

It represents an ordered collection of an object that can be indexed individually. It is basically an alternative to an array. However, unlike array, you can add and remove items from a list at a specified position using an **index** and the array resizes itself automatically. It also allows dynamic memory allocation, adding, searching and sorting items in the list.

Properties and Methods of the ArrayList Class

The following table lists some of the commonly used **properties** of the **ArrayList** class:

Property	Description
Capacity	Gets or sets the number of elements that the ArrayList can contain.
Count	Gets the number of elements actually contained in the ArrayList.
IsFixedSize	Gets a value indicating whether the ArrayList has a fixed size.
IsReadOnly	Gets a value indicating whether the ArrayList is read-only.
Item	Gets or sets the element at the specified index.

The following table lists some of the commonly used **methods** of the **ArrayList** class:

S.N.	Method Name & Purpose
1	Public Overridable Function Add <i>valueAsObject</i> As Integer Adds an object to the end of the ArrayList.
2	Public Overridable Sub AddRange <i>cAsICollection</i> Adds the elements of an ICollection to the end of the ArrayList.
3	Public Overridable Sub Clear Removes all elements from the ArrayList.
4	Public Overridable Function Contains <i>itemAsObject</i> As Boolean Determines whether an element is in the ArrayList.
5	Public Overridable Function GetRange <i>indexAsInteger, countAsInteger</i> As ArrayList Returns an ArrayList, which represents a subset of the elements in the source ArrayList.
6	Public Overridable Function IndexOf <i>valueAsObject</i> As Integer

Returns the zero-based index of the first occurrence of a value in the ArrayList or in a portion of it.

- 7 **Public Overridable Sub Insert** *indexAsInteger, valueAsObject*
Inserts an element into the ArrayList at the specified index.
- 8 **Public Overridable Sub InsertRange** *indexAsInteger, cAsICollection*
Inserts the elements of a collection into the ArrayList at the specified index.
- 9 **Public Overridable Sub Remove** *objAsObject*
Removes the first occurrence of a specific object from the ArrayList.
- 10 **Public Overridable Sub RemoveAt** *indexAsInteger*
Removes the element at the specified index of the ArrayList.
- 11 **Public Overridable Sub RemoveRange** *indexAsInteger, countAsInteger*
Removes a range of elements from the ArrayList.
- 12 **Public Overridable Sub Reverse**
Reverses the order of the elements in the ArrayList.
- 13 **Public Overridable Sub SetRange** *indexAsInteger, cAsICollection*
Copies the elements of a collection over a range of elements in the ArrayList.
- 14 **Public Overridable Sub Sort**
Sorts the elements in the ArrayList.
- 15 **Public Overridable Sub TrimToSize**
Sets the capacity to the actual number of elements in the ArrayList.

Example:

The following example demonstrates the concept:

```
Sub Main()  
    Dim al As ArrayList = New ArrayList()  
    Dim i As Integer  
    Console.WriteLine("Adding some numbers:")  
    al.Add(45)  
    al.Add(78)  
    al.Add(33)
```

```

al.Add(56)
al.Add(12)
al.Add(23)
al.Add(9)
Console.WriteLine("Capacity: {0} ", al.Capacity)
Console.WriteLine("Count: {0}", al.Count)
Console.Write("Content: ")
For Each i In al
    Console.Write("{0} ", i)
Next i
Console.WriteLine()
Console.Write("Sorted Content: ")
al.Sort()
For Each i In al
    Console.Write("{0} ", i)
Next i
Console.WriteLine()
Console.ReadKey()
End Sub
End Module

```

When the above code is compiled and executed, it produces the following result:

```

Adding some numbers:
Capacity: 8
Count: 7
Content: 45 78 33 56 12 23 9
Content: 9 12 23 33 45 56 78

```

Loading [MathJax]/jax/output/HTML-CSS/jax.js