Western Engineering

ES1036: Programming Fundamentals – **Winter 2018**

Lab04: For loops and Introduction to Functions in C++

Faculty of Engineering, Western University

In order to finish this lab on time, you must start ahead of the lab session. You must try to complete the lab early and contact your TA if you require any help. The lab is due for submission on OWL at the end of your lab section.

A. Background

In this lab, you will explore the for loop structure and begin developing modular code with functions. You will open the existing solution *"ES1036Labs"* and create a new project named *"Lab04"*.

B. How to Get Full Credit for this Lab

You will get credit for your lab when you demonstrate it to your TA and get approval. During the code-demonstration, your TA will ask questions about your code or request you to perform some modifications. When you finish working on the lab, you will be required to upload your .cpp files (**no other file types will be accepted**) to OWL after changing the file name as indicated below. A zero grade will be awarded for a **missing OWL submission** or **missing the lab demonstration**.

- **Submission Instructions**

Number of files	:	**3**
Files to be submitted	:	.cpp source files for **E1, E2** and **E3**
Naming requirements	:	your_Western_user_name_lab04_q1.cpp,
		your_Western_user_name_lab04_q2.cpp,
		your_Western_user_name_lab04_q3.cpp.

C. Review

It is important that you review and understand the sections referenced below. Studying these sections will help you complete the lab.

Switch statement: see lecture handout Unit 4: Slides: 38 through 46. (Example on slide 44)

While loop: see lecture handout Unit 4: Slides: 54 through 69.

Do-While loop: see lecture handout Unit 4: Slides: 70 through 75. (Example on slide 72)

For loop: see lecture handout Unit 4: Slides: 77 through 80. (Examples on slides 81 - 86)

Functions: see lecture handout 5: Slides 1 through 25. (Examples on slides 35 - 39)

In this lab, you will implement for loops. Consider the samples below that use these loop structures:

```
int main() {
   //sum integers from 1 to 10
   int sum =0;
   for(int i=1; i<=10; i++)
      sum = sum + i;

   //sum odd integers from 1 to 10
   int sum =0;
      for(int i=1; i<=10; i+=2)
      sum = sum + i;
}
```

In the above program, the first *for* loop is used to sum the integers 1 through 10. The variable sum is declared outside of the *for* statement, however, the counter variable *i* is declared inside the statement. This means that the variable *i* will be confined to the *scope* (more on this term in later labs) of this statement. It can be seen that both loops declare the variable *i* but they are not conflicting. Ask your TA for clarification if there is any confusion with using the same variable name in different for loop structures.

Note how the code below using a while loop can be simplified and compacted using a for loop:

```
int num = 0;
while (num < 400) {
   cout << num << endl;
   num++;
}
```

```
for (int num = 0; num < 400; num++) {
   cout << num << endl;
}
```

Also note that fields in the for loop statement can be left blank by only adding a semicolon in place of a statement clause. For example, the last modifier statement in the for loop can be left blank if no modification of the declared variable is desired, as shown below.

```
for (char choice = 'y'; choice != 'n';) {
   cout << "Contine? (y/n)" << endl;
   cin >> choice;
}
```

D. Good Programming Practice

- If you cannot determine the cause of an error during runtime, try to use the **debugger**. The debugger has not been taught yet but if you require any assistance, your TA will be able to inform you how to begin debugging. Learning this tool will be of great value to you in the course and in the future.
- It is **good practice** to enclose your conditional logic using braces. This increases the readability of your program and helps troubleshoot logical errors in complicated programs.
- When declaring variables, assigning them an **initial value** can help prevent runtime errors. Take caution as this can also induce logical errors if you forget to manipulate the variables to desired values before usage.

E. Lab Questions

You must complete the all three of the following questions to get full marks.

Note that functions have not been covered yet in lectures. Please proceed with completing E.1 and E.2 and the lecture material required to finish E.3 will be covered this upcoming week.

E.1 Inverse Tangent

Requirements:

Write a program that will approximate the inverse tangent of a number input by the user. **You must not include the <cmath> library.**

Specifications:

1. The user will be asked to enter a real number value in radians.
2. Use the following series to approximate the inverse tangent of *x* between the range of -1 to 1 inclusive*:*

$$\operatorname{atan}(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} - \cdots = \sum_{n=0}^{N} \frac{(-1)^n x^{2n+1}}{2n+1}$$

   Note that an exponent indicates how many times a variable is multiplied by itself. Use an additional nested loop to make this calculation. You must choose a value for **N,** which represents the number of iterations that the approximation will run for. The larger this value, the more accurate your approximation will be but a large value will slow down computation significantly. Experiment with different values of **N** and check the computation accuracy.
3. Validate the input to ensure that the radians entered are between -1 and 1 inclusive.

Sample Output:

```
------------------------------
Student Name
Lab #4
Date: February 25/2018
Question #1
------------------------------

Enter a real number in radians to compute the inverse tangent (-1 to 1): 0.3
The result is 0.291457
Would you like to continue? (y/n): y
Enter a real number in radians to compute the inverse tangent (-1 to 1): 1
The result is 0.785897
Would you like to continue? (y/n): y
Enter a real number in radians to compute the inverse tangent (-1 to 1): -1
The result is -0.785897
Would you like to continue? (y/n): y
Enter a real number in radians to compute the inverse tangent (-1 to 1): 2
Enter a real number in radians to compute the inverse tangent (-1 to 1): -34
Enter a real number in radians to compute the inverse tangent (-1 to 1): 0.7836
The result is 0.664661
Would you like to continue? (y/n): n
------------------------------
Goodbye!
------------------------------

Program ended with exit code: 0
```

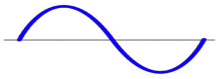
E.2 Printing a Full-wave Rectified Sine Curve

Requirements:

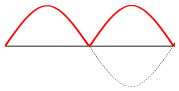
Your program will print a vertical, full-wave rectified sine curve on the console using "*" characters. **You must not include the <cmath> library.**

Specifications:

1. The program will ask the user to enter a real number value indicating how many sine wave cycles they wish to print to the screen. The reason for allowing a decimal number to be entered is so that a partial cycle can be printed.
   Remember that a single cycle of a sine wave can be shown as:

   And a single cycle of a full-wave rectified sine wave can be shown as:

   For simplicity, we are going to print this wave vertically on the console.
2. Use a *for* loop, or set of nested *for* loops to print an appropriate number of "*" characters to the console to represent the rectified sine wave as shown in the sample output section below. For more details on printing characters to the console with for loops, see lecture unit 4, slides 99-100.
   Hint*:* To find how many characters should be printed on each console line, first calculate the sine value for an increasing number of degrees. You could use a loop structure with the number of degrees being incremented on each iteration up to the number of desired degrees (number of cycles multiplied by 360 degrees/cycle). Inside the loop, you should convert this value in degrees to radians using the formula:

   $$radians = degrees * \frac{\pi}{180.0}$$

   Once the sine of this converted number is found as a real number, you can multiply this by a fixed integer (Ex. 20) representing the maximum number of "*"'s to be printed to the console.
3. Since using the ***cmath*** library is not allowed for this question, you will have to write your own computation of a sine wave. The formula listed for approximating this computation is listed below where *x* is an input in radians. You must choose a value for **N,** which represents the number of iterations that the approximation will run for. The larger this value, the more accurate your approximation will be but a large value will slow down computation significantly. Experiment with different values of **N** and check the computation accuracy. Note that an exponent indicates how many times a variable is multiplied by itself. Use an additional loop to make this calculation.

   $$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots \approx \sum_{n=0}^{N} \frac{(-1)^n x^{2n+1}}{(2n+1)!}$$

4. **No marks will be awarded for the question if the 'cmath' library is included!**

Sample Output:

```
How many rectified sine cycles should be printed? 0.75

****
*******
***********
**************
*****************
********************
************************
****************************
*******************************
********************************
**********************************
***********************************
************************************
*************************************
*************************************
*************************************
*************************************
*************************************
**************************************
************************************
***********************************
**********************************
*********************************
*******************************
*****************************
***************************
************************
**********************
****************
**************
**********
********
****
*
****
*******
***********
**************
*****************
********************
************************
****************************
*******************************
********************************
**********************************
***********************************
************************************
*************************************
*************************************
*************************************
*************************************
*************************************

Program ended with exit code: 0
```

//Second run of the program

```
How many rectified sine cycles should be printed? 1.5

****
*******
**********
**************
******************
**********************
**************************
******************************
**********************************
**************************************
******************************************
**********************************************
**********************************************
**********************************************
**********************************************
**********************************************
**********************************************
**********************************************
**********************************
******************************
**************************
**********************
******************
**************
**********
********
****
*
****
*******
**********
**************
******************
**********************
**************************
******************************
**********************************
**************************************
******************************************
**********************************************
**********************************************
**********************************************
**********************************************
**********************************************
**********************************************
**********************************
******************************
**************************
**********************
```

```
*********************
******************
**************
***********
********
****
*
****
*******
***********
**************
*****************
********************
***********************
**************************
*****************************
*******************************
*********************************
***********************************
*************************************
**************************************
***************************************
****************************************
****************************************
****************************************
****************************************
****************************************
***************************************
**************************************
*************************************
***********************************
*********************************
*******************************
*****************************
**************************
************************
*********************
******************
**************
***********
********
****

Program ended with exit code: 0
```

E.3 Value Conversion with Functions

Requirements:

Re-implement the question E.2 from the previous lab concerning value conversion with the following specifications:

1. The program must run in a loop and print five possible input choices to the user on the console. A choice of option 5 (Exit) will break the loop and end the program. **You must use a switch statement to handle the possible menu options.**
2. You must validate that the user enters an integer in the appropriate range for the menu options. You can assume the input will be an integer. The user must be able to enter real numbers for conversion inputs and converted values must be real number outputs. Assume that the user will only enter numbers for the conversions.
3. The menu options included will be:
    1. Celsius -> Fahrenheit
    2. Centimeters -> Inches
    3. Meters -> Feet
    4. Km/h -> MPH
    5. Exit
4. The formulas needed to convert the values are:

$$Fahrenheit = \frac{9}{5} * Celsius + 32$$

$$Inches = 0.39 * Centimeters$$

$$Feet = 3.28 * Meters$$

$$MPH = \frac{Km/h}{1.609}$$

5. You must use a loop to validate that a positive degree is input for conversion options 2, 3, and 4 and keep asking the user for an appropriate input until a positive value is entered.
6. If a value of less than 0° C is input for option 1 (temperature conversion), print “Ice may be possible, please be careful.” to the console.
7. For option 4, validate that the input value of Km/h entered by the user is between 0 Km/h and 160 Km/h, simulating the common range of an automotive speedometer.

Additional Specifications:

1. You will now separate the logic from each switch case (inside main() function) into separate functions with the following prototypes:

```
double convertCelsiusToFahrenheit(double degreesCelsius);

float convertCentimetersToInches(double centimeters);

double convertMetersToFeet(float meters);

float convertKilometersToMiles(float kilometers);
```

2. Inside each switch case, the program will continue to ask the user for an input, and this input will be validated inside the switch statement (inside main() function). Once validated, the value will be passed to one of the above functions as a parameter. The return value of each function will be returned and displayed to the user from the main() function **only** and there must not be any cout or cin statements in the above functions.

3. You must implement one additional function that will take no parameters but will return a number entered by the user corresponding to a menu choice and validate that the number is in the appropriate range (1-5 inclusive). This function will be called each time the menu is displayed to the user (from the main() function) and will contain the appropriate cout, cin, and loop structures required to ask the user for an input and perform validation. The prototype must be as shown:

```
int fetchMenuChoice();
```

4. To receive full credit for this lab, the function prototypes given must not be modified.

Sample Output:

```
------------------------------
Student Name
Lab #3
Date:
Question #2
------------------------------

******************************
****** Value Conversion ******
******************************
* 1. Celsius -> Fahrenheit   *
* 2. Centimeters -> Inches   *
* 3. Meters -> Feet          *
* 4. Km/h -> MPH             *
* 5. Exit                    *
******************************

Please input a choice (1-5): 7
******************************
****** Value Conversion ******
******************************
* 1. Celsius -> Fahrenheit   *
* 2. Centimeters -> Inches   *
* 3. Meters -> Feet          *
* 4. Km/h -> MPH             *
* 5. Exit                    *
******************************

Please input a choice (1-5): 1
Input a degree in Celsius: 21.3
The conversion is 70.34 F.

******************************
****** Value Conversion ******
******************************
* 1. Celsius -> Fahrenheit   *
* 2. Centimeters -> Inches   *
* 3. Meters -> Feet          *
* 4. Km/h -> MPH             *
* 5. Exit                    *
******************************

Please input a choice (1-5): 4
Input a number of Km/h (0-160): 168
Input a number of Km/h (0-160): 4
The conversion is 2.48602 MPH.

******************************
****** Value Conversion ******
******************************
* 1. Celsius -> Fahrenheit   *
* 2. Centimeters -> Inches   *
* 3. Meters -> Feet          *
* 4. Km/h -> MPH             *
* 5. Exit                    *
******************************

Please input a choice (1-5): 5

------------------------------
Goodbye!
------------------------------

Program ended with exit code: 0
```

F. Additional Practice Question (Not for marks):

1. Write a program that draws a multiplication table as follows (user supplies the size)

```
Please enter the size of the multiplication table: 6

   1  2   3  4  5
   --------------
1* 1  2   3  4  5
2* 2  4   6  8 10
3* 3  6   9 12 15
4* 4  8  12 16 20
5* 5 10  15 20 25
```