1806ICT Programming Fundamentals

Boolean Expressions

1

Booleans and Boolean Expressions

- Must use #include <stdbool.h>
- The type bool is a primitive type with only two possible values: true and false.

bool valid = true; valid = false;

Relational Operators

Math	Meaning	С	Example
=	Equal to	==	balance == 0 answer == 'y'
≠	Not equal to	!=	income != tax answer != 'y'
>	Greater than	>	expenses > income
≥	Greater than or equal to	>=	points >= 60
<	Less than	<	pressure < max
≤	Less than or equal to	<=	expenses <= income

2

Compound Boolean Expressions

- Boolean expressions can be combined using the "and" (&&) operator.
- Example

```
((score > 0) && (score <= 100))
```

Not allowed

```
(0 < score <= 100)
```

3

3

Compound Boolean Expressions

Syntax

```
(Sub_Expression_1) && (Sub_Expression_2)
```

- Parentheses often are used to enhance readability.
- The larger expression is true only when both of the smaller expressions are true.

4

Truth Tables

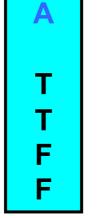
In the following slides, the letters A, B and C represent any C boolean expression. For example, A, B or C could represent any of the following (given integers x and y):

x == y x >= y y != x A T T F F

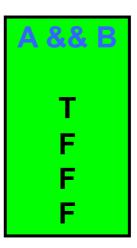
T F T F

5

Truth Table for Boolean Operator &&



B TFTF



Compound Boolean Expressions

- Boolean expressions can be combined using the "or" (||) operator.
- Example

```
((quantity > 5) || (cost < 10))
```

Syntax

```
(Sub Expression 1) || (Sub Expression 2)
```

- The larger expression is true
 - When either of the smaller expressions is true
 - When both of the smaller expressions are true.
- The C version of "or" is the *inclusive or* which allows either or both to be true.

7

Negating a Boolean Expression

- A boolean expression can be negated using the "not" (!) operator. In this context, negated means that a boolean value is changed from true to false or from false to true.
- Syntax

! (Boolean Expression)

Example

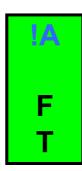
(a || b) && !(a && b)

which is the exclusive or

9

Truth Table for Boolean Operator!





10

11

C Boolean Operators (Summary) && (sum > min) && (sum < max) Logical and Logical or | | $(answer == 'y') \mid \mid (answer == 'Y')$!(number < 0)Logical not Value of A Value of **B** Value of Value of Value of ! (A) A && B $A \mid \mid B$ true false true true true false false false true true false true false true true false false false false true

Short-circuit Evaluation

- Sometimes only part of a boolean expression needs to be evaluated to determine the value of the entire expression.
 - If the first operand associated with an | | is true, the expression is true.
 - If the first operand associated with an && is false, the expression is false.
- This is called *short-circuit* or *lazy* evaluation.

13

13

Short-circuit Evaluation

- Short-circuit evaluation is not only efficient, sometimes it is essential!
- A run-time error can result, for example, from an attempt to divide by zero.

```
((number != 0) && (sum/number > 5))
```

Precedence Rules

Highest Precedence

First: the unary operators +, -, ++, --, and!

Second: the binary arithmetic operators *, /, %

Third: the binary arithmetic operators +, -

Fourth: the boolean operators <, >, <=, >=

Fifth: the boolean operators ==, !=

Sixth: the boolean operator &

Seventh: the boolean operator |

Eighth: the boolean operator &&

Ninth: the boolean operator | |

Lowest Precedence

15

15

Precedence Rules

In what order are the operations performed?

```
score < min/2 - 10 || score > 90
score < (min/2) - 10 || score > 90
score < ((min/2) - 10) || score > 90
(score < ((min/2) - 10)) || score > 90
(score < ((min/2) - 10)) || (score > 90)
```

16

Using ==

• == is appropriate for determining if two integers or characters have the same value.

```
(a == 3)
where a is an integer type
```

• == is **not** appropriate for determining if two floating points values are equal. Use < and some appropriate tolerance instead.

```
(abs(b - c) < epsilon)
where b, c, and epsilon are floating point types</pre>
```

17