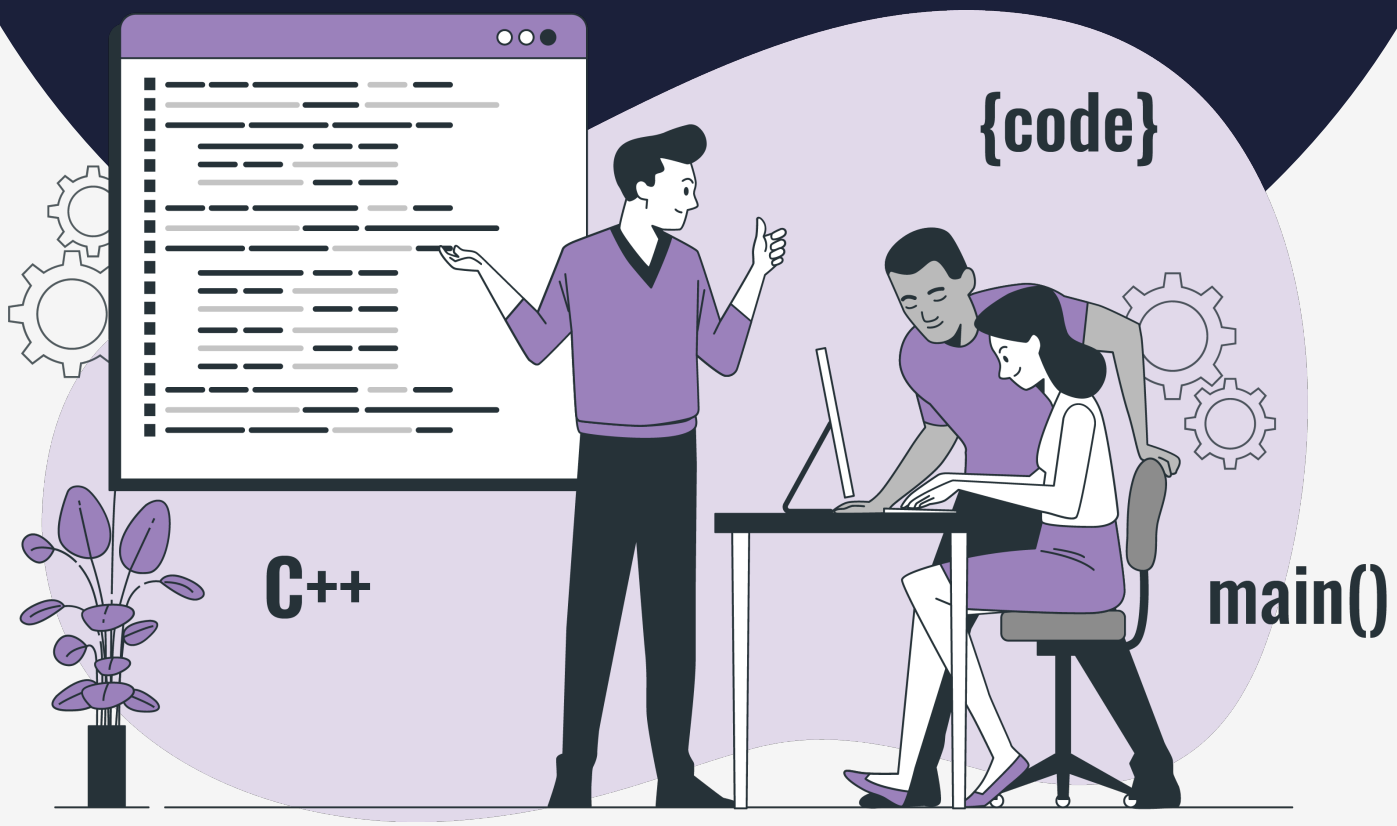
Lesson:

Recursion



PW SKILLS

Pre-Requisites

- Functions
- Arrays

List of Concepts Involved

- Introduction to recursive functions
- Working rules of recursive functions
- Problems based on Recursion

Topic 1: Introduction to recursive functions

Let's say if someone ask you to calculate 5! , you can say give me 4! and I will just multiply is to 5 and return it to you then the other person will say that no, you give me 3! and then I will multiply is by 4 to give you 4! and so on...

In this example everyone is doing the same task . A recursive function works in the same way.

More formally, a recursive function is a function that calls itself again and again until certain conditions are met. It has basically two parts:

1. A precondition that is used to stop this recursive call (Halting Condition).
2. A function that is capable of calling itself (recursive call).

Halting Condition:
Just like how we have a condition in an iterative statement to terminate the loop similarity, it must have a halting condition to terminate the recursive call; otherwise, it will result in an overflow error as it will go inside an infinite recursive call.

Why Do We Need Recursion?
Recursion is generally used when dealing with complex problems and problems that form a hierarchical pattern; it solves the original problem breaking into the smaller subproblems. Recursion is inefficient in cases where the same value is calculated again and again. It requires extra memory on the stack for each recursive call.

Topic 2: Working of Recursive functions

A recursive function has 3 parts -

1. **Base statement -** It is the statement at which the recurrence will terminate. It is generally a condition for which the solution is known or can be calculated easily. Without it the recurrence will continue forever.
2. **Recurrence statement -** It is the statement which calls the function again.
3. The rest of the function where we do our computations.

Syntax
In C++ we can write a recursive function using the following syntax:

```
methodName (N parameters )
{
    if(haltCondition){
        return result
    }
    return methodName (N parameters )
}
```

Topic 3: Problems based on Recursion

Problem 1-
Find the value of 5!.

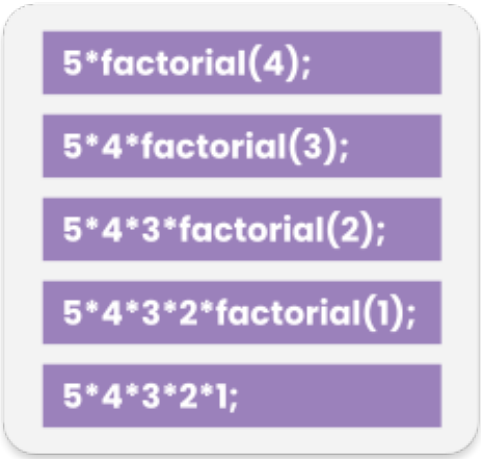
Code
https://pastebin.com/83TEvzzG

Output

```
120


...Program finished with exit code 0
Press ENTER to exit console.
```

Explanation -
For any number 'n', we can write its factorial as 'n' multiplied by the factorial of its previous number. This is a statement we know to be true. So we recursively call our function to return to us the factorial of the previous number in order to calculate the factorial of the current number by the help of this statement.



factorial(5) will call for factorial(4)
factorial(4) will call for factorial(3)
factorial(3) will call for factorial(2)
factorial(2) will call for factorial(1)
Now we know the value of factorial(1) to be 1. So we simply return the value i.e. our base case.
Now factorial(1) will return 1. Using this value factorial(2) will be calculated.
factorial(2) = 2 * factorial(1)= 2 * 1 = 2
Similarly,
factorial(3) = 3 * factorial(2) = 3 * 2 = 6
factorial(4) = 4 * factorial(3) = 4 * 6 = 24
factorial(5) = 5 * factorial(4) = 5 * 24 = 120

At the end we have factorial(5)= 120, the desired output

Problem 2: Write a Program to find nth fibonacci number.

Solution
The Fibonacci series is the sequence of numbers , where every number is the sum of the preceding two numbers, such that the first two terms are '0' and '1'.

The Fibonacci numbers are the numbers in the following integer sequence.

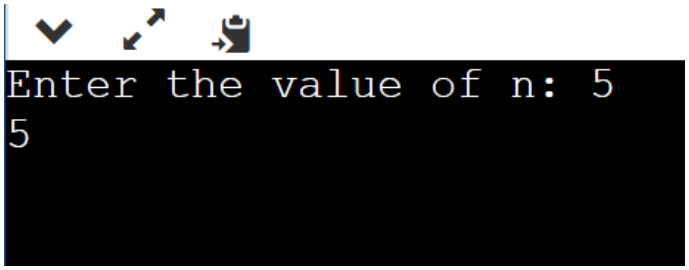
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ……..

In mathematical terms, the sequence Fn of Fibonacci numbers is defined by the recurrence relation

$F_n = F_{n-1} + F_{n-2}$
Here F0=0,F1=1, F2=1 ……

Code to find nth fibonacci number

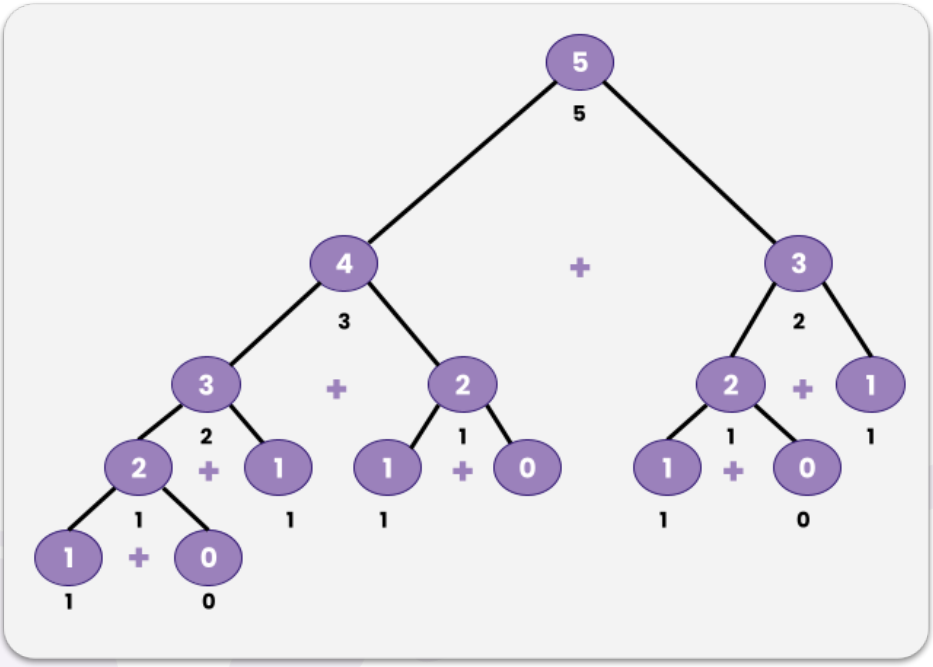
https://pastebin.com/Genivh1s



Explanation:

1. Lets define a function, say fibo() which takes an integer a and it will return the ath fibonacci number which is an integer.
2. Here the base case condition is If a == 0 then return 0 or a ==1 then return 1. Since Fibonacci of 0th term is 0 and 1st term is 1.
3. If a > 1 then return fibo(a - 1) + fibo(a-2). Since the Fibonacci of a term is the sum of the previous two terms.

For n=5,the flow of the above code will be:



Upcoming Class Teasers

- Problems based on recursion