Booleans

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

June 6, 2021 (07:51:31 PM)

Contents

This lab has four core goals: - To help you manipulate boolean values, - To practise boolean operators, - To understand the concept of *precedence*, - To practise simple calculation mentally.

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 ! \t "
+ "\n---------||------"
+ "\n" + true+ "\t ||\t" + false
+ "\n" + (!true)+ "\t ||\t" + (!false)
+ "\n\n*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*\n");
```

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

4. 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].

2 Precedence and Order of Evaluation

2.1 Reading and Understanding

If you read the documentation on operator precedence[3], you will see that operators are evaluated in a particular order. That is,

!	is evaluated before
*, /, and %	which are evaluated before
+ and -	which are evaluated before
<, >, <=, and >=	which are evaluated before
== and !=	which are evaluated before
&&	which is evaluated before
||	which comes last.

and that within those groups, operations are evaluated from left to right.

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

`! true || false && 3 * 2 == 6`	$\Rightarrow$	`false || false && 3 * 2 == 6`
`false || false && 3 * 2 == 6`	$\Rightarrow$	`false || false && 6 == 6`
`false || false && 6 == 6`	$\Rightarrow$	`false || false && true`
`false || false && true`	$\Rightarrow$	`false || false`
`false || false`	$\Rightarrow$	`false`

Note that an expression like `!3 > 2` doesn't make any sense: C# would try to take the negation of `3`, but you can't negate the truth value of an integer! Along the same lines, an expression like `false * true` doesn't 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)`

---

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

- **true** != !(**false** || **true**)

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**