Booleans

https://csci-1301.github.io/about#authors

January 11, 2023 (12:00:50 PM)

Contents

This lab serves multiple goals:

- To help you manipulate boolean values,
- To practice boolean operators,
- To understand the concept of *precedence*,
- To practice simple mental calculations.

1 Truth Tables

1. Copy-and-paste the following code into the `Main` method of a new project:

```
Console.WriteLine("Conjunction (and, &&) truth table:"
+ "\n\n && \t||  " + true + "\t| " + false
+ "\n--------||--------------|--------"
+ "\n" + true + "\t||  " + (true && true) + "\t| " + (true && false)
+ "\n" + false + "\t||  " + (false && true) + "\t| " + (false && false)
+ "\n\n*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*\n");

Console.WriteLine("Negation (not, !) truth table:"
+ "\n\n value \t||  ! "
+ "\n--------||----------"
+ "\n" + true + "\t||  " + false
+ "\n" + (!true) + "\t||  " + (!false)
+ "\n\n*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*\n");
```

2. Compile and execute it. This should display to the screen truth tables for conjunction (and, `&&`) and negation (not, `!`).
3. Make sure you understand both the code and its output.
4. After the truth table for the negation, write code to display truth tables for

   a) the binary operators disjunction (or, `||`),

   b) identity (equality, `==`) and
   c) difference (inequality, `!=`).

   Normally, using the find-and-replace feature of your IDE should make this a quick and easy task.

5. You can make sure you completed this exercise correctly by checking that your output match the truth tables on wikipedia for disjunction[1] and equality[2]. For inequality, in this case check against the table for exclusive disjunction[3]. Exclusive disjunction (XOR) is conceptually different than inequality, but has the same truth table.

2 Precedence and Order of Evaluation

2.1 Reading and Understanding

If you read the documentation on operator precedence[4], you will see that operators are evaluated in a particular order. From higher precedence (that is, evaluated first) to lower precedence (that is, evaluated last), this order is: `! (* / %) (+ -) (< > <= >=) (== !=) && ||`. Inside each group in parenthesis, operations are evaluated from left to right.

So that, for instance, `! true || false && 3 * 2 == 6` will be evaluated as

Operation		Result	Op.
`! true || false && 3 * 2 == 6`	⇒	`false || false && 3 * 2 == 6`	`!`
`false || false && 3 * 2 == 6`	⇒	`false || false && 6 == 6`	`*`
`false || false && 6 == 6`	⇒	`false || false && true`	`==`
`false || false && true`	⇒	`false || false`	`&&`
`false || false`	⇒	`false`	`||`

Note that an expression like `!3 > 2` does not make any sense: C# would try to take the negation of `3` (since `!` has a higher precedence than `>`), but you cannot negate the truth value of an integer! Along the same lines, an expression like `false * true` does not make any sense: you can not multiply booleans (what would be "true times false"?)! Similarly, `3 % false` will cause an error: can you see why? These are all examples of "illegal" expressions.

2.2 Computing Simple Boolean Expressions

Evaluate the following expressions. Try to do this "by hand," and write your answers down on paper.

- `true && false || true`
- `!true && false`
- `false || true && !false`
- `false == !true || false`
- `!(true || false || true && true)`
- `!(true || false) && (true && !false)`
- `!true || false && (true && !false)`
- `true != !(false || true)`

[1] https://en.wikipedia.org/wiki/Truth_table#Logical_disjunction_(OR)
[2] https://en.wikipedia.org/wiki/Truth_table#Logical_equality
[3] https://en.wikipedia.org/wiki/Truth_table#Exclusive_disjunction
[4] https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/#operator-precedence

2.3 Computing Expressions Involving Booleans and Numerical Values

For each of the following expressions, decide if it is “legal” or not. If it is, give the result of its evaluation.

- `3 > 2`
- `2 == 4`
- `3 >= 2 != false`
- `3 > false`
- `true && 3 + 5 * 8 == 43`
- `3 + true != false`