Calculate Factorial of a Number

In this lesson, you will see how to calculate the factorial of a number using a recursive function.

We'll cover the following

Introduction

Let's consider the example of the recursive factorial function! In this lesson, we will calculate the factorial of a number `n` denoted by `n!`.

> 📝 The factorial of a given number is the product of the number by all the numbers smaller than it until it reaches 1.

$$n! = n \times (n-1) \times (n-2) \ldots\ldots\ldots\ldots\ldots 2 \times 1$$

$$n! = n \times (n-1)!$$

Where ,

$$0! = 1 \text{ and } \ 1! = 1$$

> 📝 We can only calculate the factorial for the non-negative integers.

Illustration

Consider the illustration below for `n = 5`. You will see that we can represent the factorial problem in terms of itself, and eventually, we will reach a case that can be solved directly. When we reach that case, we will start returning value to the calling function.

1 of 9

9 of 9

Example program

RUN the program below and see the output!

```
#include <iostream>

using namespace std;

// Recursive factorial function
int factorial(int n) {
  // Invalid value
  if (n < 0){
    return -1;
  }
  // Base case
  if (n == 1 || n == 0) {
    return 1;
  }
  // Recursive Case
  else {
    return n * factorial(n - 1);
  }
}

// main function
// main function
int main() {
  int n = 5;
  int result;
  // Call factorial function in main and store the returned value in result
  result = factorial(n);
  // Prints value of result
  cout << "Factorial of " << n << " = " << result;
  return 0;
}
```

Explanation

factorial function

Line No. 6: The recursive `factorial` function takes a value of type `int`, whose factorial is to be calculated in its input parameters, and returns the factorial of value in the output.

Line No. 8 Since we cannot calculate the factorial of negative integers for `n < 0`, `factorial` simply returns **-1** in the output. **-1** indicates that we have entered an invalid value.

Line No. 12 If `n = 1` or `n = 0`, the function terminates after returning **1** to the calling point. There are no recursive calls in the `factorial` body since we cannot break the expression anymore. This is the base case of the `factorial` function.

Line No. 17 If `n > 1`, then the `factorial` returns the product of `n` by the `factorial (n-1)`. This is the recursive case.

Let's run our code for `n = 5` and see what happens inside the recursive `factorial` function.

9 of 10

10 of 10

In the above illustration, we see how to use the results of the inner function call to terminate the outer function call.

Quiz

Q

If `n = 6`, then what is the output of the following code?

```
int factorial(int n) {
  if (n < 0){
    return -1;
  }
  if (n == 1 || n == 0) {
    return 1;
  }
  else {
    return n * factorial(n - 1);
  }
}
```

Retake Quiz

Let’s see the difference between recursion and iteration in the upcoming lesson.