pictures I took recently

COMP1511/COMP1911 Week 5!

M13B: 1pm – 4pm || T11X: 11am – 2pm

Tutors: William (me!) + Jason || Daniel

My GitHub:

https://github.com/william-o-s/unsw_comp1511_tutoring

Reminders (also, this is a check-in week!)

Week 6

Week 6 does not have tutorials (you are welcome to rock up)

Assignment 1: CS Sokoban

Brief: https://cgi.cse.unsw.edu.au/~cs1511/24T2/flask.cgi/assignments/ass1/index.html

Demo: Help Sessions

Timetable: https://cgi.cse.unsw.edu.au/~cs1511/current/flask.cgi/help-sessions/

Tutorial Agenda:

Part 1

Part 2

Part 3

2D arrays are almost identical to 1D arrays

```
// 1D Array
int array_1d[3] = { 1, 2, 3 };
int array_1d_all_zeros[10] = { 0 };

// 2D Array
int array_2d[3][3] = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
int array_2d_all_zeros[10][20] = { 0 };     // yes, this is exactly the same
```

Functions have their own 'scopes', or, are their own 'boxes'

invisible wall

```
int main(void) {
    int x = 0;
    another_function();

    return 0;
}
```

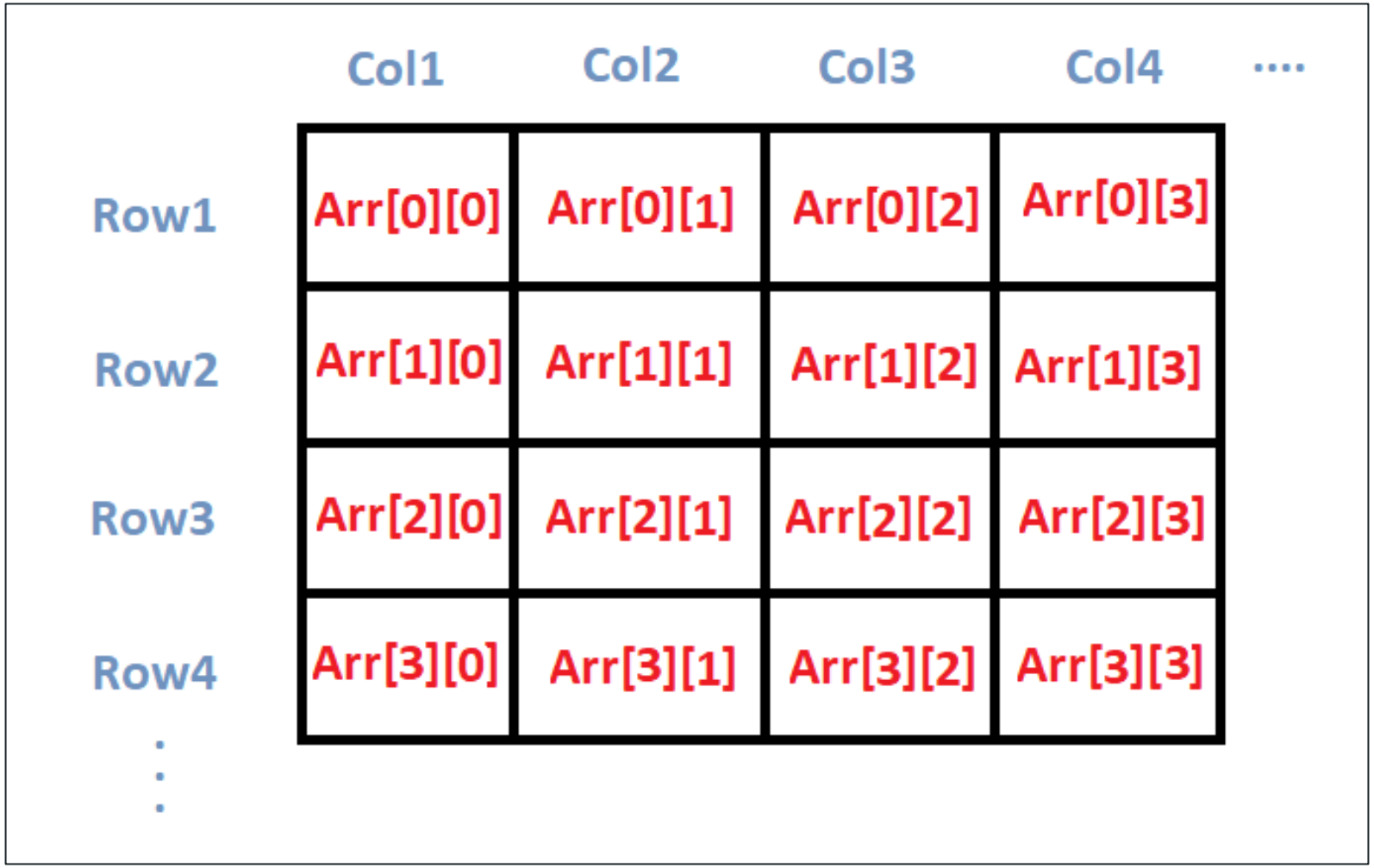
```
void another_function() {
    int x = 1;
    // this doesn't affect the `x` in main()
}
```

Let's briefly recap how input works in the terminal

2D arrays are almost identical to 1D arrays

```
// 1D Array
int array_1d[3] = { 1, 2, 3 };
int array_1d_all_zeros[10] = { 0 };

// 2D Array
int array_2d[3][3] = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
int array_2d_all_zeros[10][20] = { 0 };     // yes, this is exactly the same
```

…including array indexing…



Source: DigitalOcean

...or reading then printing...

```c
// Reading and printing the first element
printf("%d", array_1d[0]);
printf("%d", array_2d[0][0]);
```

Source: 3 cups of coffee

...or writing to an index

```
// Writing into the first element
array_1d[0] = 7;
array_2d[0][0] = 7;
```

Let’s write a galaxy program!

Functions have their own ‘scopes’, or, are their own ‘boxes’

invisible wall

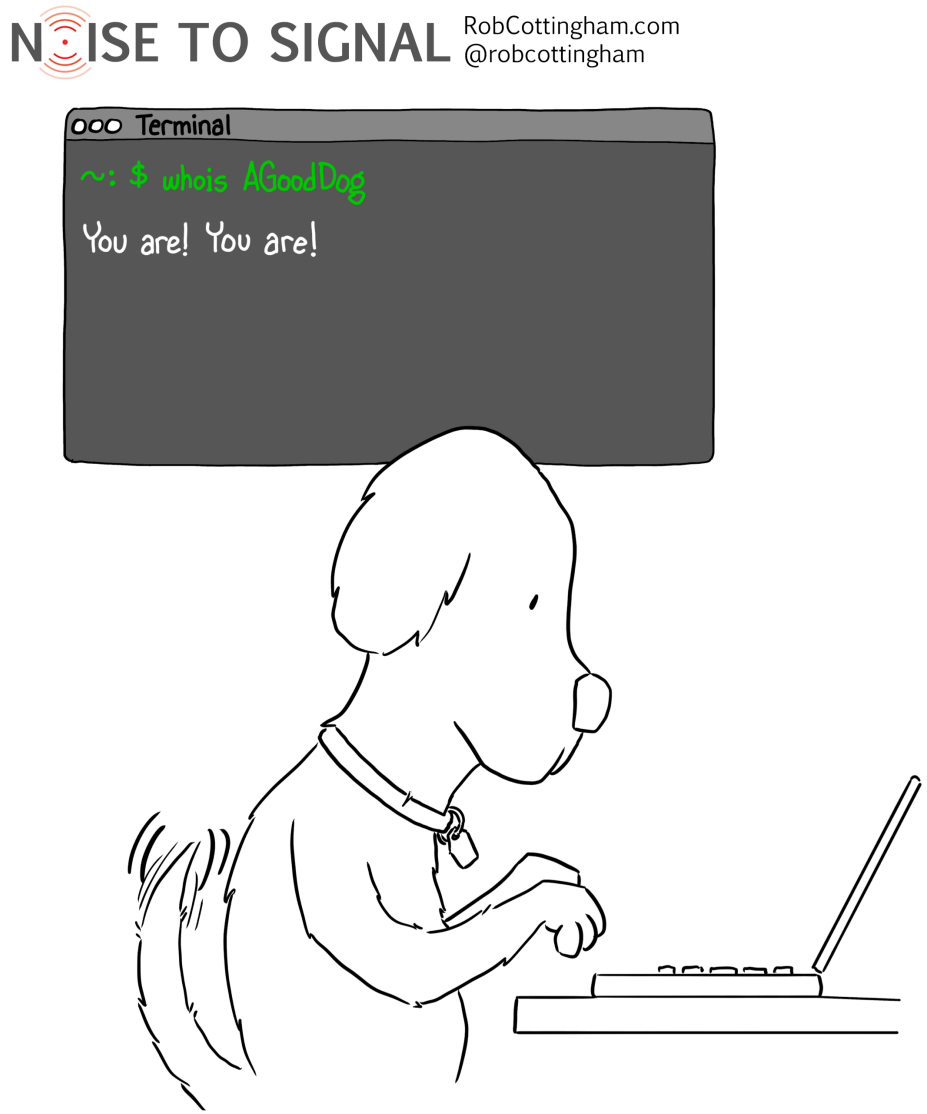
```c
int main(void) {
    int x = 0;
    another_function();

    return 0;
}
```

```c
void another_function() {
    int x = 1;
    // this doesn't affect the `x` in main()
}
```

Refactor the galaxy program with functions + add a sum function

Let’s briefly recap how input works in the terminal

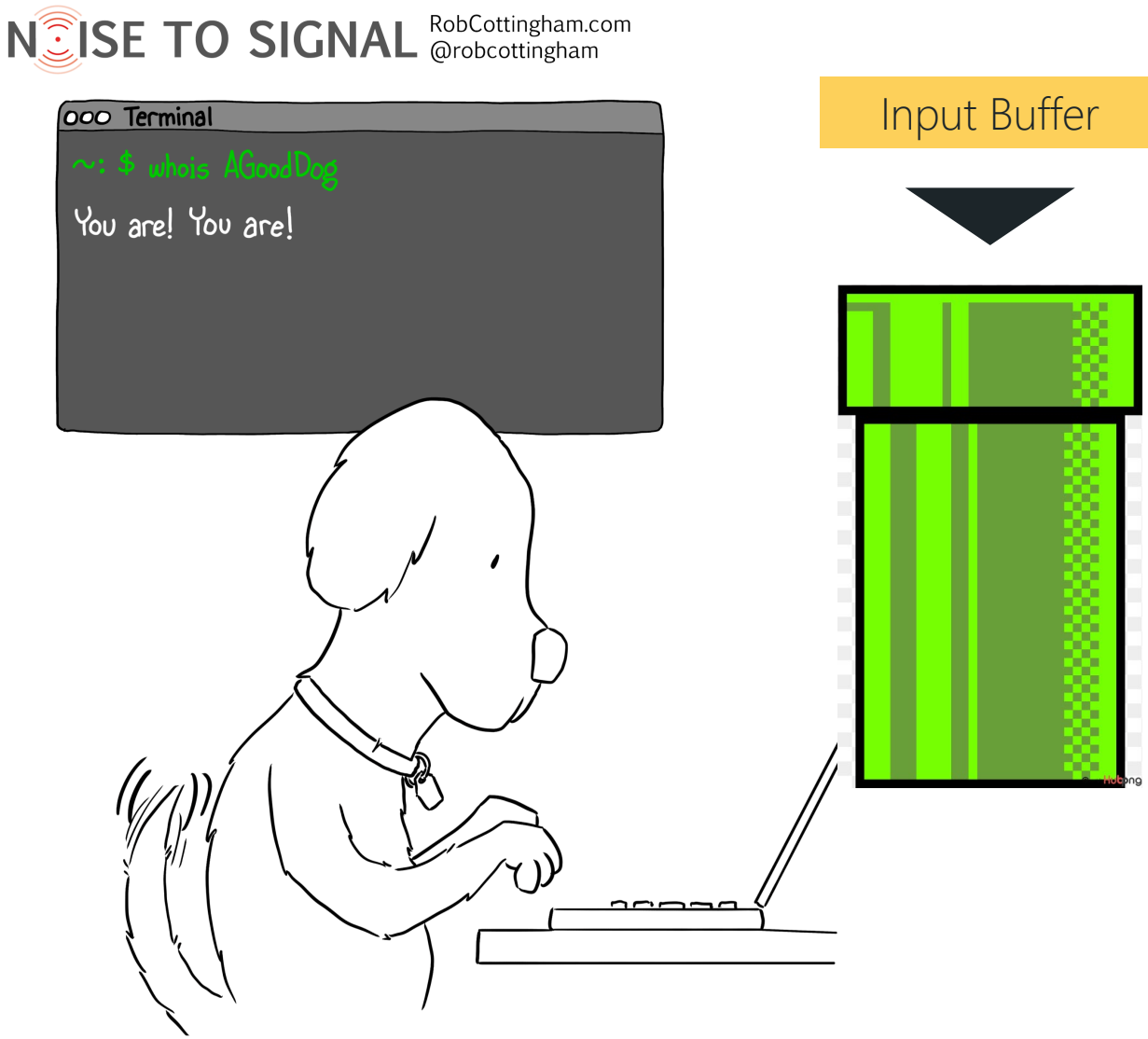
The user (that's you!) gives some input to the terminal



Input

"comp1?11"

The terminal stores all input characters in a buffer



Input

'c' → 'o' → 'm' → 'p' → '1' → '?' → '1' → '1'

The buffer contains every ‘next’ character

scanf

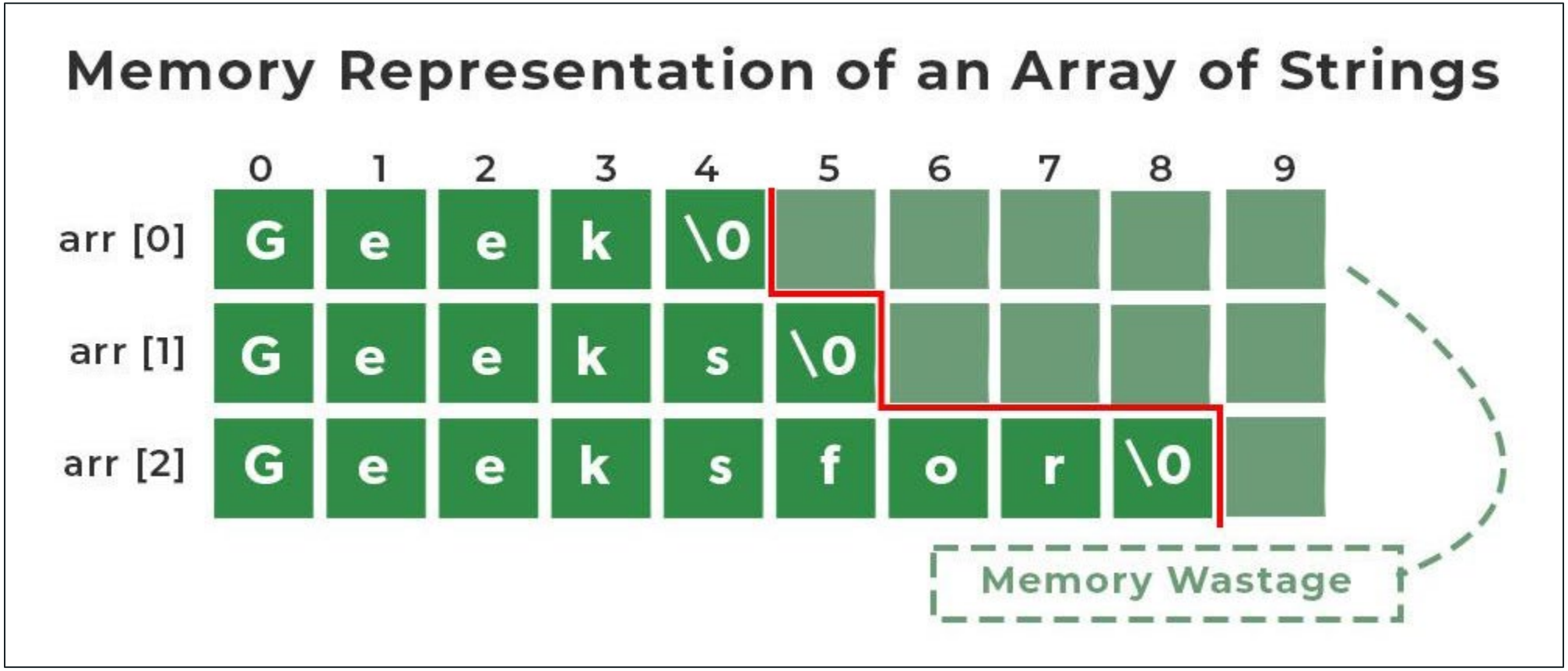
Input

‘c’ → ‘o’ → ‘m’ → ‘p’ → ‘1’ → ‘?’ → ‘1’ → ‘1’

Are all strings arrays? Are all arrays strings?

Source: GeeksforGeeks

Strings are stored as arrays, and always have a null terminator



Source: GeeksforGeeks

```c
// part3_simple_string.c
//
// This program was written by Sofia De Bellis (z5418801)
// on March 2024
//
// This program demonstrates how to work with strings in C.

#include <stdio.h>

#define MAX_SIZE 1024

int main(void) {
    // Declare and initialise a string
    char my_string[] = "Hello world!";

    // Traverse the string and print each character
    int i = 0;
    while (my_string[i] != '\0') {
        printf("%c", my_string[i]);
        i++;
    }
    printf("\n");

    // Another way to traverse the string and print each character
    for (int i = 0; my_string[i] != '\0'; i++) {
        printf("%c", my_string[i]);
    }
    // How to print a string in its entirety
    printf("My string: %s\n", my_string);

    // Declare a string
    char another_string[MAX_SIZE];

    // Read a string from the user, note we DO NOT use scanf for strings
    printf("Enter a string: ");
    fgets(another_string, MAX_SIZE, stdin);

    // Print the string using fputs
    fputs(another_string, stdout);

    return 0;
}
```

Any questions on how this code works?

Finally, let's do group exercises to practice (but first, an easter egg!)

VSCode Shortcuts

- Start with Ctrl+Shift+P
  - "Toggle Multi-Cursor Editor"
  - Convert text casing: (highlight text) → Ctrl + Shift + P → "Transform to ..."
- Multiple Cursors: Ctrl + Click anywhere
  - Cursor over multiple lines vertically: Shift + Alt + Click on line
- Duplicate Line: Ctrl + Shift + Alt + Up/Down Arrow
- Move Lines: Alt + Up/Down Arrow
- Change All Occurrences: Ctrl + Shift + L or Ctrl + D
- Indentation: (Highlight line/lines) → Ctrl + Left/Right Square Bracket
- Find and Replace: Ctrl + F → (click dropdown) → Replace next

VSCode Shortcuts

- Start with Ctrl+Shift+P
  - "Toggle Multi-Cursor Editor"
  - Convert text casing: (highlight text) → Ctrl + Shift + P → "Transform to …"
- Multiple Cursors: Ctrl + Click anywhere
  - Cursor over multiple lines vertically: Shift + Alt + Click on line
- Duplicate Line: Ctrl + Shift + Alt + Up/Down Arrow
- Move Lines: Alt + Up/Down Arrow
- Change All Occurrences: Ctrl + Shift + L or Ctrl + D
- Indentation: (Highlight line/lines) → Ctrl + Left/Right Square Bracket
- Find and Replace: Ctrl + F → (click dropdown) → Replace next