Chapter 4
Making Decisions

- Reading: Chapter 3 (3.5 pp. 101), Chapter 4 (4.4 pp. 166-168; 4.5 pp. 169-175; 4.6 pp.176-181; 4.8 pp. 182-189; 4.9 pp. 189-199; 4.14 pp. 202-210
Logical Operators

&&    And

||    Or

!      Not
Evaluating AND

expr1 && expr2

- For the complete expression to be true, both expr1 and expr2 must be true

- Example:

\[(\text{temp} > \text{HOT}) \&\& (\text{humidity} > \text{STICKY})\]

  - These are unbearable heat and humidity conditions
  - Both must be true for the entire expression to be true
Evaluating OR

\[ \text{expr1} \text{ || expr2} \]

- The complete expression is true, if either expr1 or expr2 is true

- Example:

\[ (\text{salary < MIN\_SALARY}) \text{ || (MARRIED == status)} \]

  - To qualify for financial aid, salary has to be less than some minimum salary OR you must be married
  - Only one condition has to be true
Evaluating NOT

\[ !\text{expr} \]

- If \text{expr} is true, \( !\text{expr} \) is false
- If \text{expr} is false, \( !\text{expr} \) is true
- Example:

\[ !(\text{salary} < \text{MIN\_SALARY}) \]

- What makes this true? False?
## Operator Precedence (highest to lowest)

<table>
<thead>
<tr>
<th>Operator Precedence</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unary plus &amp; minus</td>
<td>Right associative</td>
</tr>
<tr>
<td>Multiplication, division, and modulus</td>
<td>Left associative</td>
</tr>
<tr>
<td>Addition &amp; subtraction</td>
<td>Left associative</td>
</tr>
<tr>
<td>Relational operators</td>
<td>Left associative</td>
</tr>
<tr>
<td>Logical AND</td>
<td>Left associative</td>
</tr>
<tr>
<td>Logical OR</td>
<td>Left associative</td>
</tr>
<tr>
<td>Assignment</td>
<td>Right associative</td>
</tr>
</tbody>
</table>
Problem

According to the operator precedence and associativity rules given on the previous slide, how will the following expressions be evaluated?

\[
x < \text{min} + \text{max} \\
\text{min} \leq x && x \leq \text{max} \\
!x == y + 2 \\
x = a + b \% 7 * 2
\]
Problem

• Write a program segment that prints the message “The number is valid” if the variable speed is within the range 0-20 inclusive

• You must use logical operators
Problem

• A bookstore’s shipping policy is:
  1. If the order is $30 or less, shipping is $5
  2. If the order is over $30 but less than $50, shipping is $3
  3. If the order is $50 or more then shipping is $2

• Write this program using logical operators
TYPE CASTING
Explicit Type Conversion

- A type cast expression lets you manually change the data type of a value.
- The syntax for type casting is:
  
  ```
  static_cast<DataType>(Value)
  ```

  - Value is a variable or literal value.
    - If Value is a variable, the variable is **NOT** changed.
  - DataType is the data type that you are converting Value into.
Example

double number = 3.7;
int val;
val = static_cast<int>(number);

What is saved into val?
if Statement

- We may want to execute some code if an expression is true, and execute some other code when the expression is false.
- This can be done with two if statements...

```java
if (value >= LIMIT)
{
    // do something
}

if (value < LIMIT)
{
    // do something else
}
```
Double-Alternative if

- C++ provides a shortcut to combine 2 if statements

```cpp
if (expression)
{
    // stmts if expression is true
}
else
{
    // stmts if expression is false
}
```
Problem

```cpp
int number;
cout << "Enter a number, I’ll tell you";            
cout << " if it is odd or even: ";
cin >> number;
// write a double-alternative if here
```
Problem

- Are these two code snippets equivalent?

```c++
int x, y;
cin >> x >> y;
if(x > y)
{
    cout << x;
}
if(x < y)
{
    cout << y;
}
```

```c++
int x, y;
cin >> x >> y;
if(x > y)
{
    cout << x;
}
else
{
    cout << y;
}
```
Multiple-Alternative if

cout << "Enter two numbers: ";
cin >> num1 >> num2;

if (num1 > num2)
{
    cout << num1 << " is greater" << endl;
}
else if (num2 > num1)
{
    cout << num2 << " is greater" << endl;
}
else
{
    cout << "Numbers are equal" << endl;
}
Problem

- Write a C++ program segment that allows the user the ability to input an integer from the keyboard.
- If the integer is positive, increment a variable posCount by 1. If the integer is negative, increment a variable negCount by 1. If neither, increment zeroCount by 1

```cpp
int posCount = 0,
    negCount = 0,
    zeroCount = 0;
```
SWITCH STATEMENT
Let’s look at the following program segment:

```cpp
char choice;

cout << "E)dit S)ave Q)uit";
cin >> choice;

switch (choice) {
    case 'E': cout << "Time to edit " << endl; break;
    case 'S': cout << "Time to save" << endl; break;
    default: cout << "Illegal command" << endl;
}
```
switch format

```java
switch (ordinaldatatype)
{
    case constantexpression:  // one or more stmts
        break;
    case constantexpression:  // one or more stmts
        break;
    ...
    default :  // one or more stmts
}
```

What is an ordinal data type?

- *(ordinaldatatype)* can be a variable or expression
- *constantexpression* must be unique in each case
- *default* is optional
- *break*; resumes execution after the switch
Problem

1. Modify slide 22 to allow ‘E’, ‘e’, ‘S’, or ‘s’

2. Rewrite the logic for 1. as an if statement