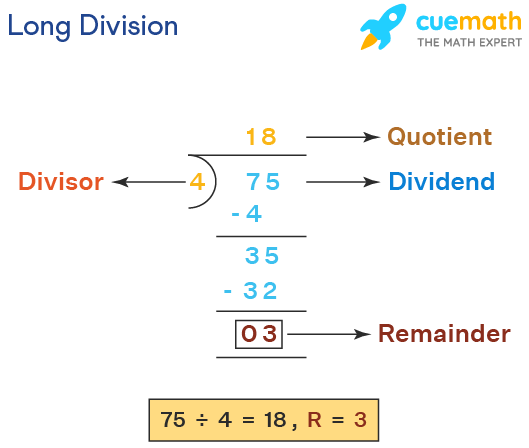
**Division and Mod Concept:**



75 (Divident) / 4 (Divisor) = 18 (Quotient), 3 (Remainder/Mod Value)

75 (Divident) mod 4 (Divisor) = 3 (Remainder/Mod Value)

**Check If a Number is Odd or Even**

10 % 2 = 0

11 % 2 = 1

**C# While Loop:**

while loop loops through a block of code as long as a specified condition is True

while (*condition*)

{

*// code block to be executed*

}

**Example:**

int i = 0;

while (i < 5)

{

Console.WriteLine(i);

i++;

}

**C# Do/While Loop**

-A variant of While Loop.

The loop will execute the **code block once**, **before checking if the condition is true**, then it will repeat the loop as long as the condition is true.

do

{

*// code block to be executed*

}

while (*condition*);

**Example:**

int i = 0;  
do

{

Console.WriteLine(i);

i++;

}

while (i < 5);

**Array:**

[**https://www.w3schools.com/cs/cs\_arrays.php**](https://www.w3schools.com/cs/cs_arrays.php)

**Default Methods and Properties:**

[**https://www.tutorialspoint.com/csharp/csharp\_array\_class.html**](https://www.tutorialspoint.com/csharp/csharp_array_class.html)

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

**Declaration:**

string[] cars;

**Declaration and Initialization:**

int[] carNo = {1,2,3,4,5};

string[] carNames = {“BMW”,”Audi”};

string[] cars = new string[4]; //// Creates array with size

**string[] cars = new string[] {"Volvo", "BMW", "Ford", "Mazda"};**

**Properties:**

carNames.**Length** //// Returns the No of Elements in an Array

**Accessing an Array Value:**

string[] cars = new string[] {"Volvo", "BMW", "Ford", "Mazda"};

**Type 1: Using Index**

Console.WriteLine(cars[0]); //// Volvo

**Type 2: Using Index and ForLoop**

for (int i = 0; i < cars.Length; i++)

{

Console.WriteLine(cars[i]);

}

**IndexOutOfRange Exception:**

(i < cars.Length) ---CHECKING if the elements is inside the bounds of ARRAY to avoid IndexOutOfArange Exception

**Type 3: Using Each Element**

foreach (string i in cars)

{

Console.WriteLine(i);

}

**Default Methods:**

Array.Sort(arrayName); //// sorts the array elements in ascending order

Array.Reverse(arrayName); //// reverse an array order

Array.FindIndex(arrayName, instance => instance.Property > 1) //// Returns the Index of the Condition

Array.Find(arrayName, instance => instance.Property > 1) //// Returns the Person Object.

Array.IndexOf(arrayCollection, arrayInstance); //// Returns Index No

Example: int personDipakIndex = Array.FindIndex(people, people => people.Id == 1);

int[] arr = { 10, 20 ,30};

Array.Reverse(arr); //// Default Method to reverse an Array

int index = Array.FindIndex(arr, ar => ar == 10);

**Types of Arrays in C#:**  
Single dimensional array

Multi-dimensional array ---->

Jagged array: Whose rows and columns are not equal

Rectangular array: Whose rows and columns are equal

**Custom Array:**

class Person {

public string Name {

get;

set;

}

public int Age {

get;

set;

}

}

Person[] people = {

new Person {

Name = "Dipak", Age = 25

},

new Person {

Name = "Karan", Age = 30

},

new Person {

Name = "Pankaj", Age = 22

}

};

Brute Force-Algorithm:

-Try solving the problem with all possible combination approach. Even if it is time consuming. If the solution for the problem exisits then try newer approach to solve the problem.

1. If you were not able to solve using brute force approach then the solution to the problem does not exists.   
     
   https://studyalgorithms.com/theory/algorithmic-paradigms---brute-force/

String:

A collection of characters.

string greeting = "Hello";

//// Length/Count of the String same as length

Console.WriteLine("The length" + greeting.Length); //// OUTPUT: 2

Default Methods:

string txt = "Hello World";

Console.WriteLine(txt.ToUpper()); // Outputs "HELLO WORLD"

Console.WriteLine(txt.ToLower()); // Outputs "hello world"

String Concatenation

The + operator can be used between strings to combine them. This is called concatenation:

string firstName = "John ";

string lastName = "Doe";

string name = firstName + lastName;

Console.WriteLine(name);

Default method:

string name = string.Concat(firstName, lastName);

String Interpolation:

Another option of string concatenation, is string interpolation, which substitutes values of variables into placeholders in a string. Note that you do not have to worry about spaces, like with concatenation:

string firstName = "John";

string lastName = "Doe";

string name = $"My full name is: {firstName} {lastName}";

Access Strings:

string myString = "Hello";

Console.WriteLine(myString[0]); // Outputs "H"

Default Methods:

myString.IndexOf(“H”);

Substring() Method

-Often combined with Index of to get the all the characters from the index

string name = "John Doe";

// Location of the letter D

int charPos = name.IndexOf("D");

// Get last name

string lastName = name.Substring(charPos);

// Print the result

Console.WriteLine(lastName);

To Analyze:

**Time Complexity** – O(1)  
**Space Complexity** – O(1)

Call By Value and Reference

Data types value and reference type

And mutable and immutable type

Datatypes which are call by value <https://dotnettutorials.net/lesson/call-by-value-and-call-by-reference-in-csharp/>

Access Modifiers