**Boolean & If Statements**

Chapter 4  
Fall 2015, CSUS

- A decision structure allows a program to perform actions only under certain conditions
- Found in virtually every programming language

**Introduction to Decision Structures**

**Different Types of Decisions**

- **If Statement**
  - gives a single, optional alternative
  - very common in programming
- **If-Then-Else**
  - gives dual (two) alternatives
  - a form of the If Statement
- **Case structure**
  - gives multiple alternative decisions
  - not supported in Flowgorithm

**Boolean Logic**

- Decision structures rely on having the computer compare data and make a decision based on the result
- For this, computers use **Boolean Logic**
  - basis of all computer (and human) logic
  - Boolean values can be either *True* or *False*
  - understanding Boolean logic is essential to knowing how to program

**Relational Operators**

- A **Relational Operator** determines whether a specific relationship exists between two values
- The most basic way to test two pieces of data
- Returns either *True* or *False*
Relational Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>Equal To</td>
</tr>
<tr>
<td>!=</td>
<td>Not Equal To</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater Than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater Than or Equal</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less Than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less Than or Equal</td>
</tr>
</tbody>
</table>

Relational Examples

1 == 4  \(\text{False}\)
24 != 31 \(\text{True}\)
11 < 10 \(\text{False}\)
100 >= 100 \(\text{True}\)

If Statements

- An action only occurs if the Boolean expression is True
- Otherwise, nothing will occur
- A diamond symbol is used in flowcharts

Book Pseudocode

Either True or False

If Condition Then
Statements
End If

Multiple statements

Example in Pseudocode

Declare Integer age

Input age
If age >= 21 Then
  Display "Kegger!"
End If

Example in a Flowchart
Example Output

```
22
Kegger!
```

Example 2 Output

```
20
Nothing. The Display Statement is not executed
```

Example

```
Declare Integer age

Input age
If age >= 21 Then
    Display "Kegger!"
End If
```

Example 2

```
Declare Integer guess

Display "Year CSUS founded? "
Input guess
If guess == 1947 Then
    Display "Correct!"
End If
```

Increment Example Output

```
Year CSUS founded? 1992
```

Increment Example Output

```
Year CSUS founded: 1947
Correct!
```
Dual Alternative Structures

- **Dual Alternative Decision Structures** selects between two different groups of statements.
- If the Boolean expression is **True** it executes one group and, if **False**, the other.

The Else Clause

- The If Statement has an optional "else" clause.
- This denotes the group that executes in the expression is **False**.
- It goes before the "End If" since it is part of the same If-Statement.

Book Pseudocode

```plaintext
If Condition Then
    Statements
else
    Statements
End If
```

Processed if the Condition is false

Pseudocode

```plaintext
If condition Then
    statement
Else
    statement
End if
```

```plaintext
If temp < 40 Then
    Display "It's cold"
    Display "Get a coat!"
Else
    Display "It's warm"
    Display "Get water!"
End if
```

Dual Example in Pseudocode

```plaintext
Declare Integer age
Input age
If age >= 21 Then
    Display "Kegger! :)
Else
    Display "Milk! :(
End If
```
### Chapter 4.3

#### Comparing Strings

- Most languages allow you to compare strings
- Textual data is often used to make decisions
- The same rules that apply to comparing numbers, applies to strings

#### String Equality

- You can test two string variables (or literals) for equality
- You can also test if one string is greater or less than another string (allows for sorting strings)

```python
name1 == name2
month != "February"
```
Case Sensitivity

- String comparisons are generally case sensitive
- This means that uppercase and lower case letters are not the same
- Why? Let's look back at ASCII...

ASCII Chart Review

<table>
<thead>
<tr>
<th>Character</th>
<th>Binary</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>01001101</td>
<td>77</td>
</tr>
<tr>
<td>o</td>
<td>01100111</td>
<td>111</td>
</tr>
<tr>
<td>e</td>
<td>01100101</td>
<td>101</td>
</tr>
</tbody>
</table>

ASCII Codes

- Each character has a unique value
- The following is how "Moe" is stored in ASCII

Comparing Letters

- "A" < "a"

Letter Examples

- "a" == "A"  
  False
- "a" > "A"  
  True
- "abC" < "abc"  
  True
- "dog" != "cat"  
  True

Example

Declare String answer

Display "Do you love programming?"

Input answer

If answer = "y" Then
  Display "Most excellent!"
End If
Do you love programming?

\[ Y \]

Most excellent!

---

**Example Output**

Do you love programming?

\[ Y \]

Nothing!

"Y" is not equal to "Y"

---

**Nested Decision Structures**

- In programming, control structures can be put inside other structures
- This is called *nesting* and allows complex control

---

*By nesting…*

- you can put almost *anything in anything*
- this allows you to create complex programs

---

**Nested If Statements**

- Often If Statements are embedded in other If Statements
- This can give multiple branches in the program
- … and have a conditional statement dependent on another
Nested Decision Structures

If-Else Chain

- Sometimes an If Statement is nested on the else clause of another If Statement
- It allows what is called an "If-Else Chain"

If-Else Chain

- It can make nested logic simpler to write
- Basically, the chain checks multiple expressions and selects the first one that is true

If-Else Chain

If score < 60 Then
    Display "Grade is F."
Else If score < 70 Then
    Display "Grade is D."
Else If score < 80 Then
    Display "Grade is C."
Else If score < 90 Then
    Display "Grade is B."
Else
    Display "Grade is A."
End If

Indenting is Vital

- You must indent each nested block of statements
- Otherwise your program quickly becomes unreadable
Logical Operators

Chapter 4.6

Logical Operators

- Logical Operators are used between comparisons to create complex Boolean expressions.
- In Boolean logic, there are just 3 operators – which are easy to learn and master.

<table>
<thead>
<tr>
<th>Name</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>And</td>
<td>True only if <strong>both</strong> operands are True. If either is False, the result is False.</td>
</tr>
<tr>
<td>Or</td>
<td>True if <strong>either</strong> operand is True. False if both operands are False.</td>
</tr>
<tr>
<td>Not</td>
<td>True if the operand is False. False if the operand is True.</td>
</tr>
</tbody>
</table>

**Boolean Table**

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>not p</th>
<th>p or q</th>
<th>p and q</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

**AND Example**

\[ 1 < 2 \text{ and } 10 < 40 \]

Result: **True**

**AND Example 2**

\[ 1 < 2 \text{ and } 12 < 4 \]

Result: **False**
**OR Example**

\[ 1 < 2 \text{ or } 1 < 4 \]

Result: True

**OR Example 2**

\[ 1 < 2 \text{ or } 6 < 4 \]

Result: True

**OR Example 3**

\[ 3 < 2 \text{ or } 6 < 4 \]

Result: False

**NOT Example**

\[ \text{not} (1 < 2) \]

Result: False

**NOT Example 2**

\[ \text{not} (3 < 2) \]

Result: True

**Examples**

- \[ 1 < 3 \text{ and } 10 < 40 \] True
- \[ 1 == 3 \text{ and } 10 < 40 \] False
- \[ \text{not} \ 1 == 2 \] True
- \[ 1 > 3 \text{ or } 30 < 20 \] False
Typical Precedence Levels

<table>
<thead>
<tr>
<th>Level</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>- (unary) not</td>
</tr>
<tr>
<td>5</td>
<td>* /</td>
</tr>
<tr>
<td>4</td>
<td>+ -</td>
</tr>
<tr>
<td>3</td>
<td>== != &gt; &gt;= &lt; &lt;=</td>
</tr>
<tr>
<td>2</td>
<td>and</td>
</tr>
<tr>
<td>1</td>
<td>or</td>
</tr>
</tbody>
</table>

Highest

Calculate The Result

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 == 4 or 1 &gt; 2 and 5 &gt; 4 or 1 &lt; 3</td>
<td>True</td>
</tr>
<tr>
<td>False or False and True or True</td>
<td>True</td>
</tr>
<tr>
<td>False or False</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

Lowest

Calculate The Result 2

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>not (1 &lt; 4) and 5 &gt; 4 or not (1 &gt; 4)</td>
<td>True</td>
</tr>
<tr>
<td>not True and True or not False</td>
<td>True</td>
</tr>
<tr>
<td>False and True or True</td>
<td>False</td>
</tr>
<tr>
<td>False or True</td>
<td>False</td>
</tr>
</tbody>
</table>

And Example

If temp < 20 AND minutes > 12 Then
   Display "Danger!"
   Display "Temperature danger zone."
End If

Example

Declare String answer

Display "Do you love programming?"
Input answer

If answer = "y" or answer = "Y" Then
   Display "Most excellent!"
End If

Example Output

Do you love programming?
Y
Most excellent!
Range Checking

- Range Checking is often used to test if a value is between two values.
- Often used to prevent invalid calculations.

Inside a Range

- When checking for a number inside a range, use **AND**.
- The following is only true if x is between 0 and 100.

```
If x >= 0 AND x <= 100 Then
    Display "Inside Range"
End If
```

Outside a Range

- When checking for a number outside a range, use **OR**.
- The following is true if x is outside the range.

```
If x < 20 OR x > 40 Then
    Display "Outside range"
End If
```

Range Checking Example

```
Declare Integer years, days
Input years

If years >= 1 and years <= 110 Then
    Set days = years * 365;
    Display "Days lived: ", days
Else
    Display "Invalid Age!"
End If
```

Range Checking Example Output

```
20
Days old: 7300
```
**Range Checking Example Output**

300

Invalid Age!

---

**Boolean Variables**

- A variable of the Boolean data type can hold one or two values: true or false
- It often holds the result of a Boolean expression – which will be used later in another expression

---

**Boolean Variable Example**

```plaintext
Declare Boolean isLunchTime

If time >= 12 and time <= 13 then
    Set isLunchTime = True
Else
    Set isLunchTime = False
End If
```

---

**Factoring**

- Often, when creating complex if statements, code is repeated in both the true branch and the false branch
- This redundant code can, sometimes, moved outside the loop

---

**Factoring**

Cleaning up code
Factoring

- This only works if the duplicate is at the beginning of both the True branch and the False branch
- When it is done, the code is "factored" out of the If Statement

Example – What's Redundant?

Input age
If age >= 18 then
    Input name
    Display name, " can vote"
Else
    Input name
    Display name, " cannot vote yet"
End If

Example – Factored Out

Input age
Input name
If age >= 18 then
    Display name, " can vote"
Else
    Display name, " cannot vote yet"
End If