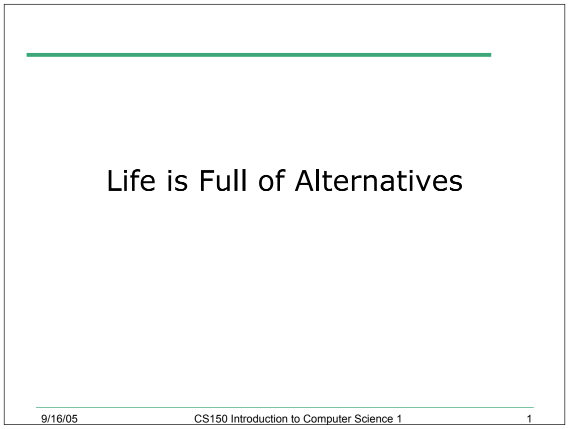
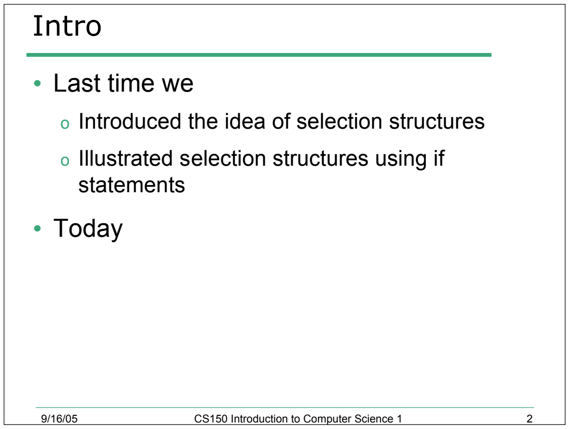
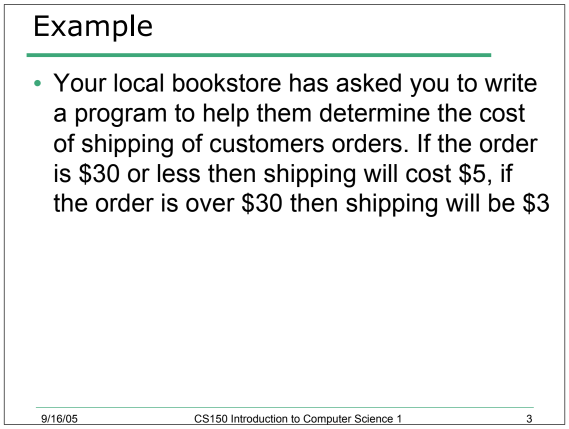


Life is Full of Alternatives



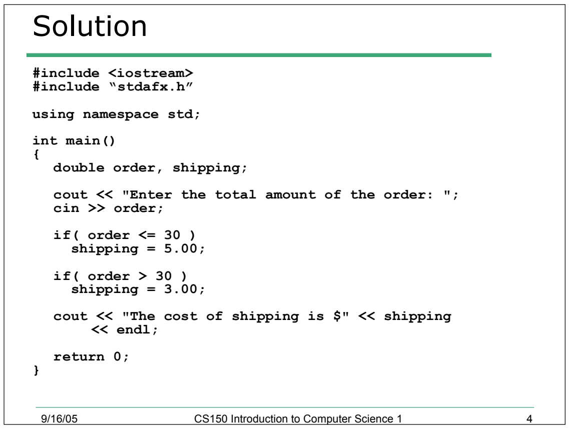
Intro

- Last time we
  - Introduced the idea of selection structures
  - Illustrated selection structures using if statements
- Today



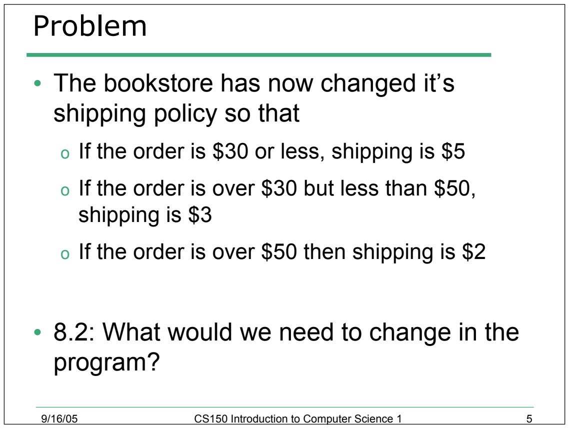
Example

- 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



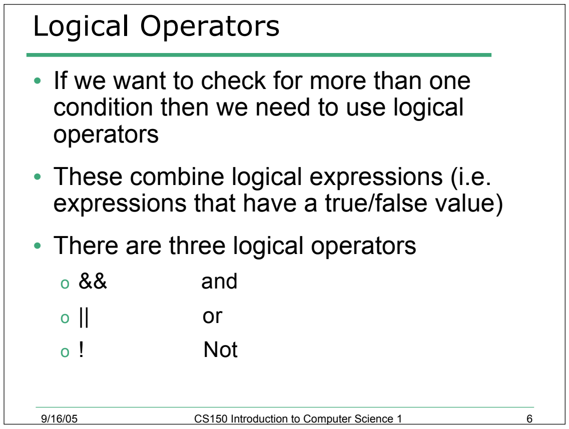
Solution

```
#include <iostream>
#include “stdafx.h”

using namespace std;

int main()
{
   double order, shipping;

   cout << "Enter the total amount of the order: ";
   cin >> order;

   if( order <= 30 )
     shipping = 5.00;

   if( order > 30 )
     shipping = 3.00;

   cout << "The cost of shipping is $" << shipping
        << endl;

   return 0;
}
```



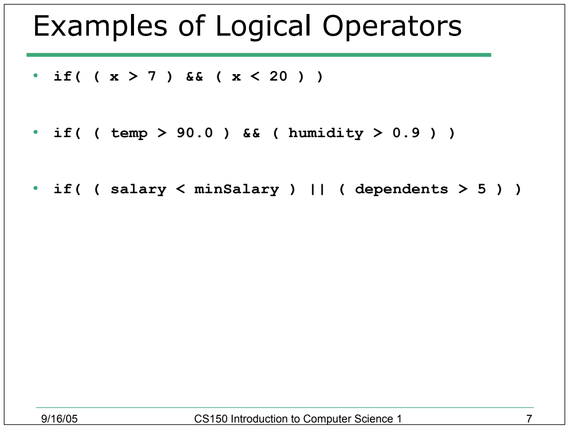
Problem

- 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
- 8.2: What would we need to change in the program?



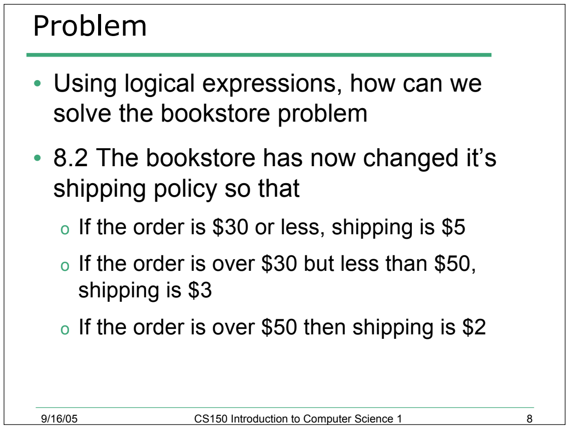
Logical Operators

- 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



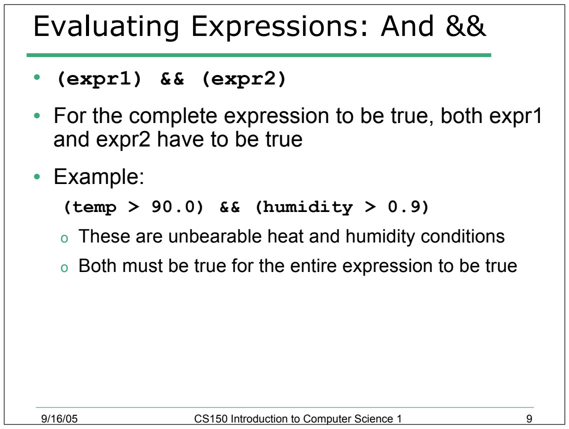
Examples of Logical Operators

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



Problem

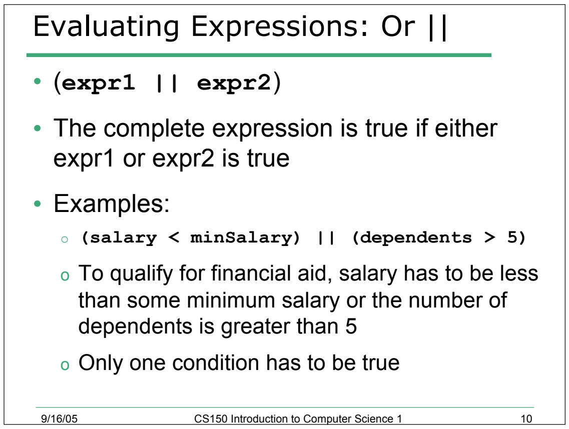
- Using logical expressions, how can we solve the bookstore problem
- 8.2 The bookstore has now changed it's shipping policy so that
  - If the order is $30 or less, shipping is $5
  - If the order is over $30 but less than $50, shipping is $3
  - If the order is over $50 then shipping is $2



Evaluating Expressions: And &&

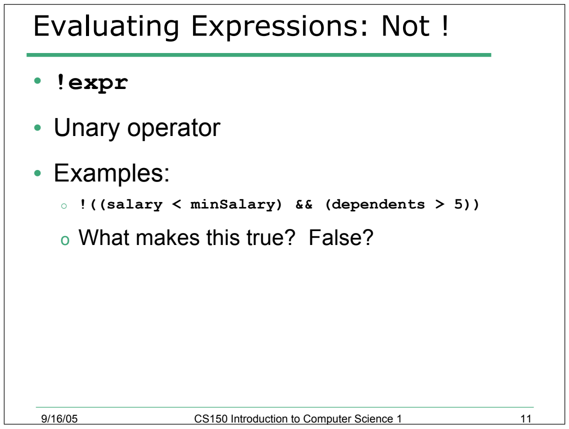
- **`(expr1) && (expr2)`**
- For the complete expression to be true, both expr1 and expr2 have to be true
- Example:

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



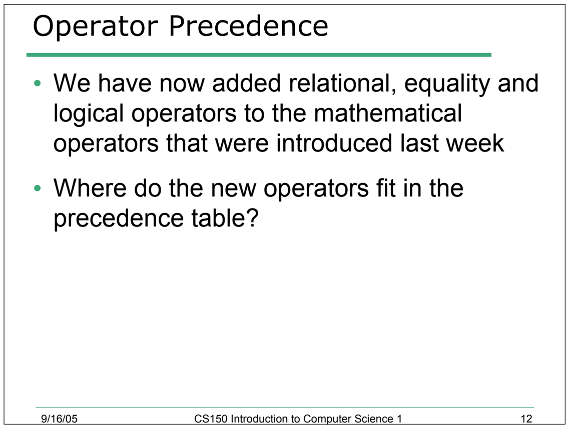
Evaluating Expressions: Or ||

- (**`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



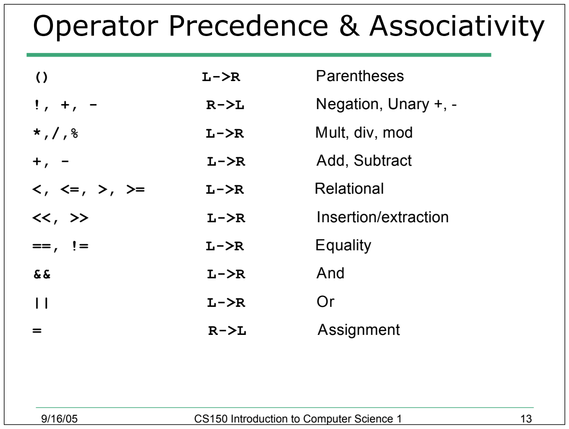
Evaluating Expressions: Not !

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



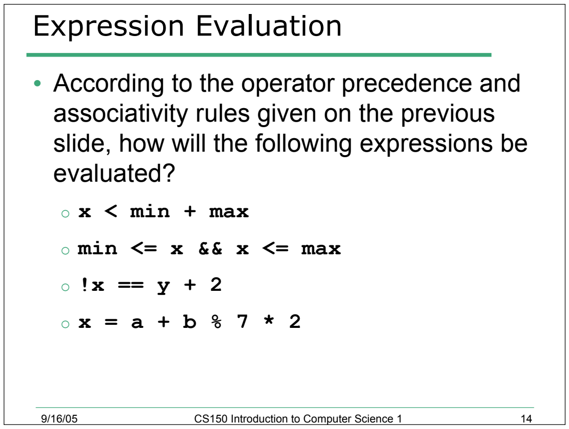
Operator Precedence

- 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?



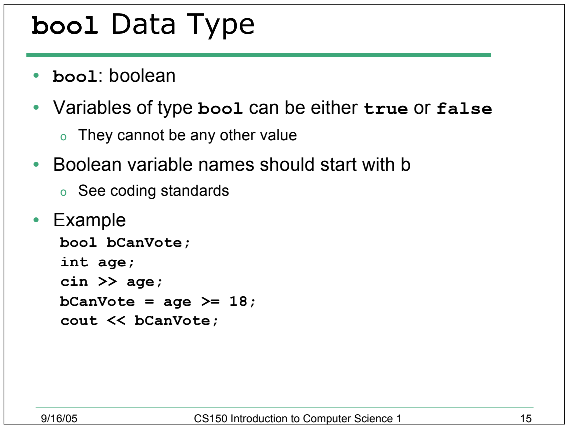
Operator Precedence & Associativity

`()`	`L->R`	Parentheses
`!, +, -`	`R->L`	Negation, Unary +, -
`*,/,%`	`L->R`	Mult, div, mod
`+, -`	`L->R`	Add, Subtract
`<, <=, >, >=`	`L->R`	Relational
`<<, >>`	`L->R`	Insertion/extraction
`==, !=`	`L->R`	Equality
`&&`	`L->R`	And
`||`	`L->R`	Or
`=`	`R->L`	Assignment



Expression Evaluation

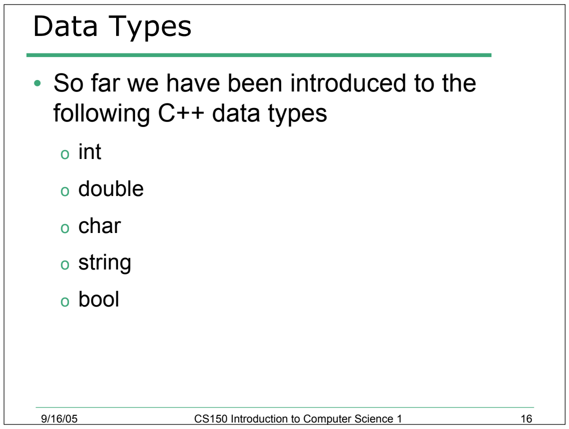
- According to the operator precedence and associativity rules given on the previous slide, how will the following expressions be evaluated?
  - **`x < min + max`**
  - **`min <= x && x <= max`**
  - **`!x == y + 2`**
  - **`x = a + b % 7 * 2`**



`bool` Data Type

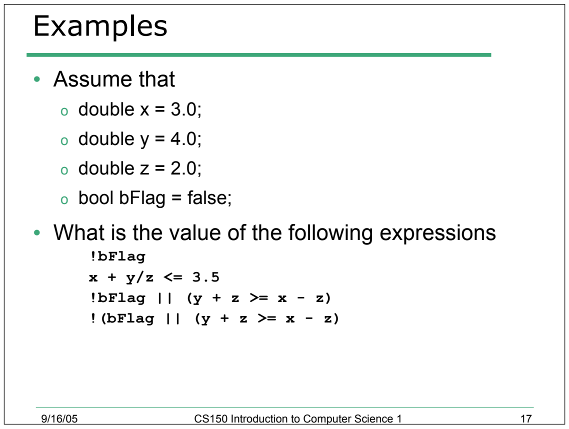
- **`bool`**: boolean
- Variables of type **`bool`** can be either **`true`** or **`false`**
  - They cannot be any other value
- Boolean variable names should start with b
  - See coding standards
- Example

```
bool bCanVote;
int age;
cin >> age;
bCanVote = age >= 18;
cout << bCanVote;
```



Data Types

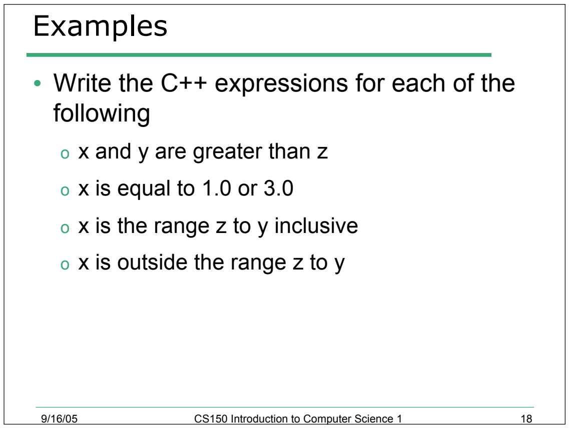
- So far we have been introduced to the following C++ data types
  - int
  - double
  - char
  - string
  - bool



Examples

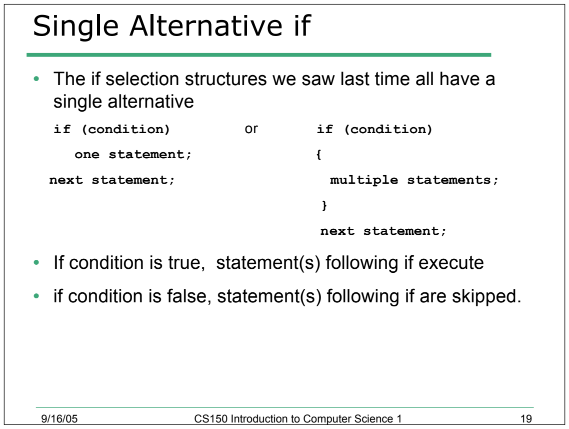
- Assume that
  - double x = 3.0;
  - double y = 4.0;
  - double z = 2.0;
  - bool bFlag = false;
- What is the value of the following expressions

```
!bFlag
x + y/z <= 3.5
!bFlag || (y + z >= x - z)
!(bFlag || (y + z >= x - z)
```



Examples

- Write the C++ expressions for each of the following
  - x and y are greater than z
  - x is equal to 1.0 or 3.0
  - x is the range z to y inclusive
  - x is outside the range z to y

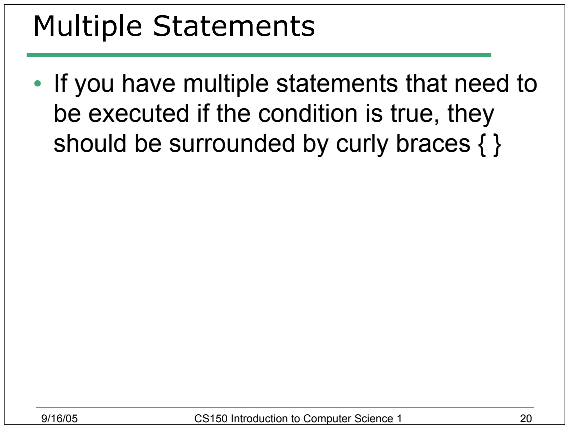


Single Alternative if

- The if selection structures we saw last time all have a single alternative

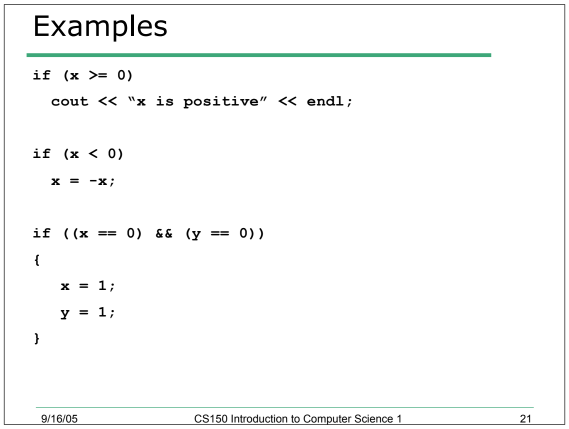
```
if (condition)          or       if (condition)
   one statement;                {
next statement;                    multiple statements;
                                  }
                                 next statement;
```

- If condition is true, statement(s) following if execute
- if condition is false, statement(s) following if are skipped.



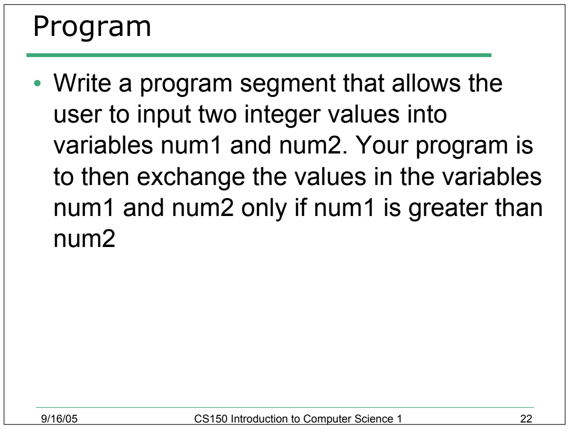
Multiple Statements

- If you have multiple statements that need to be executed if the condition is true, they should be surrounded by curly braces { }



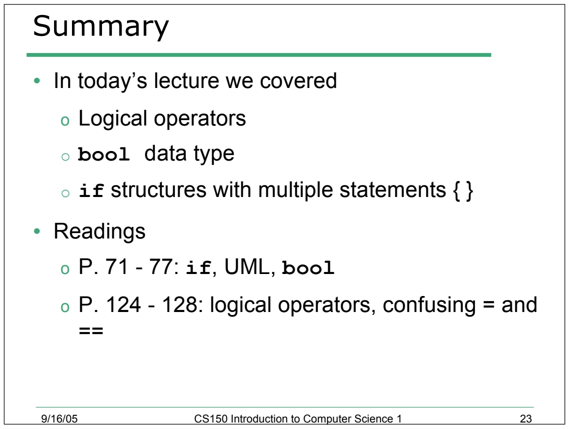
Examples

```
if (x >= 0)
  cout << "x is positive" << endl;

if (x < 0)
  x = -x;

if ((x == 0) && (y == 0))
{
   x = 1;
   y = 1;
}
```



Program

- Write a program segment that allows the user to input two integer values into variables num1 and num2. Your program is to then exchange the values in the variables num1 and num2 only if num1 is greater than num2



Summary

- In today's lecture we covered
  - Logical operators
  - `bool` data type
  - `if` structures with multiple statements { }
- Readings
  - P. 71 - 77: `if`, UML, `bool`
  - P. 124 - 128: logical operators, confusing = and ==