Western Engineering

ES1036: Programming Fundamentals – **Winter 2018**

Lab03: Practicing switch statements and loop structures 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 practice switch statements and loops (while, do-while) in a C++ programming environment. Loops constitute one of the **most basic and powerful** programming concepts. Please open the existing solution ES1036Labs and add a new project named "Lab03".

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 Lab03, 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_lab03_q1.cpp,
		your_Western_user_name_lab03_q2.cpp,
		your_Western_user_name_lab03_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)

While loops:

A `while`-loop is a useful and common loop structure in C++ programming (and many other languages). Every while-loop has a structure similar to the following:

```
while (condition) {
      statement block;
}
```

Review the following code-segment which gives a simple example of a while-loop:

```
int i = 0;
while (i <= 20) {
      std::cout << " "<< i;
      i++;
}
std::cout << std::endl;
```

Before executing the while-loop, the variable ***i*** is declared and given a value of **0**. The first statement of code after the while-loop is executed a number of times (**21** times). Multiple statements cannot be executed inside a while-loop **unless** braces { } are used to contain all the statements. The while-loop continues to run as long as the *condition*- statement is **true** (for the above case, as long as ***i*** is less than or equal to 20).

The statement ***i++*** indicates that the variable ***i*** is incremented by 1 at each iteration of the loop. The code-segment above will generate the following sample output:

```
0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20
Press any key to continue . . .
```

Break statements inside loops:

A break-statement (see lecture unit 4, slide 108) inside a loop breaks the loop. See lecture unit 4, slide 111 for an example on break-statements in a while-loop. See the following example code using a break-statement inside a switch-structure to break the switch, and inside a loop structure to break the loop.

```
#include <iostream>
using std::cout;
using std::cin;
using std::endl;

int main() {
    char choice;

    while(1) //an infinite loop
    {
        cout<<" A: Do nothing\n B: Do everything\n C: Exit\n";
        cout<<"Enter choice: "; cin>>choice; choice = toupper(choice);
        switch (choice)
        {
            case 'A':
                cout<<"Do nothing\n";
                break; //breaks the switch-structure
            case 'B':
                cout<<"Do everything\n";
                break; //breaks the switch-structure
        } //ends switch structure
        if(choice == 'C')
            break; //breaks the loop
    }// ends while loop
    cout<<"Good bye!\n";
}//ends main()
```

D. Good Programming Practice

- When you start writing code, compile and test intermittently.
- If compilation fails, read the error messages. **Address the first listed error before the others** as errors can propagate through your program causing more errors further.
- 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.
- Use comments to explain the logic of your program. Comments help programmers remember their thought process when reviewing code later in time. It may be the case that you complete your lab early in the week and you need the comment reminders to explain your program.

E. Lab Questions

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

E.1 Computing a Factorial

Requirements:

Write a program that takes an input integer and outputs the number's factorial.

Specifications:

1. You can assume the user will enter an integer number.
2. Validate that the input integer is greater than or equal to zero. Use a loop to repeatedly ask the user to input an integer until the input is not negative.

3. The factorial of an input integer ***n*** is denoted as ***n!*** and is computed as:

n! = $\boldsymbol{n(n-1)(n-2)(n-3)\ldots 2\times 1}$ = 1 x 2 x 3 x ……x n; while n>0.

n! = 1; while n = 0.

It should be clear from the definition above that an input of 0 must result in an output value of 1 because ***0! = 1***.

6. **Hint:** This problem has been solved for you in unit 4.

7. **Hint:** Always use a **double-type** variable to store the factorial result. Ask your TA if you are unsure why this is necessary.

Sample Output:

//First run of program

```
Input an integer number to calculate its factorial: 9
The resulting factorial is: 362880
Press any key to continue . . .
```

//Second run of program

```
Input an integer number to calculate its factorial: -4
Invalid Entry! Please enter a positive integer: 4
The resulting factorial is: 24
Press any key to continue . . .
```

//Third run of program

```
Input an integer number to calculate its factorial: -160
Invalid Entry! Please enter a positive integer: -8
Invalid Entry! Please enter a positive integer: 0
The resulting factorial is: 1
Press any key to continue . . .
```

E.2 Switch Menu Value Conversion

Requirements:

Write a program that displays a menu with four conversion options and exit. Your program will perform the conversions specified below.

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 and not a while or do-while loop.**
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.

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
```

E.3 Statistics on Integer Inputs

Requirements:

Write a program that calculates statistics on a set of integer inputs.

Specifications:

1. Print a header to the console similar to the one shown in the sample output.
2. First the program will ask the user to input an integer size for the set of integers they would like to use for computation (the number of integers they wish to input in the next step). Validate that the entered integer number is greater than 0. Assume that the user will only enter integer number.
3. Your program must use a loop structure (while or do-while) where the user is asked to input a new integer on each iteration, until the total number of integers have been input by the user. You can assume that the user will enter integer values and no validation is required.
4. After each iteration, the program will print: the mean, count of even integers, and count of odd integers, minimum value, and maximum value input thus far to the console.
5. Mean will be computed as $\mu = \frac{\sum_{n=1}^{N} x_n}{N}$ where the numerator is the sum of all integers input and the denominator is the total count of integers input. For example, if the user enters 3 integers, the mean will be = $(x_1 + x_2 + x_3)/3$, where x_1, x_2 and x_3 are the inputted values. Note that the mean value must be computed accordingly to accommodate real number results.
6. **Hints:**
   - To determine if an input is even or odd, see lecture unit 4, slide 18.
   - Watch out for integer truncation during division: 5/2 is 2 in C++, while 5/2.0 = 2.5

Sample Output:

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

Enter an integer number for the number of loop iterations: 3
Enter integer 1: 2
The mean of 1 input(s) is: 2
Max value: 2
Min value: 2
Even count: 1
Odd count: 0

Enter integer 2: -2
The mean of 2 input(s) is: 0
Max value: 2
Min value: -2
Even count: 2
Odd count: 0

Enter integer 3: 53
The mean of 3 input(s) is: 17.6667
Max value: 53
Min value: -2
Even count: 2
Odd count: 1

Goodbye!
Program ended with exit code: 0
```

Additional questions for practice

1- Write a program that prompts the user to enter 3 integer numbers. Display the numbers from smallest to largest and then again from largest to smallest. (Use do-while loop)
2- Write a program which calculates the sum of an undetermined number of integers. The program should keep prompting the user for a new integer until they enter a 0. The program will then display the result.
3- Write a program to compute the total number of combinations possible for your combination lock. There are ten integers (n = 10; i.e. 0 … 9) and you can always form a three digit combination number r (i.e. 423). You can use the same number multiple times to form your combination. HINT: you may implement the following formula: $\frac{n!}{(n-r)!}$
4- Using while loop write a program that prompts the user to enter his age and decide if the user is young (for an age <100), old (100 years) and (very old >100). Ask the user if he/she wants to run the program again.