Swinburne University of Technology

COS20007 Object Oriented Programming

Preparing for Object Oriented Programming

PDF generated at 10:56 on Monday 7th August, 2023

Name: Lorraine Becker
Student ID: 104584773

1.1P: Preparing for OOP – Answer Sheet

1. Explain the following terminal instructions:
    a. **cd:** This command is known as 'change directory'. It's used to allow a user to move between directories.
    b. **ls:** This command is used to list all files in a current directory, although excludes hidden files.
    c. **pwd:** This command stands for 'Print Working Directory'. It's used to output the absolute path of your present working directory.

2. Consider the following kinds of information, and suggest the most appropriate data type to store or represent each:

Information	**Suggested Data Type**
A person's name	String
A person's age in years	Integer
A phone number	String
A temperature in Celsius	Float
The average age of a group of people	Float
Whether a person has eaten lunch	Boolean

3. Aside from the examples already provided in question 2, come up with an example of information that could be stored as:

Data type	**Suggested Information**
String	A person's favourite fruit
Integer	The amount of units a student studies
Float	A person's height
Boolean	Whether a person is attending a party

4. Fill out the last two columns of the following table, evaluating the value of each expression and identifying the data type the value is most likely to be:

Expression	**Given**	**Value**	**Data Type**
`6`		6	Integer
`True`		TRUE	Boolean
`a`	`a = 2.5`	2.5	Float
`1 + 2 * 3`		7	Integer
`a and False`	`a = True`	FALSE	Boolean
`a or False`	`a = True`	TRUE	Boolean

`a + b`	`a = 1`
	`b = 2`
		3	Integer

`2 * a`	`a = 3`	6	Integer

`a * 2 + b`	`a = 2.5`
	`b = 2`
		7.0	Float (if start with float, end with float)

`a + 2 * b`	`a = 2.5`
	`b = 2`
		6.5	Float

`(a + b) * c`	`a = 1`
	`b = 1`
	`c = 5`
		10	Integer

`"Fred" + " Smith"`		Fred Smith	String
`a + " Smith"`	`a = "Wilma"`	Wilma Smith	String

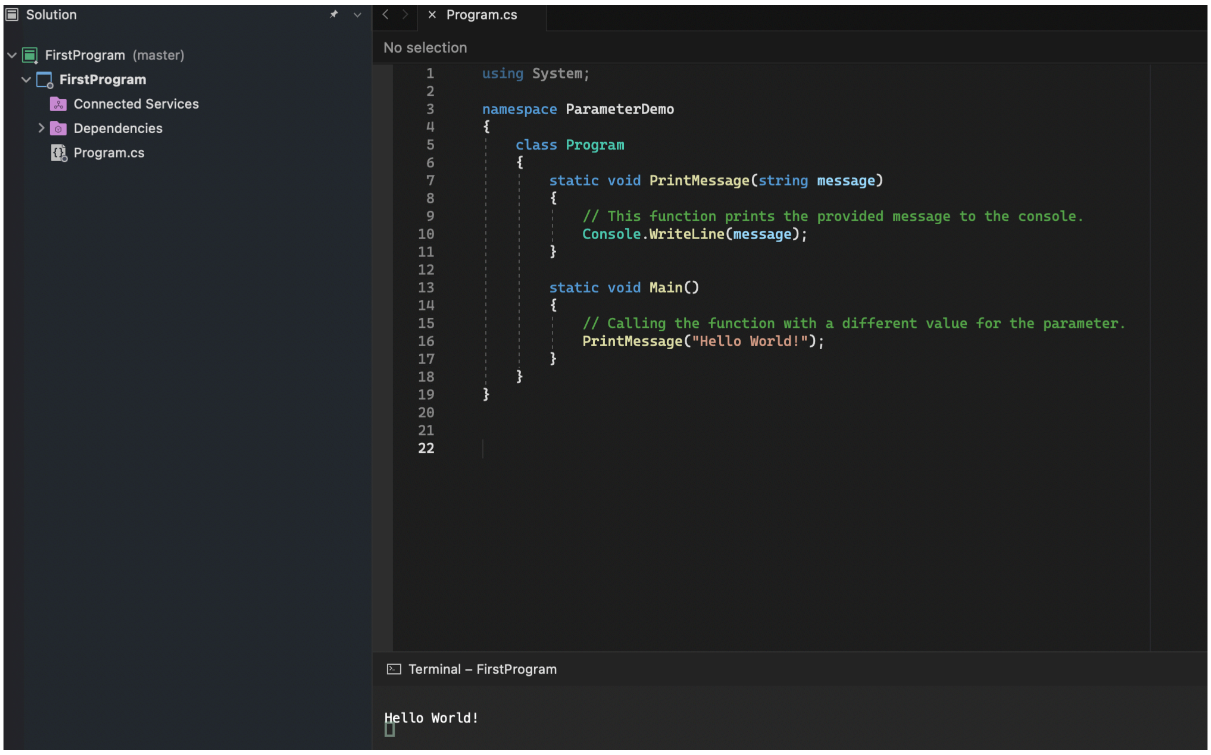
5. Using an example, explain the difference between **declaring** and **initialising** a variable.

The difference between the two is that declaring a variable is giving it a name and a type. **For eg.**
String name;
int value1;

Whereas initialising a variable is when you give it a value. **For eg.**
int favouriteNumber = 2 ;

6. Explain the term **parameter**. Write some code that demonstrates a simple of use of a parameter. You should show a procedure or function that uses a parameter, and how you would call that procedure or function.

A **parameter** is a named value provided as input to a function.



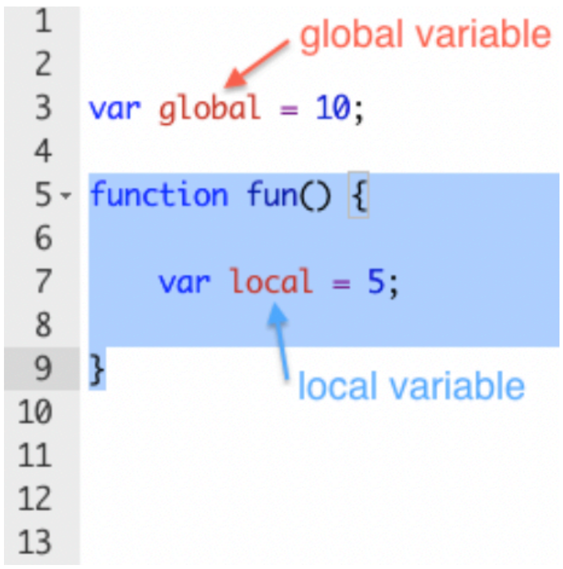
```
using System;

namespace ParameterDemo
{
    class Program
    {
        static void PrintMessage(string message)
        {
            // This function prints the provided message to the console.
            Console.WriteLine(message);
        }

        static void Main()
        {
            // Calling the function with a different value for the parameter.
            PrintMessage("Hello World!");
        }
    }
}
```

```
Hello World!
```

7. Using an example, describe the term **scope** as it is used in procedural programming (not in business or project management). Make sure you explain the different kinds of scope.

Definition: Scope is the visibility and accessibility of variables within different parts of a program.

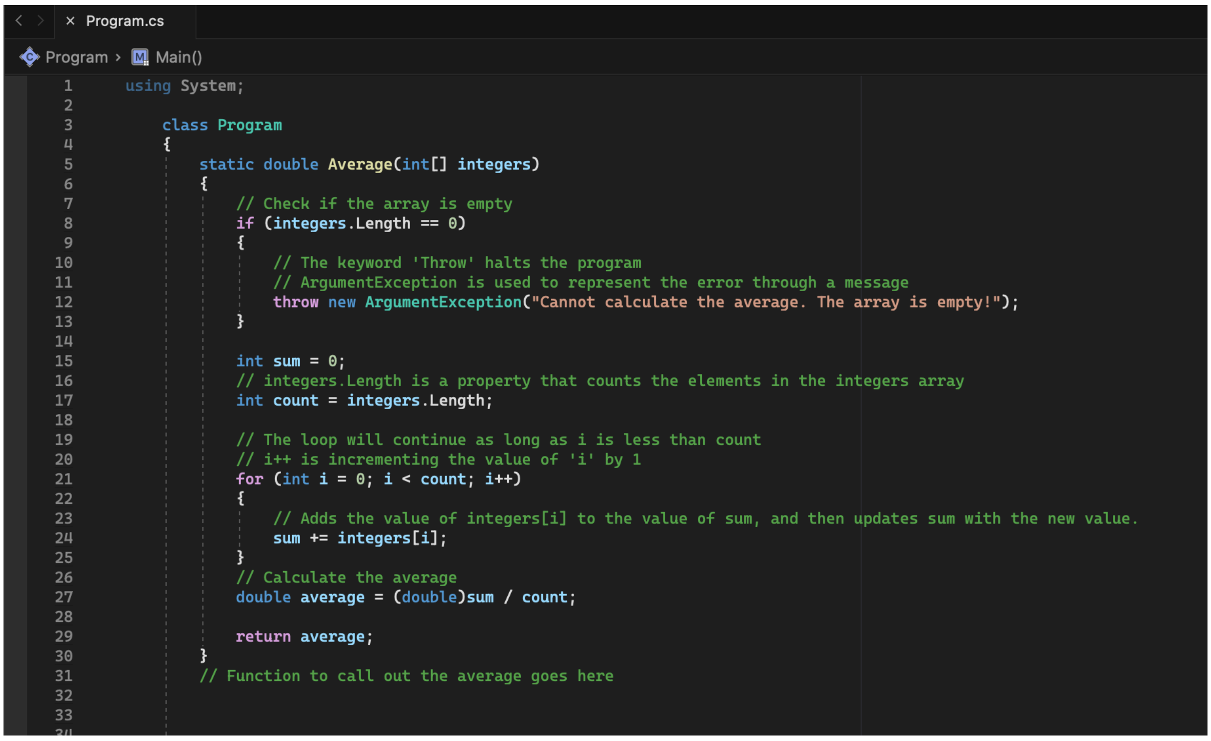
Two different kinds of scope in procedural programming are **Global Scope** and **Local Scope**. Some features of **Global Scope** include: declared variables are accessible throughout the entire program, they are declared outside of any function or block and they extend from the start of the program until its termination.
Some features of **Local Scope** include: declared variables are accessible only within a specific function or block, they are



```

var global = 10;

function fun() {

    var local = 5;

}
```

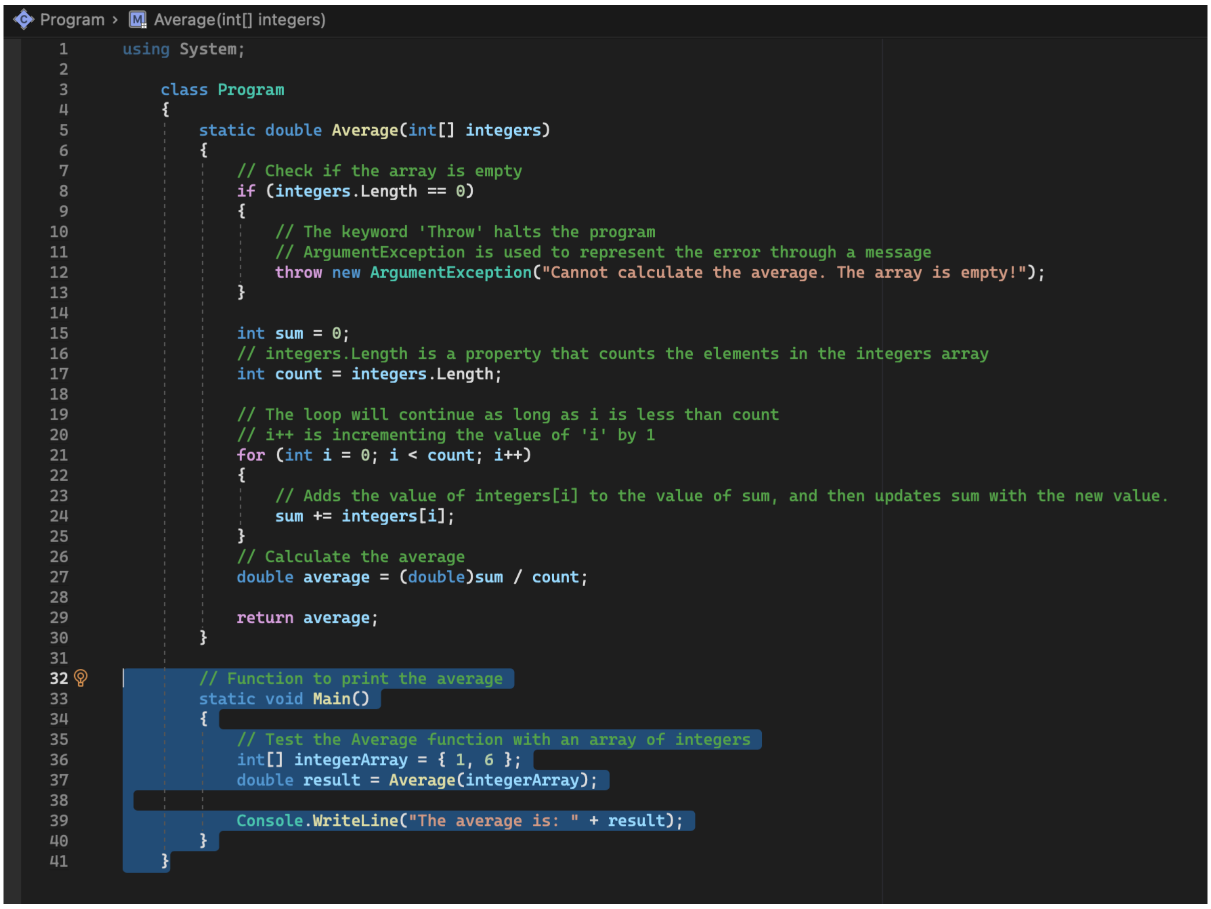
created when the function or block is executed and are destroyed when the function or block exits.

8. In a procedural style, in any language you like, write a function called Average, which accepts an array of integers and returns the average of those integers. Do not use any libraries for calculating the average. You must demonstrate appropriate use of parameters, returning and assigning values, and use of a loop. Note — just write the function at this point, we'll *use* it in the next task. You shouldn't have a complete program or even code that outputs anything yet at the end of this question.



```
using System;

    class Program
    {
        static double Average(int[] integers)
        {
            // Check if the array is empty
            if (integers.Length == 0)
            {
                // The keyword 'Throw' halts the program
                // ArgumentException is used to represent the error through a message
                throw new ArgumentException("Cannot calculate the average. The array is empty!");
            }

            int sum = 0;
            // integers.Length is a property that counts the elements in the integers array
            int count = integers.Length;

            // The loop will continue as long as i is less than count
            // i++ is incrementing the value of 'i' by 1
            for (int i = 0; i < count; i++)
            {
                // Adds the value of integers[i] to the value of sum, and then updates sum with the new value.
                sum += integers[i];
            }
            // Calculate the average
            double average = (double)sum / count;

            return average;
        }
        // Function to call out the average goes here
```

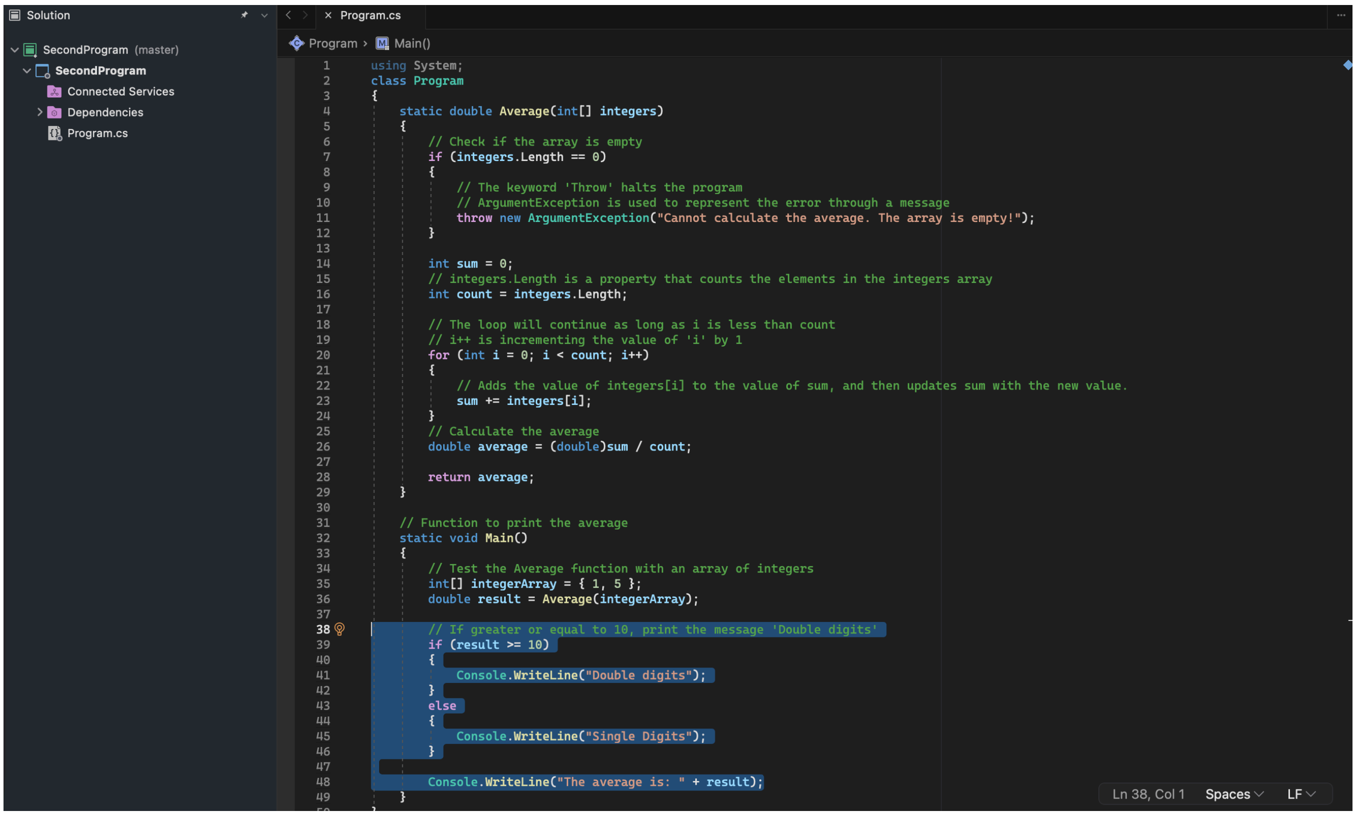
9. In the same language, write the code you would need to call that function and print out the result.



Program › Average(int[] integers)

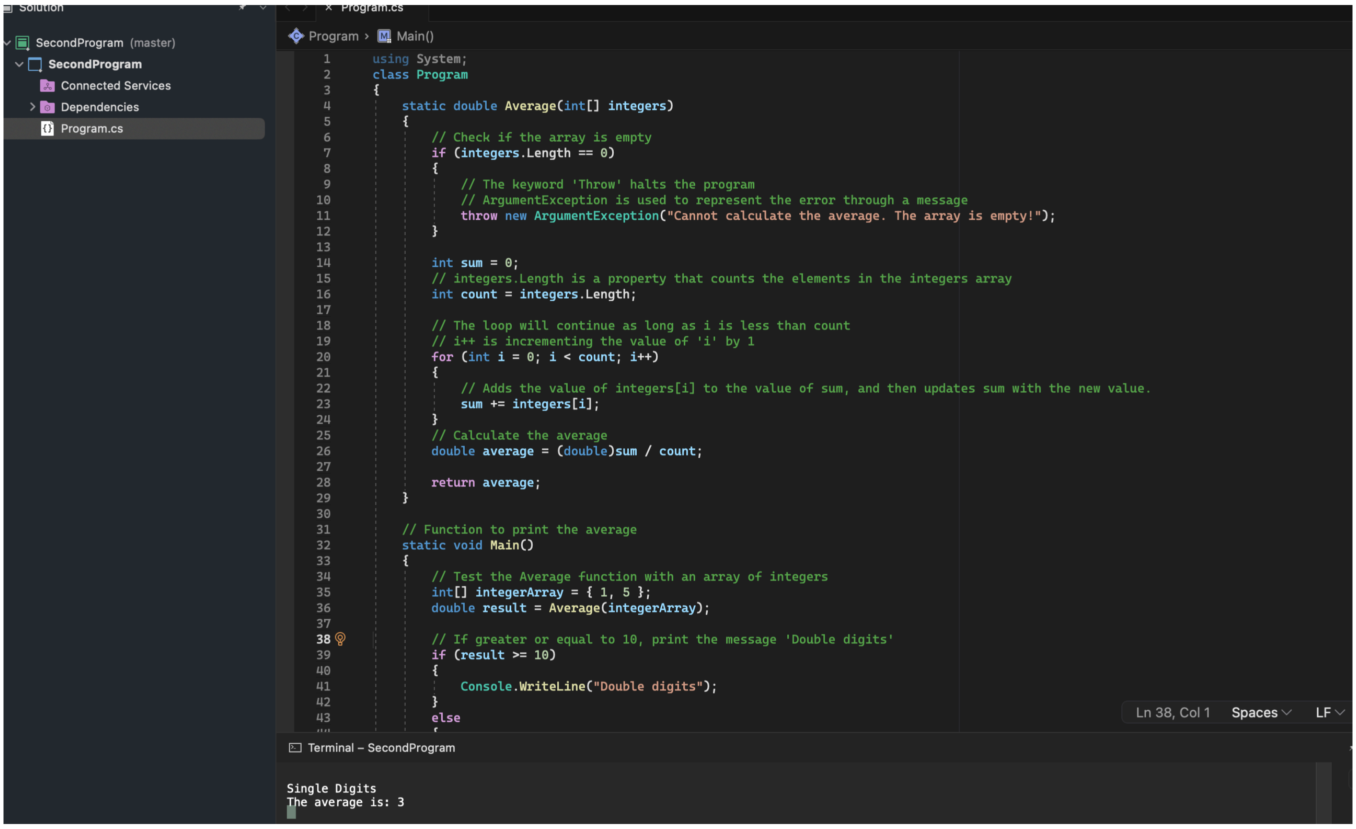
```csharp
using System;

    class Program
    {
        static double Average(int[] integers)
        {
            // Check if the array is empty
            if (integers.Length == 0)
            {
                // The keyword 'Throw' halts the program
                // ArgumentException is used to represent the error through a message
                throw new ArgumentException("Cannot calculate the average. The array is empty!");
            }

            int sum = 0;
            // integers.Length is a property that counts the elements in the integers array
            int count = integers.Length;

            // The loop will continue as long as i is less than count
            // i++ is incrementing the value of 'i' by 1
            for (int i = 0; i < count; i++)
            {
                // Adds the value of integers[i] to the value of sum, and then updates sum with the new value.
                sum += integers[i];
            }
            // Calculate the average
            double average = (double)sum / count;

            return average;
        }

        // Function to print the average
        static void Main()
        {
            // Test the Average function with an array of integers
            int[] integerArray = { 1, 6 };
            double result = Average(integerArray);

            Console.WriteLine("The average is: " + result);
        }
    }
```

10. To the code from 9, add code to print the message “Double digits” if the average is above or equal to 10. Otherwise, print the message “Single digits”. Provide a screenshot of your program running.



```
using System;
class Program
{
    static double Average(int[] integers)
    {
        // Check if the array is empty
        if (integers.Length == 0)
        {
            // The keyword 'Throw' halts the program
            // ArgumentException is used to represent the error through a message
            throw new ArgumentException("Cannot calculate the average. The array is empty!");
        }

        int sum = 0;
        // integers.Length is a property that counts the elements in the integers array
        int count = integers.Length;

        // The loop will continue as long as i is less than count
        // i++ is incrementing the value of 'i' by 1
        for (int i = 0; i < count; i++)
        {
            // Adds the value of integers[i] to the value of sum, and then updates sum with the new value.
            sum += integers[i];
        }
        // Calculate the average
        double average = (double)sum / count;

        return average;
    }

    // Function to print the average
    static void Main()
    {
        // Test the Average function with an array of integers
        int[] integerArray = { 1, 5 };
        double result = Average(integerArray);

        // If greater or equal to 10, print the message 'Double digits'
        if (result >= 10)
        {
            Console.WriteLine("Double digits");
        }
        else
        {
            Console.WriteLine("Single Digits");
        }

        Console.WriteLine("The average is: " + result);
    }
```

11. Example output in terminal



```
using System;
class Program
{
    static double Average(int[] integers)
    {
        // Check if the array is empty
        if (integers.Length == 0)
        {
            // The keyword 'Throw' halts the program
            // ArgumentException is used to represent the error through a message
            throw new ArgumentException("Cannot calculate the average. The array is empty!");
        }

        int sum = 0;
        // integers.Length is a property that counts the elements in the integers array
        int count = integers.Length;

        // The loop will continue as long as i is less than count
        // i++ is incrementing the value of 'i' by 1
        for (int i = 0; i < count; i++)
        {
            // Adds the value of integers[i] to the value of sum, and then updates sum with the new value.
            sum += integers[i];
        }
        // Calculate the average
        double average = (double)sum / count;

        return average;
    }

    // Function to print the average
    static void Main()
    {
        // Test the Average function with an array of integers
        int[] integerArray = { 1, 5 };
        double result = Average(integerArray);

        // If greater or equal to 10, print the message 'Double digits'
        if (result >= 10)
        {
            Console.WriteLine("Double digits");
        }
        else
```

```
Single Digits
The average is: 3
```