CP143 - LogiC 101

Logical operators (refer para. 4.10, last couple of slides)

1. When **evaluating any number** (as if it were a condition) as being true or false the following rule applies: **Everything except 0 is true**.

Examples:

`if (0.1) or while (345.7)` or `for (i=0; 234; i++)`. All these conditions will evaluate as being true.

`if(0)` or `while(0)` or `for(i=0; 0 ; i++)`. All the conditions will evaluate as being false.

2. When **evaluating any condition** (equality, inequality, bigger than, smaller than, or a number as above) the following rule applies: **If the comparison is true, the result is 1. If the condition is false the result is 0.**

Examples:

`printf("%d",5 > 6)` will print "0", the comparison (condition) returns 0 (false) to the print function.

`if(5 > 6)` : the comparison (condition) returns 0 (false) to the if control.

`while (0 == 3)` : the equality test (condition) returns 0 (false) to the while control.

Assume x = 5. `if(x == 5)` : the equality test (condition) returns 1 (true) to the if control.

Assume x =5. `if(x < 5)` : the smaller than test (condition) returns 0 (false) to the if control.

Assume x=5. `printf("%d", x == 3)` will print "0", the equality test (condition) returns 0 (false) to the print function.

Note: Assignment returns the assignee, so `while (x=5)` *assigns the value 5 to x and returns 5 (nonzero=true) to the while control.*

Thus `printf("%d", x = 3)` *will print "3", the assignment returns 3 (nonzero = true) to the print function.*

Note: Functions that is said to return" true" or "false", in actual fact returns an integer 1 or 0 respectively.

Thus a function that checks if an integer number is positive might have prototype:

`int isNumberPositive (int Number);` and return a 1 (for true) or a 0 (for false)

3. The **AND** operator (condition1 && condition2) is like saying "is **both** condition1 **and** condition2 true?"

The following truth table is used when evaluating &&

Condition1	Condition2	&& result
0 (false)	0 (false)	0 (false)
0 (false)	nonzero (true)	0 (false)
nonzero (true)	0 (false)	0 (false)
nonzero (true)	nonzero (true)	nonzero (true)

Examples: (assume x = 5, y =10)

`if(x == 2 && y == 7)` : reduces to `if(0 && 0)` , which returns 0 (false) to the if control.

`if(x == 2 && y == 10)` : reduces to `if(0 && 1)` , which returns 0 (false) to the if control.

`if(x == 5 && y == 7)` : reduces to `if(1 && 0)` , which returns 0 (false) to the if control.

`if(x == 5 && y == 10)` : reduces to `if(1 && 1)` , which returns 1 (true) to the if control.

4. The **OR** operator (condition1 || condition2) is like saying "is **any** of condition1 **or** condition2 true?"

The following truth table is used when evaluating ||

Condition1	Condition2	|| result
0 (false)	0 (false)	0 (false)
0 (false)	nonzero (true)	nonzero (true)
nonzero (true)	0 (false)	nonzero (true)
nonzero (true)	nonzero (true)	nonzero (true)

Examples: (assume x = 5, y =10)

`if(x == 2 || y == 7)` : reduces to `if(0 || 0)` , which returns 0 (false) to the if control.

`if(x == 2 || y == 10)` : reduces to `if(0 || 1)` , which returns 1 (true) to the if control.

`if(x == 5 || y == 7)` : reduces to `if(1 || 0)` , which returns 1 (true) to the if control.

`if(x == 5 || y == 10)` : reduces to `if(1 || 1)` , which returns 1 (true) to the if control.

5. The **NOT** (! condition) operator simply changes a 0 (false) to a 1 (true) and vice versa.

Examples: (assume x = 5, y =10)

`if(!(x == 2 || y == 7))` : reduces to `if(!(0 || 0))` , which reduces to `if(!(0))` and returns 1 (true) to the if control.

!(0.3) returns 0 (false)

By mjbooysen – Group 4