Introduction

This lab will provide students with an introduction to the fundamentals of multi-way conditional selection in the C++ programming language. The focus of this lab is on how to solve complex problems using the C++ “switch” statement. At the end of this module, students will…

- Be able to solve simple problems using C++ “switch” statement
- Understand the structure of C++ “switch” statement

These exercises have two different types of answers. Some questions ask for answers to specific questions; you should collect these answers in a Microsoft Word document. Other exercises ask you to write code; for these exercises, structure your storage drive in the following manner. There should be a palms folder, and within that folder there should be a LabSwitch folder. All coding exercises (and related folders) for this lesson should be placed in the LabSwitch folder.

As always, be sure to properly document your code. Consult the C++ Coding Guidelines document for proper coding standards. Use good design principles and design the solution before attempting to write code. All programs should make use of multiple functions besides main() (use the input/process/output cycle as your guide). Think about how each problem actually involves more than one subproblem. Construct functions aligned with input-process-output steps. Let’s begin!

Part 1: Multi-Way Conditional Selection in C++

We have previously talked about multi-way conditional selection when we discussed the cascading if statement. C++ provides another multi-way selection statement called a switch statement. This allows for a more efficient selection process for those cases in which the conditional test is only a test for equality.

In this lab we will learn how to use the C++ switch statement to perform multi-way conditional selection. The switch statement has several potential advantages, including the need for less code and a greater level of execution efficiency. However, it does not provide any more functionality; switch is simply an alternative way to decide amongst a group of choices only when you are looking for an exact match.
The Switch Statement in C++

A switch statement allows for selection from multiple branches, just like a cascading if; however, the choice of which branch to take is determined by a controlling expression given in parentheses. The general form of the switch statement is as follows:

```c++
switch ( /* controlling expression */ )
{
    case ( /* ordinal value */ ) :
    {
        // one or more statements to execute if the
        // ordinal value equals the controlling expression
        break;
    }
    case ( /* ordinal value */ ) :
    {
        // one or more statements to execute if the
        // ordinal value equals the controlling expression
        break;
    }
    ...
    // potentially more case clauses
    default:
    {
        // the default case; no match was found in the previous cases
    }
}
```

The order of execution is as follows:

- The value of the controlling expression is determined. The value in a controlling expression must be of an ordinal type (int, char, or bool).
- That value is compared against each case value in turn, from the top-down.
- As soon as an exact match is found, the block of code associated with the particular case is executed.
- When a break statement is found, execution exits the switch and continues with the code following the end of the switch statement.

Note that the default case is optional; it acts as an "else" case for the switch statement. The break statement is important, however, because unless it is found the code will "drop-through" and start executing the next block of code.

The switch statement is useful when you need to execute different blocks of code based on a test for equality involving an ordinal value. An ordinal value has a logical successor and predecessor; examples are int, char, or bool values.
Part 2: User Menus and the Switch Statement

Programs requiring menus are especially good examples of when a switch is useful. Menu choices in Microsoft Windows, MAC OS X, Linux, and other Operating Systems are often represented internally as simple integers. For example, the “File→Open” menu option could be represented by the number 1, the “File→Save” menu option by the number 2, and so on. The ordinal nature of these options makes a switch statement a useful tool in directing the code to the proper course of action based upon which menu option was chosen by the user. The following exercise demonstrates a program which uses a menu and a switch statement to execute user commands.

Exercise 1: Suppose we wanted to reproduce a text-based computer game called “Super Star Trek”, a space simulation game created for mainframes in the 1970s. Like many text-based computer games, SST uses a series of menus to facilitate interaction with the user. When the user is presented with a list of choices, the user selects a choice by typing in an identifier for the choice (usually a simple integer). In this program, you will begin creating those menus and the code to receive the user’s choice.

Your program should include an input function designed to display the main menu and allow the user to enter their selection. A prototype for this function is as follows:
The screen output should look like so:

**MAIN MENU**

(1) NAV Navigate the Starship  
(2) SRS Short-Range Sensor Scan (One Quadrant)  
(3) LRS Long-Range Sensor Scan (Nine Quadrants)  
(4) PHA Phaser Control  
(5) TOR Photon Torpedo Control  
(6) SHE Shield Control  
(7) DAM Damage and State-of-Repair Report  
(8) COM Call Library Computer  

Command?

If (and only if) the user selects (8) from the main menu, display the following sub-menu:

**LIBRARY COMPUTER MENU**

(0) Cumulative Galactic Record  
(1) Status Report  
(2) Photon Torpedo Course Data  
(3) Starbase Navigation Data  
(4) Direction/Distance Calculator  
(5) Quadrant Nomenclature Map  

Computer Active and Awaiting Command:
Your program should have another function to display this sub-menu and receive the user’s command. A prototype for this function would be as follows:

```cpp
int displayLibraryMenu();
```

The algorithm for your complete program should be as follows:

1) Display the main menu and receive the user input (i.e. the command).
2) Use a switch statement to determine which command the user selected.
   a. In response to a specific command, post a message to the screen (e.g. “You selected (4) PHA Phaser Control”)
   b. If the user selected an invalid command, alert them to this error. Use the default case of the switch to handle this.
   c. If the user selected option (8), display the library menu and receive the user input (i.e. the library command).
      i. In response to a specific library command, post a message to the screen (e.g. “You selected (4) Direction/Distance Calculator”)
      ii. If the user selected an invalid option, alert them to this error. Use the default case of the switch to handle this.
3) End the program.

Compile and run the program, and verify that it works as it should.

**Part 3: More Applications of the Switch Statement**

*Resistors* are electrical components designed to limit the flow of electricity between two points. An example of a common resistor is shown below:
The four colored bands on the resistor can be used to calculate the resistance in kΩ. For example:

- The first band is the *first significant digit* of the resistance value;
- The second band is the *second significant digit* of the resistance value;
- The third band is the decimal *multiplier*;
- The fourth band indicates the *tolerance*; this is expressed in terms of plus (+) or minus (-) the calculated value (the nominal value).

For example, suppose we have the sequence RED-VIOLET-GREEN-GOLD. We would use the following table to help calculate the resistance:

<table>
<thead>
<tr>
<th>Color</th>
<th>Digit</th>
<th>Multiplier</th>
<th>Tolerance (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black</td>
<td>0</td>
<td>10⁰</td>
<td></td>
</tr>
<tr>
<td>Brown</td>
<td>1</td>
<td>10¹</td>
<td>1</td>
</tr>
<tr>
<td>Red</td>
<td>2</td>
<td>10²</td>
<td>2</td>
</tr>
<tr>
<td>Orange</td>
<td>3</td>
<td>10³</td>
<td></td>
</tr>
<tr>
<td>Yellow</td>
<td>4</td>
<td>10⁴</td>
<td></td>
</tr>
<tr>
<td>Green</td>
<td>5</td>
<td>10⁵</td>
<td>0.5</td>
</tr>
</tbody>
</table>
The nominal value could be calculated as $27 \times 10^5 = 2700000 = 2700 \, k\Omega$, with a tolerance of ±5 percent. As another example, consider BLUE-RED-RED-GOLD. The nominal value could be calculated as $62 \times 10^2 = 6200 = 6.2 \, k\Omega$, with a tolerance of ±5 percent.

**Exercise 2:** Create a new C++ Project called **Resistor**. Write a C++ program to input four-band color patterns and produce output listing the colors along with the resistance value in $k\Omega$. Think carefully about the input – there are many ways to allow the user to input the colors.

**Exercise 3:** Create a new C++ Project called **TwelveDays**. Construct a program that prints one of the gifts of the song "The Twelve Days of Christmas". The program should input the user’s choice of day (1-12) and display the corresponding gift. Use a switch statement to determine the output. Remember the following gifts are part of the “Twelve Days of Christmas”:

- A partridge in a pear tree
- Two turtle doves
- Three French hens
- Four calling birds
- Five gold rings
- Six geese a-laying
- Seven swans a-swimming
- Eight maids a-milking
- Nine ladies dancing
- Ten lords a-leaping’
- Eleven pipers piping
- Twelve drummers drumming