Boolean and Basic Conditional Statements

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

June 4, 2021 (07:52:10 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 conjection (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`

3 Basic Conditional Statements

Read all the instructions in this part before starting to type code. Create a new project, and write portions of code that perform the following:

1. Ask the user for an integer, and display on the screen "You were born after me" if the number is strictly greater than your year of birth.
2. Ask the user for an integer, and display on the screen "Between -1 and 1" if the number is greater than or equal to -1 and less than or equal to 1.
3. Ask the user for an integer, and display on the screen "Not between -1 and 1" if the number is greater than 1 or less than -1.
4. Ask the user for an integer, and display on the screen "Odd" or "Even", depending if the number is odd or even.
5. Ask the user for an integer, and display on the screen "Negative" if the integer is negative, "Positive" if the integer in positive, and nothing if the integer is 0.
6. Ask the user for an integer, and display on the screen "positive and odd" if the number is positive and odd, "positive and even" if the number is positive and even, "negative and odd" if the number is negative and odd, "negative and even" if the number is negative and even, and "You picked 0" if the number is 0.

For each of those questions, write *on paper* whenever you should use `if`, `if-else`, `if-else-if` or nested conditional structures, and what the condition(s) should be. Once you feel confident, write the code in your IDE, and then test it intensively: enter all kinds of values (positive and odd, negative and even, 0, and remember that 0 is even[1], etc.) and make sure that what is displayed on the screen is always correct.

[1] https://en.wikipedia.org/wiki/Parity_of_zero