Lab 5
Iteration (Looping)
Dr. John Abraham

Suppose you want to display numbers 1 to 100 on the screen. Based on what we studies so far, you will have to write one hundred **cout** statements. If you think about it, all you are doing is **adding one to the previous number and displaying it** over and over again until 100 has been displayed. Let us write the steps to do it.

Number gets 1.
Display the number
Add one to it
Repeat these two statements.
Stop when 100 has been displayed.

Let us rework it with some numbers.
Number = 1
Do the following statements (in brackets) while number is less than or equal to 100.

```
{
   display number
    add one to number
}
```

Here are the steps:

1. **Initialize** the variable (this variable is also called the loop control variable or LCV, because this variable controls the loop). In this example the **LCV** is **number**.
2. Check for the condition to enter the loop. The condition should yield a **True** to **enter** the loop. If the condition yields a **false**, the loop is **not entered**.
3. Set up the **body** of the loop. You may have one or multiple things to do within the loop body. The body here appears within the brackets.
4. **Change** the value of the **LCV** within the body. In this example the number is changed by adding a one to it.

These steps will work in all programming languages. Let us write this program in c++. See program 4-1.

Program 4-1

```
/*****************************************
Display 1 to 100 on the screen
Teaching objective  - while loop
By Dr. John Abraham
Created for 1370 students
```

```
**************************************/

#include <iostream>
using namespace std;

int main()
{
  int number;
  number = 1;
  while (number <= 100)
    {
     cout << number << "   ";       //display number and put some spaces or use setw()
     number ++;                      //increment number by one
    }
  getchar();
  return (0);
}
```

Program Run 4-1

```
  1   2   3   4   5   6   7   8   9   10   11   12   13   14   15   16   17   18
 19   20   21   22   23   24   25   26   27   28   29   30   31   32   33   34
 35   36   37   38   39   40   41   42   43   44   45   46   47   48   49   50
 51   52   53   54   55   56   57   58   59   60   61   62   63   64   65   66
 67   68   69   70   71   72   73   74   75   76   77   78   79   80   81   82
 83   84   85   86   87   88   89   90   91   92   93   94   95   96   97   98
 99   100
```

This while loop is a **count controlled loop**, since we wanted to repeat the loop a certain number of times. The count controlled loop may also be referred to as a **step controlled loop**. We can write a **sentinel controlled loop** as well. A sentinel value is a value that belongs to a type but does not belong to the set you are working with. For example suppose you are entering names. Names belong to a type called string. When asked for the name what if you entered 'quit'?. 'Quit' also belongs to the string type, however, does not belong to the set of names. Parents do not name a child Quit! Let me illustrate it with Program 4-2.

Program 4-2

```
/**************************************
Accept and display names
Teaching objective  - while loop/sentinel controlled
By Dr. John Abraham
Created for 1370 students
**************************************/
```

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

int main()
{
  string name;
   cout << "Enter a name ";
   cin >> name;
   while (name !="quit")
     {
       cout << "Hello, " << name  <<"\n";
       cout << "Enter another name or type 'quit' to exit ";
       cin >> name ;
     }
   getchar();
   return (0);
}
```

Program Run 4-2

```
Enter a name Randy
Hello, Randy
Enter another name or type 'quit' to exit Sandy
Hello, Sandy
Enter another name or type 'quit' to exit James
Hello, James
Enter another name or type 'quit' to exit Roger
Hello, Roger
Enter another name or type 'quit' to exit Jill
Hello, Jill
Enter another name or type 'quit' to exit quit
```

What if you wanted to keep accepting names until ‘quit’ is entered, but do not want to accept more than 5 names? To do this we will make some modifications to the Program 4-2. First, we will change the test to include if the number of names is less than or equal to 5. Second we will increment a counter when a name is entered. See program 4-2A.

Program 4-2A

```
/*************************************************************
Accept and display names
Teaching objective  - while loop/sentinel  and count ontrolled
By Dr. John Abraham
```

```
Created for 1380 students
**********************************************************************/

#include <iostream>
#include <string>        //to handle string functions.
using namespace std;
int main()
{
  string name;
  int count;
   count = 1;
   cout << "Enter a name→ ";
   cin >> name;
   while (name !="quit" && count <= 5)
     {
       count ++;
       cout << "Hello, " << name <<"\n";
       cout << "Enter another name or type 'quit' to exit→ ";
       cin >> name ;

     }
   count --;  //actual number of names entered is count minus 1.
   cout << count << " names were entered " << "\n";
   getchar();
   return (0);
}
```

```
Enter a name--> James
Hello, James
Enter another name or type 'quit' to exit--> Mary
Hello, Mary
Enter another name or type 'quit' to exit--> Rose
Hello, Rose
Enter another name or type 'quit' to exit--> quit
3 names were entered



Enter a name--> Jack
Hello, Jack
Enter another name or type 'quit' to exit--> Jill
Hello, Jill
Enter another name or type 'quit' to exit--> Roger
```

```
Hello, Roger
Enter another name or type 'quit' to exit--> Ed
Hello, Ed
Enter another name or type 'quit' to exit--> Sandy
Hello, Sandy
Enter another name or type 'quit' to exit--> Reg
5 names were entered
```

While loop is a pre-test loop. It tests for the condition before entering the loop. We will be discussing a loop structure that checks for the condition at the bottom of the loop. If you want to use a pre-test loop controlled by a counter as seen in Program 4-1, you could either use a while loop or a **for loop**. **For loop** is variation of while loop specifically designed for count controlled loop. Program 4-3 is a modification of Program 4-1; the while loop has been changed to a for loop.

Program 4-3

```
/*****************************************
Display 1 to 100 on the screen
Teaching objective  - For loop
By Dr. John Abraham
Created for 1380 students
*****************************************/

#include <iostream>
using namespace std;

int main()
{
  int number;
  for (number =1; number <= 100; number++)
    {
     cout << number << "   ";       //display number and put some spaces
    }
  getchar();
  return (0);
}
```

Program Run 4-3

```
  1   2   3   4   5   6   7   8   9   10   11   12   13   14   15   16   17   18
 19   20   21   22   23   24   25   26   27   28   29   30   31   32   33   34
 35   36   37   38   39   40   41   42   43   44   45   46   47   48   49   50
 51   52   53   54   55   56   57   58   59   60   61   62   63   64   65   66
 67   68   69   70   71   72   73   74   75   76   77   78   79   80   81   82
 83   84   85   86   87   88   89   90   91   92   93   94   95   96   97   98
 99   100
```

As you can see, the Program Run 4-3 looks identical to Program Run 4-1. Let us understand this statement: for (number =1; number <= 100; number++). The values of the variable number are given, the initial value, loop execution condition, and the step value. In Program 4-3A, we will change all three values and see how the execution changes.

Program 4-3A

```
/*************************************************************
Display all even numbers beginning with 2 and ending with 100
Teaching objective  - For loop
By Dr. John Abraham
Created for 1380 students
*************************************************************/

#include <iostream>

int main()
{
  int number;
  for (number =2; number <= 100; number=number+2)
    {
     cout << number << "   ";       //display number and put some spaces
    }
  getchar();
  return (0);
}
```

Program Run 4-3A

```
  2   4   6   8   10   12   14   16   18   20   22   24   26   28   30   32   34
 36   38   40   42   44   46   48   50   52   54   56   58   60   62   64   66
 68   70   72   74   76   78   80   82   84   86   88   90   92   94   96   98
```

We could modify this program to display the number backwards. See Program 4-3B. Note how the initial variable, loop execution condition, and the step values are changed.

Program 4-3B

```
/*************************************************************
Display all even numbers backward beginning with 100 and ending with 2
Teaching objective  - For loop
By Dr. John Abraham
Created for 1380 students
*************************************************************/

#include <iostream>
```

```
using namespace std;

int main()
{
  int number;
  for (number =100; number >= 1; number=number-2)
    {
     cout << number << "   ";       //display number and put some spaces
    }
  getchar();
  return (0);
}
```

Program Run 4-3B

```
100  98  96  94  92  90  88  86  84  82  80  78  76  74  72  70
 68  66  64  62  60  58  56  54  52  50  48  46  44  42  40  38
 36  34  32  30  28  26  24  22  20  18  16  14  12  10  8  6  4
  2
```

Suppose you want to find the average grades for 20 students in a class on a quiz. You could either use the while loop or the for loop. However, if you want to write this program for a more general situation in which the number of students vary in different classes, you will have to use a **sentinel controlled while loop**. The sentinel value could be –99 for the grade input. Program 4-4 illustrates this.

Program 4-4

```
/************************************************************
Find average of a set of exam scores
Teaching objective  - Sentinel controlled while loop
By Dr. John Abraham
Created for 1380 students
************************************************************/

#include <iostream>
using namespace std;

int main()
{
  int grade, total, count, average; //count will keep track of number of grades
  count = 1;  // initialize both count and total
  total = 0;
  cout << "Enter a grade or -99 to quit--> ";
  cin >> grade;
  while (grade != -99)
    {
```

```
    total += grade;
    count ++;
    cout << "Enter a grade or -99 to quit--> ";
    cin >> grade;
   }
  count--;
  average = total/count;
  cout << "Sum of " << count << " grades entered ---> " << total << "\n";
  cout << "The average grade is ---> " << average;

  getchar();
  return (0);
}
```

Program Run 4-4

```
Enter a grade or -99 to quit--> 100
Enter a grade or -99 to quit--> 88
Enter a grade or -99 to quit--> 71
Enter a grade or -99 to quit--> 56
Enter a grade or -99 to quit--> 88
Enter a grade or -99 to quit--> 65
Enter a grade or -99 to quit--> -99
Sum of 6 grades entered ---> 468
The average grade is ---> 78
```

We keep a running total by initializing total to 0 and adding all grades to the previous total. We do not want to add the –99 to the total nor should it count as a valid grade. The loop continuation condition clearly states to exit the loop if a –99 entered, therefore –99 is not added to the total. However, the count was already incremented, which should be negated. This is what we do with the statement: count--. This is a very important concept, we will be using this quite a bit in many of the future programs.

The last looping structure I want to mention is the do..while loop. Do while loop is a post test loop; the condition is tested at the bottom of the loop. Therefore, a do while loop will be executed at least once. Program 4-5 is a modification of the first program in this chapter.

Program 4-4

```
/*****************************************
Display 1 to 100 on the screen
Teaching objective  - do while loop
By Dr. John Abraham
Created for 1380 students
*****************************************/
```

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

int main()
{
  int number;
  number = 1;
  do
    {
     cout << number << "   ";
     number ++;
    }
  while (number <=100);
  getchar();
  return (0);
}
```

Program run 4-4

```
1   2   3   4   5   6   7   8   9   10   11   12   13   14   15   16   17   18
 19   20   21   22   23   24   25   26   27   28   29   30   31   32   33   34
 35   36   37   38   39   40   41   42   43   44   45   46   47   48   49   50
 51   52   53   54   55   56   57   58   59   60   61   62   63   64   65   66
 67   68   69   70   71   72   73   74   75   76   77   78   79   80   81   82
 83   84   85   86   87   88   89   90   91   92   93   94   95   96   97   98
 99   100
```

Assignment:

Write a program to calculate gross pay for a pay period of no more than 14 days (2 weeks). Hours worked greater than 80 will be paid over time. Here are two examples of program runs.

Program Run 1

```
Enter name of the employee --> Mary
Enter pay rate for Mary--> 16.25
Enter hours worked for each day up to a maximum of 14 days.
Enter 0 when finished--> 8
Enter hours worked or 0 to quit --> 8
Enter hours worked or 0 to quit --> 7
Enter hours worked or 0 to quit --> 9
Enter hours worked or 0 to quit --> 10
Enter hours worked or 0 to quit --> 8
Enter hours worked or 0 to quit --> 4
Enter hours worked or 0 to quit --> 10
Enter hours worked or 0 to quit --> 0
```

```
Name of the Emplyee:     Mary
Total hours worked:      64
Total days worked :      8
Total Regular Hours :    64
Total OT Hours:          0
Regular Pay :            1040.00
Overtime Pay:            0.00
Gross pay :              1040.00
```

Program Run 2

```
Enter name of the employee --> John
Enter pay rate for John--> 10
Enter hours worked for each day up to a maximum of 14 days.
Enter 0 when finished--> 8
Enter hours worked or 0 to quit --> 8
Enter hours worked or 0 to quit --> 8
Enter hours worked or 0 to quit --> 8
Enter hours worked or 0 to quit --> 8
Enter hours worked or 0 to quit --> 8
Enter hours worked or 0 to quit --> 8
Enter hours worked or 0 to quit --> 8
Enter hours worked or 0 to quit --> 8
Enter hours worked or 0 to quit --> 8
Enter hours worked or 0 to quit --> 8
Enter hours worked or 0 to quit --> 8
Enter hours worked or 0 to quit --> 8
Enter hours worked or 0 to quit --> 8

Name of the Emplyee:     John
Total hours worked:      112
Total days worked :      14
Total Regular Hours :    80
Total OT Hours:          32
Regular Pay :            800.00
Overtime Pay:            480.00
Gross pay :              1280.00
```