Booleans

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

June 5, 2021 (09:04:56 PM)

Contents

1 Truth Tables

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

```
/*
 * We have two boolean values: true and false.
 * We can use the constant "true" and "false",
 * we can also declare constants with the same value,
 * but a shorter name:
 */
const bool t = true;
const bool f = false;

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

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

Compile and execute it.

This should display to the screen truth tables for conjunction (and, `&&`) and negation (not, `!`). Next, write code that will display truth tables for the binary operators disjunction (or, `||`), identity (equality, `==`) and difference (inequality, `!=`).

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

2 Precedence and Order of Evaluation

2.1 Reading and Understanding

If you look at https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/#operator-precedence, you will see that

`!`	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`	⇒	`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` 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't multiply booleans! 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 (where `t` stands for `true`, and `f` for `false`). Try to do this "by hand," and write your answers down on paper.

- `t && f || t`
- `!t && f`
- `f || t && !f`
- `f == !t || f`
- `!(t || f || t && t)`
- `!(t || f) && (t && !f)`
- `!t || f && (t && !f)`
- `t != !(f || t)`

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 != f`
- `3 > f`
- `t && 3 + 5 * 8 == 43`
- `3 + t != f`