

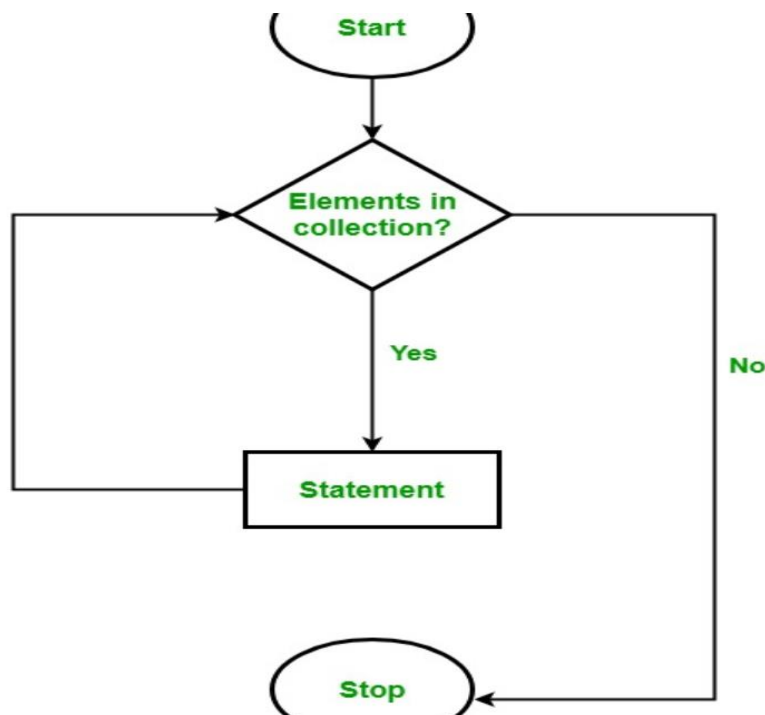
FOREACH:

The **foreach loop** is used to iterate over the elements of the collection. The collection may be an array or a list. It executes for each element present in the array.

- It is necessary to enclose the statements of foreach loop in curly braces {}.
- Instead of declaring and initializing a loop counter variable, you declare a variable that is the same type as the base type of the array, followed by a colon, which is then followed by the array name.
- In the loop body, you can use the loop variable you created rather than using an indexed array element.

Syntax:

```
foreach(data_type var_name in collection_variable)
{
    // statements to be executed
}
```





```
// Main Method
static public void Main()
{
    Console.WriteLine("Print array:");

    // creating an array
    int[] a_array = new int[] { 1, 2, 3, 4, 5, 6, 7 };

    // foreach loop begin
    // it will run till the
    // last element of the array
    foreach(int items in a_array)
    {
        Console.WriteLine(items);
    }
}
```

Output:


```
Print array:
1
2
3
4
5
6
7
```

```
for(i = 0; i <= 10; i++)
// we cannot use foreach loop in this way to print 1 to 10
// to print 1 to 10 using foreach loop we need to declare
// an array or a collection of size 10 and a variable that
// can hold 1 to 10 integer
```

Syntax of foreach Loop:

```
foreach (Data_Type variable_name in Collection_or_array_Object_name)
{
    //body of foreach loop
}
// here "in" is a keyword
```

Here *Data_Type* is a data-type of the variable and *variable_name* is the variable which will iterate the loop condition (for example, for(int i=0; i<10;i++), here i is equivalent to variable_name). The **in** keyword used in foreach loop to iterate over the iterable-item(which is here the array or the collections). The **in** keyword selects an item from the iterable-item or the array or collection on each iteration and store it in the variable(here variable_name).



```

Console.WriteLine("Array printing using foreach loop = ");

// "foreach" loop
// "ch" is the variable
// of type "char"
// "arr" is the array
// which is going to iterates
foreach(char ch in arr)
{
    Console.WriteLine(ch);
}
}

```

Output:

```

Array printing using for loop = GeeksforGeeks
Array printing using foreach loop = GeeksforGeeks

```

Note: At anywhere within the scope of foreach loop, we can break out of the loop by using the **break** keyword, and can also go to the next iteration in the loop by using the **continue** keyword.

Example: Using “continue” and “break” keyword in foreach loop

DONE!

SIMPLE METHOD:

```

static void Main(string[] args)
{
    HelloWorld();
    Console.ReadLine();
}

public static void HelloWorld()
{
    Console.WriteLine("hi wajiha arain");
}

```

Output:

Hi wajiha arain

Method	Usage
Add	Adds an element at the end of a List<T>.
AddRange	Adds elements of the specified collection at the end of a List<T>.
BinarySearch	Search the element and returns an index of the element.
Clear	Removes all the elements from a List<T>.
Contains	Checks whether the specified element exists or not in a List<T>.
Find	Finds the first element based on the specified predicate function.
Foreach	Iterates through a List<T>.
Insert	Inserts an element at the specified index in a List<T>.
InsertRange	Inserts elements of another collection at the specified index.
Remove	Removes the first occurrence of the specified element.
RemoveAt	Removes the element at the specified index.
RemoveRange	Removes all the elements that match with the supplied predicate function.
Sort	Sorts all the elements.
TrimExcess	Sets the capacity to the actual number of elements.
TrueForAll	Determines whether every element in the List<T> matches the conditions defined by the specified predicate.