**Experiment No.3b**

**Title:** Fibonacci Sequence using Recursion in C++

**Problem Statement:** Write a C++ program to generate a Fibonacci sequence using Recursion.

**Algorithm:**

**Step 1** -Start

**Step 2 –**Read number n

**Step 3 –**Call fibonacci(n)

**Step 4 –**Print fibonacci f

**Step 5 –**Stop

**Fibonacci(n)**

**Step 1** -If (n==0) or(n==1) then return f=n

**Step 2 –**Else

f=fibonacci(n-2) + fibonacci(n-1)

**Step 3 –**Return f

**Code:**

#include<iostream>

using namespace std;

int fibonacci(int n)

{

int f;

if(n==0 | n==1)

return f=n;

else

f=fibonacci(n-2)+fibonacci(n-1);

return f;

}

int main()

{

int num,i=0;

cout<<"\nEnter the number of terms of fibonacci series ";

cin>>num;

cout<<"\nThe fibonacci series:";

while(i<num)

{

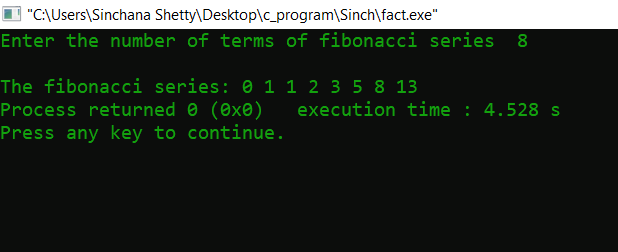
cout<<" "<<fibonacci(i);

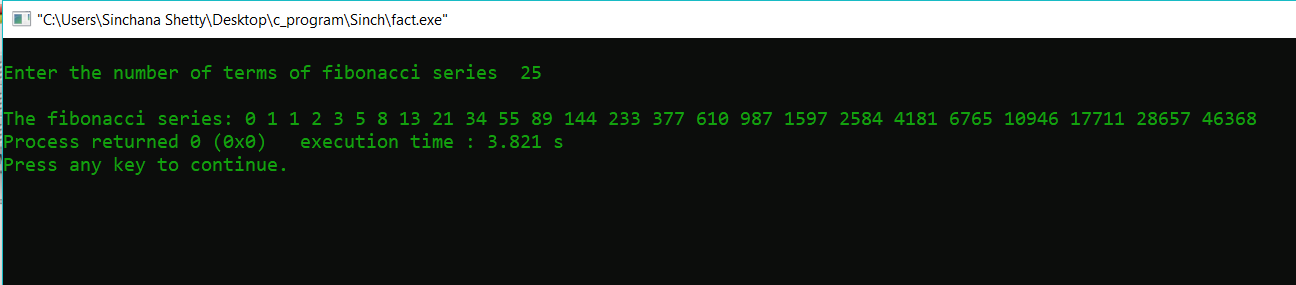
i++;

}

}

**Results:**





**Analysis(Limitations):**

This Code will take time in generating the maximum number of fibonacci terms it is just that we have to wait longer for the result. But the memory comes into picture where Analysis of recursive program space is more complicated.The space used at any time is the total space used by all recursive calls active at that time. Each recursive call takes a constant amount of space: some space for local variables and function arguments, but also some space for remembering where each call should return to.