Computer Science 101: Introduction to Java and Algorithms

Section 2: Control Statements - Selections

Erudition Labs

Computer Science 101: Introduction to Java and Algorithms

April 30, 2019

Contents

1 Relational and Logical Operators (Video Series Lecture 7)

In this part of the series, we introduce the relational operators. Interestingly enough, they are all the same operators that exist in basic mathematics. However, this would be our first experience with logic. By that I mean that it will be helpful here to start thinking of things in terms of true and false. For example, $1 < 2$ is a true statement. Whereas $5 > 1$ is a false statement. That's all there really is to say about it.

1.1 The Relational Operators

Math Symbol	Programming Symbol	Definition	Example	Value
$>$	$>$	is greater than	$5 > 4$	true
$\geq$	$>=$	is greater than or equal to	$5 >= 5$	true
$<$	$<$	is less than	$4 < 10$	false
$\leq$	$<=$	is less than or equal to	$5 <= 10$	true
$=$	$==$	is equal to	$8 == 4$	false
$\neq$	$!=$	is not equal to	$8 != 4$	true

Take note that in programming, we combine symbols such as $<=$ instead of $\leq$, why? Well simply because, do you see $\leq$ on your keyboard? If you do, then I have no idea what keyboard you are using.

A more important one to note is the 'equal to' operator. In mathematics we use $=$ as both assignment and to express an equality relation. In fact, we generally use them as one in the same. However, a computer needs to make a distinction due to additional things the compiler must do. In programming, we use $=$ as assignment. When we say

```
int a = 5;
```

what we are actually saying is, hey compiler, create a variable in memory with enough space to store an integer and store the value 5 there. Now take the case,

```
int a = 5;
int b =0;
a = b;
```

Here we tell the compiler to again create the variable a and store the value 5 and then create the variable b and store the value 0. Then we are assigning the value that is stored in the variable a into the variable b. That is, we are copying the value stored in variable a into b. So when we try to ask the program, "is a equal to b" we cannot use the $=$ symbol. If we did, the computer would think that we are trying to assign something. Instead we use the symbols $==$ to ask if something is equal. This will make more sense when we try to use these operators.

1.2 Logical Operators

The logical operators are directly rooted into propositional logic. What's great is that we use these everyday, often without even realizing it.

Logic	Programming Symbol	Definition	Example	Value
$\&(Conjunction)$	&&	AND	(5 == 5) && (5! = 10)	true
$V(Disjunction)$	||	OR	(3 == 3) || (5 == 3)	true
$\neg(Negation)$	!	NOT	!(5! = 5)	true

In programming (and logic) you can combine these logical operations to evaluate to some boolean value.

1.2.1 AND

In propositional logic, AND is a conjunction of two statements. We as humans have learned to communicate in this way as well, so it is often helpful to say it as a sentence to yourself and see if it makes sense. For example, "Fire is hot and fire burns wood". The two statements, "Fire is hot" and "fire burns wood" are both true about fire. We know this to be true because of our senses. since the two statements are true, then the conjunction of the two statements must also be true.

Let us look at the truth table for the conjunction AND (I will be using the programming symbols for the logical operators),

A	B	A && B
false	false	false
false	true	false
true	false	false
true	true	true

As you can see, the only time the AND operator is true is when both statements are true. If you really think about it, this makes sense when you speak. The next table will just be in plain English.

Statement A	Statement B	A && B	value	Explanation
Fire is cold	Fire is wet	Fire is cold and Fire is wet	false	We know fire isn‘t cold or wet so it also can‘t be both cold and wet
Fire is cold	Fire is hot	Fire is cold and Fire is hot	false	Fire is hot, but we know it isn‘t cold, so the sentence really makes no sense
Fire is hot	Fire is cold	Fire is hot and Fire is cold	false	moving the true part of the sentence to the other side doesn‘t change the fact that fire isn‘t both hot and cold
Fire is hot	Fire burns wood	fire is hot and fire burns wood	true	we know that fire is hot and that fire burns wood, so fire must be hot and also it must burn wood, which makes sense.

The next table, I will use conjunctions of relation statements, similar to what we would find in programming. In this table we will use a variable, i, let $i = 5$

A	B	A && B	Value
$(i == 3)$	$(i >= 10)$	$(i == 3)$ && $(i >= 10)$	false
$(i < 3)$	$(i <= 5)$	$(i < 3)$ && $(i <= 5)$	false
$(i == 5)$	$(i\,! = 5)$	$(i == 5)$ && $(i\,! = 5)$	false
$(i >= 5)$	$(i <= 5)$	$(i >= 5)$ && $(i <= 5)$	true

Note that these two tables both correspond to the AND truth table. Now I would like to do a more complicated example with conjunctions of relational statements. Let $i = 5$ again. Now let us evaluate

$(((i\,! = 5)$ && $(i == 10))$ && $i <= 5)$

First identify the order, we must do the inner parenthesis first.

$((i\,! = 5)$ && $(i == 10))$

Now we need to identify the statements

$(i\,! = 5)$ and $(i == 10)$

Now evaluate these statements

$(i\,! = 5)$ is false

$(i == 10)$ is false

So, $((i\,! = 5)$ && $(i == 10))$ is equivalent to ((false) && (false)).

We can consult the Truth table to see that $((i \,!= 5)$ && $(i == 10))$ is false

Now we have a conjunction of (false && $(i <= 5)$)
we know that $(i <= 5)$ is true
So, we have reduced $(((i \,!= 5)$ && $(i == 10))$ && $i <= 5)$ to (false && true)
We can again consult the truth table to see that (false && true) is false.

Therefore, $(((i \,!= 5)$ && $(i == 10))$ && $i <= 5)$ evaluates to false.

1.2.2 OR

In propositional logic, OR is the disjunction of two statements. For example, "It is sunny" or "it is cloudy". It is one or the other. The truth table looks like this:

A	B	A || B
false	false	false
false	true	true
true	false	true
true	true	true

As you can see, the only time that the whole disjunction is false is when both statements are false. This makes sense when we talk as well. When using OR we are speaking in terms of one or the other or both. When it comes to OR, we only need one statement to be true to make the whole disjunction true. Here is an English truth table.

Statement A	Statement B	A || B	value	Explanation
Fire is cold	Fire is wet	Fire is cold or Fire is wet	false	We know fire isn‘t cold or wet so it is neither.
Fire is cold	Fire is hot	Fire is cold or Fire is hot	true	Fire is hot, but we know it isn‘t cold, but since fire is hot, we know it is at least one of those two options
Fire is hot	Fire is cold	Fire is hot or Fire is cold	true	moving the true part of the sentence to the other side doesn‘t change the fact that fire is hot even if it isn‘t cold
Fire is hot	Fire burns wood	fire is hot or fire burns wood	true	we know that fire is hot and that fire burns wood, so fire must be hot and also it must burn wood, so take your pick.

Now again, but using relational statements that you would see in programming. Let $i = 5$

A	B	A || B	Value
$(i == 3)$	$(i >= 10)$	$(i == 3) \mid\mid (i >= 10)$	false
$(i < 3)$	$(i <= 5)$	$(i < 3) \mid\mid (i <= 5)$	true
$(i == 5)$	$(i\ != 5)$	$(i == 5) \mid\mid (i\ != 5)$	true
$(i >= 5)$	$(i <= 5)$	$(i >= 5) \mid\mid (i <= 5)$	true

Lets use the same statement from the above subsection, but swap out the AND‘s with OR‘s and see what happens to the value. Let us evaluate:

$(((i\ != 5) \mid\mid (i == 10)) \mid\mid i <= 5)$

First identify the order, we must do the inner parenthesis first.

$((i\ != 5) \mid\mid (i == 10))$

Now we need to identify the statements

$(i\ != 5)$ or $(i == 10)$

Now evaluate these statements

$(i\ != 5)$ is false

$(i == 10)$ is false

So, $((i\ != 5) \mid\mid (i == 10))$ is equivalent to ((false) || (false)).

We can consult the Truth table to see that $((i\ != 5) \mid\mid (i == 10))$ is false

Now we have a disjunction of (false || (i <= 5))

we know that (i <= 5) is true

So, we have reduced (((i ! = 5) || (i == 10)) || i <= 5) to (false || true)

We can again consult the truth table to see that (false || true) is true.

Therefore, (((i ! = 5) || (i == 10)) || i <= 5) evaluates to true.

1.2.3 NOT

In propositional logic, NOT is the equivalent to the negation. Another way to think of this is “the opposite of“. Obviously not true is false and not false is true, so the truth table looks like:

A	!A
false	true
true	false

It‘s sort of like saying a statement and then pausing at the end and saying “just kidding¡‘. Like this, “Fire is cold...JUST KIDDING“, we just took a false statement and made it true.

Statement A	!A	value	Explanation
Fire is cold	Fire is cold...NOT	true	We know fire is not cold, but we want the opposite value
Fire is hot	Fire is hot...NOT	false	We know fire is hot, but we want the opposite value

Let‘s see if we can make our example using the OR‘s false by using NOT

(((i ! = 5) || (i == 10)) || i <= 5)

First identify the order, we must do the inner parenthesis first.

((i ! = 5) || (i == 10))

Now we need to identify the statements

(i ! = 5) or (i == 10)

Now evaluate these statements

(i ! = 5) is false

(i == 10) is false

So, ((i ! = 5) || (i == 10)) is equivalent to ((false) || (false)).

We can consult the Truth table to see that ((i ! = 5) || (i == 10)) is false

Now we have a disjunction of (false || (i <= 5))

we know that (i <= 5) is true

So, we have reduced (((i ! = 5) || (i == 10)) || i <= 5) to (false || true)

We can again consult the truth table to see that (false || true) is true.

Therefore, (((i ! = 5) || (i == 10)) || i <= 5) evaluates to true.

To make the whole disjunction false, we need to make the second statement false. So, what if we change the expression to

(((i ! = 5) || (i == 10)) || !(i <= 5))

First identify the order, we must do the inner parenthesis first.

((i ! = 5) || (i == 10))

Now we need to identify the statements

(i ! = 5) or (i == 10)

Now evaluate these statements

(i ! = 5) is false

(i == 10) is false

So, ((i ! = 5) || (i == 10)) is equivalent to ((false) || (false)).

We can consult the Truth table to see that ((i ! = 5) || (i == 10)) is false

Now we have a disjunction of (false || !(i <= 5))

we know that (i <= 5) is true, so !(i <= 5) must be false

So, we have reduced (((i ! = 5) || (i == 10)) || !(i <= 5)) to (false || false)

We can consult the OR truth table to see that false || false is false.

2 True and False (Video Series Lecture 8 and 9)

3 if Statement (Video Series Lecture 10 and 11)

4 if else and if else if Statements (Video Series Lecture 12 and 13)

5 switch Statement (Video Series Lecture 14 and 15)