Assignment 2 :

Write a recursive function pseudocode and calculate the nth Fibonacci number and use Big O notation to analyse its efficiency. Compare this with an iterative approach and discuss the pros and cons in terms of space and time complexity.

Pseudocode :

```
Read  n

 a = 0
 b = 1
 if n <= 1:
   print  n
  for i = 2 to n:
  temp = a + b
   a = b
   b = temp
   print b
```

Pros and Cons:

1. Constant space complexity makes it more memory efficient.
2. Linear time complexity makes it efficient, especially for large values of n.
3. May be less intuitive for some programmers, especially those less familiar with iterative algorithms.

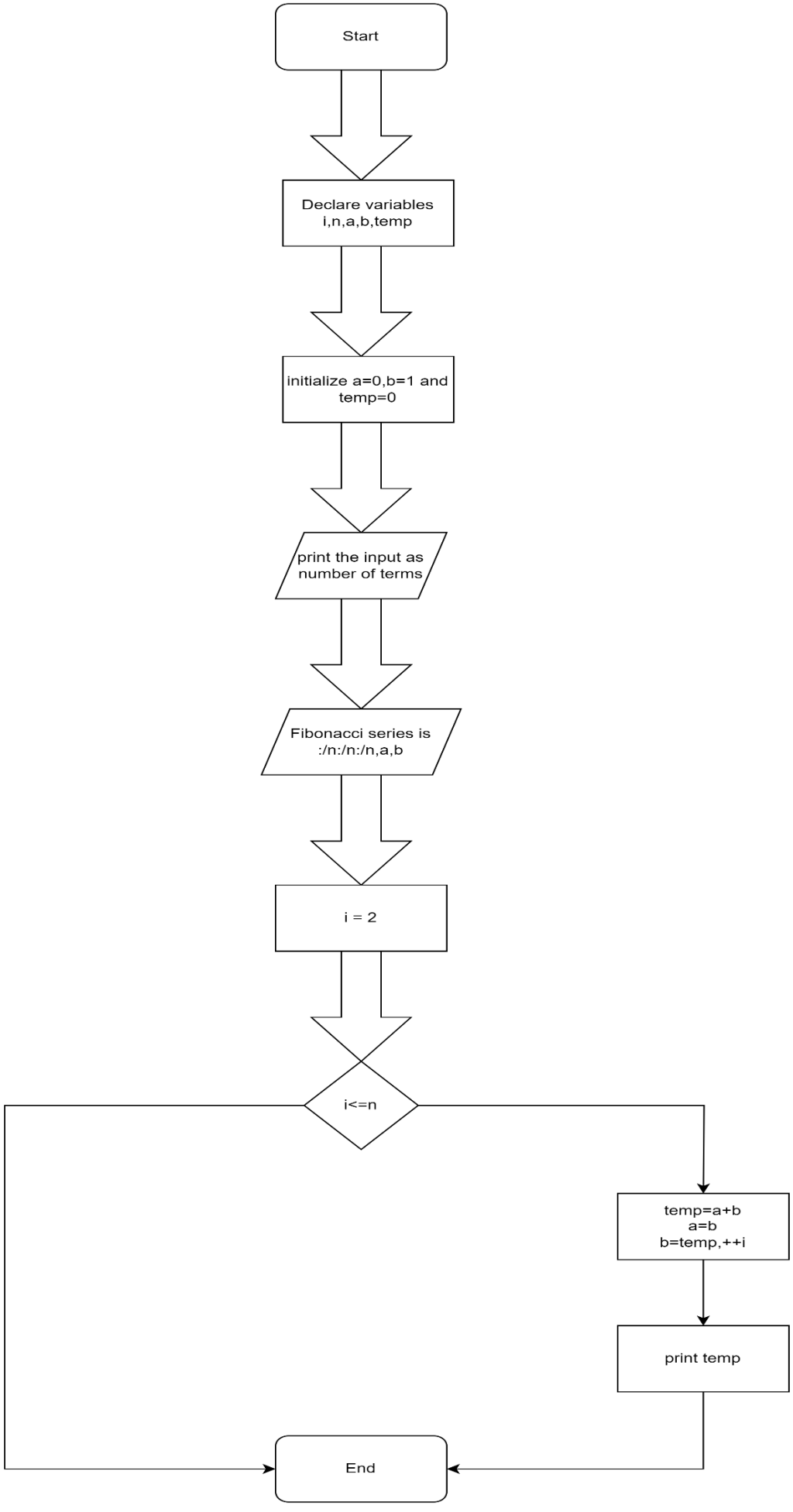
CODE:

```
#include<stdio.h>

int main()

{
   int  n,i,temp,a,b;
   a=0;
   b=1;
   printf("Enter the value of n to calculate the nth Fibonacci number:");
   scanf("%d", &n);
   if(n < 0)
     {
         printf("invalid input! n must be a non negative integer.\n");
         return 1;
      }
    for(i=2;i<=n;i++)
      {
         temp = a+b;
         a = b;
         b = temp;
       }
         printf("The %dth Fibonacci number is: %d\n", n,b);
         return 0;
  }
```

Algorithm :

1. Start
2. Declare variables i, a,b , temp
3. Initialize the variables, a=0, b=1, and temp =0
4. Enter the number of terms of Fibonacci series to be printed
5. Print First two terms of series.
6. Use loop for following steps
   i. temp =a+b
   ii. a=b
   iii. b= temp
   iv. increase value of i by 1 for each loop.
   v. print value of temp
7. End

FLOW CHART:


Start
Declare variables i,n,a,b,temp
initialize a=0,b=1 and temp=0
print the input as number of terms
Fibonacci series is :/n:/n:/n,a,b
i = 2
i<=n
temp=a+b
a=b
b=temp,++i
print temp
End