Ekaba Bisong
Programming in C++
University of Calabar

Tutorial Notes #10
May 08, 2015

Decision Making: Equality and Relational Operators

We now introduce a simple version of C++'s ***if statement*** that allows a program to take alternative action based on whether a ***condition*** is true or false. If the condition is true, the statement in the body of the if statement is executed. If the condition is false, the body statement is not executed.

Conditions in if statements can be formed by using the equality operators and relational operators summarized below.

Standard algebraic equality or relational operator	C++ equality or relational operator	Sample C++ condition	Meaning of C++ condition
Relational operators			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y
Equality operators			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y

Using the if Statement

```
1  //
2  //  IfStatement.cpp
3  //  TutorialClass
4  //  Comparing integers using if statements, relational operators
5  //  and equality operators.
6
7  #include <iostream> // allows program to perform input and output
8
9  using std::cout;    // program uses cout
10 using std::cin;     // program uses cin
11 using std::endl;     // program uses endl
12
13 // function main begins program execution
14 int main()
15 {
16     int number1; // first integer to compare
17     int number2; // second integer to compare
18
19     cout << "Enter two integers to compare: "; // prompt user for data
20     cin >> number1 >> number2; // read two integers from user
21
22     if ( number1 == number2 )
23     {
24         cout << number1 << " == " << number2 << endl;
25     }
26
27     if ( number1 != number2 )
28     {
29         cout << number1 << " != " << number2 << endl;
30     }
31
32     if ( number1 < number2 )
33     {
34         cout << number1 << " < " << number2 << endl;
35     }
36
37     if ( number1 > number2 )
38     {
39         cout << number1 << " > " << number2 << endl;
40     }
41
42     if ( number1 <= number2 )
43     {
44         cout << number1 << " <= " << number2 << endl;
45     }
46
47     if ( number1 >= number2 )
48     {
49         cout << number1 << " >= " << number2 << endl;
50     }
51 } // end function main
```

using Directives

Lines 9–11 are using directives that eliminate the need to repeat the std:: prefix as we did in earlier programs. We can now write cout instead of std::cout, cin instead of std::cin and endl instead of std::endl, respectively.

Comparing Numbers

The if statement in lines 22–25 compares the values of variables number1 and number2 to test for equality. If the values are equal, the statement in line 24 displays a line of text indicating that the numbers are equal. If the conditions are true in one or more of the if statements starting in lines 29, 34, 39, 44 and 49, the corresponding body statement displays an appropriate line of text.

Operator Precedence

The figure shows the precedence and associativity of the operators introduced in this note. The operators are shown top to bottom in decreasing order of precedence. All these operators, with the exception of the assignment operator =, associate from left to right. Addition is left-associative, so an expression like x + y + z is evaluated as if it had been written (x + y) + z. The assignment operator = associates from right to left, so an expression such as x = y = 0 is evaluated as if it had been written x = (y = 0), which, as we'll soon see, first assigns 0 to y, then assigns the result of that assignment—0—to x.

Operators				Associativity	Type
()				*[See caution in Fig. 2.10]*	grouping parentheses
*	/	%		left to right	multiplicative
+	-			left to right	additive
<<	>>			left to right	stream insertion/extraction
<	<=	>	>=	left to right	relational
==	!=			left to right	equality
=				right to left	assignment

Excerpt adapted from C++ How to Program, P.J. Deitel, H.M. Deitel. -- 8th ed