Program Input and Output

- A very common pattern for programs to follow:
  - Get input from some source
  - Process that input
  - Show the results

User Input Statement

- Looks very similar to a print statement

```
cin >> x;
cin >> myVariable;
```

- Extraction operator (>>) tells the computer to read from a *input stream* and store in a variable
  - LHS argument is the input stream to read from
    - `cin` gets characters typed into that black box on the screen
  - RHS argument is the variable to store in

User Input Using the `iostream` Library

- The *extraction* operator (`>>`) is a built-in operator
  - It retrieves characters from an *input stream* and stores their value in a variable
  - Like insertion, this requires using the `iostream` library
- The `iostream` library defines the type `istream` (input stream)
  - Input streams move characters from an output device (the keyboard, a file, etc.) to the program
- The `iostream` library also declares the variable `cin`
  - `cin` is of type `istream` (i.e. `istream cin;`)
  - `cin` reads characters typed into the black box on the screen

Stream Input

- A stream handles characters in *sequential order*
  - E.g. Characters output to the screen in order
- A program gets characters from an input stream
  - In the order they are typed by the user
  - The program can only get one character at a time
    - It can get remove it from the stream or not
- The cin iostream *only* sends characters when the user presses the return key
- Working at the level of individual characters is tedious and error-prone
  - The extraction operator (<<) provides a higher level of abstraction for you to work with

Chaining Insertion/Extraction

- You can chain together insertion/extraction expressions in the same statement

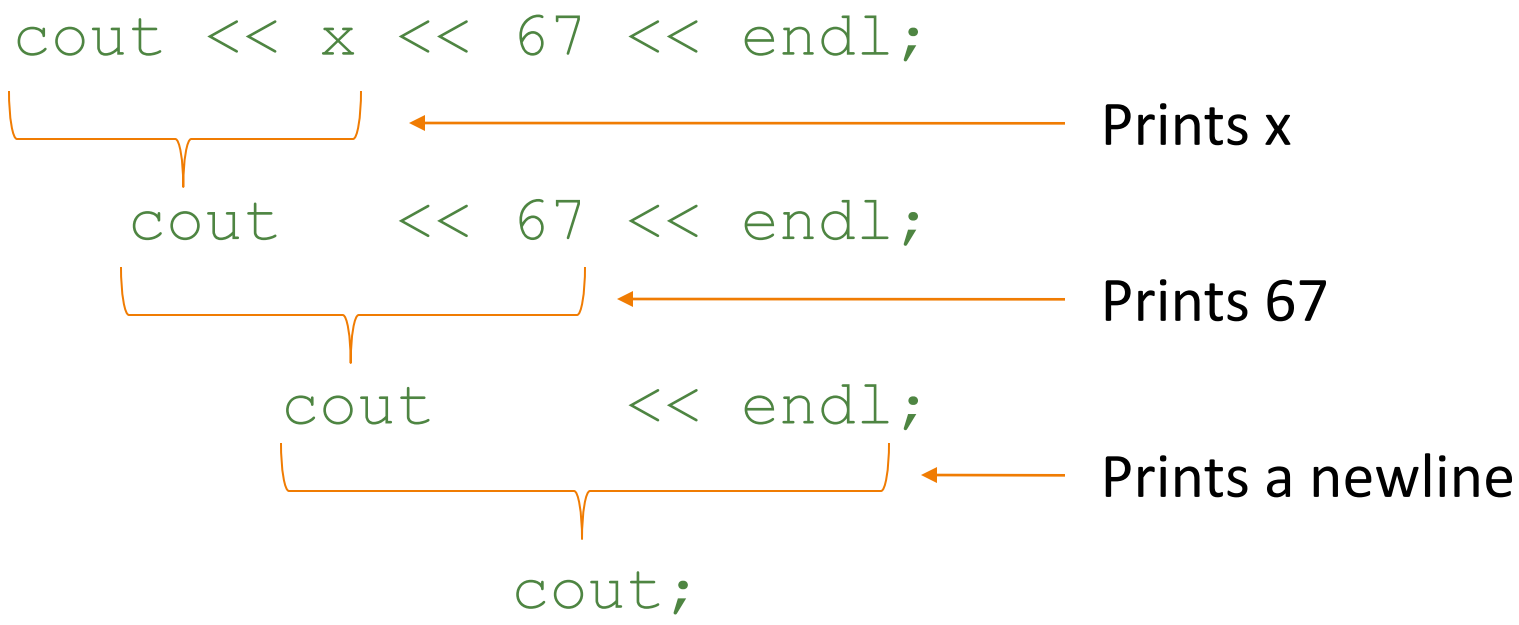
```
cout << x;
cout << 67;
cout << endl;
```

  - Does the same thing as:

```
cout << x << 67 << endl;
```

Chaining Insertion/Extraction

- This is possible because:
  - Every expression evaluates to a value
  - The insertion and extraction operators evaluate to the value of their LHS argument (the stream)
- For example:



Chaining Insertion/Extraction

- Extraction is chained in the same way

```
cin >> x;
cin >> y;
```

  - Is the same as

```
cin >> x >> y;
```

- Common mistake:

```
cin >> x >> endl;
```

  - Attempts to read characters into the variable `endl`, which is not a variable
  - Results in an error

Extraction Rules

- User input is more complicated than output
  - You expect certain data…
  - …but have to deal with it if they type something else
  - (You don't control what the user types)

- So what *algorithm* (set of steps) does the extraction operator use to turn individual characters into a proper value for the given variable?

Extraction Rules

- Figuring it out
  - You know 2 things going in about stream input:
    - It works with characters
    - It can only look at one character at a time
  - Try examples, see what ends up in the variable
    - Can use the debugger inspector to examine variables
    - How does it decide when to take a character, when to stop?
    - How does it combine the characters into a single value?
  - Test your conclusions with another example

Extraction Rules

Can you predict what that value will be, given certain input?

(note the spaces in the input!)

```
int x;
cin >> x;
```

The user types…	Value of x is…	Left on the stream is…
34	34	\n (newline character)
78 94 42	?	?
901abh29ks	?	?
-15.4	?	?
9a9a9	?	?
jk	?	?

Extraction Rules

(note the spaces in the input!)

```
double x;
cin >> x;
```

The user types…	Value of x is…	Left on the stream is…
78.56 94.2 42.09	?	?
-901abh29ks	?	?
67.84.29.19	?	?
jk	?	?

Extraction Rules

(note the spaces in the input!)

```
char x;
cin >> x;
```

The user types…	Value of x is…	Left on the stream is…
78 94 42	?	?
901abh29ks	?	?
901abh29ks	?	?
jk	?	?

Extraction Rules

(note the spaces in the input!)

```
string x;
cin >> x;
```

The user types…	Value of x is…	Left on the stream is…
78 94 42	?	?
901.23ab%!@h29ks	?	?
The rain in Spain	?	?
jk	?	?

Extraction Rules

```
int x, y;
char ch;
```

For the input:

```
5 28 36
```

What are the values of `x`, `y` and `ch` after:

```
a. cin >> x >> y >> ch;
b. cin >> x >> ch >> y;
```

Extraction Rules

```
int x, y;
double z;
```

For the input:

```
37 86.56 32
```

What are the values of `x`, `y` and `z` after:

c. `cin >> x >> y >> z;`

d. `cin >> z >> x >> y;`