Day-1 Date:26-04-2024

Assignment 3: Function Design and Modularization - Create a document that describes the design of two modular functions: one that returns the factorial of a number, and another that calculates the nth Fibonacci number. Include pseudocode and a brief explanation of how modularity in programming helps with code reuse and organization.

Solution:

Firstly, we design both the functions that returns the factorial of a number and Fibonacci series up to nth number

Designing and Pseudocode for the function that returns factorial of a number.

factorial (int n)

Logic:

```
if(n==0) {
        return 1;
        }
Else {
        int facto = 1;
        for(int i=1;i<=n; i++){
        facto facto*i;
        return facto;
}
```

Pseudocode:

1. Start

2.Get a number

3.check if the number is equal to zero or not.

4.if yes print 1 as answer.

5.else declare a variable facto

6.Run a loop for(i=1 to n)

7.if loop accessed multiply to i and store it in facto variable.

8. After loop ends return the final result in facto variable.

9.End

Designing and Pseudocode for the function that returns Fibonacci series of nth number.

Fibonacci(int n)

Logic: int a=0,b=1,c,i,n;

```
if (n<=2){
    system.out.print(a+" "+b);
}
Else {
   for (i=2; i<n; i++){
   c=a+b;
   system.out.println(c);
   a=b;
   b=c;
}
```

Pseudocode: 1. Start.

2.Get a number.

3.Declare variables a=0, b=1,c,i,n.

4. check if the number is equal to 2.

5.if yes print a and b.

6.else run a for loop for(i = 2 to n) because already 2 numbers are printed.

7.In loop previous 2 numbers were added and print it as a next number in series.

8. Then to change the previous numbers b value is assigned to a and c value assigned to b

9. Again, if loop continued then changed previous numbers were added and print it as a next number in series.

10.if loop breaks then it is final Fibonacci series

11.End.

Modularity in programming:

Modularity means making functions ready before and giving access to the end user to decide which function to be used. In programming language, we say this as a loosly coupling between functions.

To achieve this in our present situation we can use switch statement between 2 functions.

Rough code to use a switch case.

Enter a number

switch(choice):

case 1: print factorial(number).

break;

case 2: print Fibonacci(number).

break;

While using switch case we can use the modularity between functions if end user selects case1 it prints the factorial of a number or end user selects case 2 it prints the Fibonacci series upto nth number

By declaring the functions before we can reuse the code by just calling that function. Providing modularity while programming is the best way to organize the program.