**Pre-processor**

- prior to compiling or having assembler translate your object code to assembly( and eventually down to low level machine code), your IDE will import all headers specified to your program to define certain variables and functions used in the body of your "main".

**Linker**

- Links all files together

**Compiler**

- actually processes all the functions, math, arithmetic, programmed in the body of your main

**Assembler**

- translates all the information in the compiler in source code (C) to assembly code. Assembly code tells the program what to do in machine code.

Interface

* header(.h) files
  + do not include function definitions (encapsulation)
* functions, variables, structs, type definitions

Implementation

* .c file which defines all the files in the header

Interface + implementation linked to product object(.o) linked binary file

* These are sent to programmers (excluding implementation)

Precedence

* Most nested outward.
* Increment/decrement done before dereference. i.e. \*y ++ ; would do increment first, then dereference. Thus could create error if y is pointing to an address you are trying to increment.
  + (\*y)++

Increment Operators

* ++ i, incremented first, then returned
* i ++, returned, then incremented

**STYLE**

Comments

* What does this block of code do?
* Why did I implement this block in this way? (Design decision)
* Comment interesting blocks of code
* int\* pi not int \*pi

**TYPECASTING**

Data types

* int, long long, float, double
* char – 1 byte
* int, float – 4 bytes
* long long, double – 8 bytes

Typecast

* float x = 3.7
* (int) x 🡪 when displayed will give 3.

**VARIABLES**

1. Declaration
   1. int x;
   2. x = 42;
   3. cannot assign prior to declaring.
2. Naming Convention

* In C, num\_coins
* In C++, numCoins.