for (j = 0; j < c2; j++)  {  sum = 0;  for (k = 0; k < c1; k++)  sum = sum + arr1[i, k] \* brr1[k, j];  crr1[i, j] = sum;  }  }  Console.Write("\n The multiplication of two matrix is : \n");  for (i = 0; i < r1; i++)  {  Console.Write("\n");  for (j = 0; j < c2; j++)  {  Console.Write("{0}\t", crr1[i, j]);  }  }  }  Console.Write("\n\n");  Console.ReadLine();  }  }  } |
| Output: |

|  |
| --- |
| 7. What is a jagged array. What is the benefit of jagged array. |
| A jagged array is an array whose elements are arrays, possibly of different sizes. ... Each of the elements is a single-dimensional array of integers. The first element is an array of 5 integers, the second is an array of 4 integers, and the third is an array of 2 integers. |
| **Benefits of Jagged Array:**  There are several benefits of using jagged arrays and one of the most crucial advantages is, it makes things easy where there is a need to store data in a multidimensional way using the same variable name. It helps in memory management which makes the program to be executed very smoothly and fast as well. |

|  |
| --- |
| 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[4][];  names[0] = new char[] { 'A', 'N', 'U', 'S', 'H', 'A' };  names[1] = new char[] { 'M', 'A', 'D', 'H', 'U', 'R','I' };  names[2] = new char[] { 'P', 'R', 'E', 'M' };  names[3] = new char[] { 'S', 'I', 'N', 'D', 'U'};  // Printing The Jagged Array Values.  Console.WriteLine("\n Jagged Array Of Names \n");  for (int i = 0; i < 4; i++)  {  for (int j = 0; j < names[i].Length; j++)  {  Console.Write(names[i][j]);  }  Console.WriteLine("\n");  Console.ReadLine();  }  }  }  } |
| Ouput: |

|  |
| --- |
| 9. What is Recursion. |
| 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 |
| 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)  {  Console.WriteLine("Enter a number");  int number = Convert.ToInt32(Console.ReadLine());  long fact = GetFactorial(number);  Console.WriteLine("{0} factorial is {1}", number, fact);  Console.ReadLine();  }  private static long GetFactorial(int number)  {  if (number == 0)  {  return 1;  }  return number \* GetFactorial(number - 1);  }  }  } |
| Output: |

|  |
| --- |
| 11. WACP to illustrate usage of Stack<>  Write couple of points about Stack |
| Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day13Project8  {  internal class Program  {  static void Main(string[] args)  {  Stack<int> data = new Stack<int>();  data.Push(90);  data.Push(45);  data.Push(67);  data.Push(12);  Console.WriteLine($"The Stack size before removing the last item is : {data.Count}");  Console.WriteLine($"The Stack item to be removed is : {data.Peek()}");  Console.WriteLine($"In Stack ,the item removed is : {data.Count}");  Console.WriteLine($" The Stack size after removing the last item is : {data.Count}");  Console.WriteLine($"The Stack next item to be removed is: {data.Peek()}");  Console.ReadLine();  }  }  } |
| Output: |

|  |
| --- |
| **What is Stack?**  A stack (sometimes called a “push-down stack”) is an ordered collection of items where the addition of new items and the removal of existing items always takes place at the same end. ... This ordering principle is sometimes called LIFO, last-in first-out. |
| **Benefits of Stack**: |
| When a function is called the local variables are stored in a stack, and it is automatically destroyed once returned. A stack is used when a variable is not used outside that function. It allows you to control how memory is allocated and deallocated. Stack automatically cleans up the object. |
|  |

|  |
| --- |
| 12. WACP to illustrate usage of Queue<>  Write couple of points about Queue |
| 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)  {  Queue<int> data = new Queue<int>();  data.Enqueue(22);  data.Enqueue(43);  data.Enqueue(90);  data.Enqueue(67);  Console.WriteLine($"The Queue size before removing the last item is : {data.Count}");  Console.WriteLine($"The Queue item to be removed is : {data.Peek()}");  Console.WriteLine($"In Queue,the item removed is : {data.Dequeue()}");  Console.WriteLine($" The Queue size after removing the last item is : {data.Count}");  Console.WriteLine($"The Queue next item to be removed is: {data.Peek()}");  Console.ReadLine();  }  }  } |
| Output: |

|  |
| --- |
| **What is queue ?**  A Queue is a linear structure which follows a particular order in which the operations are performed. The order is First In First Out (FIFO). ... The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added. |
| **Benefits of Queue:** |
| The advantages of queues are that the multiple data can be handled, and they are fast and flexibility. &nbps; ... To include a new element in the queue, the other elements must be deleted. |