Booleans and Comparisons

Principles of Computer Programming I

Spring/Fall 20XX

AUGUSTA
UNIVERSITY

Outline

- Boolean data type
- Comparison and logic operators
- Combining conditions and operator precedence

Decision/Control Structures

- Normally C# programs are executed **sequentially**

```
ClassRoom csci = new ClassRoom("Allgood East", 356);
Console.WriteLine($"Classroom: {csci}");
csci.SetNumber(120);
csci.SetBuilding("UH");
Console.WriteLine($"Classroom: {csci}");
```

- **Decision structures** can change the flow of execution
  - Only execute code if some condition is true: `if`, `else`, `switch`
  - Execute code repeatedly, until some condition is true: `while`, `for`

Decisions and Conditions

- All decision structures must:
  - Evaluate a condition in the program
  - Decide what code to execute next
- Conditions are **Boolean** values: either true or false
- "Is the classroom's number over 300? If so, it is on the 3rd floor"
- Example:

Condition: number is over 300

Code to execute if condition is true

```
if(csci.GetNumber() > 300)
{
    Console.WriteLine("It's on the 3rd floor");
}
```

Boolean Data Type

- A condition produces a value of type `bool`
- This can be stored in a variable

```
bool isThirdFloor = csci.GetNumber() > 300;
```

Gets value `true`

`true`

- bool variables can only hold 2 values: `true` or `false`

```
bool isFriday = true;
bool after5PM = false;
```

Outline

- Boolean data type
- **Comparison and logic operators**
- Combining conditions and operator precedence

Relational and Equality Operators

- Many conditions are comparisons between values
- C# **relational operators** compare values and return a `bool`

Math Notation	C# Operator	Example
$>$	`>`	`3 > 4` $\longrightarrow$ `false`
$<$	`<`	`3 < 4` $\longrightarrow$ `true`
$\geq$	`>=`	`3 >= 4` $\longrightarrow$ `false`
$\leq$	`<=`	`3 <= 4` $\longrightarrow$ `true`

- These only work on numbers (and `char`*)

*Unicode values, not alphabetical order

Relational and Equality Operators

- Another comparison: testing for equality
- C# **equality operators** work on all built-in types

Math Notation	C# Operator	Example
$=$	`==`	`3 == 4` $\longrightarrow$ `false`
$\neq$	`!=`	`3 != 4` $\longrightarrow$ `true`

- Note: *double* equals sign, not the same as assignment!

```
bool test = myStringVar == "Bananas";
```

This does not change myStringVar

Assignment

Equality comparison

Boolean Operations

- Can't use standard math operators on `bool` values
- Instead, use *logical* operators: "and", "or", "not"

Operation	Math Notation	C# Operator
Conjunction	$a \wedge b$	`a && b`
Disjunction	$a \vee b$	`a || b`
Negation	$\neg a$	`!a`

Example:

```
bool weekend = isFriday && after5PM;
```

Boolean Logic

- C# logical operators work just like their math equivalents

Expression	Result
`true && true`	`true`
`true && false`	`false`
`false && true`	`false`
`false && false`	`false`

Expression	Result
`true || true`	`true`
`true || false`	`true`
`false || true`	`true`
`false || false`	`false`

Expression	Result
`!true`	`false`
`!false`	`true`

Outline

- Boolean data type
- Comparison and logic operators
- **Combining conditions and operator precedence**

Summary of Logical Conditions

- Relational operators: >, <, >=, <=

  `12.5 < 6.0` $\longrightarrow$ `false` `13 >= 13` $\longrightarrow$ `true`

- Equality operators: ==, !=

  `3 == 6.0` $\longrightarrow$ `false` `"food" != "bananas"` $\longrightarrow$ `true`

- Logic operators: &&, ||, !

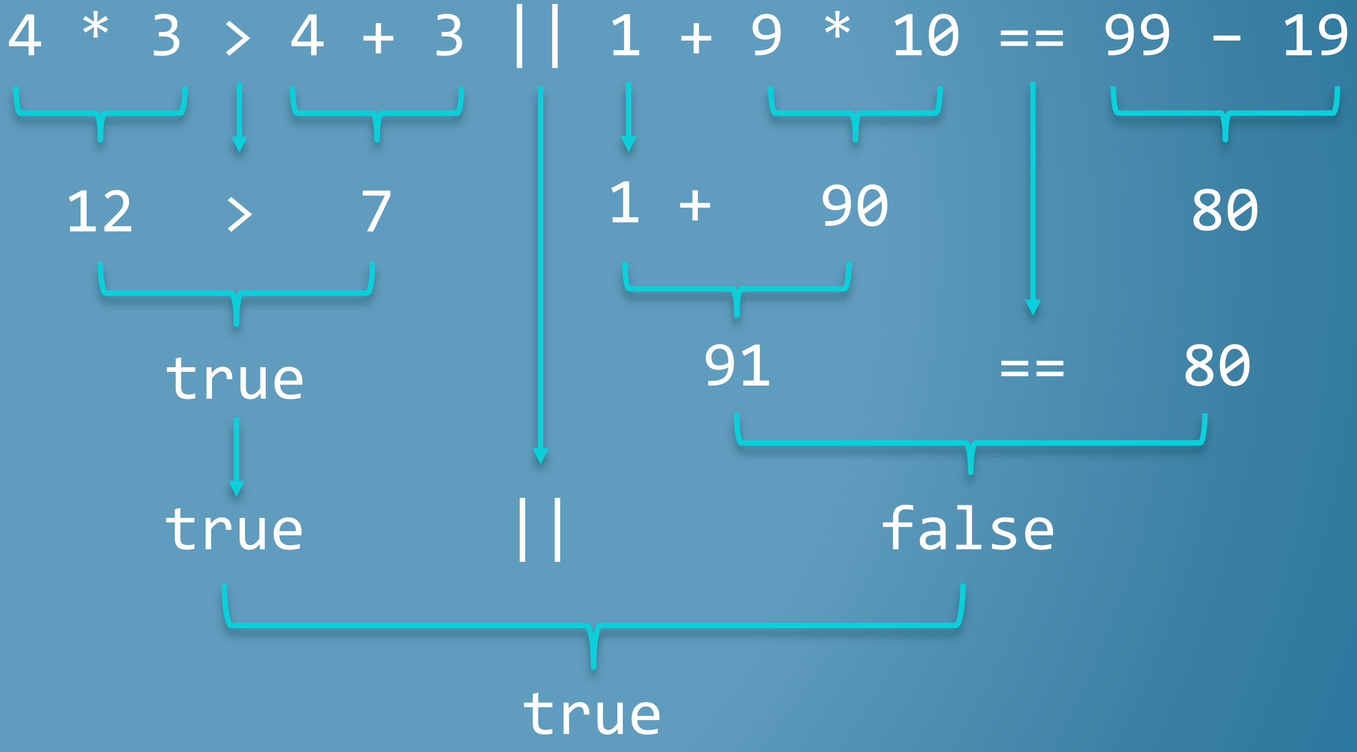
  `true || false` $\longrightarrow$ `true` `false && true` $\longrightarrow$ `false`

- What happens when we combine them?

Order of Operations

- Operator precedence:

  1. `!` ← "not"
  2. `* / %` ← Arithmetic, PEMDAS
  3. `+ -`
  4. `> < >= <=` ← Inequality
  5. `== !=` ← Equality
  6. `&&` ← "and"
  7. `||` ← "or"



Combining Conditions

- Test if `myInt` is outside the range [-5, 5]:

```
bool rangeTest = myInt > 5 || myInt < -5;
```

- Test if `myString` is “Hello”:

```
bool stringTest = myString == "Hello";
```

- Test both conditions?

$$\texttt{bool both = myInt > 5 || }\underbrace{\texttt{myInt < -5 \&\& myString == "Hello";}}_{\text{This gets evaluated first!}}$$

Combining Conditions

- Testing both conditions correctly:

```
bool both = (myInt > 5 || myInt < -5) && myString == "Hello";
```

Parentheses ensure the || is evaluated first

- Since && always comes before ||, remember to use parentheses when combining conditions

```
(condition_1) && (condition_2);
```

Comparisons and Types

- Like other C# operators, types must match in comparisons
- Implicit conversion will be used if possible to make them match

```
25 == "25"
```

$\Rightarrow$ Compile error: Can’t convert `int` to `string`

```
42 < 4.2
```

$\downarrow$

$$\underbrace{\texttt{42.0 < 4.2}}_{\texttt{false}}$$

```
8.0f == 8
```

$\downarrow$

$$\underbrace{\texttt{8.0f == 8.0f}}_{\texttt{true}}$$

```
19.99m < 20.0
```

$\Rightarrow$ Error!

Summary

- Boolean data type
- Comparison and logic operators
- Combining conditions and operator precedence