PIC 10A

Lecture 1: Hello World!

The Very Basics

What is a computer?

- Seriously?

What is a computer program?

- Series of commands that the computer interprets and executes.

- For a program to accomplish a task the commands must be put together in a meaningful way.

What are the ingredients of a program that does something?

- The program must be written in a programming language that the computer can understand. There are several languages: Pascal, Fortran, Basic, C,
 C++, Java etc.

- The program must follow the rules of the language. The programmer must know the words and the grammar.

- The instructions must be written in the correct sequence.

- The totallity of instructions must accomplish the intended task.

- Often the program must be translated into a machine readable language.

- The program must be run on a correct platform.

What is studying C++ like?

- It is much like learning a foreign language.
- You learn new words and grammar. The commands are much like words, and grammar is the interaction of the commands.
- This class is not about memorizing every single phrase.
- Think of the things you learn as tools. As you learn more of the language, you will be able to build more complex programs.
- Programming is a creative activity!

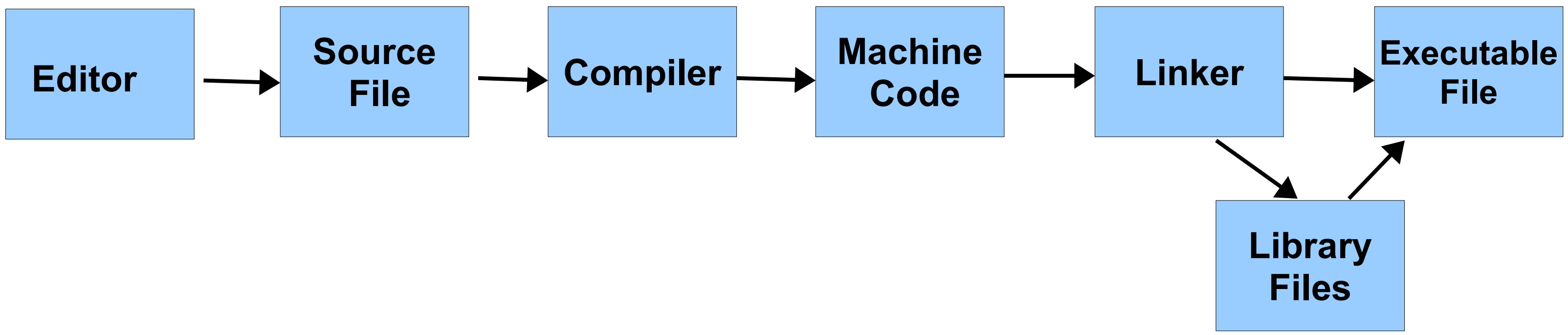
How is a C++ program written?

- It is written using the normal English alphabet and a few extra symbols such as parenthesis ( ) and semicolons ; .
- You can write a C++ program on any text editor!
- It is, however, easier to use a program such as Microsoft Visual Studio to help you write the code.

How do you run a C++ program?

Before you have a working program your code must be converted to an executable file.

The computer does not understand C++ as it is written by humans. It must be translated into machine code. The compliation process accomplishes this.



A source file has a name like `my_program.cpp.` The machine code version has name like `my_program.obj` and finally the executable has a name like `my_program.exe`.

We do not need to understand the details of this process. We only need to know the main idea. The end product is essentially a program of 0s and 1s that the computer understands.

I wrote my program so I am done with the HW assignment right?

Sadly, writing a computer program itself is probably only half the work...Three things can go wrong:

Compilation errors

- Syntax errors, typos etc.

Run time errors

- Program crashes as it is running.
- Possible reasons: Program tried to access illegal memory location or tried to divide by 0.

Logic error

- The program is not accomplishing what it is intended.
- Possibly because the instructions are given in the wrong order or are incorrect to accomplish the task at hand.

Hello world!

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

Usually the first question a programmer wants to know when they learn a new language is:

HOW DO I WRITE THE “HELLO WORLD!” PROGRAM?

Reasons:
Started by Brian Kernighan and Dennis Ritchie in their book on C.

It is simple yet accomplishes a major task

It is almost law!

Decoding the Code

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

The first line tells the computer to include the library iostream.h

This library tells the computer how “cout” works. Without this line, we would get an error for the term “cout”.

Note: C++ is case sensitive!

Libraries

- Libraries are called "header files" and end in .h

- The standard libraries built for us contain useful code that is reused over and over again such as the code that defines cout.

- 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

- In general, every program starts with: #include <LIBRARY>
- Note that there are a couple of ways to include the iostream library. Both

`#include <iostream>` and `#include <iostream.h>` are legal

One more thing:

THERE IS NO SEMICOLON AT END OF `#include <iostream>`

using namespace std;

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

In large projects it may be advantageous to rename some commands. This may lead to naming conflicts. To avoid this problem programmers create their own name spaces.

Name space is essentially a dictionary that tells the compiler what their renamed commands are.

We have no need for non-standard name spaces.

Our programs will always use the standard name space.

Note this line ends in a semi-colon.

`main` function

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

The function `main()` is where the computer starts executing code.
A computer will execute code from top down starting with the first line of the main function. Always think about the order of your code.

The () indicate that no initial input is given to the function.

The “int” means that the end result of the function being run should be an integer.

More on the `main` function

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

Semi-colon!!

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

Notice how commands inside the main function end in semi-colon ;

With very few exceptions all commands end in semi-colons.

return statement

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

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

Remember the final result of main should be an integer. So the last statement of the main routine should send out a number. This number is given back to the program that called our program. In most cases that initial program was the operating system and the value that our program returns is just ignored.

When main returns 0 it means that no errors occurred.

Our main will always return 0.

Boiler plate

This is your boiler plate for all your programs in this class.

```
#include <iostream>
using namespace std;
int main()
{

/* Your code goes here */

return 0;
}
```

Your first few assignments will use this format.

Soon we will include more libraries than iostream.

cout

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello World!\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.

Remember we need to #include <iostream> to use cout.

The angle brackets << push whatever follows to screen. The output to the screen (if it is words) 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.

More about `cout`

When we want to ouput a string it needs to be in quotes

```
cout << “Hello world!”;
```

Spaces do not make a difference in code

```
cout <<                         “Hello World!”          ;
```

is the same as

```
cout << “Hello world!”;
```

But spaces inside quotes do count

```
cout << “Hello                    world!”;  // Outputs: Hello                world!
```

`cout` cont.

Consider the lines:

```
cout << “One”;
cout << “Two”;
cout << “Three”;
```

What is the output?

`cout` cont.

Output is:

OneTwoThree

Probably not what was intended.

We can fix the code by adding in some new lines:

```
cout << “One\n”;
cout << “Two\n”;
cout << ”Three”;
```

Of course this could be written in a single line:

```
cout << “One\nTwo\nThree”;
```

cout cont.

Other useful special characters:

\t Tab
\”   A quote
\\   A single slash \

We want to output to the console the following:

“Me fail English? That's unpossible.” -Ralph Wiggum

We cannot do:

cout << ““Me fail English? That's unpossible.” -Ralph Wiggum”;

Why?

`cout` cont.

Solution:

```
cout << “\“Me fail English? Thats unpossible.\” -Ralph
Wiggum”;
```

cout cont.

We can output number values without putting them in quotes.

```
cout << 15; // outputs 15 to screen
```

In C++ the computer always evaluates numeric expressions as soon as it encounters them.

```
cout  << 3*5; // outputs 15 to screen
```

We can chain outputs together

```
cout << “3*5 is ” << 3*5; // outputs 3*5 is 15
```

When chaining outputs it is important to pay attention to spaces.

`cout` cont.

What is the difference?

```
cout << “2”;
```

```
cout << 2;
```

Answer: Outputwise there is no difference. Both print 2 to the screen. However, “2” is a string and 2 is just a number.

We could do

```
cout << 2+2;
```

But

```
cout << “2” + 2;
```

does not work.

Neither does

```
cout << “2” + “2”;
```

Practice questions

What is the output?

```
cout << “What\n”<< “\tare you” << “up to?”;

cout << “\n\\nn”;

cout << “n\nn”;

cout << “What is ” << 2+2 << “?”;

cout << “What is ” << “2+2” <<”?”;

cout << “2+2” << “ is “ << 4;
```

More Practice

What’s wrong with the following?

```
cout << “\”Hello.”” ;

cout << “Hello.”\n;

cout << 1 << “is a lonely number.\n”;

cout << “I have” << 4+3 << “cookies”;
```