Pseudocode for factorial

```
function fact(num):
        if num is equal to 0 then
                return 1
        else
                ans = 1
                for i  from 1 to num:
                        ans = ans * i
                end
                return ans
end function


userInput = input from user
resultedFact = fact(userInpt)
```

Explanation for the above pseudocode

First the userInput will take a number as input from the user and the variable resultedFact will call(execute) and store the value that returned by the fact(userInput).

The above function (function fact(num)) is a modular function. First it will check if the num is 0 then it will return 1 else declare a variable as ans and iterate i from 1 to num and reassign ans * i in ans. After end of the loop return the ans.

Pseudocode for nth Fibonacci number

```
function fib(num):
        if num is equal to 0 then:
                return 0
        else if num is equal to 1 then:
                return 1
        else :
                previous_value = 0
                current_value = 1
                for i from 2 to num:
                        next_value, previous_value  = current_value + previous_value ,
current_value
                end for loop
                return current_value
end function


userInp = input from user
resultedFib = fib(userInp)
```

Explanation for the above pseudocode

First the userInput will take a number as input from the user and the variable resultedFib will call(execute) and store the value that returned by the fact(userInput).

The function will take one parameter (i.e num) to the nth Fibonacci num. It will check if the num is equal to 0 or 1 it will return as 0 and 1 respectively. If this case is false then it will move to else condition.

Here make previous_value as 0 and current_value as 1. Now iterate i from 2 to num and with in the loop we calculate each Fibonacci number by adding the two previous_values.

After the loop end, we return the current_value, it's the value for the nth Fibonacci number.

Modularity

1) In programming modularity plays a vital role by breaking down the whole project into small functions.

2) It helps the developers for code reusability and easy maintenance.

3) As modular design is a breaking of components, if a problem occurs at a particular point, then instead of changing the whole code it's easy to change the particular component.

4) It makes the testing easier and it facilitates the troubleshooting and can make easy modifications.

5) By using this modular design in programming it improves the readability, scalable, troubleshooting, quick modification and maintenance easily.