# Method/Function/Procedure/Sub/Routine – Chapter 7

Method – this is the fourth control structure we have worked with in this course. Each control structure brings some advantage to the programmer.

* **Sequence** sets the framework to ensure that all the statements are processes exactly once and in order. i.e. no skips or repeats
* **Branching** allows the program to perform more than one task
* **Repetition** provides a structure to process a block or code multiple times
* **Method** allows the programmer to assign a name to a block or code

The coder has the luxury to reuse blocks of code, hence the job can be done with less code. In addition, it facilitates the decomposition of a complex problem into simpler sub-tasks that can be implemented and tested separately. This is possible by attaching a name to a block of code statements and then invoking the code when required.

All method consists of a header and a body. The header includes an accessibility modifier, the static keyword, the return type, the name of the method and a parameter list enclosed within a pair of parenthesis as shown in the diagram below. All the parts enclosed within square brackets are optional. However, for this part of COMP100, the static modifier is MANDATORY.

**Template of a Method**

[modifier] [static] «return\_type» «method\_name»([parameter\_type-name\_list])  
{  
 statement 1;  
 statement 2;  
}

**Definition of a Method**

/\*  
 \*This method does not require any parameter  
 \*This method does not return a value  
 \*/  
static void DisplayPersonInfo()  
{  
 Console.WriteLine("Narendra");  
 Console.WriteLine("Centennial College");  
}

**Invocation of a Method**

DisplayPersonInfo();

// This method takes a string argument  
Console.WriteLine("Programming is the best");

// This method returns a string value   
string name = Console.ReadLine();

// The ToInt32 method takes a string argument and returns an int value   
int age = Convert.ToInt32(Console.ReadLine());

Methods lays the foundations for building designing and using classes. Classes is integral to Object-Oriented Programming (OOP) and is a simpler approach to implementing large or complex systems. Although newer programming paradigms are better suited to concurrent programming or working with big data or incorporating Artificial Intelligence in an application, OOP is a natural way or solving for most of us humans.

Completing these exercises will be an excellent preparation for your last test. For this, you will create a single solution and add a new project for each part i.e. your final revision will consist of a single solution with a separate project for each parts. The exercises are grouped into parts based on the arguments (or lack of) and the return value. The problems are graded by difficulty level so the later problems are more challenging than the earlier ones.

**Sample solution available on the public drive**

# Part I ‒ Methods with an empty parameter list and do not return a value: [Questions 1-8 required]

1. Write a method called **DisplayPersonalInfo()**. This method will display your name, school, program and your favorite course. Call the **DisplayPersonalInfo()** method from your program **Main()** method

You invoke a method by its name followed by a pair of brackets and the usual semi-colon

1. Write a method called **CalculateTuition()**. This method will prompt the user for the number of courses that she is currently taking and then calculate and display the tuition cost. (cost = number of course \* 569.99). Call the **CalculateTuition()** method two times from the same **Main()** method as in question 1.
2. Write a method call **CalculateAreaOfCircle()**. This method will prompt the user for the radius of a circle and then calculate and display the area.[A = πr2].   
   Call the **CalculateAreaOfCircle()** method twice from the same **Main()** method as in question 1. Use **Math.Pi** for the value of **π**
3. Write a method call **CalculateAreaOfTriangle()**, that prompts the user for the base and height of a triangle and then calculate and display the area.[ ]   
   Call the **CalculateAreaOfTriangle()** method twice from the same **Main()** method as in question 1.
4. Write a method call **CalculateSaleCommission()**, that prompts the user for his sales figure, and then calculate and display his commission. (commission = 25% of sales in excess of 1000. If sales is equal to or below $1000.00 there is no commission)   
   Call the **CalculateSaleCommission()** method three times from the same **Main()** method as in question 1.
5. Write a method call **DisplaySineTable()**, that prompts the user for a starting value and an step size. The method will calculate and display a table (10 rows) of sine values based on the user input. Use the built-in method **Math.Sin()** to obtain the sine of an angle. e.g. if the starting value is 0.5 and the step size is 0.03 the method will display the following table:  
   0.50 0.4794  
   0.53 0.5055  
   0.56 0.5311  
     
   0.77 0.6961  
   The numbers in both columns MUST be right aligned. Call the **DisplaySineTable()** method from the same **Main()** method as in question 1.
6. In a write a method called **DisplayMenu()** to display the following text on screen:

===========Narendra’s Computer Systems================  
| 1. Display Personal Information |  
| 2. Calculate Tuition |  
| 3. Calculate Area Of A Circle |  
| 4. Calculate The Area Of A Triangle |  
| 5. Calculate Sales Commission |  
| 6. Display Sine Table |  
| 0. End program |  
| |  
======================================================  
 Enter the number of your choice ->

You may replace the instructor’s name with your name.

1. Add another method called **ShowMenu()** to your project. This method will call the method in question 7 continuously until the user presses 0. (You will have to invoke the method in a loop body, read in the user input as well as check the input). You will need to hook up all the methods that you wrote previously i.e. questions 1 to 6. Replace all the code from your **Main()** method with a single call to the **ShowMenu()** method.

# Part II ‒ Methods that do not return a value but takes an argument or possibly multiple arguments: [Questions 1-8 required]

The solution provided has a menu. You do not need to create one.

1. Write a method called DisplayHorizontalStars(int numberOfStars). This method will output the number of stars horizontally specified by its argument. Call the DisplayHorizontalStars() method three times from your program Main() method, supplying 0, 5 and 10 respectively for the number of stars.

When you invoke these kinds of methods, the value of the parameter is placed within the pair of brackets and the types are omitted.

If there are multiple parameters, than they are separated by commas.

1. Write a method similar to the one above that displays a vertical line of stars.   
   Call this method three times with arguments 0, 5 and 10 respectively.
2. Write a method that accepts an argument of type double. The method will calculate and display the volume of a sphere with radius specified by its argument. [].   
   Call this method from your main three times, with arguments 1, 2, and 10 respectively. The answers are 4.1888, 33.5103 and 4188.7902 respectively.  
   Use **Math.PI** for the value of
3. Write a method that takes two arguments: a cost price and a two letter province code. It will calculate and display the selling price. (If province is Ontario a tax of 13% is added to the price, if the province is Quebec a tax of 17% is added to the cost price. There is no tax for the rest of the provinces and territories).   
   In your main, call this method enough times to fully test it
4. Write a method that takes a single argument of type double. The will display a Celsius to Fahrenheit conversion table. The starting temperature is indicated by the argument and it will display 10 lines in increments of 1. [Fahrenheit = 9/5 Celsius + 32].   
   In your main call this method two times, each time with a different value.
5. Write a method that takes the following arguments: a starting km value of type double, an increment size of type double and the number of lines of type int. Your program will create and display a kilometer to miles conversion table. [miles = km \* 0.621371].   
   In your main call this method three times, each time with different values.
6. Modify the DisplaySineTable() method in the previous section to accept the start value, the step size and number of rows as argument to the method.
7. Write a method that converts a person’s height from centimeter to meters and centimeters and display the result on the console.   
   In your main method, you should call this method three times with argument 90, 120 and 275. The result is shown in the table below. You do not have to create the table below!

|  |  |
| --- | --- |
| Input | Result |
| 90cm | 0m 90cm |
| 120cm | 1m 20 cm |
| 275cm | 2m 75cm |

# Part III ‒ Methods that return a value and may or may not arguments: [Questions 3-11 required]

The solution provided has a menu. You do not need to create one.

1. Write a method called CaculateDifference(int firstNumber, int secondNumber). This method will calculate and return the difference between the two numbers i.e. it will subtract the smaller one from the larger one and then return that value.   
   Call the CaculateDifference() method four times from your program Main() method supplying different arguments each time. You must display the returned value for each call.

To use the result of a method call, you must assign it to a suitable variable.

If you simply want to see the value, then use Console.Write().the method call

In this part all the output are done in the Main() method i.e. none of the methods should have any Console.Write or Console.WriteLine unless you need to prompt for input

1. Write a method called CalculatePower(double current, double resistance). The method will calculate and return the electrical power dissipated in a circuit. [P=I2R].  
   Call the CalculatePower() method two times from your program Main() method supplying different arguments each time and displaying the returned value each time.
2. Write a method called CaculateTuitionFee(int numberOfCourses, double costPerCourse = 449.99). This method will calculate and return the required fees amount. [Fees = number of courses \* cost per course + 15.25].   
   Call the CaculateTuitionFee() method four times from your program Main() method supplying different arguments each time.
3. Write a method called CalculateCommission(double saleAmount). This method will calculate and return a sales representation’s commission. [over $1000 the commission will be 1%].   
   Call the CalculateCommission() method three times from your program Main() method, supplying arguments 900, 1000 and 2000.
4. Write a method that does not take any argument. The method will prompt the user for the number of burgers that he wants to buy. The method will calculate and return the amount of money that will be required. [One burger cost $1.39].   
   From your program Main() method, call the above method two times
5. *“Newton's law of universal gravitation states that every point mass in the universe attracts every other point mass with a force that is directly proportional to the product of their masses and inversely proportional to the square of the distance between them.”* Write a method called CalculateGravitationalAttraction() that takes three double arguments (masses of the two bodies and the distance between them). The method will calculate and return the force of attraction between them. [ , where G = 6.673x10-11].   
   In your main call this method with the masses of the earth, moon and the distance between them and display the resulting force. Mass of Earth = 5.972 × 1024, moon = 7.348 × 1022, Distance = 384,400000m. [Answer 1.99 x 1020N]

Use the e symbol to specify very large or very small values:

double G = 6.673e-11;

1. Heron’s formula allows you to calculate the area of any triangle given the length of the three sides. Write a method that takes the length of the three sides and return the area of the specified triangle. where .   
   In your main call this method with arguments 5.83, 4.24 and 8.00, and display the area. [Answer = 12.0].
2. Write a method called ConvertCelsiusToFahrenheit that takes a double argument. The method will calculate and return the Fahrenheit equivalent of the argument. [F = C \* 9 / 5 + 32].   
   In your main call this method three times with arguments 0, 50 and 100 respectively and display the results.
3. Write a method called ConvertFahrenheitToCelsius that takes a double argument. The method will calculate and return the Celsius equivalent of the argument. [C = (F – 32) \* 5 / 9].   
   In your main call this method three times with arguments 0, 32 and 212 respectively and display the results.
4. Write a method called ConvertCelsiusToKelvin that takes a double argument. The method will calculate and return the Kelvin equivalent of the argument. [K = C + 273.15].   
   In your main call this method three times with arguments -196, 0 and 100 respectively and display the results.
5. Write a method call ConvertFahrenheitToKelvin that takes a double argument. The method will calculate and return the Kelvin equivalent of the argument. [DO NOT RE-CODE THIS METHOD. Use the two previous methods to assist in your computation].   
   In your main call this method three times and display the results.

Supplying the output of one method call into the input of another method is referred to as method chaining. It is an elegant style of programming.

1. Write two methods: Square(int) and Cube(int), these methods return the square and the cube of its argument. In your main method prompt the user for an integer, then pass this integer to the Square method and then pass the results to the Cube method. In the main method print out the final results.
2. Write two methods that will be called from main. The first method should prompt the user for his daily sales amount. The method should accept and return this amount. The second method should calculate and return his commission based on the following: 0-999 3%, 1000-2999 4% and over 3000 5%. In your main call the two methods and display the results in a readable manner.

# Part IV ‒ Advanced Method usage: Passing argument using the ref, out and param modifiers: [Questions 2-6 required]

The solution provided has a menu. You do not need to create one.

1. Write a method call DoubleIt(ref int x) that takes a single argument and does not return a value. This method will double the value of its argument.   
   In your main, call this method twice and print the value of the parameter after the method call.

The ref and out modifier allows the method to mutate the value of a parameter.

You will have to add the modifier when sending the argument in your method invocation and in the method header of the method definition

1. Write a method call CubeIt(int x, ref int cube) that takes two arguments and does not return a value. The method will cube the first argument and assign it to the second argument.   
   In your main, call this method twice and print the value of the parameters after each method call.
2. Write a method with the following header: static void CalculateTuitionFee(int numberOfCourses, double costPerCourse, ref double fees). This method will calculate and assign the required fees amount to the third argument. [Fees = number of courses \* cost per course + 15.25].   
   From your program Main() method, call the CalculateTuitionFee () method four times supplying different arguments each time and display the value of the third argument after each method call.
3. Write a method that takes four parameter of type int. The method will assign the sum of the first two arguments to the third and the difference of the first two to the fourth. This method should be coded so that the calling method will use the value of the third and fourth parameters.   
   Call this method about three times and print out the value of the four parameters after each method call.
4. Write a method with header static void CalculateTrigValues(double degrees, out double sine, out double cosine, out double tangent). The method will use the first argument to compute the values of the other three arguments. Used the method Math.Sin, Math.Cos and Math.Tan to compute the second to fourth arguments respectively. [radians = degrees \* Math.Pi /180].   
   In the Main() method, invoke this method 20 times with the first argument taking the values 0, 5, 10, … 95 and display the four arguments in a professional tabular format.

The argument is taken as degrees

1. Write a method that takes three parameters of type double: the first represents an angle, the other two represents the sine and cosine of the angle respectively. The method must be able to change the actual value of the second and third argument. In your Main() method, call this method ten times with values 0.50, 0.51, 0.52 … 0.59 and printout the values for angle, sine and cosine in a tabular format

The argument is taken as radians

1. A quadratic equation is one in the form of ax2+bx+c = 0. Normally, there are two solutions to such equations given by the expression .The Write a method that takes five double arguments, the first three represents a, b and c respectively and the last two the solutions to the equation. [You can check your implementation, the quadratic equation 12x2+x-6 will yield solutions -0.750 & 0.667]

# Part V ‒ More difficult Methods: [Questions 1-6 required]

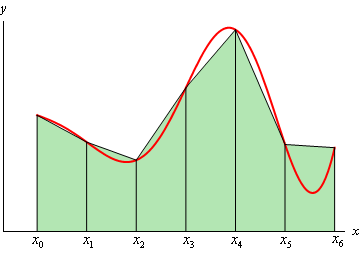
The solution provided has a menu. You do not need to create one.

1. Write a method with header static void DisplayIntArray(int[] numbers). The first argument is an array of ints. There is no return value. This method displays all the elements of the argument on a single line. Each item will occupy three columns.  
   Call this method from main with a suitable argument.
2. Write a method with header static int[] GenerateRandomIntArray(int numberOfItems, int largestValue). The first argument is an int indicating the number of elements that will be present in the return array. The second argument is an int representing the largest item in the array. The returned value is an array of integers. This method does the following:
   1. Declare and initialized a variable of type Random (Random generator = new Random();)
   2. Declare an array of type int (you may call it result)
   3. Allocate storage for the correct number of items
   4. Using your favorite looping statement, assign a random integer to each element of the array (result[i] = generator.Next(largestValue);)

In your main method, call the above method three times with arguments 15, 10 and 25, 10 and 30, 100. In each case assignment the results to a suitable variable and use the DisplayIntArray() method to display the value.

1. Write a method with header static int[] CountEvenOdd(int[] array). The argument is an int array. The returned value is an array of 2 integers. The elements of the return array will be a count of the odd and even items in the argument. The first element of the returned array represents the number of odds in the argument and the second element indicates the number of evens in the argument.  
   The method will create an int array of 2 elements (call this the result). Each item of the argument is examined and the appropriate element of the result array is incremented.   
   In your main method, call the GenerateRandomIntArray() method to create an int array using and assign it to a suitable variable. Using the DisplayIntArray() method display all the item of the return. Call the CountEvenOdd() method passing the above array as an argument and display the return value. Do a count to verify that your method is working correctly.
2. Write a method with header static int[] CalculateDigitFrequencies(int[] array). The argument is an int array with values ranging from 0 to 9. The returned value is an array of 10 integers. The elements of the return array will be a count of the frequencies of the items in the argument. The first element of the returned array represents the number of 0’s in the argument, the second element indicates the number of 1’s in the argument and the third element will indicate the number of 2’s in the argument.  
   The method will create an int array of 10 elements (call this the result). Each item of the argument is examined and the appropriate element of the result array is incremented.   
   In your main method, create an int array using the GenerateRandomIntArray() method remember to use 10 as the second argument to the method and then print out the elements using the DisplayIntArray() method. Call the above method and display the return value. Do a count to verify that your method is working correctly.
3. Write a method with header static int[] CalculateLastDigitFrequencies(int[] array). The argument is an int array. The returned value is an array of 10 integers. The elements of the return array will be a count of the frequencies of the items in the argument. The first element of the returned array represents the number with last digit of 0’s in the argument, the second element indicates the number with last digit of 1’s in the argument and the third element will indicate the number with last digit of 2’s in the argument.  
   The method will create an int array of 10 elements (call this the result). Each item of the argument is examined and the appropriate element of the result array is incremented.   
   In your main method, create an int array using the GenerateRandomIntArray() method and then print out the elements using the DisplayIntArray() method. Call the above method and display the return value. Do a count to verify that your method is working correctly.
4. Write a method with header static int[] CalculateNumberFrequencies(int[] array). The argument is an int array with values ranging from 0 to 99. The returned value is an array of 10 integers. The first element will indicate the number or unit values in the argument (i.e. values 0-9), the second element will indicate the number of 10 values (i.e. values 10-19), the third element will indicate the number of 20 values (i.e. values 20-29) etc.  
   The method will create an int array of 10 elements (call this the result). Each item of the argument is examined and the appropriate element of the result array is incremented.   
   In your main method, create an int array using the GenerateRandomIntArray() method and then print out the elements using the DisplayIntArray() method. Call the above method and display the return value. Do a count to verify that your method is working correctly.
5. Write a method that takes a string and return a distribution of the letter in the string. A distribution can be an array of the letter frequencies. To simplify code, you may assume that all the letters will be lowercased (or uppercased) and there are no periods or spaces. Call the above method and display the return value. Do a count to verify that your method is working correctly.
6. Write a method that returns a binary string representation of its argument. Call this method from your main and display the returned values. If you call the method three times with the integers 7, 128 and 15 respectively the method will return the binary strings 111, 10000000 and 1111 respectively.
7. Write a method that will return the year-end balance for a mortgage, given the beginning balance, the yearly interest rate, and the bi-weekly payments. You can assume that there are 26 payments in the year and the mortgage is re-calculated after each payment. For each period, you will need to calculate the interest (you will need to calculate the daily interest and times it by the number of days in each period), then add this to the starting balance and then subtract the payment. If the starting principal is $300,000, the interest rate is 2.5% and the bi-weekly payment is $1,000, what will be the year-end balance? How much would you save if you increased the monthly payment by $100?
8. Write a method that will display the nth term of a Fibonacci sequence. The nth term is defined as the sum of the two previous terms. The first few terms of the Fibonacci series are 1, 1, 2, 3, 5, 8, 13, 21, etc.
9. Write a method that will return the factorial of its argument number. [n! = n(n-1)(n-2)(n-3)… (1)]
10. Write a method that computes the value of PI given by the expression . How many terms are needed to match Math.PI?
11. Write a method that takes two integers and returns the greatest common divisor (gcd).
12. Write a method similar to question 3, that counts the distribution of two-letters combinations i.e. digrams

# Part VI ‒ Exotic Methods:

1. In cryptography, a Caesar cipher, also known as Caesar's cipher, the shift cipher, Caesar's code or Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, D would be replaced by A, E would become B, and so on. The method is named after Julius Caesar, who used it in his private correspondence. … source: https://en.wikipedia.org/wiki/Caesar\_cipher
2. Write a method call “Encrypt” that takes two arguments: a string that represents the message to encrypt and an int that represents the shift amount. The method returns a string of the encrypted message. The method should shift each letter of the message by the amount specified by the second argument. It might be a good idea to convert the string to a char array, do the shift and then convert the array back to a string.
3. Write a method call “Decrypt” that takes two arguments: a string that represents a secret message to decrypt and an int that represents the shift amount. The method returns a string of the original message. The method should shift each letter of the message by the amount specified by the second argument. It might be a good idea to convert the string to a char array, do the shift and then convert the array back to a string.
4. Write the necessary statements in your main method to work the two methods above.
5. Numerical integration is a technique that is particular suitable for computer applications. In exercise we will try to implement the trapezoidal rule. The integral of a mathematical function is the area between the curve and the x-axis. If the area is divided into little trapezoids, then the integral is approximated by the area of these geometrical figures.   
   You will try to find the area under the curve y = 6x2-7x+2 in the region from x = 0.5 to x = 1.5 (N.B. your answer should work out to about be 1.54, however the actual answer is 1.5). The area is given by the formula where y0 and y1 are the height of the vertical lines i.e. the value of the function.

This problem can be decomposed into three parts as follows:

* 1. Write a method called EvaluateQuadraticValue(double x, double a, double b, double c) that takes four double arguments: the value of x, the coefficient of the x2 term, the coefficient of the x and the constant. The method will compute and return the result value of y given by the expression y = ax2+ba+c.
  2. Write a method called ComputeQuadraticValues(double startX, double increments, int numberOfIntervals, double a, double b, double c) that takes six arguments: the start value of x, the increments and the number of intervals and the coefficients of the quadratic equation. This method will figure out the values of x0, x1, x2 etc. by invoking the previous question. The results of the methods calls are collected and returned as a double array.
  3. Write a method called ApplyTrapeziodalRule(double startX, double endX, int numberOfIntervals, double a, double b, double c) . The arguments are described in the previous question. This method calls the previous method and process the double array that is returned to compute the area under the curve by applying the formula .

1. Newton-Raphson method of calculating the square root of a number is given below: