**Lab Experience 4**

Lab Experience 4

**Lab Objectives**

* 1. Understanding relational and compound expressions.
  2. To learn and use conditional statements.
  3. To learn about the “dangling else” problem and how to correct it.
  4. To learn and use nested if statements.

**Background Information**

One way a decision structure is implemented in C++ utilizing and if/else/if statement. A decision structure is used whenever a decision has to be made within a computer program.

**Relational and Compound Expressions**

A relational expression is formed using the operators <, <=, >, >=, != (not equal), == (equality) and two operands. The result of the expression is either true or false, yes or no. The result is a numeric value of either 1 (means true) or zero (means false).

A common mistake is to confuse the assignment operator, (=) with the equality operator (==).

Example:

x > 5 – is a relational expression

x == 5 – is a relational expression

x = 5 – is not a relational expression but an assignment operation.

A compound expression is formed between a one or more relational expressions and a logical operator. The logical operators in C++ are: && (and), || (or), ! (not).

Truth tables are used to determine the results of a compound expression. The Truth tables for the logical operators used in C++ are shown below:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **&&(and)** | **True** | **False** | | **True** | **True** | **False** | | **False** | **False** | **False** | | |  |  |  | | --- | --- | --- | | **||(or)** | **True** | **False** | | **True** | **True** | **True** | | **False** | **True** | **False** | | |  |  | | --- | --- | | **!(Not)** | Result | | **True** | **False** | | **False** | **True** | |

Operator precedence is as follows:

! (not)

\*, /, %

+, -

<, <=, >, >=

!= (not equal), == (equality)

&&

||

All assignment operators

**The if statement**

The **if** statement is used to control the path of execution based upon the result of an expression. An if statement is used whenever a decision has to be made to solve the problem and based upon the result of the decision a different segment of code needs to be executed. The syntax and flow of execution is shown below.

true

expression

if (expression)

statement-1

statement-1;

false

Statement-1 will only execute if the expression is true, if it is false statement-1 does not execute. Recall an expression is true if it is non-zero and false only if the expression is zero. Only one valid C++ statement is allowed after the if. If more than one statement is necessary, a block statement must be used. For example:

if(expression){

statement-1, statement-2, statement-3 make up the block statement. A block statement is enclosed within curly braces, {}.

statement-1;

statement-2;

statement-3;

}

The expression is usually constructed with relational expressions, compound expressions, or C++ functions that will produce a true or false condition. Data types used in the expression can be numeric, character, or character arrays (strings).

Example:

**if(missileLocation== shieldLocation){ // Did a missile hit occur?**

**numCollisions += 1; // yes, add 1 to collision**

**shieldStrength -= missilePower; // reduce shield strength**

**}**

**if-else statement**

The syntax and flow of execution is shown below.

true

false

expression

**if (expression)**

**statement-1;**

statement-2

statement-1

**else**

**statement-2;**

The else portion allows for an alternative flow of control based upon the evaluation of expression as indicated above.

Example:

**if(missileLocation == shieldLocation){ // Did a missile hit occur?**

**numCollisions += 1; // yes, add 1 to collision**

**shieldStrength -= missilePower; // reduce shield strength**

**}**

**else**

**cout<< “Enemy missile missed” <<endl;**

Why is the statement following the else not enclosed in curly braces, { }? Since only one statement follows the else it is not a block statement and curly braces are not needed.

**if-else–if statement**

The syntax and flow of execution is shown below.

expression1

**true**

**false**

**if (expression1)**

**statement-1;**

true

expression2

statement-1

**else if(expression2)**

**statement-2;**

statement-2

**false**

if-else-if statements are used predominately

in testing a range of values. An example is

on the next page utilizing and explaining

thestrcmpfunction.

**Comparing C-Style strings:**

If comparing character arrays, the ***strcmp*** function must be used. When using ***strcmp*** you must also include the ***cstring*** header file. The format of ***strcmp*** function is shown below:

strcmp( charArr1, charArr2);

The function will return the following values based upon the comparison of the two arrays:

|  |  |
| --- | --- |
| **Comparison** | **Result** |
| If the two character arrays are identical | Zero is returned. |
| If charArr1 < charArr2 | A negative value is returned. |
| If charArr1 > charArr2 | A positive value is returned. |

**Example:**

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

**\* Programmer: Tim Wrenn**

**\***

**\* Program: An example comparing two strings using the strcmp function found in the**

**\* cstring header file.**

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

**#include<iostream>**

**#include<cstring>**

**usingnamespace std;**

**constint SIZE = 15;**

**int main(){**

**char array1[SIZE] = "Hello";**

**char array2[SIZE] = "hello";**

**cout <<"The value of strcmp is "<< strcmp(array1, array2) << endl << endl;**

**if(strcmp(array1, array2) < 0)**

**cout << array1 <<" is less than "<< array2 << endl;**

**elseif(strcmp(array1, array2) > 0)**

**cout << array1 <<" is greater than "<< array2 << endl;**

**else**

**cout << array1 <<" is equal to "<< array2 << endl;**

**cout << endl;**

**return 0;**

**}**

The output produced by the above program is:

**The value of strcmp is -1**

**Hello is less than hello**

**Press any key to continue...**

**Comparing String Objects**

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

\* Programmer: Tim Wrenn

\*

\* Program: An example comparing two strings using the strcmp function found in the

\* cstring header file.

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

#include<iostream>

#include<string>

usingnamespacestd;

int main(){

string name1 = "Hello";

string name2 = "hello";

cout<<"The value of name1 > name2 is "<< (name1 > name2) <<endl<<endl;

if(name1 < name2)

cout<< name1 <<" is less than "<< name2 <<endl;

elseif(name1 > name2)

cout<< name1 <<" is greater than "<< name2 <<endl;

else

cout<< name1 <<" is equal to "<< name2 <<endl;

cout<<endl;

return 0;

}// end main

**Trailing else**

Since the if-else-if statement is a selection structure, it is sometimes convenient to have a trailing else after the last else-if statement as shown on the previous page.

**Nested if statements**

A nested if statement occurs when an if is enclosed within another if statement. The if statement can be after the if clause or after the else clause. The following rule concerning else’s is used whenever a nested if statement is used.

Each else pairs up with the closest unmatched if statement.

Example:

**1 if(taxCode ==0)**

**2 if(income > 10000)**

**3 taxOwed = true;**

**4 else**

**5 taxOwed = false;**

**6 else**

**7 // do some more processing**

When proper indentation is used it is clear statements 1 and 6 are together and statements 2 and 4 are together, but consider the same code snippet without indentation below:

**if(taxCode ==0)**

**if(income > 10000)**

**taxOwed = true;**

**else**

**taxOwed = false;**

**else**

**// do some more processing**

Without indentation it is not clear which else clause belongs to which if clause statement. Therefore, indentation is used to clearly identify what statements are connected to each statement.

Consider the following segment of code:

**if(taxCode == 1)**

**if(income > 10000)**

**taxAmount = income \* TAXRATE/100;**

**else**

**taxOwed = false;**

In the above situation, it is clear the programmers intent (because of the indentation) is that the else should be attached to the first if clause, but because of the **second if** it will be associated with

**if(income > 10000).**

**The above situation is called the dangling else condition and causes a significant amount of logic errors in programs. To correct this problem curly braces are used to detach the else from the second if statement as follows:**

**if(taxCode == 1){**

**if(income > 10000)**

**taxAmount = income \* TAXRATE/100;**

**}**

**else**

**taxOwed = false;**

**Lab Exercises**

**Directions:**

Start Microsoft word and record the questions and answers to all of the exercises in the lab 5 word document   
Answer the following questions based on material presented in lecture and found in chapters 1-4 of our textbook.

**Lab Exercises**

1. The >>in a cin statement is called the \_\_\_\_\_**extraction**\_\_\_\_\_\_ operator.
2. What is the result of the expression:**35<= x <= 55**assuming x has a value of 30? (Hint: This is not a mathematical expression but a C++ expression so associatively rules apply and the result will either be a 1 or a zero.)

**The result is 1. Left to right associative rules apply which evaluates 35 <= x as 0, and then evaluates 0 <= 55 which is true, therefore 1.**

1. Assume **i = 2, j = 3, k = 4**, and**m = 1**. What does each of the following statements output? (**Note: It should be a numeric value.**)

|  |  |
| --- | --- |
| cout<< (i!= 1) <<endl;  1 | cout<<(**(**i< j**)**< m) <<endl;  0 |
| cout<< (j >= i || k == m) <<endl;  1 | 5 < 3 or 0 >= 4  cout<< (k + m < j || 3 - j >= k) <<endl;  0 |
| J assigned 2  cout<< (j = 2) <<endl;  2 | 3 >= 2 and 4 == 1  cout<< (j >= i&& k == m) <<endl;  0 |

1. What is the result of the compound expression: **x < 110 &&< 200** if x contains 120? Why?

**There will be a syntax error because there need to be 2 operands on the right side of the AND (&&) statement and there is only 1.**

1. Write an if-else statement that outputs the word **Passed** provided the value of the variable exams is greater than or equal to 60 and also the value of the variable **programsDone** is greater than or equal to 10. Otherwise, the if-else statement should display the word **Failed**.

**If (programs >= 10 && exams >= 60)**

**cout<< “Passed” ;**

**else**

**cout<< “Failed”;**

1. Write an if-else statement that outputs the word **Warning** provided that either the value of **temperature** is greater than or equal to 100, or the value of **pressure** is greater than or equal to 200. Otherwise the if-else statement outputs the word **OK**.

**if (temperature >= 100 || pressure >= 200)**

**cout <<"Warning";**

**else**

**cout <<"OK";**

1. What happens when the following code segment executes?

string name;

intnumEmployees;

cin.ignore(80);

cout<<"Enter the name of the company ";

getline(cin, name);

cout<<"Enter the number of employees ";

cin>>numEmployees;

cout<<"Company Name: "<< name <<"Number of employees: "<<numEmployees;

**It will ignore 80 characters from the user, which will essentially confuse the user since no prompt will appear, and if the user enters more than 80 characters by accident, those characters will be assigned as a string, which the user most likely would not have intended.**

1. Start Visual Studio and download the program lab5\_1.cpp by copying and pasting the program from the web browser. Load lab5\_1. cpp into a Visual Studio project.

The purpose of problem 1 is to experiment with the code to duplicate the results. This is accomplished just by adding block statements. Recall using curly braces forms a block statement.

1. Insert the following comments at the top of the program.

// Computer Science 1106

// Lab 5 Problem 1

// your name here

1. The program, Lab5\_1.cpp, is printed below :( Note: The line numbers are for future reference and are not part of the actual code.)
2. **#include <iostream>**
3. **#include <cstdlib>**
4. **using namespace std;**
5. **int main(){**
6. **char answer;**
7. **intx,y ;**
8. **do{**
9. **cout<<"\n\nEnter a value for x ===>";**
10. **cin>> x;**
11. **cout<<"Enter a value for y ===>";**
12. **cin>> y;**
13. **cout<<"\n\n\n"; // double space**
14. **if(y == 8)**
15. **if(x == 5)**
16. **cout<<"@@@@@"<<endl;**
17. **else**
18. **cout<<"#####"<<endl;**
19. **cout<<"$$$$$"<<endl;**
20. **cout<<"&&&&&"<<endl;**
21. **cout<<"\n\nDo you wish to continue?(\"Y\" or \"N\")";**
22. **cin>> answer;**
23. **answer = toupper(answer); // convert to upper case**
24. **}while(answer == 'Y');**
25. **return 0;**
26. **}**

Modify program lab5\_1.cpp by **inserting curly braces** (specifically lines 13-19) to produce the output that is shown below**. The insertion of curly braces is the only modification allowed and NULL BLOCKS are not allowed. (A null block is {} without anything between the curly braces.)** Use proper indentation techniques. Note: It is possible that no modification is necessary to the program. If no modification is occurred, so state.

* 1. Enter the value of 5 for x and 8 for y when you are prompted. Was the following output produced? If no, modify your code by inserting braces. Paste the modified code (**i.e. only the portion of the program you changed**) and the output of the program with your changes into your word document.

**Yes. No modification was needed.**

@@@@@

$$$$$

&&&&&

* 1. Enter the value of 5 for x and 8 for y when you are prompted. Was the following output produced? If no, modify your code by inserting braces. Paste the modified code (**i.e. only the portion of the program you changed**) and the output of the program with your changes into your word document.

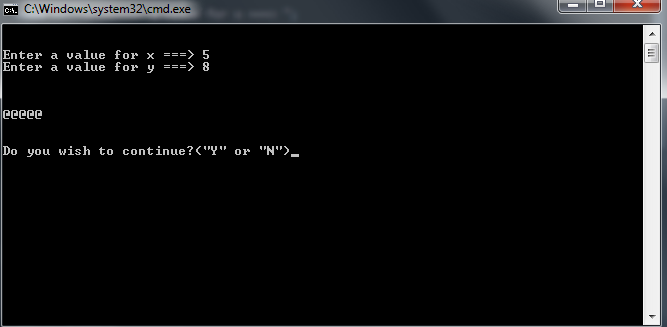
**No**

else{

cout << "#####" << endl;

cout << "$$$$$" << endl;

cout << "&&&&&" << endl;}



@@@@@

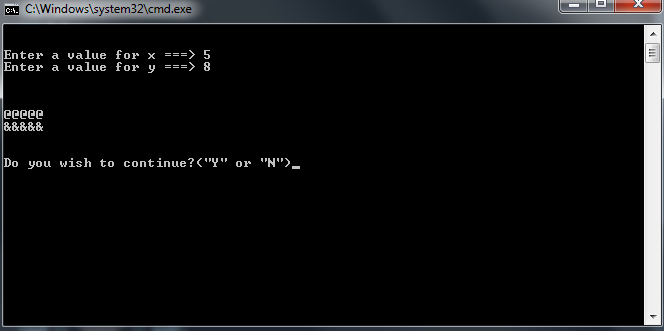
* 1. Enter the value of 5 for x and 8 for y when you are prompted. Was the following output produced? If no, modify your code by inserting braces. Paste the modified code (**i.e. only the portion of the program you changed**) and the output of the program with your changes into your word document.

**No**

else{

cout << "#####" << endl;

cout << "$$$$$" << endl;}



@@@@@

&&&&&

* 1. Enter the value of 5 for x and 7 for y when you are prompted. Was the following output produced? If no, modify your code by inserting braces. Paste the modified code (**i.e. only the portion of the program you changed**) and the output of the program with your changes into your word document.

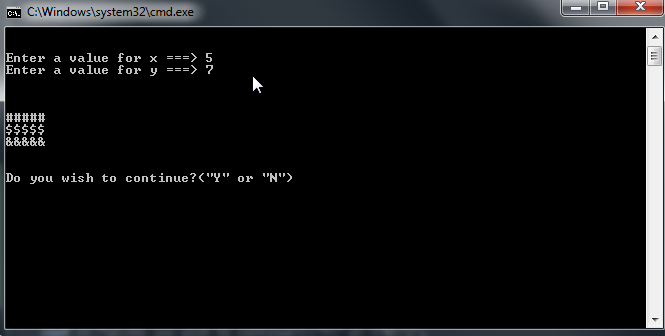
No

if(y == 8){

if(x == 5)

cout << "@@@@@" << endl;

}



#####

$$$$$

&&&&&

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

**What to hand in:**

1. Place the word processed document created to answer all of the questions using the filename**{yourname}Lab5.doc or docx** e.g. timwrennlab5.docNote:**If your name is not part of the filename, I will not open the file.**
2. Hand in a print out of your word document.
3. Place the zipped file into the D2L DropBox folder titled Lab 5.