Notes 1/5

Real programs don’t put everything into one huge .cpp file, they split it up. This is to minimize recompile time if you have to fix a bug in some area of the program, and makes the whole thing more organized and easier to follow.

.cpp files are called source files and they contain the instructions that run the program. When they compile they each produce an *object file* of suffix .obj in windows and .o in mac/linux. The object file is the machine language translation of the .cpp file.

To produce the final executable program, all of these files must be brought together so they work cohesively by the *linker.* The linker can also pull in any code required from various libraries such as *cmath* or *cstring*.

Remember, if you define a function in a separate file from where you call it, you need a function prototype to precede the call so that the compiler knows the function exists. You can’t just call a function without making some sort of declaration beforehand in the same file.

It would be irritating, however, to have to declare a function prototype in every single file you wanted to call it in. Instead we put all the function prototypes in a single *header* file so we don’t keep repeating ourselves (suffix is .h). Then in each function you can simply write “#include foo.h” at the top and you now have all the prototypes. *Don’t implement the functions in the header file!* Only use prototypes. If you implement the functions, every time you include the header you will be defining the function again, so if you do this more than once you will be breaking the rules of the language (can only define a function once.)

You can also include headers from standard libraries such as cmath or cstring. The standard syntax is to include your own headers in quotes, and include libraries in angle brackets <>.