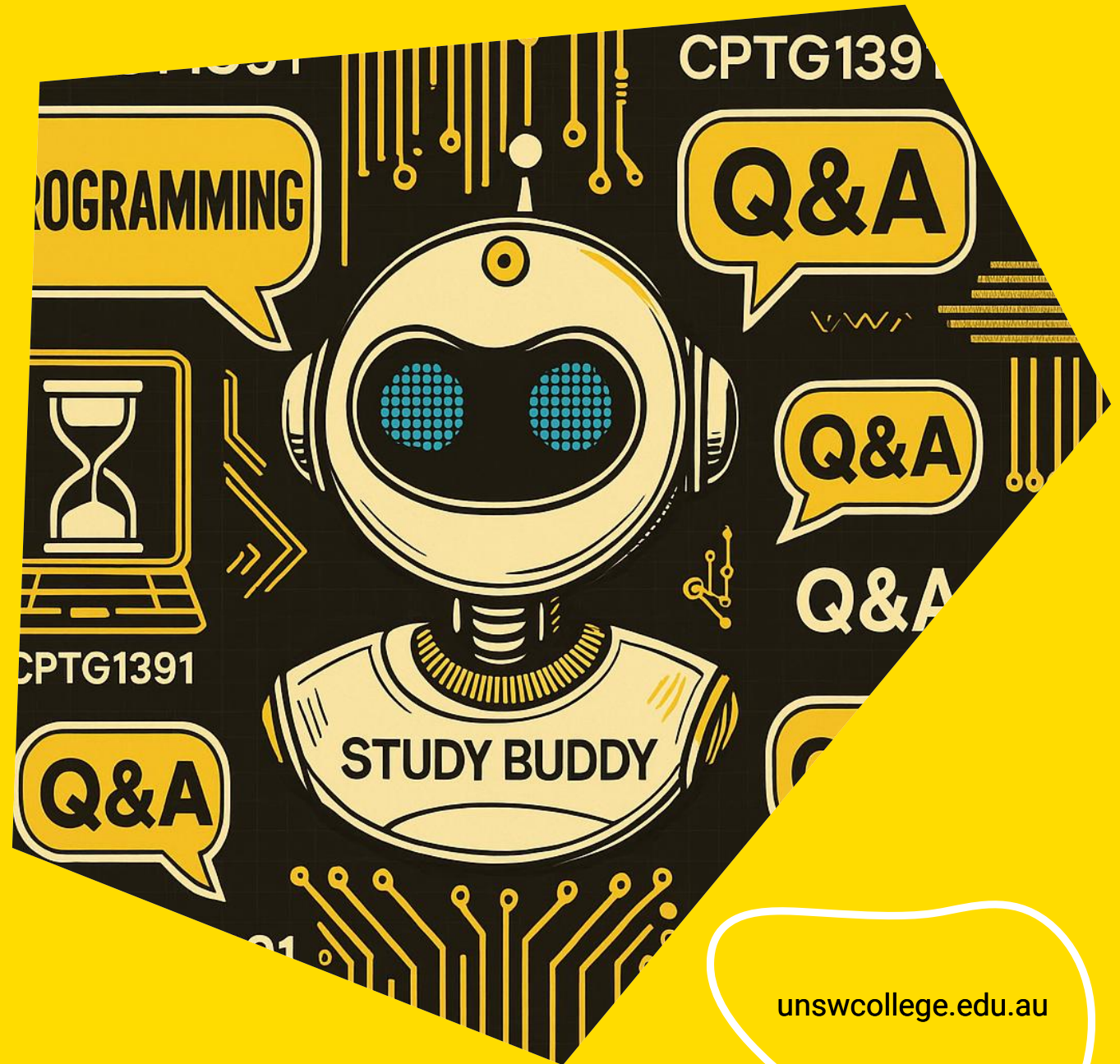


DPST1091 / CPTG1391
Introduction to Programming
Week 4 – Lecture 1

Lecturer and Course Convener:

Dr Pantea Aria

2D arrays



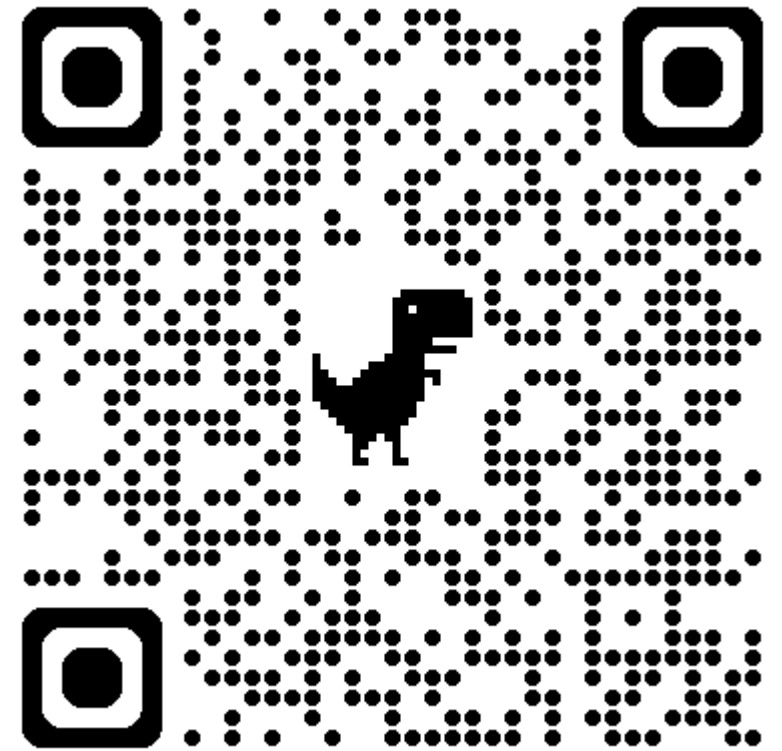
unswcollege.edu.au

Agenda

- **Today**
 - **arrays of structs**
 - **2D arrays**

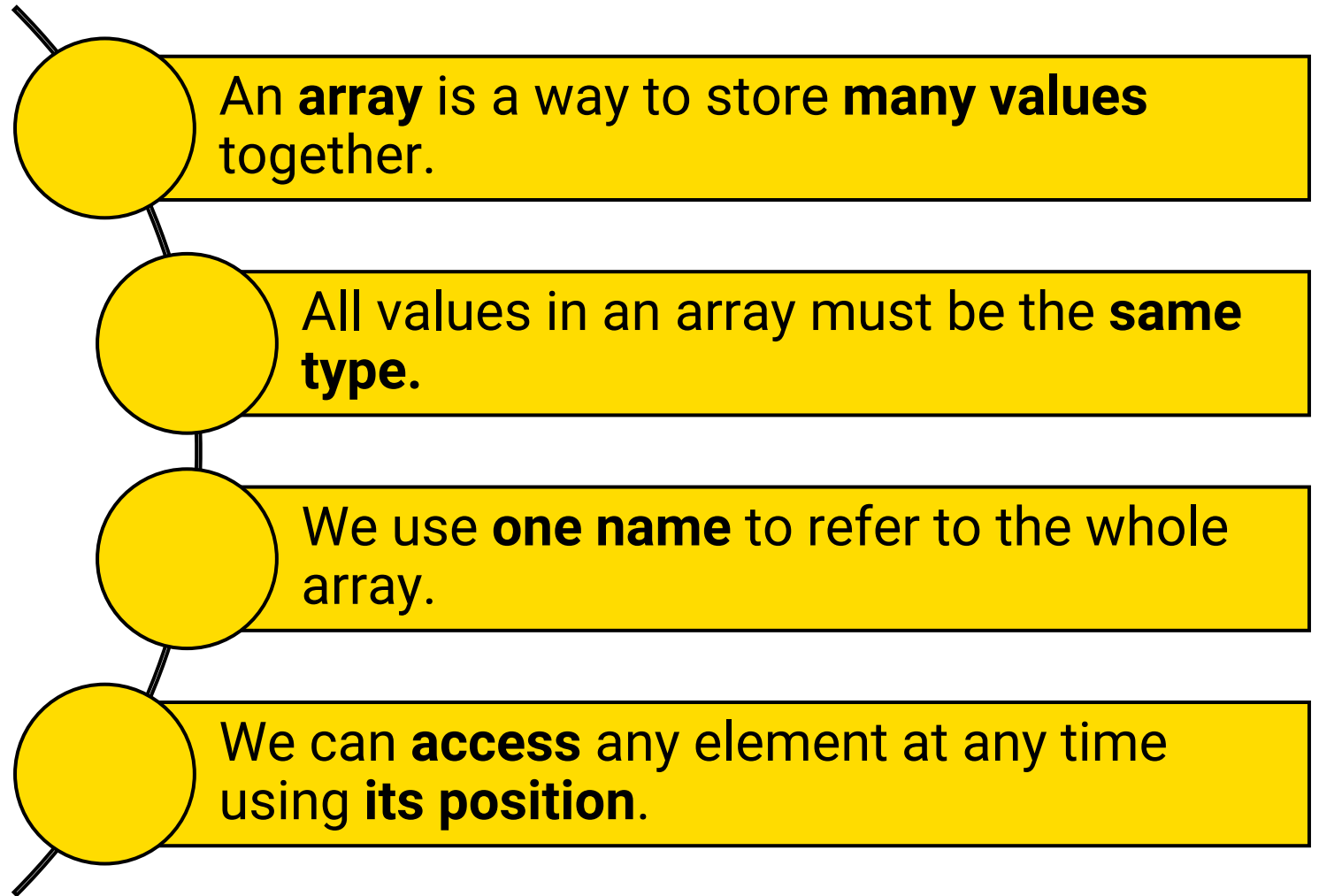
Link to the Live Lecture Code

<https://cgi.cse.unsw.edu.au/~dp1091/26T2/code/04>



Live lecture code is written for teaching, not perfection.
It may include extra comments and may not always follow
ideal coding style.

Arrays recap



Array Declaration

```
int quiz_scores[5];
```



0	1	2	3	4

Declare and initialise

```
int quiz_scores[5] = {12, 15, 18, 20, 17};
```

```
// initialise all elements to 0  
int quiz_scores[5] = {};
```

```
// this will create a 4-element array  
int ages[] = {18, 19, 20, 21};
```

```
// this Will create a 14-element array,  
// with the first 7 elements initialised and  
// the remaining 7 set to 0  
int age[14] = {3, 2, 1, 2, 1, 3, 5};
```

Accessing the elements

```
int quiz_scores[5] = { 12, 15, 18, 20, 17};
```



0	1	2	3	4
12	15	18	20	17

```
int first_score = quiz_scores[0];
```

first_score

12

Demo

→array_recap.c

Arrays and Functions

Arrays can be **passed** into functions just like other variables.

When **passing an array**, the function must also be told **how many elements** the array contains.

This allows the function to safely work with **arrays of different sizes**.

```
// Function that works with an array of integers of any length  
void display_array(int length, int numbers[]);
```

Example

```
void display_array(int length, int numbers[]);

int main(void) {
    int temperatures[] = {18, 21, 25, 19};
    int scores[] = {100, 75};

    display_array(4, temperatures);
    display_array(2, scores);

    return 0;
}

void display_array(int length, int numbers[]) {
    int i = 0;
    while i < length) {
        printf("%d ", numbers[i]);
        i++;
    }
}
```

Important Behaviour of Arrays in Functions

When an array is passed to a function, **the entire array is not copied.**

The function works with the **original array** created in the caller function.

Any changes made inside the function affect the array outside the function.

This behaviour will be explained in more depth in **later weeks.**

Initialising an Array Using a Function

- A **function** can be used to **fill an array** with values.
- The **array is created in main** and **passed** into helper functions.

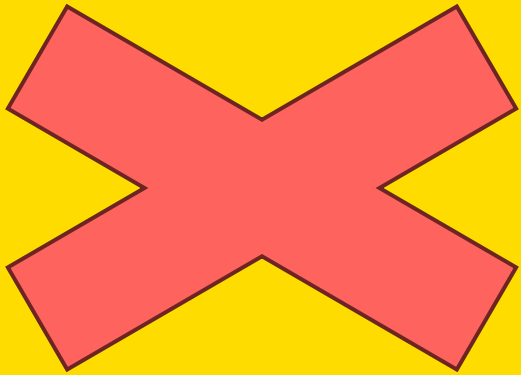
```
#define SIZE 5
```

```
int main(void) {  
    int values[SIZE];  
  
    read_values(SIZE, values);  
    display_array(SIZE, values);  
  
    return 0;  
}
```

```
void read_values(int length, int array[]) {  
    int i = 0;  
    while (i < length) {  
        scanf("%d", &array[i]);  
        i++;  
    }  
}
```

array is not
a copy of
the **values**
array.

Returning Arrays from Functions



- Trying to **return an entire array** from a function **does not work**.
- Although the code **may compile**, it leads to serious **runtime problems**.
- We will explore why this happens later in the course.

```
// This does NOT work
int[] read_values(void) {
    int array[SIZE];

    for (int i = 0; i < SIZE; i++) {
        scanf("%d", &array[i]);
    }

    return array;    // X invalid
}
```

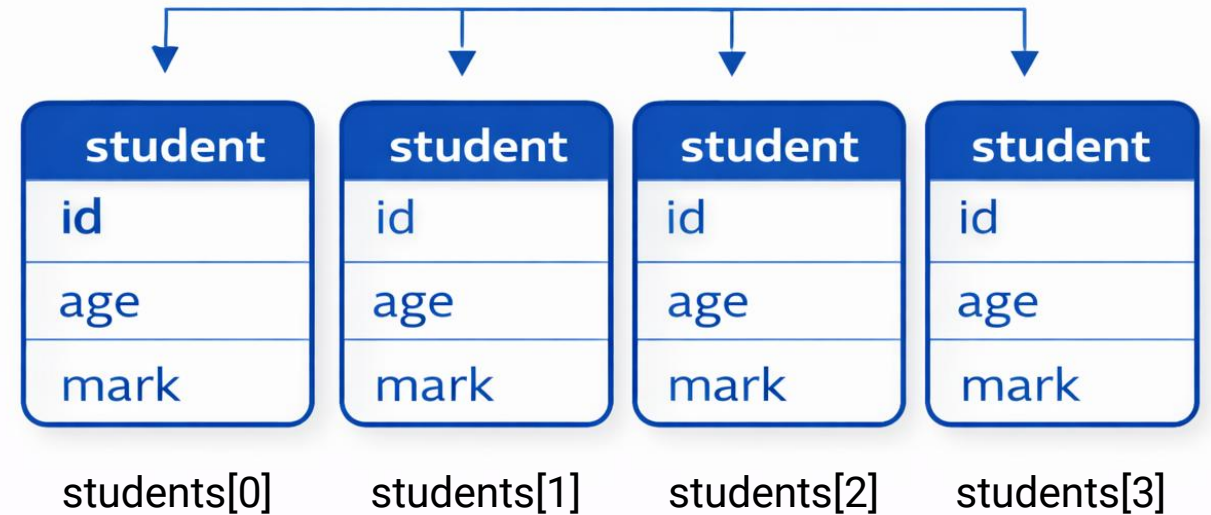
Demo

→array_function_recap.c

Array of Struct

```
struct student {  
    int id,  
    int age,  
    double mark;  
};
```

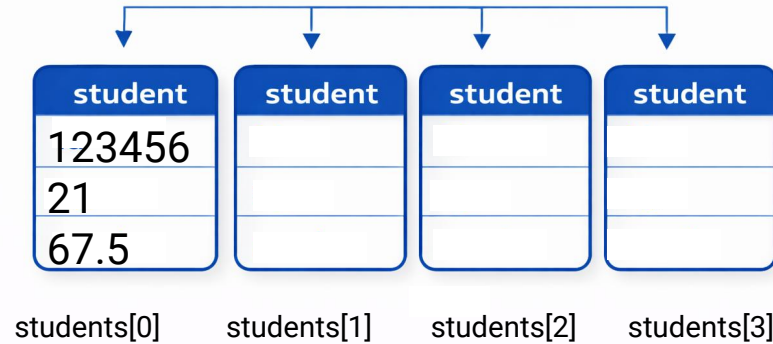
```
struct student students[4];
```



Accessing the elements of an array of structures

```
struct student {  
    int id,  
    int age,  
    double mark;  
};
```

```
struct student students[4];
```



```
students[0].id = 123456;  
students[0].age = 21;  
students[0].mark = 67.5;
```

Demo

→array_of_struct.c

2 Dimentional Arrays

arrays of arrays

It stores data in **rows and columns**, like a **table or grid**.
How to declare?

```
<type> <identifier>[<rows>][<cols>;
```

```
// Declaring a 2D array  
int marks[3][4];
```

This means:

3 rows

4 columns

Total elements = $3 \times 4 = 12$

You can picture it like this:

```
[] [] [] []  
[] [] [] []  
[] [] [] []
```

Initialising a two-dimensional array

```
// Initialising a 2D array
int marks[3][4] = {
    {10, 12, 14, 16},
    {11, 13, 15, 17},
    {18, 19, 20, 21}
};
```

Each inner {} is one row.

	col 0	col 1	col 2	col 3
row 0	10	12	14	16
row 1	11	13	15	17
row 2	18	19	20	21

marks

How to traverse the array

A 2D array has
two dimensions:

- rows
- columns

So, we need:

- one loop to move through the **rows**
- one loop to move through the **columns**

```
int row = 0;
// outer loop for rows
while (row < 3) {
    int col = 0;
    // inner loop for columns
    while (col < 4) {
        printf("%d ", marks[row][col]);
        col++;
    }
    printf("\n");
    row++;
}
```

Demo

- 2Darrays_input_elements
- 2Darrays_print_elements
- 2Darrays_sum_elements
- 2Darrays_sum_diagonal

What did we learn today?

Topic	Summary	Example
Arrays and Functions	Arrays can be passed into functions just like variables. The function also needs the array size.	<code>display_array(5, numbers);</code>
Function Parameter for Arrays	Function prototype with arrays, need to declare correctly	<code>void display_array(int length, int numbers[])</code>
Arrays Are Not Copied	When an array is passed into a function, the original array is used, not a copy.	Changing <code>numbers[0]</code> inside the function changes the original array.
Returning Arrays from Functions	In C, entire arrays cannot be directly returned from functions.	<code>return array; // invalid</code>
Array of Structures	Arrays can store structures, allowing multiple records of the same type.	<code>struct student students[4];</code>
Accessing Struct Elements	Use array index and dot operator to access structure fields.	<code>students[0].mark = 67.5;</code>
2D Arrays	A 2D array stores data in rows and columns like a table.	<code>int marks[3][4];</code>
Declaring 2D Arrays	First number = rows, second number = columns.	<code>int grid[2][3];</code>
Initialising 2D Arrays	Each inner <code>{ }</code> represents one row.	<code>int marks[2][2]={{1,2},{3,4}};</code>
Accessing 2D Array Elements	Use two indexes: one for row and one for column.	<code>marks[1][2]</code>
Traversing 2D Arrays	Nested loops are used: one loop for rows and one for columns.	<code>while(row < 3) { while(col < 4) { ... } }</code>

Voice of the Student

Anonymous ongoing feedback
Anything you wanted to share with me



Voice of the Student



[Voice of the Student – Fill out form](#)

See you soon ...