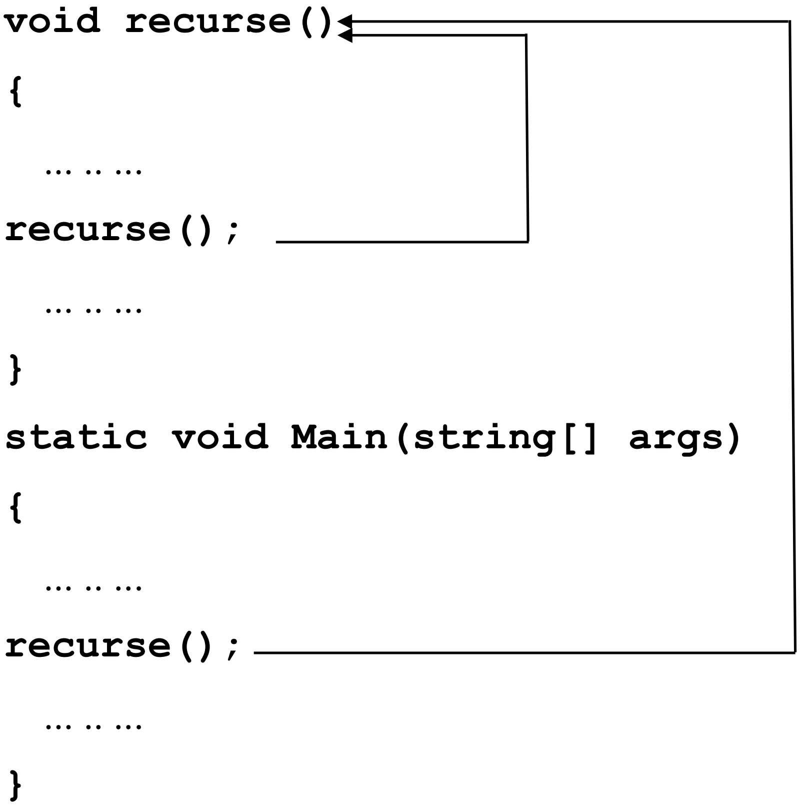
Advanced Programming in C#

Recursion in C#

- A function can call another function and sometimes may call itself.
- A function that calls itself is known as recursive function. And, this technique is known as recursion.
- Recursive function typically divides the problem into two conceptual pieces: a piece that the function knows how to do and a piece that it does not know how to do.
- It is terminated when the main condition no longer continues to be satisfied.

How does recursion work in C#?

- Recursion, takes the following general style:



```
void recurse()
{
  ... .. ...
recurse();
  ... .. ...
}
static void Main(string[] args)
{
  ... .. ...
recurse();
  ... .. ...
}
```

- The recursion continues until the termination condition is met.
- To prevent infinite recursion, if...else statement (or similar approach) can be used where one branch makes the recursive call and other doesn't.
- Therefore, it can be concluded that recursion consists of two key parts to work as follows:
  - **The recursion part:** which calls the recursion function
  - **The termination condition:** which stops the recursion function upon satisfying a certain condition
- As mentioned above, the problem is divided into number of smaller problems
- Each of these new problems look like the original, so the function calls a copy of itself to work on the smaller problem− this is referred to as a **recursive call** and is also called the **recursion step.**
- The **recursion step** often includes the keyword **return**, because its result will be combined with the portion of the problem the function knew how to solve to form the result passed back to the original caller,possibly main.
- Example 1: Factorial of a Number Using Recursion

```
// Factorial of n = 1*2*3*...*n
public static int factorial(int n)
{
    if (n > 1)
    {
        return n*factorial(n-1);
    }
    else
    {
        return 1;
    }
}
static void Main(string[] args) {
{
int n;
Console.WriteLine("Enter a number to find
factorial:);
n = Convert.ToInt32(Console.ReadLine());;
Console.WriteLine("Factorial of " + n +" = " +
factorial(n));
 }
```

Output

```
Enter a number to find factorial: 4

Factorial of 4 = 24
```

Explanation: How this example works

- Suppose the user entered 4, which is passed to the factorial() function.

4 * factorial(3)
3 * factorial(2)
2 * factorial(1)
1 *factorial(0)

1- In the first factorial() function, test expression inside if statement is true. The return num*factorial(num-1); statement is executed, which calls the second factorial() function and argument passed is num-1which is 3.

2- In the second factorial() function, test expression inside if statement is true. The return num*factorial(num-1); statement is executed, which calls the third factorial() function and argument passed is num-1 which is 2.

3- In the third factorial() function, test expression inside if statement is true. The return num*factorial(num-1); statement is executed, which calls the fourth factorial() function and argument passed is num-1 which is 1.

4- In the fourth factorial() function, test expression inside if statement is false. The return 1; statement is executed, which returns 1 to third factorial() function.

5- The third factorial() function returns 2 to the second factorial() function.

6- The second factorial() function returns 6 to the first factorial() function.

7- Finally, the first factorial() function returns 24 to the main() function, which is displayed on the screen.

Example: count down recursive function

```
public static void count_down(int n)

{

    if(n!=0)

    {

    Console.WriteLine (n);

     n--;
    count_down(n);

    }

    else

    {

        Console.WriteLine (n);

    }

}

static void Main(string[] args) {

{

    int k = 5;

    count_down(k);

}
```

Example: calculating the sum of all the numbers from n to m recursively:

```
public static int CalcSum(int n, int m)
 {
    int sum = n;
    if (n < m)
     {
       n++;
       return sum += CalcSum (n, m);
     }
  return sum;
}
static void Main(string[] args)
{
Console.WriteLine("Enter number n: ");
int n = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("Enter number m: ");
int m = Convert.ToInt32(Console.ReadLine());
int sum = CalcSum (n, m);
Console.WriteLine(sum);
 }
```

Example: Check if a string can be read from both sides or not

```
public static bool check(string str)
 {
 if (str.Length <=1)
   return true;
 if (str[0] == str[str.Length - 1])
   {
     str = str.Substring(1, str.Length - 2);
     return check(str);
    }
   else
   {  return  false;  }
 }
 static void Main(string[] args)
 {
```

```
 Console.WriteLine("Enter a string ");
 string str = Console.ReadLine();
if (check(str))
 {
   Console.WriteLine("Yes");
}
 else
  {
   Console.WriteLine("No");
  }
```