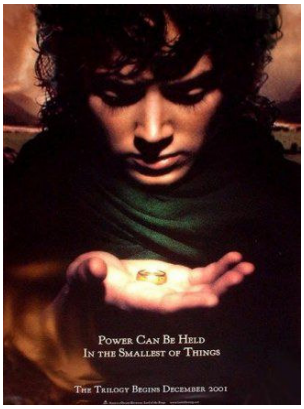
Lecture 1
Introduction: Dissecting Hello World!

PIC 10A

Todd Wittman



Syllabus Highlights

- The textbook is Big C++, *2nd ed*. You can buy either version of it at the bookstore.
- Yesterday your TA took you to the PIC Lab in Boelter 2817. Most students do their HW there.

- Midterms are in class Mon 2/8 and Mon 3/1.
- Final is Sun 3/14, 3:00-6:00pm.

- This is a 5-credit course. So we expect a lot of you!

Grading System

•Your grade will be calculated two ways.
•The method that gives the higher numerical score will be used to determine your course letter grade.

- Exam 1 20%
- Exam 2 20%
- Final Exam 30%
- Homework 30%

- Drop lowest midterm.
- One midterm exam 20%
- Final Exam 50%
- Homework 30%

Homework

- Homework is due every Friday 5:00pm, starting next week. **No late HW will ever be accepted.**
- You must place the specified files in the Submit folder. **We will not accept HW by e-mail.**
- **You must use Microsoft Visual Studio 2005.**
- Microsoft software will be available for download after the 2nd week. Or you can bring 2 blank CDs to the PIC Lab and burn a copy of Visual Studio.
- You can work at home, but you are responsible for correctly submitting the HW.

- Don't wait to the last minute on the HW!!!

Sec 1.1: What is A Computer?

Your textbook has a hilarious answer.

Page 2:

"You have probably used a computer for work or fun. Many people use computers for everyday tasks such as balancing a checkbook or writing a term paper. Computers are good for such tasks."

What is a program?

Page 3:

"A *computer program* tells a computer, in minute detail, the sequence of steps that are needed to fulfill a task. The act of designing and implementing these programs is called *computer programming*."

A program is a set of instructions for the computer to follow.

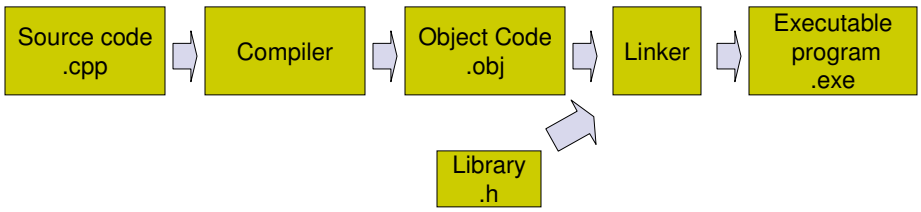
In order for the computer to understand your instructions, you have to learn a certain syntax or set of grammar rules.

C++ As A Foreign Language

- Learning C++ is like learning a foreign language.
- You have to learn the basic words, but also the grammar for arranging the words sensibly.
- This class is a lot like a language class. We're not going to teach you every word or phrase that you'll ever need.
- We give you the basics and you have to learn how to form new phrases on your own.
- We do not expect you to parrot back the examples given in class. We expect you to build upon them.
- Programming is a creative activity!!!

Sec 1.10: The Compilation Process

Compiling changes the source code into instructions the computer can understand.



- 2 Types of errors
  - Compile errors (missing ; , mis-spelling, linking error)
  - Run-time errors (logic error, roundoff, calculations, a program that doesn't end)

The Project Folder

- In Visual Studio, you build a project folder.
- There are many files in this folder, which are needed for the computer to run the program.
- Source File
  - The cpp file is a text file where you type your program code.
  - This is what you will submit for HW.
- Solution File
  - This file organizes the pieces of your project.
  - To work on your program, click on this.
- Executable File
  - This is located in your debug folder.
  - Clicking on this will run your program like an application.

hello.cpp

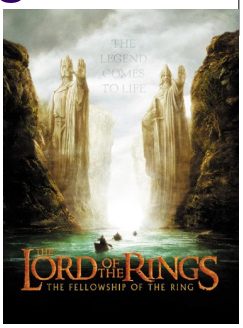
testing.sln


Testing.exe

Sec 1.8: A Simple Program

Tomorrow you will run the following program:

```
# include <iostream>
using namespace std;
int main() {
        cout << "Hello Middle Earth!\n";
        return 0;
}
```



The "Hello World!" program is the universal introductory program. It's like a freakin' law or something. I changed the planet for fun.

Let's examine this line by line.

The explanation may not make perfect sense today, but hopefully it will make some sense.

We'll look at input/output, variables, and functions in greater detail later in the course.

```
# include <iostream>
using namespace std;
int main()    {
            cout << “Hello Middle Earth!\n”;
            return 0;
}
```

The first line tells the computer to include the library iostream.h
Libraries are called “header files” and end in .h

There are many libraries we might call:

- iostream -- Input and output stream, needed for cout
- cmath -- Math functions like cos, sin, and log
- string -- Maninpulates text strings like “hello”
- MyLibrary.h -- We can define our own header files

Without the first line, we would get an error for the term “cout”.
In general, every program starts with: # include <LIBRARY>

```
# include <iostream>
using namespace std;
int main()    {
            cout << “Hello Middle Earth!\n”;
            return 0;
}
```

Tells the computer to use the standard namespace.
So when the computer comes to the unknown term “cout”, it knows to look up the term in the included library.

The namespace is a fairly new addition to C++.
Added for large projects with many programmers.
Just write this statement at the top every time.

Note this line ends in a semi-colon.
Every statement in C++ ends in a semi-colon, but not necessarily every line.
Forgetting the semi-colon is the single most common error.

```
# include <iostream>
using namespace std;
int main()    {
            cout << “Hello Middle Earth!\n”;
            return 0;
}
```

The function main() is called the driver.
This is the first place the computer looks to run code.

The () indicate that no arguments are passed to the function.
(More on this later when we discuss functions.)

The “int” means that the program is expecting an integer when we’re done.
The C++ convention is to send the computer a zero if everything ran smoothly.
We can send other numbers to indicate errors without crashing.

The { } enclose the code.
Balance your brackets and parantheses!

```
# include <iostream>
using namespace std;
int main()    {
            cout << “Hello Middle Earth!\n”;
            return 0;
}
```

The “return” command sends a value to the computer, telling us that the function main is complete.

Remember main expected an integer. So the last statement of the main routine should send out a number.

We return zero to say everything is A-OK.

Sending a different number could indicate an error without crashing the program.

Except for the one "cout" line, you can use this basic form for your beginning programs.

```
# include <iostream>
using namespace std;
int main()   {
        ** STATEMENTS **
        return 0;
}
```

This is the standard boiler-plate C++ program.

You should get used to this, as your first few homework assignments will look just like this.

The ** STATEMENTS ** part could be very long and could even call other functions.

Note we generally indent the STATEMENTS, so it's clear they are within the brackets { } of main.

```
# include <iostream>
using namespace std;
int main()   {
        cout << "Hello Middle Earth!\n";
        return 0;
}
```

The cout statement writes whatever follows to the console (cout = "console out").
The console is the black screen that popped up when you ran your program.
(How do you think you would get input from the screen? Use cin)
Remember we need to #include <iostream> to use cout.

The angle brackets << push whatever follows to screen.
(How would you pull input from the screen? Use >>)

The output to the screen should be enclosed in quotes " ". Balance your quotes!

The \n does not appear on the screen.
It acts as a carriage return and skips to the next line.

The backslash \ is a special escape character.
A character that follows will not appear on screen.
Instead it gives a special command to the output.

More fun with cout

Noteworthy escape characters

\n Carriage return, skip to next line.

\" Put a quotation mark " on the screen.

\\ Put a backslash on the screen.

Escape characters should appear inside the quotes "".

We can push numbers to the screen, e.g.

cout << 22; SHOWS: 22

cout << 7+2*5; SHOWS: 17

We can chain the pushes << to output numbers and text, e.g.

cout << "2+3 is " << 5 << "\nNot " << 6 << "!";

SHOWS: 2+3 is 5
Not 6!

Question: Where will the next cout statement start?

You should plan ahead for the next output statement, if it starts on the next line.

Sec 1.8 Worksheet

1.) cout << "My precious!\n";

My precious!

2.) cout << "My \n precious!\n";

My
precious!

Note the second line is indented one space.

3.) cout << "n\nn";

n
n

4.) cout << "n\\nn";

n\nn

5.) cout << "There are 4 hobbits.";

There are 4 hobbits.

Sec 1.8 Worksheet, cont'd.

6.) cout << "There are " << 4 << " hobbits.\n";

There are 4 hobbits.

7.) cout << "There are" << 4 << "hobbits.\n";

There are4hobbits.

8.) cout << "There are " << 2 + 2 << " hobbits.\n";

There are 4 hobbits.

9.) cout << "There are " << "2+2" << " hobbits.\n";

There are 2+2 hobbits.

10.) How do we output "My precious!" with quotes?

cout << "\"My precious!\"";

or if we want to pick up at the next line:

cout << "\"My precious!\"\n";

A couple more questions...

What's wrong with the following?

cout << "\"Hello."" ;

cout << "Hello."\n;

cout << 1 << "is a lonely number.\n";

cout << "The" << 3*3 << "rings"