SortedSet in C# with Examples

geeksforgeeks.org/sortedset-in-c-sharp-with-examples

March 15, 2019

In C#, SortedSet is a collection of objects in sorted order. It is of the generic type collection and defined under **System.Collections.Generic** namespace. It also provides many mathematical set operations, such as intersection, union, and difference. It is a dynamic collection means the size of the SortedSet is automatically increased when the new elements are added.

Important Points:

- The SortedSet class implements the *ICollection*, *IEnumerable*, *IReadOnlyCollection*, *ISet*, *ICollection*, *IEnumerable*, *IDeserializationCallback*, and *ISerializable* interfaces.
- The capacity of a SortedSet is the number of elements it can hold.
- In SortedSet, the elements must be unique.
- In SortedSet, the order of the element is ascending.
- It is generally used when we want to use SortedSet class if you have to store unique elements and maintain ascending order.
- In SortedSet, the user can only store the same type of elements.

How to create a SortedSet?

The SortedSet class provides 5 different types of constructors which are used to create a SortedSet, here we only use *SortedSet()*, constructor. To read more about SortedSet's constructors you can refer to **C# | SortedSet Class**.

SortedSet(): It is used to create an instance of the SortedSet class.

Step 1: Include *System.Collections.Generic* namespace in your program with the help of *using* keyword:

```
using System.Collections.Generic;
```

Step 2: Create a SortedSet using the SortedSet class as shown below:

```
SortedSet<type_of_sortedset> sortedset_name = new SortedSet<type_of_sortedset>();
```

Step 3: If you want to add elements in your SortedSet, then use *Add()* method to add elements in the SortedSet. And you can also store elements in your SortedSet using collection initializer.

Step 4: The elements of SortedSet is accessed by using a foreach loop. As shown in the below example.

Example:

filter_none
edit

play_arrow

brightness_4

```
using System;
using System.Collections.Generic;
class GFG {
static public void Main()
{
SortedSet< int > my_Set1 = new SortedSet< int >();
mv Set1.Add(101);
mv Set1.Add(1001);
mv Set1.Add(10001);
mv Set1.Add(100001);
Console.WriteLine( "Elements of my_Set1:" );
foreach ( var val in my_Set1)
{
Console.WriteLine(val);
}
SortedSet< int > my Set2 = new SortedSet< int >() {
202.2002.20002.200002};
Console.WriteLine( "Elements of my_Set2:" );
foreach ( var valu in my_Set2)
{
Console.WriteLine(valu);
}
}
}
```

Output:

```
Elements of my_Set1:
101
1001
10001
100001
Elements of my_Set2:
202
2002
20002
200002
```

How to remove elements from the SortedSet?

In SortedSet, you are allowed to remove elements from the SortedSet. SortedSet<T> class provides three different methods to remove elements and the methods are:

- **Remove(T):** This method is used to remove a specified item from the SortedSet.
- **RemoveWhere(Predicate):** This method is used to remove all elements that match the conditions defined by the specified predicate from a SortedSet.
- **Clear():** This method is used to remove all elements from the set.

Example:

filter_none
edit

play_arrow

brightness_4

```
using System;
using System.Collections.Generic;
class GFG {
static public void Main()
{
SortedSet< int > my_Set = new SortedSet< int >();
my_Set.Add(101);
my_Set.Add(1001);
my_Set.Add(10001);
my_Set.Add(100001);
Console.WriteLine( "Total number of elements " +
"present in my_Set:{0}" , my_Set.Count);
my_Set.Remove(1001);
Console.WriteLine( "Total number of elements " +
"present in my_Set:{0}" , my_Set.Count);
my_Set.Clear();
Console.WriteLine( "Total number of elements " +
"present in my_Set:{0}" , my_Set.Count);
}
}
```

Output:

```
Total number of elements present in my_Set:4
Total number of elements present in my_Set:3
Total number of elements present in my_Set:0
```

How to check the availability of elements in SortedSet?

In SortedSet, you can check whether the given element is present or not using the Contains method. This method is used to determine whether the set contains a specific element.

Example:

filter_none
edit

play_arrow

brightness_4

```
using System;
using System.Collections.Generic;
public class GFG {
static public void Main()
{
SortedSet< int > my_Set = new SortedSet< int >();
mv_Set.Add(101);
mv_Set.Add(1001);
mv_Set.Add(10001);
my_Set.Add(100001);
if (my_Set.Contains(101) == true )
{
Console.WriteLine( "Element is available..!" );
}
else
{
Console.WriteLine( "Element is not available..!" );
}
}
}
```

Output:

Element is available..!

---

ankita_saini
Check out this Author's contributed articles.

If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please Improve this article if you find anything incorrect by clicking on the "Improve Article" button below.