**.NET ASSESSMENT THEORY**

1. **Call by Value and Call by Reference:**
2. Call by Value:

In call by value original value is not modified. Only variable value is passed to the function and not reference.

Ex:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Func\_Example

{

class Example

{

public void Swap(int a, int b)

{

int temp = a;

a = b;

b = temp;

Console.WriteLine("After swapping the values are:{0}\t{1}", a, b);

}

}

}

2.Call by Reference:

Call by reference modifies the original value as the reference of the variable is passed.

Class Example

public void Swap1(ref int x,ref int y)

{

int temp = x;

x= y;

y= temp;

}

internal class Class1

static void Main(string[]args)

{

int a = 10, b = 20;

Console.WriteLine("Before swapping the values are{0}\t{1},a,b");

Sample sample = new Sample();

sample.Swap(a, b);

sample.swap1(ref a, ref b);

Console.WriteLine("After Swapping the Values are :{0}\t{1},a,b");

}

}

}

**2.Program to print output of adding,subtracting,multiplying and dividing of two numbers which will be entered by the user.**

|  |  |
| --- | --- |
|  |  |
|  | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Example |
|  | { |

public class Exam

{

public static void Main()

{

Console.Write("Enter a number: ");

int num1 = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter another number: ");

int num2 = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("{0} + {1} = {2}", num1, num2, num1 + num2);

Console.WriteLine("{0} - {1} = {2}", num1, num2, num1 - num2);

Console.WriteLine("{0} x {1} = {2}", num1, num2, num1 \* num2);

Console.WriteLine("{0} / {1} = {2}", num1, num2, num1 / num2);

Console.WriteLine("{0} mod {1} = {2}", num1, num2, num1 % num2);

}

}

}

**3.Method to find the nth highest or lowest number in a series of numbers.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assessment

{

internal class Class3

namespace Exam

{

{

class

{

static void Main()

{

List<long> list = new List<long> { 100, 200, 250, 300, 400, 600 };

foreach (long ele in list)

{

Console.WriteLine(ele);

}

long max\_num = list.AsQueryable().Max();

long min\_num = list.AsQueryable().Min();

Console.WriteLine("Smallest number = {0}", min\_num);

Console.WriteLine("Largest number = {0}", max\_num);

}

}

}

**4.Program to convert first letter of each word into upper case and others into lower case froma given string.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assessment

{

internal class Class3

{

public static void Main()

{

string Str;

int i, length, lower, upper;

Str = "Shreya";

Console.Write("String: " + Str);

upper = 0;

lower = 0;

length = Str.Length;

for (i = 0; i < length; i++)

{

if (Str[i] >= 'a' && Str[i] <= 'b')

{

lower++;

}

else if (Str[i] >= 'A' && Str[i] <= 'B')

{

upper++;

}

}

Console.Write("\nCharacters in lowecase: {0}\n", lower);

Console.Write("Characters in uppercase: {0}\n\n", upper);

}

}

}

**5.Program to print how many caps letters,small letters and words are present in a given string.**

using System;

public class Exam

{

public static void Main()

{

string myStr;

int i, len, lower\_case, upper\_case;

myStr = "Hello";

Console.Write("String: " + myStr);

lower\_case = 0;

upper\_case = 0;

len = myStr.Length;

for (i = 0; i < len; i++)

{

if (myStr[i] >= 'a' & amp; &amp; myStr[i] <= 'z') {

lower\_case++;

} else if (myStr[i] >= 'A' & amp; &amp; myStr[i] <= 'Z') {

upper\_case++;

}

}

Console.Write("\nCharacters in lowecase: {0}\n", lower\_case);

Console.Write("Characters in uppercase: {0}\n\n", upper\_case);

}

}

}

**6.Explain the differences between Abstract class and interfaces.**

An abstract class is a way to achieve the abstraction in C#. An Abstract class is never intended to be instantiated directly. This class must contain at least one abstract method, which is marked by the keyword or modifier abstract in the class definition. The Abstract classes are typically used to define a base class in the class hierarchy.

**7.Explain about method overloading and overriding with example.**

Overloading:

Overloading is having multiple methods within the same class with the same name, but with different parameters.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assessment\_7

{

internal class Class4

{

static void Main()

{

public class Calculator

{

public int Add(int a, int b) => a + b;

public int Add(int a, int b, int c) => a + b + c;

public double Add(double a, double b) => a + b;

public double Add(double a, double b, double c) => a + b + c;

}

}

Overriding:

Overriding, on the other hand, is the ability to redefine the implementation of a method in a class that inherits from a parent class.

public class Name

{

public virtual void Shreya()

{

Console.WriteLine("Shreya");

}

}

**8.Write a method with 2 parameters(sorting type,sort data) and return the sorted data.**

var nums = new List<int> { 2, 1, 8, 0, 4, 3, 5, 7, 9 };

nums.Sort();

Console.WriteLine(string.Join(",", nums));

nums.Reverse();

Console.WriteLine(string.Join(",", nums));

**9.Program to remove duplicate characters from a given string.**

using System;

using System.Linq;

using System.Collections.Generic;

namespace Assessment 9

{

class Program

{

static void Main(string[] args)

{

string myStr = "12344562427";

Console.WriteLine("Initial String: " + myStr);

var unique = new HashSet<char>(myStr);

Console.Write("New String after removing duplicates: ");

foreach (char c in unique)

Console.Write(c);

}

}

}

The initial string is 12344562427 and the string after removing duplicates will be 1234567.

**10.C# method which takes string parameter and returns a reversed string/number given in a parameter.**

private static void ReverseStringWithoutInbuiltMethod(string stringInput)

{

// Reverse using While loop

string reversestring = "";

int length;

length = stringInput.Length - 1;

while (length >= 0)

{

reversestring = reversestring + stringInput[length];

length--;

}

Console.WriteLine(reversestring);

Console.ReadLine();

}