Logical Operators and if/else statement

Floating Point and Relational Operators

- Floating point math may not work out as you expect because of round off errors.
- In Math
  - 6 * 2/3 = 4
- In C++, where 0.66666 is equivalent to 2/3
  - 6.0 * 0.66666 =
  - 6.0 * 0.66667 =
  - 6.0 * 0.666666 =
  - 6.0 * (2.0 / 3.0) =

11.1 Example (page 180)

```
double result;

result = 6.0 * 0.666666;

if ( result == 4.0 )
{
  cout << "result == 4.0" << endl;
}

cout << setprecision(6) << fixed;
cout << result << endl;
cout << setprecision(2) << result;
cout << endl;
```

Example

```
#include "stdafx.h"
#include <iostream>
#include <iomanip>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{

    double result;
    result = 6.0 * 0.666666;
    if( result == 4.0 )
    {
        cout <<" TRUE, result does equal 4.0" <<endl;
    }
    cout <<setprecision(6) <<fixed<< result <<endl;
    cout <<setprecision(2) <<fixed<< result <<endl;
    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
3.999996
4.00
Press any key to continue . . .
```


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…

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


If/Else (4.3)

- C++ provides a shortcut to combine two **`if`** statements:
- The statements in the else clause are executed only when the expression is false.

```
if( expression )
{
   // do stuff
}
else
{
   // do other stuff
}
```

11.2 Example

```
int number;
cout << "Enter a number, I'll tell you";
cout << " if it is odd: ";
cin >> number;

// use an if/else statement here
```


If/Else: Syntax and Formatting

```
if(expression)
{
    // do stuff
}
else
{
    // do other stuff
}
```

- Note the braces with the `else` keyword and the alignment of the `else` under the `if` on its own line

If/Else: Braces

```
if(expression)
{
    // do stuff
}
else
    x += 9;
```

- Always use braces with the `else`!

If/Else: Commenting

```
// the expression I'm using here
// checks for . . .
if(expression)
{
    // if the expression is true
    // I need to ...
}
else
{
    // if the expression is false
    // I need to ...
}
```


11.3 Practice

- Turn this code into an if/else statement:

```
int x, y;
if ( x > y )
{
    x += y;
}
if ( y <= x)
{
    y += x;
}
```


11.4 Practice

- Are these two code snippets equivalent?

```
int x, y;
if ( x > y )
{
    x += y;
}
if ( y < x)
{
    y += x;
}
```

```
int x, y;
if ( x > y )
{
    x += y;
}
else
{
    y += x;
}
```

Logical Operators (4.7)

- If we want to check for more than one condition then we need to use logical operators
- These combine logical expressions (i.e. expressions that have a true/false value)
- There are three logical operators
  - && and
  - || or
  - ! Not

11.5 Examples of Logical Operators

- `if( ( x > 7 ) && ( x < 20 ) )`
- `if( ( temp > 90.0 ) && ( humidity > 0.9 ) )`
- `if( ( salary < minSalary ) || ( dependents > 5 ) )`

Evaluating Expressions: And &&

- **`(expr1) && (expr2)`**
- For the complete expression to be true, both expr1 and expr2 have to be true
- Example:

  `(temp > 90.0) && (humidity > 0.9)`
  - These are unbearable heat and humidity conditions
  - Both must be true for the entire expression to be true

Evaluating Expressions: Or ||

- (**`expr1 || expr2`**)
- The complete expression is true if either expr1 or expr2 is true
- Examples:
  - `(salary < minSalary) || (dependents > 5)`
  - To qualify for financial aid, salary has to be less than some minimum salary or the number of dependents is greater than 5
  - Only one condition has to be true

Evaluating Expressions: Not !

- **`!expr`**
- Unary operator
- Examples:
  - `!((salary < minSalary) && (dependents > 5))`
  - What makes this true? False?

11.6 Example

- Your local bookstore has asked you to write a program to help them determine the cost of shipping of customers orders. If the order is $30 or less then shipping will cost $5, if the order is over $30 then shipping will be $3

11.7 Problem

- The bookstore has now changed it's shipping policy so that
  - If the order is $30 or less, shipping is $5
  - If the order is over $30 but less than $50, shipping is $3
  - If the order is over $50 then shipping is $2

Operator Precedence

- We have now added relational, equality and logical operators to the mathematical operators that were introduced last week
- Where do the new operators fit in the precedence table?

Precedence (page 1125)

Precedence Operators (Highest to Lowest)	
`- (unary negation), ! (Logical NOT)`	
`* / %` `- +`	Arithmetic Operators
`<= => > <` `== !=`	Relational Operators
`&& (Logical AND)` `|| (Logical OR)`	
`= += -= *= /= %=`	Assignment Operators

11.8 Expression Evaluation

- According to the operator precedence and associativity rules given on the previous slide, how will the following expressions be evaluated?
  - `x < min + max`
  - `min <= x && x <= max`
  - `!x == y + 2`
  - `x = a + b % 7 * 2`

`exit()`

- To terminate a program we can use the `exit(int status)` function
  - This is a *function*, not part of the language
    - `#include <stdlib.h>`
  - The `status` is returned to the operating system to denote program success or failure
    - Success: 0
    - Failure: non-zero

11.9 Practice

- Write a program that will ask the user for two integers. Display both integers to the screen if they are each greater than 1000 and terminate the program with `exit()` otherwise. Use exactly one `if/else`

```
#include <iostream>

using namespace std;

int main()
{
```