# C# Non-Generic Collections with Examples

Non-generic collections in C# are found in the `System.Collections` namespace and store elements as `object` type. These collections do not enforce type safety and require explicit type casting when retrieving values.

## Types of Non-Generic Collections

1. ArrayList – Dynamic array that can hold different data types.

2. Hashtable – Stores key-value pairs, where keys and values can be of any type.

3. Stack – Last-In-First-Out (LIFO) collection.

4. Queue – First-In-First-Out (FIFO) collection.

5. SortedList – Stores key-value pairs in sorted order.

## Example 1: ArrayList

using System;  
using System.Collections;  
  
class Program  
{  
 static void Main()  
 {  
 ArrayList list = new ArrayList();  
 list.Add(10);  
 list.Add("Hello");  
 list.Add(3.14);  
 list.Add(true);  
 list.Insert(1, "Inserted");  
  
 Console.WriteLine("ArrayList Elements:");  
 foreach (var item in list) { Console.WriteLine(item); }  
  
 list.Remove(3.14);  
 list.RemoveAt(0);  
  
 Console.WriteLine("\nAfter Removal:");  
 foreach (var item in list) { Console.WriteLine(item); }  
 }  
}

## Example 2: Hashtable

using System;  
using System.Collections;  
  
class Program  
{  
 static void Main()  
 {  
 Hashtable ht = new Hashtable();  
 ht.Add(1, "One");  
 ht.Add(2, "Two");  
 ht.Add(3, "Three");  
  
 Console.WriteLine("Hashtable Elements:");  
 foreach (DictionaryEntry entry in ht) { Console.WriteLine($"{entry.Key}: {entry.Value}"); }  
  
 Console.WriteLine("\nValue for key 2: " + ht[2]);  
  
 ht.Remove(1);  
 Console.WriteLine("\nAfter Removing key 1:");  
 foreach (DictionaryEntry entry in ht) { Console.WriteLine($"{entry.Key}: {entry.Value}"); }  
 }  
}

## Example 3: Stack (LIFO)

using System;  
using System.Collections;  
  
class Program  
{  
 static void Main()  
 {  
 Stack stack = new Stack();  
 stack.Push("First");  
 stack.Push("Second");  
 stack.Push("Third");  
  
 Console.WriteLine("Popped: " + stack.Pop());  
 Console.WriteLine("Top Element: " + stack.Peek());  
  
 Console.WriteLine("\nStack Elements:");  
 foreach (var item in stack) { Console.WriteLine(item); }  
 }  
}

## Example 4: Queue (FIFO)

using System;  
using System.Collections;  
  
class Program  
{  
 static void Main()  
 {  
 Queue queue = new Queue();  
 queue.Enqueue("First");  
 queue.Enqueue("Second");  
 queue.Enqueue("Third");  
  
 Console.WriteLine("Dequeued: " + queue.Dequeue());  
 Console.WriteLine("Front Element: " + queue.Peek());  
  
 Console.WriteLine("\nQueue Elements:");  
 foreach (var item in queue) { Console.WriteLine(item); }  
 }  
}

## Example 5: SortedList

using System;  
using System.Collections;  
  
class Program  
{  
 static void Main()  
 {  
 SortedList sortedList = new SortedList();  
 sortedList.Add(3, "Three");  
 sortedList.Add(1, "One");  
 sortedList.Add(2, "Two");  
  
 Console.WriteLine("SortedList Elements:");  
 foreach (DictionaryEntry entry in sortedList) { Console.WriteLine($"{entry.Key}: {entry.Value}"); }  
  
 sortedList.Remove(2);  
 Console.WriteLine("\nAfter Removing key 2:");  
 foreach (DictionaryEntry entry in sortedList) { Console.WriteLine($"{entry.Key}: {entry.Value}"); }  
 }  
}

## Summary Table of Non-Generic Collections

|  |  |
| --- | --- |
| Collection | Description |
| ArrayList | Dynamic array that allows different data types. |
| Hashtable | Key-value pairs, unordered. |
| Stack | Last-In-First-Out (LIFO) collection. |
| Queue | First-In-First-Out (FIFO) collection. |
| SortedList | Key-value pairs sorted by key. |

## Key Takeaways

- Non-generic collections are flexible but lack type safety (you must cast objects).

- Prefer generic collections (List<T>, Dictionary<TKey,TValue>, etc.) for better performance and type safety.

- Use ArrayList for a resizable array, Hashtable for key-value pairs, Stack/Queue for ordered data processing.