

# Goals for This Lecture:

- Understand C++ relational operators and expressions
- Understand C++ boolean operators and expressions

# Input using `cin`

- Values of variables can be read in from STDIN using the `cin` object
- `cin` can input any number of items as long as they are each preceded by the `>>` operator
- Example:  

```
cin >> x >> y ;
```
- Input is not read until the return key is typed
  - This allows user to correct input
- Values must be separated by spaces or line break signaled by returns
- You can also use `endl` in place of `"\n"`

# C++ Relational Operators

- C++ provides relational operators similar to those in FORTRAN 95
- Relational expressions between numerical types evaluate to bool values **true** and **false**
- Avoid comparing **ints** with **floats** or **doubles**

Relationship	C++ Operator	F95 Operator
Less than	<	<
Greater than	>	>
Less than or equal to	<=	<=
Greater than or equal to	>=	>=
Equal to	==	==
Not equal to	!=	/=

# C++ Boolean Operators

- C++ provides Boolean operators similar to those in FORTRAN 95
- Boolean expressions evaluate to bool values **true** and **false**

Relationship	C++ Operator	F95 Operator
Logical “and”	<b>&amp;&amp;</b>	<b>.and.</b>
Logical “or”	<b>  </b>	<b>.or.</b>
Logical negation	<b>!</b>	<b>.not.</b>

# Precedence Rules

- C++ evaluates un-parenthesized expressions that involve multiple types of operators in the following general order:
  1. Function calls
  2. Post-fixed increment operator `++`, post-fixed decrement-operator `--`
  3. Pre-fixed increment operator `++`, pre-fixed decrement operator `--`, logical negation `!`, unary minus and plus ( `-` and `+` )
  4. Multiplication, Division, and remainders
  5. Addition and subtraction
  6. Insertion and Extraction operators `<<`, and `>>`
  7. Relational operators `<`, `>`, `<=`, and `>=`
  8. Relational operators `==`, and `!=`
  9. Logical “and” (`&&`) operations
  10. Logical “or” (`||`) operation
  11. Assignment operators `=`, `+=`, `-=`, `/=`, `*=`, and `%=`
- Parenthesis can be used to clarify order of evaluation
- Unary operators and assignment operators are evaluated right-to-left
- Other binary operators are evaluated left-to-right

# Assignment

- Read sections 3.1-3.2 of Savitch