COMP1511 Programming Fundamentals

Week 4 Lecture 1

Arrays of structs

2D Arrays

Census Date this thursday

Census data is Thursday 13th March 11:59pm
Last day to drop (T1) courses without financial liability.

Assignment 1

- Assignment 1 will be released after this lecture at 1pm
- The assignment will be due **Mon Wk7 @ 5PM**
- 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

Revision Sessions

We will be running two revision classes this week to go over the content that we have learned for the past four weeks and solidify our knowledge.

The revision sessions are:

- **Week 4 Monday 10/03/2025 2PM-4PM - Brass Lab J17 305**
- **Week 4 Thursday 13/03/2025 4PM-6PM - Online**

Please sign up for the revision sessions here.

The access code is "COMP1511", case sensitive and without the quotes.

More info in this post.

Help Sessions

All help sessions held for the term will be on this timetable:

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

They are drop-in. You do not need to book these.

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

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

Link to Week 4 Live Lecture Code

https://cgi.cse.unsw.edu.au/~cs1511/25T1/code/week_4/

Disclaimer:

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

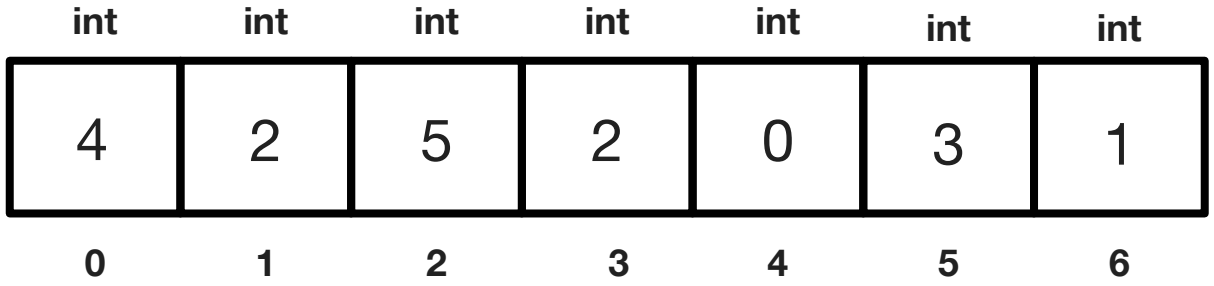
I also sometimes have extra comments explaining how C works that would not be needed usually.

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:

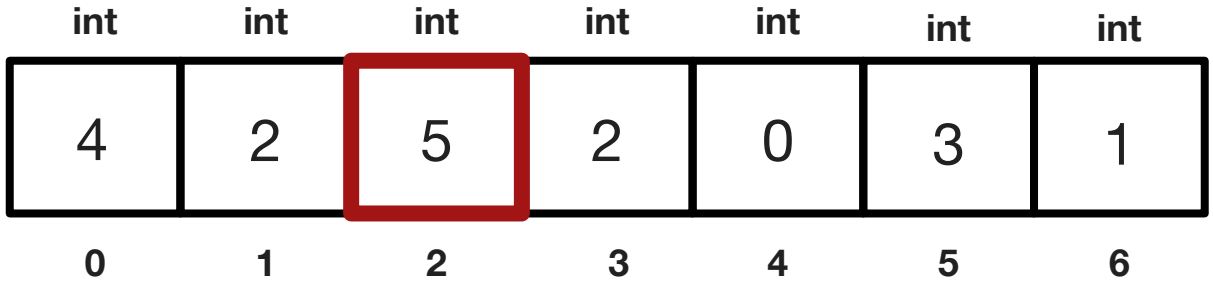


Note: The array holds 7 elements. Indexes start at 0

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};
```



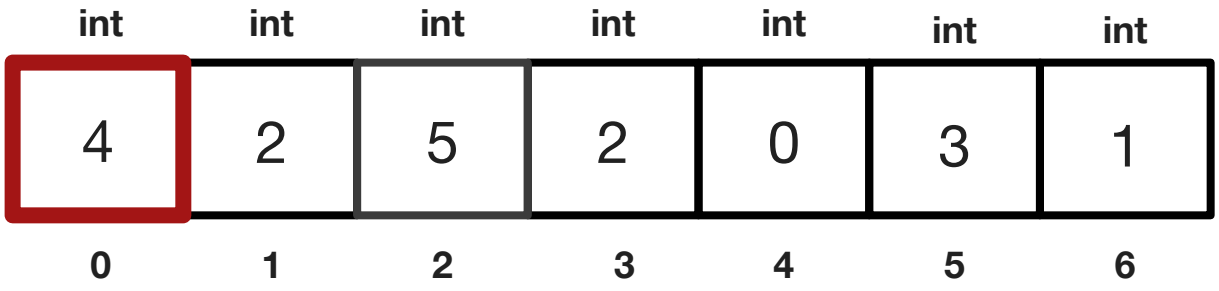
`chocolate_eating[2]` would access the third element

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]`

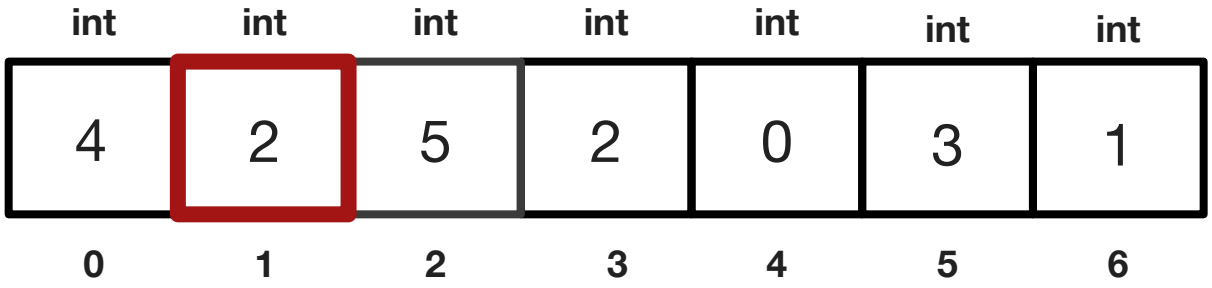


Traversing an Array

```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]`

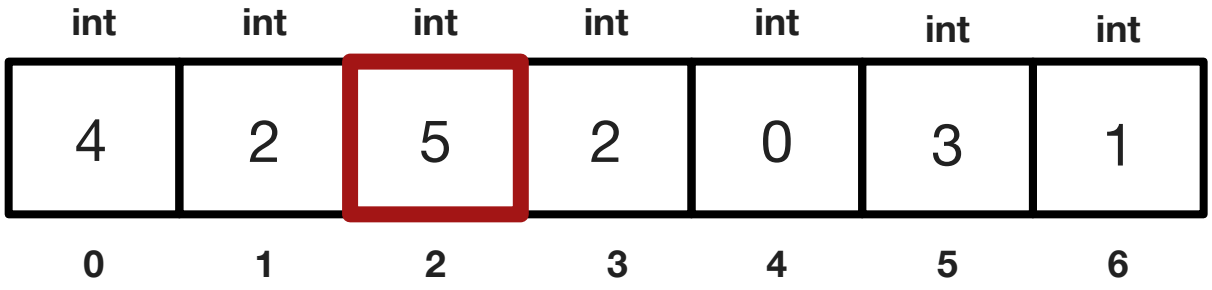


Traversing an Array

```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]`

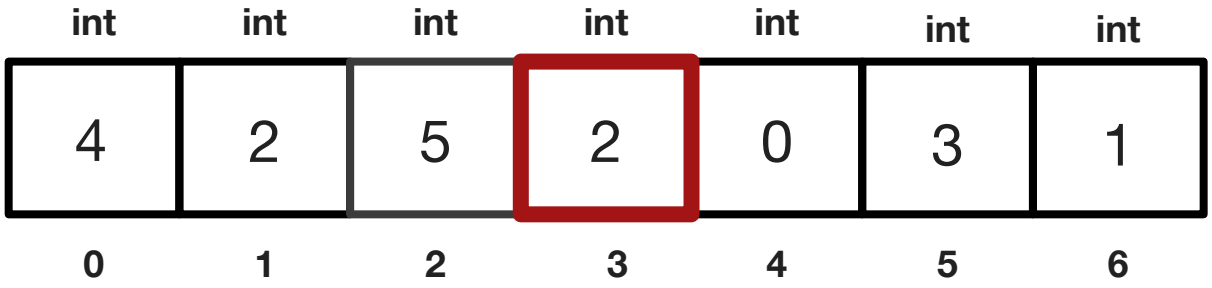


Traversing an Array

```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]`

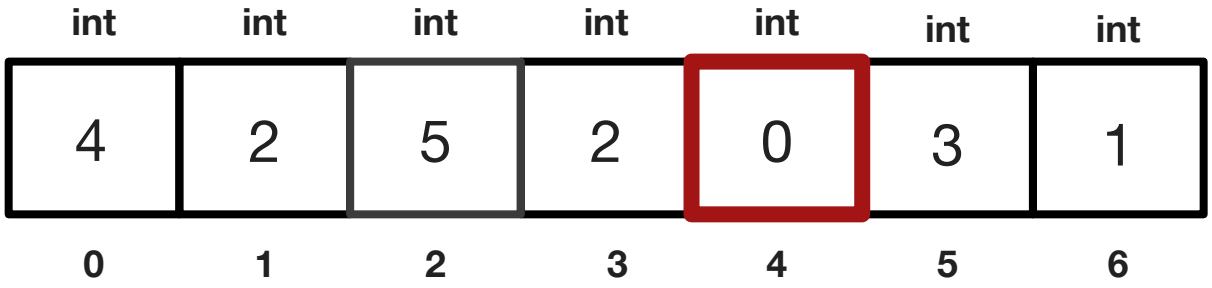


Traversing an Array

```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]`

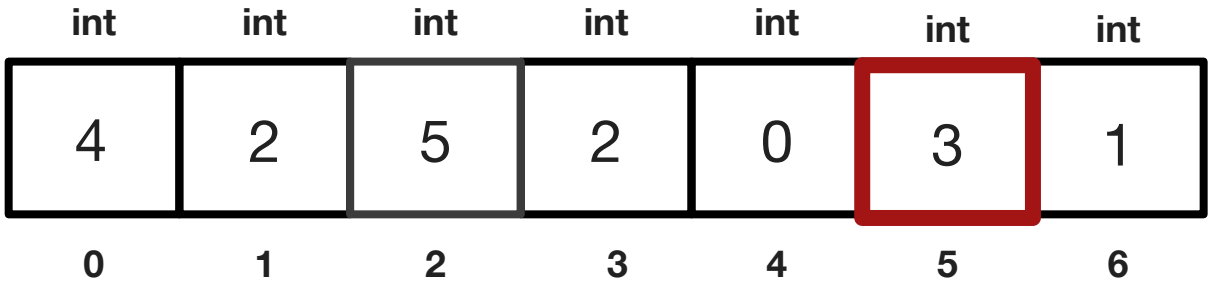


Traversing an Array

```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]`

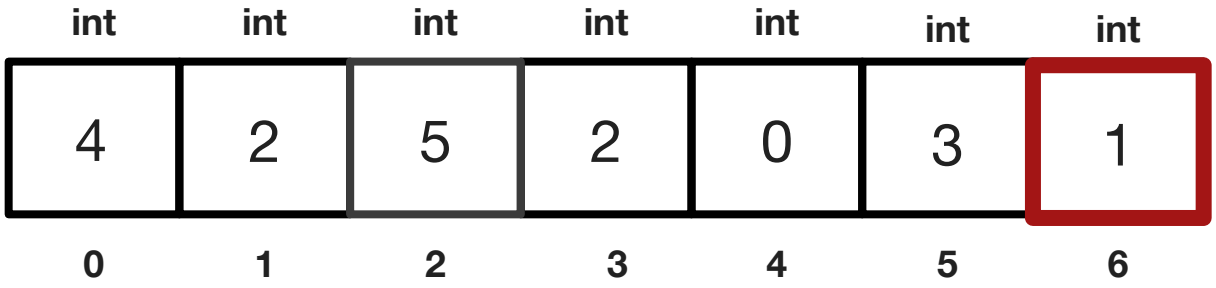


Traversing an Array

```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]`



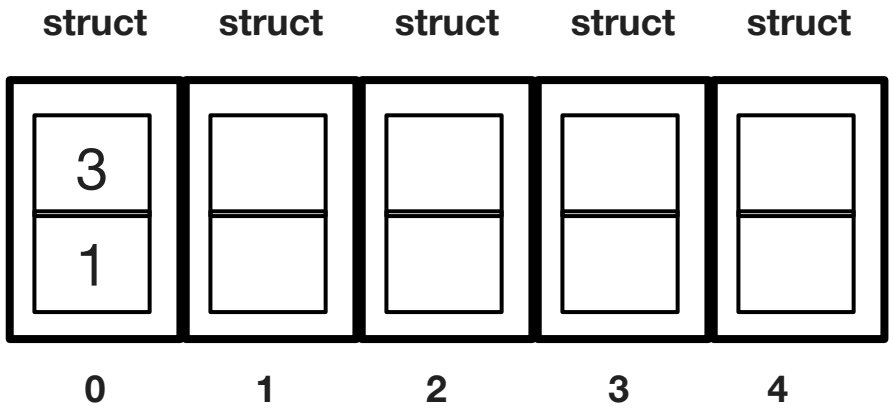
Array Coding Exercises

- 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;
```

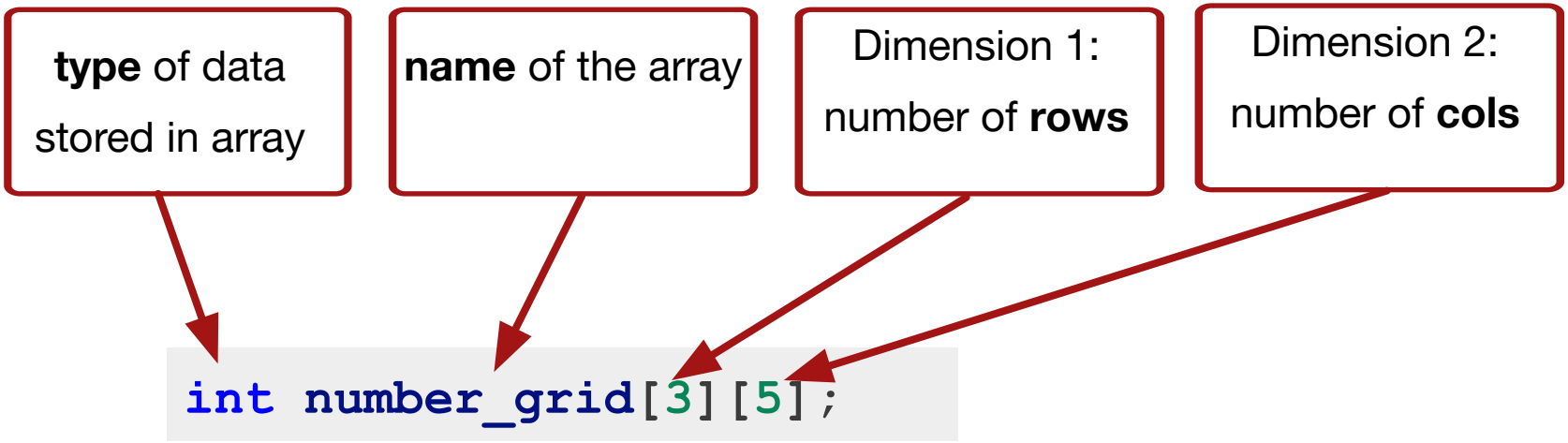


Code Demo of Array of structs

- `struct_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)

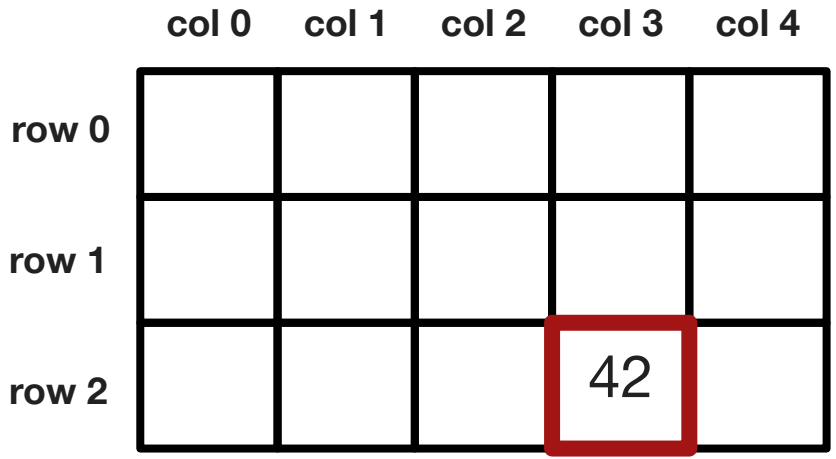
2D Arrays: Declaring



- 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.

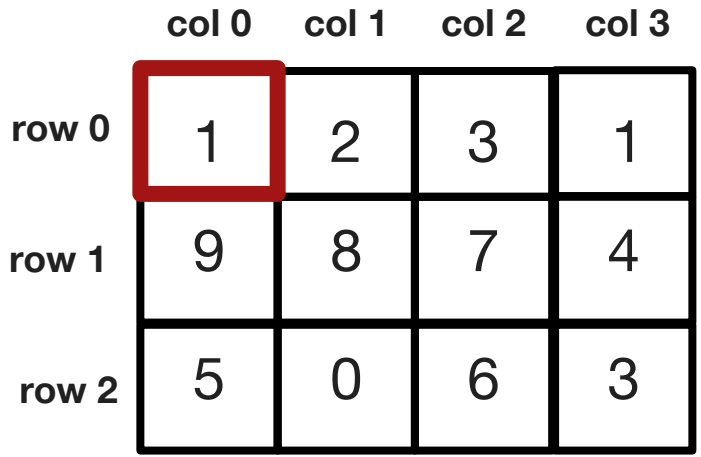
```
int row = 0;
while (row < SIZE) {
    int col = 0;
    while (col < SIZE) {
        printf("%d ", col);
        col++;
    }
    printf("\n");
    row++;
}
```

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

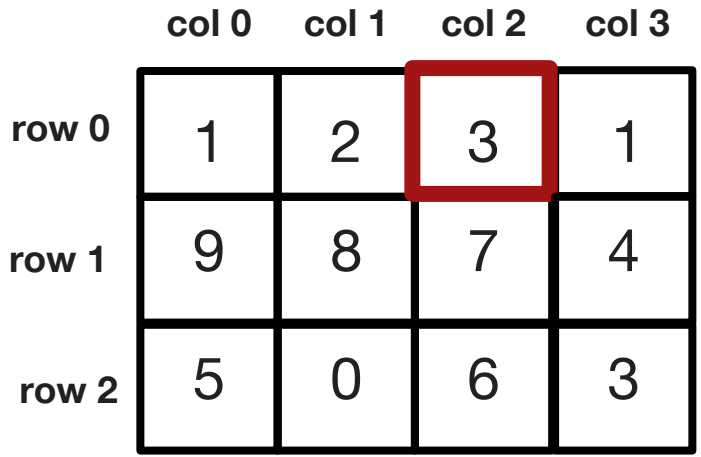
	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

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 = 2



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

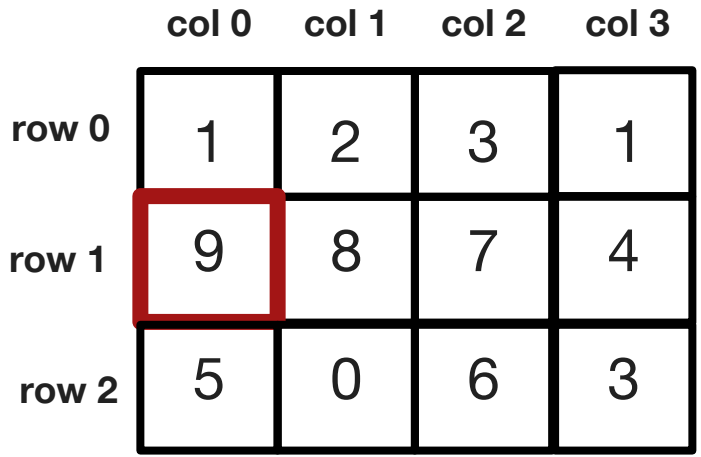
	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

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



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

	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 = 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

	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 = 0

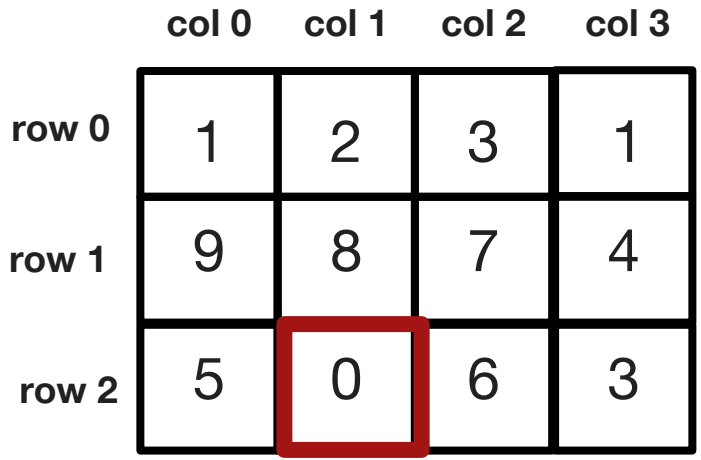
	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

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
  - 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

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 (numbers_functions.c)
- Arrays of structs (struct_array.c)
- 2D Arrays (2D_array.c, diagonals.c)

Assignment 1 should be out now! Go have a look!!!

Next Lecture

- strings
  - We will finally be able to store text in our variables!!!!!
- arrays of strings
- command Line arguments (if time)

Reach Out

Content Related Questions:
Forum

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