Applications with Repetition Statements

This lesson is aimed to introduce the ideas of algorithm implementation. Algorithm implementation is an important job of engineer. We have learnt sufficient basic tools to implement programming solution for some real world problems. Some tools will be taught afterwards and they will further strengthen your ability to implement algorithms.

Algorithm

According to the author of our reference book [Dietel & Dietel], algorithm is defined as

"*Any solvable computing problem can be solved by executing of a series of actions in a specific order. A procedure for solving a problem in terms of*

1. *The actions to execute*
2. *The order in which the actions execute*

is called an algorithm."

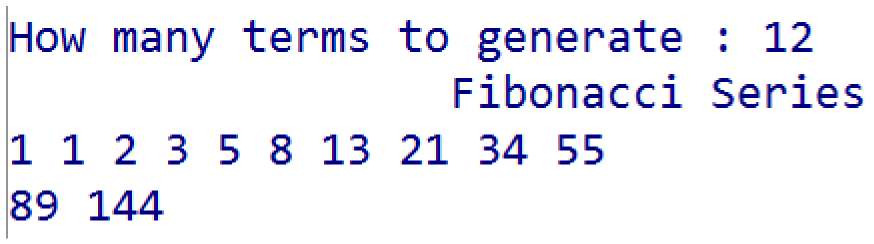
Objective of this lesson is to implement some simple algorithms using repetition statements.

les_06_code_01.cpp

Fibonacci Sequence with for loop

```
#include<iostream>
using namespace std;
int main (void)
{
// variable definition
int counter, n_terms;
cout<<"How many terms to generate : ";
cin>>n_terms;
// initializing newterm and prevterm to 1
//sum cantains the next number in the sequence
int newterm=1,prevterm=1,sum;
cout<<"\t\t Fibonacci Series"<<endl;
// display the first term
cout<<prevterm<<" ";
// for rest of the terms
for(counter=1;counter<n_terms;counter++)
{
if(counter%10==0)cout<<endl;
cout<<newterm<<" ";
sum=prevterm+newterm;
prevterm=newterm;
newterm=sum;
}
return 0;
}
```

Sample Output



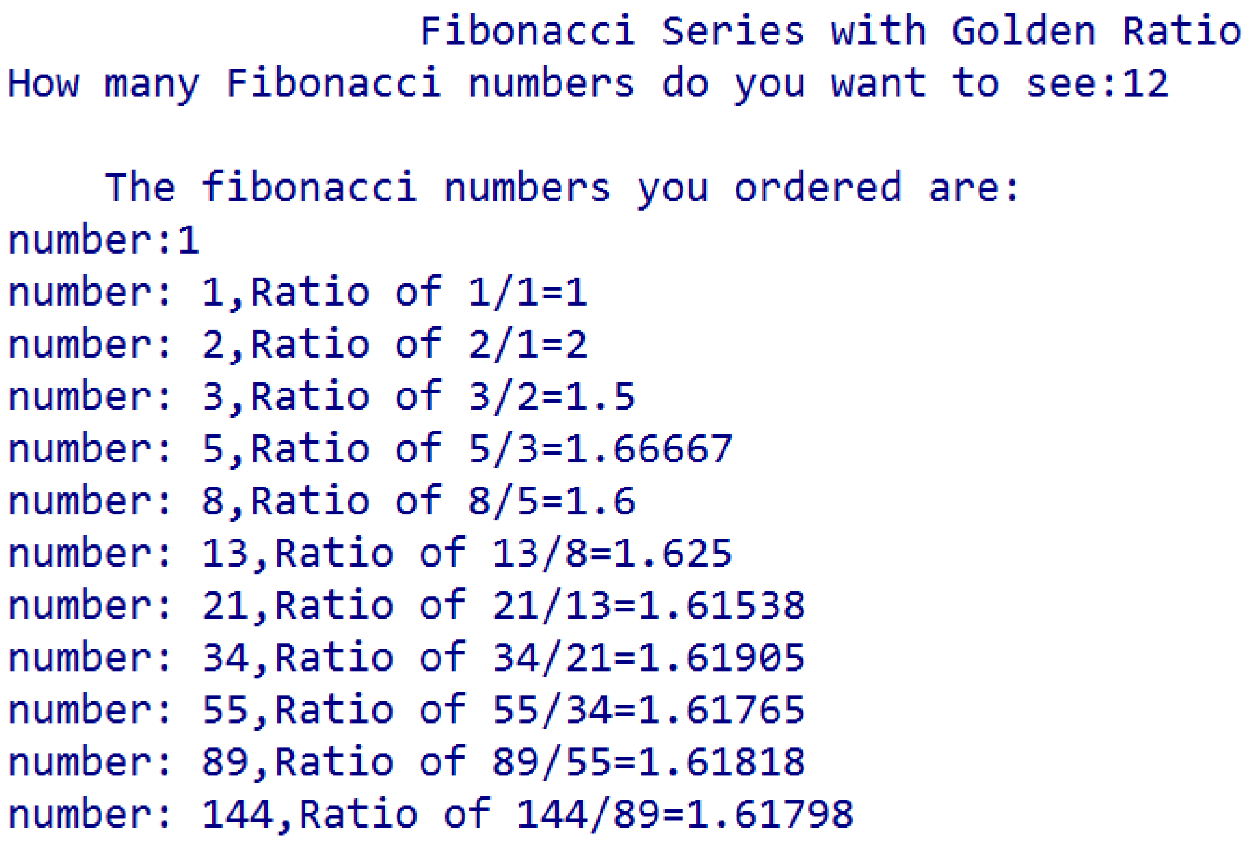
```
How many terms to generate : 12
                  Fibonacci Series
1 1 2 3 5 8 13 21 34 55
89 144
```

les_06_code_02.cpp

Generating Fibonacci Series and computing Golden ratio by dividing successive terms

```
#include<iostream>

using namespace std;
int main (void)
  {
  // variable definition
  int counter, loopruns;
  // initializing newterm and prevterm to 1
  //sum cantains the next number in the sequence
  int newterm=1,prevterm=1,sum;
  cout<<"\t\t Fibonacci Series with Golden Ratio";
  // Input
  cout<<"\nHow many Fibonacci numbers do you want to see:";
  cin>>loopruns;
  // display the first term
  cout<<"\n    The fibonacci numbers you ordered are:\nnumber:"<<prevterm;
  // for rest of the terms
  for(counter=1;counter<loopruns;counter++)
     {
     cout<<"\nnumber: "<<newterm;
     cout<<",Ratio of "<<newterm<<"/"<<prevterm<<"="<<((float)newterm/(float)prevterm);
     sum=prevterm+newterm;
     prevterm=newterm;
     newterm=sum;
     }

  return 0;
  }
```

Sample Output



```
                 Fibonacci Series with Golden Ratio
How many Fibonacci numbers do you want to see:12

    The fibonacci numbers you ordered are:
number:1
number: 1,Ratio of 1/1=1
number: 2,Ratio of 2/1=2
number: 3,Ratio of 3/2=1.5
number: 5,Ratio of 5/3=1.66667
number: 8,Ratio of 8/5=1.6
number: 13,Ratio of 13/8=1.625
number: 21,Ratio of 21/13=1.61538
number: 34,Ratio of 34/21=1.61905
number: 55,Ratio of 55/34=1.61765
number: 89,Ratio of 89/55=1.61818
number: 144,Ratio of 144/89=1.61798
```

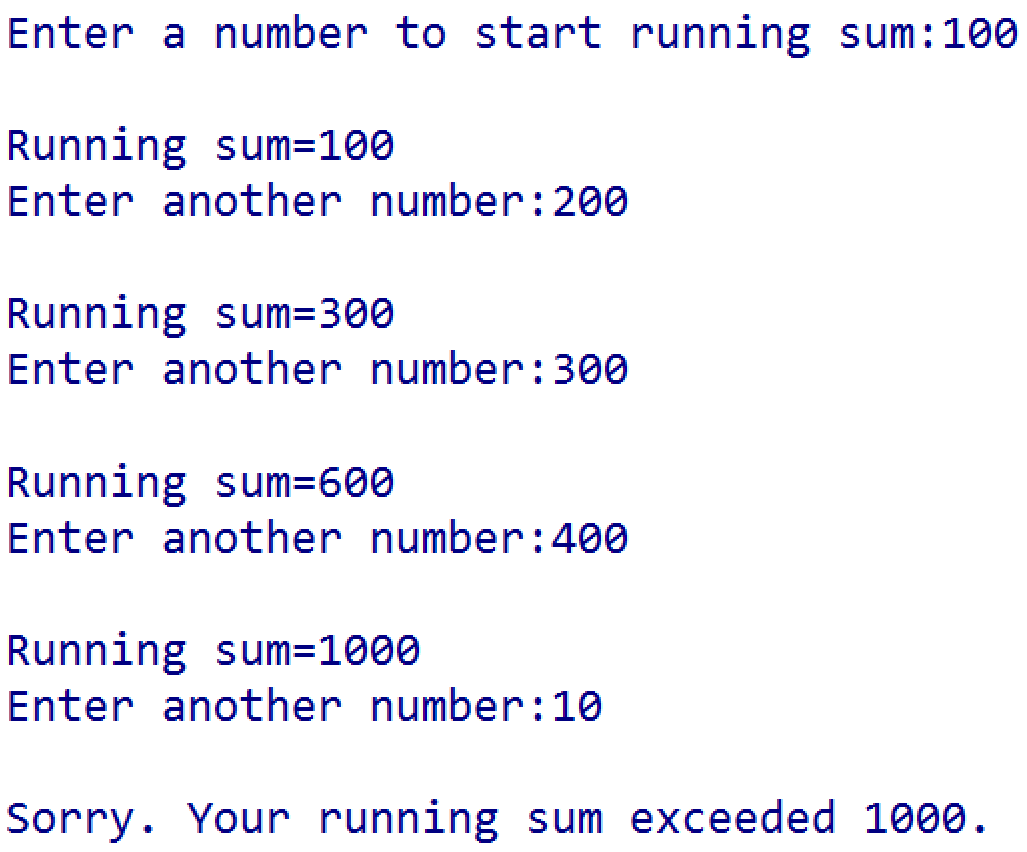
les_06_code_03.cpp

Running sum with while loop

```
#include<iostream>
using namespace std;
int main(void)
   {
   // variable definition
   int num,running_sum=0;

   // Taking inputs and doing processing before loop starts
   cout<<"\nEnter a number to start running sum:";
   cin>>num;
   running_sum=running_sum+num;


   // while loop to calculate the running sum of input
   while(running_sum<=1000) // loop terminates when running_sum
   exceeds
        {                                                  //n= 1000
        cout<<"\nRunning sum="<<running_sum;
        cout<<"\nEnter another number:";
```

```
            cin>>num;
            running_sum=running_sum+num;
            }

        // displaying why the program has stopped
        cout<<"\nSorry. Your running sum exceeded 1000.";
        return 0;
        }
```

Sample Output



```
Enter a number to start running sum:100

Running sum=100
Enter another number:200

Running sum=300
Enter another number:300

Running sum=600
Enter another number:400

Running sum=1000
Enter another number:10

Sorry. Your running sum exceeded 1000.
```

les_06_code_04.cpp

Digit counting (UIY Program)

```
#include<iostream>

using namespace std;
int main(void)
   {
   // variable definition
   int num,num_alias,digitcnt=0;
   // Taking input
   cout<<"\nEnter an integer to find "<<
   "the number of digits in it:";
   cin>>num;
   num_alias=num;     // saving num so that it can be used at the end
   // while loop to calculate the running sum of input
   while(num_alias>0) // loop terminates when num_alias becomes zero
       {
       num_alias=num_alias/10;
       digitcnt=digitcnt+1;
       }
   // displaying result
   cout<<"\nNo of digits in "<<num<<
   " = "<<digitcnt;

   return 0;
   }
// NOTE: The program is not valid for -ve numbers but can be made to work if
// we make num_alias +ve, if it is -ve.
```

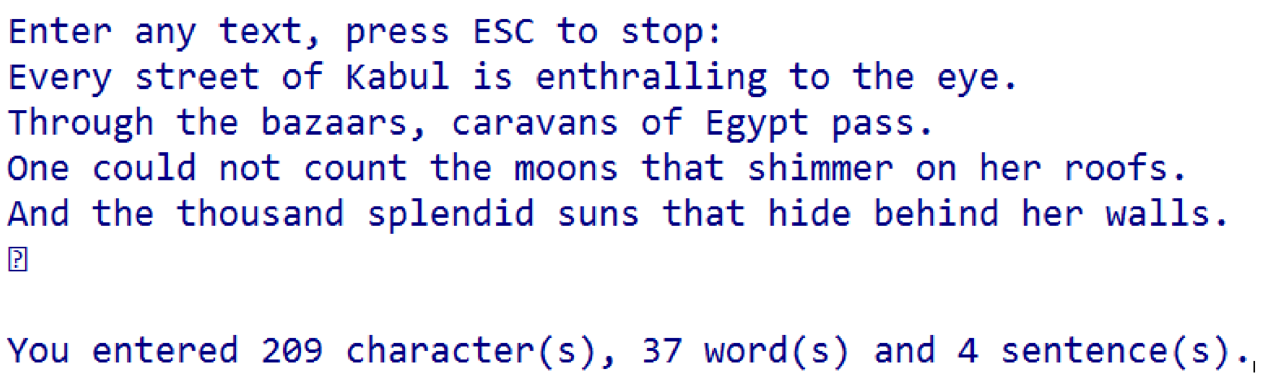
les_06_code_05.cpp

Character(s), word(s) and sentence(s) calculation

```
#include<iostream>
#include<conio.h>
using namespace std;
int main(void)
    {
    // variable definition
    // to count characters, words and sentences
    int ch_cnt=0,word_cnt=0,sentence_cnt=0;

    // Taking one character at a time
    // as input in chinput
    char chinput;
    cout<<"\nEnter any text,"<<
    " press ESC to stop:\n";
    chinput=getche(); // taking input in
    //loop control variable
    //(chinput) before
    // entering the loop



    while(chinput!=27)// loop terminates when
       {                  // chinput is ASCII 27,
    // which is for ESC key
       ch_cnt=ch_cnt+1;
```

```
        if(chinput==' ') // we could have used
        {                // ASCII value over here
        word_cnt=word_cnt+1;
        }

        // any condition where a sentence ends
        if((chinput=='.')||(chinput=='?'))
        {
        sentence_cnt=sentence_cnt+1;
        word_cnt=word_cnt+1;
        }

        if(chinput==13)      // The ASCII character for ENTER, its character mask is '\n'
        {
        cout<<endl;
        }

        chinput=getche();  // taking next character before the loop starts again
        }

     ch_cnt=ch_cnt-1;       // to subtract the last ESC key

     // displaying why the program has stopped
     cout<<"\n\nYou entered "<<ch_cnt<<
     " character(s), "<<word_cnt<<
     " word(s) and "<<sentence_cnt<<
     " sentence(s).";
     getch();
     return 0;
     }
```

Sample Output



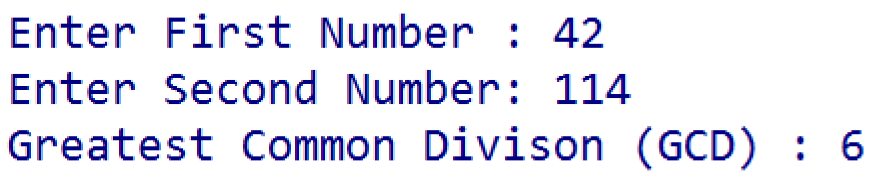
```
Enter any text, press ESC to stop:
Every street of Kabul is enthralling to the eye.
Through the bazaars, caravans of Egypt pass.
One could not count the moons that shimmer on her roofs.
And the thousand splendid suns that hide behind her walls.
?

You entered 209 character(s), 37 word(s) and 4 sentence(s).
```

les_06_code_06.cpp

Finding GCD , the Naive method

```cpp
#include<iostream>
using namespace std;

int main()
   {

   int first_number;
   cout<<"Enter First Number : ";cin>>first_number;

   int  second_number;
   cout<<"Enter Second Number: ";cin>>second_number;

   int  gcd;
   for(int i=1;i<=first_number&&i<=second_number;i++)
      {
      if(first_number%i==0 && second_number%i == 0 )
          {
          gcd=i;
          }
      }

   cout<<"Greatest Common Divison (GCD) : "<<gcd<<endl;
   return 0;
   }
```

Sample Output



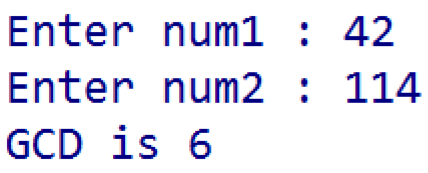
```
Enter First Number : 42
Enter Second Number: 114
Greatest Common Divison (GCD) : 6
```

les_06_code_07.cpp

Euclid Subtraction GCD Algorithm

```cpp
#include <iostream>

using namespace std;

int main()
    {

    int num1, num2;
    cout << "Enter num1 : ";
    cin >> num1;
cout << "Enter num2 : ";
cin >> num2;

while (num1 != num2 )
    {
    if (num1 > num2)
       {
       num1 = num1 - num2;
       }
    else
       {
       num2 = num2 - num1;
       }
    }

cout << "GCD is " << num1;
return 0;
}
```

Sample Output



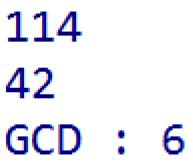
```
Enter num1 : 42
Enter num2 : 114
GCD is 6
```

les_06_code_08.cpp

Faster Euclid GCD

```
#include <iostream>

using namespace std;

int main()
   {
   int a, b;
   cin >> a >> b;
   while (b != 0)
       {
       int r = a%b;
       a = b;
       b = r;
       }
   cout <<"GCD : "<< a << endl;
   return 0;
   }
```

Sample Output



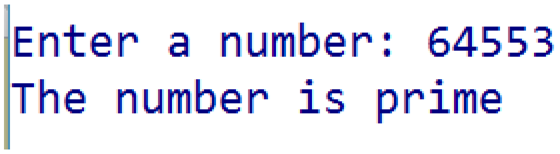
```
114
42
GCD : 6
```

les_06_code_10.cpp

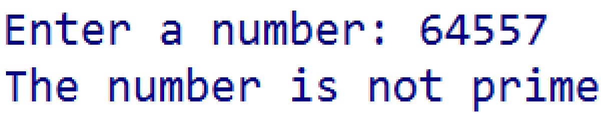
Prime Testing Naive Method

```
#include <iostream>
using namespace std;
int main()
{
int num;
cout << "Enter a number: ";
cin >> num;
for (int i = 2; i < num; i++)
   {
   if (num%i == 0)
       {
       cout << "The number is not prime" << endl;
       return 0;
       }
   }
cout << "The number is prime" << endl;
return 0;
}
```

Sample Outputs



```
Enter a number: 64553
The number is prime
```



```
Enter a number: 64557
The number is not prime
```

les_06_code_11.cpp

Prime Testing improved version

```
#include<iostream>
#include<conio2.h>
#include<cmath>
using namespace std;
```

```
int main (void)
{
int num,divisor,remainder;
cout<<"Enter a number:";
cin>>num;
// Some conditions to increase program speed
if((num>2) && (num%2==0))
{
cout<<"\n The no. is composite";
return 0;
}
if((num>3)&&(num%3==0))
{
cout<<"\n The no. is composite";
return 0;
}
// some defensive conditions
if(num==2)
{
cout<<"\nThe number is prime";
return 0;
}

//int count = 0;
// divisor is required to increment only till
//sqrt(num) and not num, prove it yourself.
for(divisor=2;divisor<=(int)sqrt((double)num);divisor++)
{
remainder=num%divisor;
if(remainder==0)
{
cout<<"\n    Composite";
break;
}

}
if(remainder!=0)
{
cout<<"\n    Prime";

}
return 0;
}
```

les_06_code_12.cpp

Square Root using Newton Raphson Method

```
#include<iostream>
#include<conio2.h>
#include<cmath>
using namespace std;
int main(void)
   {
   float N,low,high,root;
   cout<<"Enter a +ve number to calculate its square root:";
   cin>>N;
   if(N<0)
      {
      cout<<"\n Sorry, wrong input, Re-run the program";
      }
   else
      {
      root=N/2.0;
      while(fabs(((root*root)-N))>0.0001)
         {
         root=root-((root*root-N)/(2.0*root));
         }
      }
   cout<<"\n square root of "<<N<<" = "<<root;
   return 0;
   }
```

les_06_code_12.cpp

Square Root using Bisection Method

```
#include<iostream>
#include<conio2.h>
#include<cmath>
using namespace std;
int main(void)
   {
   float N,low,high,root;
   cout<<"Enter a +ve number to calculate its square root:";
   cin>>N;
   if(N<0)
      {
      cout<<"\nSorry, wrong input, Re-run the program";
      }
   else
      {
      low=0.0;
      if(N>1.0)// if number is greater than 1 (eg. 1.7)
```

```
         {
         high=N;
         }
       else  // if number is less than 1 (eg. 0.5)
         {
         high=1.0;
         }
       root=(low+high)/2.0;
         while(fabs(((root*root)-N))>0.0001)
         {
         if((root*root)<N)
           {
           low=root;
           }
         else
           {
           high=root;
           }
         root=(low+high)/2.0;
         }  // end of while()
       }  // end of else
     cout<<"\n square root of "<<N<<" = "<<root;
     return 0;
     }
```

les_06_code_14.cpp

Trapezoidal Integration

```
#include<iostream>
#include<conio2.h>
#include<cmath>
using namespace std;
int main(void)
   {
   double x, delta_x=0.001, xmin, xmax, sum=0.0, fx1, fx2;
   int counter, counter_max;
   cout<<"Enter the value of xmin in radian:";
   cin>>xmin;
   cout<<"Enter the value of xmax in radian:";
   cin>>xmax;
   counter_max=((xmax-xmin)/delta_x)+1;
   x=xmin;
   for(counter=1;counter<=counter_max;counter++)
      {
      fx1=cos(x);// integrating sin function
      fx2=cos(x+delta_x);
      sum=sum+(0.5*(fx1+fx2));
      x=x+delta_x;
      }
   sum=sum*delta_x;
   cout<<"integral= "<<sum;
   return 0;
   }
```