I/O (1)

Streams, functions, and cin, and string input
By the end of this lesson you will be able to:

- Understand what a stream is
- Understand how `cin` works in better detail
- Understand the concept of pre-defined functions
- Use `cin.get`, `cin.ignore`, `cin.peek`, and `cin.putback`, and `cin.clear` functions
- Understand and deal with input failure
- Accurately get input into a string from a stream
Streams

- A **stream** is a sequence of bytes (usually characters) from a source to a destination
- **Input stream:** data flows from an input device to the computer
- **Output stream:** data flows from the computer to an output device
Streams

• There are two datatypes defined in the `iostream` header: `istream` and `ostream`.

• There are also two specific variables already defined in `iostream`:
  - `cin` (common input) which has datatype `istream`
  - `cout` (common output) which has datatype `ostream`
Stream functions

- Variables of type `istream` and `ostream` have certain **functions** associated with them.

- The operators `<<` (for `ostream`) and `>>` (for `istream`) are two operator functions.

- Other functions on `istream` variables include `get`, `ignore`, `peek`, `putback`, and `clear`. 
Extracting from a stream

- You can extract data from an `istream` variable (such as `cin`) using the stream extraction operator `>>`
- `>>` ignores whitespace and also cuts the stream off after a whitespace
- `>>` will also cut the stream off once all possible sufficient valid data has been read into the variable
  - For a `char`, one character is read
  - For an `int`, only digits are read
- If data remains on the stream after data has been stored in a variable, that data will remain there for the next extraction
- If there is another extraction, the program waits for input
Using pre-defined functions

- **Functions** exist in C++ similar to how they are used in math
  - If you have \( f(x) = 2 \times x \), then you can calculate \( f(5) \), \( f(0) \), \( f(300) \), etc.
- Pre-defined functions in C++ come from the use of headers
- There are special datatypes called **classes**, whose variables are called **objects**, and there are functions built in
  - Example: `stringvar.length()` — available from `string` header
- Other functions exist on their own:
  - Example: `pow(x, y)` — available from `cmath` header
- Functions may do something, return a value, or both
cin functions

- **cin** is an object of type **istream** which has several functions built in and can be used as follows:
  - `cin.get(charvar)` – get one character from the input stream (even a space!) and store it in `charvar`
  - `cin.ignore()` - ignore the next character in the input stream
  - `cin.ignore(intvalue, charvalue)` – ignore the next `intvalue` characters in the input stream, or until we encounter a `charvalue`, whichever comes first
  - `cin.putback(charvalue)` – add `charvalue` to the stream
  - `cin.peek()` - evaluates to the next character in the stream without removing it from the stream

- Note the **dot operator** or **member access operator**
Input failure

- If you try to store stream data into a variable of the wrong datatype, **input failure** occurs.

- If an input failure occurs, erroneous output may result and all further input is ignored from the stream.

- You can restore the input status from failure to working by using \texttt{cin.clear()}

- Useless data may remain on the stream, so you can use \texttt{cin.ignore (with a large integer value)} to remove the bad data.
In practice: string parsing

- In practice, the most common thing to do with a lot of data where users may enter erroneous values is to read in a string and then parse the string.
- Parsing we will examine in more detail, but we can look at string input right now.
string input

- Using `cin >> stringvar;` has the problem that if a user types a whitespace, no more data is read from the stream.
- To read all input until enter is hit, use this function (from the `string` header) instead:
  ```cpp
getline(istreamvar, stringvar)
```
- `istreamvar` can be any `istream` variable, such as `cin`.
- All whitespaces will be included in the string!
Exploration

• Think about all the variable data types you know, what values (that can be entered by a keyboard) would cause an input failure, if any?

• A doctor wants to enter a code such as SFA30W160 where S, A, and W refer to sex, age, and weight – write a program that accurately processes that input into three different variables and then outputs the information in a nicely formatted way