|  |
| --- |
| Day-13 Morning Assignment  By  U.Joshna  [9-2-2022] |

|  |
| --- |
| 1. Declare a 2 dimentional array of size (2,2) and initialize using indexes and print the values using nested for loop |
| Code: |
| using System;  namespace Day\_13\_Project\_1  {  internal class Program  {  static void Main(string[] args)  {  int[,] data = new int[2, 2];  data[0, 0] = 10;  data[0, 1] = 15;  data[1, 0] = 20;  data[1, 1] = 25;    Console.WriteLine("\n Ente 2-D Array Representation \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: |
|  |
|  |
|  |

|  |
| --- |
| 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: |
| namespace Day\_13\_Project\_2  {  internal class Program  {  static void Main(string[] args)  {  int[,] data = new int[,] { {3,4 }, {5,6}, {7,8} };    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;  namespace Day\_13\_Project\_3  {  internal class Program  {  static void Main(string[] args)  {  int[,] data = new int[,] { { 11,12,13 }, { 14, 15, 16 }, { 17, 18, 19 } };  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();  }  }  } |
| Output: |
|  |
|  |
| 4. Declare a 2-D array of size (2,2) and read values from user and print the array values. |
| Code: |
| using System;  namespace Day\_13\_Project\_4  {  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;  namespace Day\_13\_Project\_5  {  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();  }  }  }  } |
| Output: |
|  |
|  |
|  |

|  |
| --- |
| 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;  namespace Day\_13\_Project\_6  {  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\tMultiplication of two Matrices");  Console.Write("\nInput 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("Mutiplication of Matrix is not possible.");  Console.Write("\nColumn 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\nEnter 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("\nThe 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: |
| using System;  namespace Day\_13\_Project\_6  {  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\tMultiplication of two Matrices");  Console.Write("\nInput 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("Mutiplication of Matrix is not possible.");  Console.Write("\nColumn 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\nEnter 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("\nThe 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();  }  }  }  using System;  namespace Day\_13\_Project\_6  {  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\tMultiplication of two Matrices");  Console.Write("\nInput 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("Mutiplication of Matrix is not possible.");  Console.Write("\nColumn 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\nEnter 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("\nThe 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? |
| Jagged array: |
| .A jagged 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 |
| Benefit of Jagged array: |
| .There are several benefits of using jagged arrays. One of the most crucial adventages is that it makes things easy where there is a need to store data in multidimensional way using the same variable name . further, it more 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;  namespace Day\_13\_Project\_8  {  internal class Program  {  static void Main(string[] args)  {  char[][] names = new char[4][];  names[0] = new char[] { 'J', 'O', 'S', 'H', 'A' };  names[1] = new char[] { 'L', 'A', 'T', 'H', 'A' };  names[2] = new char[] { 'V', 'Y', 'N', 'I', 'T' ,'H','A'};  names[3] = new char[] { 'S', 'A', 'I' };  // 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();  }  }  }  } |
| Output: |
|  |
|  |
| 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 needed 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;  namespace Day\_13\_Project\_10  {  internal class Program  {  public static void PrintOutput(int n)  {  Console.WriteLine("\nFactorial of {0} is {1}", n, Factorial(n));  }  public static int Factorial(int input)  {  if (input == 0)  return 1;  else  return input \* Factorial(input - 1);  }  static void Main(string[] args)  {  int input;  Console.WriteLine("\n Welcome To Find Factorial Using Recursion : ");  Console.WriteLine("\n\n Enter any Number, To find It's Factorial : ");  input = Convert.ToInt32(Console.ReadLine());  Console.WriteLine("\n\n Developer of this Code is Joshna U : ");  Console.ReadLine();  }  }  } |
| Output: |
|  |
| Benefits of Recursion:  .Recursion can reduce time complexity  .Recursion reduces unnecessary calling functions  .Through Recursion one can solve problems in easy way while its iterative solution is very big and complex  .Extremely useful when applying the same solution  .Recursion adds clarity and reduces the time needed to write and debug code  . Recursion is better at tree traversal |

|  |
| --- |
| 11. WACP to illustrate usage of Stack<>  Write couple of points about Stack |
| Code: |
| using System;  namespace Day\_13\_Project\_11  {  internal class Program  {  static void Main(string[] args)  {  Stack<int> data = new Stack<int>();  data.push(20);  data.push(30);  data.push(40);  data.push(50);  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: |
| .Stack is a Special type of collection that stores elements in LIFO style (Last In First Out)  . Stack is useful to store temporary data in LIFO style, and you might want to delete an element after retrieving its value. |
| Benefits of Stack: |
| .Helps you to manage the data in a Last In First Out(LIFO) method which is not possible with Linked list and array  . It allows you to control how memory is allocated and deallocated.  . Not easily corrupted  .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. |

|  |
| --- |
| 12.WACP to illustrate usage of Queue<>  Write couple of points about Stack |
| Output: |
| using System;  using System.Collections.Generic;  namespace Day\_13\_Project\_12  {  internal class Program  {  static void Main(string[] args)  {  Queue<int> data = new Queue<int>();  data.Enqueue(20);  data.Enqueue(30);  data.Enqueue(40);  data.Enqueue(50);  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: |
| .Queue is a Special type of collection that stores elements in FIFO style (First In First Out)  . It contains the elements in the order they were added |
| Benefits of Queue: |
| .We use when you need to perform actions on a set of objects in a sequence  .It supports multiple readers simultaneously  . Multiple data can be handled, and they are fast and flexibility |