Streams

Reid Watson
(rawatson@stanford.edu)

Prelude: Code Snippets

I often present code snippets like:

```
cout << ((5 * 4) << 1) + 2 << endl;
```

Prelude: Code Snippets

When you see a code snippet like that, interpret it as:

```
#include <iostream> // + other headers
using namespace std;

int main() {
   cout << ((5 * 4) << 1) + 2 << endl;
   return 0;
}
```

Prelude: Conventions

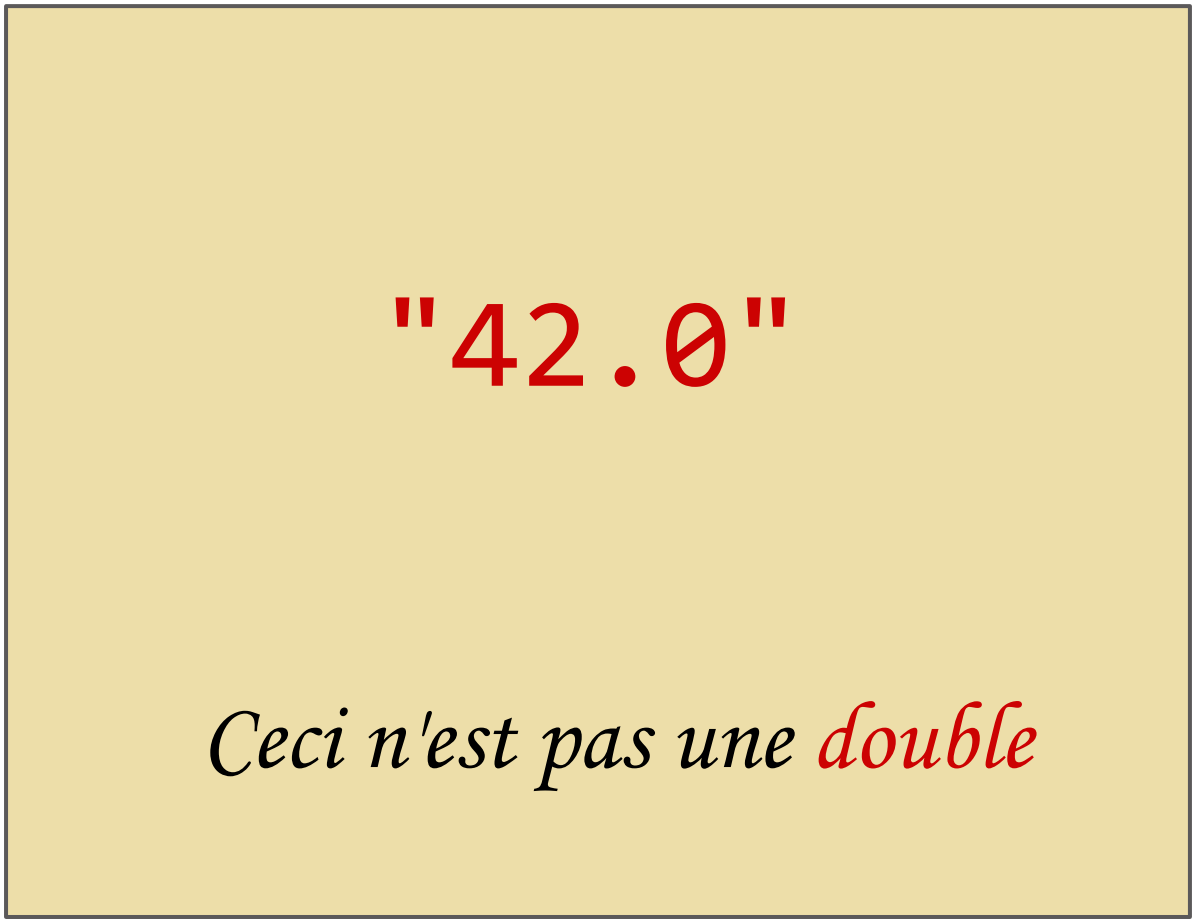
This is unstyled text, describing my lecture conventions.

```
// This is a C++ code snippet
cout << “Isn’t it great?” << endl;
```

The word **lederhosen** is an important topic that you should think critically about.

The Ideas Behind Streams

The Ideas Behind Streams



The Ideas Behind Streams

```
// In C++ terms

// What's the difference between:
int number = 8675309;

// and
string text = "8675309";
```

The Ideas Behind Streams

- This doesn't work:

```
// Print five times a number
void multiplyBy5(string s) {
   cout << s * 5 << endl;
}
```

The Ideas Behind Streams

- Does this work?

```
// Print a number with 4 at the end
void appendFour(int s) {
  cout << s + “4” << endl;
}
```

Output streams

Let's take a look at why this distinction matters

See code in NuclearLaunchFacility.pro

The Ideas Behind Streams

- When we read data from the user, we must read it in the form of text (`string`)
- When we write data to the user, we must write it in the form of text (`string`)
- But to perform computations on data, we need it in the correct C++ type (`int`, `double`, etc.)

The Ideas Behind Streams

Streams allow a C++ programmer to convert between the string representation of data, and the data itself.

What is a Stream?

- A **stream** is a C++ object which can send or receive data
- **cout** is a stream, which prints output to the user when it receives data

```
// Send the string “Hello world!”
// to the stream cout
cout << “Hello world!” << endl;
```

What is a Stream

- Any stream which can receive data (like `cout`) is called an **output stream**, or `ostream`
- We send using this little guy: `<<`
- This sends data to an ostream, converting it to a string in the process

What is a Stream?

- The idea of `ostream` can be applied to more than printing text to the console
- You can print text to a file using an `ofstream`
- Let's take a look at an example of this in OutputStreams.pro

What is a Stream?

- Output streams are conceptually simple:
  - Stream gets data
  - Stream converts data to string
  - Stream writes the string to wherever it should go (a file or the user's terminal)
- Let's look at a more complex example of why streams are useful

What is a Stream?

Of course, you've probably seen code like this too:

```
#include <iostream>
using namespace std;
int main() {
   int x;
   cin >> x;  // the important line
   cout << x << endl;
}
```

What is a Stream

- Any stream which can give you data (like `cin`) is called an **input stream**, or `istream`
- This is done using this little guy: `>>`
- This gets data from an istream, converting it from a string to the appropriate type in the process

What is a Stream?

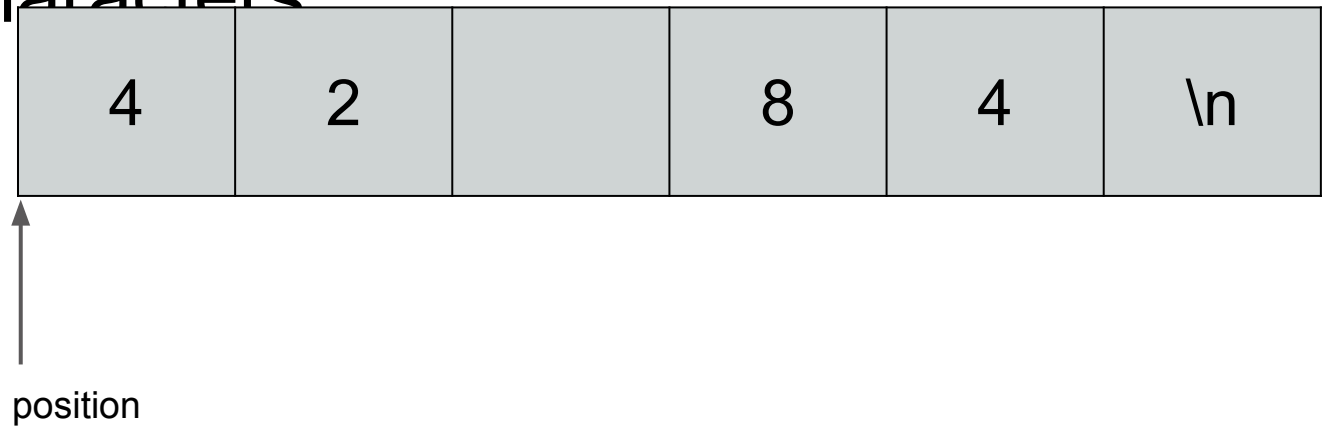
- Just like `ostream`, `istream` can be used for more than console input
- You can read text from a file using an `ifstream`
- Let's take a look at an example of this in InputStreams.pro

What is a Stream?

Let’s take a look at how we read data from a file to understand input streams better.

Reading Data From a File

The easiest way to understand an istream is to think of an istream as a sequence of characters



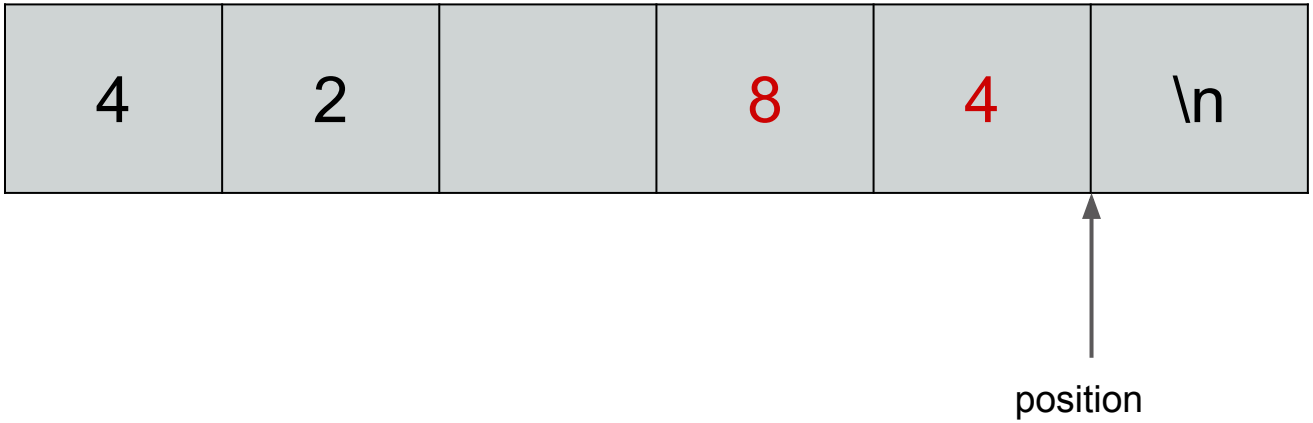
Reading Data From a File

Reading an integer will read as many digit characters as possible into an int.


```
int value;
istream >> value; // value == 42
```

Reading Data From a File

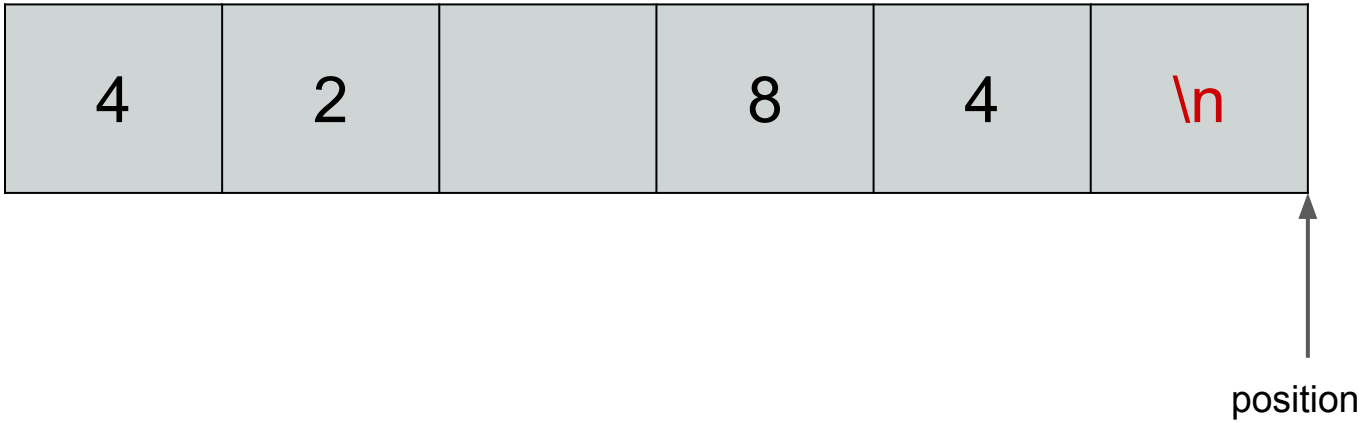
Reading data will skip over any whitespace before the next sequence of digits.




```
int value;
istream >> value; // value == 84
```

Reading Data From a File

When it's no longer possible to read valid data from a file, the **fail bit** will be set.




```
int value;
istream >> value; // value == ?
```

Reading Data From a File

Reading strings from a file has a couple nuances

- Reading into a string using >> will read a single word, not a line
- To read a line, use the getline(istream&, string&) function
- We'll also see how to read a file once you've already read it once.

Reading Data from a file

Let's take a look at how `istream`s work with strings in code

See code in InputStreams.pro

Input Streams

- Writing data to an **ostream** is fairly simple -- just convert the data into a string and then append that string to the stream.
- Reading data from an **istream** is a bit more complicated. We need to understand the rules which govern reading data from a string.
- We also need to think about what happens when we do something invalid, like try and read the string "foo" into an int.

Interlude: Stringstreams

- There's another type of stream: a **string stream**
- Unlike other streams, string streams don't send data anywhere
- They store data in a temporary string object
- This makes them useful for converting between `string` and int (or `double`, `float`, etc.)

Playing Around With Stringstreams

Let's take a look at how stringstreams work in code

See code in StringStream.pro