# Recursion in C or C++

*Jaydeep Shah (radhey04ec@gmail.com)*

Recursion: When function call itself for solving big and same repetitive task known as recursion.

Useful when task contains same repetitive pattern, for more details see below some examples.

Recursion base idea is “Break big problem into small chunk”.

1) Sum of n number : 1 + 2 + 3 + 4…. up-to n
   Here addition of two number and process repeat n times.

2) Factorial of n!  : n * (n-1) * (n-2) * …. 1
   Multiplication process repeat up to 1 to n

3) Fibonacci series: 0,1,1,2,3,5…..
   Addition of previous two number for getting next number, and same process will be repeat.

So Recursion is useful when task is same and repeat manner.

NOTE: Because function call itself, and that cause every time new call consume new stack area on RAM, so when our goal achieve need to break this repeat process using specific condition and this condition known as BASE condition.

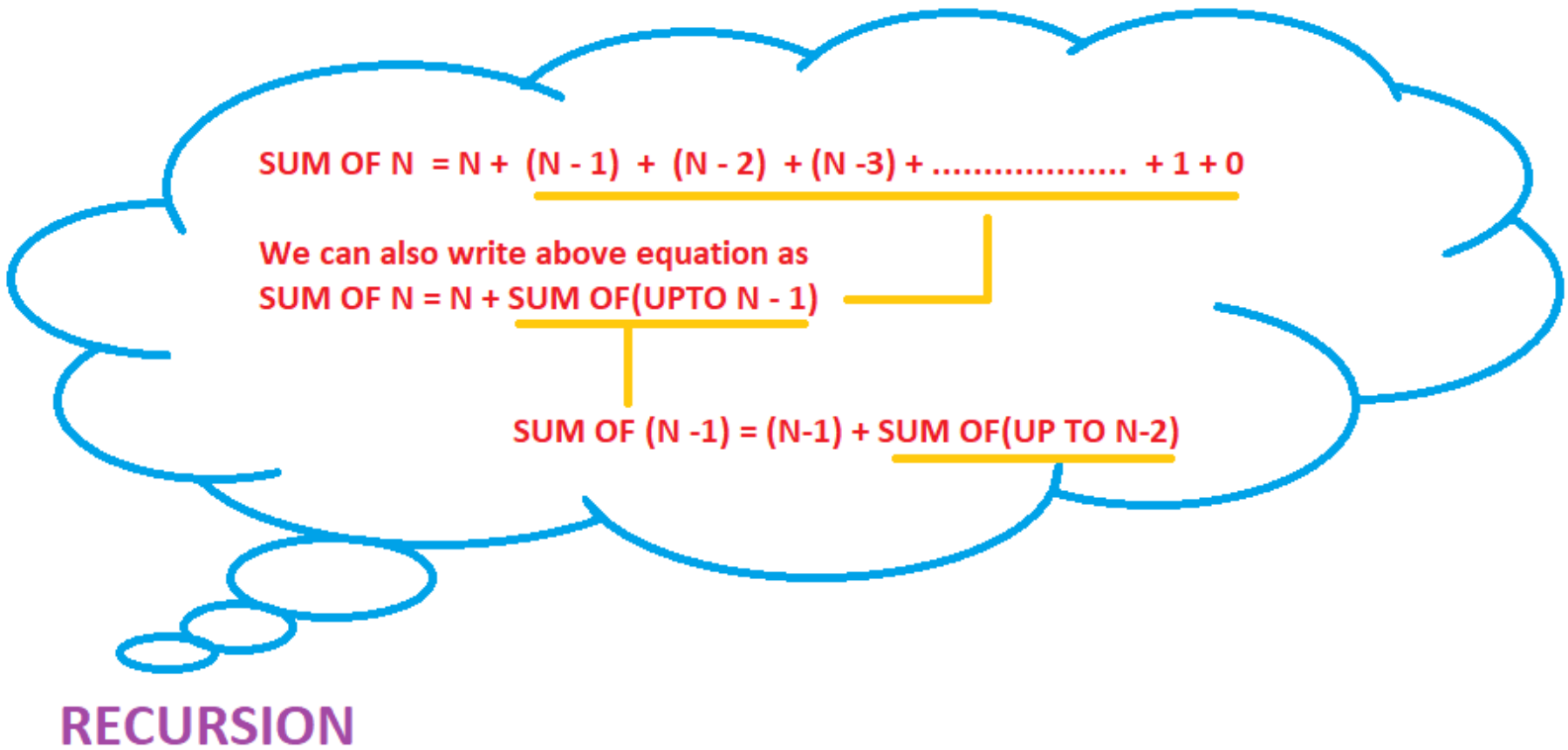
- Embedded system has limited RAM and that time needs enough care before using recursion.
- Break condition / base condition always needed inside recursion function.

## Code SUM of N digit using RECURSION:

```
//Tutorial :1 Recursion : When function call itself, useful to do repeat task
//Jaydeep shah (radhey04ec@gmail.com)

#include<iostream>
using namespace std;


//Function for finding sum of N number
//Recursive function
int SUM_OF_N(int n)
{
	//Base conditon
	if (n == 0)
	{
		return 0;
	}

	int PRV_SUM = SUM_OF_N(n - 1);
	return(n + PRV_SUM);

	//Or we can write return(n + SUM_OF_N(n-1))
}
```

```cpp
int main()
{
	//Eneter last digit
	int n;
	cout << "Last digit of N : ";
	cin >> n;
	cout << endl << "SUM OF UPTO N = " << SUM_OF_N(n);
	return 0;
}
```

```
Last digit of N : 5

SUM OF UPTO N = 15
--------------------------------
```




## Factorial Finding using RECURSION:

```cpp
//Tutorial :2 Recursion : When function call itself, useful to do repeat task
//Jaydeep shah (radhey04ec@gmail.com)


//Find Factorial of n, using recursion

#include<iostream>
using namespace std;

//Function for calculating factorial of n number
double factorial(int n)
{

	//Base condition
	if (n == 0)
	{
		return (1);
```

```
	}

	double prv_factorial = factorial(n - 1);
	return(n * prv_factorial);
}

int main()
{
	//Eneter last digit
	int n;
	cout << "Last digit of N : ";
	cin >> n;
	cout << endl << "Factorial OF UPTO N = " << factorial(n);
	return 0;
}
```

OUTPUT:

```
Last digit of N : 7

Factorial OF UPTO N = 5040
--------------------------------
```

## Is given array sorted or not ?? Find using recursion.

Code:

```
//Tutorial 4 : Recursion problems
//Find Array is sorted or not
//Jaydeep Shah (radhey04ec@gmail.com)


#include <iostream>>
using namespace std;


//Recursive function for finding given array is sorted or not ?
bool sorted(int data[], int size)
{

	if (size == 0)
	{
		//When only one element inside array it is always sorted
		return true;
	}
	bool answer = (data[size] > data[size - 1]) && sorted(data - 1, size - 1);
	return answer;
}

int main()
{
	//Store elements in Array
	int n;
	cout << "Number of digits inside Array ? ";
	cin >> n;
	int ARRAY[n];

	//Store elements
	cout << endl << "Enter Array digits : " << endl;
	for (int k = 0; k < n; k++)
	{
		cin >> ARRAY[k];
	}
```

```
	//Find sorted or not
	cout << endl << "Given Array is : " << sorted(ARRAY, n - 1);  //n-1 because of zero indexing
	return 0;
}
```

Output:

```
Number of digits inside Array ? 5

Enter Array digits :
1 3 2 4 2

Given Array is : 0
--------------------------------
```

```
Number of digits inside Array ? 5

Enter Array digits :
1 2 3 4 5

Given Array is : 1
--------------------------------
```

## Finding the first and last occurrence of the element inside ARRAY using the recursion.

### Code:

```
//Tutorial 5 : Recursion problems
//Find first and lat occurance of given key element inside array
//Jaydeep Shah (radhey04ec@gmail.com)


#include<iostream>>
using namespace std;

//Function for finding first occurance of the key element inside ARRAY
int FirstOCC(int data[], int size, int key, int index)
{
	//If element not found from whole Array
	if (index >= size)
	{
		return (-1);
	}
	if (data[index] == key)
	{
		return index;
	}
	return(FirstOCC(data, size, key, index + 1));
}

//Finding last occurance
int LastOCC(int data[], int size, int key, int index)
{
	//Base condition
	if (index == 0)
	{
		return (-1);
	}
	if (data[index] == key)
	{
		return index;
	}
```

```
	return(LastOCC(data, size, key, index - 1));

}
int main()
{
	//Store elements in Array
	int n;
	cout << "Number of digits inside Array ? ";
	cin >> n;
	int ARRAY[n];

	//Enter Array digits
	cout << endl << "Enter Array digits : " << endl;
	for (int k = 0; k < n; k++)
	{
		cin >> ARRAY[k];
	}

	//Key number input
	cout << endl << "Key number = " << endl;
	int key;
	cin >> key;

	//Print Fist and last occurance
	int first_occurance = FirstOCC(ARRAY, n, key, 0);
	int last_occurance = LastOCC(ARRAY, n, key, n);
	cout << endl << "First Occurance = " << first_occurance << " Last Occurance = " << last_occurance;
	return 0;
}
```

OUTPUT:

```
Number of digits inside Array ? 8

Enter Array digits :
1 5 3 5 68 5 89 90

Key number =
5

First Occurance = 1 Last Occurance = 5
--------------------------------
```