COMP1511/1911 Programming Fundamentals

Week 4 Lecture 1

Arrays of structs

2D Arrays

Last Week

- Functions
- Style
- Arrays

Census Date this sunday

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

Assignment 1 : CS Moonlander

- Assignment 1 will be released after this lecture at 1pm
- It is an individual assignment
- Aims of the assignment
  - Apply arrays and two-dimensional arrays to problem solving
  - Apply the use of functions in code
  - Practice skills in debugging code, and skills in patience as you search for your missing semicolons
  - Apply good style
    - You will be assessed on style! 20% of your mark

Assignment 1 : Stages

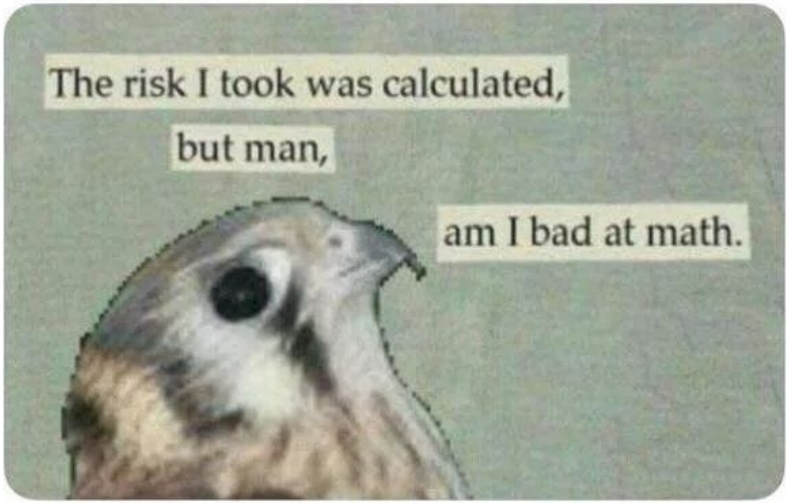
- 4 stages, each stage ramps up with difficulty
  - just like the lab exercises
  - suggest going through the stages in order
- Stage 4 is similar level of difficulty as 3 dot lab exercises
  - You can get a great mark in the assignment without doing stage 4
  - Stage 4 is a challenge and to discriminate between HDs

Assignment 1 Live Stream

Time: Tuesday 12:30pm
Place: Assn 1 Overview (youtube.com)
Recording will also be available

Assignment Due Date: Monday Week 7 5pm!
Don't leave it until the last minute!
Help sessions will be very busy the week before the deadline!!!!!!!!

When you convince yourself you can finish your entire assignment the night before its due



Plagiarism

Get help from the right places

- COMP1511/1911 staff in lectures, tuts, labs
- Forum, Help Sessions, Revision Sessions

- Do not get ‘help’ or submit code from external sources like:
  - ChatGPT, external tutors, other people’s code etc

Plagiarism

Plagiarism is taken very seriously

We run plagiarism checking on all submissions

For full details read the course outline and also refer to

student.unsw.edu.au/plagiarism

Revision Sessions (Not recorded)

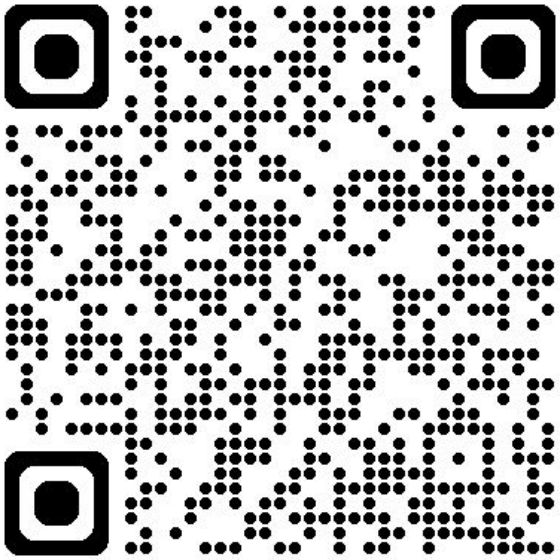
- Topics: Vote now for the topics!
- Week 4:
    - Mon 9-11am Kora lab in person: how was it?
    - Wed 12-2pm and Fri 12-2pm Online
- You can **sign up** for the revision sessions by following the link Select tickets – COMP1511 Revision Sessions – UNSW or Online (tickettailor.com)
- You can find this link and the voting on this forum announcement

Today's Lecture

- Recap arrays and functions with arrays
- Array of structs
- 2D Arrays

Link to Week 4 Live Lecture Code

https://cgi.cse.unsw.edu.au/~cs1511/24T3/live/week_4/



Disclaimer:

Sometimes live lecture code is not cleaned up and polished!!! It may have some things that are not 100% perfect style.

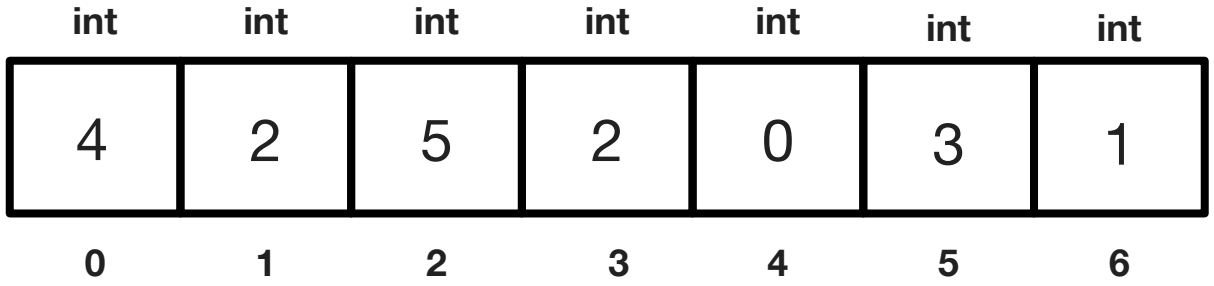
I also sometimes have extra comments explaining how C works that would not be needed usually.

Visualising an Array

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

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

This is what it looks like visually:

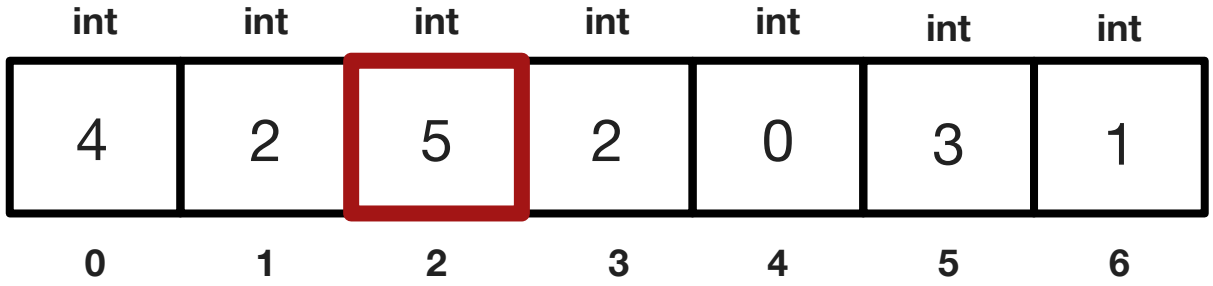


Note: The array holds 7 elements. Indexes start at 0

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



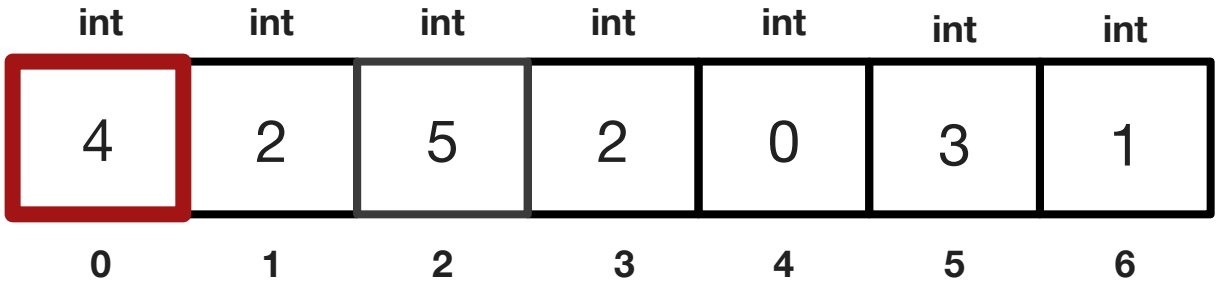
`chocolate_eating[2]` would access the third element

Traversing an Array

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

Start at index 0

`chocolate_eating[0]`

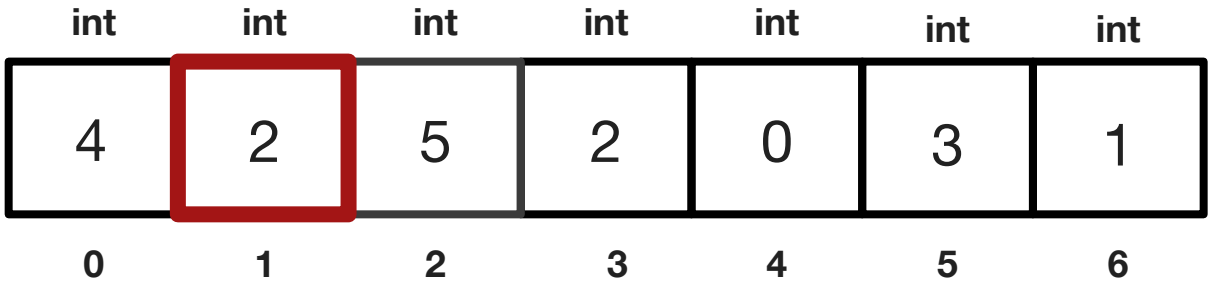


Traversing an Array

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

Increment index by 1

`chocolate_eating[1]`

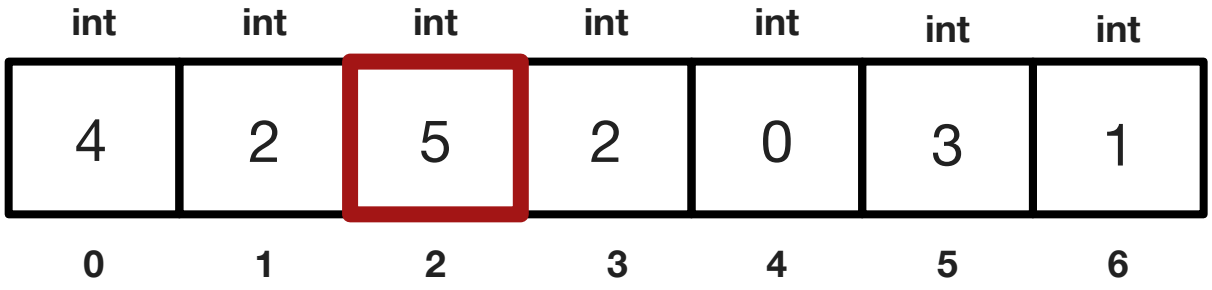


Traversing an Array

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

Increment index by 1

`chocolate_eating[2]`

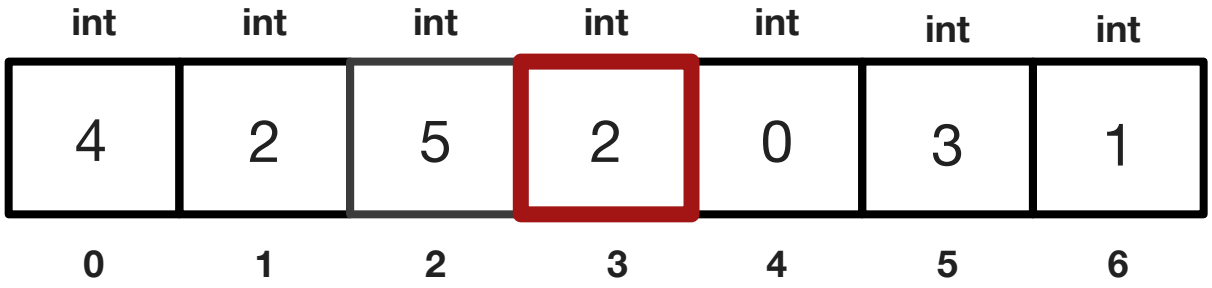


Traversing an Array

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

Increment index by 1

`chocolate_eating[3]`

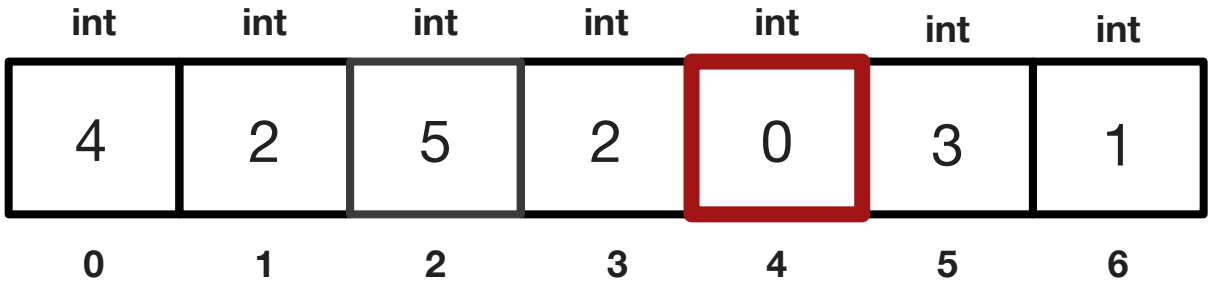


Traversing an Array

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

Increment index by 1

`chocolate_eating[4]`

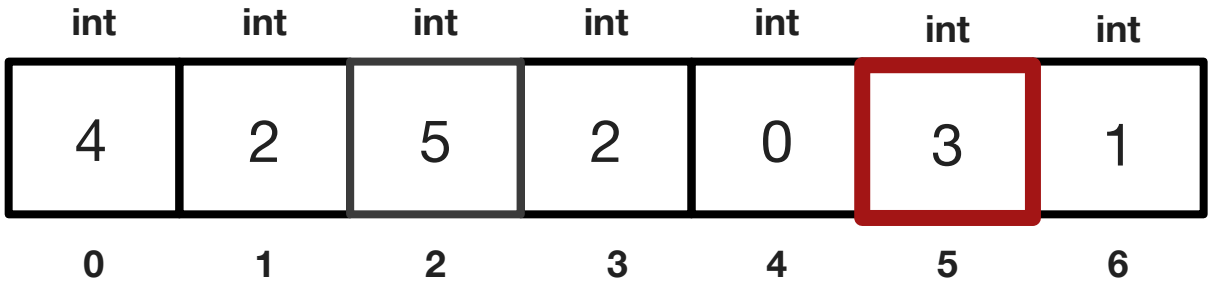


Traversing an Array

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

Increment index by 1

`chocolate_eating[5]`

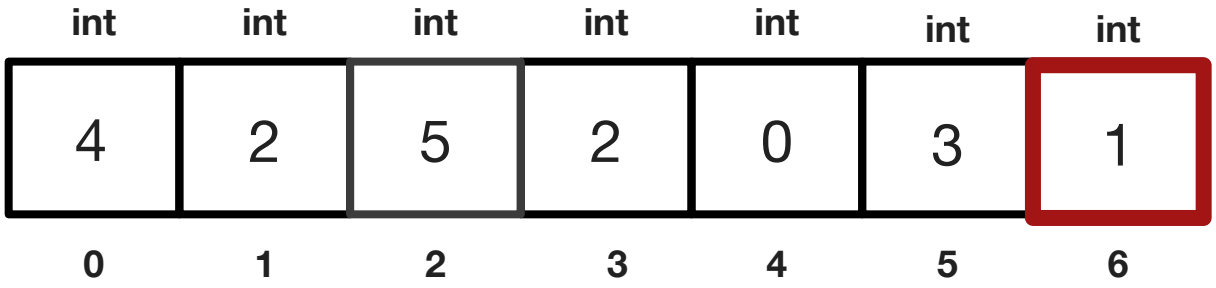


Traversing an Array

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

Increment index by 1

`chocolate_eating[6]`



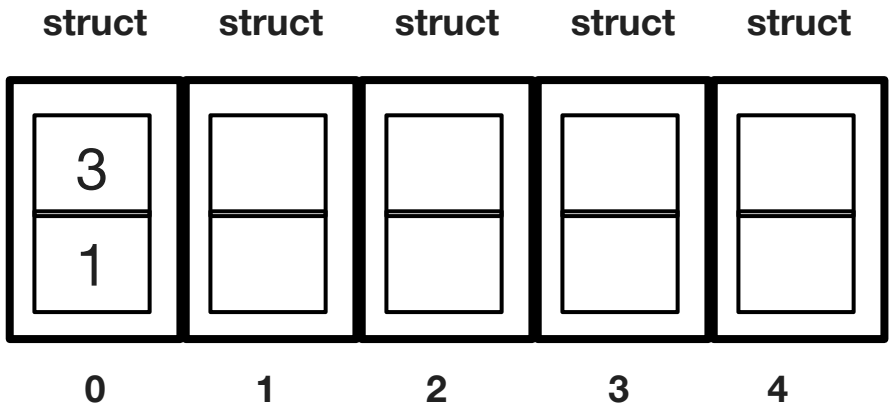
Array Coding Exercises

- Read in and print array with functions
- Write a function to print out odd numbers in the array as a function
- Find the maximum in an array as a function
- What test cases should we choose?

Can you have an array of structs?

Arrays of structs

```
struct coordinate {
    int x;
    int y;
};
// Declare an array of
// type struct coordinate
// of size 5
struct coordinate map[5];
map[0].x = 3;
map[0].y = 1;
```

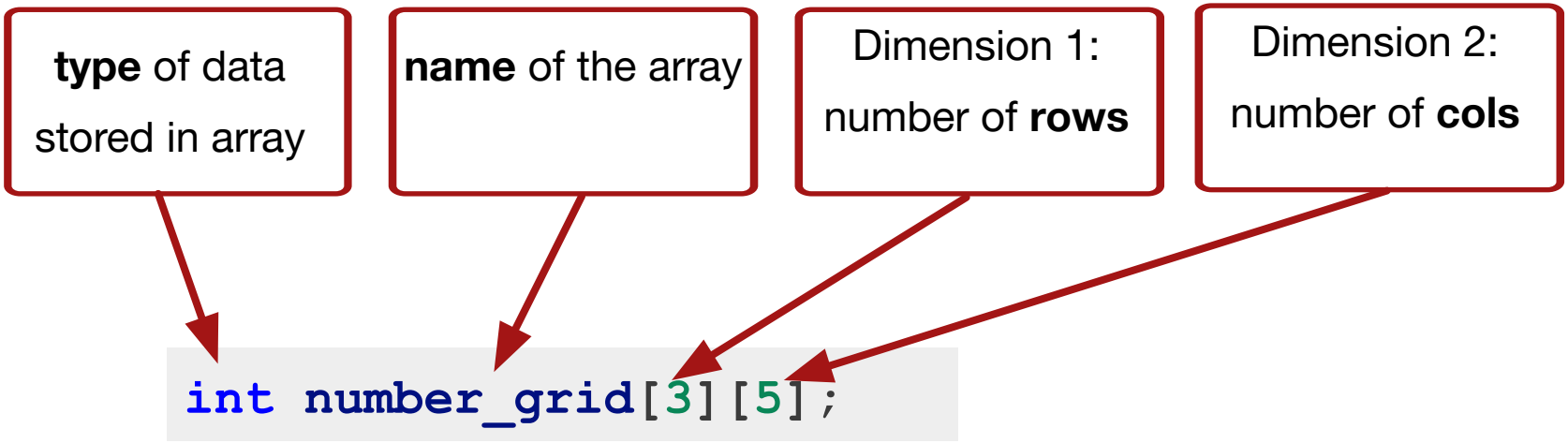


Code Demo of Array of structs

- `coordinate_array.c`
  - Read in data for an array of coordinates
  - Print out the array of coordinates
  - Move all coordinates by a constant value in the x direction

2D Arrays
(Arrays of Arrays)

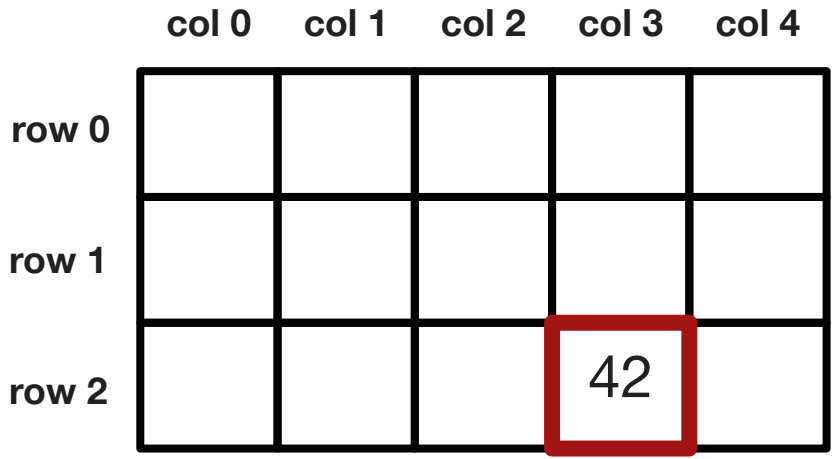
2D Arrays: Declaring



- This declares a 2D array (an array of arrays) called **`number_grid`** that can store **`3`** rows with **`5`** columns of **`int`**s 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;
```



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 the code we wrote with nested while loops that printed out a grid of numbers.

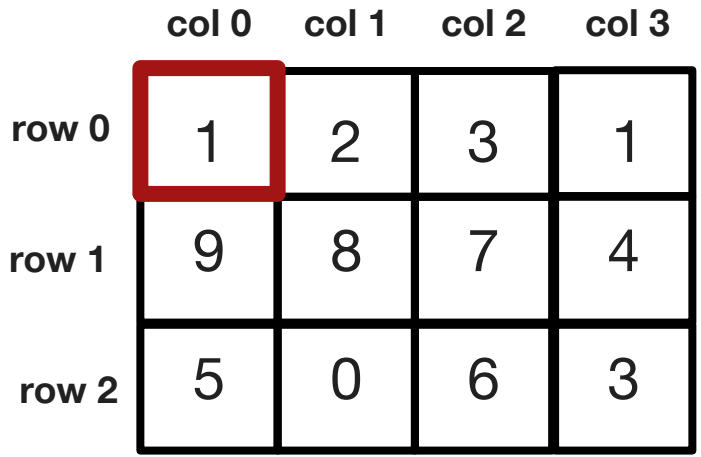
```
int row = 0;
while (row < SIZE) {
    int col = 0;
    while (col < SIZE) {
        printf("%d ", col);
        col++;
    }
    printf("\n");
    row++;
}
```

2D Arrays: Traversal

```c
// 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

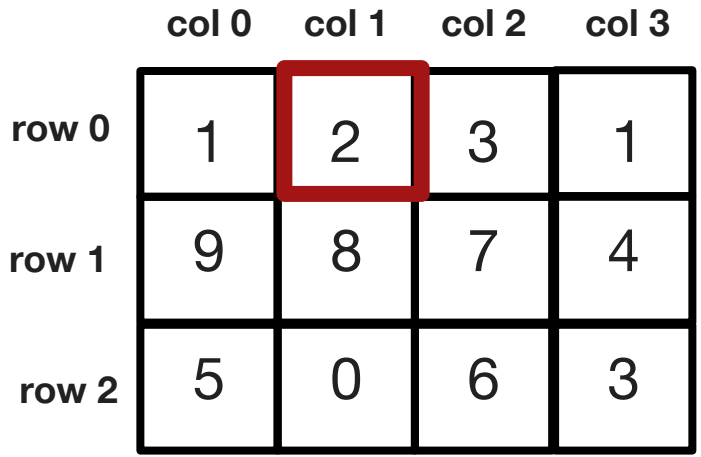


2D Arrays: Traversal

```c
// 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



2D Arrays: Traversal

```c
// 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

```c
// 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

```c
// 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

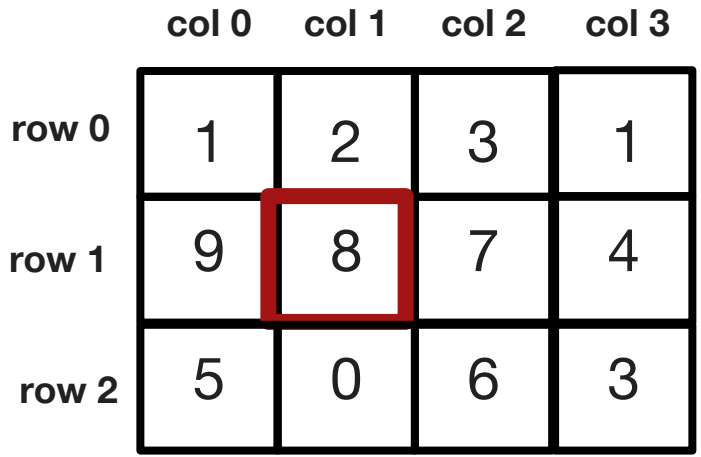
	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

2D Arrays: Traversal

```c
// 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



2D Arrays: Traversal

```c
// 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

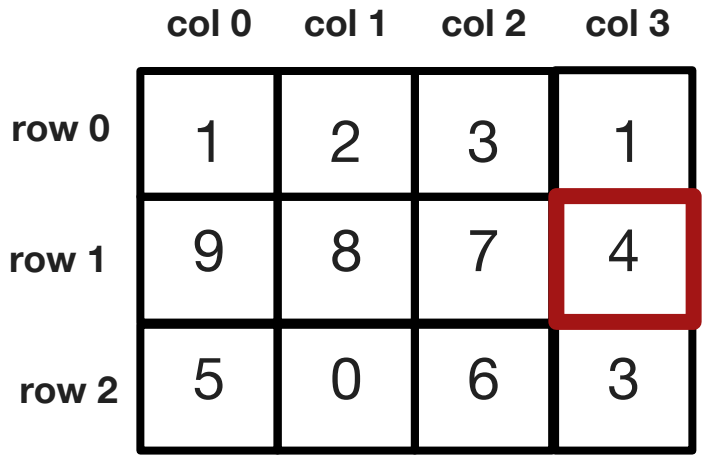
	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

2D Arrays: Traversal

```c
// 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



2D Arrays: Traversal

```c
// 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

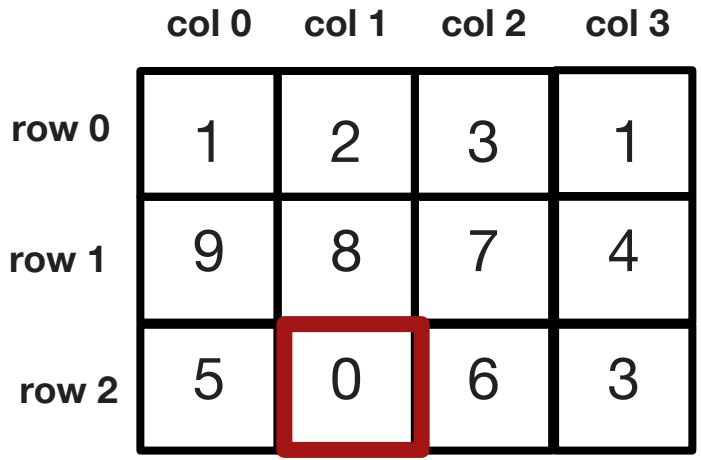
	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

2D Arrays: Traversal

```c
// 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



2D Arrays: Traversal

```c
// 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

```c
// 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

Demo

2D_array_numbers.c

- print array
- read in 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

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/ueCvbJmQCi

What did we learn today?

- Assignment 1 released
- Recap arrays (numbers_functions.c)
- Arrays of structs (struct_array.c)
- 2D Arrays (2D_array.c, diagonals.c)

Next Lecture

- strings
  - We will finally be able to store text in our variables!!!!!

Reach Out

Content Related Questions:
Forum

Admin related Questions email:
cs1511@unsw.edu.au