COMP1511 PROGRAMMING FUNDAMENTALS

Lecture 6

We have hit the amazing data structure
- ARRAYS !

YESTERDAY...

- We styled to our heart's content
- Learned about Functions

TODAY...

- Let's return to talking about storing things in memory as we work
- Introducing, the one, the only, the fabulous ARRAY

WHERE IS THE CODE?

LIVE LECTURE CODE CAN BE FOUND HERE:

https://cgi.cse.unsw.edu.au/~cs1511/21T3/live/Week03/

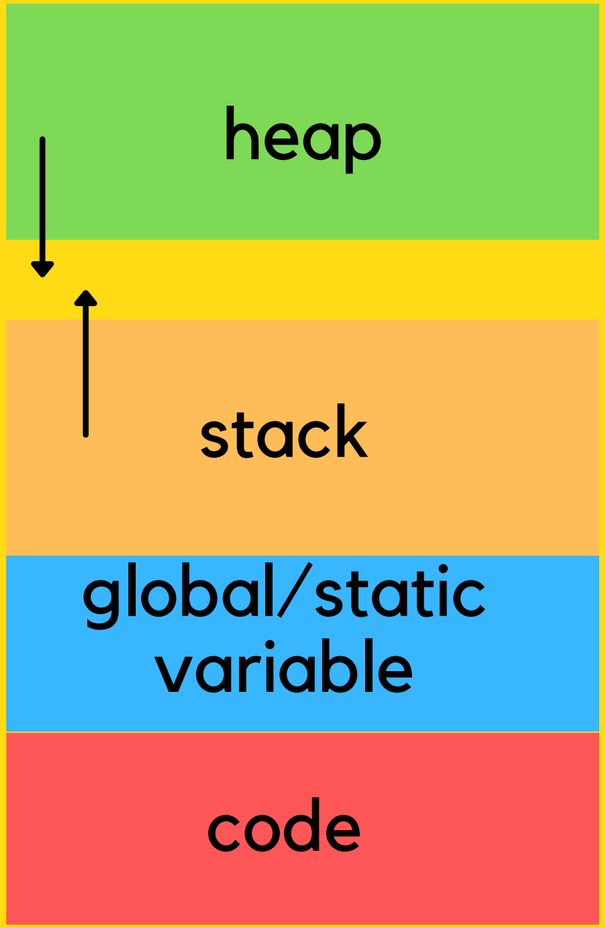
LET US AGAIN REVISIT MEMORY

A BASIC OVERVIEW

- Things are a lot more complex, but for our needs and for your understanding, you really need a basic knowledge of what is happening.

MEMORY

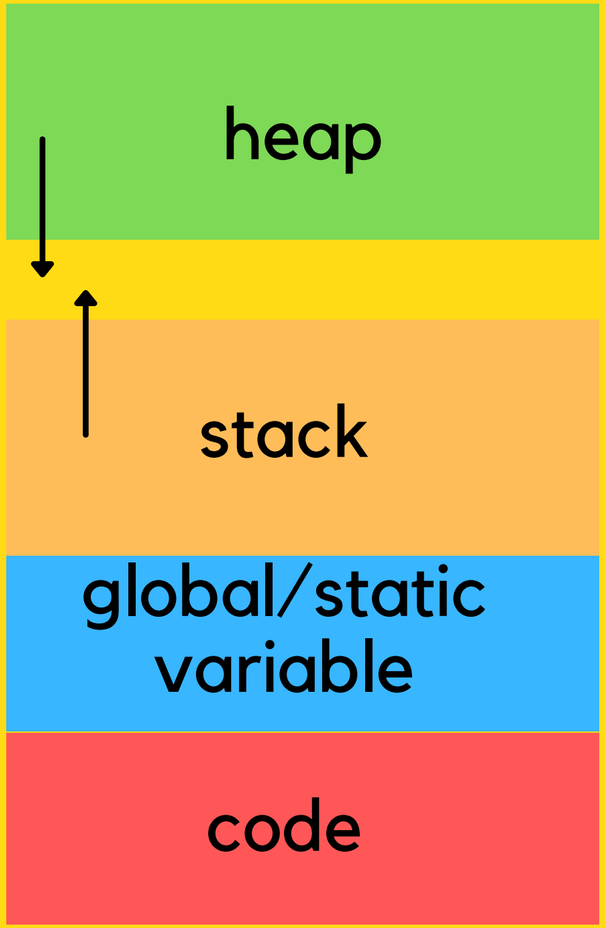
THE WAY THINGS ARE...



- Our C file is stored on the hard drive
- Our Compiler compiles the code into another file that the computer can read
- When we execute code, the CPU will actually process the instructions and perform basic arithmetic, but the RAM will keep track of all the data needed in those instructions and operations, such as our variables.
- Reading and writing to variables will change the numbers in RAM
- Memory is divided into the stack and the heap
- The stack is an ordered stack and the heap is a random free for all - insert something where you can find space for it.

MEMORY

THE WAY THINGS ARE...



- Stack memory is where relevant information about your program goes:
  - which functions are called,
  - what variables you created,
- Once your block of code finishes running {}, the function calls and variables will be removed from the stack (it's alive!)
- It means at compile time we can allocate stack memory space (not at run time)
- The stack is controlled by the program NOT BY THE developer
- The heap is controlled by the developer (more on this in a few weeks) and can be changed at run time

MEMORY IS IMPORTANT

WITHOUT MEMORY, WE CAN'T REALLY RUN ANYTHING

- Think of your own memory and what it allows you to do.
- Computer memory is important to consider when you are writing your code (we don't focus on this in 1511, but you will in later courses)
- The more you waste memory, the slower your program gets... you will learn all about this in later computing courses!

HOW DO WE EFFICIENTLY SOLVE PROBLEMS?

DIFFERENT PROBLEMS HAVE DIFFERENT OPTIMUM SOLUTIONS

- In this course we will learn about two pretty cool data structures:
  - Arrays (NOW!)
  - Linked Lists (after flexibility week)
- There are of course other data structures that you will learn about in further computing courses
- Choosing the right structure to house our data depends on what the problem is and what you are trying to achieve. Some structures lend themselves better to certain types of problems.

WHAT IS AN ARRAY?

I BELIEVE IT IS THE SINGLE MOST IMPORTANT DATA STRUCTURE ...

- A collection of variables all of the same type
  - Think about how this is very different to a struct
- We want to be able to deal with this collection as a whole entity, where we can:
  - Access any variable in this collection easily
  - Change any variable in this collection easily

SO WHAT KINDS OF PROBLEMS DO ARRAYS SOLVE?

NOTICE THAT EACH OF THESE COLLECTIONS HAS THE SAME TYPE OF VARIABLE I AM RECORDING

- Let's say I want to record the daily case numbers in NSW during the COVID-19 pandemic
- What about the daily temperatures?
- The amount of time daily that I spend walking my dogs
- How many deliveries I get per day during lockdown

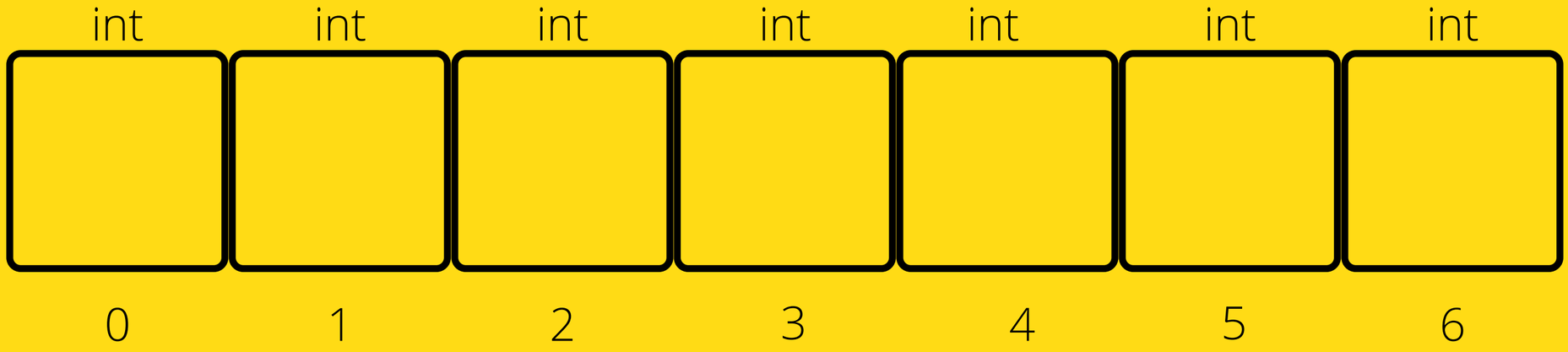
Can you guys think of other examples?

ARRAY VISUALLY

NOTE: ALL ELEMENTS OF AN ARRAY MUST BE OF THE SAME DATA TYPE (HOMOGENOUS)

- If we group our data type as a collection, for example a collection of integers:
  - We can access them as a group(collection)
  - We can loop through and access each individual element of that collection

this array holds 7 integers



You can access elements of an array by referring to their index

LET'S TRY TO SOLVE ONE OF OUR PROBLEMS NOW WITHOUT ARRAYS AND WITH ARRAYS

IT'S DEMO TIME...

lockdown_parcels.c

Let's say I am tracking how many parcels I receive each day over the course of a given week in lockdown - without arrays:

```c
int monday = 1;
int tuesday = 2;
int wednesday = 3;
int thursday = 4;
int friday = 5;
int saturday = 6;
int sunday = 7; //self-isolation peaked

//Any day that has more than two parcels is extreme, check whether any day
//has been extreme

if (monday > 2) {
    printf("Slow down on the parcels, %d parcels is a bit extreme", monday);
}
if (tuesday > 2) {
    printf("Slow down on the parcels, %d parcels is a bit extreme", tuesday);
}
if (wednesday > 2) {
    printf("Slow down on the parcels, %d parcels is a bit extreme", wednesday);
}
if (thursday > 2) {
    printf("Slow down on the parcels, %d parcels is a bit extreme", thursday);
}
if (friday > 2) {
    printf("Slow down on the parcels, %d parcels is a bit extreme", friday);
}
if (saturday > 2) {
    printf("Slow down on the parcels, %d parcels is a bit extreme", saturday);
}
if (sunday > 2) {
    printf("Slow down on the parcels, %d parcels is a bit extreme", sunday);
}
```

LET'S TRY TO SOLVE ONE OF OUR PROBLEMS NOW WITHOUT ARRAYS AND WITH ARRAYS

IT'S DEMO TIME...

What do we think? Does that look like an efficient way to do things?

instead of this mess!

Declaring each day of the week as a separate variable

```
int monday = 1;
int tuesday = 2;
int wednesday = 3;
int thursday = 4;
int friday = 5;
int saturday = 6;
int sunday = 7; //self-isolation peaked
```

WITH ARRAYS

IT'S DEMO TIME...

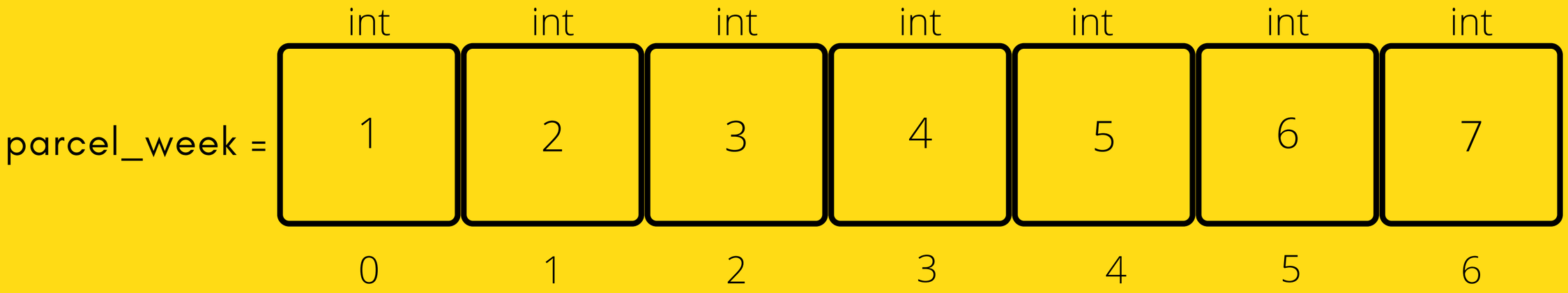
Similar to declaring a variable, we can create an array because every data type in that collection is an **int**, this means we would make an array of ints

```
int parcel_week[7] = {1, 2, 3, 4, 5, 6, 7}
```

declare an array

Declaring this as an array

this is what this array looks like if you want to visualise it



DECLARING AN ARRAY

A CLOSER LOOK

```
int parcel_week[7] = {1, 2, 3, 4, 5, 6, 7}
```

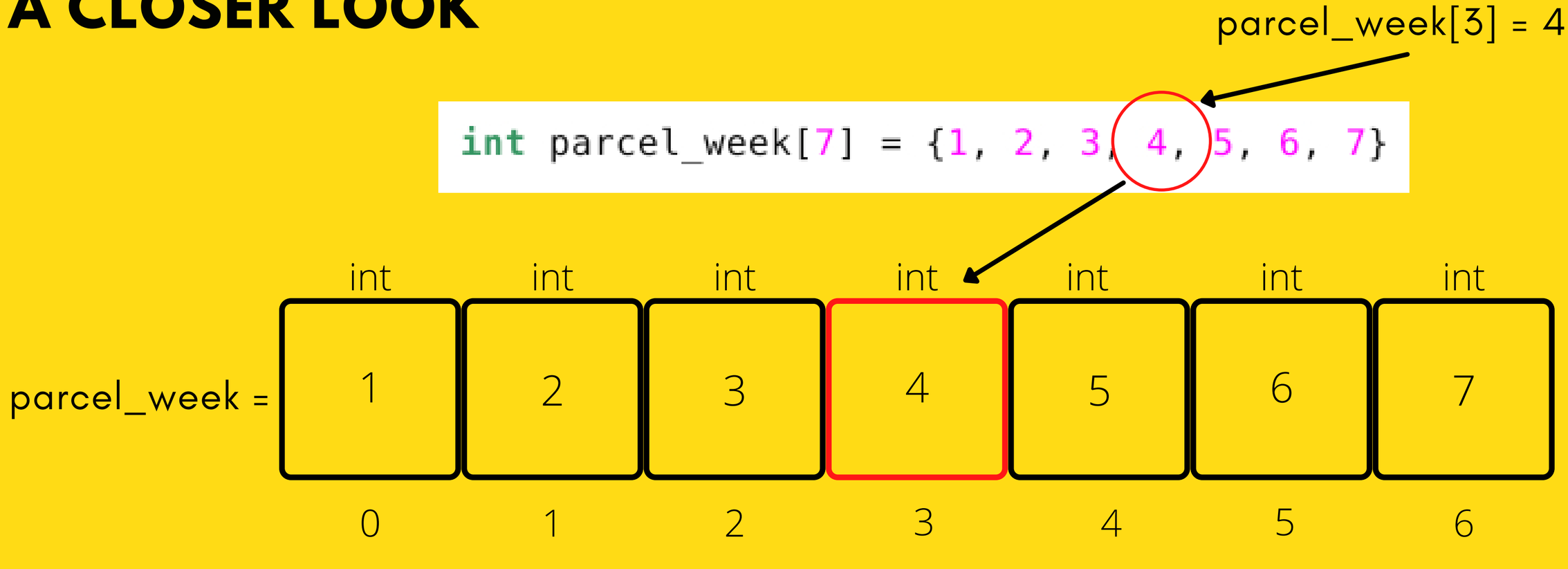
- You can access any element of the array by referencing its index
- Note, that indexes start from 0
- Trying to access an index that does not exist, will result in an error
- To declare an array, just like a variable:
  - state the type of array it will be first,
  - then give your array a name
  - square brackets that follow variable name tell C it is an array and the number inside the square brackets is what says how many elements there will be
- To initialise array - curly brackets will contain all elements separated by commas. If you have empty {}, it means to intialise the whole array to 0

ACCESSING, WRITING TO AN ARRAY

A CLOSER LOOK

- You can access any element of the array, by saying what index of the array you want to access. For example, parcel_week[3] = 4

Remember that indexing starts with 0, which is why the third index actually refers to the fourth element...



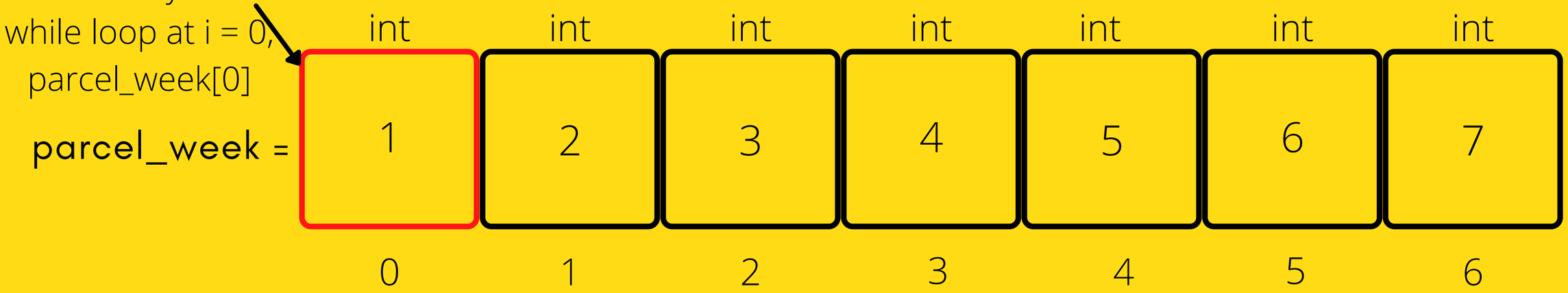
HOW DO WE PLAY WITH ARRAYS?

A CLOSER LOOK

```c
int parcel_week[7] = {1, 2, 3, 4, 5, 6, 7}

int i = 0;
while (i < 7) {
    printf("%d ", parcel_week[i]);
    i++;
}
```

First entry into while loop at i = 0, parcel_week[0]

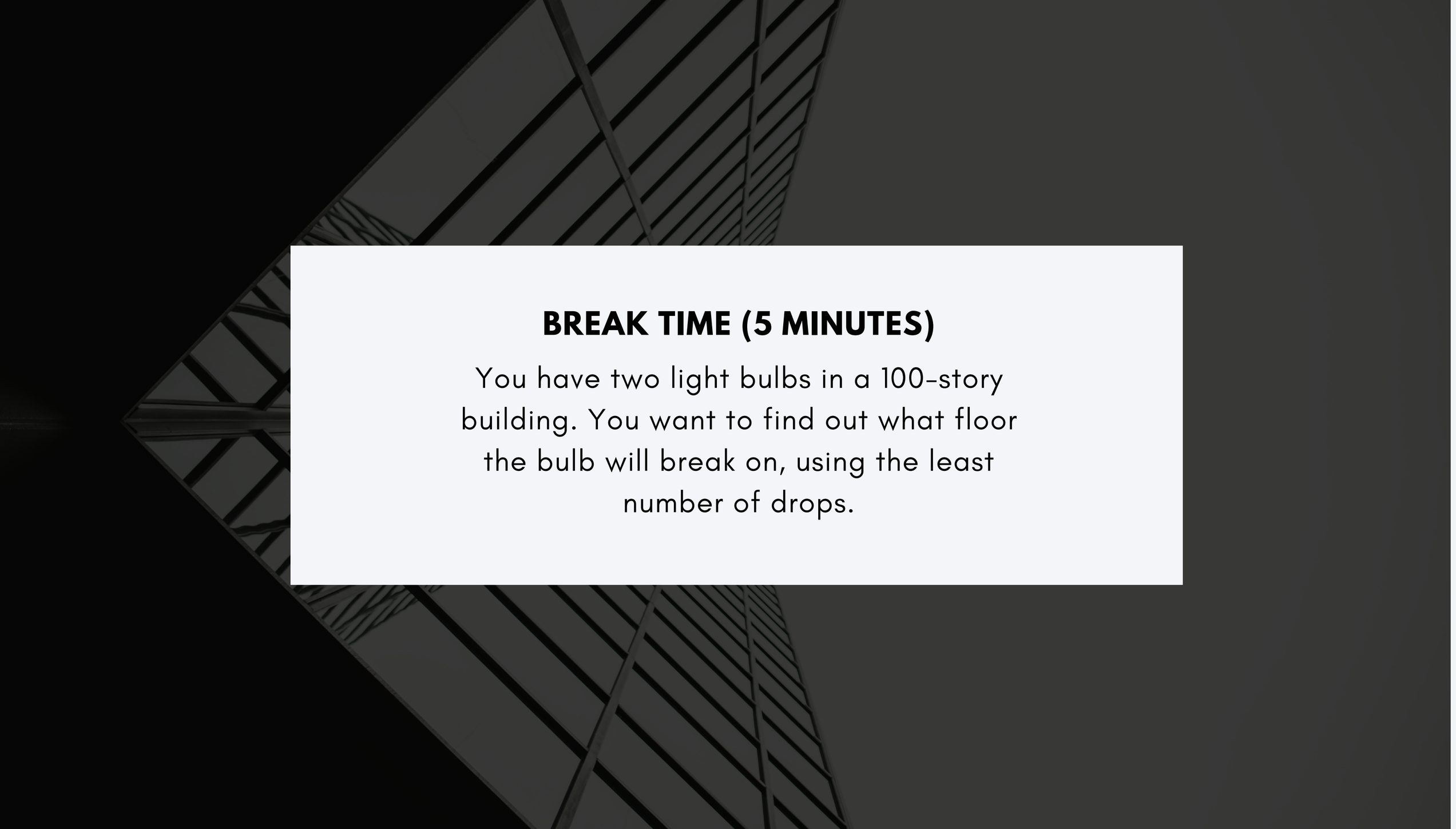


- You can't printf() a whole array, but you can print individual elements (consider how you could go through the array to print out every element...)
- You can't scanf() a whole array, i.e. a line of user input test into an array, but you can can scanf() individual elements (think how to do every element in an array...)

DEMO TIME

ARRAYS: DECLARE, INITIALISE, SIMPLE ACCESS

lockdown_parcel.c



BREAK TIME (5 MINUTES)

You have two light bulbs in a 100-story building. You want to find out what floor the bulb will break on, using the least number of drops.

LET'S SOLVE A PROBLEM

1. TRACKING SCORES

array_scores.c

Four players are playing a dice game. For every round of the game, each player rolls two dice, and the sum of those dice is their final score for that round. After everyone has rolled their dice in the game, we want to be able to find out who won that game (highest score), and we also want to know what the total of the scores is in any given round.

Break down the problem into steps... Which of those steps can we make into functions?

FEEDBACK?

PLEASE LET ME KNOW ANY FEEDBACK FROM TODAY'S LECTURE!

www.menti.com

Code: 3758 2755

WHAT DID WE LEARN TODAY?

MEMORY

LET'S START EXPLORING ARRAYS

lockdown_parcel.c

array_scores.c

ANY QUESTIONS?

DON'T FORGET YOU CAN ALWAYS EMAIL US ON CS1511@CSE.UNSW.EDU.AU FOR ANY ADMIN QUESTIONS

PLEASE ASK IN THE FORUM FOR CONTENT RELATED QUESTIONS