**Experiment No. 06**

**Control Structures (Basic For Loop & While Loop)**

**OBJECTIVE**:

 To be able to understand the working of loops.

**Repetition Statements**

Repetition statements are called *loops*, and are used to repeat the same code multiple times in succession.The number of repetitions is based on criteria defined in the loop structure, usually a true/false expression.

The three loop structures in C++ are:

 while loops

 for loops

Three types of loops are not actually needed, but having the different forms is convenient

**while and do-while loops**: Format of while loop:

// while loop format while (expression)

{

statement1; statement2;

// ... statementN;

}

**How they work:**

The expression is a test condition that is evaluated to decide whether the loop should repeat or not.

 true means run the loop body again.

 false means quit.

The while and do/while loops both follow the same basic flowchart -- the only exception is that:

 In a while loop, the expression is tested first

 In a do/while loop, the loop "body" is executed first.

**For Loop:**

The **for** loop is most convenient with *counting loops* -- i.e. loops that are based on a counting variable, usually a known number of iterations

Remember that the statement can be a single statement or a compound statement (block), so an alternate way to write the format might be:

for (initialCondition; testExpression; iterativeStatement)

{

statement1; statement2;

// ... statementN;

}**How it works**

 The *initialCondition* runs once, at the start of the loop

 The *testExpression* is checked. (This is just like the expression in a while loop). If it's false, quit. If it's true, then:

 Run the loop body

 Run the *iterativeStatement*

 Go back to the *testExpression* step and repeat

**Special statements: break and continue**

 These statements can be used to alter the flow of control in loops, although they are not specifically *needed*. (Any loop can be made to exit by writing an appropriate *test expression*).

 **break**: This causes immediate exit from any loop (as well as from switch blocks)

 **continue**: When used in a loop, this statement causes the current loop iteration to end, but the loop then moves on to the next step.

o In a while or do-while loop, the rest of the loop body is skipped, and execution moves on to the *test condition*

o In a for loop, the rest of the loop body is skipped, and execution moves on to the *iterative statement*

**Exercise –2 (10 points)**

**For** loops can always be re-written as **while** loops, and vice-versa. Are the following two programs equivalent, and what is their output? Explain your answer, and run the programs to check.

Program (a):

#include <iostream> using namespace std;

int main()

{

for (int count=1; count <= 5 ; count++)

{

int count = 1;

cout << count << "\n";

}

return 0;

**code:**

#include <iostream>

using namespace std;

int main()

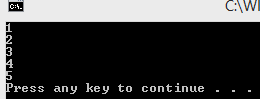
{

for(int count=1;count<=5;count++){

cout<<count<<endl;

}return 0;

}



#include <iostream> using namespace std;

int main()

{

int count = 1; while (count <= 5)

{

int count = 1;

cout << count << "\n"; count++;

}

return 0;

}

Code:

#include <iostream>

using namespace std;

int main()

{

int count =1;

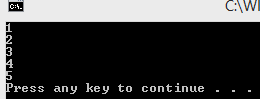
while(count<=5){

cout<<count<<endl;

count++;

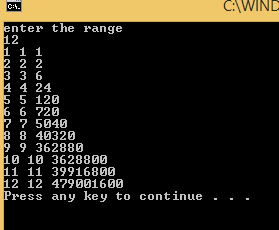
}return 0;

}



**Exercise –4 (10 points)**

Write a C ++ program that calculates the **sum** and **factorial** of the integers for each integer from 1 to n. Where n is a positive integer that you enter.

 (**Hint**: The positive integer is used to control a loop.)

#include <iostream>

using namespace std;

int main()

{

int sum,n,f=1;

cout<<"enter the range"<<endl;

cin>>n;

for(int i=1;i<=n;i++){

sum=0;

sum=sum+i;

f\*=i;

cout<<i<<" "<<sum<<" "<<f<<endl;

}

return 0;

}

**Post Lab:**

Determine the pattern followed by the series below:

1, 2, 3, 5, 8, 13, 21, 34, …

Write a program that takes a number ‘n’ and prints first ‘n’ numbers of this series. For example, if user enters 6, your program should display

1, 2, 3, 5, 8, 13

#include <iostream>

using namespace std;

int main()

{

int n,t1=0,t2=1,next=0;

cout<<"enter the number"<<endl;

cin>>n;

for(int i=1;i<=n;i++)

{

if(i==1){

cout<<" "<<t1;

}

if(i==2){

cout<<t2<<" ";}

next=t1+t2;

t1=t2;

t2=next;

cout<<next<<endl;

}

return 0;

}

