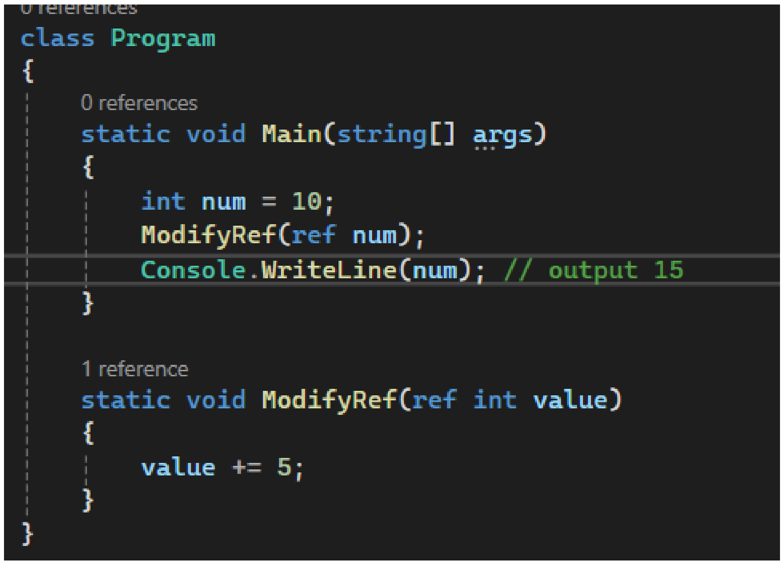
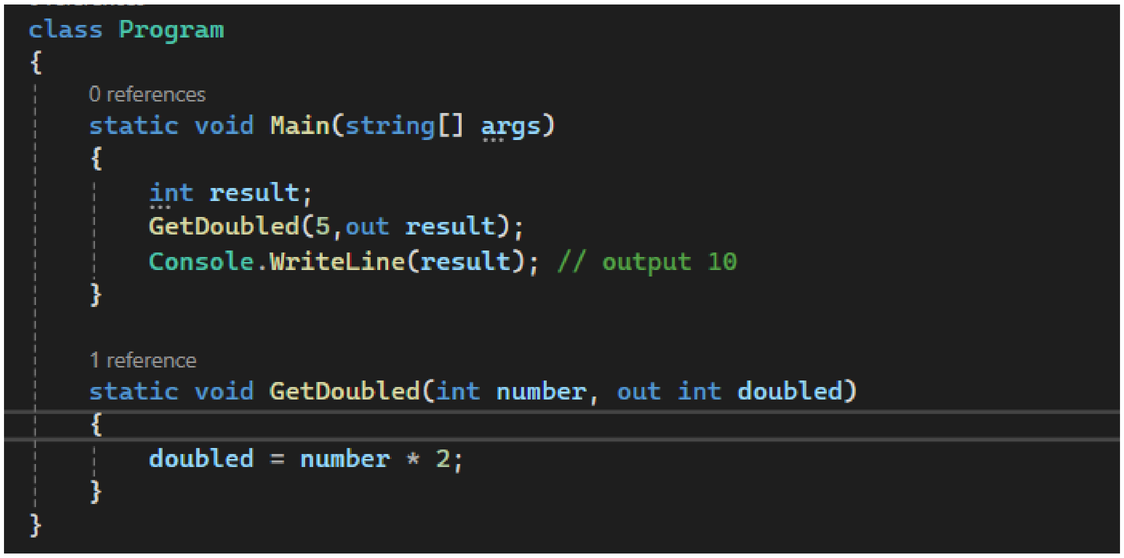
C# Parameters

1. Ref – reference parameters
    - Passes a variable by reference, meaning the method works with original variable. Not copy.
    - The variable must be initialized before passing it.
    - The method can read and write the variable.



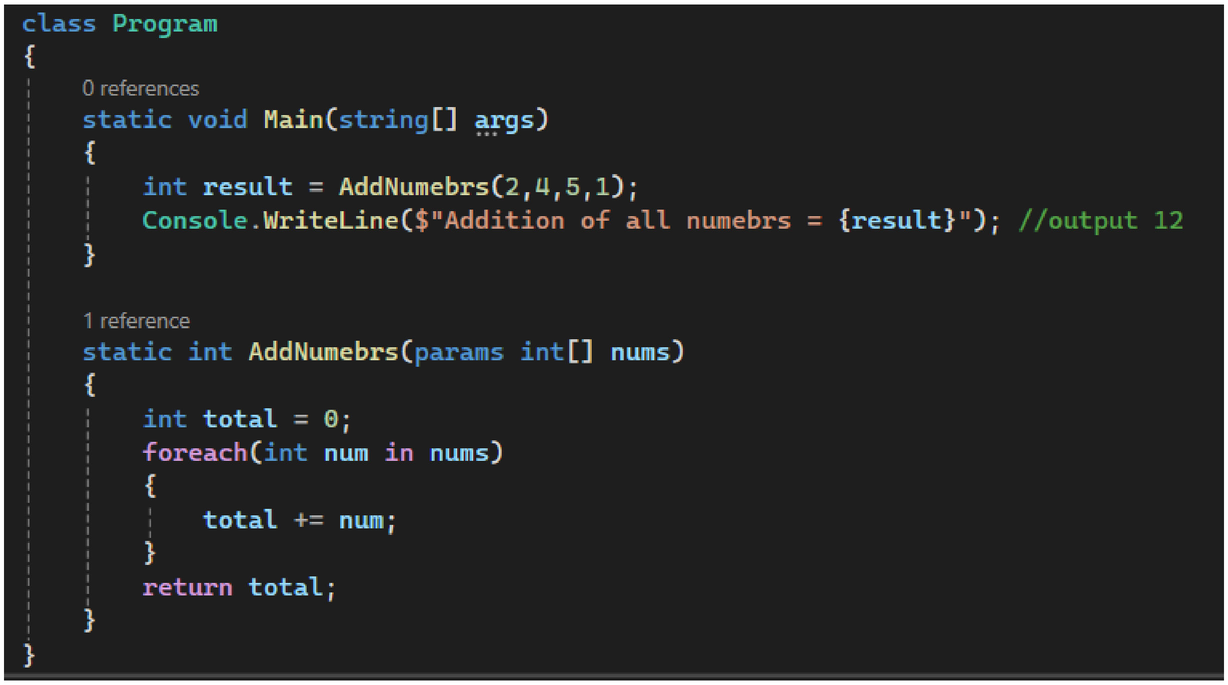
```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        int num = 10;
        ModifyRef(ref num);
        Console.WriteLine(num); // output 15
    }

    1 reference
    static void ModifyRef(ref int value)
    {
        value += 5;
    }
}
```

2. Out – output parameters
    - Also passes a variable by reference.
    - The variable does not need to initialize before passing it.
    - The method must assign a value to it before returning.
    - Useful when you want to return multiple values from a method.



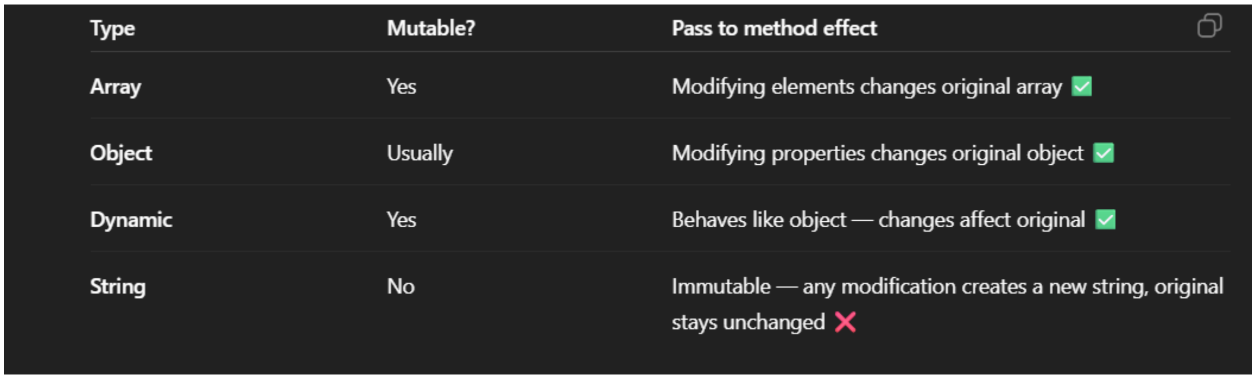
```
class Program
{
    0 references
    static void Main(string[] args)
    {
        int result;
        GetDoubled(5,out result);
        Console.WriteLine(result); // output 10
    }

    1 reference
    static void GetDoubled(int number, out int doubled)
    {
        doubled = number * 2;
    }
}
```

3. Params – parameter arrays
   - Allows you to pass zero or more arguments of the same type to a method.
   - Must be the last parameter inside the method.
   - Only one params parameter is allowed in a method.
   - Can be call with zero arguments.



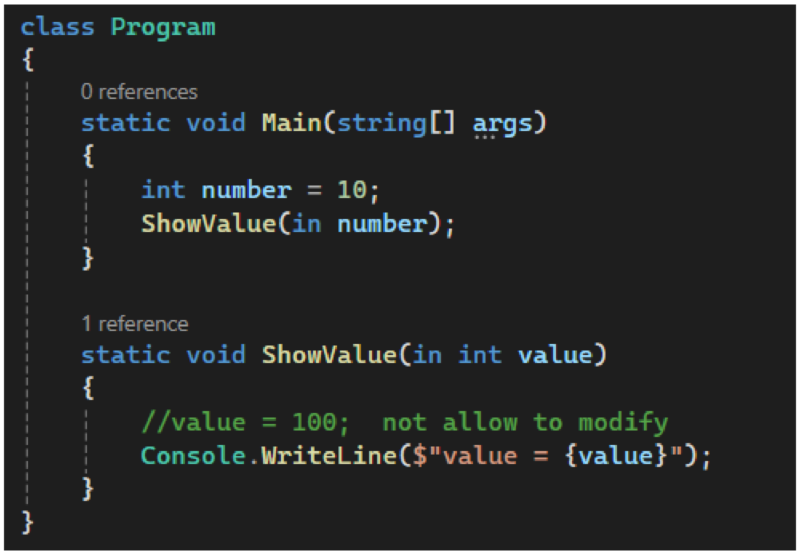
```
class Program
{
    0 references
    static void Main(string[] args)
    {
        int result = AddNumebrs(2,4,5,1);
        Console.WriteLine($"Addition of all numebrs = {result}"); //output 12
    }

    1 reference
    static int AddNumebrs(params int[] nums)
    {
        int total = 0;
        foreach(int num in nums)
        {
            total += num;
        }
        return total;
    }
}
```

4. Default – value parameter
   - Passes copy of the value.
   - Changes inside the method do not affect the original variable.



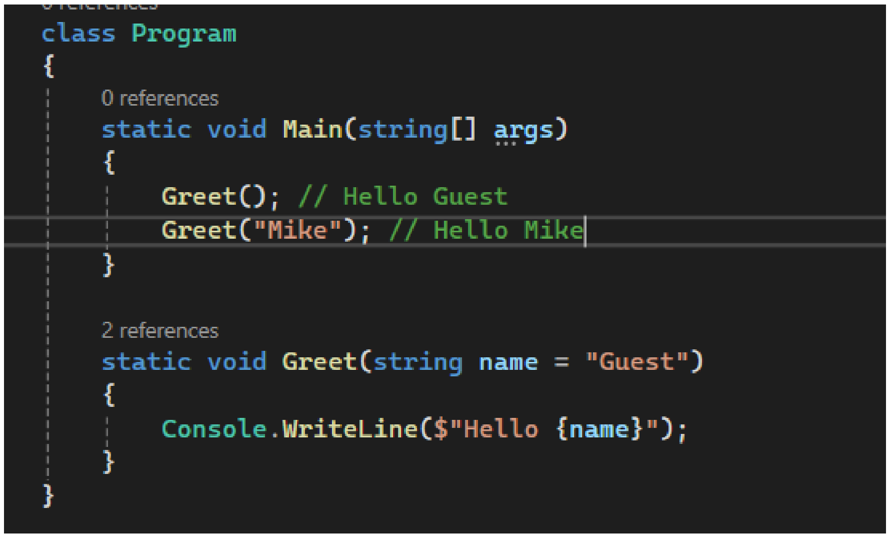
Type	Mutable?	Pass to method effect
Array	Yes	Modifying elements changes original array ✅
Object	Usually	Modifying properties changes original object ✅
Dynamic	Yes	Behaves like object — changes affect original ✅
String	No	Immutable — any modification creates a new string, original stays unchanged ❌

5. In – input parameter
    - Passes variable by reference.
    - Read only inside the method.
    - Variable must be initialized before passing.



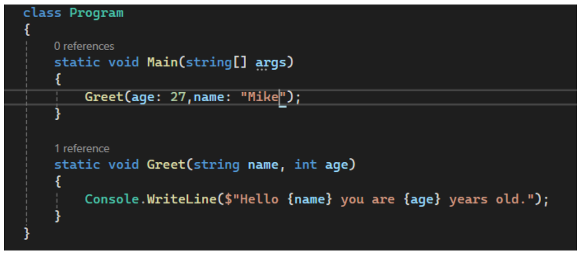
```
class Program
{
    0 references
    static void Main(string[] args)
    {
        int number = 10;
        ShowValue(in number);
    }

    1 reference
    static void ShowValue(in int value)
    {
        //value = 100;  not allow to modify
        Console.WriteLine($"value = {value}");
    }
}
```

6. Optional parameters
    - Provide a default value in the method signature.



```
class Program
{
    0 references
    static void Main(string[] args)
    {
        Greet(); // Hello Guest
        Greet("Mike"); // Hello Mike
    }

    2 references
    static void Greet(string name = "Guest")
    {
        Console.WriteLine($"Hello {name}");
    }
}
```

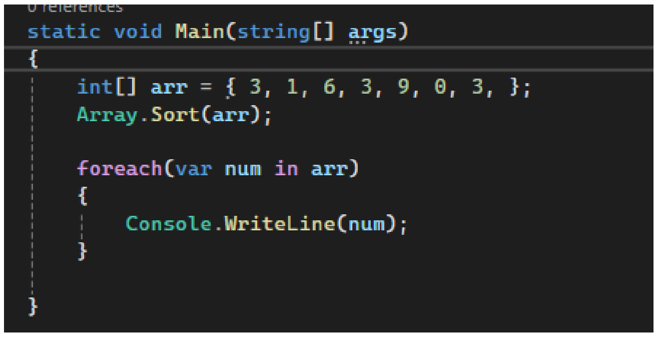
7. Named Parameters
    - pass arguments by name, not just position.



```
class Program
{
    0 references
    static void Main(string[] args)
    {
        Greet(age: 27,name: "Mike");
    }

    1 reference
    static void Greet(string name, int age)
    {
        Console.WriteLine($"Hello {name} you are {age} years old.");
    }
}
```

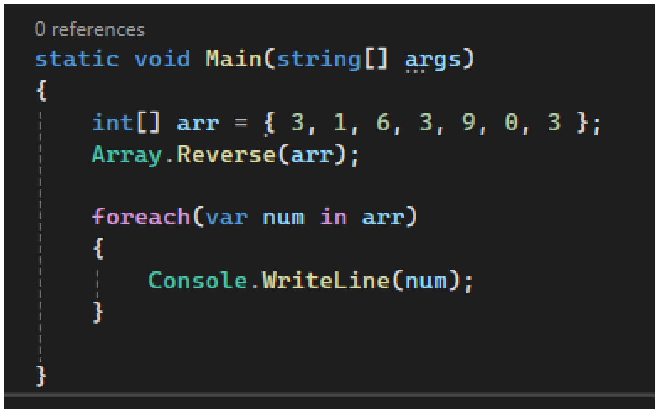
Common Built in Array Methods

8. Sorts elements in ascending order



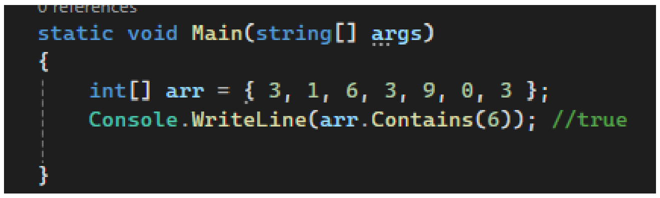
```
0 references
static void Main(string[] args)
{
    int[] arr = { 3, 1, 6, 3, 9, 0, 3, };
    Array.Sort(arr);

    foreach(var num in arr)
    {
        Console.WriteLine(num);
    }

}
```

9. Reverse elements in an array



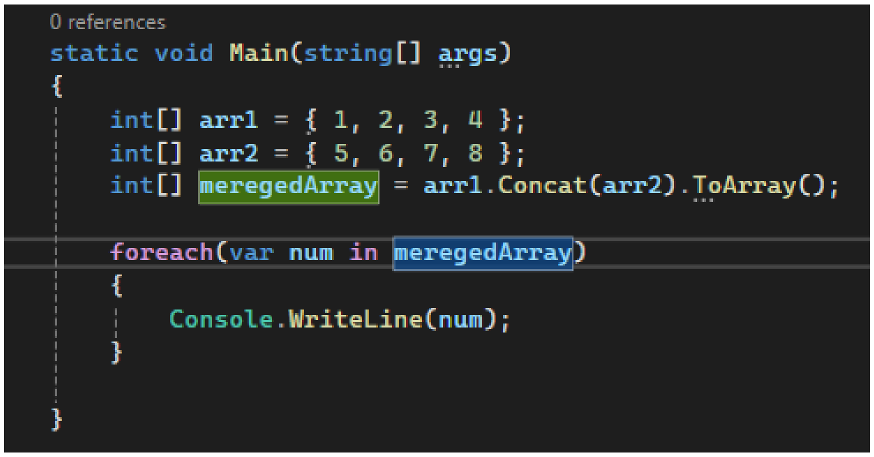
```
0 references
static void Main(string[] args)
{
    int[] arr = { 3, 1, 6, 3, 9, 0, 3 };
    Array.Reverse(arr);

    foreach(var num in arr)
    {
        Console.WriteLine(num);
    }

}
```

10. Check specific element exists in array



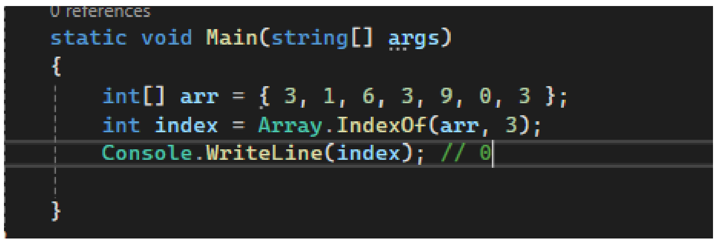
```
0 references
static void Main(string[] args)
{
    int[] arr = { 3, 1, 6, 3, 9, 0, 3 };
    Console.WriteLine(arr.Contains(6)); //true

}
```

11. Concat two arrays



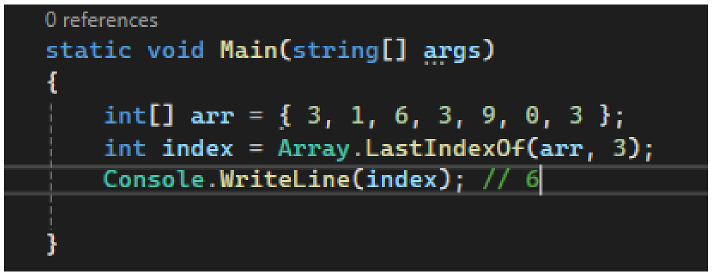
```
0 references
static void Main(string[] args)
{
    int[] arr1 = { 1, 2, 3, 4 };
    int[] arr2 = { 5, 6, 7, 8 };
    int[] meregedArray = arr1.Concat(arr2).ToArray();

    foreach(var num in meregedArray)
    {
        Console.WriteLine(num);
    }

}
```

12. Finds the index of the first occurrence of a value



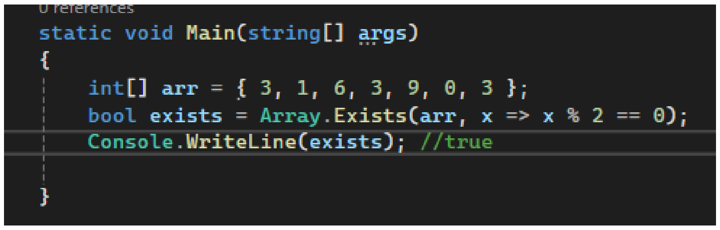
```
0 references
static void Main(string[] args)
{
    int[] arr = { 3, 1, 6, 3, 9, 0, 3 };
    int index = Array.IndexOf(arr, 3);
    Console.WriteLine(index); // 0

}
```

13. Finds the index of the last occurrence of a value



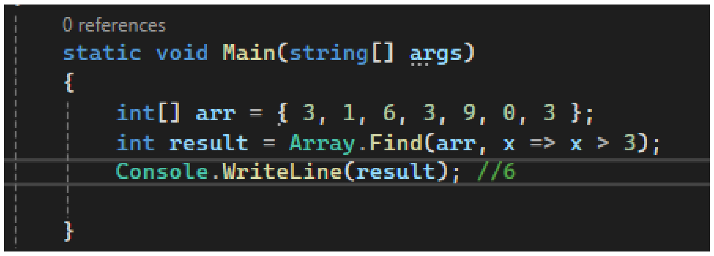
```
0 references
static void Main(string[] args)
{
    int[] arr = { 3, 1, 6, 3, 9, 0, 3 };
    int index = Array.LastIndexOf(arr, 3);
    Console.WriteLine(index); // 6

}
```

14. Checks if any element matches a condition



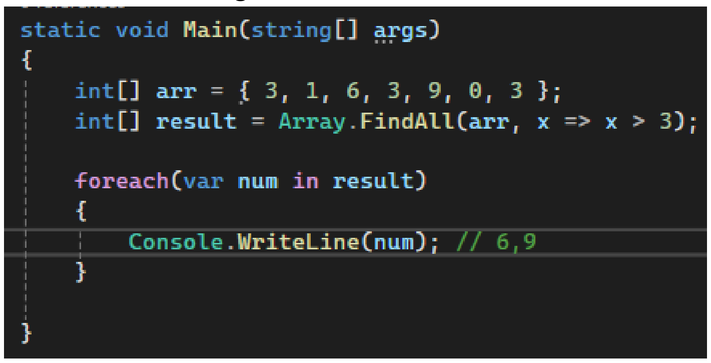
```
0 references
static void Main(string[] args)
{
    int[] arr = { 3, 1, 6, 3, 9, 0, 3 };
    bool exists = Array.Exists(arr, x => x % 2 == 0);
    Console.WriteLine(exists); //true

}
```

15. Returns the first element matching the condition



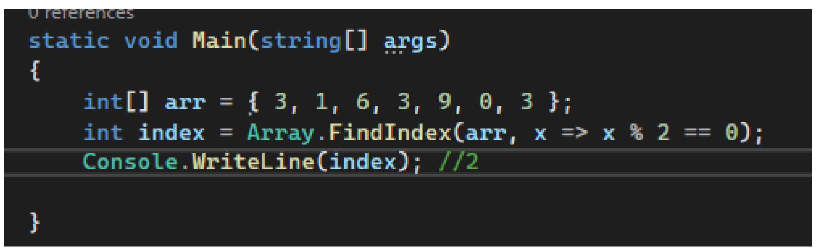
```
0 references
static void Main(string[] args)
{
    int[] arr = { 3, 1, 6, 3, 9, 0, 3 };
    int result = Array.Find(arr, x => x > 3);
    Console.WriteLine(result); //6

}
```

16. Returns all matching elements



```
static void Main(string[] args)
{
    int[] arr = { 3, 1, 6, 3, 9, 0, 3 };
    int[] result = Array.FindAll(arr, x => x > 3);

    foreach(var num in result)
    {
        Console.WriteLine(num); // 6,9
    }

}
```

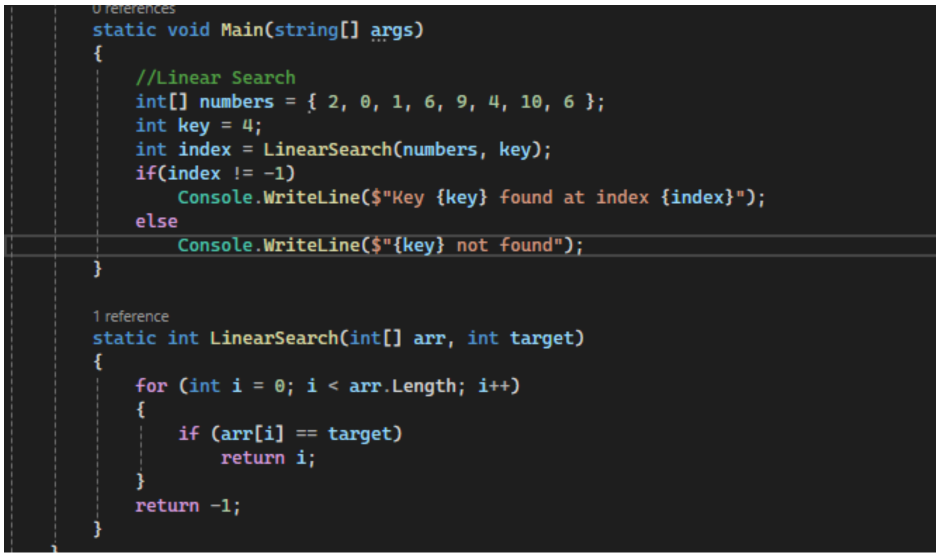
17. Gets index of first match



```
0 references
static void Main(string[] args)
{
    int[] arr = { 3, 1, 6, 3, 9, 0, 3 };
    int index = Array.FindIndex(arr, x => x % 2 == 0);
    Console.WriteLine(index); //2

}
```

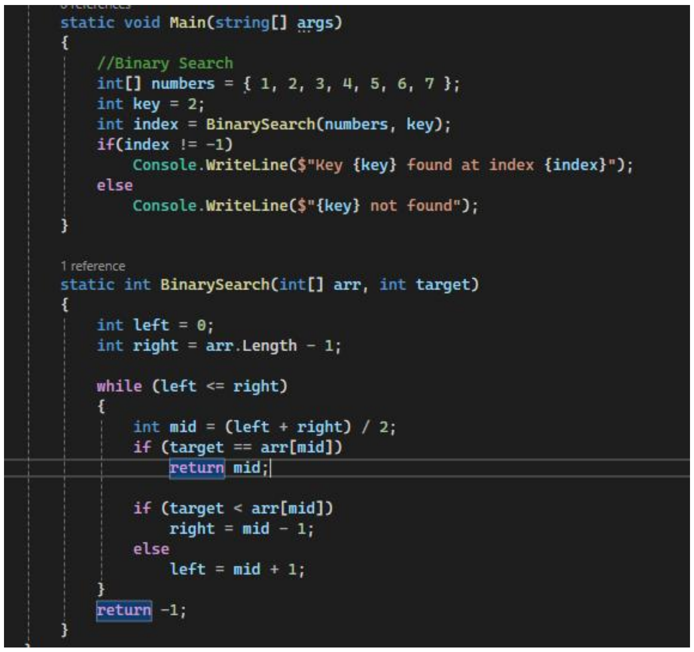
Searching Algorithms

18. Linear Search
    - Check the element one by one until the target found.
    - Time complexity => Best Case(element at start) – o(1), Worst Case(element not found/found at the end) -o(n).
    - This is good for the small or unsorted data sets.



```
0 references
static void Main(string[] args)
{
    //Linear Search
    int[] numbers = { 2, 0, 1, 6, 9, 4, 10, 6 };
    int key = 4;
    int index = LinearSearch(numbers, key);
    if(index != -1)
        Console.WriteLine($"Key {key} found at index {index}");
    else
        Console.WriteLine($"{key} not found");
}

1 reference
static int LinearSearch(int[] arr, int target)
{
    for (int i = 0; i < arr.Length; i++)
    {
        if (arr[i] == target)
            return i;
    }
    return -1;
}
```

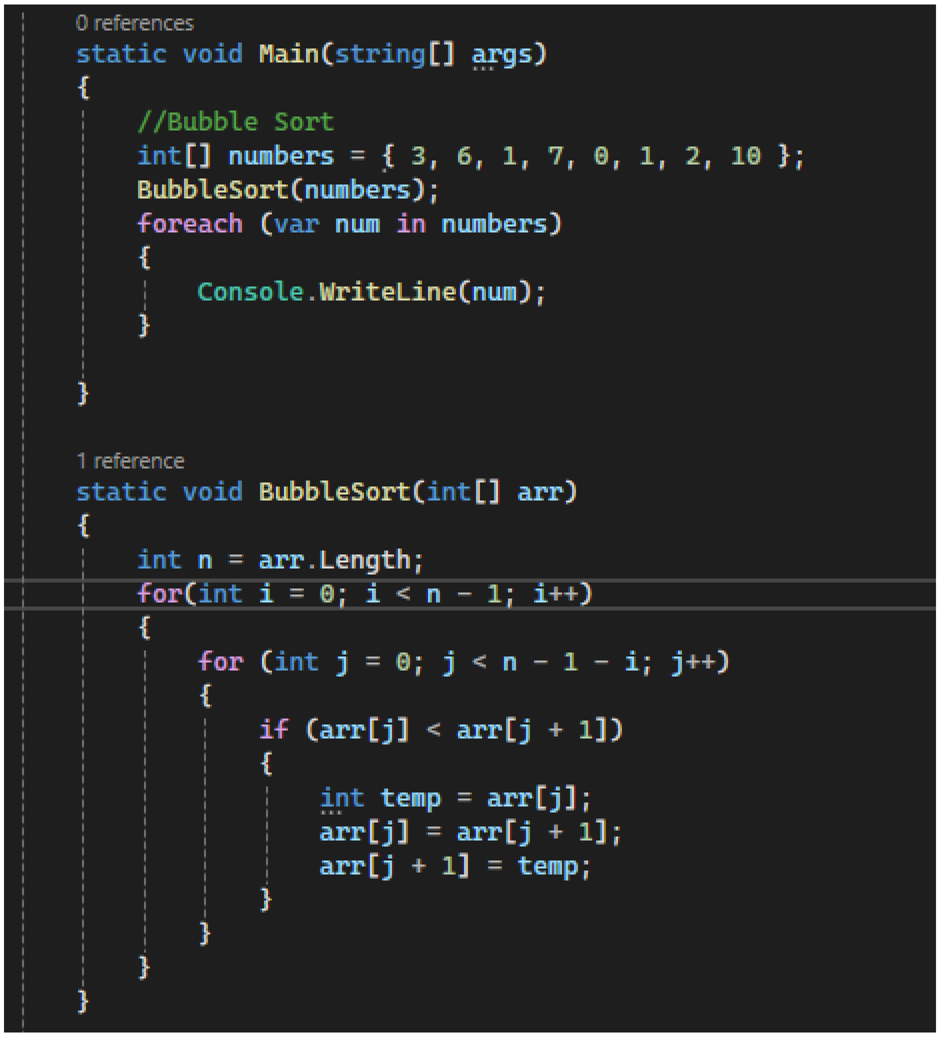
19. Binary Search
    - Works only on sorted data.
    - Repeatedly divide array into halves and check the middle element.
    - Time complexity – o(log n).
    - This is good for large sorted data sets.



```
0 references
static void Main(string[] args)
{
    //Binary Search
    int[] numbers = { 1, 2, 3, 4, 5, 6, 7 };
    int key = 2;
    int index = BinarySearch(numbers, key);
    if(index != -1)
        Console.WriteLine($"Key {key} found at index {index}");
    else
        Console.WriteLine($"{key} not found");
}

1 reference
static int BinarySearch(int[] arr, int target)
{
    int left = 0;
    int right = arr.Length - 1;

    while (left <= right)
    {
        int mid = (left + right) / 2;
        if (target == arr[mid])
            return mid;

        if (target < arr[mid])
            right = mid - 1;
        else
            left = mid + 1;
    }
    return -1;
}
```

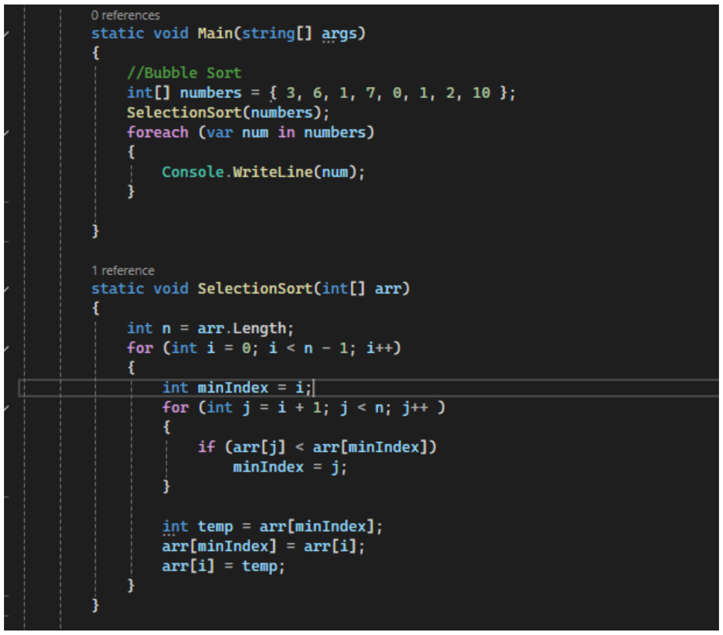
Sorting Algorithms

20. Bubble Sort
    - Repeatedly swaps adjacent elements if they are in wrong order.
    - Time complexity => Best case(when already sorted) – O(n), Worst Case – O(n^2).



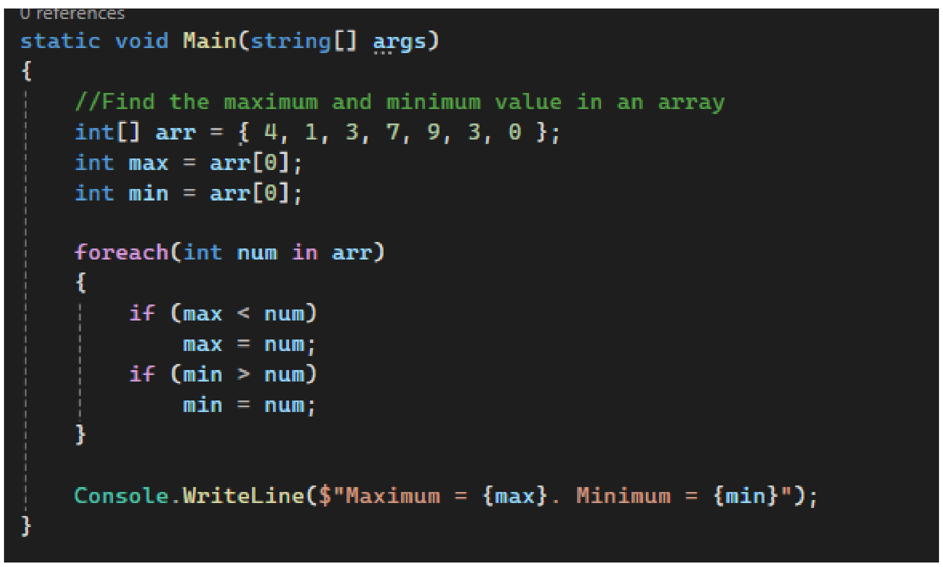
```
0 references
static void Main(string[] args)
{
    //Bubble Sort
    int[] numbers = { 3, 6, 1, 7, 0, 1, 2, 10 };
    BubbleSort(numbers);
    foreach (var num in numbers)
    {
        Console.WriteLine(num);
    }

}

1 reference
static void BubbleSort(int[] arr)
{
    int n = arr.Length;
    for(int i = 0; i < n - 1; i++)
    {
        for (int j = 0; j < n - 1 - i; j++)
        {
            if (arr[j] < arr[j + 1])
            {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
```

21. Selection Sort
    - Repeatedly finds the smallest element and places it at the beginning.
    - Time complexity => O(n^2) (always).
    - Insufficient for large data sets.

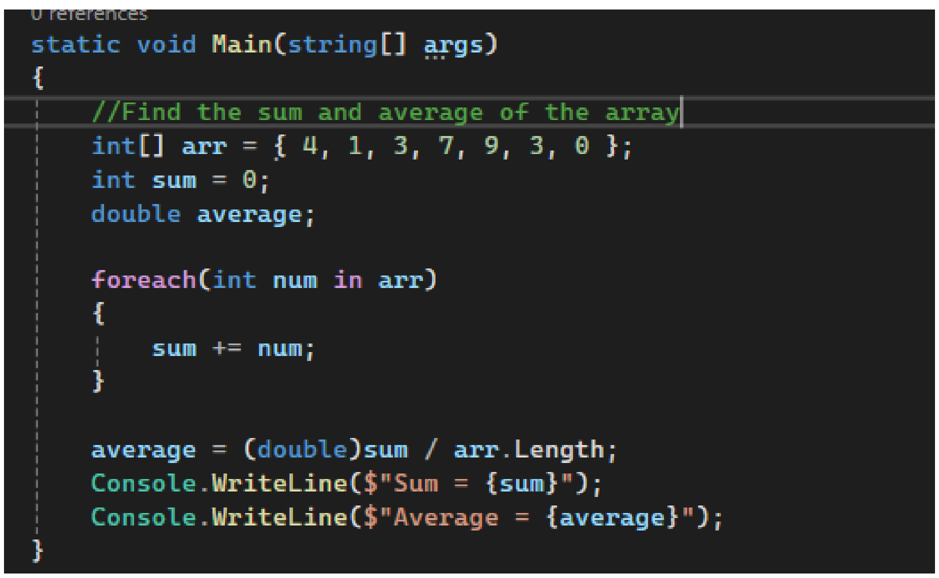


```
0 references
static void Main(string[] args)
{
    //Bubble Sort
    int[] numbers = { 3, 6, 1, 7, 0, 1, 2, 10 };
    SelectionSort(numbers);
    foreach (var num in numbers)
    {
        Console.WriteLine(num);
    }

}

1 reference
static void SelectionSort(int[] arr)
{
    int n = arr.Length;
    for (int i = 0; i < n - 1; i++)
    {
        int minIndex = i;
        for (int j = i + 1; j < n; j++ )
        {
            if (arr[j] < arr[minIndex])
                minIndex = j;
        }

        int temp = arr[minIndex];
        arr[minIndex] = arr[i];
        arr[i] = temp;
    }
}
```

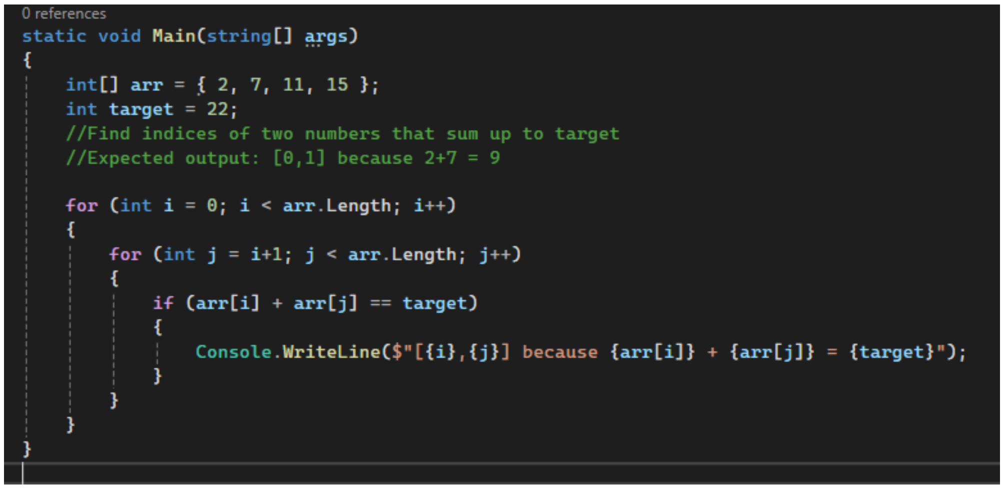
Array Operations – Practical Questions



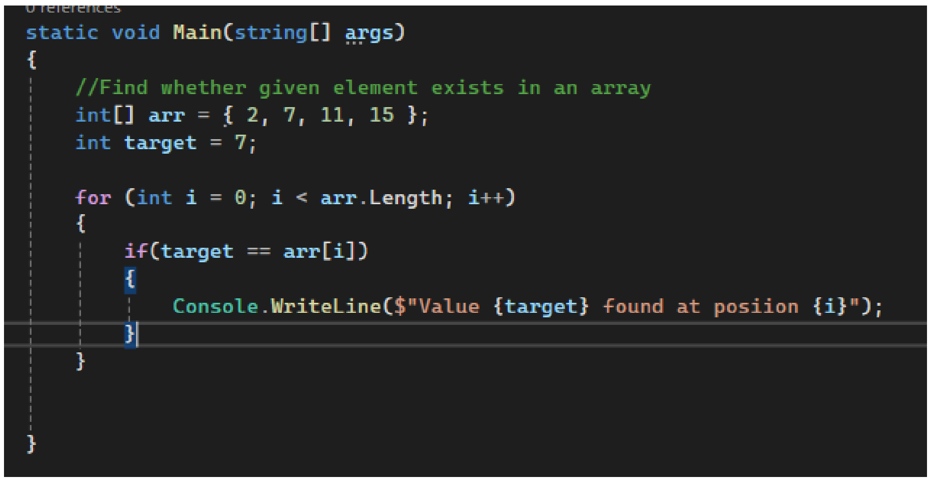
```
0 references
static void Main(string[] args)
{
    //Find the maximum and minimum value in an array
    int[] arr = { 4, 1, 3, 7, 9, 3, 0 };
    int max = arr[0];
    int min = arr[0];

    foreach(int num in arr)
    {
        if (max < num)
            max = num;
        if (min > num)
            min = num;
    }

    Console.WriteLine($"Maximum = {max}. Minimum = {min}");
}
```



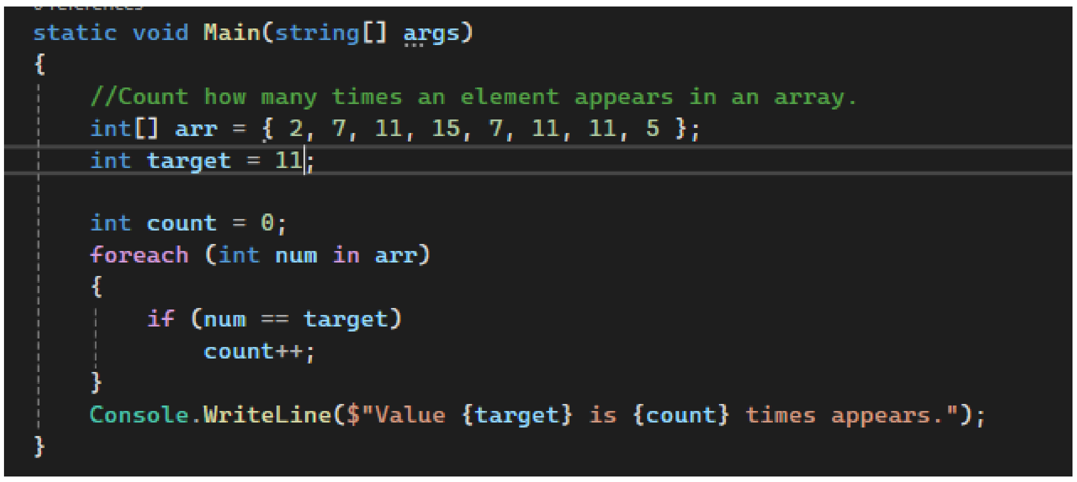
```
0 references
static void Main(string[] args)
{
    //Find the sum and average of the array
    int[] arr = { 4, 1, 3, 7, 9, 3, 0 };
    int sum = 0;
    double average;

    foreach(int num in arr)
    {
        sum += num;
    }

    average = (double)sum / arr.Length;
    Console.WriteLine($"Sum = {sum}");
    Console.WriteLine($"Average = {average}");
}
```



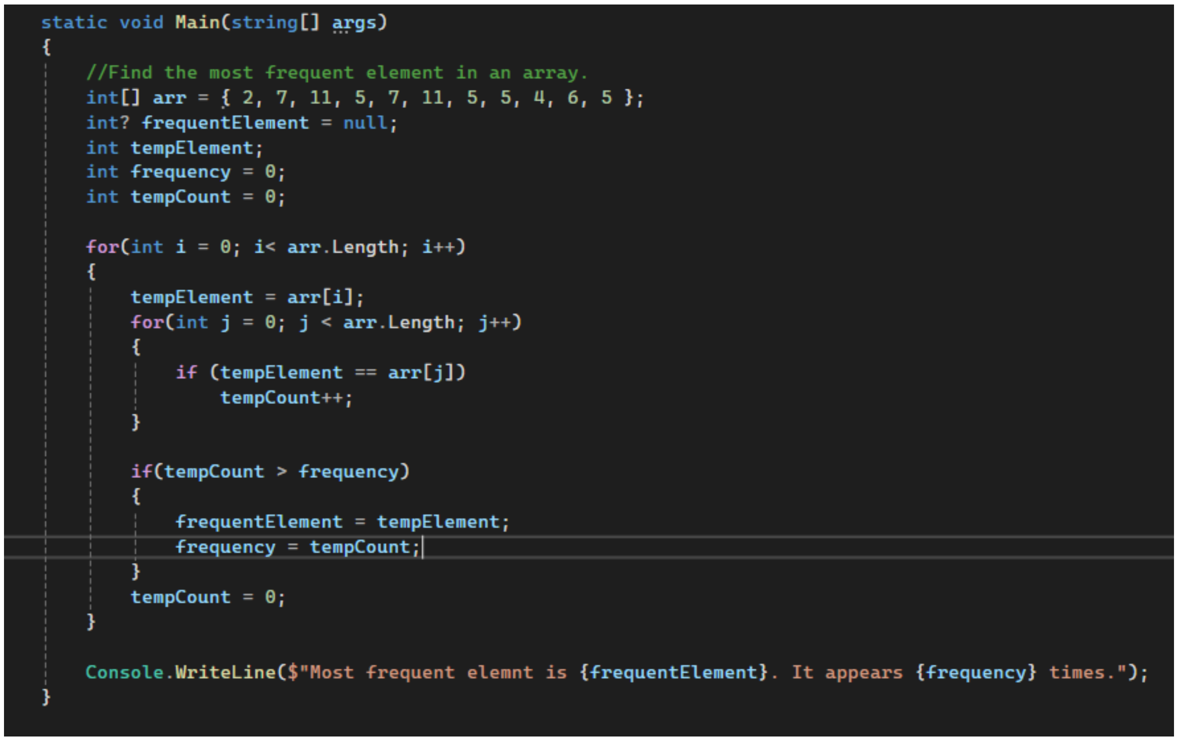
```
0 references
static void Main(string[] args)
{
    int[] arr = { 2, 7, 11, 15 };
    int target = 22;
    //Find indices of two numbers that sum up to target
    //Expected output: [0,1] because 2+7 = 9

    for (int i = 0; i < arr.Length; i++)
    {
        for (int j = i+1; j < arr.Length; j++)
        {
            if (arr[i] + arr[j] == target)
            {
                Console.WriteLine($"[{i},{j}] because {arr[i]} + {arr[j]} = {target}");
            }
        }
    }
}
```



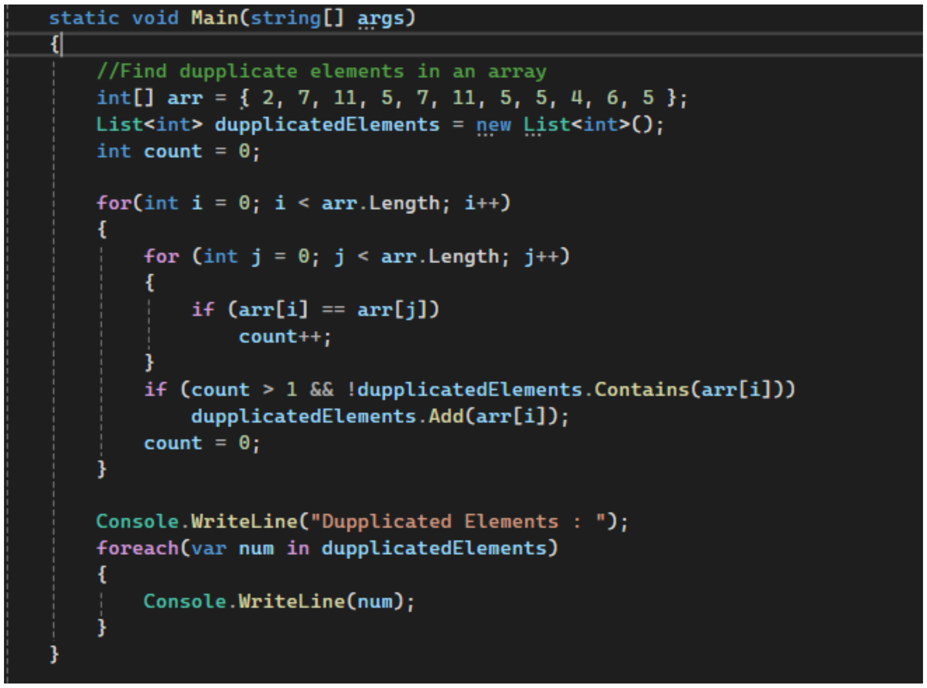
```
0 references
static void Main(string[] args)
{
    //Find whether given element exists in an array
    int[] arr = { 2, 7, 11, 15 };
    int target = 7;

    for (int i = 0; i < arr.Length; i++)
    {
        if(target == arr[i])
        {
            Console.WriteLine($"Value {target} found at posiion {i}");
        }
    }


}
```



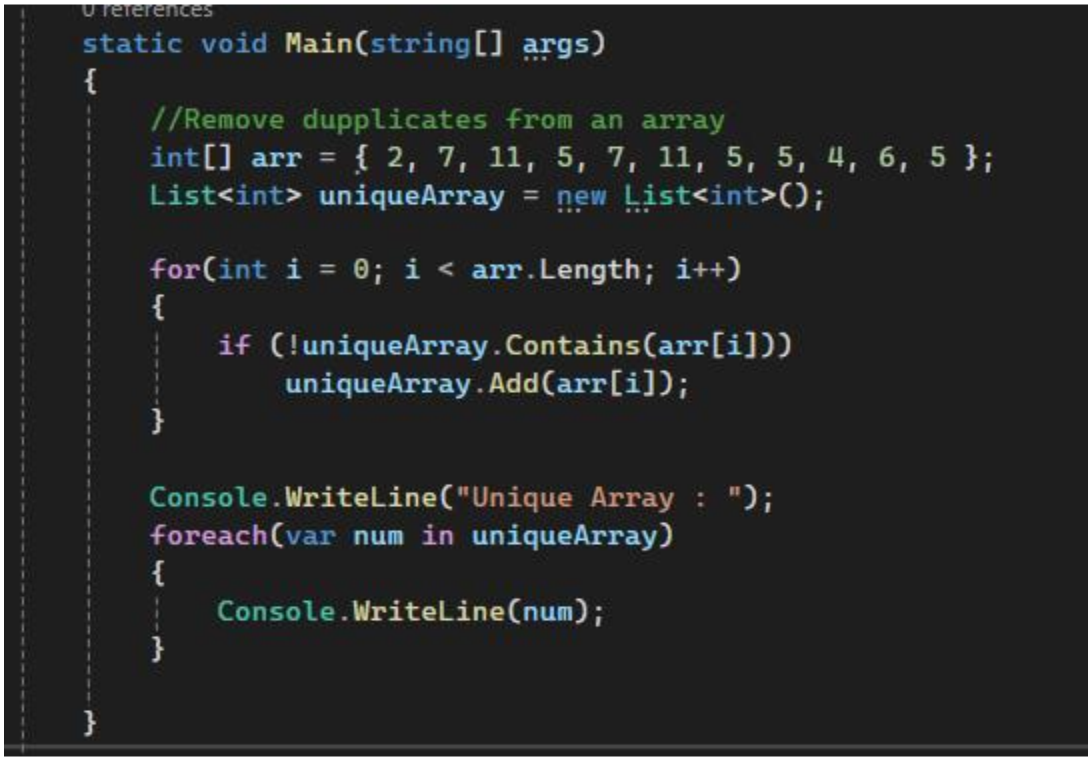
```
static void Main(string[] args)
{
    //Count how many times an element appears in an array.
    int[] arr = { 2, 7, 11, 15, 7, 11, 11, 5 };
    int target = 11;

    int count = 0;
    foreach (int num in arr)
    {
        if (num == target)
            count++;
    }
    Console.WriteLine($"Value {target} is {count} times appears.");
}
```



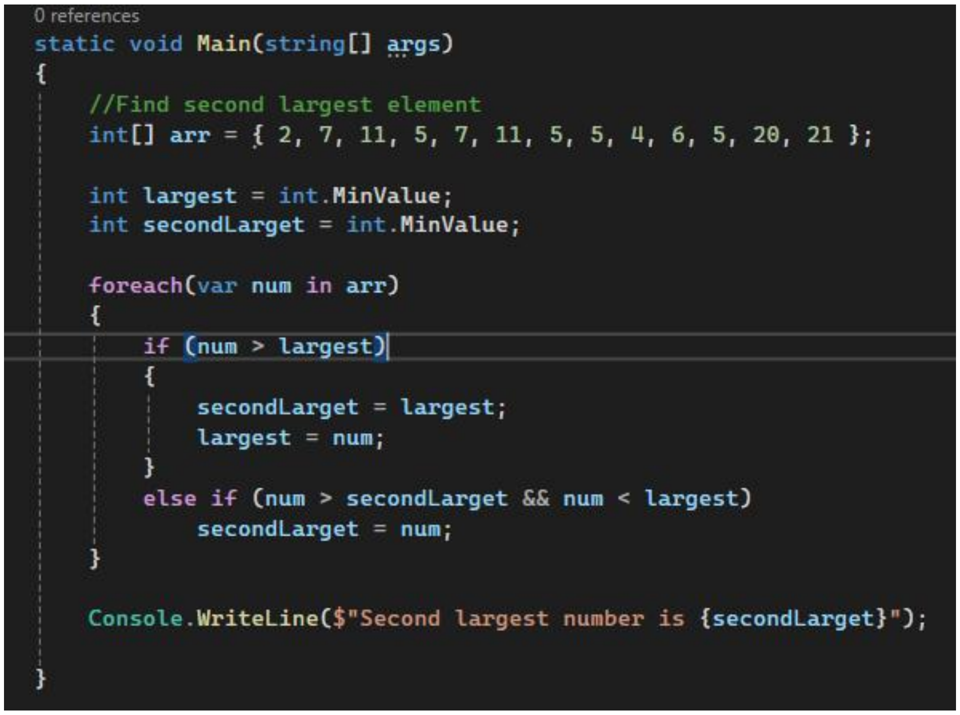
```
static void Main(string[] args)
{
    //Find the most frequent element in an array.
    int[] arr = { 2, 7, 11, 5, 7, 11, 5, 5, 4, 6, 5 };
    int? frequentElement = null;
    int tempElement;
    int frequency = 0;
    int tempCount = 0;

    for(int i = 0; i< arr.Length; i++)
    {
        tempElement = arr[i];
        for(int j = 0; j < arr.Length; j++)
        {
            if (tempElement == arr[j])
                tempCount++;
        }

        if(tempCount > frequency)
        {
            frequentElement = tempElement;
            frequency = tempCount;
        }
        tempCount = 0;
    }

    Console.WriteLine($"Most frequent elemnt is {frequentElement}. It appears {frequency} times.");
}
```



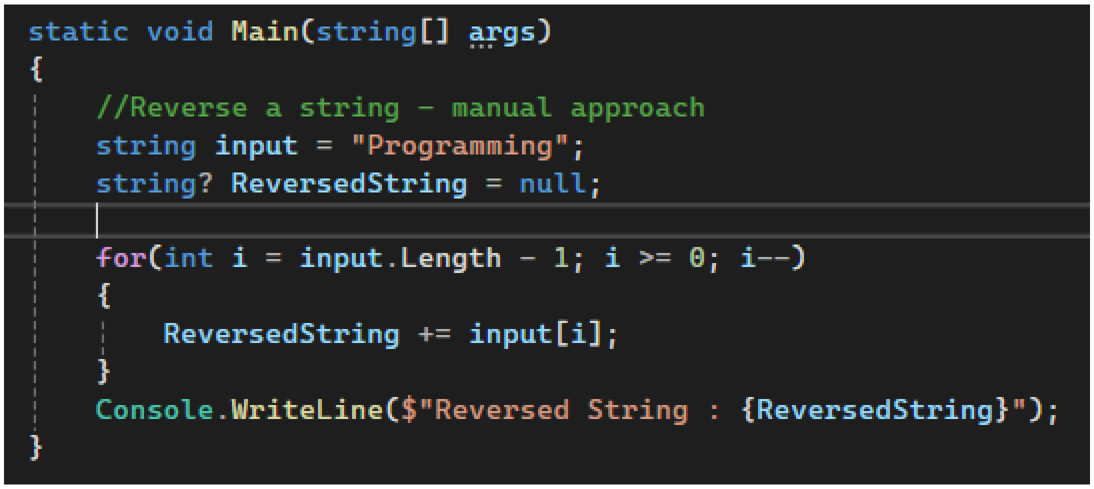
```
static void Main(string[] args)
{
    //Find dupplicate elements in an array
    int[] arr = { 2, 7, 11, 5, 7, 11, 5, 5, 4, 6, 5 };
    List<int> dupplicatedElements = new List<int>();
    int count = 0;

    for(int i = 0; i < arr.Length; i++)
    {
        for (int j = 0; j < arr.Length; j++)
        {
            if (arr[i] == arr[j])
                count++;
        }
        if (count > 1 && !dupplicatedElements.Contains(arr[i]))
            dupplicatedElements.Add(arr[i]);
        count = 0;
    }

    Console.WriteLine("Dupplicated Elements : ");
    foreach(var num in dupplicatedElements)
    {
        Console.WriteLine(num);
    }
}
```



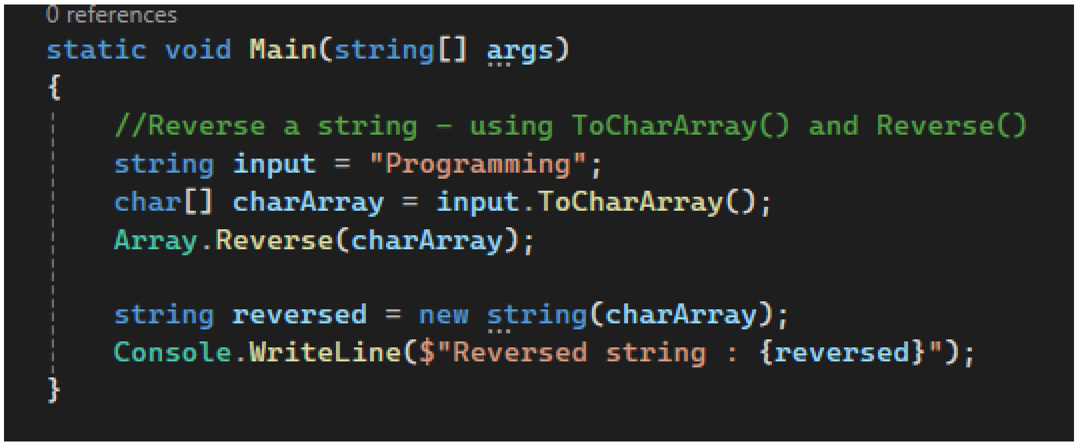
```
0 references
static void Main(string[] args)
{
    //Remove dupplicates from an array
    int[] arr = { 2, 7, 11, 5, 7, 11, 5, 5, 4, 6, 5 };
    List<int> uniqueArray = new List<int>();

    for(int i = 0; i < arr.Length; i++)
    {
        if (!uniqueArray.Contains(arr[i]))
            uniqueArray.Add(arr[i]);
    }

    Console.WriteLine("Unique Array : ");
    foreach(var num in uniqueArray)
    {
        Console.WriteLine(num);
    }

}
```



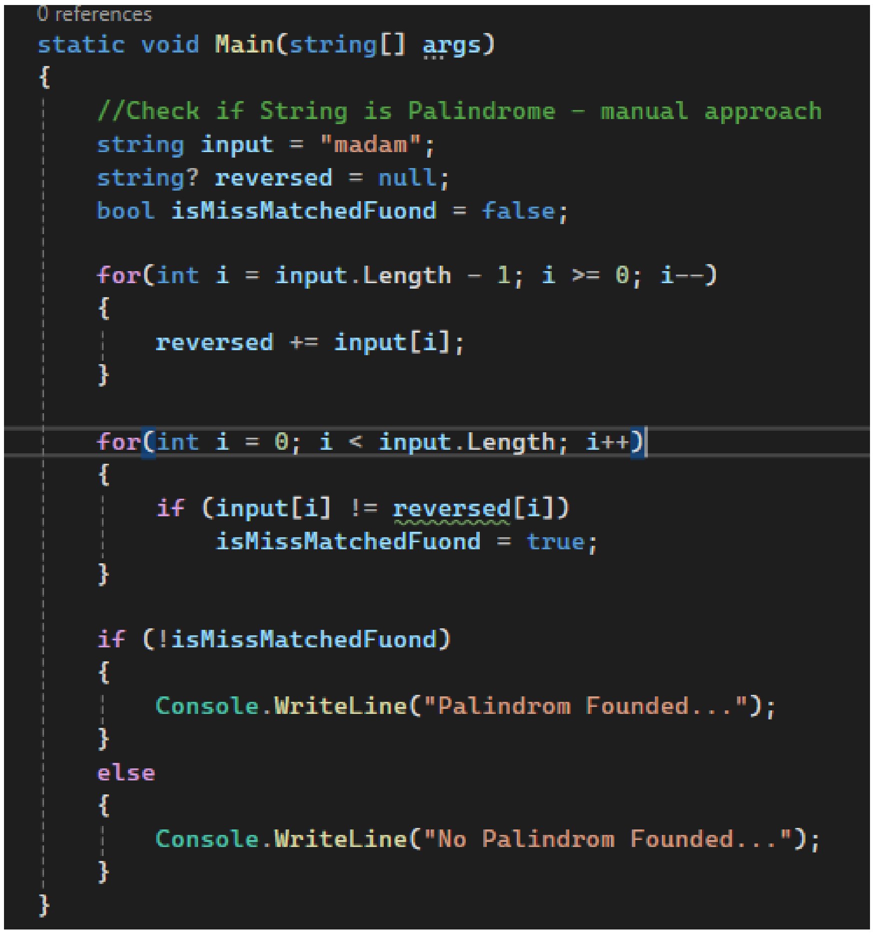
```
0 references
static void Main(string[] args)
{
    //Find second largest element
    int[] arr = { 2, 7, 11, 5, 7, 11, 5, 5, 4, 6, 5, 20, 21 };

    int largest = int.MinValue;
    int secondLarget = int.MinValue;

    foreach(var num in arr)
    {
        if (num > largest)
        {
            secondLarget = largest;
            largest = num;
        }
        else if (num > secondLarget && num < largest)
            secondLarget = num;
    }

    Console.WriteLine($"Second largest number is {secondLarget}");

}
```



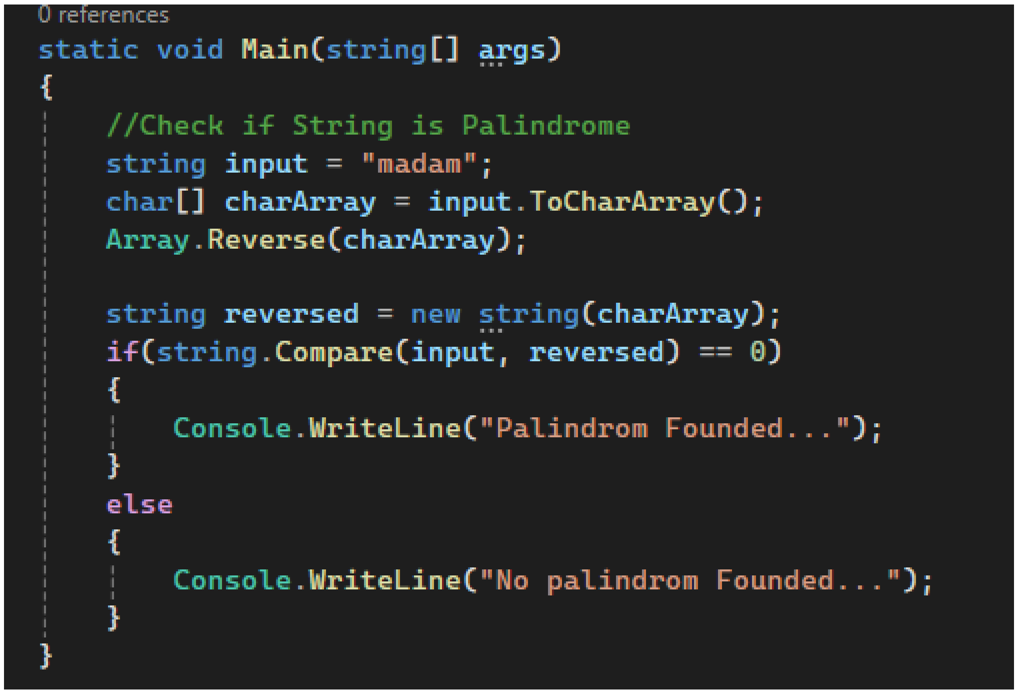
```
static void Main(string[] args)
{
    //Reverse a string - manual approach
    string input = "Programming";
    string? ReversedString = null;

    for(int i = input.Length - 1; i >= 0; i--)
    {
        ReversedString += input[i];
    }
    Console.WriteLine($"Reversed String : {ReversedString}");
}
```



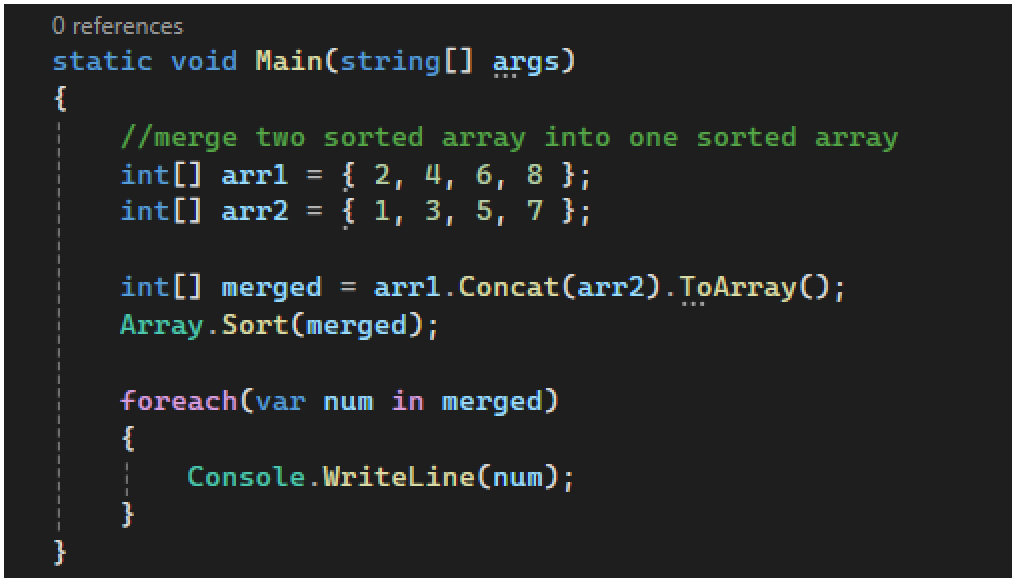
```
0 references
static void Main(string[] args)
{
    //Reverse a string - using ToCharArray() and Reverse()
    string input = "Programming";
    char[] charArray = input.ToCharArray();
    Array.Reverse(charArray);

    string reversed = new string(charArray);
    Console.WriteLine($"Reversed string : {reversed}");
}
```



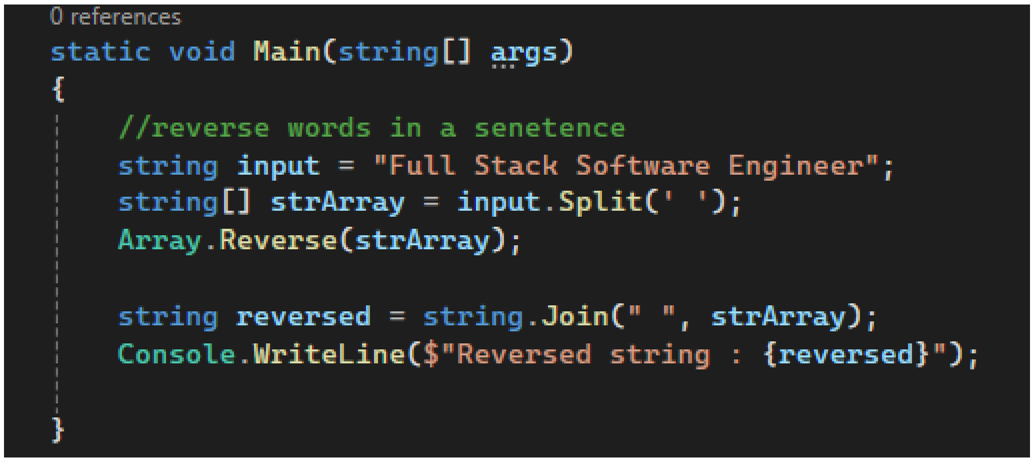
```
0 references
static void Main(string[] args)
{
    //Check if String is Palindrome - manual approach
    string input = "madam";
    string? reversed = null;
    bool isMissMatchedFuond = false;

    for(int i = input.Length - 1; i >= 0; i--)
    {
        reversed += input[i];
    }

    for(int i = 0; i < input.Length; i++)
    {
        if (input[i] != reversed[i])
            isMissMatchedFuond = true;
    }

    if (!isMissMatchedFuond)
    {
        Console.WriteLine("Palindrom Founded...");
    }
    else
    {
        Console.WriteLine("No Palindrom Founded...");
    }
}
```



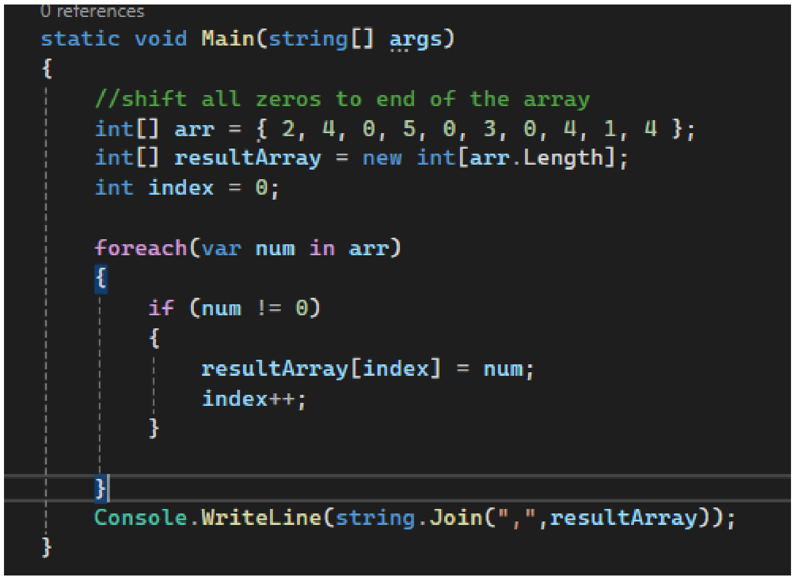
```
0 references
static void Main(string[] args)
{
    //Check if String is Palindrome
    string input = "madam";
    char[] charArray = input.ToCharArray();
    Array.Reverse(charArray);

    string reversed = new string(charArray);
    if(string.Compare(input, reversed) == 0)
    {
        Console.WriteLine("Palindrom Founded...");
    }
    else
    {
        Console.WriteLine("No palindrom Founded...");
    }
}
```



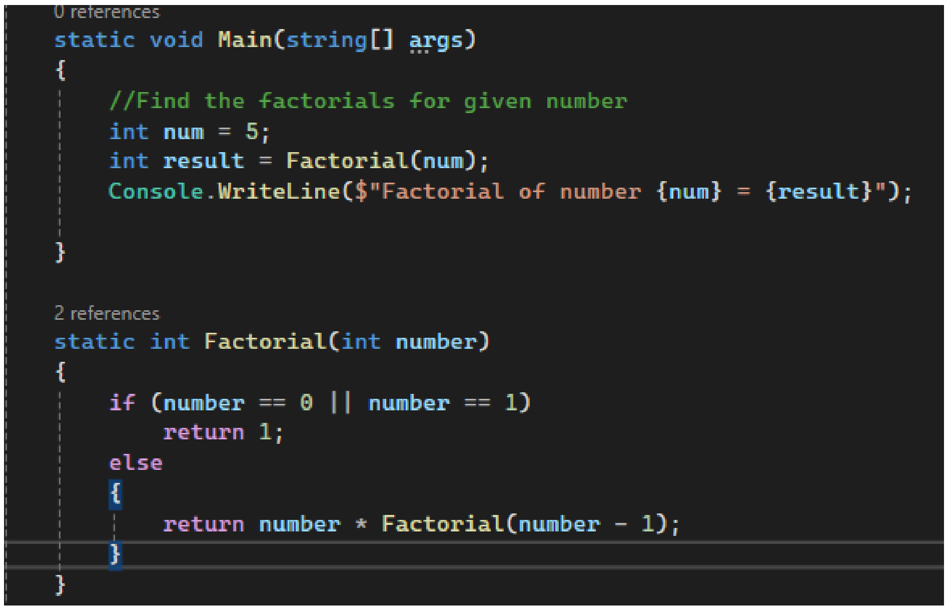
```
0 references
static void Main(string[] args)
{
    //merge two sorted array into one sorted array
    int[] arr1 = { 2, 4, 6, 8 };
    int[] arr2 = { 1, 3, 5, 7 };

    int[] merged = arr1.Concat(arr2).ToArray();
    Array.Sort(merged);

    foreach(var num in merged)
    {
        Console.WriteLine(num);
    }
}
```



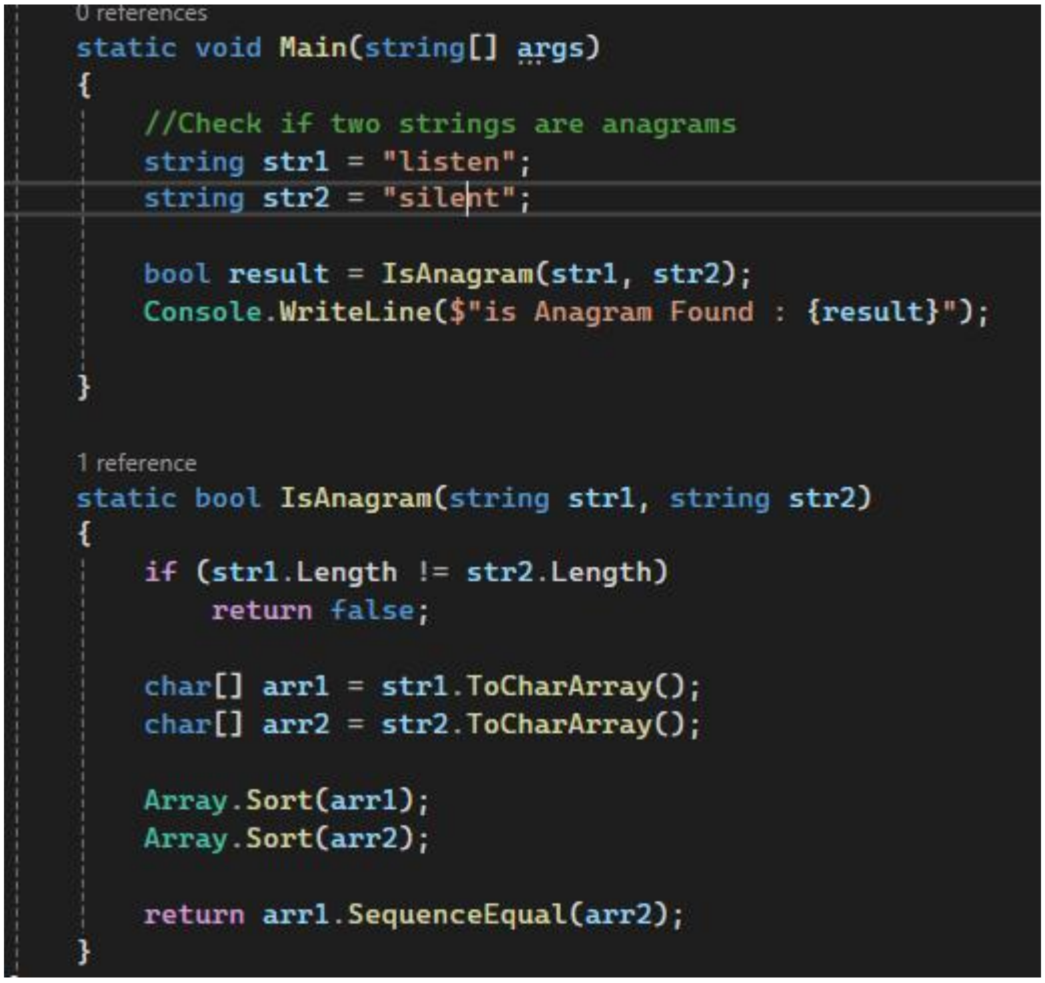
```
0 references
static void Main(string[] args)
{
    //reverse words in a senetence
    string input = "Full Stack Software Engineer";
    string[] strArray = input.Split(' ');
    Array.Reverse(strArray);

    string reversed = string.Join(" ", strArray);
    Console.WriteLine($"Reversed string : {reversed}");

}
```



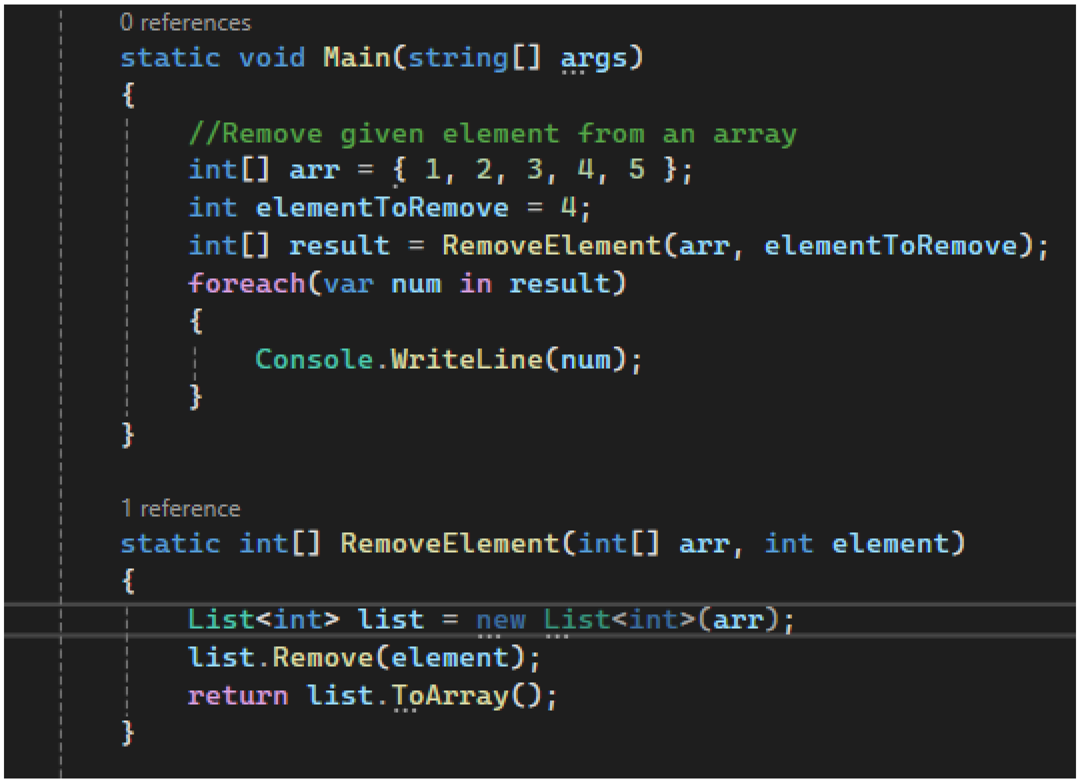
```
0 references
static void Main(string[] args)
{
    //shift all zeros to end of the array
    int[] arr = { 2, 4, 0, 5, 0, 3, 0, 4, 1, 4 };
    int[] resultArray = new int[arr.Length];
    int index = 0;

    foreach(var num in arr)
    {
        if (num != 0)
        {
            resultArray[index] = num;
            index++;
        }

    }
    Console.WriteLine(string.Join(",",resultArray));
}
```



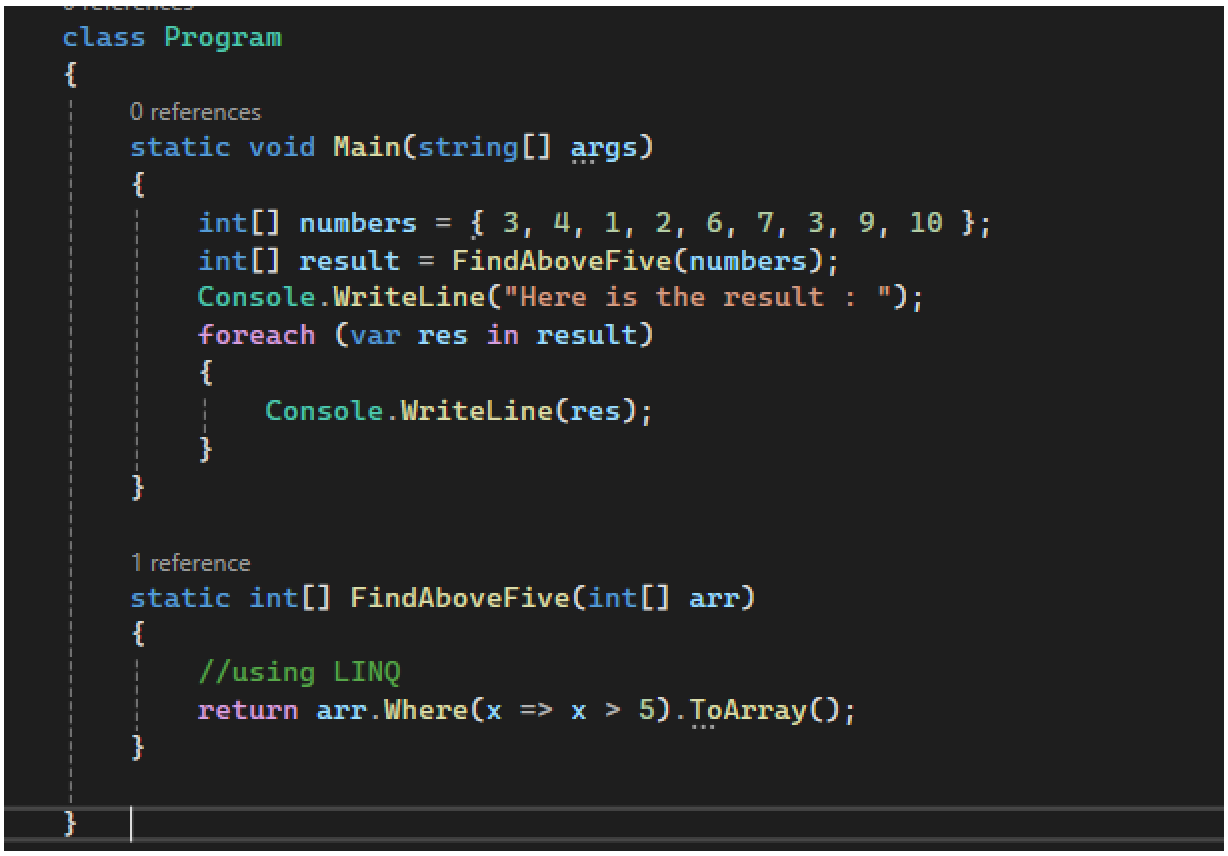
```
0 references
static void Main(string[] args)
{
    //Find the factorials for given number
    int num = 5;
    int result = Factorial(num);
    Console.WriteLine($"Factorial of number {num} = {result}");

}

2 references
static int Factorial(int number)
{
    if (number == 0 || number == 1)
        return 1;
    else
    {
        return number * Factorial(number - 1);
    }
}
```



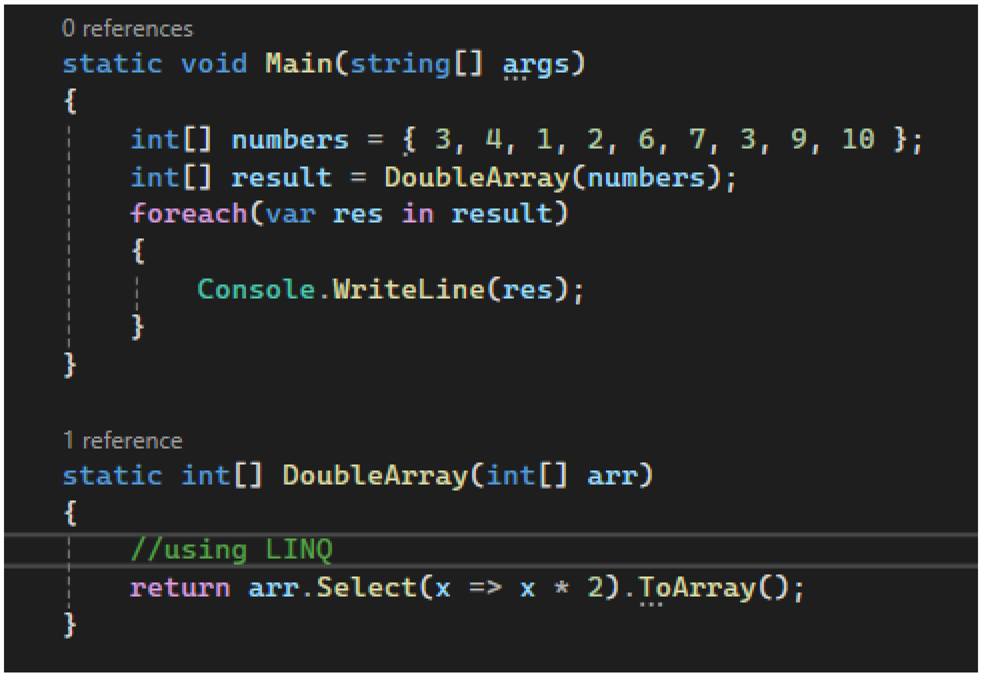
```
0 references
static void Main(string[] args)
{
    //Check if two strings are anagrams
    string str1 = "listen";
    string str2 = "silent";

    bool result = IsAnagram(str1, str2);
    Console.WriteLine($"is Anagram Found : {result}");

}

1 reference
static bool IsAnagram(string str1, string str2)
{
    if (str1.Length != str2.Length)
        return false;

    char[] arr1 = str1.ToCharArray();
    char[] arr2 = str2.ToCharArray();

    Array.Sort(arr1);
    Array.Sort(arr2);

    return arr1.SequenceEqual(arr2);
}
```



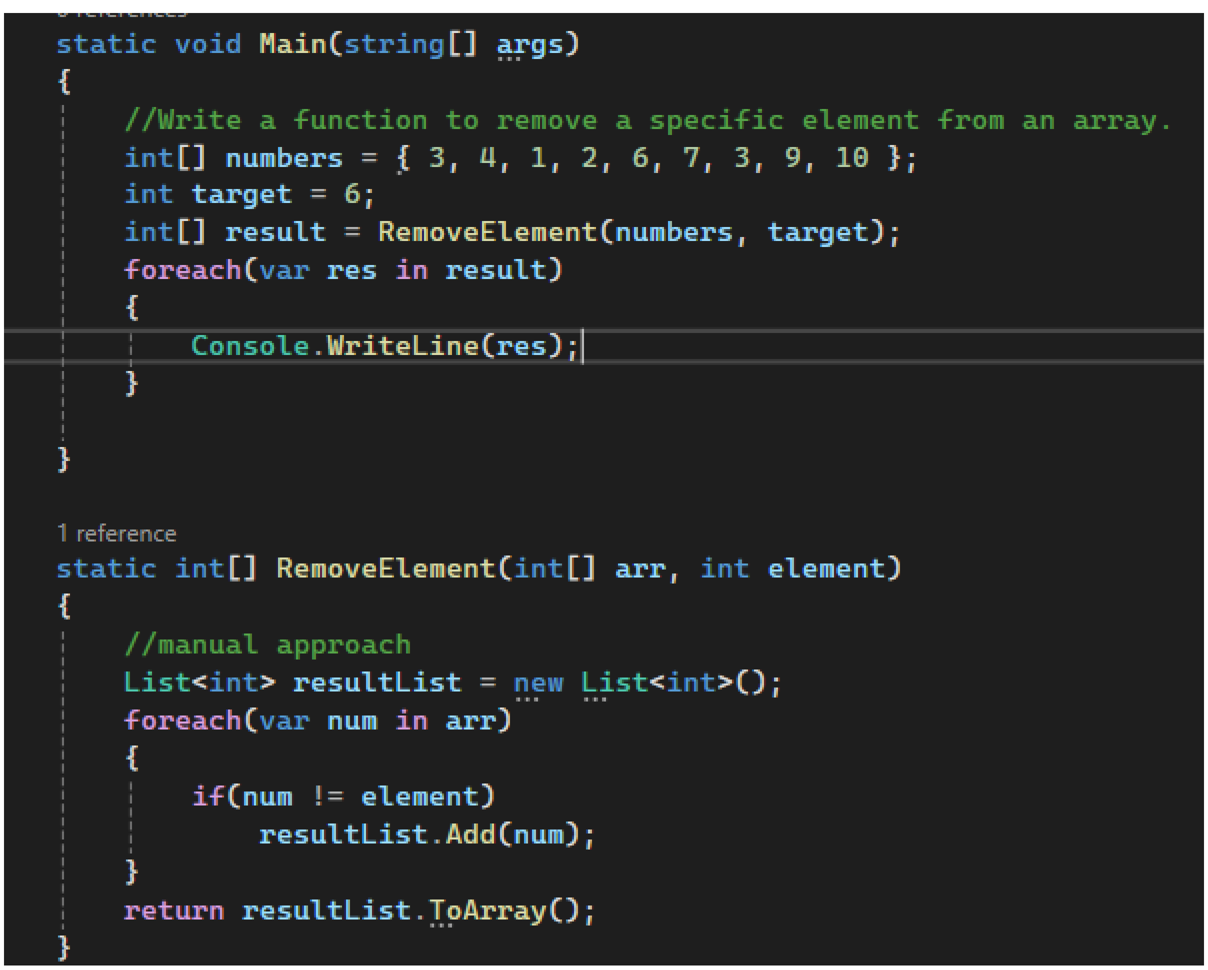
```
0 references
static void Main(string[] args)
{
    //Remove given element from an array
    int[] arr = { 1, 2, 3, 4, 5 };
    int elementToRemove = 4;
    int[] result = RemoveElement(arr, elementToRemove);
    foreach(var num in result)
    {
        Console.WriteLine(num);
    }
}

1 reference
static int[] RemoveElement(int[] arr, int element)
{
    List<int> list = new List<int>(arr);
    list.Remove(element);
    return list.ToArray();
}
```



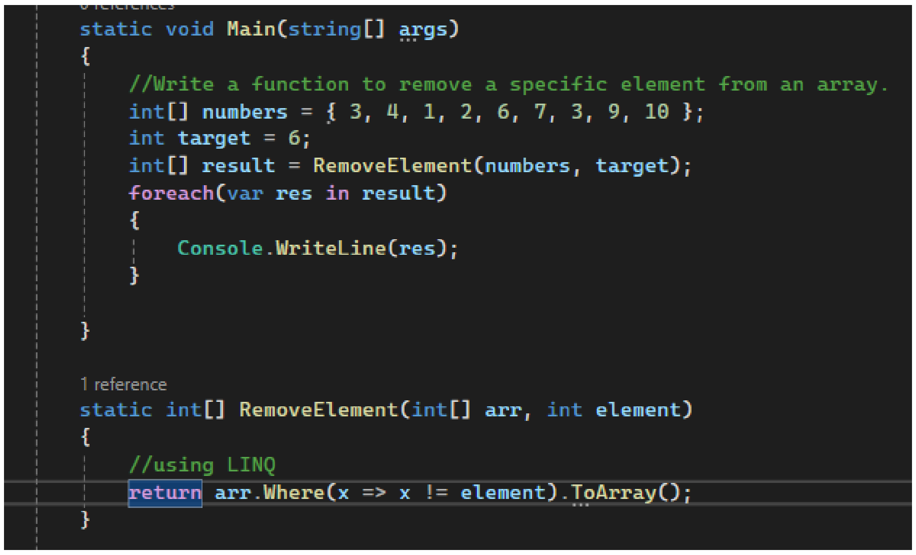
```
class Program
{
    0 references
    static void Main(string[] args)
    {
        int[] numbers = { 3, 4, 1, 2, 6, 7, 3, 9, 10 };
        int[] result = FindAboveFive(numbers);
        Console.WriteLine("Here is the result : ");
        foreach (var res in result)
        {
            Console.WriteLine(res);
        }
    }

    1 reference
    static int[] FindAboveFive(int[] arr)
    {
        //using LINQ
        return arr.Where(x => x > 5).ToArray();
    }

}
```



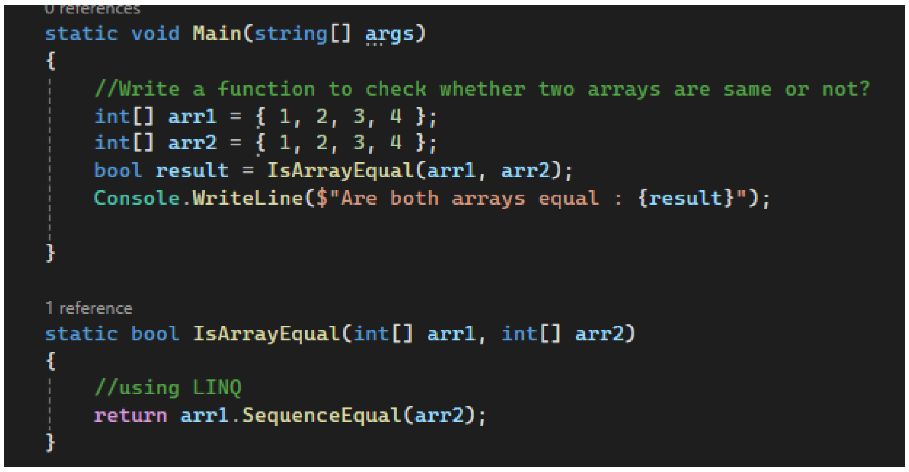
```
0 references
static void Main(string[] args)
{
    int[] numbers = { 3, 4, 1, 2, 6, 7, 3, 9, 10 };
    int[] result = DoubleArray(numbers);
    foreach(var res in result)
    {
        Console.WriteLine(res);
    }
}

1 reference
static int[] DoubleArray(int[] arr)
{
    //using LINQ
    return arr.Select(x => x * 2).ToArray();
}
```



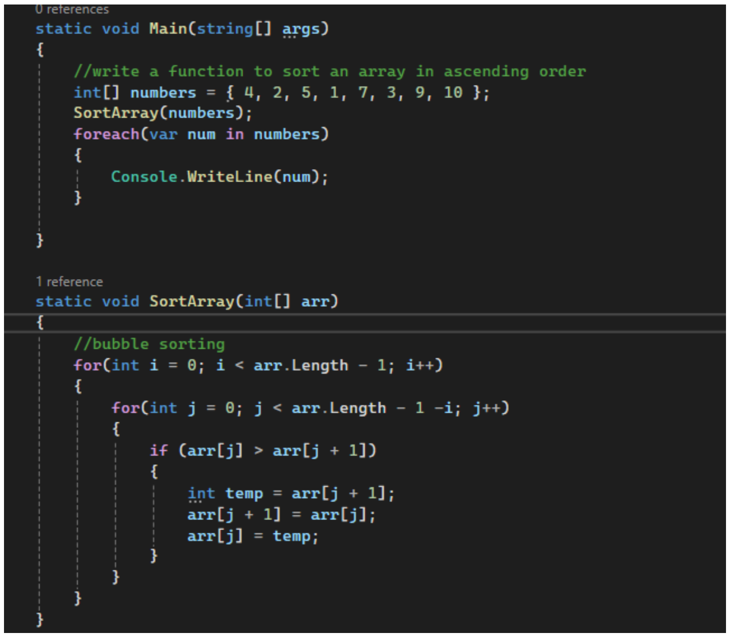
```
static void Main(string[] args)
{
    //Write a function to remove a specific element from an array.
    int[] numbers = { 3, 4, 1, 2, 6, 7, 3, 9, 10 };
    int target = 6;
    int[] result = RemoveElement(numbers, target);
    foreach(var res in result)
    {
        Console.WriteLine(res);
    }

}

1 reference
static int[] RemoveElement(int[] arr, int element)
{
    //manual approach
    List<int> resultList = new List<int>();
    foreach(var num in arr)
    {
        if(num != element)
            resultList.Add(num);
    }
    return resultList.ToArray();
}
```



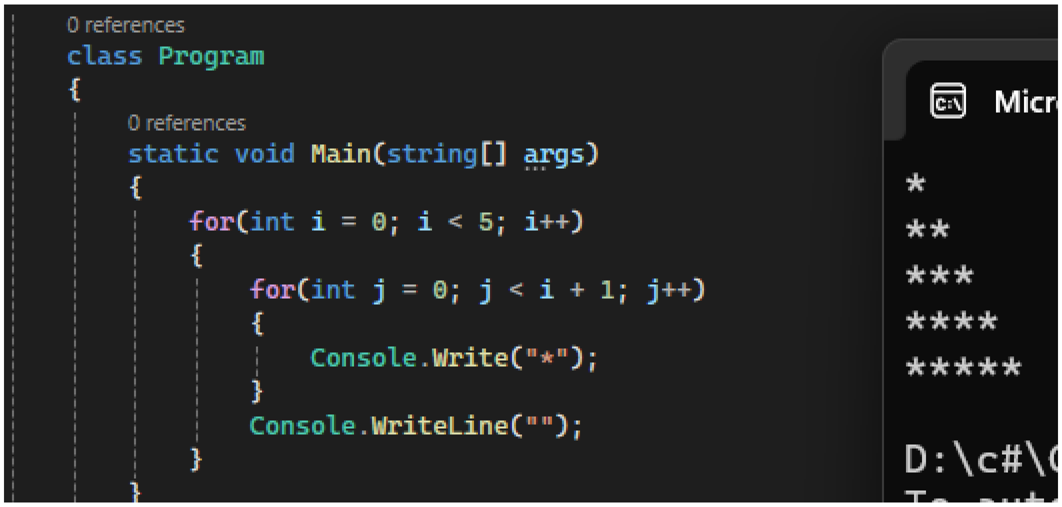
```
static void Main(string[] args)
{
    //Write a function to remove a specific element from an array.
    int[] numbers = { 3, 4, 1, 2, 6, 7, 3, 9, 10 };
    int target = 6;
    int[] result = RemoveElement(numbers, target);
    foreach(var res in result)
    {
        Console.WriteLine(res);
    }

}

1 reference
static int[] RemoveElement(int[] arr, int element)
{
    //using LINQ
    return arr.Where(x => x != element).ToArray();
}
```



```
0 references
static void Main(string[] args)
{
    //Write a function to check whether two arrays are same or not?
    int[] arr1 = { 1, 2, 3, 4 };
    int[] arr2 = { 1, 2, 3, 4 };
    bool result = IsArrayEqual(arr1, arr2);
    Console.WriteLine($"Are both arrays equal : {result}");

}

1 reference
static bool IsArrayEqual(int[] arr1, int[] arr2)
{
    //using LINQ
    return arr1.SequenceEqual(arr2);
}
```

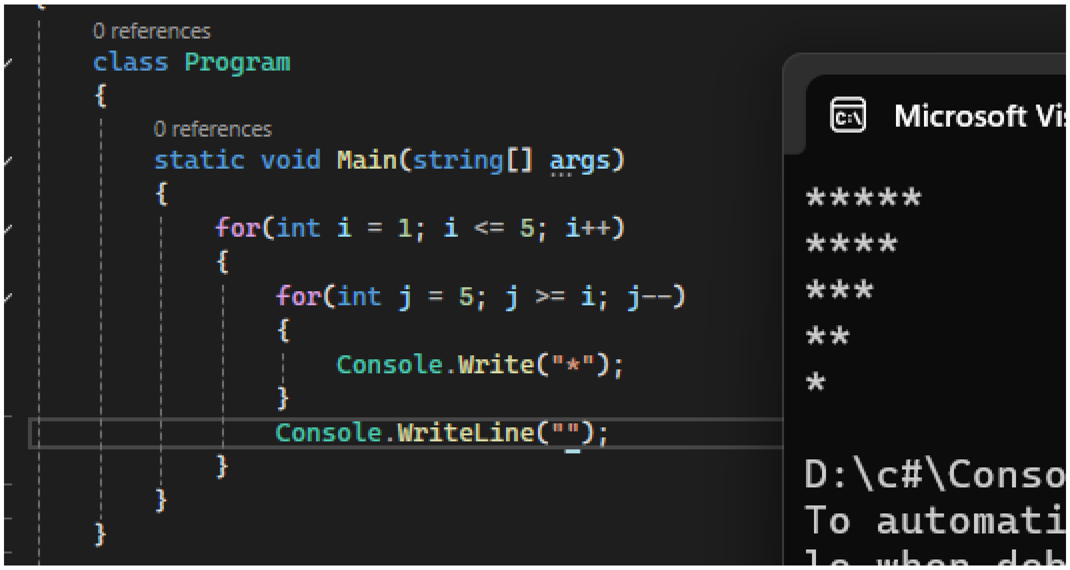


```
0 references
static void Main(string[] args)
{
    //write a function to sort an array in ascending order
    int[] numbers = { 4, 2, 5, 1, 7, 3, 9, 10 };
    SortArray(numbers);
    foreach(var num in numbers)
    {
        Console.WriteLine(num);
    }

}

1 reference
static void SortArray(int[] arr)
{
    //bubble sorting
    for(int i = 0; i < arr.Length - 1; i++)
    {
        for(int j = 0; j < arr.Length - 1 -i; j++)
        {
            if (arr[j] > arr[j + 1])
            {
                int temp = arr[j + 1];
                arr[j + 1] = arr[j];
                arr[j] = temp;
            }
        }
    }
}
```



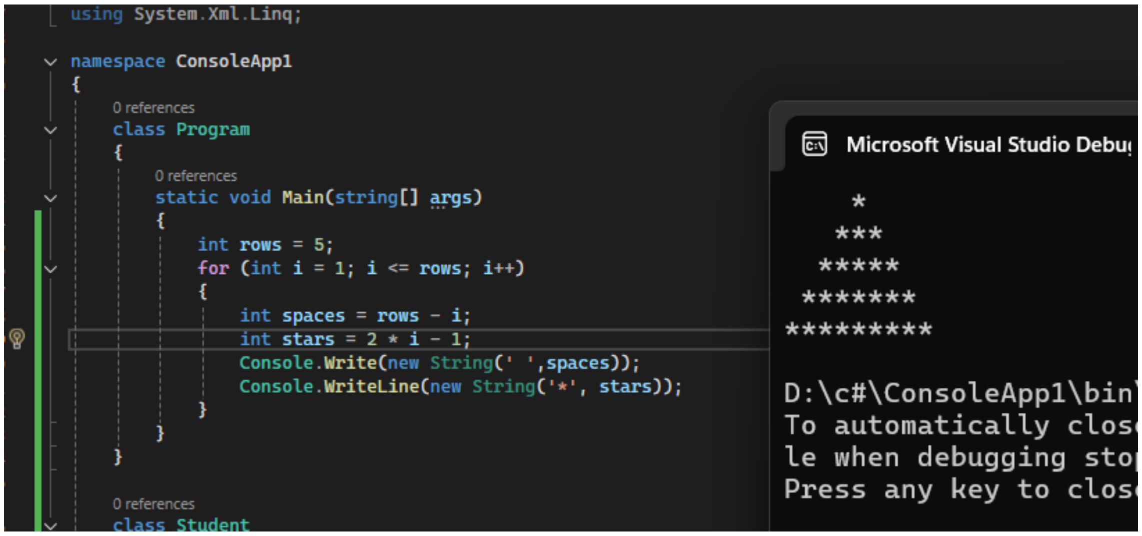
```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        for(int i = 0; i < 5; i++)
        {
            for(int j = 0; j < i + 1; j++)
            {
                Console.Write("*");
            }
            Console.WriteLine("");
        }
    }
```

```
Micr
*
**
***
****
*****

D:\c#\C
```



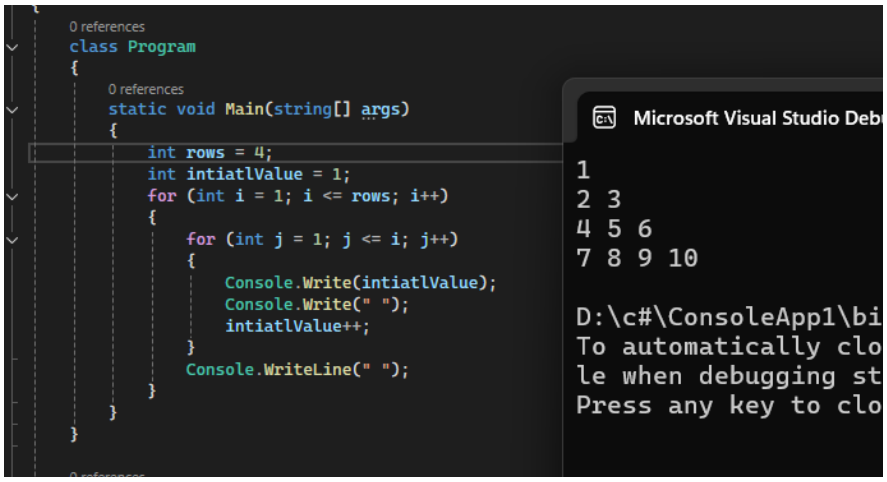
```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        for(int i = 1; i <= 5; i++)
        {
            for(int j = 5; j >= i; j--)
            {
                Console.Write("*");
            }
            Console.WriteLine("");
        }
    }
}
```

```
Microsoft Vi
*****
****
***
**
*

D:\c#\Conso
To automati
```



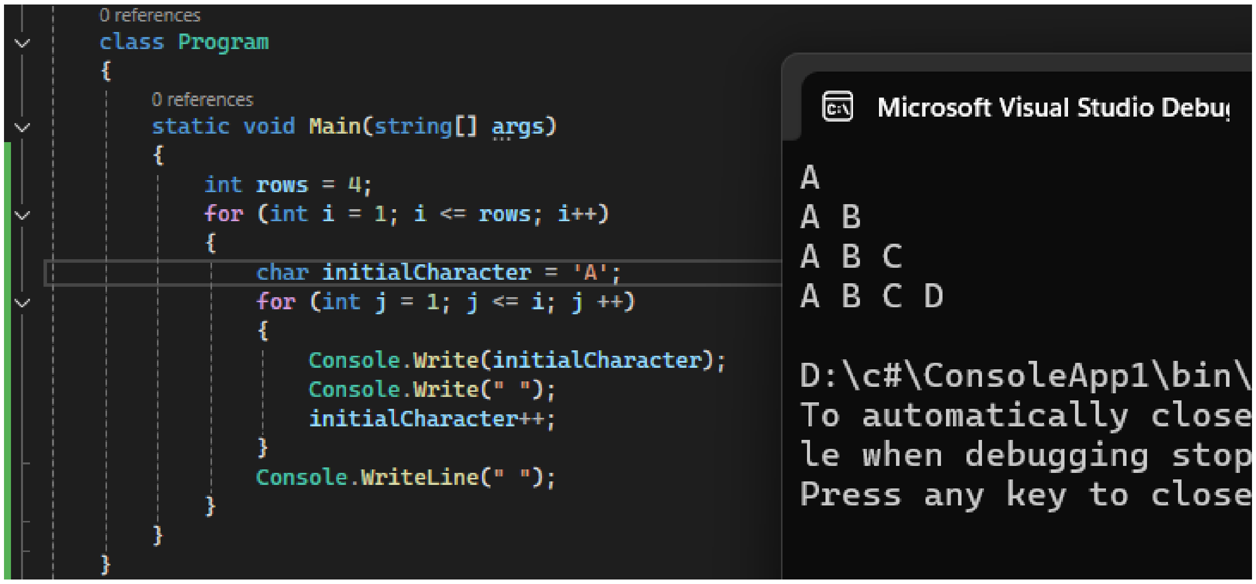
```
using System.Xml.Linq;

namespace ConsoleApp1
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            int rows = 5;
            for (int i = 1; i <= rows; i++)
            {
                int spaces = rows - i;
                int stars = 2 * i - 1;
                Console.Write(new String(' ',spaces));
                Console.WriteLine(new String('*', stars));
            }
        }
    }

    0 references
    class Student
```

```
Microsoft Visual Studio Debu
    *
   ***
  *****
 *******
*********

D:\c#\ConsoleApp1\bin\
To automatically clos
le when debugging sto
Press any key to clos
```



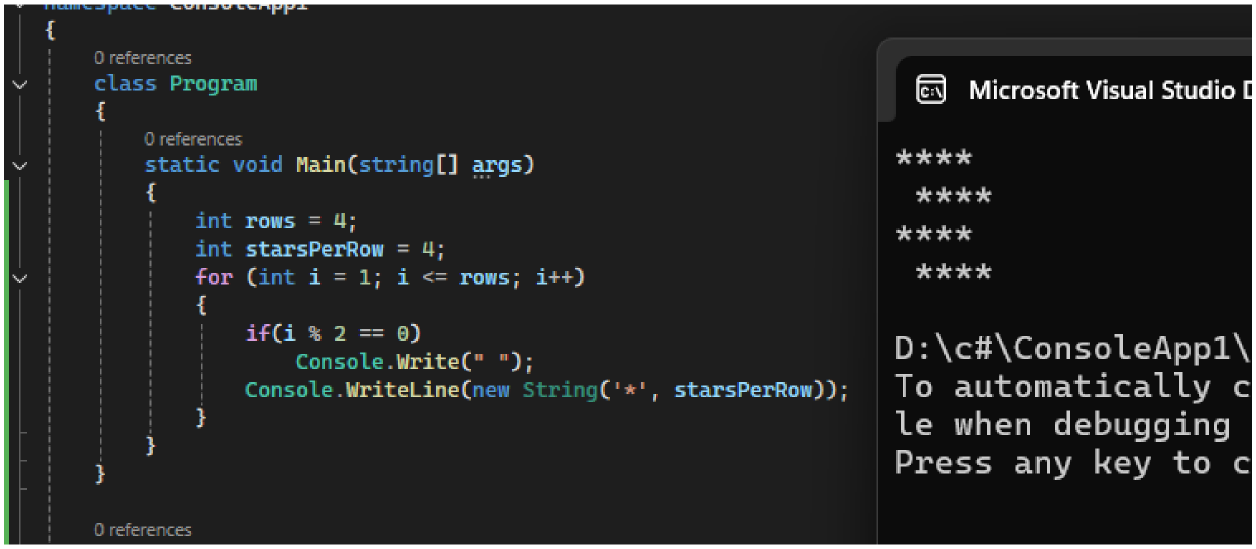
```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        int rows = 4;
        int intiatlValue = 1;
        for (int i = 1; i <= rows; i++)
        {
            for (int j = 1; j <= i; j++)
            {
                Console.Write(intiatlValue);
                Console.Write(" ");
                intiatlValue++;
            }
            Console.WriteLine(" ");
        }
    }
}
```

```
1
2 3
4 5 6
7 8 9 10

D:\c#\ConsoleApp1\bi
To automatically clo
le when debugging st
Press any key to clo
```



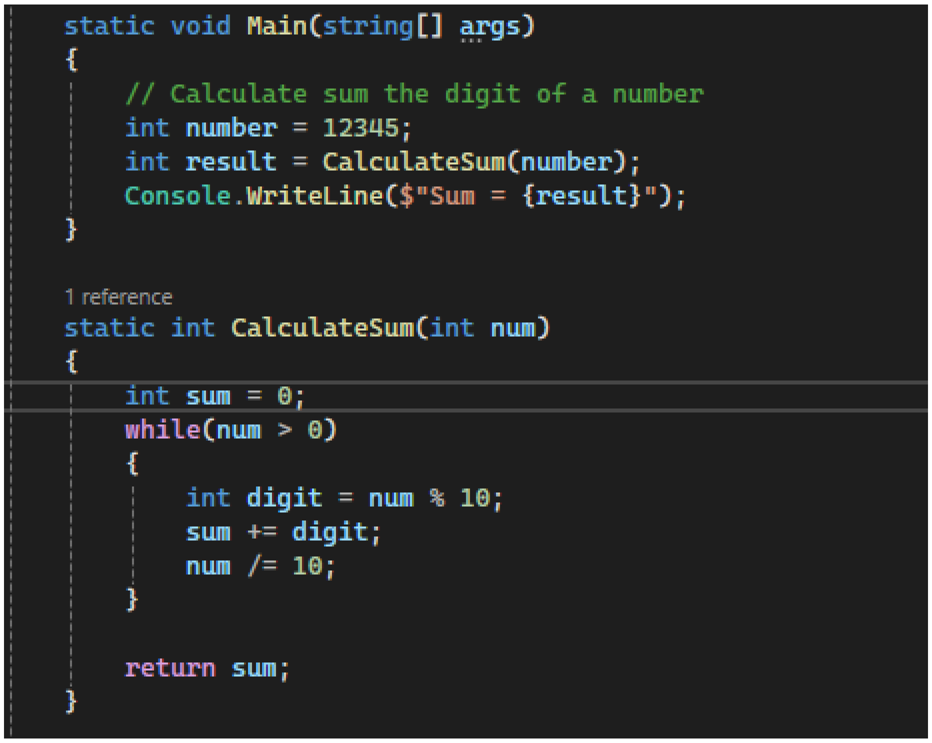
```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        int rows = 4;
        for (int i = 1; i <= rows; i++)
        {
            char initialCharacter = 'A';
            for (int j = 1; j <= i; j ++)
            {
                Console.Write(initialCharacter);
                Console.Write(" ");
                initialCharacter++;
            }
            Console.WriteLine(" ");
        }
    }
}
```

```
A
A B
A B C
A B C D

D:\c#\ConsoleApp1\bin\
To automatically close
le when debugging stop
Press any key to close
```

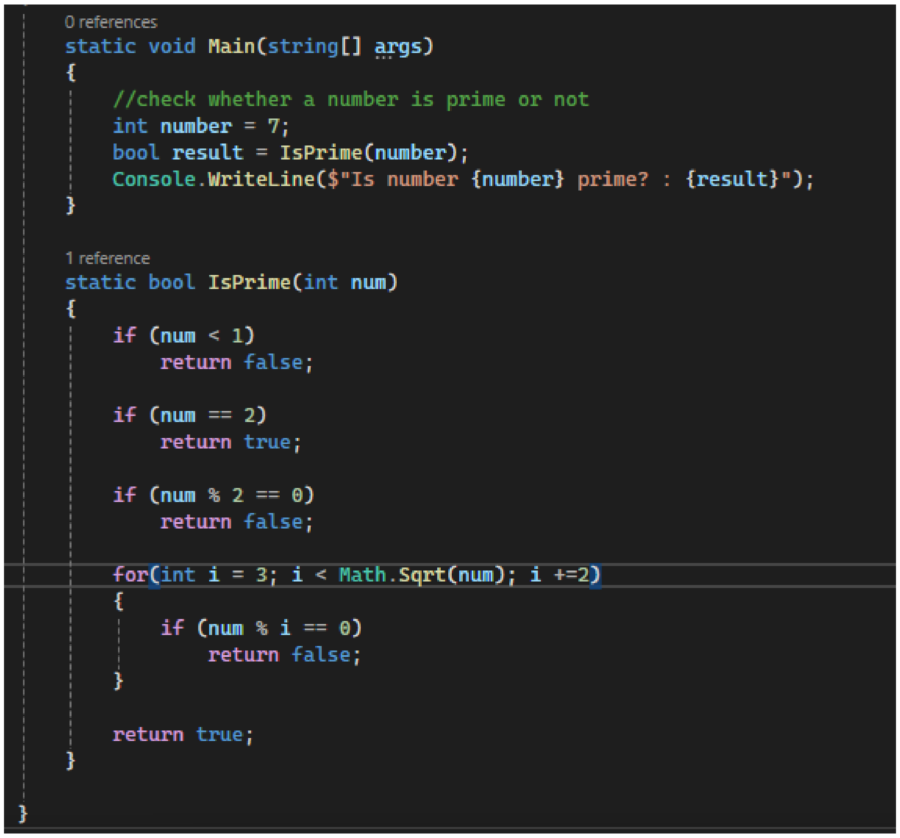


```
namespace ConsoleApp1
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            int rows = 4;
            int starsPerRow = 4;
            for (int i = 1; i <= rows; i++)
            {
                if(i % 2 == 0)
                    Console.Write(" ");
                Console.WriteLine(new String('*', starsPerRow));
            }
        }
    }

    0 references
```

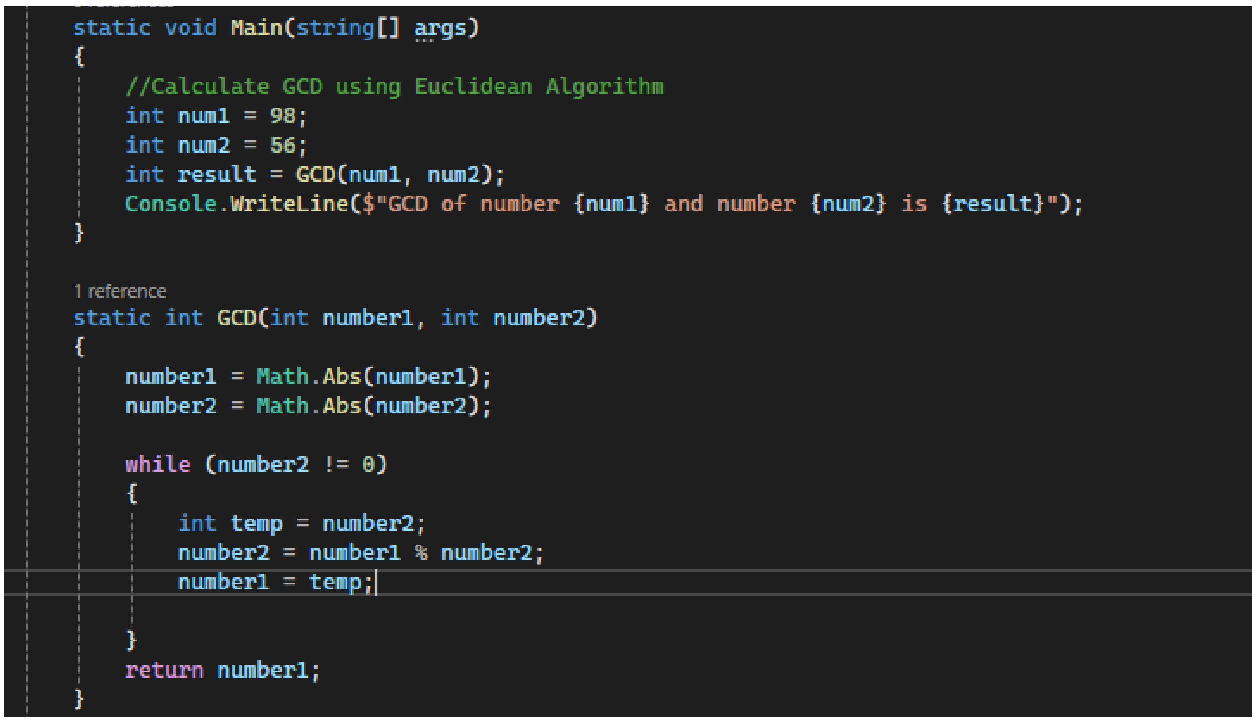
```
****
 ****
****
 ****

D:\c#\ConsoleApp1\
To automatically c
le when debugging
Press any key to c
```



```
static void Main(string[] args)
{
    // Calculate sum the digit of a number
    int number = 12345;
    int result = CalculateSum(number);
    Console.WriteLine($"Sum = {result}");
}

1 reference
static int CalculateSum(int num)
{
    int sum = 0;
    while(num > 0)
    {
        int digit = num % 10;
        sum += digit;
        num /= 10;
    }

    return sum;
}
```



```
0 references
static void Main(string[] args)
{
    //check whether a number is prime or not
    int number = 7;
    bool result = IsPrime(number);
    Console.WriteLine($"Is number {number} prime? : {result}");
}

1 reference
static bool IsPrime(int num)
{
    if (num < 1)
        return false;

    if (num == 2)
        return true;

    if (num % 2 == 0)
        return false;

    for(int i = 3; i < Math.Sqrt(num); i +=2)
    {
        if (num % i == 0)
            return false;
    }

    return true;
}

}
```



```
static void Main(string[] args)
{
    //Calculate GCD using Euclidean Algorithm
    int num1 = 98;
    int num2 = 56;
    int result = GCD(num1, num2);
    Console.WriteLine($"GCD of number {num1} and number {num2} is {result}");
}

1 reference
static int GCD(int number1, int number2)
{
    number1 = Math.Abs(number1);
    number2 = Math.Abs(number2);

    while (number2 != 0)
    {
        int temp = number2;
        number2 = number1 % number2;
        number1 = temp;

    }
    return number1;
}
```