CSCI 1301 – Lab 16

October 9, 2018

1 Truth Tables

Copy-and-paste the following code in 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("Conjonction (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.

Then, write the code that will display on the screen the truth tables for the binary operators disjunction (or, `||`), identity (equality, `==`) and difference (inequality, `!=`).

Normally, using the find-and-replace feature of VS should make this task easy and fast.

2 Precedence and Order of Evaluation

2.1 Reading and Understanding

If you look at https://docs.microsoft.com/en-us/cpp/c-language/precedence-and-order-of-evaluation, 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 == 5` 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 an integer! Along the same way, an expression like `false * true` doesn't make any sense: you can't multiply booleans! Similarly, `3 % false` will cause an error: can you decide why?

2.2 Computing Simple Boolean Expressions

Evaluate the following expressions (where `t` stands for `true`, and `f` for `false`):

- `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 expression, decide if they are "legal" or not. If they are, give the result of their evaluation.

- `3 > 2`
- `2 == 4`
- `3 >= 2 != f`
- `3 > f`
- `t && 3 + 5 * 8 == 43`
- `3 + t != f`

3 A Guessing Game (Optional)

Write a program that

1. Store your favorite number in a variable

2. Ask the user to enter a numerical value, and store the user's answer in a variable.
3. With an `if` statement, display on the screen "You guessed correctly" if the number entered by the user is your favorite number.
4. Once this part of the program works, add an `if` statement that displays on the screen "Too high!" if the number entered by the user is strictly greater than your favorite number.
5. Once this part of the program works, add an `if` statement that displays on the screen "Too low!" if the number entered by the user is strictly lower than your favorite number.
6. Once this part of the program works, add an `if` statement that displays on the screen "You found a multiple of my favorite number!" if the number entered by the user is a multiple of your favorite number, but different from it.