|  |
| --- |
| **Day 13 Morning Assignments**  **By**  **Manoj Yekolla**  **09-Feb-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;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day13\_Project1  {  internal class Program  {  static void Main(string[] args)  {  int[,] data = new int[2, 2];  data[0, 0] = 5;  data[0, 1] = 10;  data[1, 0] = 15;  data[1, 1] = 20;  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 :  Screenshot (213) |

|  |
| --- |
| **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 Day13\_Project2  {  internal class Program  {  static void Main(string[] args)  {  int[,] data = new int[,] { { 4, 5 }, { 6, 7 } } ;    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 :  Screenshot (215) |

|  |
| --- |
| **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 Day13\_Project3  {  internal class Program  {  static void Main(string[] args)  {  int[,] data = new int[,] { { 3, 2, 3, }, { 5, 3, 4, }, { 2, 4,6 } };  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("Enter the Sum Value in Trace is :{0}",sum);  Console.ReadLine();  }  }  } |
| Output :  Screenshot (217) |

|  |
| --- |
| **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 Day13\_Project4  {  internal class Program  {  static void Main(string[] args)  {  int[,] data = new int[2, 2];  //Read User  for (int i = 0; i < 2; i++)  {  for (int j = 0; j < 2; j++)  {  Console.WriteLine($"Enter the array value ({i},{j})");  data[i, j] =Convert.ToInt32( Console.ReadLine());  }  }  // Print User  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 :  Screenshot (219) |

|  |
| --- |
| **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 Day13\_Project5  {  internal class Program  {  static void Main(string[] args)  {  int i, j, n;  int[,] arr1 = new int[2, 2];  int[,] arr2 = new int[2, 2];  int[,] arr3 = new int[2, 2];  // setting matrix row and columns size  n = 3;  Console.Write("Enter elements in the first matrix:\n");  for (i = 0; i < 2; i++)  {  for (j = 0; j < 2; j++)  {  arr1[i, j] = Convert.ToInt32(Console.ReadLine());  }  }  Console.Write("Enter elements in the second matrix:\n");  for (i = 0; i < 2; i++)  {  for (j = 0; j < 2; j++)  {  arr2[i, j] = Convert.ToInt32(Console.ReadLine());  }  }  Console.Write("\nFirst matrix is:\n");  for (i = 0; i < 2; i++)  {  Console.Write("\n");  for (j = 0; j < 2; j++)  Console.Write("{0}\t", arr1[i, j]);  }  Console.Write("\nSecond matrix is:\n");  for (i = 0; i < 2; i++)  {  Console.Write("\n");  for (j = 0; j < 2; j++)  Console.Write("{0}\t", arr2[i, j]);  }  for (i = 0; i < 2; i++)  for (j = 0; j < 2; j++)  arr3[i, j] = arr1[i, j] + arr2[i, j];  Console.Write("\nAdding two matrices: \n");  for (i = 0; i < 2; i++)  {  Console.Write("\n");  for (j = 0; j < 2; j++)  Console.Write("{0}\t", arr3[i, j]);  }  Console.Write("\n\n");  Console.ReadLine();  }  }  } |
| Output :  Screenshot (221) |

|  |
| --- |
| **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 Day13\_Project6  {  internal class Program  {  static void Main(string[] args)  {  int i, j;  int[,] arr1 = new int[2, 2];  int[,] arr2 = new int[2, 2];  int[,] arr3 = new int[2, 2];  Console.Write("Enter elements in the first matrix:\n");  for (i = 0; i < 2; i++)  {  for (j = 0; j < 2; j++)  {  arr1[i, j] = Convert.ToInt32(Console.ReadLine());  }  }  Console.Write("Enter elements in the second matrix:\n");  for (i = 0; i < 2; i++)  {  for (j = 0; j < 2; j++)  {  arr2[i, j] = Convert.ToInt32(Console.ReadLine());  }  }  Console.Write("\nFirst matrix is:\n");  for (i = 0; i < 2; i++)  {  Console.Write("\n");  for (j = 0; j < 2; j++)  Console.Write("{0}\t", arr1[i, j]);  }  Console.Write("\nSecond matrix is:\n");  for (i = 0; i < 2; i++)  {  Console.Write("\n");  for (j = 0; j < 2; j++)  Console.Write("{0}\t", arr2[i, j]);  }  for (i = 0; i < 2; i++)  for (j = 0; j < 2; j++)  arr3[i, j] = arr1[i, j] \* arr2[i, j];  Console.Write("\nProduct two matrices: \n");  for (i = 0; i < 2; i++)  {  Console.Write("\n");  for (j = 0; j < 2; j++)  Console.Write("{0}\t", arr3[i, j]);  }  Console.Write("\n\n");  Console.ReadLine();  }  }  } |
| Output :  Screenshot (223) |

|  |
| --- |
| **7. What is a jagged array?**  **What is the benefit of jagged array ?** |
| **Jagged Array :**   * Jagged Array is a two diminesional Array ,it will have different sizes of different rows ,   The way to declare different , compare to normal array . we don’t have mention the last column.   * Syntax :     char[][] names = new char[3][];  **Benifit of jagged Array :**   * We will save some memory , if the array size is different. |

|  |
| --- |
| **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 Day13\_project7  {  internal class Program  {  static void Main(string[] args)  {  char[][] names = new char[3][];  names[0] = new char[] { 'm', 'a', 'n', 'u' };  names[1] = new char[] { 'n', 'i', 't', 'h', 'e', 's', 'h' };  names[2] = new char[] { 'r', 'a', 'v', 'i' };  for (int i=0;i<3;i++)  {  for (int j=0;j<names[i].Length;j++)  {  Console.Write(names[i][j]);  }  Console.Write("\n");  }  Console.ReadLine();  }  }  } |
| Output :  Screenshot (225) |

|  |
| --- |
| **9. What is Recursion** |
| * A Function calling itself repeatedly until a specified condition statisfied. |

|  |
| --- |
| **10 .WACP to illustrate usage of Recursion.** |
| Code : |
| using System;  namespace Recursion  {  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 :  Screenshot (227) |
| Benifits :   * Recursion is better at tree traversal. * Recursion can reduce time complexity. * Iteration :A function repeats adefined process until a condition fails. |

|  |
| --- |
| **11. WACP to illustrate usage of Stack<>** |
| Code : |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day13Project11  {  internal class Program  {  static void Main(string[] args)  {  Stack<int> data = new Stack<int>();  data.Push(55);  data.Push(28);  data.Push(68);  Console.WriteLine(data.Count);  Console.WriteLine(data.Pop()); //it will remove the element  //Console.ReadLine(data.Peek()); //it will not remove the element  Console.WriteLine(data.Count);  Console.ReadLine();  }  }  } |
| Output :  Screenshot (229) |
| **Write couple of points about Stack :** |
| * Stack represents a last in first out collection of object. * It is used when you need a last in ,first out access to items. * When you add the element in the list ,it is called pushing. * When you remove the element in the list , it is called popping. |

|  |
| --- |
| **12.WACP to illustrate usage of Queue<>** |
| Code : |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day13\_Project11  {  internal class Program  {  static void Main(string[] args)  {  Queue<int> data = new Queue<int>();  data.Enqueue(55);  data.Enqueue(28);  data.Enqueue(68);  Console.WriteLine(data.Count);  Console.WriteLine(data.Dequeue());  Console.WriteLine(data.Count);  Console.ReadLine();  }  }  } |
| Output :  Screenshot (231) |
| **Write couple of points about Stack :** |
| * Queue represents a first in first out collection of object. * It is used when you need a first in ,first out access to items. * When you add the element in the list ,it is called enqueue. * When you remove the element in the list , it is called dequeue. |