Control Structures: The IF statement

Objectives

- To learn when and how to use an IF statement.
- To be able to form Boolean (logical) expressions using relational operators
  - <, <=, >, ==, !=, ....
- To be able to combine Boolean expressions using logical operators
  - AND (&&), OR (||), NOT (!)
Textbook

- Read chapter 4
  - Ignore SWITCH structures; and ASSERT.
- Things you need to get from the textbook
  - Comparison of characters and strings

Control Structures

- A computer can proceed:
  - In sequence
  - Selectively (branch) - making a choice
  - Repetitively (iteratively) - looping
- Some statements are executed only if certain conditions are met
- A condition is represented by a logical (Boolean) expression that can be true or false
- A condition is met if it evaluates to true
The IF-ELSE statement

```cpp
if (temp > 37.0)
    cout << "You have a fever.";
else
    cout << "You're fine.";
```

The IF-ELSE statement allows you to do different things depending on a condition, (temp > 37.0) in this case.

First we need to understand how to write conditions.
## Conditions

- A condition is an expression whose value is TRUE or FALSE.

  - `temp > 37`  true if the value of the variable `temp` is greater than 37
  - `n == 0`  true if the value of `n` is equal to 0
  - `count != n`  true if the value of `count` is NOT equal to the value of `n`
  - `(price / area) >= 5.5`  true if the value of `price` divided by the value of `area` is greater than or equal to 5.5

## Logical (Boolean) Expressions

- The `bool` Data Type and Logical (Boolean) Expressions

  - The data type `bool` has logical (Boolean) values `true` and `false`
  - `bool`, `true`, and `false` are reserved words
  - The identifier `true` has the value 1
  - The identifier `false` has the value 0
Relational Operators

- Relational operators:
  - Allow comparisons
  - Require two operands (binary)
  - Return 1 if expression is true, 0 otherwise

- Comparing values of different data types may produce unpredictable results
  - For example, 8 < '5' should not be done

- Any nonzero value is treated as true
  - This is a hangover from old C days. Clearer to use bool true and false, and not 0 and 1.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>equal to</td>
</tr>
<tr>
<td>!=</td>
<td>not equal to</td>
</tr>
<tr>
<td>&lt;</td>
<td>less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than or equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than or equal to</td>
</tr>
</tbody>
</table>
Logical (Boolean) Operators

- Logical (Boolean) operators enable you to combine logical expressions.
- Three logical (Boolean) operators:
  - ! = not
  - && = and
  - || = or
- Logical operators take logical values as operands and yield logical values as results.
- Putting ! in front of a logical expression reverses its value.

&& and ||

- (x > 0) && (x <= 50)
  - True if x is both greater than 0 and less than or equal to 50.
  - NB (0 < x <= 50) won’t work properly!!
  - Always true!
- (x%4 != 0) || (x < 1538)
  - True if x is not divisible by 4 or if x is less than 1538, or if both are true.
  - Note x%4 is the remainder from dividing x by 4. If it’s not equal to 0, x is not divisible by 0.
Precedence of Operators

- Relational and logical operators are evaluated from left to right
- The associativity is left to right
- Parentheses can override precedence

<table>
<thead>
<tr>
<th>Operators</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>!, +, - (unary operators)</td>
<td>first</td>
</tr>
<tr>
<td>*, /, %</td>
<td>second</td>
</tr>
<tr>
<td>+=, -=</td>
<td>third</td>
</tr>
<tr>
<td>&lt;, &lt;=, &gt;=, &gt;, ==, !=</td>
<td>fourth</td>
</tr>
<tr>
<td>==, /=</td>
<td>fifth</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>sixth</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>= (assignment operator)</td>
<td>last</td>
</tr>
</tbody>
</table>
Which is false?

int x = 10, y = 15, z = 20;

A. !(x > 10)
B. x <= 5 || y <= 15
C. (x != 5) && (y != z)
D. x >= z || (x + y >= z)
E. (x <= y-2) && (y >= z) || (z-2 ==20)

Note: textbook ch4 ex 3(e) answer is wrong

Logical expressions

- (a + 2 <= b) && !flag
- If a is 5 and b is 8 and flag is false what is the value of the above condition?

- Study Example 4.5 in the textbook.
Short-Circuit Evaluation

- Evaluation of a logical expression stops as soon as the value of the expression is known.
  - If the first part of an && is false, the whole lot must be false.
  - If the first part of an || is true, the whole lot is true.

Short-circuit - Example

- Example:
  
  \[(x \neq 0) \&\& (y/x > 4)\]

  If \(x\) is 0, \((x\neq0)\) is false, and the AND can't be true. The expression is false. There is no need to evaluate the second part \((y/x > 4)\).
  So the \((y/x)\) is not evaluated.
  This is handy because dividing by zero would be an error.
**IF branching**

- **temp > 37**
  - **yes**
    - `cout << "fever"`;
  - **no**
    - `cout << "fine"`;

**IF-ELSE syntax**

```cpp
if ( condition )
    what to do if true;
else what to do if false;
```

This version is for when you only need one statement in each branch.

```cpp
if ( condition )
    { what to do if true }
else
    { what to do if false }
```

Otherwise use `{}` braces to group together the multiple statements.
Example that needs {

Suppose we want to do more things if someone has a fever: (see temp-IF-multiple.cpp)

```cpp
if (temp > 37.0)
{
    cout << "You have a fever."
    cout << " Your temperature is " << (temp - 37);
    cout << " degrees above normal.";
}
else
    cout << "You're fine."
```

The {} braces group together multiple statements to form the “what to do if true” part.

Which numbers produce “1 3”?

```cpp
if (n % m == 3)
    { n = 3; m = 1; }
else if (n % m == 2)
    { n = 2; m = 2; }
else {n = 1; m = 3; }
cout << n << " " << m;
```

- A. n = 33, m = 10
- B. n = 12, m = 5
- C. n = 30, m = 10
- D. n = 10, m = 4
Let’s try

- Write a program to compute employee’s PRSI given that *(fictitious rates)*
  - Rate is 2% if weekly pay <= 256 euro
  - Rate is 5% if weekly pay > 256 euro
- Use a variable to store the rate.
- Assign the rate value inside an **IF** statement.

**IF Syntax**

- Note that the condition must be in ( ) parentheses.
- Note that there is no ; after the condition.

- You can leave out the **ELSE** part if you don’t want to do anything in that branch.

```cpp
if (yards != 0 )
    cout << yards << " yards."
if (feet != 0 ) cout << feet << " feet."
```
Exercises

- Write the statements that tell the user which pizza is better value, given that $\text{ppsi}_1$ and $\text{ppsi}_2$ have been computed.
  - Handle case where both pizzas are equal value.
- Extend the quadratic roots program to handle $a=0$, discriminant $= 0$, or $< 0$.
- Write a program to determine if a year is a leap year.
  - Note that since 1538, a year divisible by 100 is not a leap year, unless it is divisible by 400.