Online Training
C# NET Core Training Program
C# ASP.NET Core Training
Microservices Online Training Program
Microsoft Azure Training
Introduction & Environment Setup
Introduction to C# .net
Visual Studio Express Editions
Introduction to Programming Languages
First Computer Programs
Introduction to C# .net
Different Types of Application
Programming Methodologies
Algorithms, Flowcharts, Pseudocode, Programs, and Flowcharts
Introduction to .NET Framework
.NET Framework Architecture and Components
Introduction to C# .net
Introduction to Programming Language
How to Download and Install Visual Studio on Windows
Creating First Console Application in C# .net
Different Types of Application
.NET Developer Roadmap for 2025
Coding Standard Best Practices
C# Basics
.NET Basics
Basic Structure of C# Programs
Methods and Properties of Console Class in C#
Data Types in C#
Literals in C#
Variables in C#
Tokens in C#
Operators in C#
Control Flow Statements in C#
If Else Statements in C#
Switch Statements in C#
While Loop in C#
While Loop in C#
Do While Loop in C#
For Loop in C#
Break Statement in C#
Continue Statement in C#
Goto Statement in C#
Functions in C#
User Defined Functions in C#
Call By Value and Call By Reference in C#
Recursion in C#
Scope and Lifetime in C#
Private Constructors in C#
Destructors in C#
Exchange Collection in .NET Framework
Difference Between Positive and Negative Int
Access Specifiers in C#
Encapsulation in C#
Abstraction in C#
Inheritance in C#
Inheritance in C#
How to use Inheritance in Application Development
Is-A and Has-A Relationship in C#
Sealed Classes in C#
Static Classes in C#
Static in Non Static Classes
Static Members in C#
Static Constructor in C#
Difference Between Constructor Chaining and Calling Method in C#
Constructor and Overloaded Key words in C#
Stack and Heap Memory in .NET
Boxing and Unboxing in C#
OOPs in C#
Object Oriented Programming (OOPs) in C#
Class and Objects in C#
Properties in C#
Types of Constructors in C#
Why We Need Constructors in C#
Static vs Non Static Members in C#
Private Constructors in C#
Destructors in C#
Exchange Collection in .NET Framework
Difference Between Positive and Negative Int
Access Specifiers in C#
Encapsulation in C#
Abstraction in C#
Inheritance in C#
Operator Overloading in C#
Method Overloading in C#
Method Hiding in C#
Partial Class and Partial 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 in C#
Real-Time Examples of Inheritance in C#
Real-Time Examples of Polymorphism in C#
Real-Time Examples of Abstraction in C#
Real-Time Examples of Inheritance Principle in C#
Real-Time Examples of Polymorphism Principle in C#
Real-Time Examples of Abstraction Principle in C#
Real-Time Examples of Interface in C#
Real-Time Examples of Abstract Class in C#
Exception Handling
Exception Handling in C#
Multiple Catch Blocks in C#
Nested Try Catch in C#
Exceptions in C# Using Exceptions in C#
Finally Exceptions in C#
Exception Handling Abuse in C#
Events, Delegates and Lambda Expressions in C#
Course Structure of Events, Delegates and Lambda Expression
Rules 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
Multitasking in C#
Threads in C#
How to Pass Data to Thread Function in Type Safe Manner in C#
How to Retrieve Data from a Thread Function in C#
Foreground and Background Threads in C#
Thread Life Cycle in C#
Thread Priorities in C#
How to Terminate a Thread in C#
How Thread Communication in C#
How to Debug a Multi-threaded Application in C#
Collections in C#
Array in C#
2D Array in C#
Jagged Array in C#
Advantages and Disadvantages of Arrays in C#
Collections in C#
Arraylist in C#
Hashtable in C#
Non Generic Stack in C#
Non Generic Queue in C#
Non Generic Sortedlist in C#
Advantages and Disadvantages of Non Generic Collection in C#
Generic Collection in C#
Generic Stack in C#
Generic Queue in C#
Generic Constructors in C#
Generic Sort Collection in C#
How to Sort a List of Complex Type Data in C#
Comparable and Comparer in C#
Dictionary Collection Class in C#
Conversion Between Array, Non Generic Collection and Generic Collection in C#
Generic Stack Collection Class in C#
Generic Queue Collection Class in C#
Linked List in C#
Generic HashSet Collection Class in C#
Generic SortedSet Collection Class in C#
Generic LinkedList Collection Class in C#
Generic SortedDictionary Collection Class in C#
Concurrent Collection in C#
ConcurrentDictionary Collection Class in C#
ConcurrentQueue Collection Class in C#
ConcurrentStack Collection Class in C#
ConcurrentBag Collection Class in C#
Blocking Collection in C#
File Handling
File Handling in C#
Text File Handling in C#
StreamReader and StreamWriter in C#
File Class in C#
TextWriter and TextReader in C#
BinaryWriter and BinaryReader in C#
StringWriter and StringReader in C#
FileInfo Class in C#
DirectoryInfo Class in C#
Asynchronous Programming in C#
Asynchronous Programming
Introduction to Concurrency, Async and Await in C#
Task in C#
How to Return a Value from a Task in C#
How to Execute Multiple Tasks in C#
How to Limit Number of Concurrent Tasks in C#
Task Continuation in C# using Continuation Tasks
How to Create Synchronous Method using Task in C#
Retry Patterns in C#
Task Cancellation in C#
How to Cancel the Result of a Task in C#
Task Based Asynchronous Programming in C#
Task Scheduler in C# Using TaskScheduler Class
How to Attached Child Tasks to a Parent Task in C#
IsFaulted in C#
How to Cancel a Client Initiated Task in C#
Cancellation Token in C#
How to Cancel Asynchronous Stream in C#
Parallel Programming
Task Parallel Library in C#
Parallel for in C#
Parallel ForeachLoops in C#
Parallel Invoke in C#
Maximum Degree of Parallelism in C#
How to Cancel Parallel Operations in C#
Parallel Exception Handling and Nested Exceptions in C#
Interlocked in Lock in C#
Parallel LINQ in C#
Multithreading vs Asynchronous Programming vs Parallel Programming in C#
AutoMapper
AutoMapper
How to Map Complex Type to Primitive Type using AutoMapper in C#
AutoMapper Reverse Mapping in C#
AutoMapper Conditional Mapping in C#
AutoMapper Ignore Method in C#
Fixed and Dynamic Values in Destination Property in AutoMapper
Optional Parameter, Indexers and Enums
Optional Parameters in C#
Indexers in C#
Indexers Real Time Example in C#
Enums in C#
.NET Framework Architecture
.NET Framework
Common Language Runtime (CLR) in .NET Framework
.NET Project Execution Process
Intermediate Language (CIL)/MSIL & IL/OPC Code in .NET
Common Type System in .NET Framework
Common Language Specification in .NET Framework
Managed and Unmanaged Code in .NET Framework
Assembly DLL/EXE in .NET Framework
Namespaces in .NET Framework
Strong and Weak Assemblies in .NET Framework
How to Install an Assembly in GAC in .NET Framework
DLL Hell Problem and Solution in .NET Framework
Var, Dynamic and Reflection
Reflection in C#
Dynamic Type in C#
Var Keyword in C#
Var vs Dynamic in C#
Dynamic vs Reflection in C#
Dynamic Keyword in C#
Var vs Dynamic in C#
Var Keyword in C#
C# 7.0 Live Features
C# 7.0New Features
Enhancement in Out Variables in C# 7
Pattern Matching in C#
Tuple Expressions in C#
Tuples in C# 7
Splitting Tuples in C# 7
Local Functions in C# 7
Ref Returns and Ref Locals in C#
Discards in C#
Generalized Async Return Types in C# 7
Expression Bodied Members in C#
Throwing Expression in C#
Async Main in C#
C# 8 New Features
C# 8.0 Features
Readonly Structure in C#
Default Interface Methods in C#
Pattern Matching in C#
Using Declarations in C#
Static Local Functions in C#
Disposable Ref Structs in C#
Nullable Reference Types in C#
Indices and Ranges in C#
Null Coalescing Assignment Operator in C#
Unmanaged Constructed Types in C#
Stackalloc in C#
Most Popular C# Books
Pro C# 10 with .NET 6 Recommended Books
Best Recommended Data Structure and Algorithms Books using C#

Advantages and Disadvantages of Arrays in C##

Back to: C# .NET Tutorials For Beginners and Professionals

Advantages and Disadvantages of Arrays in C#

In this article, I am going to discuss the **Advantages and Disadvantages of Arrays in C#** with examples. This is one of the most frequently asked C# interview questions. Before proceeding to this article, I strongly recommend you read the following two articles:

1. Arrays in C# – Here we discussed the basic concepts of arrays along with a one-dimensional array with examples.
2. Two-Dimensional Arrays in C# – Here we discussed the two-dimensional arrays along with jagged arrays in C# with examples.

Advantages of using an Array in C#:

The advantages of using an array in C# are as follows:

1. It is used to store similar types of multiple data items using a single name.
2. We can access or represent other data structures such as queue, stack, trees, lists, graphs, stacks, queues, etc.
3. The two-dimensional arrays in C# are used to represent matrices.
4. The Arrays in C# are strongly typed. That means they are used to store similar types of multiple data items using a single name.

As the arrays are strongly typed so we are getting two advantages. First, the performance of the application will be much better because boxing and unboxing will not happen. Secondly, runtime errors will be prevented because of a type mismatch. In this case, at compile time a type error will occur if there is a type mismatch.

Example: Arrays gives us compilation error when type mismatch

In the following example, we create an integer array with the name numbersArray. As it is an integer array so we can store only the integer values into the array. As you can see when we try to store a string value within the array, immediately it gives us a compiler error saying "**cannot implicitly convert string to int**". This is the reason why we call the arrays in C# strongly typed.

```
class Program
{
    static void Main(string[] args)
    {
        //Creating an Integer Array to store 3 integer numbers
        int[] numbersArray = new int[3];

        //Storing Integer Number is fine
        numbersArray[0] = 10;

        //Trying to store string value
        numbersArray[1] = "ABC";
    }
}
```

You can get type mismatches and runtime errors with collection classes like **ArrayList**, **Queue**, **Stack**, etc. which are present in the **System.Collections** namespace. Now, let's understand how we can overcome all these problems in our upcoming articles. But in this article, let me demonstrate the type mismatch error with an example using the ArrayList which is a collection in C#.

Type Mismatch Example using ArrayList Collection in C#:

In the following example, we create one collection using the ArrayList collection class with the name numbersArray. The collection classes that are present in the System.Collections namespace and as **ArrayList** is newly typed, Looking from the name, you can store any type of data in the collection. The **ArrayList** in C# is operated on object data type, which makes the ArrayList totally typed. So, you will not get any compile-time error, but when you run the application, you will get a runtime error.

```
using System.Collections;
namespace ArrayTest
{
    class Program
    {
        static void Main(string[] args)
        {
            //Creating a Collection using Array List
            ArrayList numbersArray = new ArrayList();
            numbersArray.Add(10);
            numbersArray.Add("ABC");

            //Iterating through the collection
            foreach (int num in numbersArray)
            {
                Console.WriteLine(num);
            }
            Console.ReadKey();
        }
    }
}
```

When you run the application, at runtime you will get the following exception. Here, it is trying to convert and store the string value into an integer variable and the casting is not valid, and hence it is throwing an exception saying that the Specified cast is not valid. If we run Array instead of ArrayList, then this kind of Exception will never get

```
//Use Get Runtime Error, when we access the 3rd Element
foreach (int num in numbersArray)
{
    Console.WriteLine(num);
}

Console.ReadKey();
```


Disadvantages of using Arrays in C#:

1. The array size is fixed. So, we should know in advance how many elements are going to be stored in the array. Once the array is created, then we can never increase or decrease the size of an array. If you want then we can do it using dynamic memory allocation, which is not recommended in the modern programming.
2. As the array size is fixed, it can allocate more memory than the required then the extra memory will be wasted. On the other hand, it can allocate less memory than the requirement, then it will create a problem.
3. We can insert an element into the middle of an array. It is also not possible to delete or remove elements from the middle of an array.

To overcome the above problems Collections are introduced in C#. And from the next and a few upcoming articles, we are going to discuss the Collection Framework in detail.

In the article, we are going to start Collections in C# with this. In this article, I try to explain the **Advantages and Disadvantages of Arrays in C#** with examples. I hope the article will help you with your needs. I would like to have your feedback. Please post your feedback, question, or comments about this article.

Dot Net Tutorials
About the Author: Pranaya Rout
Pranaya Rout has published more than 3000+ articles over his 11-year career. Pranaya Rout has very good experience with Microsoft Technologies including C#, ASP.NET MVC, ASP.NET Web API, ASP.NET Core, ASP.NET SignalR, LINQ, Entity Framework, SOLID, Design Patterns, Azure, ASP.NET Core, Cloud Computing, Microservices, Design Patterns and still learning new technologies.

Registration Open – Full-Stack .NET with Angular & ASP.NET Core
New Batch Starts: 10th September, 2025
Session Time: 8:30 PM – 10:30 PM IST
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.
View Course Details & Get Demo Credentials
Registration Form
Join Telegram Group
Contact: +91 70218 91773 (Call / WhatsApp)

Previous Lesson
2D Array in C#
Next Lesson
Collections in C#

1 thought on "Advantages and Disadvantages of Arrays in C#"

THIRUPALI RAO
APRIL 23, 2025 AT 5:01 PM

good explanation.
each explanation you give example that nice

Thanks your support for learning arrays

Reply

Leave a Reply

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

Comment *

Name *
Email *
Website

Post Comment

© Dot Net Tutorials | Website Design by Sunrise Pixel