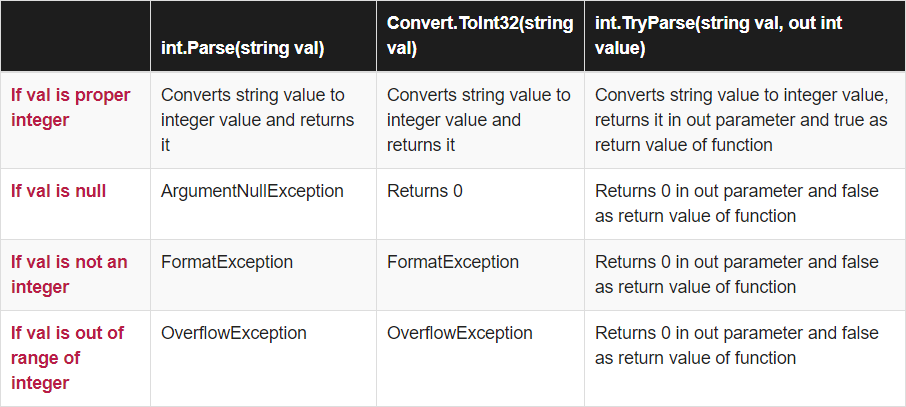
Q. Difference between int.Parse(), Convert.toInt(32) and int.TryParse



Q. Min-Max Program

using System;  
  
class Program  
{  
  
class MaxMin  
{  
public int min;  
public int max;  
}  
  
static MaxMin getMinMax(int []arr, int n)  
{  
MaxMin obj = new MaxMin();  
int i;  
  
if (arr[0] > arr[1])  
{  
obj.max = arr[0];  
obj.min = arr[1];  
}  
else  
{  
obj.max = arr[1];  
obj.min = arr[0];  
}  
  
for (i = 2; i < n; i++)  
{  
if (arr[i] > obj.max)  
{  
obj.max = arr[i];  
}  
else if (arr[i] < obj.min)  
{  
obj.min = arr[i];  
}  
}  
return obj;  
}  
  
  
public static void Main(String []args)  
{  
int []arr = {700, 987, 547, 171, 336, 989};  
// int arr\_size = 6;  
MaxMin obj1 = getMinMax(arr, arr.Length);  
Console.Write("Minimum value is {0}",  
obj1.min);  
Console.Write("\nMaximum value is {0}",  
obj1.max);  
}  
}

Q. Difference between Const and ReadOnly. Ans:

|  |  |
| --- | --- |
| Const Keyword | ReadOnly Keyword |
| In C#, constant fields are created using const keyword. | In C#, readonly fields can be created using readonly keyword. |
| Const is a compile time constant. | ReadOnly is a runtime constant. |
| The value of the const field can not be changed. | The value of ReadOnly field can be changed. |
| It can be declared inside the method. | It cannot be declared inside a method. |
| In const fields, we can only assign values in declaration part. | In readonly fields, we can assign values in declaration and in the constructor part. |
| It cannot be used with static modifiers. | It can be used to used with static modifiers. |

Q. Use of following interfaces

1. IEnumerable

IEnumerable is an interface defining a single method GetEnumerator() that returns an IEnumerator interface. It is the base interface for all non-generic collections that can be enumerated.

This works for read-only access to a collection that implements that IEnumerable can be used with a foreach statement.

It has a single method −

* + **GetEnumerator()** − This method returns an enumerator that iterates through a collection.

The following is the implementation of the GetEnumerator() method of the IEnumerable interface in C# −

IEnumerator IEnumerable.GetEnumerator() { return (IEnumerator) GetEnumerator();

}

The following are the extension methods of the IEnumerable interface in C# −

|  |  |
| --- | --- |
| **Sr.No** | **Method Name & Description** |
| 1 | **AsParallel()**  Enables parallelization of a query |
| 2 | **AsQueryable()**  The method converts an IEnumerable to an IQueryable. |
| 3 | **Cast<TResult>()**  The method casts the elements of an IEnumerable to the specified type |
| 4 | **OfType<TResult>()**  Filters the elements of an IEnumerable based on a specified type. |

1. ICollection

The ICollection interface in C# defines the size, enumerators, and synchronization methods for all nongeneric collections. It is the base interface for classes in the System.Collections namespace.

The following are the properties of ICollection interface –

|  |  |
| --- | --- |
| **Sr.No.** | **Property Name & Description** |
| 1 | **Count**  The number of elements in the ICollection |
| 2 | **SyncRoot**  Gets an object that useful to synchronize access to the ICollection. |

The following are the methods of ICollection interface −

|  |  |
| --- | --- |
| **Sr.No.** | **Method Name & Description** |
| 1 | **CopyTo(Array^,Int32)**  The method copies the elements of the ICollection to an Array. |
| 2 | **GetEnumerator()**  The GetEnumerator() method returns an enumerator that iterates through a collection |

1. IList

The IList interface has a non-generic collection of objects that can be individually accessed by index.

The following are the properties of interface IList in C# −

|  |  |
| --- | --- |
| **Sr.No** | **Property Name & Description** |
| 1 | **Count**  Gets the number of elements contained in the ICollection. |
| 2 | **isFixedSize**  Gets a value indicating whether the IList has a fixed size. |
| 3 | **isReadOnly**  Gets a value indicating whether the IList is read-only. |
| 4 | **isSynchronized**  Gets a value indicating whether access to the ICollection is synchronized. |
| 5 | **Item(Int32)**  Gets or sets the element at the specified index. |

The following are the methods of the IList interface −

|  |  |
| --- | --- |
| **Sr.No** | **Property Name & Description** |
| 1 | **Add(Obj)**  Adds an item to the IList. |
| 2 | **Clear()**  Removes all items from the IList |
| 3 | **Contains(Obj)**  Whether the list contains a specific value |
| 4 | **GetEnumerator()**  Returns an Enumerator |
| 5 | **IndexOf(Obj)**  Index of the specified item in the IList |
| 6 | **Remove(Obj)**  The method removes the first occurrence of a specific object. |