|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CS 1400 Lab #13**  **Boolean Expressions**  **Version 1.0**  **Objectives:**  This lab has a few important objectives:   * To help you understand and correctly write ***if***, ***if/else***, ***switch*** and conditional (***?:***) statements. * To help you understand and correctly write ***Boolean expressions*** with the various ***conditional*** (decision) constructs. * To help you use methods to simplify your program. * After completing this lab, you should be able to develop solutions to simple computational problems that use branching, based on the testing of some conditions in the program, and you should be able to write a C# program, using methods, that correctly use the ***if***, ***if/else***, ***switch*** and (***?:***) Conditional statements.   **Study Material**   |  | | --- | | **Diagramming if Statements**  Suppose that you are getting ready to go to work. You get up, shower, dress, eat breakfast, and brush your teeth. Just before you walk out the door, you check the temperature outside. If it is cold outside, you put on a warm jacket. Then you go to work.  In this example, getting up, showering, dressing, eating your breakfast, and brushing your teeth are activities that all happen in a given order. All of the programs that you have written so far work this same way. Each line in the program is executed in order, one line after the other. But, after you checked the temperature, you did something very different. If it was cold outside, you put on your coat.  This kind of decision logic is common in most computer programs. The program makes a decision to do something different, based on some condition. We can draw a picture of this logic using an Activity Diagram similar to the one shown below. When you are asked to draw an Activity Diagram that shows decision logic, your diagram should follow this model.  if diagram |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Boolean Expressions Often times when making a decision, you have to evaluate a number of different conditions, each of which may be ***true*** or ***false***. For example:   * You can sleep in, **if** it is a Saturday **or** a Sunday. * You want the buzzer in your car to sound **if** the keys are in the ignition **and** the door is open. * You want the fire alarm in your house to sound **if** it senses heat **or** smoke. * There is an election coming up. You are allowed to vote **if** you are a citizen **and** you are 18 **or** older.   In C# a conditional expression, such as one of those above, is written as a Boolean expression. When evaluated, a Boolean expression is always either true or it is false. For example, we might ask  ***is a < b?***  Either ***a*** is less than ***b*** (the expression is **true**) or it is not (the expression is **false**). In C#, the symbol ***<*** is what is called the less-than Relational Operator. **Relational (Conditional) Operators** The relational operators in C# are defined in the following table:   |  |  |  |  | | --- | --- | --- | --- | | **Operator** | **Description** | **Example** | **Meaning** | | **==** | equal to | a == b | Is a equal to b? | | **!=** | not equal to | a != b | Is a not equal to b? | | **<** | less than | a < b | Is a less than b? | | **>** | greater than | a > b | Is a greater than b? | | **<=** | less than or equal to | a <= b | Is a less than or equal to b? | | **>=** | greater than or equal to | a >= b | Is a greater than or equal to b? |  **Boolean Operators** In the examples above, we presented situations in which a combination of expressions had to be evaluated in order to make a decision. Combinations of such conditions are formed with Boolean Operators. Boolean operators take Boolean operands (values) (***true*** or ***false***) and yield a Boolean result (value).   |  |  |  | | --- | --- | --- | | **Operator** | **Description** | **Example** | | **!** | Not | ! (a < b) | | **&&** | And | a < b  &&  x < y | | **||** | Or | a > b  ||  x < y |   The definition of Boolean operators is often expressed in what is called a Truth Table.  **NOT Truth Table**   |  |  | | --- | --- | | **Expression** | **! Expression** | | true | false | | false | true |   **AND** Truth Table   |  |  |  | | --- | --- | --- | | **Expression\_1** | **Expression\_2** | **Expression\_1 && Expression\_2** | | true | true | true | | true | false | false | | false | true | false | | false | false | false |   **OR** Truth Table   |  |  |  | | --- | --- | --- | | **Expression\_1** | **Expression\_2** | **Expression\_1 || Expression­\_2** | | true | true | true | | true | false | true | | false | true | true | | false | false | false |  **Short Circuit Evaluation** When Boolean expressions are combined with either the **&&** operator or the **||** operator, C# uses what is called ***Short Circuit Evaluation***. Consider, for example, the expression  ***(x > y) || (x == 5)***  The computer first evaluates the left hand operand ***(x > y)*** of the expression. If this operand is ***true***, then the ***OR*** of the two operands must also be ***true***. Consult the truth tables above if you are not convinced of this. Since the computer knows that the complete expression must be true, it does not even bother to evaluate the right hand operand ***(x == 5).*** If, on the other hand if left-hand operand,  ***(x > y)***  were ***false***, then we don't know if the ***OR*** of the two operands is true until the right operand,  ***(x == 5)***  is evaluated to be ***true*** or ***false***. So, in this case the computer must evaluate both left and right operands of the expression.  Now, consider, the expression  ***(x > y) && (x == 5).***  The computer first evaluates the left hand operand ***(x > y)*** of the expression. If this left-operand is ***false***, then the ***AND*** of the two operands (left and right) must also be ***false***. Consult the truth tables above if you are not convinced of this. Since the computer knows that the complete expression must be false, it does not bother to evaluate the right hand operand ***(x == 5).***    If, on the other hand, if the left-operand,  ***(x > y)***  were ***true***, then we don't know if the ***AND*** of the two operands is true until we know whether or not the right-operand,  ***(x == 5)***  is ***true***. So, in this case the computer must evaluate both operands of the expression. |  |  | | --- | | De Morgan's Law Suppose that you need to test a set of conditions to see if they are all **not** true. For example, you want to go to work **if** it is **not** Saturday, Sunday, or a Holiday. The code to make this test might look something like  **if ( ! (today == "Sunday" || today == "Saturday" || today == "Holiday") )         Console.WriteLine("Go to Work!");**  **if ((today != "Sunday" && today != "Saturday" && today != "Holiday") )**  Sometimes the use of the not operator can be confusing when applied to ***and*** and ***or*** conditions like this. The British Mathemetician Augustus  De Morgan (1806 - 1871) developed a scheme for simplifying expressions that  use the **not**operator. **De Morgan's Law** has two forms:  **! ( A && B)     is the same as (!A || !B)     ! ( A || B)     is the same as (!A && !B)**  Also note that you can simplify the expression  ***! ( A == B)***  by bringing the **!** inside of the parentheses and writing  ***( A != B).***  So, in the formulation of our code, we could equally as well have written  **if ((today != "Sunday" && today != "Saturday" && today != "Holiday") )         Console.WriteLine("Go to Work!");** |  |  | | --- | | Activity Diagrams When you are designing code, that uses decisions, it is often helpful to draw a picture of the logic flow of the program. These pictures are called **Activity Diagrams**. The form of an ***Activity Diagram*** is very specific. ***Rectangles*** are used to describe an activity, step(s) or statement(s) in the program. ***Diamond shapes*** are used to show a ***decision***. Arrows link the activities together. The example below is an Activity Diagram for the "Should I go to work" program in this lab.  http://debryro.tc.uvu.edu/1400/labs/lab13/activity.png  I use a drawing program to create my ***Activity Diagrams***. You can also use PowerPoint, Microsoft Paint, Visio, or any other program that lets you draw the required elements. To submit a diagram, convert your picture into a PDF file. |   **Programming Exercise** |
| **CS 1400 Lab 13:**  **Should I go to work?**  **Introduction**  **The problem**  Your friend Bob has written a program to tell him when he has to go to work. The program prompts him for the day of the week. If day is a week-day (Monday through Friday) are entered, the program tells him to go to work. If the day is a week-end (Saturday or Sunday) are entered, the program announces that there is no work today.  **The code**  Here is the program that Bob wrote.  // File Prolog // Name: Bob LazyBones // CS 1400 - 001 // Project: Work? // Date: June 2013 //   using System;   static class Program {    static void Main()    {       // declare some constants for Saturday and Sunday       const string SAT = "Sat" ;       const string SUN = "Sun";         // declare a variable to hold user's input       string today = “” ;         // prompt the user to enter a day and get the input       Console.Write("Please enter a day of the week, e.g. Tue: ");       today = Console.ReadLine( );         // see if it is a work day       if (!(today == SUN || today == SAT))       {          // it is a workday, display the go to work message          Console.WriteLine("You have to go to work today...");       }       else       {          // its not a workday, display the weekend message          Console.WriteLine("Ahh... the weekend. No work!");       }       Console.ReadLine();    }//End Main() }//End class Program  **Your Assignment**   1. The Boolean expression   **if (!(today == SUN || today == SAT))**  is a little bit awkward and hard to decipher. Use DeMorgan's Law to change the Boolean expression so that it uses an ***AND*** condition instead of an ***OR*** condition. Compile and execute this program with your changes, and make sure that it works correctly. Use this code as the basis for part 2 below.   1. Your friend Bob has decided that he wants you to modify his "Go to work" program so that it works as follows:    * If it is neither Saturday nor Sunday, and it is above freezing, print the message "go to work".    * If it is neither Saturday nor Sunday, but it is not above freezing, print the message "go to work and dress warmly".    * If it is a Saturday or a Sunday, no matter what the temperature is, print the message "yeah! No work today!".   For a bit of brain teaser, arrange your program so that it does not ask for the temperature if you enter Saturday or Sunday.  Now modify you program to use a ***switch*** instead of the ***if-else*** construct you used above.  Now modify your program to use the conditional operator (***?:***) instead of the ***switch*** construct.  Use a menu to allow the user to decide which type of decision expression they would like to test, i.e.  **i)f-else construct**  **s)witch construct**  **c)onditional construct**  **q)uit program**  **File(s) to Submit:**  Place the complete project folder in a zip file and name the zip file lab\_13\_your-initials\_V1.0.zip. For example, I would name my file lab\_13\_RKD\_v1.0.zip. Submit this assignment as Lab #13 on Canvas.   |  |  |  | | --- | --- | --- | |  | **Grading Checklist** |  | | # | Program | C(correct)  X(incorrect) | | 1 | Meets & works to specifications | 6 points | | 2 | Error Free, elegant & efficient | 4 points | | 3 | Pseudo-Code | -3 points | | 4 | Style Guidelines | -2 points | | 6 | Source Files(s) & Formatting | -2 points | | 7 | Project Prolog | -1 points | | 8 | Function Prologs | -1 points | | 9 | Zip Filename | -1 points | | 10 | Lab & Project Names | -1 points | | 11 | Zip File is invalid or will not unzip | Lab = 0 pts | |  | Total Points | 10 | 0-9 | |