Input Validation with `if` Statements

Let's return to the elevator program and consider input validation.

Input Validation with `if` Statements

- Assume that the elevator panel has buttons labeled 1 through 20 (*but not 13!*).
- The following are illegal inputs:
  - The number 13
  - Zero or a negative number
  - A number larger than 20
  - A value that is not a sequence of digits, such as five ← non integers
- In each of these cases, we will want to give an error message and exit the program.

Input Validation with `if` Statements

It is simple to guard against an input of 13:

```
if (floor == 13)
{
   cout << "Error: "
      << " There is no thirteenth floor."
      << endl;
   return 1;
}
```

print an error message

Input Validation with `if` Statements

The statement:

```
   return 1;
```

immediately exits the `main` function and therefore terminates the program.

It is a convention to return with the value 0 if the program completes normally, and with a non-zero value when an error is encountered.

Input Validation with `if` Statements

To ensure that the user doesn't enter a number outside the valid range:

OR

```
if (floor <= 0 || floor > 20)
{
   cout << "Error: "
      << " The floor must be between 1 and 20."
      << endl;
   return 1;
}
```

err msg

Input Validation with `if` Statements

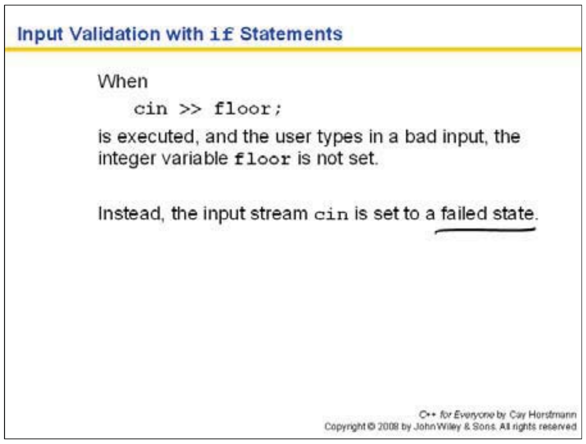
Dealing with input that is not a valid integer is a more difficult problem.

What if the user does not type a number in response to the prompt?

'F' 'o' 'u' 'r' is not an integer response.

↑
characters
string

"floor" cannot hold any values except integers



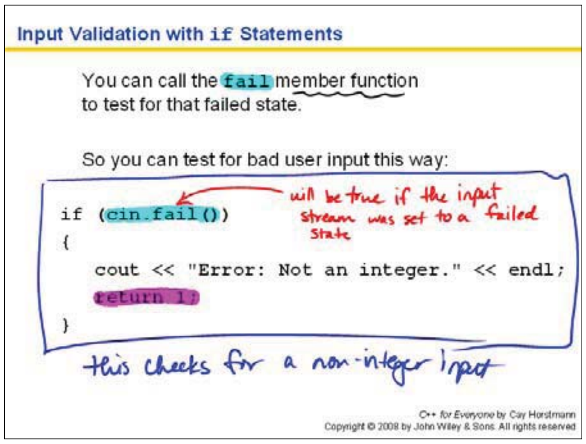
Input Validation with `if` Statements

When

```
cin >> floor;
```

is executed, and the user types in a bad input, the integer variable `floor` is not set.

Instead, the input stream `cin` is set to a failed state.



Input Validation with `if` Statements

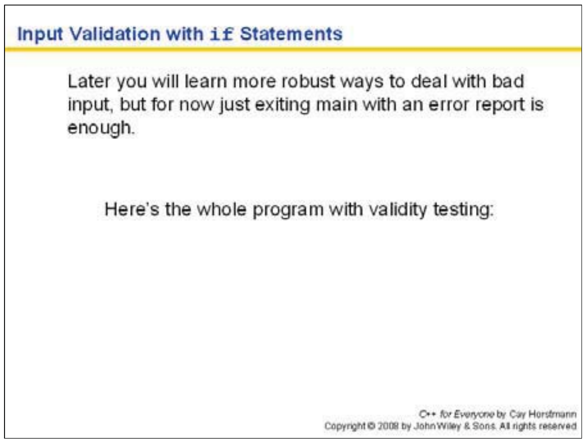
You can call the `fail` member function to test for that failed state.

So you can test for bad user input this way:

```
if (cin.fail())
{
   cout << "Error: Not an integer." << endl;
   return 1;
}
```

will be true if the input stream was set to a failed state

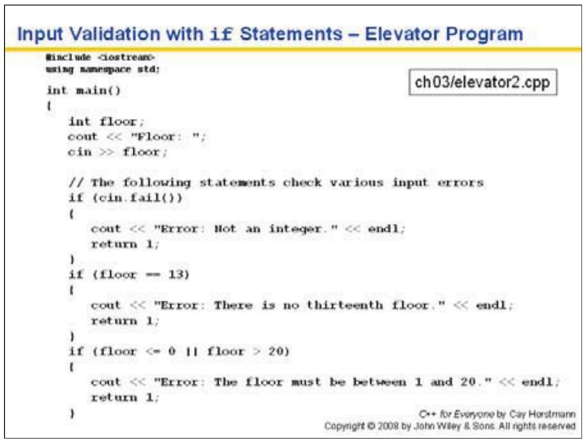
this checks for a non-integer input



Input Validation with `if` Statements

Later you will learn more robust ways to deal with bad input, but for now just exiting main with an error report is enough.

Here's the whole program with validity testing:



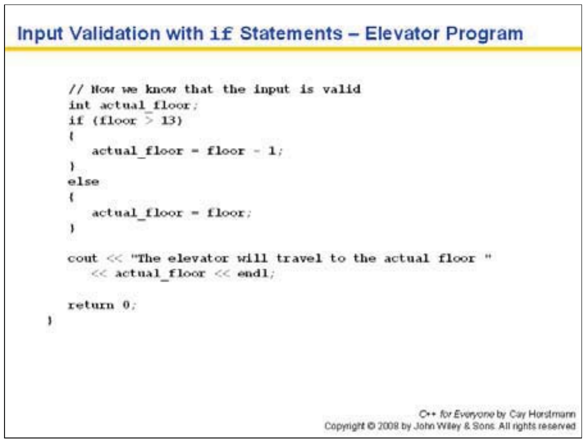
Input Validation with `if` Statements – Elevator Program

ch03/elevator2.cpp

```
#include <iostream>
using namespace std;

int main()
{
   int floor;
   cout << "Floor: ";
   cin >> floor;

   // The following statements check various input errors
   if (cin.fail())
   {
      cout << "Error: Not an integer." << endl;
      return 1;
   }
   if (floor == 13)
   {
      cout << "Error: There is no thirteenth floor." << endl;
      return 1;
   }
   if (floor <= 0 || floor > 20)
   {
      cout << "Error: The floor must be between 1 and 20." << endl;
      return 1;
   }
```

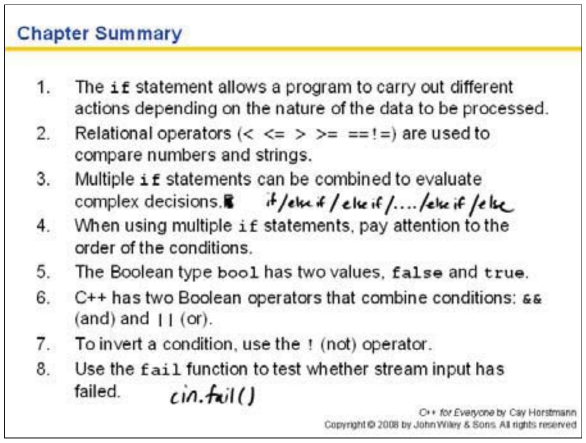


Input Validation with `if` Statements – Elevator Program

```
   // Now we know that the input is valid
   int actual_floor;
   if (floor > 13)
   {
      actual_floor = floor - 1;
   }
   else
   {
      actual_floor = floor;
   }

   cout << "The elevator will travel to the actual floor "
      << actual_floor << endl;

   return 0;
}
```



Chapter Summary

1. The `if` statement allows a program to carry out different actions depending on the nature of the data to be processed.
2. Relational operators (< <= > >= == !=) are used to compare numbers and strings.
3. Multiple `if` statements can be combined to evaluate complex decisions. if / else if / else if / .... / else if / else
4. When using multiple `if` statements, pay attention to the order of the conditions.
5. The Boolean type `bool` has two values, `false` and `true`.
6. C++ has two Boolean operators that combine conditions: `&&` (and) and `||` (or).
7. To invert a condition, use the `!` (not) operator.
8. Use the `fail` function to test whether stream input has failed. cin.fail()

Chp 3 Homework

p.121 R3.4 $\rightarrow |x| = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x < 0 \end{cases}$

p.127 P3.6

Write a program that will use the quadratic formula to find the real roots of a quadratic equation.

$$x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$$

Have user enter coefficients a, b, c. Do input validation.

Logging In

1. Restart the computer and hold down the Shift key when the blue screen comes up. Hold it until a login screen appears.

   Applying computer settings.

2. In drop down menu

   change Mount to Liblab... (this computer)

   username: cis_stu

   password: Msmc2010

3. Before leave, we must restart the computers (do not hold down shift this time)

Typing up programs

at prompt type:

$\underset{\uparrow \text{ capital E}}{\text{Edit}}$ $\underbrace{\text{nameofprogram}}_{\text{no spaces allowed}}\text{.cpp}$