PIC 10A: Week 1

Section 1A, Spring 2018

Prof. Ernest Ryu (TA: Hui Jin)

C++: Dissecting a simple program

C++, line by line

Question: What happens when I try to compile+run this program?

```
#include <iostream>
using namespace std;
// Will print "Hi!" to the screen.
int main() {
  cout << "Hi!\n";
  return 0;
}
```

Answer: Simply outputs "Hi!" to the user, then exits immediately.

C++, line by line: include

Include statement.
Purpose: Unlocks additional functionality for the program.

Syntax: #include LIBRARYNAME

```
#include <iostream>

using namespace std;
// Will print "Hi!" to the screen.
int main() {
  cout << "Hi!\n";
  return 0;
}
```

What is a Library?

- A library is collection of code that has functionality that will likely be useful to other programs.
- Example: If you want your program to have a user interface (ie windows, buttons), then you'll need to find a **graphical user interface library** (GUI).
- Example: If you want your program to recognize faces in a picture, you'll want to use a **face detection library**
- Lots of people release libraries online that are free to use!
    - Open source code: code that is free for use by anyone

C++ Standard Libraries

- Most languages (including C++) offer standard, "built-in" libraries
  - Common: File reading/writing, text manipulation, core data structures
- Popular C++ standard libraries include:
  - iostream, string, random
- List of standard libraries here:
  - http://en.cppreference.com/w/cpp/header

iostream

- Purpose: "...defines the standard **input/output** stream objects."
- The documentation about iostream says it defines: cin, **cout**, cerr, clog
  - http://www.cplusplus.com/reference/iostream/
- So, including iostream tells our program that **cout** *exists*.

What if we removed the include?

Question: What happens if I try to compile this program?

Answer: The program doesn't compile!
Error: "cout" is an undeclared identifier.

```
#include <iostream>

using namespace std;
// Will print "Hi!" to the screen.
int main() {
  cout << "Hi!\n";
  return 0;
}
```

Aside: cout vs cin vs cerr vs clog

- cout: "Console Out", aka "standard out"
  - Writing to cout -> output text to user
- cin: "Console In", aka "standard in"
  - Reading from cin -> get text/number input from user
- cerr: "Console Error", aka "standard error"
  - Writing to cerr -> output warnings/error-messages
- clog: "Console Log"
  - Writing to clog -> output text relating to logging/debugging/whatever-you-like

In this class: focus on cout and cin.

Note: cerr, clog are meant for programmer, not for the user.

C++, line by line: namespaces

Purpose: Introduces variables/functions from a *namespace* into your program.

Syntax: using namespace ID;

```
#include <iostream>

using namespace std;

// Will print "Hi!" to the screen.
int main() {
  cout << "Hi!\n";
  return 0;
}
```

using namespace std;

- Tells compiler we are using the "standard namespace"
  - std: "standard"
- Imports all of the functions/variables that a namespace **defines**
- **Example**: the `std` namespace defines `cout` and `cin`
  - More generally: all C++ standard library identifiers live in the `std` namespace

In this class, we will only use the standard namespace: std.

What if we remove "using namespace std;"?

Question: What happens when I try to compile+run this code?

```
#include <iostream>

using namespace std;

// Will print "Hi!" to the screen.
int main() {
  cout << "Hi!\n";
  return 0;
}
```

Answer: Program doesn't compile!
Error message: cout is an undeclared identifier.

With/Without using namespace std

```
#include <iostream>
using namespace std;
// Will print "Hi!" to the screen.
int main() {
  cout << "Hi!\n";
  return 0;
}
```

With

```
#include <iostream>
// Will print "Hi!" to the screen.
int main() {
  std::cout << "Hi!\n";
  return 0;
}
```

Without

Verdict: "using namespace std;" simply lets us not have to type "std::" a bunch of times.

std::cout means to access the identifier "cout" from the namespace "std".
Anything that the C++ **standard library** defines lives in the **std namespace**.

C++ line by line: Comments

Comment
Purpose: Provide information or explanation useful for a programmer/reader.

Computer **ignores** everything you put in a comment.

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

// Will print "Hi!" to the screen.

int main() {
  cout << "Hi!\n";
  return 0;
}
```

C++ line by line: Comments

Question: What happens when I try to compile+run this program?

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

// Will print "Hi!" to the screen.

int main() {
  // cout << "meow" << endl;
  cout << "Hi!\n";
  return 0;
}
```

Answer: Compiles correctly, and outputs "Hi".
The "meow" isn't output because it's part of a **comment**.

Multiple ways to comment

```
// (1) Single line comments must always start
//     with two forward slashes.

/* (2) Anything in here is
       considered to be
a comment.
*/
```

(1) Single-line comments
(2) Multi-line comments

C++ line by line: the main() function

main
Purpose: Contains code that actually runs when you run the executable.

```
#include <iostream>
using namespace std;
// Will print "Hi!" to the screen.

int main() {
  cout << "Hi!\n";
  return 0;
}
```

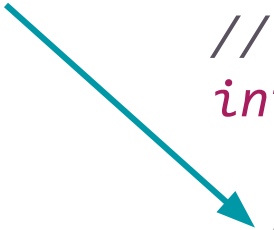
The main() function

- The *return value* of the main function is known as the **status code**
- As convention, 0 means that the program terminated normally.
- non-zero return values (ie -1) mean that the program exited abnormally
    - Examples: File wasn't found, invalid input, etc.

C++ line by line: cout

Purpose: Output text to the user.

`cout`: Console output
Defined by: `<iostream>`



```
#include <iostream>
using namespace std;
// Will print "Hi!" to the screen.
int main() {

  cout << "Hi!\n";

  return 0;
}
```

cout: Chaining

- Can chain "<<" together to output multiple things
- Example: `cout << "I am taking " << 3 << " classes this quarter.\n";`
- Outputs: `I am taking 3 classes this quarter.`

cout: numbers

- cout understands numbers as well!
- Examples:

```
cout << "I am " << 26 << " years old.";
```
Outputs:
```
I am 26 years old.
```

```
cout << "There are " << 42+57 << " red balloons.";
```
Outputs:
```
There are 99 red balloons.
```

"Special" characters, ie \n, \t,

- We've seen that "\n" is special: it creates a new line. Known as the new-line **escape sequence**.
- Other escape sequences:
  - \t Tab
  - \" Double-quote
  - \' Single-quote
  - \\ Back-slash
  - \a Creates an audible beep
  - \b Moves the cursor backwards, ie to the left.

Exercises: cout

Question: What do the following output? If it errors, explain the error.

```
cout << "For" << "No\n";
cout << "One";
```

Answer:

```
ForNo
One
```

```
cout << "Toe\n";
cout << "\n" << "To " << "Toe";
```

Answer:

```
Toe

To Toe
```

Exercises: cout

Question: What do the following output? If it errors, explain the error.

```
cout << ""Hello"" << "Goodbye";
```

Answer:

```
Compile error!
The word Hello is not
contained within double-
quotation marks, so it
doesn't make sense.
```

```
cout << "Revolution " << "3+6";
```

Answer:

```
Revolution 3+6
```

Exercises: cout

Question: Write some code that will exactly generate the following output:

```
I "love" waking up at 6 AM!
```

Answer:

```
cout << "I \"love\" waking up at 6 AM!";
```

Exercises: cout

Question: Write some code that will exactly generate the following output:

```
I "love" waking up at 6 AM!
```

Question: Is the following answer correct?

```
cout << "I " << " << "love" << " << " waking up at 6 AM!";
```

Answer: Nope! This will actually error.

```
cout << "I " << " << "love" << " << " waking up at 6 AM!";
```

String 1 String 2

Uhoh, what's that?
Error!

cout: endl

- Alternative to typing "\n" a bunch of times: endl
  - Stands for: "end line"

```
cout << "Hi there\n" << "Face here";
```

outputs the same thing as:

```
cout << "Hi there" << endl << "Face here";
```

Output:

```
Hi there
Face here
```

String Literal

- To create a string literal, wrap some text with double quotation marks
- Examples: "Hi there", "3+4", "bye\n" are all **string literals**
    - We've been creating string literals all along!

String Literals

- Important: Computer will not "execute" contents of string literals. Leaves the contents as-is.
- Example: `cout << "3+4";`
  - Outputs: 3+4, not 7
- **Exception**: Escape sequences. `\n, \t, \\, \", \'`
  - Example: `cout << "hi\nthere";`
  - The `\n` is **expanded** out to a new-line.

Exercise

Question: Write some code that outputs the following:

```
I put
a newline \n there!
```

Answer:

```
cout << "I put\n" << "a newline \\n there!";
```

or:

```
cout << "I put" << endl << "a newline \\n there!";
```