lab 1 notes:

compiler: is a program that allows you to "build" a program within it. a compiler does not do the work for you it just provides an area in which you can accomplish your task. much like a kitchen is a space for making food, and not a place that *makes* food (thats the chef/programmer's job).

source file: is the instruction list you write on. it is nothing more than a slightly fancy textedit file. this is where your main instructions/statement list will go.

.exe: this is what happens when you "compile and run" your .cpp (source) file. it is the same as having a working version of your program (the space where the user will interact with your code to accomplish the task you have programmed it to do.

Pre-processor directives: simply put "libraries with f'ns you can use to build and run your program" these are found at the top of your program just underneath the name/date/ abstract commented area at the very beginning of your code. without the much needed #include <iostream> you could never run a program that uses inputs or outputs... how useless!

commenting :

/* text */ is used to comment out anything contained within the star-slash. one of the best uses of this commenting type is to troubleshoot fragments or parts of code.

// text is used to comment out a single line of code. as soon as "enter" is hit or the text is put onto a new line, the text on a new line is no longer commented out. it iis then considered to be an argument/structure/statement,

console-out, the screen's way of communicating:

cout<< is the basic command structure used to output information. it has 3 primary ways it can be used.

1) cout<< " text "; anything within the " " will be printed out to the user on the computer screen (.exe) exactly as it is typed within the " " .

2) cout<< var; in this case var is a place holder for any given variable type that has been defined.

3) cout<< " text " << var << "text" << var << var ; in this case there is a stream of outputs under the same cout statement. the above could be accomplished in this same way: [cout<< "text" ; cout<< var; cout<< "text;] and so on.

console-in - the user's way of communicating with the computer:

cin>> is the basic command structure used to input variables to the program. It has 2 primary ways of being used.

1) cin>> var; this inputs information directly into the computer after "enter" has been pressed.

2) cin>> var1>>var2; this uses one cin statement to do the job of 2.

Variable declaration:

In order for something to be used, say a variable x, you must first create it. this means inside of your main program you need to make a variable named x. More importantly x needs to be made of a specific type of memory...

we have various kinds of variable types, for now lets just use a double.

ex:

```
double x;  // this makes a variable of memory space type double, called x.
```

Finally, the template in lab one will be used for the rest of the semester as a starting point to make all other programs. dont lose it.

Go over,

an example of basic math functions...

x= c+v;

x= 4+5;

x= c+5;

cout different forms of couts (text and variables)