|  |
| --- |
| Day 20 Assignment  By Ramakrishna |

|  |
| --- |
| 1. Research and understand scope of variables in C# |
| In C# variable can be divided into 3 types  Class Level Scope  Method Level Scope  Block Level Scope |
| Code: Block Level |
| * These variables are generally declared inside the for loop. * Generally, a loop inside a method has three level of nested code blocks |
| Code: Method Level |
| * Variables which hare declared inside method have method level scope. these are not accessible outside the method. * These variables are called as Local variables. * There will be a compile – time error if these variables are declared twice with the same name in the same scope. |
| Code: Class Level |
| * Access modifier of class level variables doesn’t affect their scope within a class. * Class level scoped variable can be accessed by non-static methods of the class in which it is declared. * Member variables can be accessed outside the class by using the access modifiers. |

|  |
| --- |
| 2. What are delegates in C#  Write the points dicussed about delegates in the class |
| A delegate is a type that represents references to methods with a particular parameter list and return type. When you instantiate a delegate, you can associate its instance with any method with a compatible signature and return type. ... In other words, a method must have the same return type as the delegate. |
| * A delegate is like a function pointer. * Using delegate we can cll or point to one or more methods. * When declaring a delagte,return type and paramter must watch with the methds you want to point using delegate. * Benfits of deletage is that, * Using singlr call from delegate,all your methods pointing to delegaate will be called. * Types of delegate are  1. Single Cast Delegate. 2. Multi cast delegate. |

|  |
| --- |
| Write C# code to illustrate the usage of delegates. |
| Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day\_20\_\_Project\_2  {  public delegate void mathematics(int a, int b);  internal class Program  {  public static void Add(int a ,int b)  {  Console.WriteLine(a+b);  }  public static void sub(int a, int b)  {  Console.WriteLine(a-b);  }  public static void mul(int a, int b)  {  Console.WriteLine(a\*b);  }  public static void Div(int a, int b)  {  Console.WriteLine(a/b);  }  static void Main(string[] args)  {  mathematics m = new mathematics(Add);  m += sub;  m += mul;  m += Div;  m(4, 5);  m -=Add;  m(10, 20);  m-=sub;  m-=mul;  m(5, 10);  Console.ReadLine();  }  }  } |
| Output: |
|  |

|  |
| --- |
| 3. What are nullable types in C#  WACP to illustrate nullable types |
| Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day\_\_20\_Project\_\_3  {  internal class Program  {  static void Main(string[] args)  {  byte? input = null;  if (input.HasValue)  {  Console.WriteLine(input \* input);  }  else  {  Console.WriteLine("No Value");  }  Console.ReadLine();  }  }  } |
| Output: |
|  |

|  |
| --- |
| 4 . out, ref - parameters  please research on these two types of parameters  write a C# program to illustrate the same. |
| The **out**is a keyword in C# which is used for the passing the arguments to methods as a reference type. It is generally used when a method returns multiple values. The out parameter does not pass the property. |
| Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day\_\_20\_Project\_4  {  internal class Program  {  public static void Sum(out int i)  {  i = 90;  i += 90;  }  static void Main(string[] args)  {  int i;  Sum(out i);  Console.WriteLine($"Sum of Value is : {i}");  Console.ReadLine();  }  }  } |
| Output: |
|  |

|  |
| --- |
| ref – parameters:  The **ref**is a keyword in C# which is used for the passing the arguments by a reference. Or we can say that if any changes made in this argument in the method will reflect in that variable when the control return to the calling method. The ref parameter does not pass the [property](https://www.geeksforgeeks.org/c-properties/). |
| Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day\_20\_\_project\_5  {  internal class Program  {  private static void SetValue(ref string s)  {  if (s == "Hai")  {  Console.WriteLine("Welcome to my C# world");  }  s = "Rk";  }  static void Main(string[] args)  {  string s1 = "Hai";  SetValue(ref s1);  Console.WriteLine(s1);  Console.ReadLine();  }  }  } |
| Output: |
|  |