

Introductions, C Programming

Computer Operating Systems, Spring 2024

Instructors: Joel Ramirez Travis McGaha

Head TAs: Adam Gorka Daniel Gearhardt
Ash Fujiyama Emily Shen

TAs:

Ahmed Abdellah	Ethan Weisberg	Maya Huizar
Angie Cao	Garrett O'Malley Kirsch	Meghana Vasireddy
August Fu	Hassan Rizwan	Perrie Quek
Caroline Begg	Iain Li	Sidharth Roy
Cathy Cao	Jerry Wang	Sydnie-Shea Cohen
Claire Lu	Juan Lopez	Vivi Li
Eric Sungwon Lee	Keith Mathe	Yousef AlRabiah



pollev.com/tqm

❖ How are you?

Lecture Outline

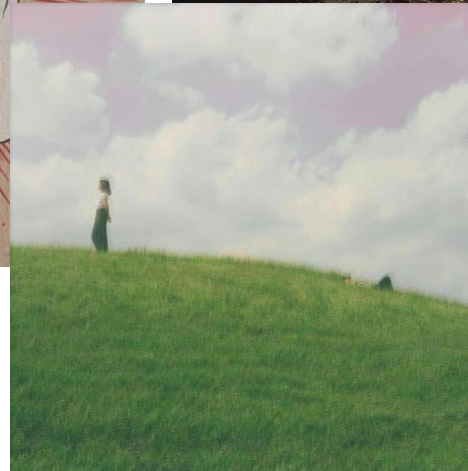
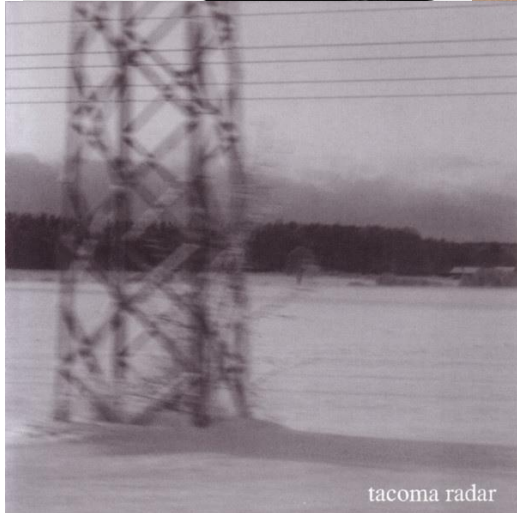
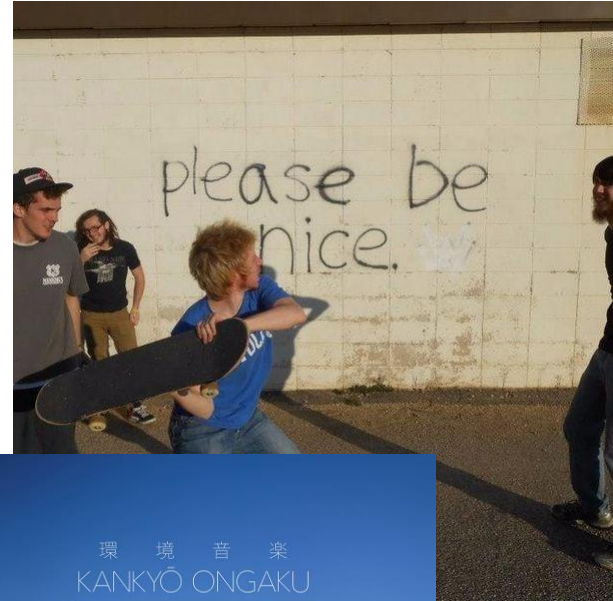
- ❖ **Introduction & Logistics**
 - **Instructor Introduction(s)**
 - Course Overview
 - Assignments & Exams
 - Policies
- ❖ C Intro
 - Functions
 - Declaration Before use
 - Types, Variables, uninitialized
 - “String”, int, float, double
 - “import” and printing

Instructor: Travis McGaha

- ❖ UPenn CIS faculty member since August 2021
 - Currently my seventh semester at UPenn
 - Third Semester with CIS 2400... and I am still trying new stuff
 - Lots of the same content, but in a different order, new assignments and more of a focus on C Programming
- ❖ Education: University of Washington, Seattle
 - Masters in Computer Science in March 2021
 - Bachelors in Computer Engineering in June 2019
 - Instructed a course that covers very similar material

Instructor: Travis McGaha

❖ I like most music



Instructor: Travis McGaha

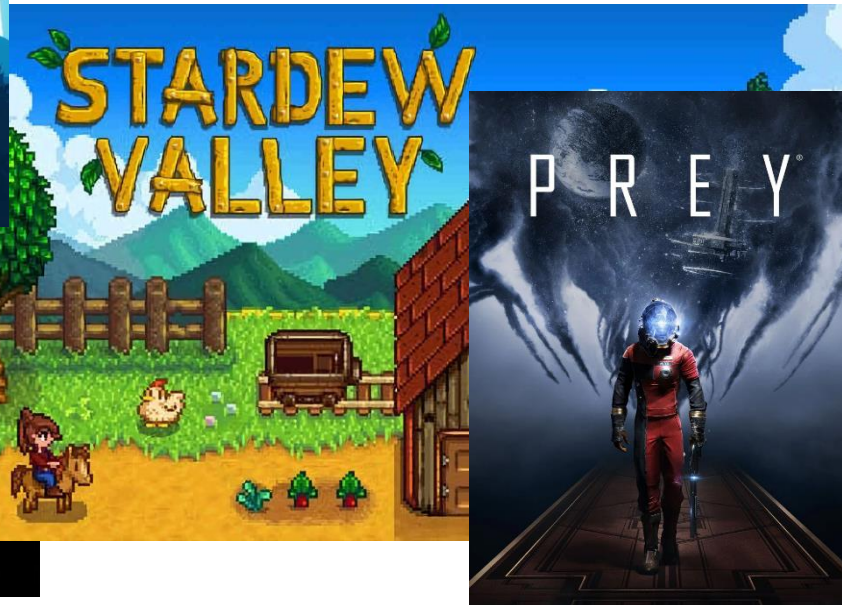
- ❖ I like animals and going outside (especially birds, cats and mountains)



Instructor: Travis McGaha



❖ I like video games



Instructor: Travis McGaha

- ❖ I have a general dislike of food
(Breakfast is pretty good tho)



Instructor: Joel

- ❖ UPenn CIS faculty member since this week
- ❖ Previously Lecturer @ Stanford
 - Where I taught computer systems and probability fundamentals
 - Had a whole lot of fun doing it
- ❖ Education: Stanford University @ Stanford?
 - Masters in Computer Science in June 2023
 - Bachelors in Symbolic Systems in June 2021

Instructor: Joel

❖ I love the outdoors....



Instructor: Joel

❖ I play Mexican folk music.....



Instructor: Joel

❖ I love to cook...



Instructor: Joel

❖ I have two awesome cats (sometimes)

- Ube Donut



- Miso Soup



Instructors: Both

- ❖ We care a lot about your actual learning and that you have a good experience with the course
- ❖ We are human beings, and we know that you are one too. If you are facing difficulties, please let us know and we can try and work something out.
- ❖ More on my personal website:
<https://www.cis.upenn.edu/~tqmcgaha/>
- ❖ Joel's Website: TBD

Lecture Outline

- ❖ **Introduction & Logistics**
 - Instructor Introduction(s)
 - **Course Overview**
 - **Assignments & Exams**
 - **Policies**
- ❖ C Intro

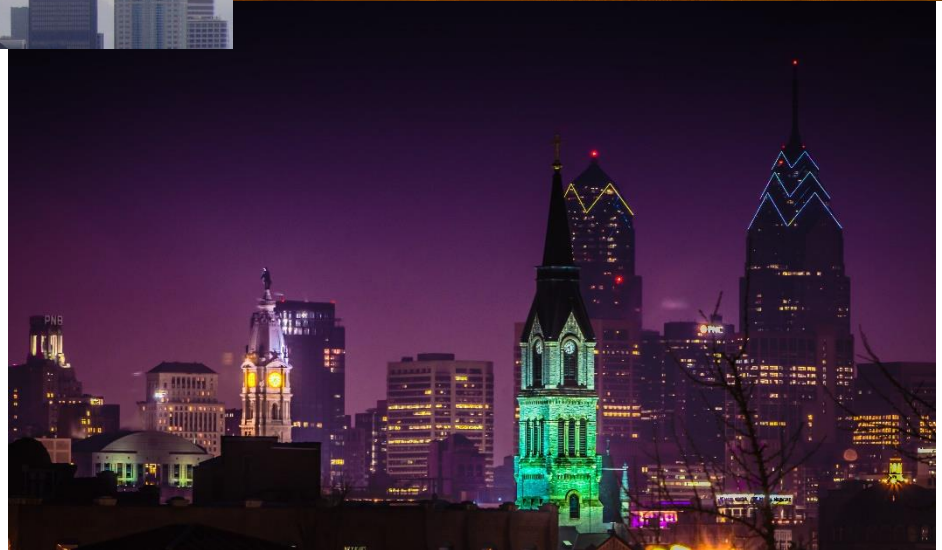
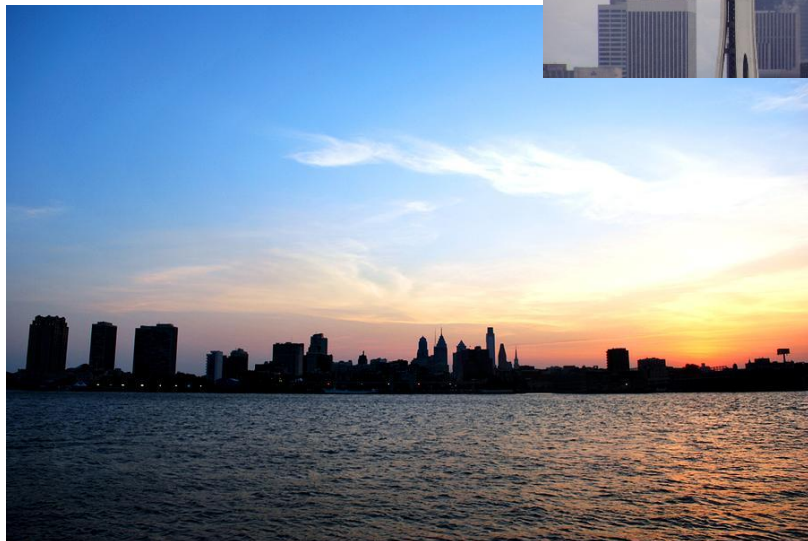
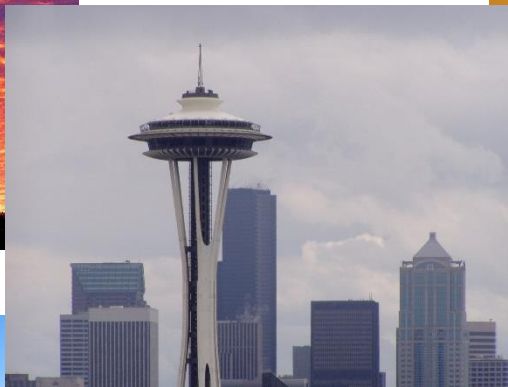


Poll Everywhere

pollev.com/tqm

- ❖ What color is the sky?
 - This is a “trick” question
 - Talk to your neighbor

Are these all pictures of the sky?



Is this true?



daisyowl

@daisyowl



if you ever code something that
"feels like a hack but it works," just
remember that a CPU is literally a rock
that we tricked into thinking

5:03 p.m. · 14 Mar 17

3,196 RETWEETS **5,051** LIKES



daisyowl @daisyowl · 17h



@daisyowl not to oversimplify: first
you have to flatten the rock and put
lightning inside it

← 4

↻ 270

♥ 592



Sorta.....
But a lot is missing

“Lies-to-children”

- ❖ "The necessarily simplified stories we tell children and students as a foundation for understanding so that eventually they can discover that they are not, in fact, true."
 - Andrew Sawyer (Narrativium and Lies-to-Children: 'Palatable Instruction in 'The Science of Discworld'')

- ❖ "A lie-to-children is a statement that is false, but which nevertheless leads the child's mind towards a more accurate explanation, one that the child will only be able to appreciate if it has been primed with the lie"
 - Terry Pratchett, Ian Stewart & Jack Cohen (The Science of Discworld)

Wittgenstein's Ladder

- ❖ "My propositions serve as elucidations in the following way: anyone who understands me eventually recognizes them as nonsensical, when he has used them—as steps—to climb beyond them. (He must, so to speak, throw away the ladder after he has climbed up it.)

He must transcend these propositions, and then he will see the world aright."

- Ludwig Wittgenstein (Tractatus Logico-Philosophicus)

We're going to lie to you (but good)

- ❖ "All models are wrong, but some are useful."
 - Same source as below.
- ❖ "If it were necessary for us to understand how every component of our daily lives works in order to function - we simply would not."
 - AnRel (UNHINGED: A Guide to Revolution for Nerds & Skeptics)
- ❖ This course will reveal a lot of high-level details about how a computer works, but there is still a ton I am leaving out. Even what I say that is accurate, will likely change in the future.

I'M ALREADY LYING TO YOU

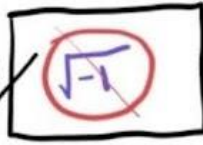
- ❖ This idea of a “ladder” and how one just goes up it is a lie. Education is often not linear and often is a tangled web of ideas.
- ❖ But it is a good metaphor :)
- ❖ I think there is also a good discussion of whether these count as “lies”. Is using the word “lie” a lie?

This goes beyond this course



thewest-isdead S'abonner

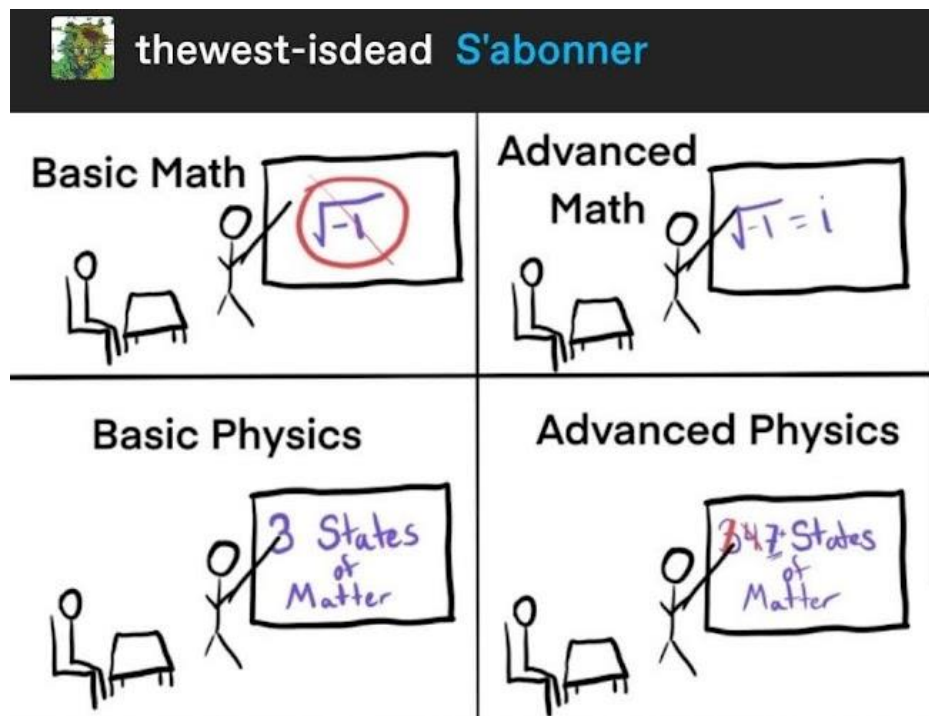
Basic Math



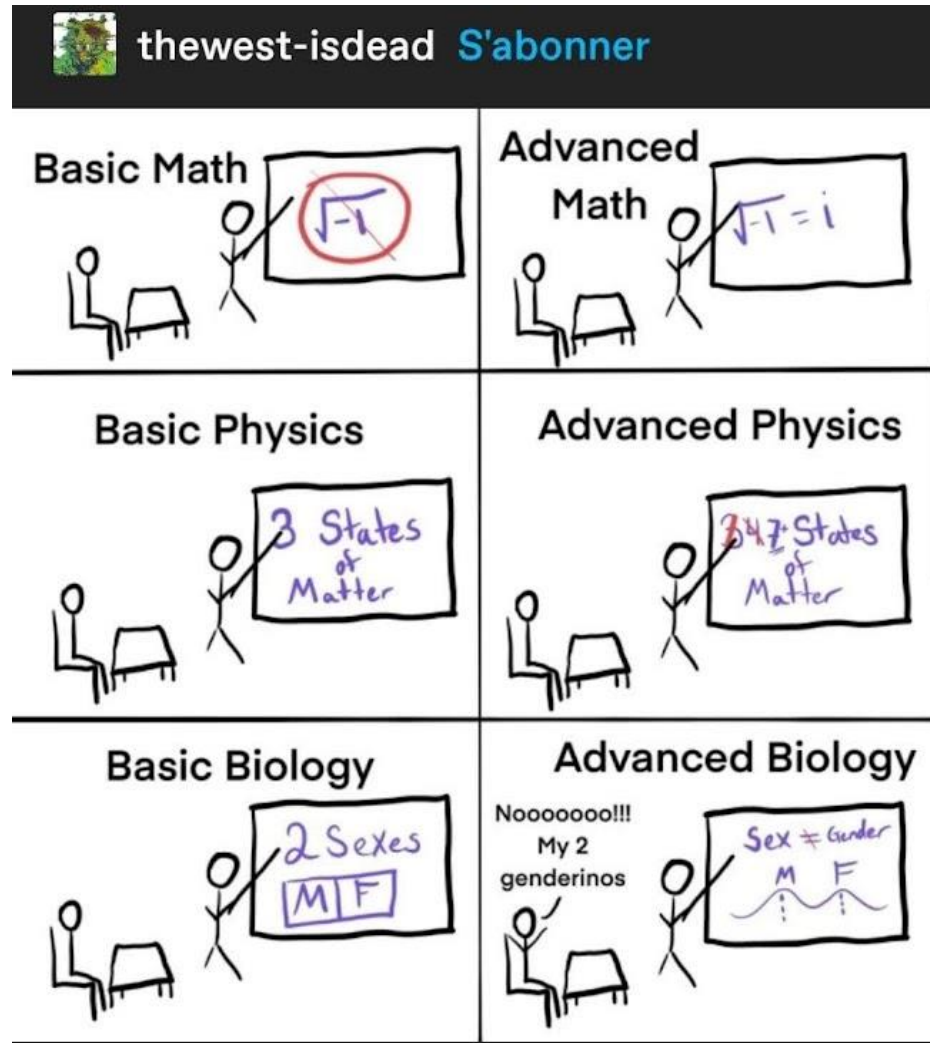
Advanced
Math



This goes beyond this course



This goes beyond this course



This goes beyond this course



xceansicemane S'abonner

I like this version of this comic.

states of matter entering slippery slope

V · T · E	States of matter (list)	
State	Solid · Liquid · Gas / Vapor	Plasma
Low energy	Bose-Einstein condensate · Fermionic condensate · Degenerate matter · Quantum Hall · Rydberg matter · Rydberg polaron · Strange matter · Superfluid · Supersolid · Photonic molecule	
High energy	QCD matter · Lattice QCD · Quark-gluon plasma · Color-glass condensate · Supercritical fluid	
Other states	Colloid · Glass · Crystal · Liquid crystal · Time crystal · Quantum spin liquid · Exotic matter · Programmable matter · Dark matter · Antimatter · Magnetically ordered (Antiferromagnet · Ferrimagnet · Ferromagnet) · String-net liquid · Superglass	

↑
mental illnesses

37 619 notes

↪ 💬 ↺ ❤️

Course Overview

- ❖ You should be familiar with how to program from taking the prerequisite course: CIS 1200

- ❖ Topics of this course:
 - What does lower-level programming look like? How is it different than Higher-Level Programming languages?
 - How do computers actually work as physical machines?
 - What is "Memory" in computer programming?
 - What is the hardware/software "interface"?

Prerequisites

❖ Course Prerequisites:

- CIS 1200
- Willingness/happy to spend substantial time coding

❖ What you should be familiar with already:

- How to program on your own
- Keeping track of state
 - A lot of higher order things in Java / Ocaml do not exist in C. You must keep track of a lot of things yourself
- Boolean logic
- A little bit of discrete math, and finite machines would be nice

CIS 2400 Learning Objectives

- ❖ To leave the course with a better understanding of:
 - How a computer runs works as a physical machine
 - How the previous point may affect the code we write
 - Experience writing some programming projects FROM SCRATCH
 - Experience writing C code and understanding of how it is different from other languages
 - What is "Memory" in computer programming?

- ❖ Topics list/schedule can be found on course website
 - Note: These topics may be tweaked

Disclaimer

- ❖ This is a digest, **READ THE SYLLABUS**
 - <https://www.seas.upenn.edu/~cis2400/current/documents/syllabus>

Course Components pt. 1

- ❖ Lectures (~26)
 - Introduces concepts, slides & recordings available on canvas
 - In lecture activities.

- ❖ Recitations (~10)
 - Reiterates lecture content, lecture clarifications, assignment & exam preparation. Optional, details TBD

- ❖ Check-ins “Quizzes” (~10)
 - Unlimited attempt low-stake quizzes on canvas to make sure you are caught up with material
 - Lowest two are dropped
 - No extensions granted except for serious cases

Course Components pt. 2

- ❖ Homework Assignments (~11)
 - Due every week
 - Most are programming
 - Very flexible on-request late policy

- ❖ Exams (2)
 - Two in-person exams, two pages of notes allowed
 - One midterm shortly after fall break
 - One final exam during final exam period
 - Clobber policy

- ❖ Textbook (0)
 - No official textbook, but some suggested on course site

Course Components pt. 3

- ❖ Office Hours (Starting Soon*)
 - Levine 501 (Tables on 5th floor of Levine hall near elevators)
 - Every day of the week in evenings (exact time TBD)

- ❖ 1-on-1's
 - We will maintain a way for students to request individualized help
 - Can choose who you get help from
 - These are not a replacement for office hours, but can help with similar problems if necessary

- ❖ Course-wide participation
 - Credit for lecture participation, recitations, Ed, Office hours, etc.
 - Very low bar to get full credit, almost everyone does.

Course Policies

❖ HW Late Policy

- Late days given on request (Request usually granted)
- No cap on the number of late days per assignment
- More than 3 on an assignment requires approval from Travis & Joel
- Written assignments get at max 3 late days
- End of the semester is the end
(unless there is particularly special circumstances)

❖ Midterm Clobber Policy

- Final is cumulative
- If you do better on the “midterm section” of the final, your midterm grade can be overwritten.

Collaboration Policy Violation

- ❖ You will be caught:
 - Careful grading of all written homeworks by teaching staff
 - Measure of Software Similarity (MOSS):
<http://theory.stanford.edu/~aiken/moss/>
 - Successfully used in several classes at Penn

- ❖ Zero on the assignment, F grade if caught twice.
 - First-time offenders will be reported to Office of Student Conduct with no exceptions. Possible suspension from school
 - Your friend from last semester who gave the code will have their grade retrospectively downgraded.

Collaboration Policy Violation

❖ Generative AI

- I am skeptical of its usefulness for your learning and for your success in the course
- Some articles on the topic:
 - <https://www.aisnakeoil.com/p/chatgpt-is-a-bullshit-generator-but>
 - <https://www.aisnakeoil.com/p/gpt-4-and-professional-benchmarks>
- Not banned, but not recommended. Use your best judgement.

❖ You will not help your overall grade and happiness:

- If you can't explain your code in OH, we can turn you away.
 - This is different than being confused on a bug or with C, this is ok
- This stuff is foundational for ALL programmers. Why are you spending 4 years and \$70,000 on this degree if you don't like it?
- We give a lot of help in the class, come get help or an extension if you need it.

Course Grading (Tentative)

❖ Breakdown:

- Course-Wide Participation (3%)
- Check-in Quizzes (5%)
- Homeworks (62%)
- Exams (30%)
 - Midterm 15%
 - Final 15%

❖ Final Grade Calculations:

- I would LOVE to give everyone an A+ if it is earned
- Final grade cut-offs will be decided privately at the end of the Semester.

Course Infrastructure

- ❖ Course Website: www.seas.upenn.edu/~cis2400/current/
 - Materials, Schedule, Syllabus ...
- ❖ Docker or Speclab
 - Coding environment for hw's
- ❖ Gradescope
 - Used for HW Submissions
- ❖ Poll Everywhere
 - Used for lecture polls
- ❖ Ed Discussion
 - Course discussion board

Programming Facilities

- ❖ Docker
 - Same environment as the autograder
 - Instructions for setup to be posted soon
- ❖ Speclab cluster, as a fallback incase Docker does not work
 - Instructions on course website
 - To see status:
<https://www.seas.upenn.edu/checklab/?lab=speclab>
- ❖ **DO NOT use Eniac machines to develop projects for this class!**

Getting Help

- ❖ Ed
 - Announcements will be made through here
 - Ask and answer questions
 - Sign up if you haven't already!

- ❖ Office Hours:
 - Can be found on calendar on front page of course website
 - Starts next week for all TAs

- ❖ 1-on-1's:
 - Can schedule 1-on-1's with Travis
 - Should attend OH and use Ed when possible, but this is an option for when OH and Ed can't meet your needs

We Care

- ❖ We are still figuring things out, but we do care about you and your experience with the course
 - Please reach out to course staff if something comes up and you need help

- ❖ **PLEASE DO NOT CHEAT OR VIOLATE ACADEMIC INTEGRITY**
 - We know that things can be tough, but please reach out if you feel tempted. We want to help
 - Read more on academic integrity in the syllabus



Poll Everywhere

pollev.com/tqm

- ❖ Ask Us Anything!
- ❖ Any questions, comments or concerns so far?

Lecture Outline

- ❖ Introduction & Logistics
 - Instructor Introduction(s)
 - Course Overview
 - Assignments & Exams
 - Policies
- ❖ **C Intro**
 - **Basics & printf**
 - **Forward declaration**
 - **Compiling (simple)**

C: main(), #include, Hello World!

- ❖ This program just prints: "Hello World!"
 - Let's break it down

hello.c

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[]) {
    printf("Hello World!");
    return EXIT_SUCCESS;
}
```

C: main(), #include, Hello World!

- ❖ This program just prints: "Hello World!"
 - Let's break it down

hello.c

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[]) {
    printf("Hello World!");
    return EXIT_SUCCESS;
}
```

← "import" stdio, which includes printf
← "import" stdlib, which includes EXIT_SUCCESS

C: main(), #include, Hello World!

- ❖ This program just prints: "Hello World!"
 - Let's break it down

hello.c

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[]) {
    printf("Hello World!");
    return EXIT_SUCCESS;
}
```

← "import" stdio, which includes printf
 ← "import" stdlib, which includes EXIT_SUCCESS

Defines the main() function

- main() is where every C program starts running.
- main() has return type "int" and returns a number to indicate whether the program ran successfully or what error was encountered
 - return EXIT_SUCCESS on success, or EXIT_FAILURE on failure
- main() either takes no args or command line args (more on this later)

Demo: Downloading & Running (Terminal)

❖ To download code, do one of:

- `curl -o hello.c https://www.seas.upenn.edu/~cis2400/current/code/hello.c`
- `wget https://www.seas.upenn.edu/~cis2400/current/code/hello.c`
- Download manually from site

❖ To compile:

- `clang-15 hello.c`

❖ To run:

- `./a.out`

Some fundamental C types

❖ Int

- Integer types

❖ Double

- Floating point numbers

❖ String

- No type called “string” instead it is char*
- Cover more on strings in the future, for now char* = string

❖ Psssst variations on float and int

❖ Psssssst others exist, but just these for now :)

```
int x = 10;  
double a = 10.1;  
char *name = "miso";
```

C: `printf` format specifiers

- ❖ To specify how arguments to *printf* should be interpreted for printing we must use a format string/format specifiers
- ❖ A format string is just a string with formatting specifiers:
 - `%d` – a decimal integer value
 - `%x` – a hexadecimal value
 - `%s` – a string
 - `%f` – a floating point value

```
printf("Hello there, %s.\n", "miso");  
// "Hello there, miso."
```

```
printf("Imma take a %d hour nap after CIS2400.\n", 10);  
// "Imma take a 10 hour nap after CIS2400."
```

C: Function declarations

- ❖ How to define them?

```
type name (type1 arg1, ...) {  
    return [something of type type]  
}
```

- ❖ You can define them anywhere in your program
- ❖ e.g. a function that converts Fahrenheit to C.
 - (no pun intended)

```
float fahrenheitToCelsius (float f) {  
    return (f - 32) * 5 / 9;  
}
```

C: Function declarations

❖ Let's see this example!

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[]) {
    char* string = get_silly();
    printf("I'm feeling %s\n", string);
    return EXIT_SUCCESS;
}

char* get_silly() {
    return "silly as a goose.\n";
}
```


C: Function declarations

❖ So why didn't this work?

```
int main(int argc, char* argv[]) {
    char* string = get_silly();
    printf("I'm feeling %s\n", string);
    return EXIT_SUCCESS;
}
char* get_silly() {
    return "silly as a goose.\n"
}
```

← here

- ❖ Functions and variables are defined from “**top to bottom**”.
 - ❖ This is due to how the code is compiled.
 - ❖ We will learn more about this as the semester continues...
- ❖ `get_silly()` is not defined when we use it *here*.

C: Function declarations

❖ Forward Declarations

```

char* get_silly(); ←
int main(int argc, char* argv[]) {
    char* string = get_silly();
    printf("I'm feeling %s\n", string);
    return EXIT_SUCCESS;
}
char* get_silly() { ←
    return "silly as a goose.\n"
}
    
```

Declaration to specify return type, name, and parameters.

Definition where you provide the actual code that performs the function's task.

- ❖ A promise to define it 'later' so that we don't get errors thrown at us by the compiler.

Poll Everywhere

pollev.com/tqm

- ❖ Does this C code compile?
 - The format specifiers (e.g. "%d\n") are fine

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[]) {
    int x = 5;
    printf("%d\n", x);
    char* string = get_string();
    printf("%s\n", string);
    return EXIT_SUCCESS;
}

char* get_string() {
    return "Hello, World!";
}
```

Demo: hello_print.c

Arrays

- ❖ Definition: `type name [size]`
 - Allocates `size` x `sizeof(type)` bytes of *contiguous* memory
 - Normal usage is a compile-time constant for `size` (e.g. `int scores[5];`)
 - **Initially, array values are “garbage”**

address	0x2000	0x2001	0x2002	0x2003	0x2004
value	10	9	9	9	10

- ❖ Size of an array
 - **Not stored anywhere** – array does not know its own size!
 - The programmer will have to store the length in another variable or hard-code it in

Using Arrays


Optional when initializing

❖ Initialization: `type name [size] = {val0, ..., valN};`

- `{ }` initialization can *only* be used at time of definition
- If no `size` supplied, infers from length of array initializer

❖ Array name used as identifier for “collection of data”

- `name [index]` specifies an element of the array and can be used as an assignment target or as a value in an expression

❖  Array name (by itself) produces the address of the start of the array

- The array name cannot be assigned to / changed

```
int primes[6] = {2, 3, 5, 6, 11, 13};
primes[3] = 7;
primes[100] = 0; // memory smash!
```

No IndexOutOfBounds
Hope for segfault

Arrays as Parameters

❖ It's tricky to use arrays as parameters

- What happens when you use an array name as an argument?
- Arrays do not know their own size

Passes in address of start of array

```
int sumAll(int a[]) {  
    int i, sum = 0;  
    for (i = 0; i < ...???)  
}
```

```
int sumAll(int* a) {  
    int i, sum = 0;  
    for (i = 0; i < ...???)  
}
```

Equivalent

❖ Note: Array syntax works on pointers using pointer arithmetic

- E.g. `ptr[3] = ...;`

Solution: Pass Size as Parameter

```
int sumAll(int* a, int size) {  
    int i, sum = 0;  
    for (i = 0; i < size; i++) {  
        sum += a[i];  
    }  
    return sum;  
}
```

- ❖ Standard idiom in C programs

C command line args

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[]) {
    for (int i = 0; i < argc; i++) {
        printf("%s\n", argv[i]);
    }
    return EXIT_SUCCESS;
}
```

- ❖ **argc** is the number of arguments given to the program.
 - The name of the program is counts as an argument.
- ❖ **argv** is an array of **char***'s (strings) that are the arguments
 - The name of the program is the first argument.

C command line args [Live Demo]

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[]) {
    for (int i = 0; i < argc; i++) {
        printf("%s\n", argv[i]);
    }
    return EXIT_SUCCESS;
}
```

❖ **Let's see an example...**

Formatted I/O

- ❖ Many programs need to convert between the bit values a computer manipulates and something a human can read
 - Example: converting between the binary encoding for an int into readable string
- ❖ Often done by the following functions or variants of them
 - `printf`
 - Prints a formatted string to the console
 - `scanf`
 - Reads a formatted string from the console

Strings without Objects

- ❖ Strings are central to C, very important for I/O
- ❖ In C, we don't have Objects but we need strings
- ❖ If a string is just a sequence of characters, we can have use array of characters as a string

- ❖ Example:

```
char str_arr[] = "Hello World!";  
char *str_ptr = "Hello World!";
```

Note: although these two are both "similar" they are quite different under the hood.

Null Termination

DO NOT FORGET THIS. THIS IS THE CAUSE OF MANY BUGS

- ❖ Arrays don't have a length, but we mark the end of a string with the null terminator character.

- The null terminator has value `0x00` or `'\0'`
- Well formed strings ***MUST*** be null terminated
- How else would `printf` know how to stop printing?

```
char str[] = "Hello";
```

- ❖ Example:

- Takes up 6 characters, 5 for "Hello" and 1 for the null terminator

address	0x2000	0x2001	0x2002	0x2003	0x2004	0x2005
value	'H'	'e'	'l'	'l'	'o'	'\0'