**Lab Experience Seven**

**Objectives:**

1. Understand the relationship between the switch statement and the if/else if statement.
2. Introduction to repetition structures.

**switch statement**

The switch statement is used when testing for exact values and cannot be used to test a range of values. Whenever an if-else-if statement is used using the equality operator or the not equal operator, the switch statement can be used.

The syntax for the switch statement is:

**switch(ordinal value){**

**case constantvalue-1: statement-1;**

**break;**

**case constantvalue-2: statement-2;**

**break;**

**// continuing with the switch**

**casecontantvalue-n: statement-n;**

**break;**

**default: statement;**

**break;**

**}**

An ordinal value in C++ is any value that has a definite predecessor/successor. This means only char and integer data types can be used in a switch statement. (Note: This does not necessarily exclude values that have decimal points since they can be type casted to **int** by using a **static\_cast** statement.) Each constantValue must be unique since the switch statement tries to match the ordinal value with one of the cases. If no match occurs with any of the cases, then the statement(s) following the default is used. The default part of the switch statement is optional, so if there are no matches the next executable statement following the switch is executed.

The break statement is used to terminate execution of the switch statement. If it is omitted, the next statement will be executed in the switch statement following the case that was matched will be executed, i.e. execution will drop through. More than one valid C++ statement can be placed after any of the cases and curly braces are not necessary.

**Example:**

Below is an example of an if-else-if statement and its corresponding switch statement.

|  |  |
| --- | --- |
| **if (modelNum == 300)**  **cost = 500;**  **else if (modelNum == 200)**  **cost = 300;**  **else if (modelNum == 100)**  **cost = 150;**  **else**  **cout<<modelNum**  **<< “ is an invalid value\n\n”;** | **switch (modelNum){**  **case 300: cost = 500;**  **break;**  **case 200: cost = 300;**  **break;**  **case 100: cost = 150;**  **break;**  **default:**  **cout<<modelNum**  **<< “ is an invalid value\n\n”;**  **}//end switch** |

The above two statements execute exactly the same way and produce the same executable code.

**Background**

Computers were first created to solve complex formulas which involved iterating over the same set of instructions repeatedly. Since the majority of all computer operations involve this type of repetition, all languages contain some type of loop structure in one form or another. In C++ there are three types of loop structures a programmer can use to solve a problem that is repetitive in nature. These are as follows:

1. **while** loop
2. **for** loop
3. **do-while** loop

The loop structures listed in one and two above are called pre-test loops because the test condition is always executed first. The **do-while** loop is called a post-test loop because the test condition is always executed last.

**Increment and Decrement Operators**

The increment operator is defined as **++** and is used to increment an ordinal value by one. Recall from Lab Experience Five an ordinal value in C++ is either type int or type char.

The decrement operator is defined as **--**and is used to decrement an ordinal value by one.

Examples:

The following C++ statements are used to increment the memory location by one:

1. **count = count + 1;**
2. **count += 1;**

The increment operator can perform these same tasks without using an assignment operator. Equivalent statements are:

**++count; // Prefix operation**

**count++; // Postfix operation**

The following C++ statements are used to decrement the memory location by one:

1. **count = count - 1;**
2. **count -= 1;**

The decrement operator can perform these same tasks without using an assignment operator. Equivalent statements are:

**--count; // Prefix operation**

**count--; // Postfix operation**

When used by themselves as shown above, the post and prefix operators execute exactly the same way. When used with expressions the behavior changes. Prefix operators follow the rule: Increment/decrement the value contained in the variable, and then use it. Postfix operators follow the rule: Use the value contained in the variable, and then increment/decrement the value.

**y = x++; // postfix operation**

This is translated by the compiler as:

**y = x; // use the value first**

**x++; // then increment**

Given the following statement:

**y = ++x; // prefix operation**

This is translated by the compiler as:

**x++; // increment the value**

**y = x; // then use the value**

**The while loop**

The while loop is a pre-test loop structure designed to execute a series of statements repeatedly as long as the expression is true. Once the expression becomes false, the loop terminates and execution will continue with the statements following the loop structure.

The syntax for the while loop structure is:

expression

statement-1 is called the loop body. Somewhere within the loop body must be a statement that will cause the expression to become false.

**while(expression)**

**statement-1;**

statement-1

false

expression is either a relational statement, compound statement, or any other valid C++ statement that results in either a nonzero or zero result. Recall that a nonzero result is true, whereas a zero result is false.

If more than one statement follows the while clause then these statements must be a block statement. Recall a block statement is defined as a series of statements enclosed between curly braces, **{}**.

The syntax for more than one statement following the loop is shown below:

This is also called the loop body. Somewhere within the loop body must be a statement that will cause the expression to become false.

**while(expression){**

**statement-1;**

**statement-2;**

**statement-3;**

**// continuing**

**statement-n;**

**}**

The flow of execution is the same as shown above.

**Counter-Controlled Loops**

A counter controlled loop is a loop that utilizes a counter to control how many times the loop will execute.

**Example:**

**#include <iostream>**

**using namespace std;**

**int main(){**

**int count = 1; // counter variables must be initialized**

**// to the starting value before being used.**

**int sum = 0; // initialize the accumulator**

**while (count <= 5){**

**sum = sum + count; // start adding the first n integers**

**count++; // increment the counter**

**}// end while**

**cout<< “The sum of the first 5 integers is “ << sum <<endl;**

**return 0;**

**}// end main**

**Hand-Program Trace**

|  |  |  |
| --- | --- | --- |
| count | sum | count <= 5 |
| 1 | 0 | ? |
| 1 | 1 | True |
| 2 | 3 | True |
| 3 | 6 | True |
| 4 | 10 | True |
| 5 | 15 | True |
| 6 | 15 | False |

The flowchart representation of the program is shown below:

cout<< “The sum of the first 5 integers is “ << sum;

count++;

sum = sum + count;

count<=5

sum = 0;

count = 1;

True

False

The variable count will be called the loop control variable in this instance because the value contained in count controls how many times the loop will execute. It is customary to place the statement that contains the variable used in the expression test as the last statement in the loop body as shown above, i.e. **count++**. What would happen if count didn’t change values inside the loop body? The loop would never stop executing since the expression will always be true. This is called an infinite loop and every programmer has written an infinite loop at some point of their career. How do you know if you have an infinite loop? 1) The output never stops or 2) nothing is displayed on the output screen except the blinking cursor.

**Sentinel Values**

A sentinel value is used to terminate a data set being entered by the user. The value should be a value that is not part of the data set. For example, the sentinel value -999 could be used to terminate a series of positive numbers.

The purpose of using a sentinel value is to process a list of numbers where the actual length of the list of numbers is not known in advance. Please note: When processing a list of numbers of indeterminate length, we never ask the user how many numbers are going to be entered. Instead we use the sentinel value.

**Example:**

Find the average temperature for the equator. An abnormal temperature for equatorial temperatures would be -1. Therefore, this value can work as a sentinel value.

**#include <iostream>**

**#include <iomanip>**

**using namespace std;**

**const double SENTINEL = -1.0; // use a constant in case a change needs to be made.**

**int main(){**

**double temperature, avgTemp;**

**doubletemperatureSum = 0; // total of all temperatures recorded**

**int temperatureNumber = 0; // how many temps to process?**

**// always prime the loop before beginning the process**

**cout<< “Enter a temperature “ << SENTINEL << “ to quit “ ;**

**cin>> temperature;**

**while (temperature != SENTINEL){**

**temperatureSum += temperature;**

**temperatureNumber++;**

**cout<< “Enter a temperature “ << SENTINEL << “ to quit “ ;**

**cin>> temperature;**

**}// end while**

**cout<< fixed <<showpoint<<setprecision(2);// set the output flags**

**if(temperatureNumber){ // a nonzero value means at least one temp was processed**

**avgTemp = temperatureSum / temperatureNumber; // no typecast. Why?**

**cout<< “The average temperature is “ <<avgTemp**

**<<“ degrees Fahrenheit” <<endl;**

**}// endif**

**else**

**cout<< “User terminated processing before any temperatures were entered.\n”;**

**return 0;**

**}// end main**

**Lab Exercises**

**Directions:**

Start Microsoft word and record the questions and answers to all of the exercises in the lab 6 word document   
Answer the following questions based on material presented in lecture and found in chapters 1-5 of our textbook.

**Exercise One:**

1. Convert the following if/else if statement into a switch statement.

if(grade == 'A' || grade == 'a')

aCount++;

else if(grade == 'B' || grade == 'b')

bCount++;

else if(grade == 'C' || grade == 'c')

cCount++;

else if(grade == 'D' || grade == 'd')

dCount++;

else

fCount++;

**switch (grade){**

**case ‘A’:**

**case ‘a’: aCount++;**

**break;**

**case ‘B’:**

**case ‘b’: bCount++;**

**break;**

**case ‘C’:**

**case ‘c’: cCount++;**

**break;**

**case ‘D’:**

**case ‘d’: dCount++**

**default:**

**fCount++;**

**}//end switch**

1. Convert the following switch statement into an if/else if statement.

switch(choice){

case 1: charges = months \* 40;

break;

case 2:

case 3:

case 4: charges = months \* 20;

break;

case 5:

case 6:

case 7:

case 8:

case 9: charges = months \* 10;

break;

}

**If (choice == 1)**

**{**

**charges = months \* 40;**

**}**

**else if (choice == 2 || choice == 3 || choice == 4)**

**charges = months \* 20;**

**else if (choice == 5 || choice == 6 || choice == 7 || choice == 8 || choice == 9)**

**charges = months \* 10;**

1. What is the output of following code segment if ***choice*** contains the value 1 and ***months*** contains the value 5?

switch(choice){

case 1: charges = months \* 40;

case 2:

case 3:

case 4: charges = months \* 20;

case 5:

case 6:

case 7:

case 8:

case 9: charges = months \* 10;

}

cout<< “Your total charges are $ “ << charges <<endl;

**Output is : Your total charges are $ 200**

1. Perform a hand trace to determine the output of the following program segments. A snapshot of the program execution will receive no credit.An example of a hand trace is given above in the description of the lab.

a)

**inti = 3;**

**cout<<"12345678901234567890"<<endl;**

**while (i>= 0){**

**cout<<setw(3) <<i ;**

**i--;**

**}**

b)

**inti = 5,**

**j = 4;**

**while (i>= 0 && j > 0){**

**j--;**

**if(i % 5 == 0){**

**i--;**

**cout<<"Case 1: i = "<<i<<" , j = "<< j <<endl;**

**}**

**else{**

**i -= 2;**

**cout<<"Case 2: i = "<<i<<" , j = "<< j <<endl;**

**}**

**}**

**Fill in the blank**

1. A block of code that repeats forever is called a(n) \_\_\_\_\_\_**Infinite Loop**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. To keep track of the number of times a particular loop is repeated, one can use a(n) \_\_\_\_\_\_**counter**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
3. A while loop is a(n) \_\_\_\_**pre-test**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ loop that will never be executed if the test expression is initially false.
4. In the conditional if(++number < 9), number is incremented \_\_\_\_**1**\_\_\_\_\_\_\_\_\_\_\_it is compared to 9.
5. In the conditional if(number++ < 9), number is incremented \_\_\_\_**0**\_\_\_\_\_\_\_\_\_\_\_ it is compared to 9.

**Exercise 2**

Write a program that reports the contents of a compressed-gas cylinder based on the first letter of the cylinder's color. The program input is a character representing the observed color of the cylinder: 'Y' or 'y' for yellow, 'O' or 'o’ for orange, and so on. This process should continue until the user enters the letter ‘E’ or ‘e’ to exit the program. Cylinder colors and their associated contents are as follows:

orange ammonia brown carbon monoxide

yellow hydrogen green oxygen

If an invalid letter is entered, notify the user of incorrect input by stating Contents Unknown.

**Implement the above by using a switch statement.**

Copy and paste your program into your word document. Capture the output window and paste it below your program. You should have several output windows testing your program.

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//**

**// Programmer: Chris Dang Class: CSCI 1106 Fall 2014**

**//**

**// Description: This program will get the a character from the user**

**// (the character will represent the first letter of color of a compressed gas**

**// tank), the program will output the contents of the tank based on the**

**// character.**

**//**

**// The program will continue to run until the user exits by entering E or e**

**//**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**#include <iostream>**

**#include <string>**

**using namespace std;**

**int main () {**

**char color;**

**string content;**

**// Priming the loop; initilizes color**

**cout << "To determine contents held inside of tank,\nplease enter the first letter of the color of the tank. \nEnter the letter E to exit: ";**

**cin >> color;**

**while (color != 'e' && color != 'E'){**

**cout << endl; // seperates the contents descrption from the question; for readbility**

**switch (color){**

**case 'O': //Orange**

**case 'o':**

**cout << "Contents: Ammonia";**

**break;**

**case 'Y': // Yellow**

**case 'y':**

**cout << "Contents: Hydrogen";**

**break;**

**case 'B': // Brown**

**case 'b':**

**cout << "Contents: Carbon Monoxide";**

**break;**

**case 'G': // Green**

**case 'g':**

**cout << "Contents: Oxygen";**

**break;**

**default:**

**cout << "Contents unknown";**

**}//end switch (color)**

**cout << endl; // seperates content description from next prompt; for readibility**

**if (color != 'e' && color != 'E'){**

**cout << endl << "To determine contents held inside of tank,\nplease enter the first letter of the color of the tank. \nEnter the letter E to exit: ";**

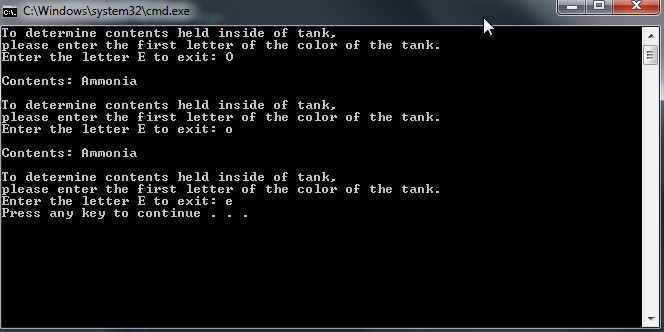
**cin >> color;**

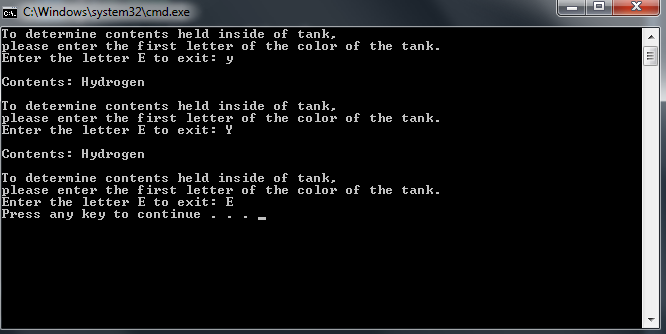
**}// end if**

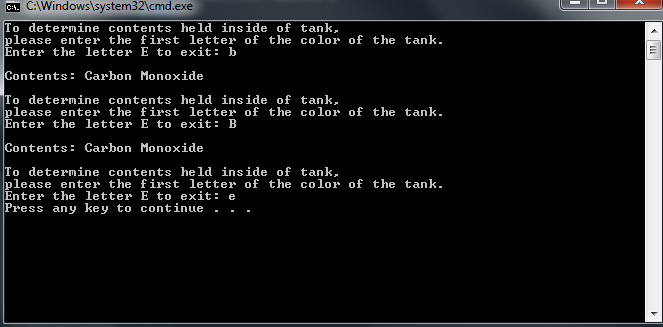
**} //end while**

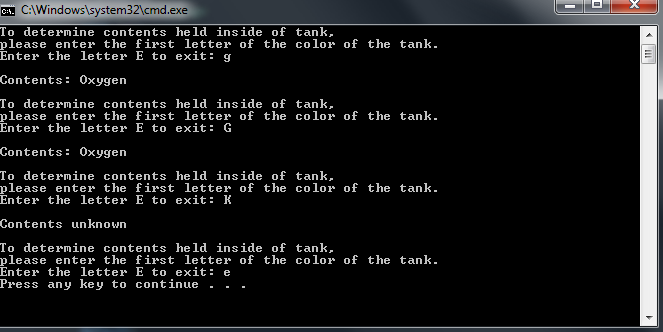
**return 0;**

**}**









**Due Dates:** According to the due date posted for the drop box folder.

**What to hand in:**

1. Compress the word document and the .cpp file created in exercise two into a single file called yournamelab7.zip, i.e. timwrennlab7.zip. Place the compressed file into the Lab Experience Seven drop box.
2. Hand in a print out of your word document.
3. Hand in a print out of your program.