Recursion

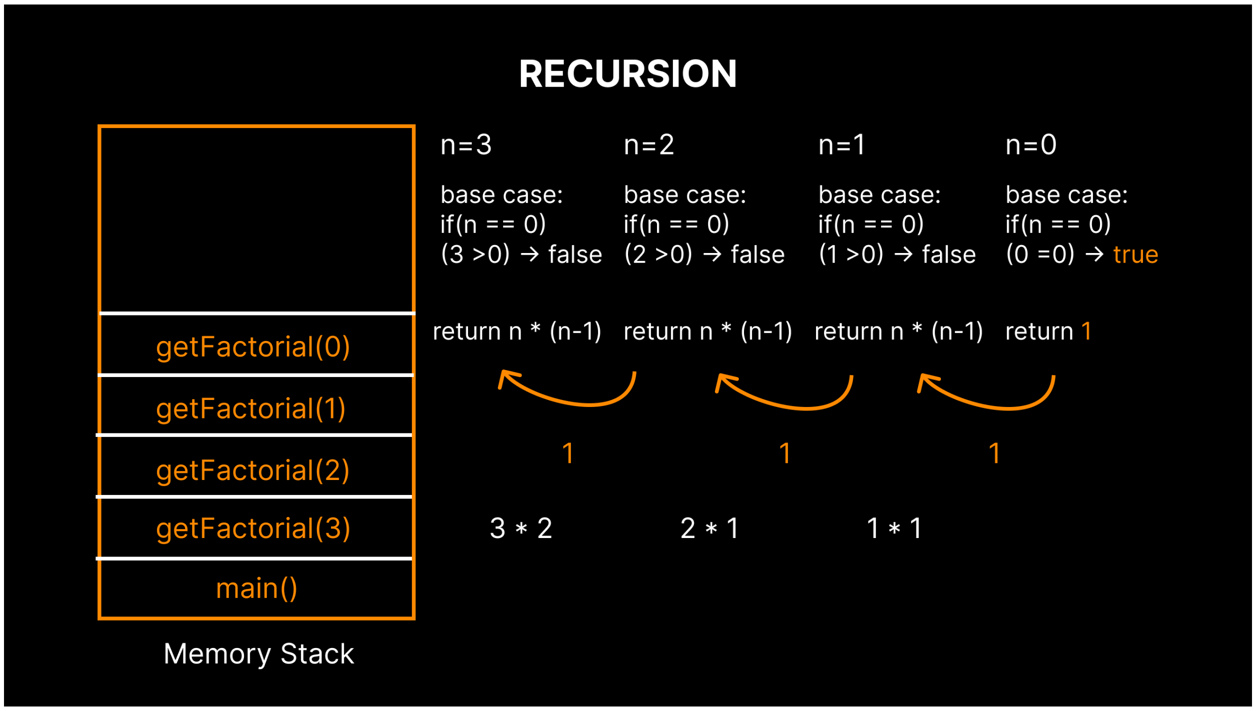
Recursion is a programming and mathematical concept that refers to the process of a function or algorithm calling itself to solve a problem. In a recursive function or algorithm, a problem is divided into smaller, similar sub-problems, and each sub-problem is solved using the same function or algorithm. This process continues until a base case is reached, which is a condition that signals the termination of the recursion.

Recursion consists of two main components:

1. Base Case: The base case is the condition that stops the recursive calls and provides a result without further recursion. It's essential to prevent infinite recursion and ensure that the recursion eventually terminates.
2. Recursive Case: The recursive case defines how the problem is broken down into smaller sub-problems and how the function or algorithm calls itself to solve these sub-problems.

Example:

Finding the factorial of a number using recursion.



Code for the program:

```
#include<iostream>
using namespace std;

int getFactorial(int n){

    // base case is mandatory
    if(n == 0)
        return 1;

    return n * getFactorial(n-1);
}

int main()
{
    int n;
    cout << "Enter the number: ";
    cin >> n;
```

}

```
    int ans = getFactorial(n);

    cout<<"Factorial of "<< n <<" is: "<< ans <<endl;
    return 0;
}
```

Power of Two

Code for power of 2:

```
#include<iostream>
using namespace std;

long int PowerOfTwo(int n){
    if(n == 0)
        return 1;

    return 2 * PowerOfTwo(n-1);
}

int main()
{
    int n;
    cout << "enter the number: ";
    cin >> n;

    int ans = PowerOfTwo(n);

    cout << "2 raised to " << n <<" is: "<< ans <<endl;

    return 0;
}
```