Boolean Expressions

Lecture 9
Sections 2.11, 4.1, 4.7

Robb T. Koether

Hampden-Sydney College

Mon, Sep 17, 2018

Outline

Boolean Variables and Operators

- A boolean variable may take on one of only two boolean values
  - true
  - false
- There are four standard boolean operators
  - and
  - or
  - not
  - exclusive or (xor)
- A boolean expression is an expression which takes on a boolean value (whether or not its components are boolean).
  - $x > 2$
  - $x \leq 0$ or $x \geq 1$

Logical "And"

- If p and q are boolean expressions, then the expression "p and q" is true if and only if p is true and q is true.

p	q	p and q
T	T	
T	F	
F	T	
F	F	

Logical "And"

- If p and q are boolean expressions, then the expression "p and q" is true if and only if p is true and q is true.

p	q	p and q
T	T	T
T	F	F
F	T	F
F	F	F

Logical "Or"

- If p and q are boolean expressions, then the expression

  "p or q"

  is true if and only if p is true or q is true.

p	q	p or q
T	T	
T	F	
F	T	
F	F	

Logical "Or"

- If p and q are boolean expressions, then the expression

  "p or q"

  is true if and only if p is true or q is true.

p	q	p or q
T	T	T
T	F	T
F	T	T
F	F	F

Logical “Not”

- If p is a boolean expression, then the expression

  “not p”

  is true if and only if p is false, i.e., if p is not true.

p	not p
T	
F	

Logical "Not"

- If p is a boolean expression, then the expression

$$\text{“not } p\text{”}$$

is true if and only if p is false, i.e., if p is not true.

p	not p
T	F
F	T

Logical "xor"

- If p and q are boolean expressions, then the expression

  "p xor q"

  (exclusive or) is true if and only if p is true and q is false or p is false and q is true.
- Equivalently, p or q is true, but not both.

p	q	p xor q
T	T	
T	F	
F	T	
F	F	

Logical “xor”

- If p and q are boolean expressions, then the expression

  “p xor q”

  (exclusive or) is true if and only if p is true and q is false or p is false and q is true.
- Equivalently, p or q is true, but not both.

p	q	p xor q
T	T	F
T	F	T
F	T	T
F	F	F

Truth Tables

- A truth table for a Boolean expression is a table that shows every possible combination of boolean values of the variables, together with the boolean values of the expression.
- If there are n variables, then there are 2^n combinations of boolean values.

Example: Truth Table

- Truth Table for "*p* and not (*q* or *r*)."

p	q	r	q or r	not (q or r)	p and not (q or r)
T	T	T			
T	T	F			
T	F	T			
T	F	F			
F	T	T			
F	T	F			
F	F	T			
F	F	F			

Example: Truth Table

- Truth Table for "*p* and not (*q* or *r*)."

p	q	r	q or r	not (q or r)	p and not (q or r)
T	T	T	T		
T	T	F	T		
T	F	T	T		
T	F	F	F		
F	T	T	T		
F	T	F	T		
F	F	T	T		
F	F	F	F		

Example: Truth Table

- Truth Table for "*p* and not (*q* or *r*)."

p	*q*	*r*	*q* or *r*	not (*q* or *r*)	*p* and not (*q* or *r*)
T	T	T	T	F	
T	T	F	T	F	
T	F	T	T	F	
T	F	F	F	T	
F	T	T	T	F	
F	T	F	T	F	
F	F	T	T	F	
F	F	F	F	T	

Example: Truth Table

- Truth Table for "*p* and not (*q* or *r*)."

p	*q*	*r*	*q* or *r*	not (*q* or *r*)	*p* and not (*q* or *r*)
T	T	T	T	F	F
T	T	F	T	F	F
T	F	T	T	F	F
T	F	F	F	T	T
F	T	T	T	F	F
F	T	F	T	F	F
F	F	T	T	F	F
F	F	F	F	T	F

Distributive Properties

- The distributive properties state that

$$p \text{ or } (q \text{ and } r) \equiv (p \text{ or } q) \text{ and } (p \text{ or } r),$$
$$p \text{ and } (q \text{ or } r) \equiv (p \text{ and } q) \text{ or } (p \text{ and } r).$$

- These properties can be handy when simplifying logical expressions without writing truth tables.

DeMorgan's Laws

- DeMorgan's Laws state that

$$\text{not } (p \text{ and } q) \equiv (\text{not } p) \text{ or } (\text{not } q),$$
$$\text{not } (p \text{ or } q) \equiv (\text{not } p) \text{ and } (\text{not } q).$$

- DeMorgan's Laws are handy when simplifying logical expressions without writing truth tables.

Examples of DeMorgan's Laws

- Simplifications by DeMorgan's Laws.

$$p \text{ and not } (q \text{ or } r) \equiv p \text{ and } ((\text{not } q) \text{ and } (\text{not } r))$$

Examples of DeMorgan's Laws

- Simplifications by DeMorgan's Laws.

$$\begin{aligned} p \text{ and not } (q \text{ or } r) &\equiv p \text{ and } ((\text{not } q) \text{ and } (\text{not } r)) \\ &\equiv p \text{ and } (\text{not } q) \text{ and } (\text{not } r). \end{aligned}$$

Examples of DeMorgan's Laws

- Simplifications by DeMorgan's Laws.

$$\begin{aligned} p \text{ and not } (q \text{ or } r) &\equiv p \text{ and } ((\text{not } q) \text{ and } (\text{not } r)) \\ &\equiv p \text{ and } (\text{not } q) \text{ and } (\text{not } r). \end{aligned}$$

$$p \text{ or not } (p \text{ and } q) \equiv p \text{ or } ((\text{not } p) \text{ or } (\text{not } q))$$

Examples of DeMorgan's Laws

- Simplifications by DeMorgan's Laws.

$$\begin{aligned} p \text{ and not } (q \text{ or } r) &\equiv p \text{ and } ((\text{not } q) \text{ and } (\text{not } r)) \\ &\equiv p \text{ and } (\text{not } q) \text{ and } (\text{not } r). \end{aligned}$$

$$\begin{aligned} p \text{ or not } (p \text{ and } q) &\equiv p \text{ or } ((\text{not } p) \text{ or } (\text{not } q)) \\ &\equiv (p \text{ or } (\text{not } p)) \text{ or } (\text{not } q) \end{aligned}$$

Examples of DeMorgan's Laws

- Simplifications by DeMorgan's Laws.

$$\begin{aligned} p \text{ and not } (q \text{ or } r) &\equiv p \text{ and } ((\text{not } q) \text{ and } (\text{not } r)) \\ &\equiv p \text{ and } (\text{not } q) \text{ and } (\text{not } r). \end{aligned}$$

$$\begin{aligned} p \text{ or not } (p \text{ and } q) &\equiv p \text{ or } ((\text{not } p) \text{ or } (\text{not } q)) \\ &\equiv (p \text{ or } (\text{not } p)) \text{ or } (\text{not } q) \\ &\equiv \text{true or } (\text{not } q) \end{aligned}$$

Examples of DeMorgan's Laws

- Simplifications by DeMorgan's Laws.

$$\begin{aligned} p \text{ and not } (q \text{ or } r) &\equiv p \text{ and } \left((\text{not } q) \text{ and } (\text{not } r)\right) \\ &\equiv p \text{ and } (\text{not } q) \text{ and } (\text{not } r). \end{aligned}$$

$$\begin{aligned} p \text{ or not } (p \text{ and } q) &\equiv p \text{ or } \left((\text{not } p) \text{ or } (\text{not } q)\right) \\ &\equiv \left(p \text{ or } (\text{not } p)\right) \text{ or } (\text{not } q) \\ &\equiv \text{true or } (\text{not } q) \\ &\equiv \text{true}. \end{aligned}$$

Outline

The `bool` Data Type

- In C++, there is the **`bool`** data type.
- A **`bool`** object can take on one of only two **`bool`** values.
  - **`true`**
  - **`false`**
- The **`bool`** type is in the integer family.
  - **`true`** is stored as `1`.
  - **`false`** is stored as `0`.
- **`bool`** objects occupy one byte of memory, even though they need only one bit.

Printing Boolean Values

- Normally, when we output a **bool**, we get 0 if it is false and 1 if it is true.
- To see the words “true” and “false” in the output, we need to include the iomanip header file

```
#include <iomanip>
```

  and write

```
cout << setiosflags(ios_base::boolalpha);
```

  in the program (before outputting the **bool**s).

The Boolean Operators

- There are three (not four) logical operators in C++.
  - The “and” operator is `&&`
  - The “or” operator is `||`
  - The “not” operator is `!`

Examples

- “p and (q or r)” would be written as

```
p && (q || r)
```

  which is the same as

```
(p && q) || (p && r)
```

- “p or (q and r)” would be written as

```
p || (q && r)
```

  which is the same as

```
(p || q) && (p || r)
```

Examples

- “not (p or q)” would be written as

```
!(p || q)
```

which is the same as

```
!p && !q
```

- “not (p and q)” would be written as

```
!(p && q)
```

which is the same as

```
!p || !q
```

Relational Operators

- Relational operators are operators that compare objects.
- Equality Operators
  - The “equal to” operator is ==.
  - The “not equal to” operator is !=.
- Order Operators
  - The “greater than” operators is >.
  - The “less than” operator is <.
  - The “greater than or equal to” operator is >=.
  - The “less than or equal to” operator is <=.

Boolean Expressions and Relational Operators

- Typically, boolean expressions are created by using relational operators to compare numerical or other quantities.
- Examples
  - Integer: `count != 0`
  - Floating-point: `x < 123.4`
  - Character: `c >= 'A' && c <= 'Z'`
  - String: `answer == "yes"`
  - Mixed: `count > 0 && sum <= 100.0`
- The operands may be of various types, but the result is always **`bool`**.

Relational Operators

- The equality operators == and != should be defined on all data types since they always make sense.
- The order operators <, >, <=, and >= are defined on a data type only if they make sense for that type.

Relational Operators

- For which types do the order operators make sense?
  - **short**, **int**, and **long**?
  - **float** and **double**?
  - **char**?
  - string?
  - **bool**?

Outline

Precedence Rules

- Precedence order from highest to lowest.
  - Post-increment and post-decrement `++`, `--`
  - Logical "not" `!`
  - Unary operators `+`, `-`
  - Pre-increment and pre-decrement `++`, `--`
  - Multiplicative operators `*`, `/`, `%`
  - Additive operators `+`, `-`
  - Insertion and extraction `<<`, `>>`
  - Relational ordering operators `<`, `>`, `<=`, `>=`
  - Relational equality operators `==`, `!=`
  - Logical "and" operator `&&`
  - Logical "or" operator `||`
  - Assignment operators `=`, `+=`, `-=`, `*=`, `/=`, `%=`

Compound Boolean Expressions

Examples

```
x == -y || z != 0
x < y && y < z
x = b == 0 || a / b == c && !p
```

Improved Examples

```
(x == -y) || (z != 0)
(x < y) && (y < z)
x = ((b == 0) || ((a / b == c) && !p))
```

Compound Boolean Expressions

Examples

```
0 <= x <= 1             // Wrong
0 <= x && x <= 1        // Right
```

- We cannot “chain together” inequalities

A Special Case

- Note that `<<` has a higher precedence than the relational operators `==`, `!=`, `<`, `>`, `<=`, and `>=` and the logical operators `&&` or `||`.
- Therefore, the statement

```
cout << a == 0 << endl;
```

  is illegal because it is interpreted as

```
(cout << a) == (0 << endl);
```

- It must be written

```
cout << (a == 0) << endl;
```

Outline

Example

- Example
  - `BoolOperators.cpp`

Outline

Assignment

Assignment

- Read Sections 2.11, 4.1, 4.7.