East West University
Department of Computer Science & Engineering
Plot No-A/2, Main Road, Jahurul Islam City, Aftabnagar, Dhaka-1219

Lab Manual
Course : CSE -105
Credit Title : Structured Programming
Instructor : Dr. Maheen Islam, Associate Professor, CSE Department

Lab-4: Basic Program Control: Loops.

Loop control statements are one of the most important parts of structured programming. The basis of program control starts with loop. In this lab we will go to the detail of C's while loop and do while loop. A while statement is a C looping statement. It allows repeated execution of a statement or block of statements as long as the condition remains true (nonzero). If the condition is not true when the while command is first executed, the statement(s) is never executed. The form while Statement is following:

```
while (condition)
   statement
```

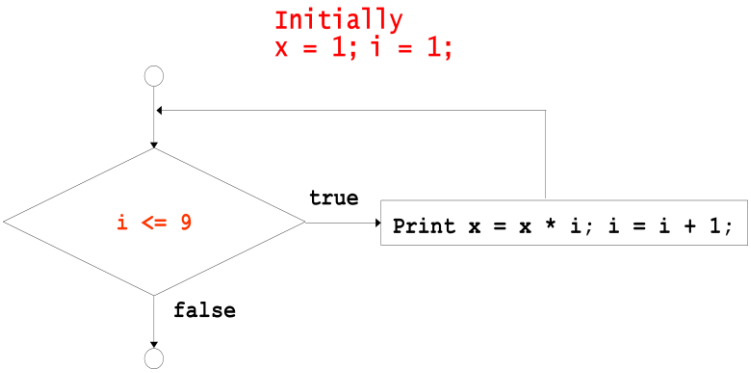
Here, ***condition*** is any valid C expression, usually a relational expression. When condition evaluates to false (zero), the while statement terminates, and execution passes to the statement following first statement; otherwise, the first C statement in the statements is executed. Statement is the C statement that is executed as long as condition remains true. Note that, the statement is singular here too, not plural alike ***if-else-if.*** To make a ***while*** statement control two or more statements, generally we use a compound statement. Compound statement has the form:

```
{
Statement_1;
Statement_2;
  …      …       …
Statement_n;
}
```

Placing braces around a group of statements forces the compiler to treat it as a single statement, i.e., for the following C code, all n statements will be executed while the expression is true.

```
while(condition)
        {
                Statement_1;
                Statement_2;
                  …      …       …
                 Statement_n;
        }
```

Exercise: 1 Consider the following flowchart,



As you can imagine, this flow chart continuously print the value of ***x*** and ***i*** until ***i*** is becomes greater than 9. In other words, it allows repeated execution of a statements ***x=x*i,*** and ***i=i+1*** as long as the (***i<=9)*** remains true. The C code for this problem is as follows,

```
int main()
{
int x = 1, i =1;
while (i <= 9)
 {
   x = x*i;
   i = i+1;
   printf("\nThe value of x and i is %d, %d, respectively\n", x,i );
}
   return 0;
}
```

Output:

```
The value of x and i is 1, 2, respectively

The value of x and i is 2, 3, respectively

The value of x and i is 6, 4, respectively

The value of x and i is 24, 5, respectively

The value of x and i is 120, 6, respectively

The value of x and i is 720, 7, respectively

The value of x and i is 5040, 8, respectively

The value of x and i is 40320, 9, respectively

The value of x and i is 362880, 10, respectively
Press any key to continue_
```

Your 1st task is to write the above code and match your output will given one.
Can you code program this without loop? Let's try it for 10 Mins,

Exercise: 2 In this program we want to take input from command line until we get one greater than 99, i.e., our program should repeatedly take inputs form keyboard and print it values. However, if the value of the input is greater than 99 we stop. Write the following code and check you output.

```
int nbr=0;
while (nbr <= 99)
{
   scanf("%d", &nbr );
   printf("You Enter -> %d\n", nbr );
}
```

Output:

```
1
You Enter -> 1
2
You Enter -> 2
20
You Enter -> 20
100
You Enter -> 100
Press any key to continue
```

```
120
You Enter -> 120
Press any key to continue_
```

Match your output with the given two on right. Next, your task to draw a Flowchart (in a paper) of the above program. Can you solve this problem without loop. The ans is _____.

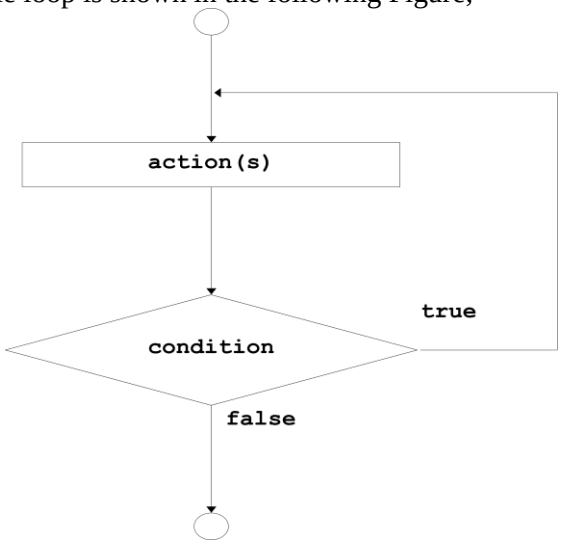
Exercise: 3 C's second loop construct is the do...while loop, which executes a block of statements as long as a specified condition is true. The do...while loop tests the condition at the end of the loop rather than at the beginning, as is done by the while loop. The structure of the do...while loop is as follows:

```
do
   statement
while (condition);
```

Here, condition is any C expression, and statement is a single or compound C statement. When program execution reaches a do...while statement, the following events occur:

1. The statements in statement are executed.
2. condition is evaluated.
3. If it's true, execution returns to step 1. If it's false, the loop terminates.

The operation of a do...while loop is shown in the following Figure,



The statements associated with a do...while loop are always executed at least once. This is because the test condition is evaluated at the end, instead of the beginning, of the loop. In contrast, while loops evaluate the test condition at the start of the loop, so the associated statements are not executed at all if the test condition is initially false. The do...while loop is used less frequently than while loops. It is most appropriate when the statement(s) associated with the loop must be executed at least once.

Now, your task is to Code the **Exercise 1 and 2 with do while loop**. The output should be identical as shown in that exercise.

Exercise: 4 Write a program that will take an integer number `n` as input and print the result of the following series:

a. 1+2+3+. . .+n
b. 1+1/2+1/3+. . . + n
c. 1.2+3.4+5.6+. . .+(n-1).n
d. 1.2.3+2.3.4+3.4.5+. . .+n.(n+1).(n+2)
e. 1+2+4+7+11+ . . . (up to `n` numbers)

Sample Input	Sample Output
4	a) 10 b) 2.08 c) 14 d) 210 e) 14

Exercise 5: Write a program that reads two positive integers corresponding to two year values, ensures that the first year value is less than the second, and then determines and outputs all year values for leap years. A leap year is one that can be evenly divided by 4, unless it is a centennial, in which case it must be evenly divided by 400. For example, 1600 and 1992 are leap years, whereas 1700 and 1998 are not. Your program should output all the leap years between this two input years.

Exercise 6: Here, we will solve the Grading problem (LAB 2) in a more specific way. Now, the grading of each course is based on the following weighted scale:

- Term 1 – 20%
- Term 2 – 20%
- Final – 30%
- Attendance – 10%
- Class Tests – 20%

The letter grades are given based on the total marks obtained by a student and is shown below:

- A >= 90%
- B >= 80% & < 90%
- C >= 70% & < 80%
- D >= 60% & < 70%
- F < 60%

Term 1 and Term 2 exams are out of 20 each, Final is out of 30 and Attendance given is out of 10. Three class tests are taken per semester and the average of best two is counted towards the final grade. Every class test is out of 20. Example: Say Tara obtained marks of 15, 18, 25 and 8 in Term 1, Term 2, Final and Attendance respectively. Her 3 class test marks are 15, 12 and 17. Since average of best 2

will be counted, her class test mark will be equal to (15 + 17) / 2 = 16. Therefore, total marks = 15 + 18 + 25 + 8 + 16 = 82 and she will be getting a B.

Input: The first line of input is an integer T(T<100) that indicates the number of test cases. Each case contains 7 integers on a line in the order Term1 Term2 Final Attendance Class_Test1 Class_Test2 Class_Test3. All these integers will be in the range [0, total marks possible for that test].

Output: For each case, output the case number first followed by the letter grade {A B C D F}. Follow the sample for exact format.

Sample input	Sample Output
3 15 18 25 8 15 17 12 20 20 30 10 20 20 20 20 20 30 10 18 0 0	Case 1: B Case 2: A Case 3: B

Home Works :

1. Write a program that will take a number `n` as input and print `n` lines of output according to the following sample output.

Sample Input	Sample Output
3	1 2 3 4 5 6
5	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

2. Write a program to read an integer number `n` and print the value of 3^n(using loop).

Sample Input	Sample Output
2	9
4	81
0	1