

## **Lecture 3: Flow of Control**

Curtin FIRST Robotics Club (FRC) Pre-season Training

---

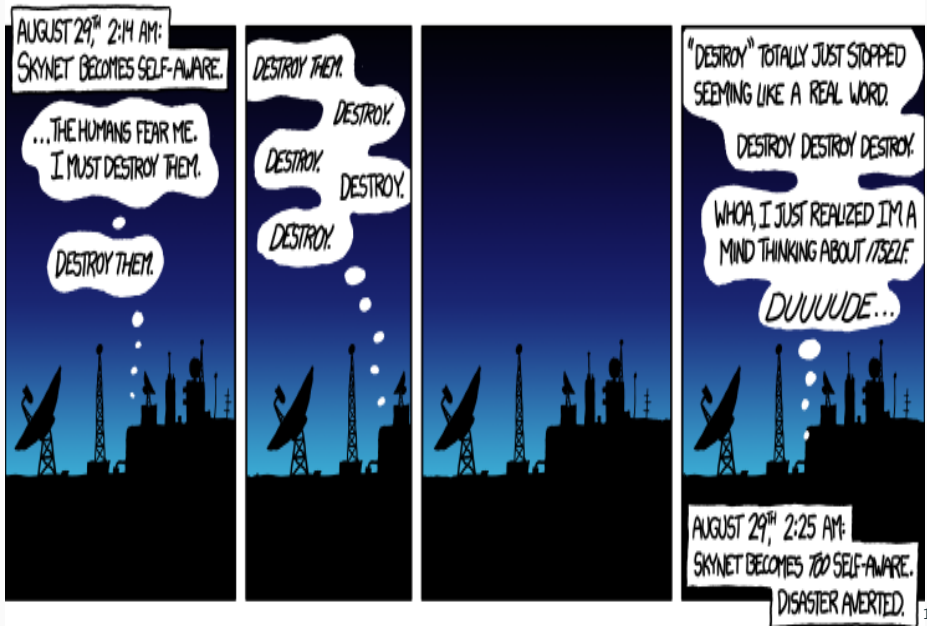
Scott Day

265815F@curtin.edu.au

October 29, 2016

Curtin University

## Insert Mandatory Programming Joke



1. Conditionals
2. Loops
3. Nested Control Structures

## Conditionals

---

t

t

## Relational Operators

Operator	Shorthand	Meaning
>		Greater than
>=	$\geq$	Greater than or equal to
<		Less than
<=	$\leq$	Less than or equal to
==		Equal to
!=	$\neq$	Not equal to

Operator	Meaning
&&	and
	or
!	not



# Truth Tables

Operators return true or false, according to the rules of logic:

<b>a</b>	<b>b</b>	<b>a &amp;&amp; b</b>
true	true	true
true	false	false
false	true	false
false	false	false

<b>a</b>	<b>b</b>	<b>a    b</b>
true	true	true
true	false	true
false	true	true
false	false	false

<b>a</b>	<b>!a</b>
true	false
false	true

Examples using logical operators (assume  $x = 6$  and  $y = 2$ ):

# Truth Tables

Operators return true or false, according to the rules of logic:

<b>a</b>	<b>b</b>	<b>a &amp;&amp; b</b>
true	true	true
true	false	false
false	true	false
false	false	false

<b>a</b>	<b>b</b>	<b>a    b</b>
true	true	true
true	false	true
false	true	true
false	false	false

<b>a</b>	<b>!a</b>
true	false
false	true

Examples using logical operators (assume  $x = 6$  and  $y = 2$ ):

$!(x > 2)$

# Truth Tables

Operators return true or false, according to the rules of logic:

<b>a</b>	<b>b</b>	<b>a &amp;&amp; b</b>
true	true	true
true	false	false
false	true	false
false	false	false

<b>a</b>	<b>b</b>	<b>a    b</b>
true	true	true
true	false	true
false	true	true
false	false	false

<b>a</b>	<b>!a</b>
true	false
false	true

Examples using logical operators (assume  $x = 6$  and  $y = 2$ ):

$!(x > 2)$                       false

# Truth Tables

Operators return true or false, according to the rules of logic:

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

a	b	a    b
true	true	true
true	false	true
false	true	true
false	false	false

a	!a
true	false
false	true

Examples using logical operators (assume  $x = 6$  and  $y = 2$ ):

$!(x > 2)$                       false  
 $(x > y) \ \&\& \ (y > 0)$

# Truth Tables

Operators return true or false, according to the rules of logic:

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

a	b	a    b
true	true	true
true	false	true
false	true	true
false	false	false

a	!a
true	false
false	true

Examples using logical operators (assume  $x = 6$  and  $y = 2$ ):

$!(x > 2)$	false
$(x > y) \ \&\& \ (y > 0)$	true

# Truth Tables

Operators return true or false, according to the rules of logic:

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

a	b	a    b
true	true	true
true	false	true
false	true	true
false	false	false

a	!a
true	false
false	true

Examples using logical operators (assume  $x = 6$  and  $y = 2$ ):

<code>!(x &gt; 2)</code>	false
<code>(x &gt; y) &amp;&amp; (y &gt; 0)</code>	true
<code>(x &lt; y) &amp;&amp; (y &gt; 0)</code>	

# Truth Tables

Operators return true or false, according to the rules of logic:

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

a	b	a    b
true	true	true
true	false	true
false	true	true
false	false	false

a	!a
true	false
false	true

Examples using logical operators (assume  $x = 6$  and  $y = 2$ ):

$!(x > 2)$	false
$(x > y) \ \&\& \ (y > 0)$	true
$(x < y) \ \&\& \ (y > 0)$	false

# Truth Tables

Operators return true or false, according to the rules of logic:

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

a	b	a    b
true	true	true
true	false	true
false	true	true
false	false	false

a	!a
true	false
false	true

Examples using logical operators (assume x = 6 and y = 2):

!(x > 2)	false
(x > y) && (y > 0)	true
(x < y) && (y > 0)	false
(x < y)    (y > 0)	true



# Truth Tables

Operators return true or false, according to the rules of logic:

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

a	b	a    b
true	true	true
true	false	true
false	true	true
false	false	false

a	!a
true	false
false	true

Examples using logical operators (assume  $x = 6$  and  $y = 2$ ):

$!(x > 2)$	false
$(x > y) \ \&\& \ (y > 0)$	true
$(x < y) \ \&\& \ (y > 0)$	false
$(x < y) \    \ (y > 0)$	true

Boolean variables can be used directly in these expressions, since they hold `true` and `false` values.

Funny enough, any kind of value can be used in a Boolean expression due to a quirk C++ has:

`false` is represented by a value of 0 and anything that is not 0 is `true`.

So, `"Hello, world!"` is `true`, `2` is `true`, and any `int` variable holding a non-zero value is `true`. This means `!x` returns `false` and `x && y` returns `true`!

The `if` condition has the form:

```
1 if(condition)
2 {
3     statement1
4     statement2
5     ...
6 }
```

```
1 if(condition)
2     statement
```

The `if-else` form is used to decide between two sequences of statements referred to as blocks:

```
1 if(condition)
2 {
3     statementA1
4     statementA2
5     ...
6 }
7 else
8 {
9     statementB1
10    statementB2
11    ...
12 }
```

```
1 if(condition)
2     statementA1
3 else
4     statementB1
```

The `else if` is used to decide between two or more blocks based on multiple conditions:

```
1  if(condition1)
2  {
3      statement1
4      statement2
5      ...
6  }
7  else if(condition2)
8  {
9      statementB1
10     statementB2
11     ...
12 }
```

# If Example

Here is an example using these control structures:

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int x = 6;
7     int y = 2;
8
9     if(x > y)
10         cout << "x is greater than y" << endl;
11     else if(y > x)
12         cout << "y is greater than x" << endl;
13     else
14         cout << "x and y are equal" << endl;
15
16     return 0;
17 }
```

The output of this program is x is greater than y. If we set the lines 6 and 7 to `int x = 2;` and `int y = 6;` respectively, then the output is y is greater than x.

If we replace the lines with `int x = 2` and `int y = 2;` then the output is x and y are equal.

The switch-case is another conditional structure that may or may not execute certain statements. However, the switch-case has peculiar syntax and behavior:

```
1 switch(expression)
2 {
3     case constant1:
4         statementA1
5         statementA2
6         ...
7         break;
8     case constant2:
9         statementB1
10        statementB2
11        ...
12        break;
13    ...
14    default:
15        statementZ1
16        statementZ2
17        ...
18 }
```

# If Example

Here is an example using switch-case:

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int x = 6;
7
8     switch(x)
9     {
10         case 1:
11             cout << "x is 1" << endl;
12             break;
13         case 2:
14         case 3:
15             cout << " x is 2 or 3" << endl;
16             break;
17         default:
18             cout << "x is not 1, 2, or 3" << endl;
19     }
20
21     return 0;
22 }
```

This program will print x is not 1, 2, or 3. If we replace line 6 with

```
int x = 2;
```



## Loops

---

t

t

t

## Nested Control Structures

---

t

t

# Nested Loops

t



