* IDE interprets my text to code (text editor)
* Compiler interprets my code to binary
  + Input/source code -> output/machine code
  + Not all languages use one
* Terminal window/Linux
  + Run commands on server having exclusive access by programmer = CLI (command line interface). Commands like run program!
  + Context: I am writing human code on a file called source-code. This file is only writable and readable. But the algorithms never run on that file, but on an executable.
    - Btw these commands are to be used in CS50’s VSC as I’d have to install WSL in Windows to run “make exec-file” on a .c file to then be able to ./exec-file.
    - make exec-file = “creates” an executable-code-file (a new version file of the binary code the machine understands)
      * note: save source-code after changes and again make exec-file if wanting to command ./source-code.
      * “make” is a program that finds and uses the compiler on the system to create the executable program file. If using source-code and compiler the code would be much longer.
      * Why don’t they let me save more under different names? Of course, different than xxx.c because it needs to be executable.
    - ./exec-code = go to current folder (dir) where code is saved and run the file exec\_code in the folder
    - code = open/”create” a file in a language in which I am going to code in. (e.g code calculator.c)
    - rm exec-code-file = remove file | answer: y
    - ls = list of files in GUI (the left box where non-techs manage files)
    - cd = go to folder specified | cd ../../.. is going back 3x
      * If nothing after, it will take me to original location
      * pwd for present working directory
    - cp = copy
      * cp <source> <destination>
      * cp -r <directory><dirdestination>
    - mkdir = make folder containing the current files
    - mv = move/rename <source><destination>
    - rmdir or rm -r <directory> or even rm -rf <directory>
    - chmod
    - ln
    - touch
    - man
    - diff
    - sudo
    - telnet
    - crtl+l or “clear” it will clean shit up | crtl+c abort operation
    - clicking pointer up will bring previous commands
    - clicking tab when writing a file’s name concludes it
* C Language
  + Return values and variables (later-output stored to be called dependent on human. Every time and input is needed multiple times, variable.)
    - String/int answer = get\_string(“How are you?\n“); means I am assigning value to answer from user from right to left.
      * Have to first #include <cs50.h> (for functions not originally with C) and #include <stdio.h> (printf and input-output stuff)
      * Have to int main(void)\n {\n ….. \n } as the “when greenflag clicks” function in Scratch (first project)
      * The .h is the header file, the name before are the libraries (menu of functions)
      * For answer printf(“Hello, %s”, answer); -> %... is a placeholder type before the variable
        + %i for integer, %c char, %f float/double, %li long int
    - Bool =
    - Char = single charac ter
    - Double = a lot of decimals
    - Float = real number with limited decimals
    - Long = bigger integer
    - Use get\_... each one above with the CS50 library
  + Operators
    - +
    - -
    - \*
    - /
    - %
    - Assign value by:
      * int variable = 0; means copying 0 to variable
      * variable = variable + 1; is possible because the = sign is not equality. It assigns variable+1 and updates value of variable on the left to the new computation of previous current variable +1 (var1 = var0 + 1;)
        + == is equality
        + But syntax dumb because var += 1 OR ++;

Can also --,//,\*\*

* + - * + Not needed to remind of int var 1 as I already assigned int data type to var0.
  + Structure code and practises:
    - if (x<y)\n {printf(“…”)\n }\n else if (x>y)\n { printf(…)\n else\n {printf(…)
      * Is better than adding an if to the last else, as every if is a question. So asking 2 questions and conditionaling the last is better than asking 3 separately.
    - When I problem solve:
      * 1st list the variables and think of their range
      * 2nd write structured and organized commentary mixed with functions or segments I know how to apply
    - A forever stupid loop: while (2==2)\n { … }
      * A forever boolean loop: while (true) || while (1)
      * For (start; exp; increment)
        + Execution, true or false (run code), and then increment. Otherwise it would keep incrementing without iterating body.