**Data Structure and Algorithm**

**LAB 05**

**Name: Ahmad Amjad Mughal**

**Reg No: 121672**

**Class: BSCS-6C**

**Task 1**

**Code of Fibonacci series using Iteration and Recursion:**

//Ahmad Amjad mughal

//BSCS-6C

//121672

#include <iostream> // input output library

#include <ctime> // library to calculate time consumed in seconds

using namespace std; //use for cout and cin

void iteration() //iteraion function that uses loop

{

long int series = 10000000; // number that is local variable

int first = 0, second = 1; //using variables

int fibo;

for (long int c = 0; c < series; c++) //loop that continues till c is equal to series

{

if (c <= 1) // for c is equal to 1

fibo = c;

else //else statement executes when if condition fails

fibo = first + second;

first = second;

second = fibo;

}

}

int fibonacci(int n); //predefined function

int recursion() //recursion function that call function fibonacci

{

long int n = 32; //local variablle

int i = 0;

while (i<n) //while loop

{

cout<<" "<<fibonacci(i); //function call

i++;

}

cout << endl; //print next line

return 0;

}

int fibonacci(int n) //function that call itself

{

if ((n == 1) || (n == 0)) //base case for fibonacci series

{

return(n); //returns number

}

else

{

return(fibonacci(n - 1) + fibonacci(n - 2)); //else statement which is fibonacci operatiuon executes

}

}

void timeIteration(){ //timeIteration function do display the time consumed by compiler to execute fibonacci by iteration

double cpu\_time\_used;

clock\_t start = clock(); // start timer

iteration();

clock\_t stop = clock();

cpu\_time\_used = double(stop - start) / CLOCKS\_PER\_SEC; //time in seconds

cout << "Completion time for Iteration is " << cpu\_time\_used <<"sec"<< endl;

}

void timeRecursion(){ //timeRecursion do display the time consumed by compiler to execute fibonacci by recursion

double cpu\_time\_used;

clock\_t start = clock(); // start timer

recursion();

clock\_t stop = clock();

cpu\_time\_used = double(stop - start) / CLOCKS\_PER\_SEC; //time in seconds

cout << "Completion time for Recursion is " << cpu\_time\_used <<"sec"<< endl;

}

void main() //main that calls time function of both iteration and recursion

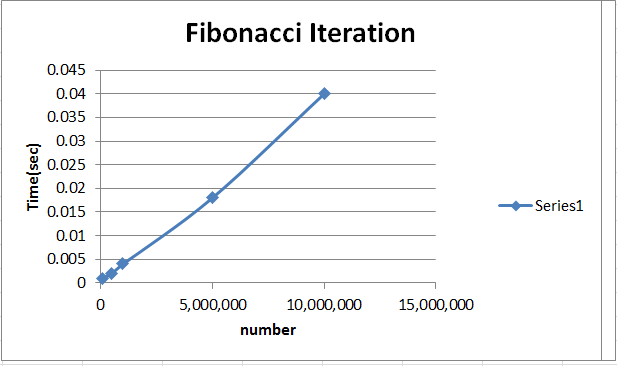
{

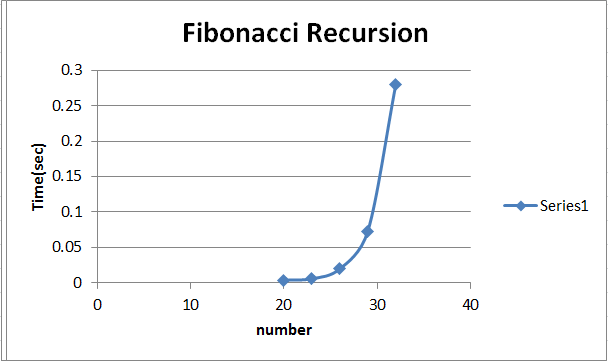
timeIteration();

timeRecursion();

}

**Graphs:**

****

****

**Comparison between Fibonacci by recursion and iteration:**

* For smaller values of n, Iteration is quick and recursion is bit slow.
* For greater values of n, Recursion is fast process than Iteration because it make function calls while iteration contain loops.
* Iterative is evaluating in polynomial time while recursive approach is evaluating in exponential time

**Task 2**

**Code of Factorial series using Iteration and Recursion:**

//Name: Ahmad Amjad mughal

//Class: BSCS-6C

//Reg No: 121672

#include<iostream> //library input output stream

#include<ctime> //library to display time in seconds

using namespace std; //for cout and cin

int iteration(){ //function iteration that contain loop

int n = 10;

long count = 1;

for (int i = 1; i <= n; i++) //loop esecution

{

count = count \* i; //operation

} cout << "The factorial for the number using loop is " << count << endl; //prompt

return 0;

}

int factorial(int n);

int recursion() //function recursion that uses input and call a function

{

int n; //local variable

cout << "Enter a positive integer: "; //user defined number

cin >> n;

cout << "Factorial of " << n << " = " << factorial(n) << endl; //prompt for display

factorial(n); //function call

return 0;

}

int factorial(int n) //funcation that call itself

{

if (n > 1)

return n \* factorial(n - 1); //for n > 1 then if statement executes

else

return 1; //base case

}

int timeIteration()

{

double time\_used;

clock\_t start = clock(); // start timer

iteration();

clock\_t stop = clock(); // stop timer

time\_used = float(stop - start) / CLOCKS\_PER\_SEC; //time in seconds

cout << " Time used for Iteration is " << time\_used << endl; //to display consumed time by iteration

return 0;

}

int timeRecursion()

{

double time\_used;

clock\_t start = clock(); // start timer

recursion();

clock\_t stop = clock(); // stop timer

time\_used = float(stop - start) / CLOCKS\_PER\_SEC; //time in seconds

cout << " Time used for recursion is " << time\_used << endl; //to display consumed time by recursion

return 0;

}

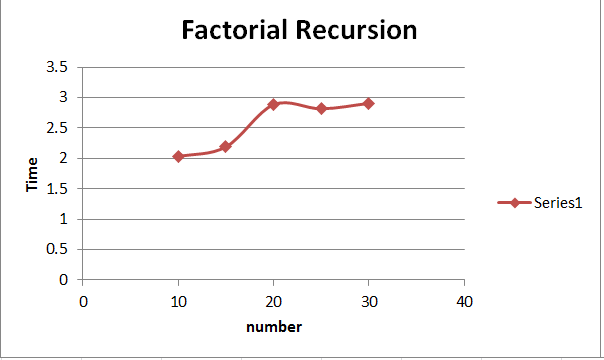
void main(){ //main function that calls time functions

timeIteration();

timeRecursion();

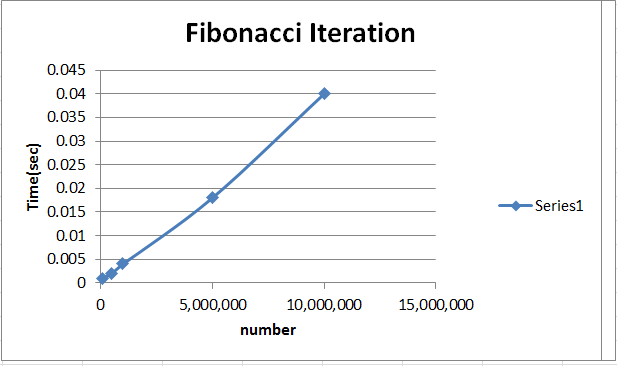
}

**Graphs:**

****

Graph for factorial Iteration for smaller values consume 0.05 sec time.





**Comparison between Fibonacci by recursion and iteration:**

* In Iteration, stack memory is allocated before runtime while in recursion, stack memory is allocated during runtime.
* Time complexity of Iteration is O(n) while for recursion it is much bigger as there are two cases to be executed. One is base case and other is recursive call.