Lab 7
Post-Test and Nested Loops
Dr. John P. Abraham

So far we learned pre-test loops such as the general while loop, counter controlled loops specifically for loop, sentinel controlled loops and flag controlled loops. In all these loops we initialized the loop control variable(s) and tested the condition before entering the loop. For instance, data-validation would require the data to be read at least once and then test for its validity. There are special situations were the loop must be executed at least once before the testing is done. Two examples come to mind, a circular queue, and sorting. In such cases the post test looping construct is used. There are also situations where it makes better reading when a post test loop is used. In such cases post test construct is not required.

Do-While

In C++ the post-test construct is implemented using do-while. In case of data-validation a do-while loop would look some thing like:

```
Do
  Prompt and read the data
While the data is invalid ‘keep reading the data if the data entered is not valid.
```

Program 7-1

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

int main()
{

	int monthNum;

	do
	{
	  cout <<"Enter the month (a number ranging from 1 to 12) -- >";
	  cin >> monthNum;
	  if (monthNum <1 || monthNum >12) cout <<"Error! month not in 
acceptable range!\n";
	}
	 while (monthNum <1 || monthNum >12);


	return 0;
}
```

```
Enter the month (a number ranging from 1 to 12) -- >20
Error! month not in acceptable range!
Enter the month (a number ranging from 1 to 12) -- >13
```

```
Error! month not in acceptable range!
Enter the month (a number ranging from 1 to 12) -- >0
Error! month not in acceptable range!
Enter the month (a number ranging from 1 to 12) -- >
```

Nested Loops

Here is a loop to display a multiplication table for the number 2.

Program 7-2

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

int main()
{

      int tableFor =2;
      int multiplier=1;

      while (multiplier <=12)
      {
            cout << setw(3)<<tableFor << " x " <<setw(3)<< multiplier
<< " = " << setw(4)<<tableFor * multiplier <<endl;
            multiplier ++;
      }

      getchar();
      return 0;
}
```

Program Run 7-2

```
 2 x   1 =    2
 2 x   2 =    4
 2 x   3 =    6
 2 x   4 =    8
 2 x   5 =   10
 2 x   6 =   12
 2 x   7 =   14
 2 x   8 =   16
 2 x   9 =   18
 2 x  10 =   20
 2 x  11 =   22
 2 x  12 =   24
```

Now let us add an outer loop to do the table for every number from 2 to 10.

Program 7-3

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

int main()
{

      int tableFor =2;
      int multiplier=1;
while (tableFor <= 10)
{
            while (multiplier <=12)
            {
                  cout << setw(3)<<tableFor << " x " <<setw(3)<<
multiplier << " = " << setw(4)<<tableFor * multiplier <<endl;
                  multiplier ++;
            }
  cout << endl;
  multiplier =1; //reset it for the next loop
  tableFor++;
}
      getchar();
      return 0;
}
```

Program Run 7-3

```
  2 x   1 =    2
  2 x   2 =    4
  2 x   3 =    6
  2 x   4 =    8
  2 x   5 =   10
  2 x   6 =   12
  2 x   7 =   14
  2 x   8 =   16
  2 x   9 =   18
  2 x  10 =   20
  2 x  11 =   22
  2 x  12 =   24

  3 x   1 =    3
  3 x   2 =    6
  3 x   3 =    9
  3 x   4 =   12
  3 x   5 =   15
  3 x   6 =   18
  3 x   7 =   21
  3 x   8 =   24
  3 x   9 =   27
  3 x  10 =   30
  3 x  11 =   33
  3 x  12 =   36
```

4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
4 x 11 = 44
4 x 12 = 48

5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
5 x 11 = 55
5 x 12 = 60

6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
6 x 11 = 66
6 x 12 = 72

7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42

7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
7 x 11 = 77
7 x 12 = 84

8 x 1 = 8
8 x 2 = 16
8 x 3 = 24
8 x 4 = 32
8 x 5 = 40
8 x 6 = 48
8 x 7 = 56
8 x 8 = 64
8 x 9 = 72
8 x 10 = 80
8 x 11 = 88
8 x 12 = 96

9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90
9 x 11 = 99
9 x 12 = 108

10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100
10 x 11 = 110
10 x 12 = 120

Here is the same program using a for-loop. You can see how simple it is to implement this using a for loop, because they both (inner and outer) are counter-controlled.

Program 7-4

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

int main()
{

      int tableFor;
      int multiplier;

      for (tableFor = 2;tableFor <=10;tableFor++)
{
            for (multiplier =1; multiplier <= 12; multiplier++)

                  cout << setw(3)<<tableFor << " x " <<setw(3)<<
multiplier << " = " << setw(4)<<tableFor * multiplier <<endl;


  cout << endl;


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

Assignment:

1. Write a data-validation loop (do-while) to test for the age of a person.

2. Re-write the program from Lab 5 (where you were to calculate gross pay for an employee as given below) to calculate gross pay for all employees. Exit the outer loop when "quit" is entered for the name input.

> 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
```