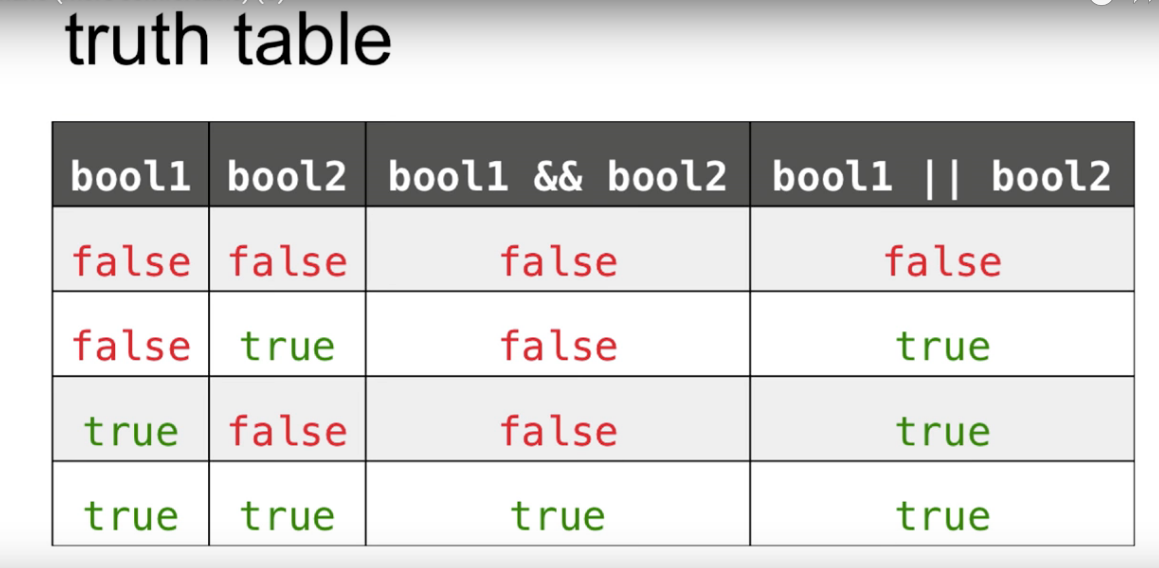
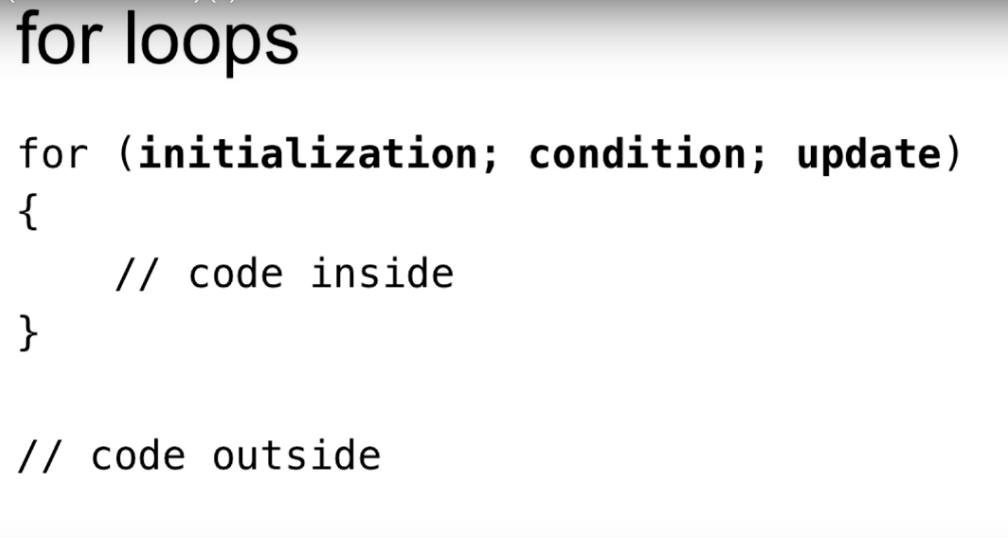
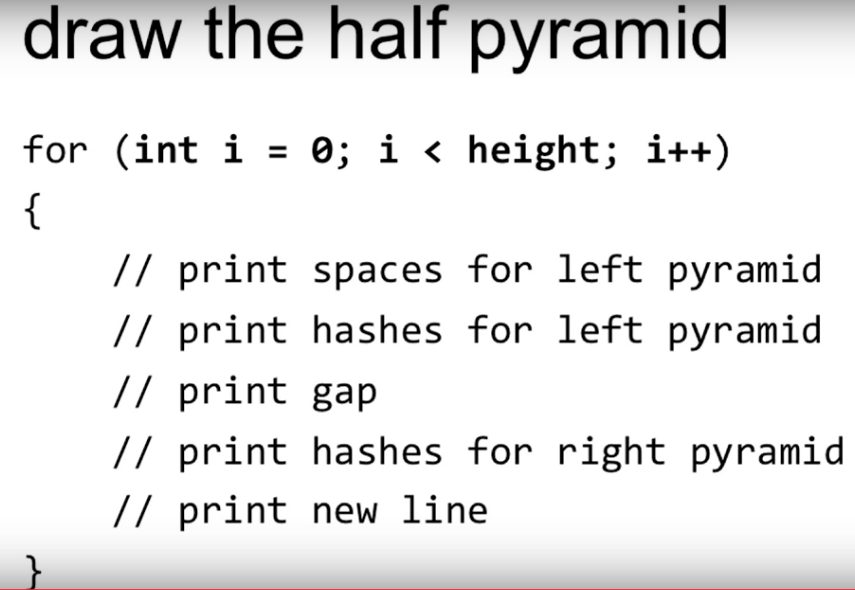
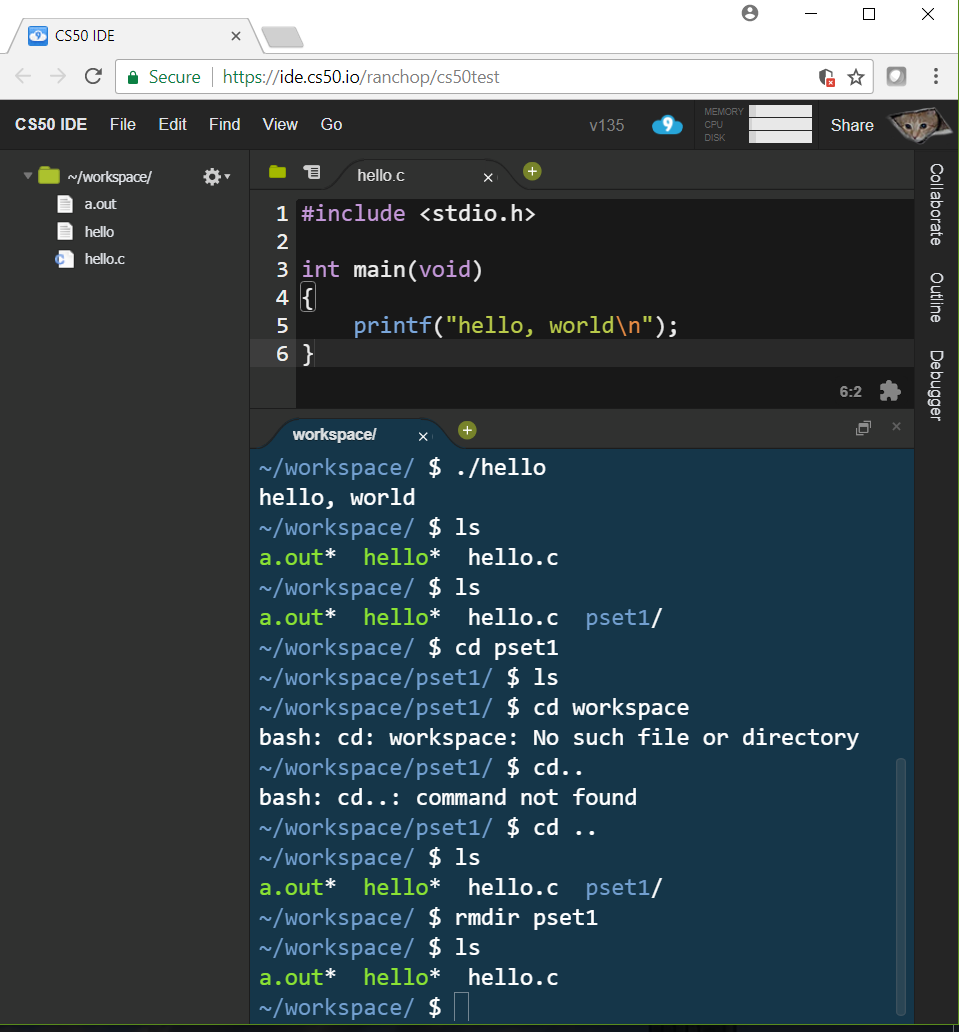
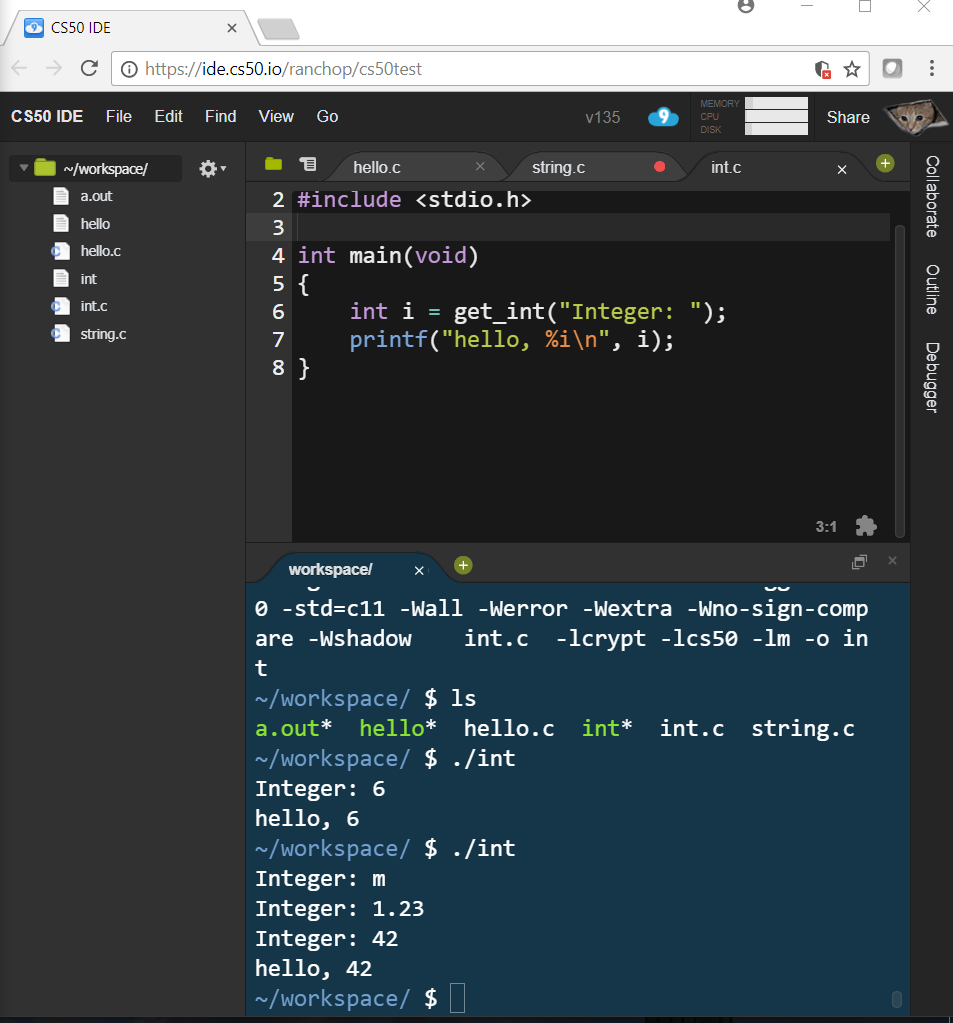
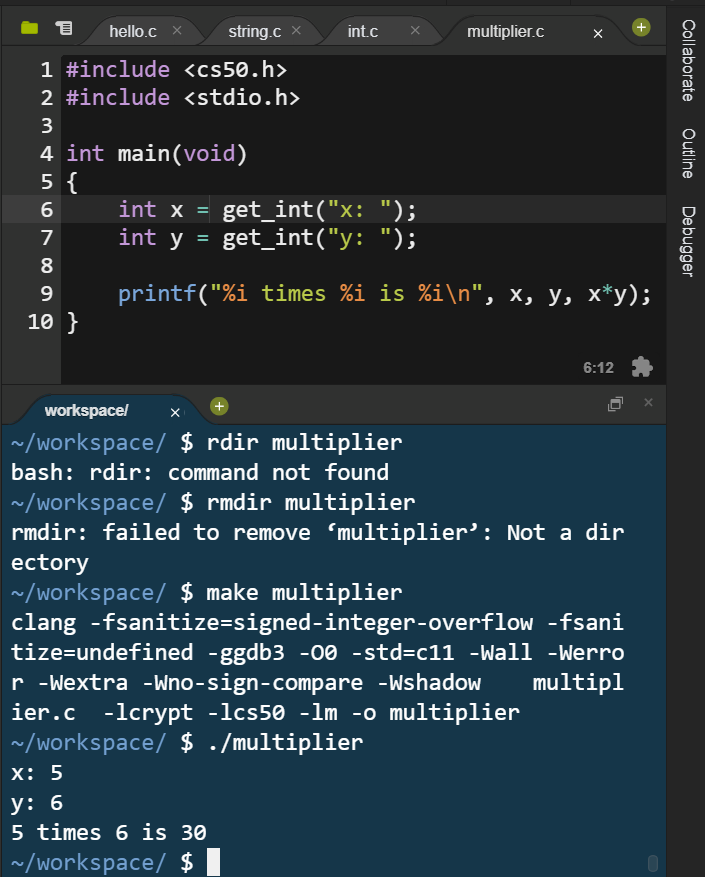
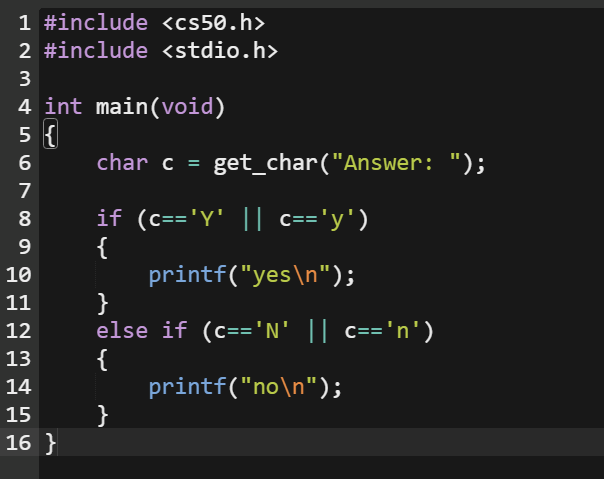
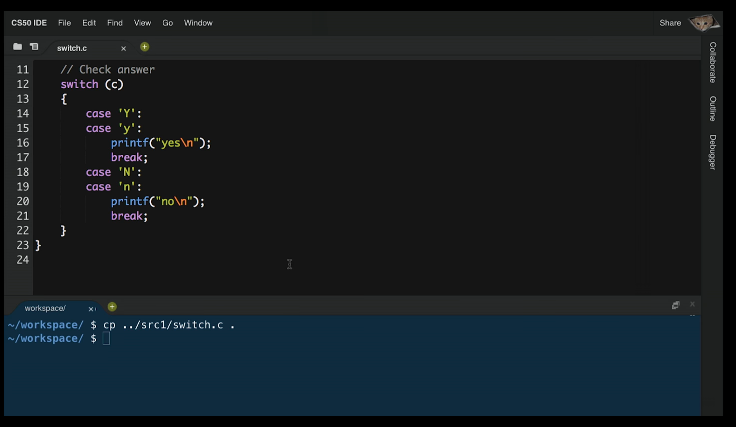
CS50 Week 1 Notes

* IDE = Integrated Development Environment
* A piece of program written in Python, Java or C would be **source code**. **Source code** because it is what is passed into a **compiler** or interpreter, **Source** as in**source** of instructions translated by a **compiler** to produce the object **code**.**Machine code** is the directly executable **binary** representation of a computer program.
* Offline IDE <https://manual.cs50.net/ide/offline>
* Online IDE <https://cs50.io/new?name=ide50&description=CS50%20IDE&workspaceType=cs50-2017-001&private=true&selection_file=x>
* 
* 
* TIP: use comments to write pseudo code FIRST;
* 

C

* 
* “./hello” does the same thing as “./a.out”
* “ls” lists files under current folder
* “cs ..” goes up a folder
* “rmdir” removes the filer you name after it form directory
* Green with a \*, eg “a.out\*” above, means an executable file
* Ctrl+l clears the screen in the terminal
* ↑ goes through your command history, so you don’t need to keep typing commands
* Ctrl+c is cancel, it stops the program in the window form running
* Write comments, comments, comments and notes!!!
* 
* 
* The “.h” means header file
* Float allows decimal values up to 32 bits
* Double allows the same up to 64 bits
* == equals (“=” already used for assignment)
* ‘Y’ single quotes for single characters
* “Yes” double quotes for multiple characters (strings)
* || means “or”
* Clang is the compiler command!
* ./a.out runs a program – it’s like double clicking

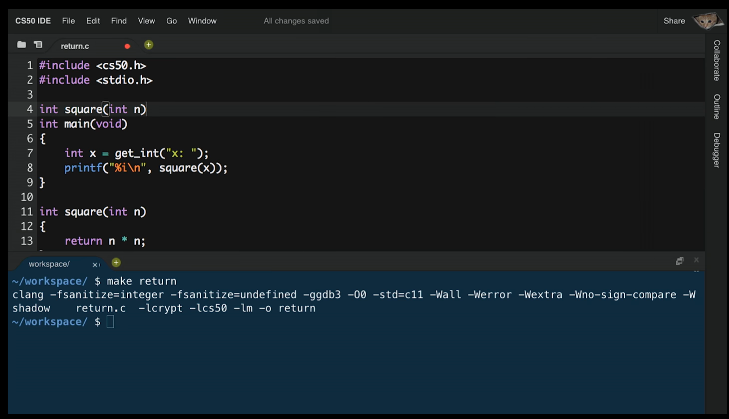




The programs above are an example of two different ways to create the same outputs from the same inputs.

The 2nd one uses the switch function

Return



In the program above, the function “square” is defined *after* the main

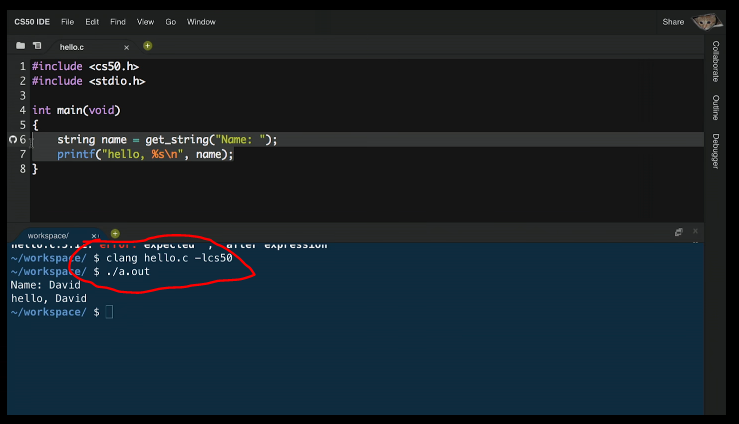
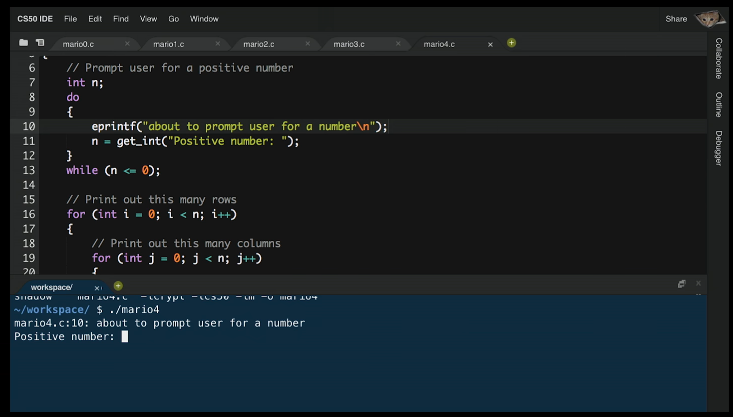
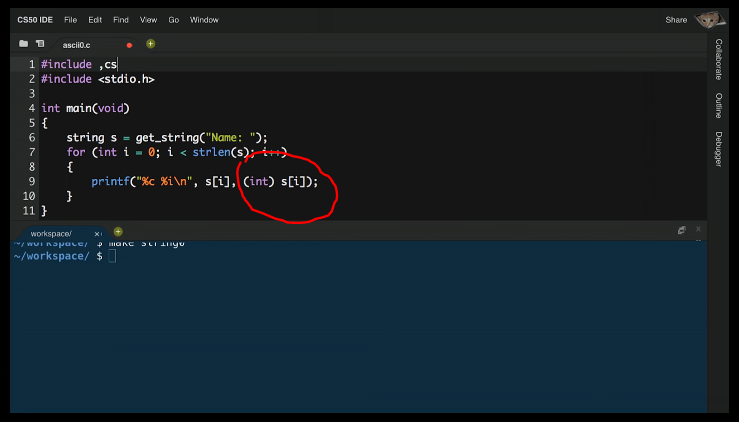
It is generally best practice to have main first

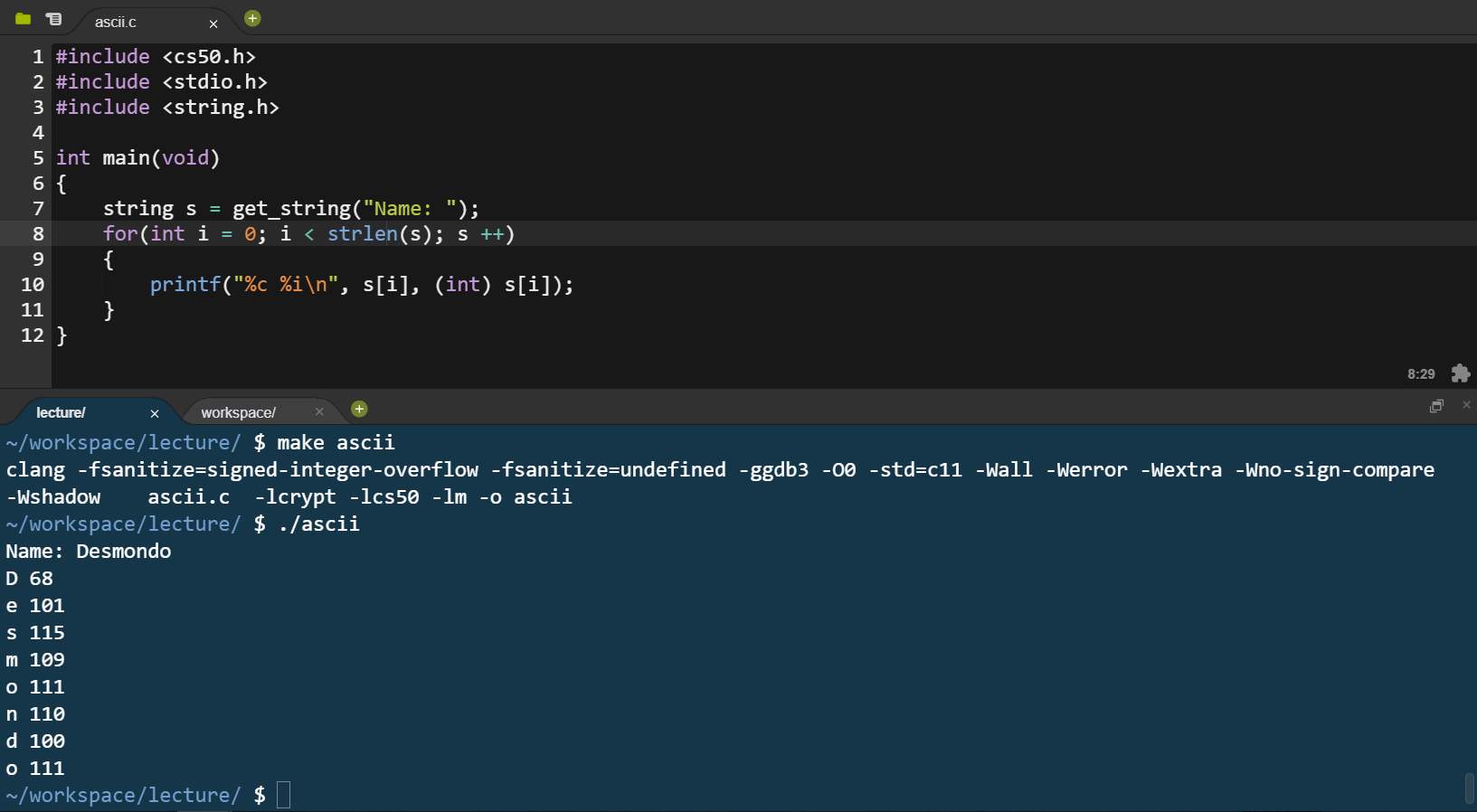
However, you will need to put the “prototype” (line 4) *before* main in order to use the function in main. This is like a teaser, or movie trailer, for the function

You basically declare it before, but define it after, main

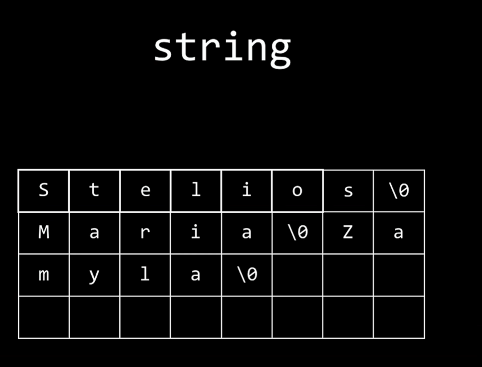
Integer overflow means “not enough bits” to store the number/value (think Ghandi in civilisation – underflow also exists!)

Week 1 Lecture 2 notes

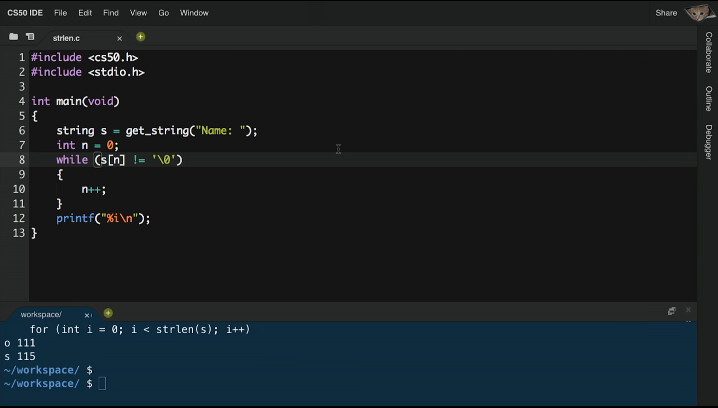
* Help50 command from <cs50.h> gives you hints when you encounter common errors
* 
* Clang hello.c –lcs50 links to the cs50 library. If you don’t do this you may get an error
* The “make” command does this, among other thing, for you
* Anything after “#” copies and pastes the code from the file, e.g. stdio.h, into your program. The function printf, for example, is part of this code. This is called PREPROCESSING
* Check50 and style50 help you by seeing if your file exist/compiles/works and looks nice, eg has enough notes, respectively
* Can’t stress enough – take steps/chunks of code, then run/test etc, then continue!!!
* 
* Eprintf() is kind of like a comment, but it explicitly prints a note while running the program, and tells you what line of the code is executing
* <https://manual.cs50.net/style>
* STYLE GUIDE!
* Always, always debug
  + “Debug50 ./Mario”
  + Allows you to see the program running line by line @ “human” speed
* Check the manual!
  + “man get\_int” will show you the manual page for this function
* If the manual page is confusing, go here instead;
  + <https://reference.cs50.net/>
* Typecasting:
  + 
  + This says print the “ith” character of the string s, but typecast it to be an int instead of a character. (int) typecasts it



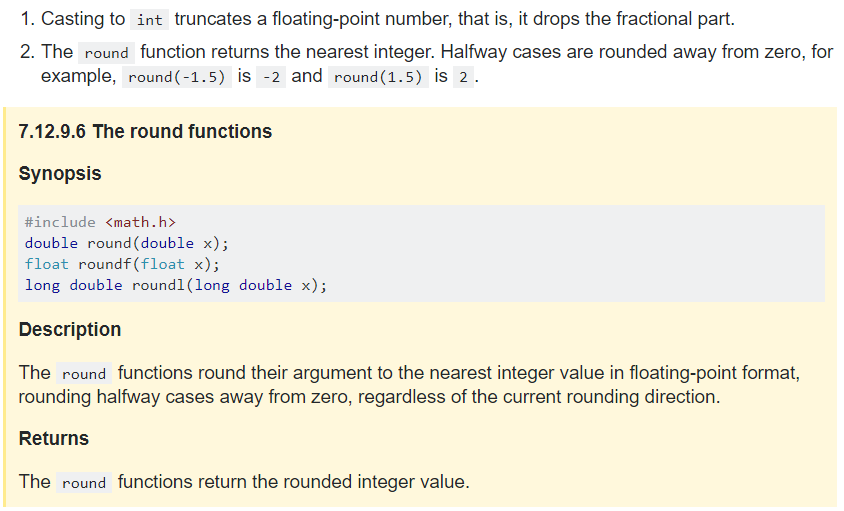
A string is really a series of individual characters. C automatically puts ‘\0’ as the character at the end of each string;



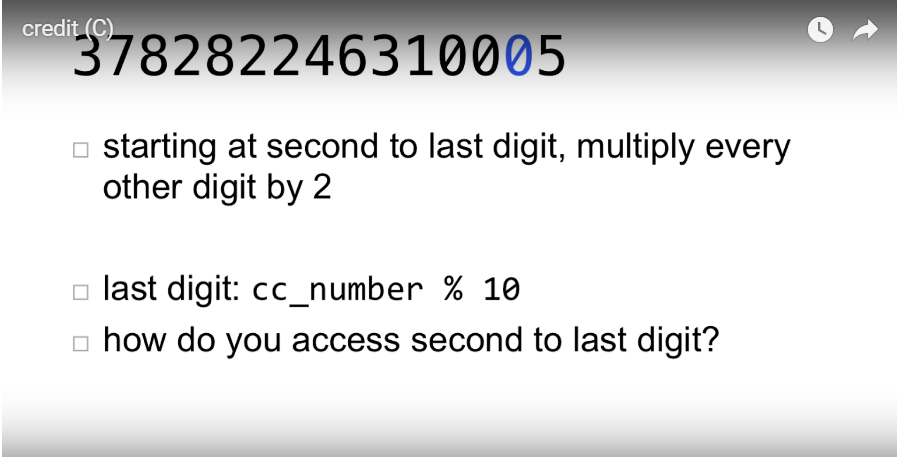
…..which means the program below counts how long a string is;

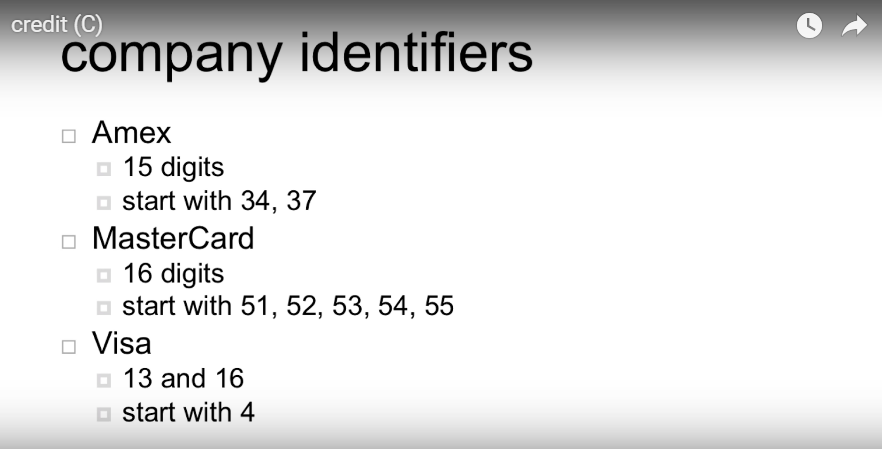


* “Segmentation fault” means you have touched memory/RAM that you should not have
* An array is a bunch of “things” back to back, eg
  + Int int int int……
  + String string string…..



# PSET 1 CREDIT





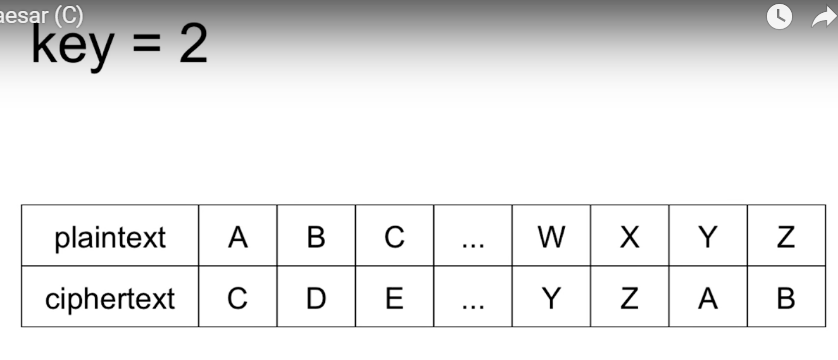
# Caesar

To Do

* Get the key
* Get the plaintext
* Encipher
* Print ciphertext

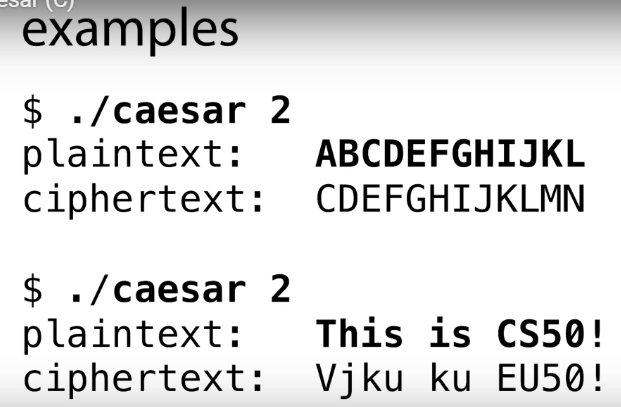
Example;

* Note that ‘Y’ wraps around to ‘A’

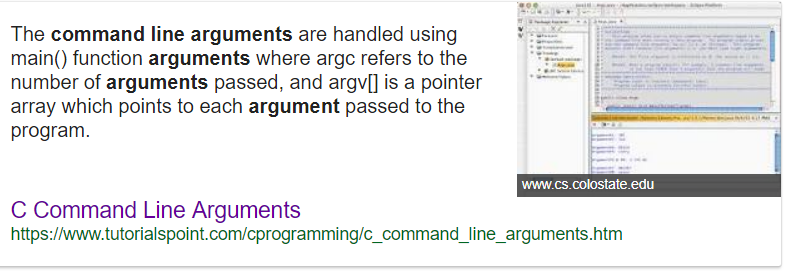


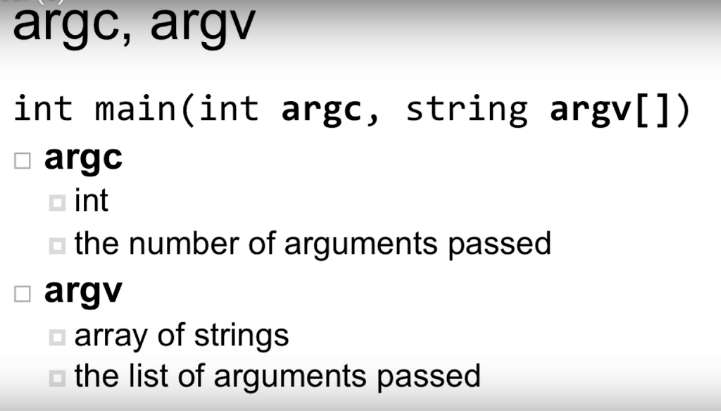
Example;

* Note upper/lower case are preserved
* Other characters/punctuation kept “as-is”



This time we’re not using “get\_int”, but this timewe are using a command line argument and a new function called atoi



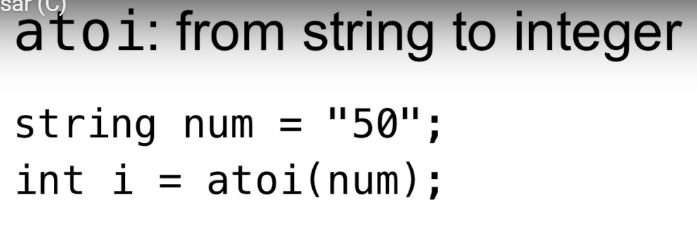


You don’t have to explicitly declare these variables (argc, argv) because they are computed for you by the compiler

Atoi

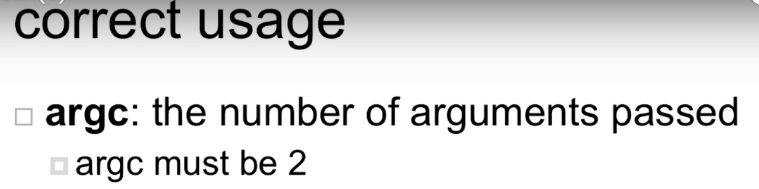
The string num contains a “number”, but the datathe type is a string

Atoi converts this to data type integer



Argc must be 2 for Caesar;

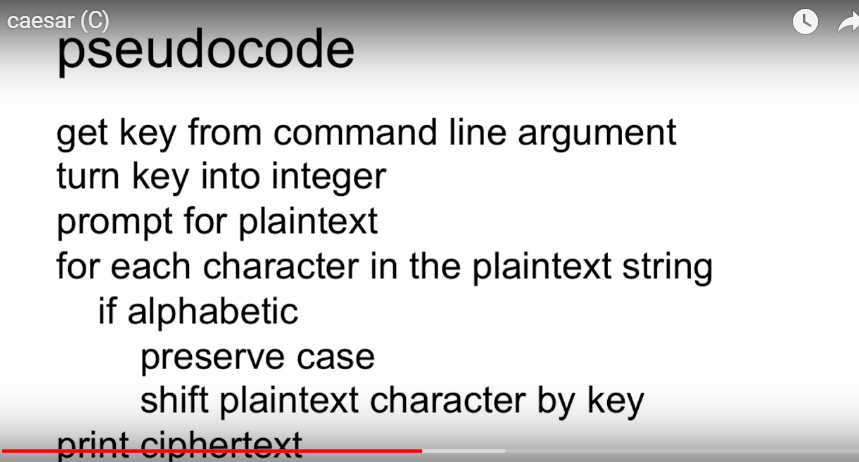
1. ./caesar
2. k

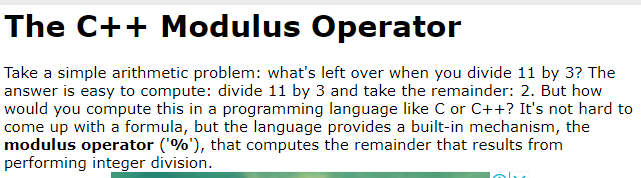


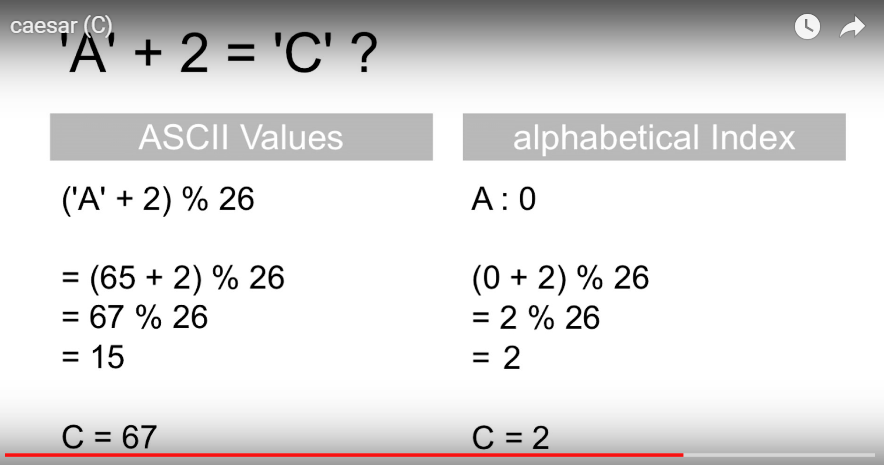


Example above

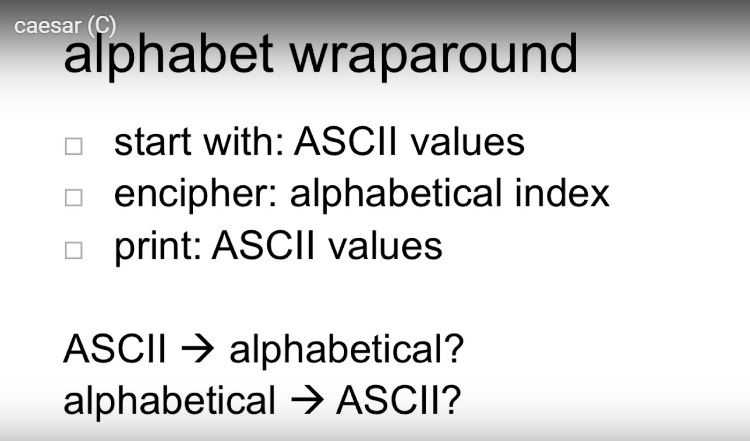
Caesar pseudocode example:







This shows that we must use the alphabetical index rather that ASCII values



## [**Reminders**](https://docs.cs50.net/2018/x/psets/2/pset2.html#reminders)

Per [Week 2’s lecture](https://video.cs50.net/2017/fall/lectures/2):

* Use help50 as needed.
* Use eprintf as needed.
* Use debug50 as needed.

VIGENERE

