Calculating the Factorial of a Number

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 x (n-1) x (n-2).....2 x 1

n! = n x (n-1)!

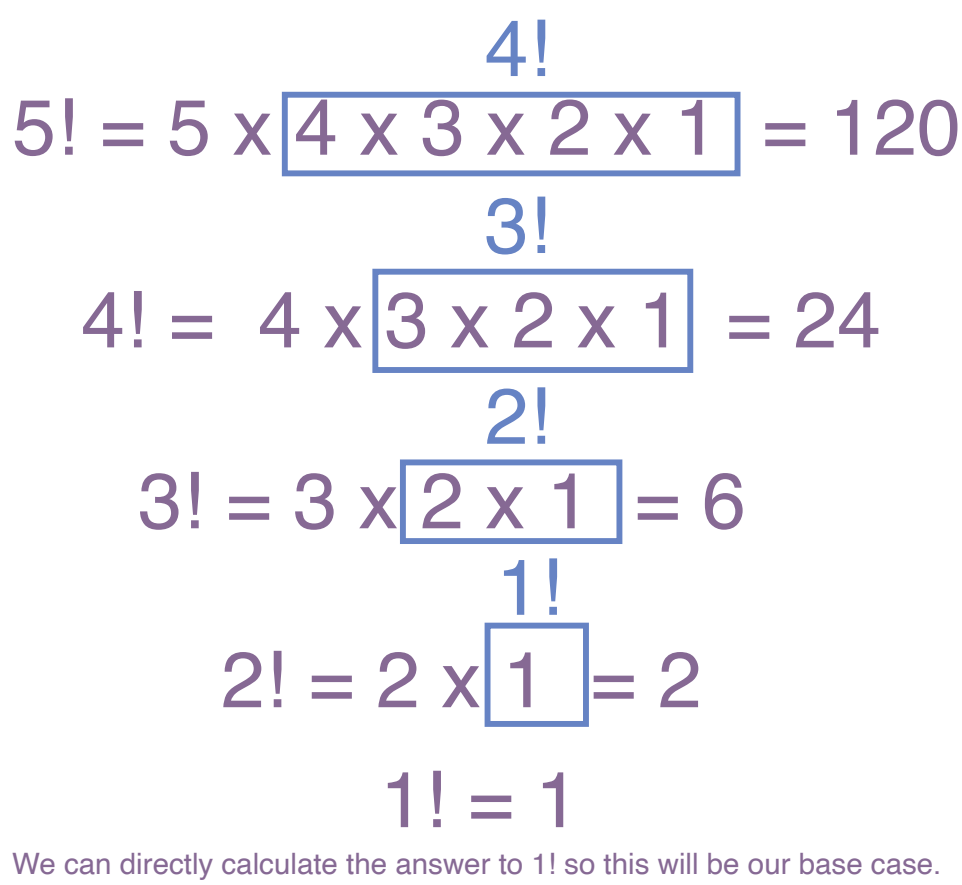
Where ,

0! = 1 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, reach a case that can be solved directly. When we reach that case, we will start returning value to the calling function.



Example program#

Run the program below and see the output!

```
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
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;
}
```

Output 1.01s

```
Factorial of 5 = 120
```

Explanation#

factorial function

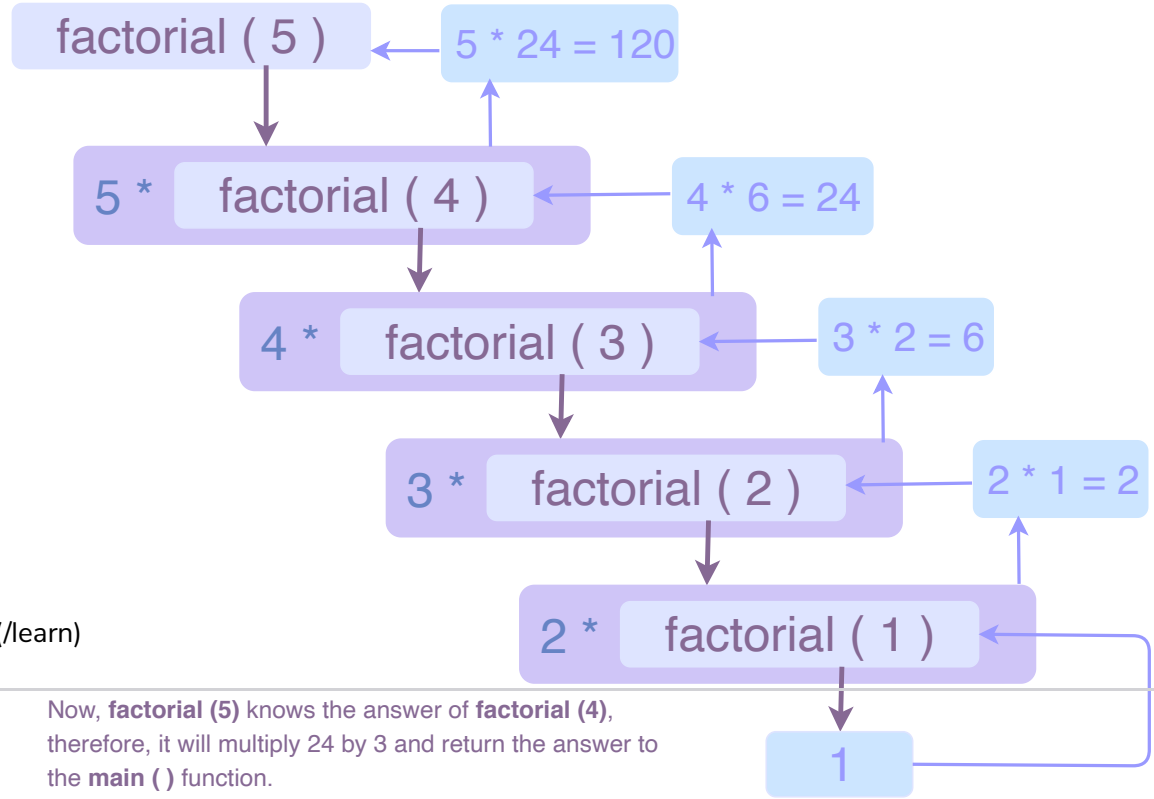
Line No. 6: The recursive `factorial` function takes a value of type `int`, whose factorial is to be calculated in its input parameters, and returns the factorial of value in the output.

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`, 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.



Now, **factorial (5)** knows the answer of **factorial (4)**, therefore, it will multiply 24 by 3 and return the answer to the **main ( )** function.

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`, what is the output of the following function?

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

A) 120

Your Answer

B) 720

C) 5040

D) 40320

Submit Answer

Reset Quiz

Let's learn about the difference between recursion and iteration in the upcoming lesson.