# Lecture 3: Flow of Control

Curtin FIRST Robotics Club (FRC) Pre-season Training

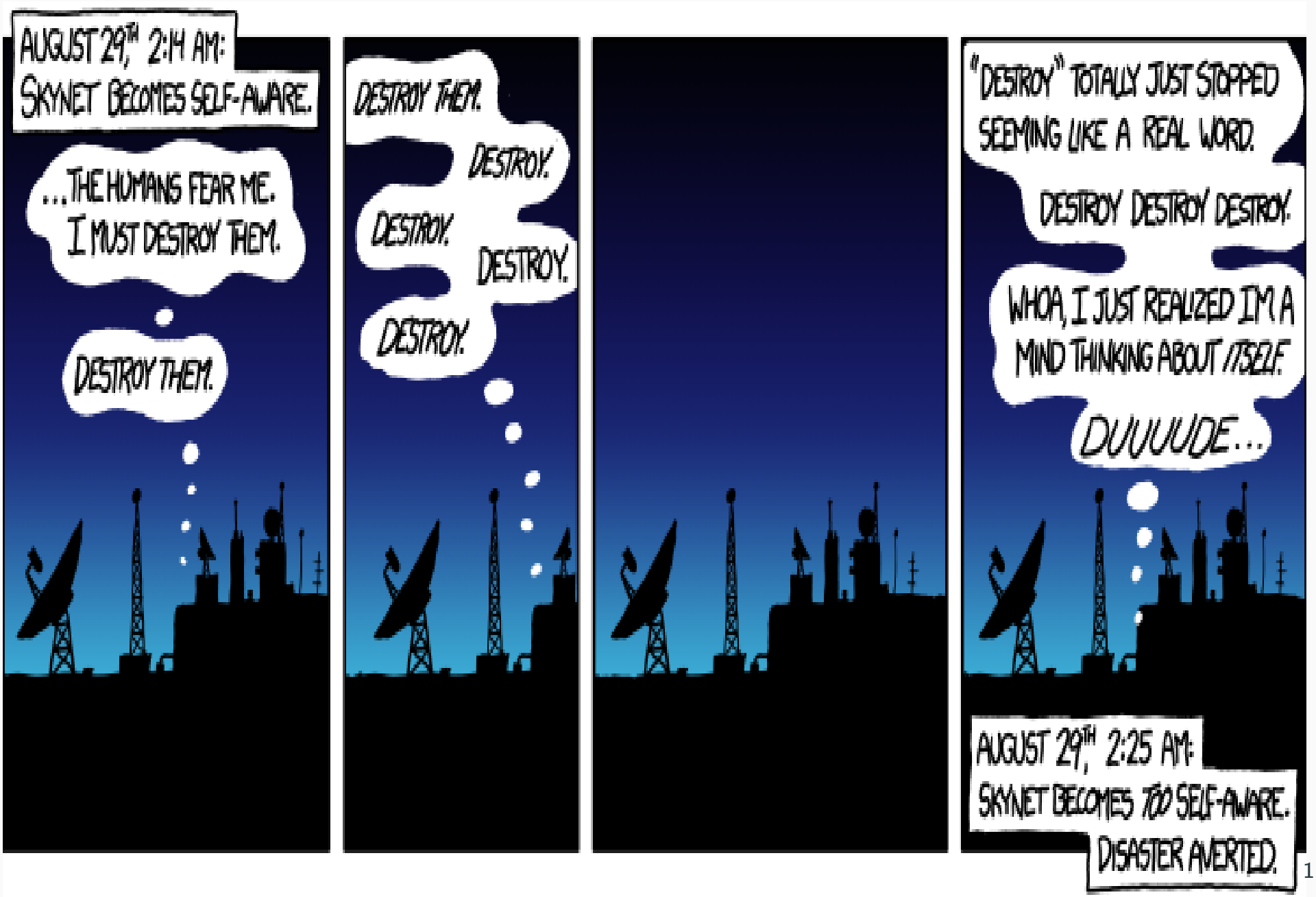
Scott Day

265815F@curtin.edu.au

October 29, 2016

Curtin University

# Insert Mandatory Programming Joke



# Table of contents

# Conditionals

# Conditionals

```
t
```

# Operators

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 |

# Logical Operators

| Operator | Meaning |
|---|---|
| && | and |
| \|\| | or |
| ! | not |

# 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`):

# 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)
```

# 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

# 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`):

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

# 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
```

# C++ Boolean

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`!

# If, If-Else and Else-If

t

# Switch-Case

t

# Loops

# Loops

t

# While and Do-While

t

# For

t

# Nested Control Structures

# Nested Control Structures

t

# Nested Conditionals

```
t
```

# Nested Loops

t

# References I