

COMP1511/1911 Programming Fundamentals

Week 4 Lecture 1

Arrays of structs

2D Arrays

Census Date

Term 3, 2026 - Census date (T3)

Last day to drop Teaching Period Three (T3) courses without financial liability.

Date	Academic period	Event Type
8 Oct 2026, 11.59pm	Term 3	Census date, Enrolment/Drop course

[About Census Dates | UNSW Current Students](#)

Announcement: Public Holiday

Yesterday was a Public Holiday

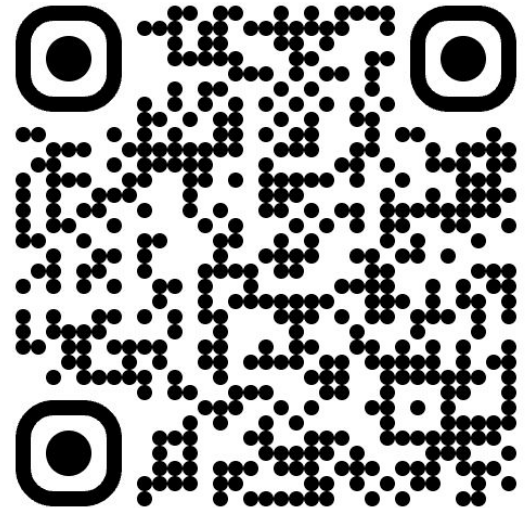
Pre-recorded lecture upload tomorrow.

Students in Monday **tutorial/labs** can book tickets to attend extra make up classes

<https://buytickets.at/comp1511unsw/2401086>

Lab 3 due - week 4 Tuesday 6pm (tonight)

Lab 4 due - week 5 Tuesday 6pm



Revision Sessions This Week (Not Recorded)

- Like a Hybrid tutorial/lab session
 - Structured with lab style questions
- **Week 4 Wednesday 07/10/2026 2pm-4PM in Lyre Lab K17-G12**
- **Week 4 Thursday 08/10/2026 3-5PM in Lyre Lab K17-G12**
- [More information and sign up details](#)

Help Sessions

Help Session Schedule

- [COMP\(1511|1911\) 26T3 – COMP\(1511|1911\) Help Sessions](#)
- Please note that help sessions are not in labs and are BYOD (Bring Your Own Device) sessions. 

Drop in Sessions

Email the course account if you feel you need more one on one time and would like to book a longer one-on one session for extra help.

Last Week

- Functions
- Style
- Arrays

We have covered a lot in the course so far!

You will get plenty of chances to practice these skills in tutorials and labs and assignments.

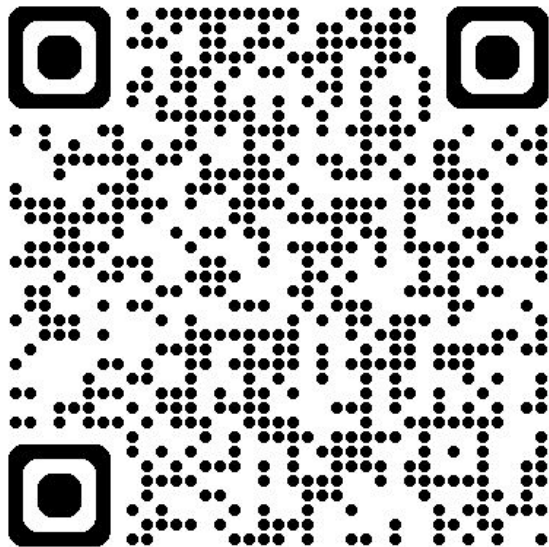
We don't expect you to have mastered them all yet.

Today's Lecture

- For loop recap
- Arrays recap
- Array Functions
- Arrays of structs
- 2D arrays
- Assignment 1 intro

Link to Week 4 Live Lecture Code

https://cgi.cse.unsw.edu.au/~cs1511/26T3/code/week_4/



Recap: while loop vs for loop

These two loops do exactly the same thing!

```
int i = 0;
while (i < 10) {
    printf("%d\n", i);
    i++;
}
```

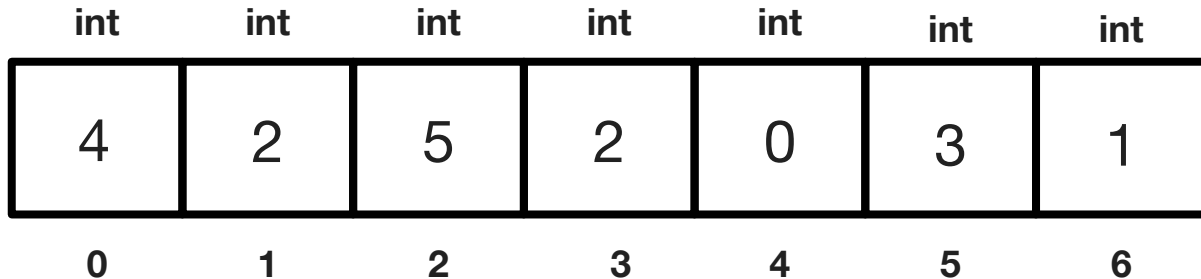
```
for (int i = 0; i < 10; i++) {
    printf("%d\n", i);
}
```

Recap: Visualising an Array

So let's say we have this declared and initialised:

```
int chocolates_eaten[7] = {4, 2, 5, 2, 0, 3, 1};
```

This is what it looks like visually:

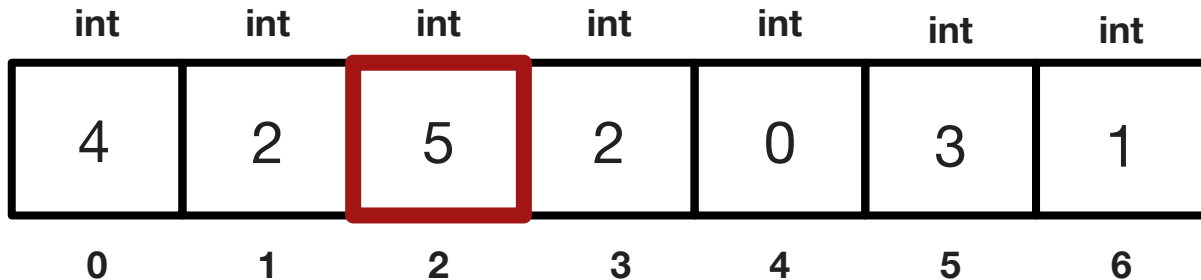


Note: The array holds 7 elements. Indexes start at 0

Recap: Accessing Elements in an Array

- You can access any element of the array by using its index
 - Indexes start from 0
 - Trying to access an index that does not exist, will result in an error

```
int chocolates_eaten[7] = {4, 2, 5, 2, 0, 3, 1};
```

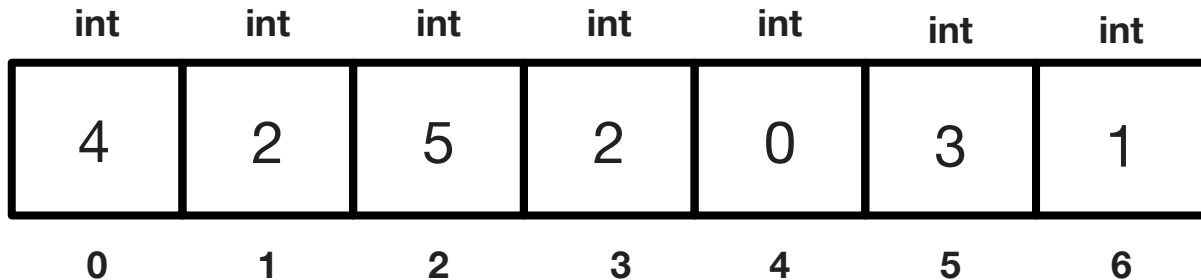


`chocolates_eaten[2]` would access the third element

Recap: Accessing Elements in an Array

- You can access any element of the array by using its index
 - Indexes start from 0
 - Trying to access an index that does not exist, will result in an error

```
int chocolates_eaten[7] = {4, 2, 5, 2, 0, 3, 1};
```



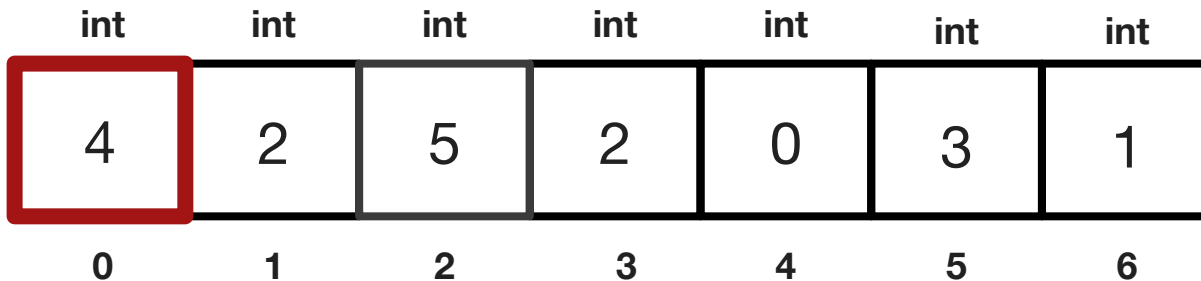
`chocolates_eaten[7]` would cause a run-time error with dcc

Recap: Traversing an Array

```
int chocolates_eaten[7] = {4, 2, 5, 2, 0, 3, 1};  
int i = 0;  
while (i < 7) {  
    printf("%d ", chocolates_eaten[i]);  
    i++;  
}
```

Start at index 0

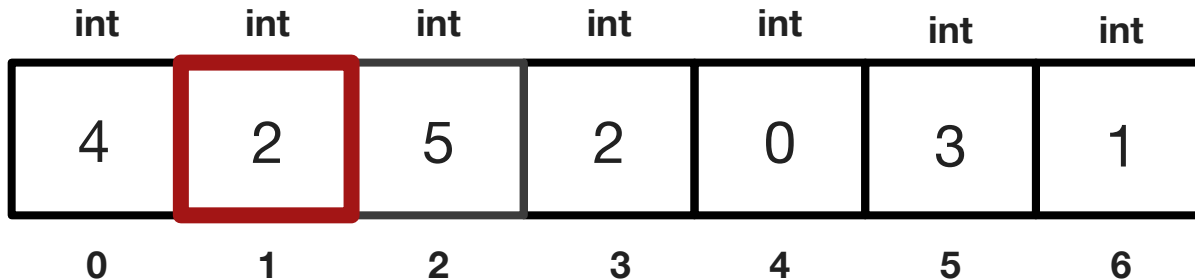
chocolates_eaten[0]



Recap: Traversing an Array

```
int chocolates_eaten[7] = {4, 2, 5, 2, 0, 3, 1};  
int i = 0;  
while (i < 7) {  
    printf("%d ", chocolates_eaten[i]);  
    i++;  
}
```

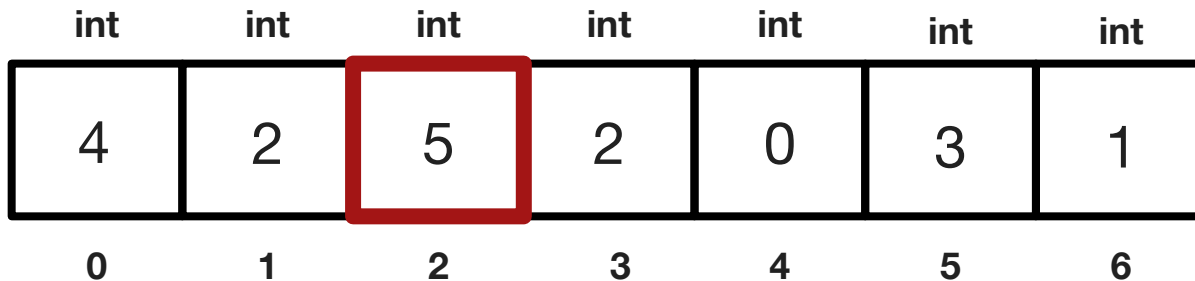
Increment index by 1
`chocolates_eaten[1]`



Recap: Traversing an Array

```
int chocolates_eaten[7] = {4, 2, 5, 2, 0, 3, 1};  
int i = 0;  
while (i < 7) {  
    printf("%d ", chocolates_eaten[i]);  
    i++;  
}
```

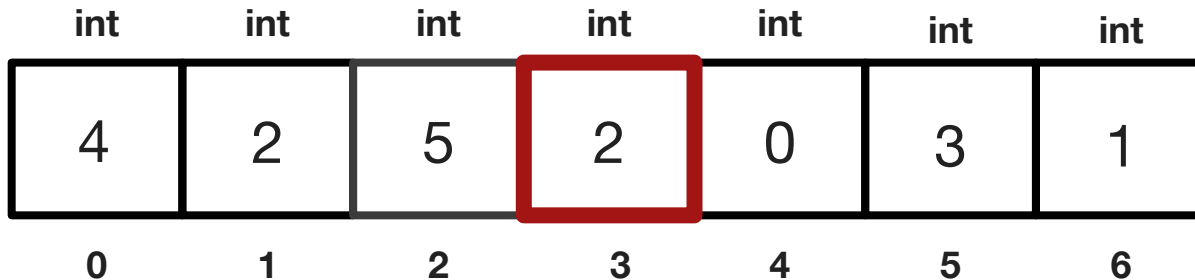
Increment index by 1
chocolates_eaten[2]



Recap: Traversing an Array

```
int chocolates_eaten[7] = {4, 2, 5, 2, 0, 3, 1};  
int i = 0;  
while (i < 7) {  
    printf("%d ", chocolates_eaten[i]);  
    i++;  
}
```

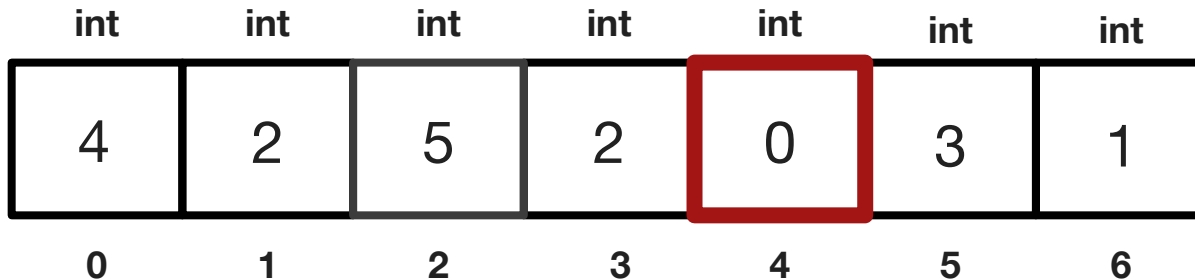
Increment index by 1
`chocolates_eaten[3]`



Recap: Traversing an Array

```
int chocolates_eaten[7] = {4, 2, 5, 2, 0, 3, 1};  
int i = 0;  
while (i < 7) {  
    printf("%d ", chocolates_eaten[i]);  
    i++;  
}
```

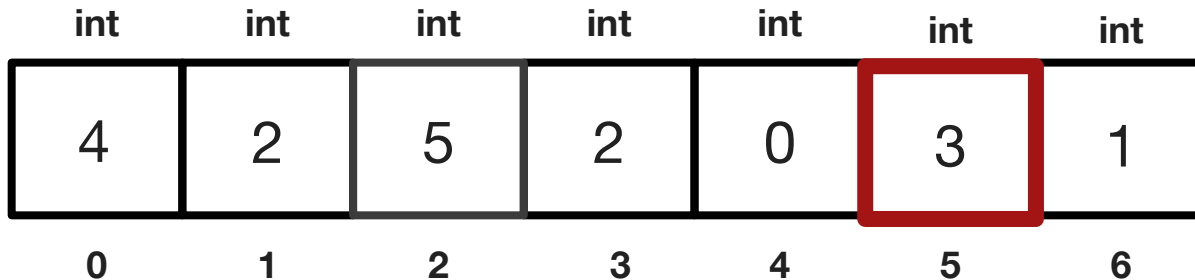
Increment index by 1
`chocolates_eaten[4]`



Recap: Traversing an Array

```
int chocolates_eaten[7] = {4, 2, 5, 2, 0, 3, 1};  
int i = 0;  
while (i < 7) {  
    printf("%d ", chocolates_eaten[i]);  
    i++;  
}
```

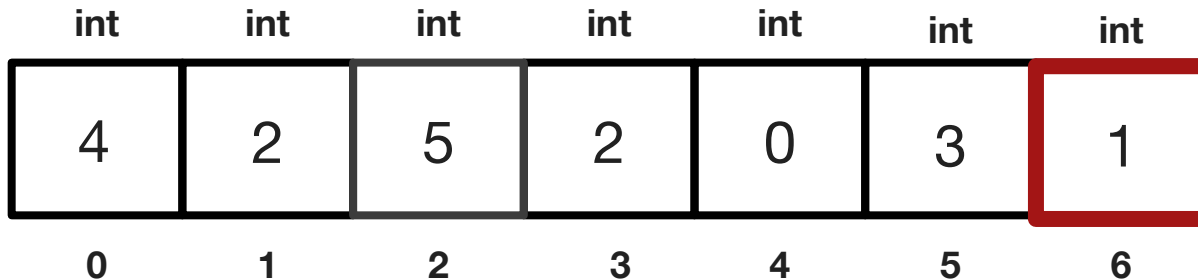
Increment index by 1
chocolates_eaten[5]



Recap: Traversing an Array

```
int chocolates_eaten[7] = {4, 2, 5, 2, 0, 3, 1};  
int i = 0;  
while (i < 7) {  
    printf("%d ", chocolates_eaten[i]);  
    i++;  
}
```

Increment index by 1
chocolates_eaten[6]



Recap: Arrays Exercise

- game_scores.c
- Write code to
 - Print values 10 or more
 - Print in reverse

Arrays and Functions

- We can pass arrays into functions!
- The function needs a way of knowing the size of the array

```
// Can pass in array of int of any size  
void print_array(int size, int array[]);
```

Arrays and Functions

```
void print_array(int size, int array[]);  
  
int main(void) {  
    int marks[] = {9, 8, 10, 2, 7};  
    int ages[] = {21, 42, 11};  
    print_array(5, marks);  
    print_array(3, ages);  
    return 0;  
}  
  
void print_array(int size, int array[]) {  
    for (int i = 0; i < size; i++) {  
        printf("%d ", array[i]);  
    }  
}
```

Arrays and Functions

- Functions do not get a copy of the array values passed into them.
- They can access the original array from the calling function
- This means **they can modify the values** directly from the function
- More about this in future weeks!

Arrays and Functions

- `array` is not a copy of the `marks` array.
- The function can actually modify the values in the original `marks` array that the main passed in

```
int main(void) {  
    int marks[SIZE];  
    scan_marks(SIZE, marks);  
    print_marks(SIZE, marks);  
    return 0;  
}  
  
void scan_marks(int size, int array[]) {  
    for (int i = 0; i < size; i++) {  
        scanf("%d", &array[i]);  
    }  
}
```

Arrays and Functions

- This will not compile.
- You cannot return an array like this from a function.
- You need to pass in the array then modify the data.
- We will learn more about why and see other alternatives later in the course

```
// You can't return an array like
// this from a function
int[] scan_marks(void) {
    int array[SIZE];
    for (int i = 0; i < SIZE; i++) {
        scanf("%d", &array[i]);
    }
    return array;
}
```

Array Function Coding Exercises

- `game_scores_functions.c`
- Write functions to
 - print an array
 - read data into an array
 - scale data within a range

Can you have an array of structs?

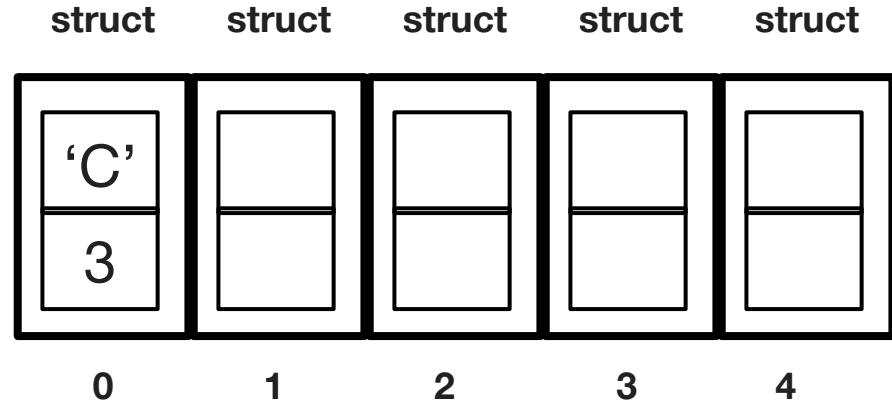
Arrays of structs

- Yes!
- We can have arrays of structs of any type we like

```
struct pokemon pokedex[MAX_POKEMONS];  
struct student comp1511_1911_students[MAX_CLASS];
```

Arrays of structs

```
struct tile {  
    char letter;  
    int score;  
};  
  
// Declare an array of  
// type struct tile of  
// size 5  
  
struct tile tiles[5];  
tiles[0].letter = 'C';  
tiles[0].score = 3;
```

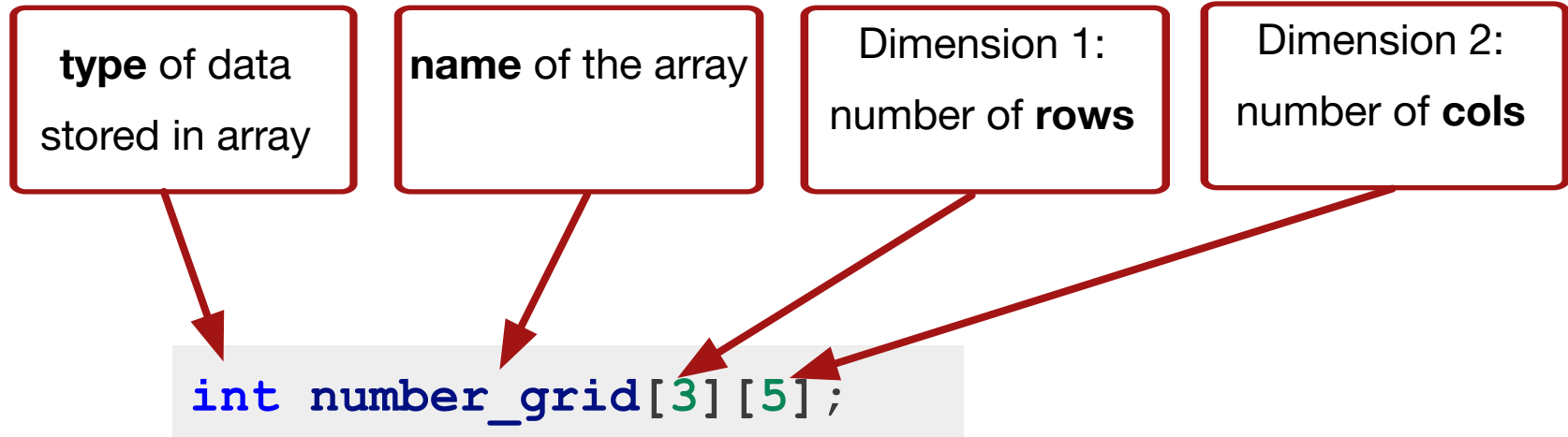


Code Demo of Array of structs

- `struct_tile_array.c`
 - Read in data for an array of tiles
 - Print out the array of tiles
 - Search for a tile with a given letter

2D Arrays

2D Arrays: Declaring



This declares a 2D array (an array of arrays) called `number_grid` that can store 3 rows with 5 columns of `ints` in each row

2D Arrays: Accessing Indexes

```
// A 2D array with 3 rows and 5 columns of int  
int number_grid[3][5];  
// To access an element you need to give 2 indexes  
number_grid[2][3] = 42;
```

	col 0	col 1	col 2	col 3	col 4
row 0					
row 1					
row 2				42	

2D Arrays: Declaring and Initialising

```
// A 2D array with 3 rows and 5 columns of int  
int number_grid[3][5] = {{2, 4, 6, 8, 10},  
                           {1, 2, 3, 4, 5},  
                           {9, 7, 0, 8, -1}};
```

	col 0	col 1	col 2	col 3	col 4
row 0	2	4	6	8	10
row 1	1	2	3	4	5
row 2	9	7	0	8	-1

2D Arrays: Nested While Loops

Think back to this type of code we wrote with nested while loops.

```
int row = 0;
while (row < SIZE) {
    int col = 0;
    while (col < SIZE) {
        printf("*");
        col++;
    }
    printf("\n");
    row++;
}
```

2D Arrays: Traversal

```
// Assume ROWS is 3 and COLS is 4
int array[ROWS][COLS] = {{1, 2, 3, 1},
                          {9, 8, 7, 4},
                          {5, 0, 6, 3}};

int row = 0;
while (row < ROWS) {
    int col = 0;
    while (col < COLS) {
        printf("%d ", array[row][col]);
        col++;
    }
    printf("\n");
    row++;
}
```

Outer loop: row = 0
Inner loop: col = 0

	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

2D Arrays: Traversal

```
// Assume ROWS is 3 and COLS is 4
int array[ROWS][COLS] = {{1, 2, 3, 1},
                          {9, 8, 7, 4},
                          {5, 0, 6, 3}};

int row = 0;
while (row < ROWS) {
    int col = 0;
    while (col < COLS) {
        printf("%d ", array[row][col]);
        col++;
    }
    printf("\n");
    row++;
}
```

Outer loop: row = 0
Inner loop: col = 1

	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

2D Arrays: Traversal

```
// Assume ROWS is 3 and COLS is 4
int array[ROWS][COLS] = {{1, 2, 3, 1},
                          {9, 8, 7, 4},
                          {5, 0, 6, 3}};

int row = 0;
while (row < ROWS) {
    int col = 0;
    while (col < COLS) {
        printf("%d ", array[row][col]);
        col++;
    }
    printf("\n");
    row++;
}
```

Outer loop: row = 0
Inner loop: col = 2

	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

2D Arrays: Traversal

```
// Assume ROWS is 3 and COLS is 4
int array[ROWS][COLS] = {{1, 2, 3, 1},
                          {9, 8, 7, 4},
                          {5, 0, 6, 3}};

int row = 0;
while (row < ROWS) {
    int col = 0;
    while (col < COLS) {
        printf("%d ", array[row][col]);
        col++;
    }
    printf("\n");
    row++;
}
```

Outer loop: row = 0

Inner loop: col = 3

	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

2D Arrays: Traversal

```
// Assume ROWS is 3 and COLS is 4
int array[ROWS][COLS] = {{1, 2, 3, 1},
                          {9, 8, 7, 4},
                          {5, 0, 6, 3}};

int row = 0;
while (row < ROWS) {
    int col = 0;
    while (col < COLS) {
        printf("%d ", array[row][col]);
        col++;
    }
    printf("\n");
    row++;
}
```

Outer loop: row = 1
Inner loop: col = 0

	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

2D Arrays: Traversal

```
// Assume ROWS is 3 and COLS is 4
int array[ROWS][COLS] = {{1, 2, 3, 1},
                          {9, 8, 7, 4},
                          {5, 0, 6, 3}};

int row = 0;
while (row < ROWS) {
    int col = 0;
    while (col < COLS) {
        printf("%d ", array[row][col]);
        col++;
    }
    printf("\n");
    row++;
}
```

Outer loop: row = 1
Inner loop: col = 1

	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

2D Arrays: Traversal

```
// Assume ROWS is 3 and COLS is 4
int array[ROWS][COLS] = {{1, 2, 3, 1},
                          {9, 8, 7, 4},
                          {5, 0, 6, 3}};

int row = 0;
while (row < ROWS) {
    int col = 0;
    while (col < COLS) {
        printf("%d ", array[row][col]);
        col++;
    }
    printf("\n");
    row++;
}
```

Outer loop: row = 1
Inner loop: col = 2

	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

2D Arrays: Traversal

```
// Assume ROWS is 3 and COLS is 4
int array[ROWS][COLS] = {{1, 2, 3, 1},
                          {9, 8, 7, 4},
                          {5, 0, 6, 3}};

int row = 0;
while (row < ROWS) {
    int col = 0;
    while (col < COLS) {
        printf("%d ", array[row][col]);
        col++;
    }
    printf("\n");
    row++;
}
```

Outer loop: row = 1
Inner loop: col = 3

	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

2D Arrays: Traversal

```
// Assume ROWS is 3 and COLS is 4
int array[ROWS][COLS] = {{1, 2, 3, 1},
                          {9, 8, 7, 4},
                          {5, 0, 6, 3}};

int row = 0;
while (row < ROWS) {
    int col = 0;
    while (col < COLS) {
        printf("%d ", array[row][col]);
        col++;
    }
    printf("\n");
    row++;
}
```

Outer loop: row = **2**
Inner loop: col = 0

	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

2D Arrays: Traversal

```
// Assume ROWS is 3 and COLS is 4
int array[ROWS][COLS] = {{1, 2, 3, 1},
                          {9, 8, 7, 4},
                          {5, 0, 6, 3}};

int row = 0;
while (row < ROWS) {
    int col = 0;
    while (col < COLS) {
        printf("%d ", array[row][col]);
        col++;
    }
    printf("\n");
    row++;
}
```

Outer loop: row = **2**
Inner loop: col = 1

	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

2D Arrays: Traversal

```
// Assume ROWS is 3 and COLS is 4
int array[ROWS][COLS] = {{1, 2, 3, 1},
                          {9, 8, 7, 4},
                          {5, 0, 6, 3}};

int row = 0;
while (row < ROWS) {
    int col = 0;
    while (col < COLS) {
        printf("%d ", array[row][col]);
        col++;
    }
    printf("\n");
    row++;
}
```

Outer loop: row = **2**
Inner loop: col = 2

	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

2D Arrays: Traversal

```
// Assume ROWS is 3 and COLS is 4
int array[ROWS][COLS] = {{1, 2, 3, 1},
                          {9, 8, 7, 4},
                          {5, 0, 6, 3}};

int row = 0;
while (row < ROWS) {
    int col = 0;
    while (col < COLS) {
        printf("%d ", array[row][col]);
        col++;
    }
    printf("\n");
    row++;
}
```

Outer loop: row = **2**
Inner loop: col = 3

	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

2D Arrays as Function Parameters

When passing a 2D array to a function, the number of columns must be specified in the array parameter.

These prototypes are fine:

```
void print_array(int rows, int array[][COLS]);  
void print_array(int array[ROWS][COLS]);
```

This is not:

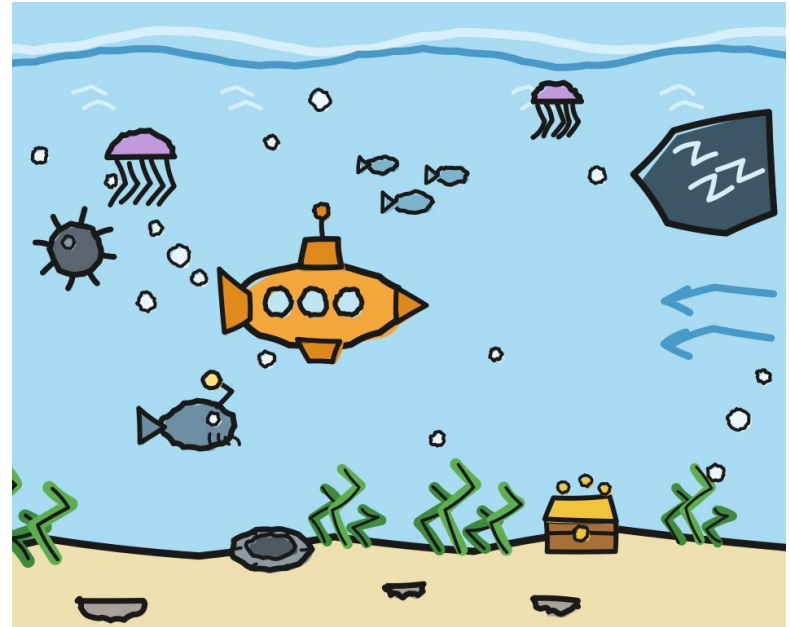
```
void print_array(int rows, int cols, int array[][]);
```

Code Demo and Exercise

- 2D_array_numbers.c
 - print array
 - read in data
 - sum data
 - print sum of each row
 - print sum of each column
- diagonals.c
 - sum diagonal starting at top left
 - sum diagonal starting at top right

Assignment 1 CS Submarine is released

- Due: **Mon Wk7 @ 6PM**
- It is an individual assignment
- Aims of the assignment
 - Solve a larger problem
 - Use arrays and 2D arrays
 - Break down code into functions
 - Practice debugging
 - Practice coding with good style
 - 20% of your mark
 - Have fun learning!!!!



Assignment 1 Stages

- Broken into stages
 - Stages 1, 2, 3 and 4
- Stage 4:
 - Is a challenge stage.
 - Not everyone will finish stage 4.
 - You can get great marks without completing this stage
 - You are on track and doing well in the course even if you don't get up to this stage

Assignment 1 Getting Started

- Read spec and watch video
- Download starter code
- Run reference implementation to see what the finished product should do!!!
- Implement and test each substage, one at a time, give regularly
- If stuck, get help from
 - tutors
 - forum
 - help sessions

Feedback Please!

Your feedback is valuable!

If you have any feedback from today's lecture, please follow the link below or use the QR Code.

Please remember to keep your feedback constructive, so I can action it and improve your learning experience.



<https://forms.office.com/r/ED8H2Fbs9R>

What did we learn today?

- Recap arrays (game_scores.c)
- Arrays and Functions (game_score_functions.c)
- Arrays of structs (struct_array.c)
- 2D Arrays (2D_array.c, diagonals.c)

Next Lecture

- 2D array recap
- strings
 - We will finally be able to store text in our variables!!!!
- arrays of strings

Reach Out

Content Related Questions:

[Course Forum](#)

Admin related Questions email:

cs1511@unsw.edu.au



Student Support - I Need Help With...

My Feelings and Mental Health

Feeling depressed, overwhelmed
or not your usual self?



Mental Health Connect
student.unsw.edu.au/mhc



TalkCampus
student.unsw.edu.au/Talkcampus



UNSW (24/7) Mental Health Support: [+61\(2\) 9385 5418](tel:+61293855418)

Text support After Hours: [0485 826 595](tel:0485826595)



Offshore
Call 24-hour Medibank Hotline: [+61 2 8905 0307](tel:+61289050307)

Uni and Life in Australia

Stress, financial, visas,
Accommodation & More



Student Support
Indigenous Student Support

student.unsw.edu.au/advisors
nura-gili-centre-indigenous-programs

Reporting Sexual Assault/Harassment



Equity Diversity & Inclusion (EDI)

edi.unsw.edu.au/sexual-misconduct

Educational Adjustments

To manage my studies and
disability / health condition



Equitable Learning Services (ELS)

student.unsw.edu.au/els

Academic and Study Skills



Academic Skills

student.unsw.edu.au/skills

Special Consideration

Because life impacts our
studies and exams



Special Consideration

student.unsw.edu.au/special-consideration

Physical Health

Doctors visits, Vaccinations



Health Service

student.unsw.edu.au/hsu