Ch 4. Making Decisions
Part 2

CS 1428
Fall 2011

Jill Seaman

Lecture 10

Logical Operators

- Operators over boolean values:
  - **&&** AND (binary)

    a && b is true when both a and b are true
  - **||** OR (binary)

    a || b is true when either a or b is true
  - **!** NOT (unary)

    !a is true when a is false

Logical Operators

- Examples

```
int x=6;
int y=10;

a. x == 5 && y <= 3
b. x > 0 && x < 10
c. x == 10 || y == 10
d. x == 10 || x == 11
e. !(x > 0)
f. !(x > 6 || y == 10)

bool flag;
flag = (x > 0 && x < 25);
g. !flag
h. flag || x < 100
```


Precedence and Logical Operators

- ! is higher than most operators, so use parentheses:

```
int x;

...  !(x < 0 && x > -10) ...
```

- && is higher than ||

```
int x, y;
...
... flag || x * 5 >= y + 10 && x == 5

   // which op is first? second? etc?
```

- && and || are lower than arithmetic+relational operators: parens not usually needed

Checking Numeric Ranges

- Want x to be in the range from 1 to 10 (incl)

```
a. if (1 <= x <= 10)
      cout << "YES" << endl;

   //NO, which op is done first?  second?


b. if (1 <= x && x <= 10)
      cout << "YES" << endl;

   -check: x=0?

   -check: x=5?

   -check: x=100?
```


Short Circuit Evaluation

- What is the value of: x != x && y > 10
  - true
  - false
  - don't have enough information to determine
- Actually it is false.
  - x!=x is always false.
  - false && ?? is always false
    - false && false is false
    - false && true is false

Short Circuit Evaluation

- If expression on the left of && is false,
  the expression on the right is not evaluated,
  the result is false.

- If expression on the left of || is true,
  the expression on the right is not evaluated,
  the result is true.

Watch out

- What is output?

```
int x=10, y=15;
if (x+y)
   cout << "x+y is true." << endl;
```

  - anything not 0 (zero) is true.
  - 0 (zero) is false.

- What is output?

```
int x;
cin >> x;
if (x = 5)
   cout << x << endl;
```

  - It always outputs: 5. Why??

Watch out

- What is output?

```
double x = 1.0/9.0;

if (x+x+x + x+x+x + x+x+x == 1.0)
   cout << "Nine ninths is one" << endl;
else
   cout << "Uh Oh" << endl;
```

  - Some fractional numbers cannot be stored exactly using binary (round-off errors)
  - Computation can compound these errors.
  - Don't test floating point values using equality