Chapter 5

Logic; Got Any?

Flow of Control

- Flow of Control
  - The order in which the computer executes statements in a program
- Control Structure
  - A statement used to alter the normally sequential flow of control
  - Selection (Chapter 5)
  - Iteration (Chapter 6)
Selection

- Choosing between one or more options
- Example:
  - If statements
  - Switch statements

Boolean Data

- We state an assertion and check it’s validity
  - If the assertion is true, we do one thing.
  - Else we do some other thing.
- We use bool data types
  - Two values
  - true and false
Logical Expressions

- Boolean Variables and Constants
  - `eventFound = false;`
- Relational Operators
  - `lessThan = (i < j);`

Relational Operators

- Operator                  | Relationship Tested
  - `==`                     | Equality
  - `!=`                     | Inequality
  - `>`                      | Greater Than
  - `<`                      | Less Than
  - `>=`                     | Greater Than or Equal To
  - `<=`                     | Less Than or Equal To
Logical Operators

Three operators
- **And** `&&`
  - Both relationship must be true for the and to be true
- **Or** `||`
  - One relationship must be true for the or to be true
- **Not** `!`
  - Inverts the relationships truth value

And Truth Table

<table>
<thead>
<tr>
<th>Value of x</th>
<th>Value of y</th>
<th>Value of x &amp;&amp; y</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>
### Or Truth Table

| Value of x | Value of y | Value of x || y |
|------------|------------|-------------|
| true       | true       | true        |
| true       | false      | true        |
| false      | true       | true        |
| false      | false      | false       |

### Not Truth Table

<table>
<thead>
<tr>
<th>Value of x</th>
<th>Value of !x</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>
Short Circuit

- If the value of a statement can be determined before the entire statement must be evaluated, than do it.

Precedence of Operators

<table>
<thead>
<tr>
<th>! Unary + Unary -</th>
<th>Highest Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>* / %</td>
<td></td>
</tr>
<tr>
<td>+ -</td>
<td></td>
</tr>
<tr>
<td>&lt; &lt;= &gt; &gt;=</td>
<td></td>
</tr>
<tr>
<td>== !=</td>
<td></td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>=</td>
<td></td>
</tr>
</tbody>
</table>

Lowest Precedence
The if Statement

- If-Then-Else Form
  
  ```
  if ( Expression )
  Statement A
  else
  Statement B
  ```

Expressions and Statements

- An expression is a statement that evaluates to a logic value
- A statement is what you want the computer to do
Example

```cpp
if ( hours <= 40.0 )
    pay = rate * hours;
else
    pay = rate * ( 40.0 + ( hours -40.0 ) * 1.5);
cout << pay;
```

Compound Statements

- Not limited to one statement
- Use { and } to indicate statements that are to be grouped together
Example

```c++
if ( divisor != 0 )
{
    result = dividend / divisor;
    cout << "Division performed.\n";
}
else
{
    cout << "Division by zero is not allowed\n";
    result = 9999;
}
```

If-Then Form

- Sometimes there is only one choice
- Use:
  ```c++
  if ( Expression )
      Statement
  ```
Nested If’s

- You can stack ifs as deep as you want

```
if ( month == 1 )
    cout << “January”;
else if ( month == 2 )
    cout << “February”;
...
else
    cout << “December”;
```

Dangling Else

- Where does the else go?

```
if ( average < 70.0 )
    if ( average < 60.0 )
        cout << “FAILING”;
    else
        cout << “Barely Passing 😊”;
```
How about this one

```cpp
if ( average >= 60.0 )
    if ( average < 70.0 )
        cout << "Barely Passing 😞";
    else
        cout << "FAILING";
```

Correct Version

```cpp
if ( average >= 60.0 )
{
    if ( average < 70.0 )
        cout << "Barely Passing 😞";
}
else
    cout << "FAILING";
```
Testing streams

- You can test whether the last operation performed on a stream failed or not
  - if ( cin )
  - if ( cin.fail() )
  - if ( inFile )
  - if ( inFile.fail() )