University of Burgundy

Software Engineering

Tutorial 1

# Lab Report-1

*Author:*
Hardiksinh Parmar

*Supervisor:*
Dr. Yohan Fougerolle

October 7, 2018

# 1 Hello World on different platforms

Example of Hello world on Qt.

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello World" << endl;
    return 0;
}
```

# 2 Variables

## 2.1 Local and Global Variables

1. Local variable: It is a variable that is given local scope which means it is limited to the block enclosed in braces where they are declared. For example, if they are declared at the beginning of the body of a function their scope is between its declaration point and the end of that function. We cannot access these variables out of this block, program cannot compiled and displays Error).

2. Global variable: It is a variable defined outside of all the functions or blocks and are accessible throughout the program.Once declared, these can be accessed by any function in the program, even inside functions, whenever it is after its declaration. Changing the value of global variable from one block will effect the value of the same variable in the other block of the same program. To access global variables declared outside of a file where we need them, use the keyword extern.

Example:

```
#include <iostream>
using namespace std;
int gVar1, gVar2;   // 'gVar1', 'gVar2' are global variables
int main ( int argc, char**argv)  //notice the arguments here
{
int num = 5;          //'num' is a Local variable here
cin >> gVar1 >> gVar2;
cout << gVar1-num << " " << gVar2 + num << endl;
return 0;
}
```

## 2.2 Min, Max and Mean of two variables

### 2.2.1 Program

```
//test.cpp file ------- definition of min, max, mean

#include <iostream>
using namespace std;
#include <test.h>

int min1 (int a,int b){
    if (a>b)
    { cout <<"the min of a and b is b: " << b << endl; }
     else{ cout <<"the min of a and b is a: " << a << endl;}
}

int max1 (int a,int b){
    if (a<b)
    { cout <<"the max of a and b is b: " << b << endl; }
     else{ cout <<"the max of a and b is a: " << a << endl; }
}

int mean1 (int a, int b){
    int r;
    r= (a+b)/2;
    return r;
}
```

```
//main.cpp ---- implementation of the min,max and mean
#include <iostream>
using namespace std;
#include <test.h>

int main()
{
    double y,z;
    float k;
     cout<<"enter a:";  cin >> y;
     cout<<"enter b:";  cin >> z;

  //----min/max/mean----
        k=min1(y,z);
        k=max1(y,z);
        k=mean1(y,z);
        cout <<"the mean of a and b is k: " << k << endl;

}

//test.h file ---- declaration of min,max and mean
int min1 (int ,int );
int max1 (int ,int );
int mean1 (int,int);

                          ****
```

### 2.2.2 Flow of execution

1. We read input variables y and z.And output variable is k.
2. Respective function call is established between the calling function and called functions sequentially. Input values are compared and min of two numbers are displayed.
3. Then max and mean is calculated.

# 3 Combination

## 3.1 Factorial

To find factorial of a number we multiply the number with the same number decreasing in the steps of 1 for every iteration.

factorial of n = n(n-1)(n-2)(n-3)...1

## 3.2 Combination without repetitions

To find nCr1 we use Factorial function multiple times.

$$\text{nCr1:} \quad nCr1 = \frac{n!}{r!\,(n-r)!}$$

## 3.3 Combination with repetitions

To find nCr2 we use Factorial function multiple times.

$$\text{nCr2:} \quad nCr2 = \frac{(n+r-1)!}{n!\,(n-1)!}$$

## 3.4 Permutations

To find nPr we use Factorial function multiple times.

$$\text{nPr:} \quad nPr = \frac{n!}{(n-r)!}$$

### 3.4.1 Program

```
//test.cpp file ------ definition of factorial1,nCr1,nCr2,nPr

#include <iostream>
using namespace std;
#include <test.h>

int  nCr1(int n,int r){
    return (factorial1(n)/(factorial1(r)*factorial1(n-r)));
}

int  nCr2(int n,int r){
    return (factorial1(n+r-1)/(factorial1(r)* factorial1(n-1)));
}

int  nPr(int n,int r){
    return (factorial1(n)/(factorial1(n-r)));
}
```

```
long factorial1(int n){
    int c;
    long result=1;
    for(c=1; c<=n;c++){
        result=result*c;
    }
     return result;
}


//main.cpp ---- implementation of factorial1,nCr1,nCr2,nPr
#include <iostream>
using namespace std;
#include <test.h>

int main()
{
    double y,z;
     cout<<"enter a:";  cin >> y;
     cout<<"enter b:";  cin >> z;

  //----factorial----
  if (y>=0) { cout <<"factorial of a: " << factorial1(y)<<endl ;}

  //----nCr----
      if (y>=z) {
      cout <<"nCr1 : with no repetitions" << nCr1(y,z) <<endl;
      cout <<"nCr2 : with repetitions" << nCr2(y,z) <<endl;}
 else cout <<"nCr doesnot exist because r > n :: b > a" << endl;

  //----nPr----
 if (y>=z) cout <<"nPr : " << nPr(y,z) <<endl;
 else cout <<"nPr doenot exist because r > n :: b > a " << endl;

}

//test.h file ---- declaration of factorial1,nCr1,nCr2,nPr
long factorial1(int);
int  nCr1(int ,int) ;
int  nCr2(int ,int );
int  nPr(int ,int );
```

****

### 3.4.2 Flow of execution

1. We read input variables y and z.
2. Respective function call is established between the calling function and called functions sequentially.
3. Input value y¿=0 factorial1 is called.
4. from definition, we take int c and initialize result=1. Using 'for' loop from c=1 to c¡=n (n is the input value) with c++ increment we find result.
5. We do these iteration till n. Finally, 'result' is printed as output, the value of FACTORIAL of the input number.
6. for nCr1, nCr2, nPr: the inputs must be y ¿ z. Respective function call is established between the calling function and called functions sequentially. factorial1 function is called in each of these 3 cases.
7. nCr1, nCr2, nPr values are printed.

# 4 List of Fibonacci numbers and its relation with Golden ratio

## 4.1 List of Fibonacci

$F_0 = 0$

$F_1 = 1$

$F_n = F_{n-1} + F_{n-2}$

Here for every increase in n we get a added sum and $F_n$ is updated.

### 4.1.1 Program

```
//test.cpp file ——— definition of fibanocci
```

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

int fibonacci(int n){
    if((n==1)||(n==0)) { return(n); }
    else return (fibonacci(n-1)+fibonacci(n-2));
}

//main.cpp file ----- implementation of fibanocci

#include <iostream>
using namespace std;
#include <test.h>


int main()
{
    double y;
     cout<<"enter a:";  cin >> y;

 //----Fibonacci----
cout << "User Entered :: 'y' numbers for Fibonacci series :: ";
        int i=0;
             while(i<y)
             {
                 cout<< " " << fibonacci(i);
                 i++;
             }
                cout << endl;
 }

//test.h file ---- declaration of fibanocci
int fibonacci(int n);
```

****

### 4.1.2 Flow of execution

1. We initialize variable i=0, read input y from user.
2. for every value i¡y while loop iterates. Respective function call is established between the calling function and called functions.
3. If Input values are 0 o 1.Result is 0 or 1. Else returns F(n-1)+F(n-2) this iterates for given input n times.

4. Then series formed is output.

## 4.2 Golden ratio

The ratio of F(n+1)/F(n) approaches a limit as n goes to infinity, which is approximately 1.61803. This number is known as the golden ratio.

### 4.2.1 Program

```
//test.cpp file ------ definition of GoldenRatio
#include <iostream>
using namespace std;
#include <test.h>
#include <cmath>
#define psi (1+sqrt(5))/2.0
void goldenRatio(){
    double ratio;
    int n(1);
    const double eps(1e-6);
    cout <<"iterating while || r-" <<psi <<"||" <<eps<<endl;
    do{
        ratio = fibonacci(n+1)/fibonacci(n);
        cout << ratio<< endl;
        n++;
    }while(abs(ratio-psi)>eps);

    cout<<"no of iterations:" <<n<<endl;
}

//main.cpp file ------ implementation of GoldenRatio

#include <iostream>
#include <test.h>
#include <string>
using namespace std;

int main()
{

cout<<"Computing ratio of consecutive no :" <<endl;  goldenRatio() ;
cout << endl;
}
```

//test.cpp file — declaration of GoldenRatio

void goldenRatio();
*****

### 4.2.2 Flow of execution

1. We call GoldenRatio function.
2. Respective function call is established between the calling function and called functions.
3. declaring a constant eps(1e-6);
4. calculate the ratio of the consecutive fibonacci numbers on while on condition abs(ratio-psi)¿eps.
5. Output is the series of ratio of consecutive fibonacci numbers

# 5 Pascal's Triangle

Inorder to compute first n rows of Pascal Triangle we have to find the binomial coefficient $n C_0$ $n C_1$ $n C_2$ $n C_3$ ..... $n C_n$ in n rows for each row from n to 0.

### 5.0.1 Program

```
//test.cpp file ---- definition of PascalsTriangle

#include <iostream>
using namespace std;
#include <test.h>
int  nCr1(int n,int r){
    return (factorial1(n)/(factorial1(r)*factorial1(n-r)));
}
long factorial1(int n){
    int c;
    long result=1;
    for(c=1; c<=n;c++){
        result=result*c;
    }
     return result;
}
```

```
void PascalsTriangle (int n){
    int i,j,x;
    for(i=0;i<n;i++)
    {
        x=1;
        for(j=0;j<=i;j++)
        {
            cout << x << " ";
            x = nCr1(i,j) * (i - j) / (j + 1);
        }
        cout << endl;

    }
}

//main.cpp file ----- implementation of PascalsTriangle

#include <iostream>
using namespace std;
#include <test.h>
int main()
{
    double y;
     cout<<"enter a:";  cin >> y;

       PascalsTriangle(y);
       cout << endl;
}

//test.h file ----- declaration of PascalsTriangle
void PascalsTriangle(int);

                        *******
```

### 5.0.2 Flow of execution

1. We read input rows y from user.
2. Respective function call is established between the calling function and called functions.
3. In subfunction i is rows and j is colums. x is final element. Iterations takesplace columnwise first and rowwise next.
4. As per input number of rows, series formed is output.

## 5.1 Working with Strings

+ Operator : concatenates the contents of a string to new string

== Operator : compares to strings and gives a boolean value

append() : add a substring the original string.

length() : returns length of the string.

insert() : used to insert the characters at the certain index postion

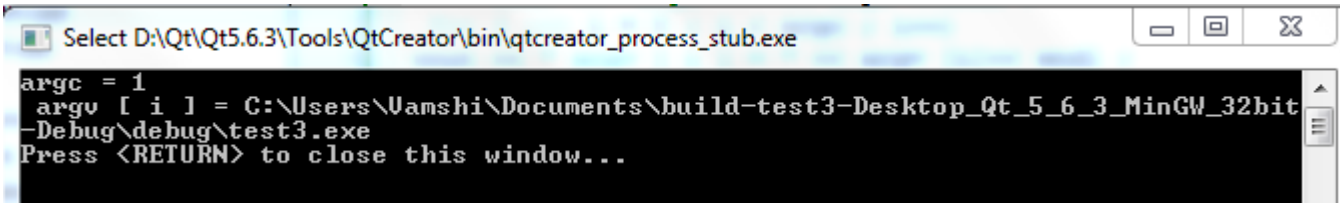
getline() : used when you have to take input strings which have spaces in between them.

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

int main()
{
//----- Strings ----------------
          string str1 = "Hello";
          string str2 = "World";
             string str3, str4;
             int  len ;

             // copy str1 into str3
             str3 = str1;
             cout << "str3 : " << str3 << endl;

             // concatenates str1 and str2
             str3 = str1 + str2;
             cout << "str3: str1 + str2 : " << str3 << endl;

             //append()str2 to str1
             str4 = str1.append(str2,3,2);
             cout << "str4 : " << str4 << endl;

             //insert()str2 to str1
             str2 = str3.insert(2,str4);
             cout << "str2 : " << str2 << endl;

             // total length of str3 after concatenation
             len = str3.length();
             cout << "str3.length() : " << len << endl;
```

```
        // == operator for comparing
        if (str2 == str4)
        {cout << "equi : str2 ==str4 "<< endl;}
  else { cout << " str2 and str4 are not equal"<< endl;}

}
```

### 5.1.1 Understanding int main (int argc, char** argv)

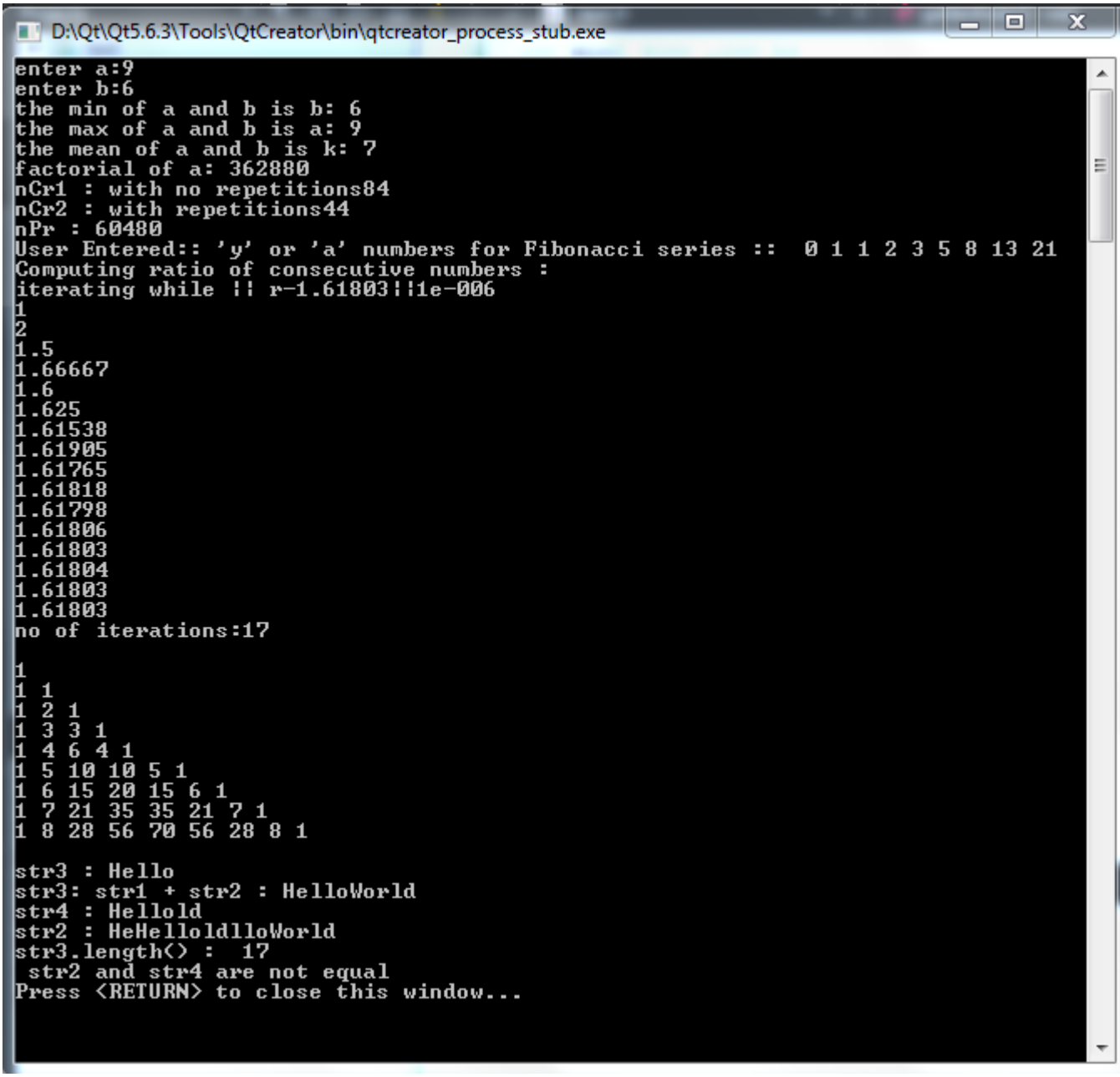
The output of the program ExampleMain.cpp is shown as a screenshot below. Parameter args corresponds to number of arguments passed from the command line and argv are the array of those arguments.




Output of the int main (int argc, char** argv)

We have 1 argument and that is path of the test.cpp file
**********

# 6 Output of the project is mentioned in the screenshot below:


D:\Qt\Qt5.6.3\Tools\QtCreator\bin\qtcreator_process_stub.exe
enter a:9
enter b:6
the min of a and b is b: 6
the max of a and b is a: 9
the mean of a and b is k: 7
factorial of a: 362880
nCr1 : with no repetitions84
nCr2 : with repetitions44
nPr : 60480
User Entered:: 'y' or 'a' numbers for Fibonacci series :: 0 1 1 2 3 5 8 13 21
Computing ratio of consecutive numbers :
iterating while || r-1.61803||1e-006
1
2
1.5
1.66667
1.6
1.625
1.61538
1.61905
1.61765
1.61818
1.61798
1.61806
1.61803
1.61804
1.61803
1.61803
no of iterations:17
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
str3 : Hello
str3: str1 + str2 : HelloWorld
str4 : Hellold
str2 : HeHelloldlloWorld
str3.length() : 17
str2 and str4 are not equal
Press <RETURN> to close this window...