Specialized Collections:

and then becomes a hashtable.

using collections in C#.

array.

Queue - FIFO (First in First out)

What are Array and Their disadvantages in C#?

array and then destroy the old array.

What is a Collection in C#?

handled by our program.

required.

index.

Generic Collections Classes in C#:

namespace has the following classes:

and manipulate lists.

Types of Collections in C#

System.Collections classes

2. System.Collections.Generic classes

3. System.Collections.Concurrent classes

The **System.Collections** namespace contains the following classes:

3. Queue: It represents a first-in, first-out collection of objects.

3. Queue<T>: It represents a first-in, first-out collection of objects.

System.Collections.Concurrent.IProducerConsumerCollection.

by multiple threads concurrently.

DOT NET

TUTORIALS

technologies.

Core

Leave a Reply

Comment *

Name*

Post Comment

f in X • (a)

New Batch Starts: 15th September, 2025

Session Time: 8:30 PM – 10:00 PM IST

View Course Details & Get Demo Credentials

Your email address will not be published. Required fields are marked *

Email*

Website

ConcurrentBag<T>: It represents a thread-safe, unordered collection of objects.

ConcurrentStack<T>: It represents a thread-safe last-in-first-out (LIFO) collection.

ConcurrentQueue<T>: It represents a thread-safe first-in-first-out (FIFO) collection.

your feedback. Please post your feedback, question, or comments about this article.

About the Author: Pranaya Rout

Dot Net Tutorials

2. Stack: It represents a simple last-in-first-out (LIFO) non-generic collection of objects.

2. We can never insert an element into the middle of an array

3. Deleting or removing elements from the middle of the array.

To overcome the above problems, the Collections are introduced in C# 1.0.

There are 3 ways to work with collections. The three namespaces are given below:

Following are the Limitations of Array in C#: 1. The array size is fixed. Once the array is created we can never increase the size of the array. If we want then we can do it manually by creating a new array and copying the old array elements into the new array or by using the Array class Resize method which will do the same thing means to create a new array and copy the old array elements into the new

The Specialized Collections are specifically designed for a specific purpose. For example, a Hybrid Dictionary starts as a list

Now, let us understand what are the problems with the Traditional Array in C#, and how we can overcome such problems

In simple words, we can say that the Arrays in C# are the simple data structure that is used to store similar types of data items

For example, you need to specify the array's size while creating the array. If at execution time, you want to modify it that

means if you want to increase or decrease the size of an array, then you need to do it manually by creating a new array or by

using the Array class's Resize method, which internally creates a new array and copies the existing array element into the new

in sequential order. Although the arrays in C# are commonly used, they have some limitations.

Stack - LIFO (Last in First Out)

storing multiple values but with the following features. 1. Size can be increased dynamically. 2. We can insert an element into the middle of a collection. 3. It also provides the facility to remove or delete elements from the middle of a collection.

The collections in C# are classes that represent a group of objects. With the help of C# Collections, we can perform different

types of operations on objects such as Store, Update, Delete, Retrieve, Search, and Sort objects, etc. In short, all the data

structure work can be performed by collections in C#. That means Collections standardize the way in which the objects are

The Collections in C# are a set of predefined classes that are present in the System.Collections namespace that provides

greater capabilities and functionalities than the traditional arrays. The collections in C# are reusable, more powerful, and more

So in simple words, we can say a Collection in C# is a dynamic array. That means the collections in C# have the capability of

efficient and most importantly they have been designed and tested to ensure quality and performance.

System.Collections.Concurrent System.Collections.Generic System.Collection Non-Generic Collections Classes in C#:

The Non-Generic Collection Classes come with C# 1.0 and are defined under the System.Collections namespace. The Non-

Generic collection classes in C# operate on objects, and hence can handle any type of data, but not in a safe-type manner.

1. ArrayList: It Implements the System. Collections. IList interface using an array whose size is dynamically increased as

5. SortedList: It represents a collection of key/value pairs that are sorted by the keys and are accessible by key and by

The Generic Collection Classes come with C# 2.0 and are defined under the System.Collection.Generic namespace. It

provides a generic implementation of standard data structures like linked lists, stacks, queues, and dictionaries. These

collection classes are type-safe because they are generic means only those items that are type-compatible with the type of the

collection can be stored in a generic collection, it eliminates accidental type mismatches. The System.Collections.Generic

1. List<T>: It represents a strongly typed list of objects that can be accessed by index. Provides methods to search, sort,

2. Stack<T>: It represents a variable size last-in-first-out (LIFO) collection of instances of the same specified type.

4. HashSet<T>: It represents a set of values. It removes duplicate elements from the collection.

4. Hashtable: It represents a collection of key/value pairs that are organized based on the hash code of the key.

Collections

Dictionary<TKey, TValue>: It represents a collection of keys and values. 6. SortedList<TKey, TValue>: It represents a collection of key/value pairs that are sorted by key based on the associated System.Collections.Generic.IComparer implementation. 7. **SortedSet<T>:** It represents a collection of objects that are maintained in sorted order. SortedDictionary<TKey, TValue>: It represents a collection of key/value pairs that are sorted on the key. LinkedList<T>: It represents a doubly linked list. **Concurrent Collection Classes in C#:** It came in .NET Framework Version 4 and onwards. It provides various threads-safe collection classes that are used in place of the corresponding types in the System.Collections and System.Collections.Generic namespaces, when multiple threads are accessing the collection simultaneously. The System.Collections.Concurrent namespace provides classes for threadsafe operations. Now multiple threads will not create problems for accessing the collection items. The System.Collections.Concurrent namespace has the following classes: 1. BlockingCollection<T>: It provides blocking and bounding capabilities for thread-safe collections that implement

5. ConcurrentDictionary<TKey, TValue>: It represents a thread-safe collection of key/value pairs that can be accessed

In the next article, I am going to discuss the Non-Generic ArrayList Collection Class in C# with Examples. Here, in this

article, I give you a brief introduction to Collections in C#. I hope this article will help you with your needs. I would like to have

Pranaya Rout has published more than 3,000 articles in his 11-year career. Pranaya Rout has very good experience with

Microsoft Technologies, Including C#, VB, ASP.NET MVC, ASP.NET Web API, EF, EF Core, ADO.NET, LINQ, SQL

Server, MYSQL, Oracle, ASP.NET Core, Cloud Computing, Microservices, Design Patterns and still learning new

Registration Open – Full-Stack .NET with Angular & ASP.NET

Advance your career with our expert-led, hands-on live training program. Get complete course details, the

syllabus, and Zoom credentials for demo sessions via the links below.

Registration Form Join Telegram Group Contact: +91 70218 01173 (Call / WhatsApp) Previous Lesson **Next Lesson** ArrayList in C# Advantages and Disadvantages of Arrays in C#

in C# ✓ How to Use Abstract Classes and Methods in C#

Methods in C# Sealed Class and Sealed Methods in C# Extension Methods in C# Static Class in C# ✓ Variable Reference and Instance of a Class in C# OOPs Real-Time Examples Real-time Examples of Encapsulation Principle in C# Real-Time Examples of Abstraction Principle in C# Real-Time Examples of Inheritance Principle in C#

Course Structure of Events, Delegates and Lambda Expression Roles of Events, Delegates and Event Handler in C# Delegates in C# Multicast Delegates in C# Delegates Real-Time Example in C# Generic Delegates in C# Anonymous Method in C# Lambda Expressions in C# Events in C# with Examples Multi-Threading Multithreading in C#

Thread class in C#

 ▼ Thread Life Cycle in C# ▼ Threads Priorities in C# How to Terminate a Thread in C# Inter Thread Communication in C# How to Debug a Multithreaded Application in C# Arrays in C# 2D Arrays in C# Advantages and Disadvantages of Arrays in C# Collections in C# ArrayList in C#

Generic Constraints in C# Generic List Collection in C# How to Sort a List of Complex Type in C# Comparison Delegate in C# Dictionary Collection Class in C# Conversion Between Array List and Dictionary in C# List vs Dictionary in C# Generic Stack Collection

Class in C#

Class in C#

Class in C#

Class in C#

Foreach Loop in C#

Generic Queue Collection

✓ Generic HashSet Collection

✓ Generic SortedList Collection

Class in C# Generic SortedDictionary Collection Class in C# Class in C# Concurrent Collection in C# Collection Class in C# ConcurrentQueue Collection Class in C# Class in C#

Export and Import Excel Data in C# **Asynchronous Programming** ✓ Introduction to Concurrency Async and Await in C# ▼ Task in C# How to Return a Value from Task in C# How to Execute Multiple Tasks in C# How to Limit Number of Concurrent Tasks in C# using Cancellation Token How to Create Synchronous Method using Task in C# Retry Pattern in C# Only One Pattern in C# How to Control the Result of a Task in C# ▼ Task-Based Asynchronous

Programming in C#

Chaining Tasks by Using **Continuation Tasks**

to a Parent Task in C#

How to Cancel a Non-

Cancellable Task in C#

Asynchronous Streams in C#

✓ ValueTask in C#

How to Attached Child Tasks

Atomic Methods Thread Interlocked vs Lock in C# Parallel LINQ in C# Multithreading vs AutoMapper AutoMapper in C# AutoMapper Complex Mapping in C# Primitive Type using AutoMapper in C# in C# AutoMapper Conditional

Optional Parameter, Indexers and Enums How to make Optional Parameters in C# ✓ Indexers in C# Indexers Real-Time Example in C# Enums in C# .NET Framework Architecture Common Language Runtime in .NET Framework .NET Program Execution **Process** ✓ Intermediate Language (ILDASM & ILASM) Code in C# Common Type System in .NET Framework Common Language

DLL Hell Problem and Var, Dynamic and Reflection Reflection in C# Dynamic Type in C# ✓ Var Keyword in C# ✓ Var vs Dynamic in C# ✓ Volatile Keyword in C# Ref vs Out in C# Named Parameters in C# C# 7.X new Features C# 7 New Features in C# 7 Pattern Matching in C# Digit Separators in C# 7 ▼ Tuples in C# 7 Splitting Tuples in C# 7 Local Functions in C# 7

ReadOnly Structs in C# Pattern Matching in C# Using Declarations in C# C#8 C# ✓ Indices and Ranges in C# Operator in C# Unmanaged Constructed Types in C# Stackalloc in in C#

Mapping in C# AutoMapper Ignore Method in Fixed and Dynamic Values in **Destination Property in** AutoMapper

> Specification in .NET Framework Managed and Unmanaged Code in .NET Framework Assembly DLL EXE in .NET Framework App Domain in .NET Framework Strong and Weak Assemblies in .NET Framework How to Install an Assembly into GAC in .NET Framework Solution in .NET Framework

Dynamic vs Reflection in C# Enhancement in Out Variables Ref Returns and Ref Locals in

Static Local Functions in C# Disposable Ref Structs in C# Nullable Reference Types in Asynchronous Disposable in Null-Coalescing Assignment

Asynchronous Streams in C# Most Popular C# Books Most Recommended C# **Books** Most Recommended Data Structure and Algorithms Books using C#

Example in C# Partial Class and Partial

Exception Handling Finally Block in C# How to Create Custom Exceptions in C# ✓ Inner Exception in C# C# Expression in C#

How to Pass Data to Thread in C# Thread Function in C# Join Method and IsAlive C# Lock in C# Monitor Class in C# Mutex Class in C# Semaphore Class in C# Deadlock in C# Performance Testing of a ▼ Thread Pool in C# Foreground and Background

Threads in C#

AutoResetEvent and

ManualResetEvent in C#

Collections in C# ✓ Hashtable in C# Advantages and Disadvantages of Non-Generics in C#

> ConcurrentDictionary ConcurrentStack Collection ConcurrentBag Collection Class in C# ✓ BlockingCollection in C# File Handling File Handling in C# FileStream Class in C# StreamReader and StreamWriter in C# File Class in C# C# BinaryWriter and BinaryReader in C# StringWriter and StringReader in C#

 How to Cancel Asynchronous Stream in C# Parallel Programming ▼ Task Parallel Library in C# Parallel For in C# Parallel Foreach Loop in C# Parallel Invoke in C# Maximum Degree of Parallelism in C# How to Cancel Parallel Operations in C# Asynchronous Programming

C# 7 Types in C# 7 in C#

Application ✓ Interface in C# ✓ Interface Interview Questions and Answers in C# ✓ Interface Realtime Examples in C# Multiple Inheritance in C# Multiple Inheritance Realtime Polymorphism in C# Method Overloading in C# Operator Overloading in C# Method Overriding in C# Method Hiding in C#

Real-Time Examples of Polymorphism Principle in C# Real-Time Examples of Interface in C# Real-Time Examples of Abstract Class in C# Exception Handling in C# ✓ Multiple Catch Blocks in C# Exception Handling Abuse in Events, Delegates and Lambda

Function in Type Safe Manner How to Retrieve Data from a Property of Thread Class in ▼ Thread Synchronization in C# SemaphoreSlim Class in C# Multithreaded Application

Non-Generic Stack in C# Non-Generic Queue in C# Non-Generic SortedList in C# Generic Collection in C# Generic Collections in C#

TextWriter and TextReader in FileInfo Class in C# DirectoryInfo Class in C#

Safety and Race Conditions in vs Parallel Programming in C# How to Map Complex Type to AutoMapper Reverse Mapping

Generalized Async Return Expression Bodied Members ▼ Thrown Expression in C# Async Main in C# C# 8 New Features C# 8 New Features Default Interface Methods in

About Us Privacy Policy Contact ADO.NET Tutorial Angular Tutorials ASP.NET Core Blazor Tuturials ASP.NET Core Tutorials ASP.NET MVC Tutorials ASP.NET Web API Tutorials C Tutorials C#.NET Programs Tutorials C#.NET Tutorials Cloud Computing Tutorials Data Structures and Algorithms Tutorials Design Patterns Tutorials DotNet Interview Questions and Answers Core Java Tutorials Entity Framework Tutorials JavaScript Tutorials LINQ Tutorials Python Tutorials SOLID Principles Tutorials SQL Server Tutorials Trading Tutorials JDBC Tutorials Java Servlets Tutorials Java Struts Tutorials C++ Tutorials JSP Tutorials MySQL Tutorials Oracle Tutorials ASP.NET Core Web API Tutorials HTML Tutorials • © Dot Net Tutorials | Website Design by Sunrise Pixel