COMP1511/1911 Programming Fundamentals

Week 4 Lecture 1

Arrays of structs

2D Arrays

Last Week

- Functions
- Style
- Arrays

We have covered a lot in the course so far!

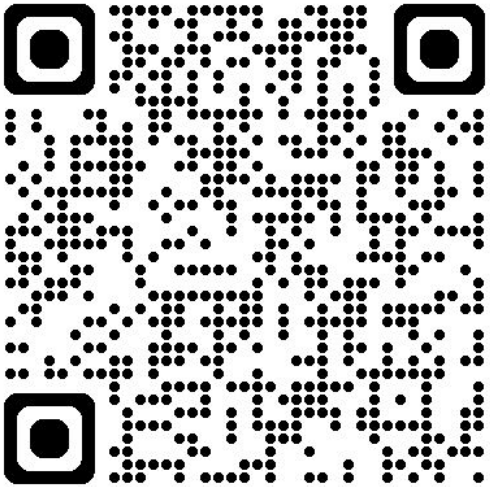
You will get plenty of chances to practice these skills in tutorials and labs and assignments.
We don’t expect you to have mastered them all yet.

Today’s Lecture

- Recap arrays
- Functions with arrays
- Array of structs
- 2D Arrays

Link to Week 4 Live Lecture Code

https://cgi.cse.unsw.edu.au/~cs1511/25T3/code/week_4/

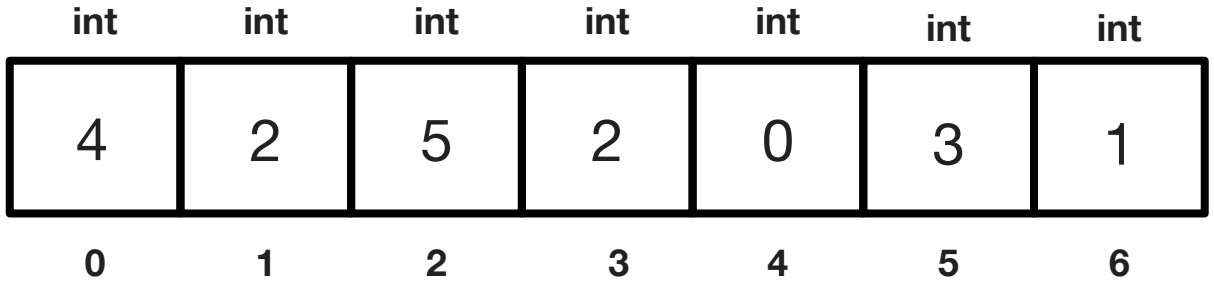


Visualising an Array

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:

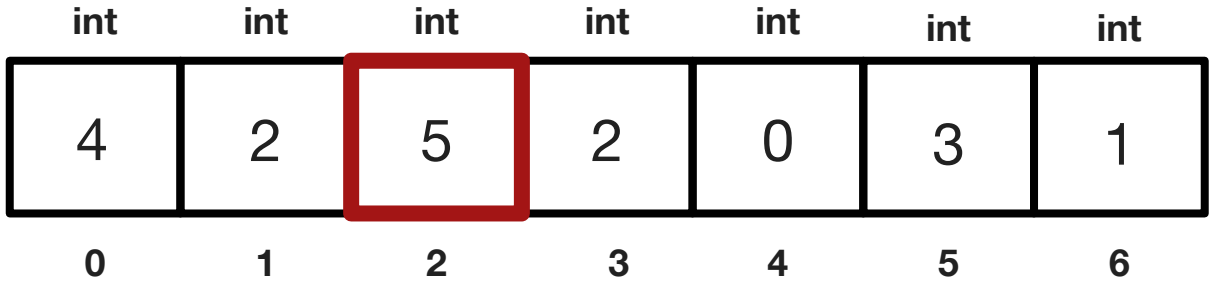


Note: The array holds 7 elements. Indexes start at 0

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};
```

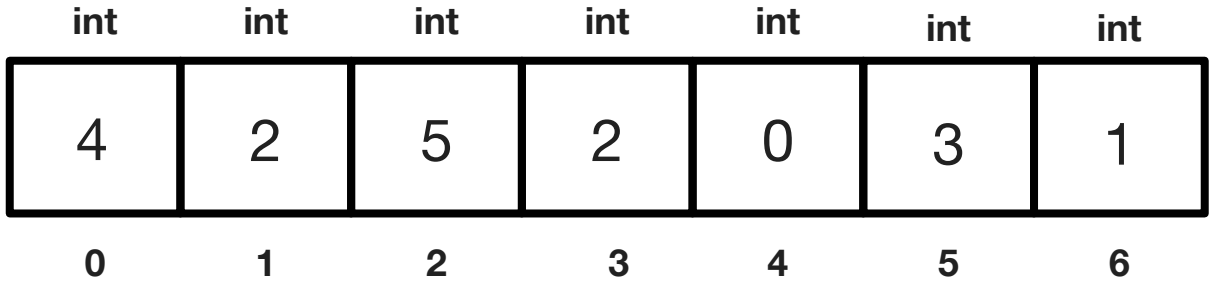


`chocolates_eaten[2]` would access the third element

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};
```



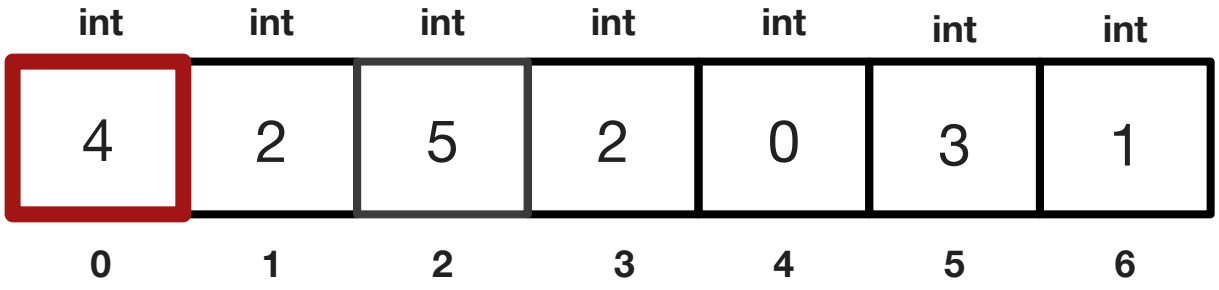
`chocolates_eaten[7]` would cause a run-time error

Traversing an Array

```c
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]`

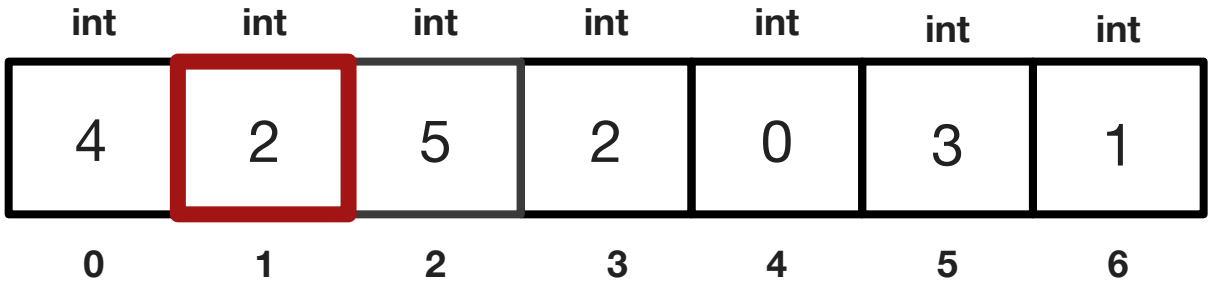


Traversing an Array

```c
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]`

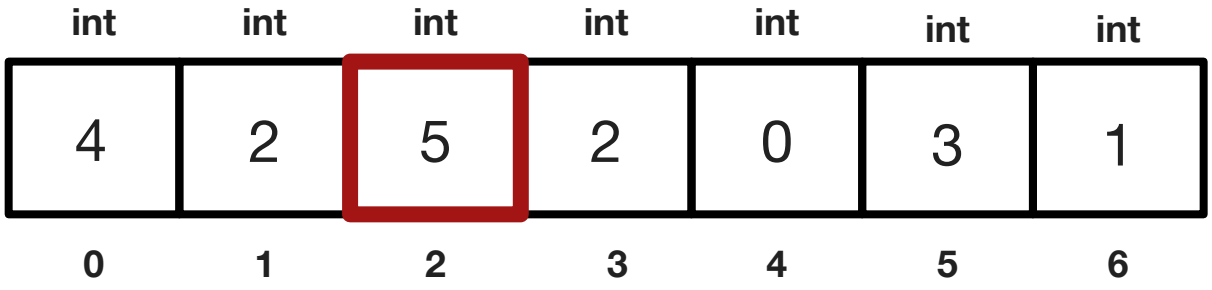


Traversing an Array

```c
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]`

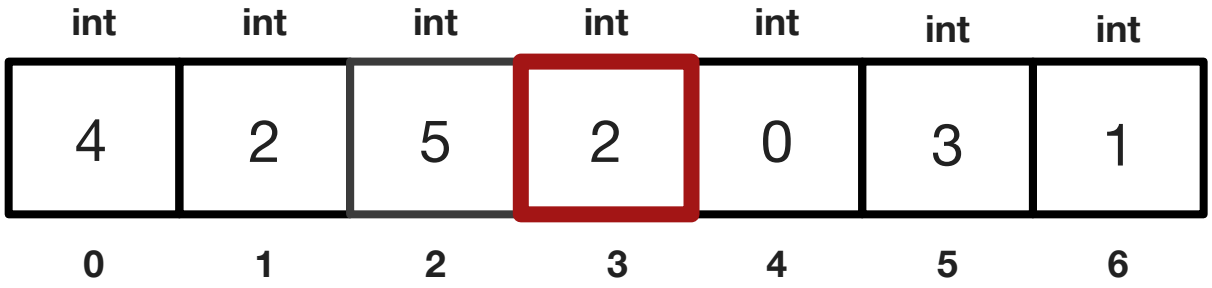


Traversing an Array

```c
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]`

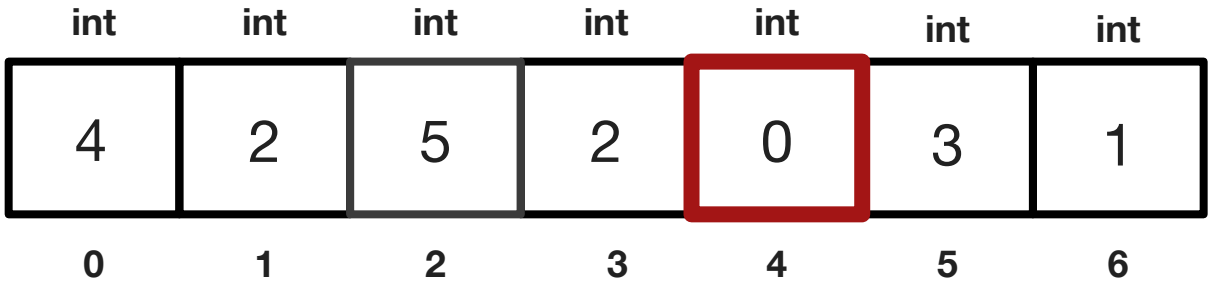


Traversing an Array

```c
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]`

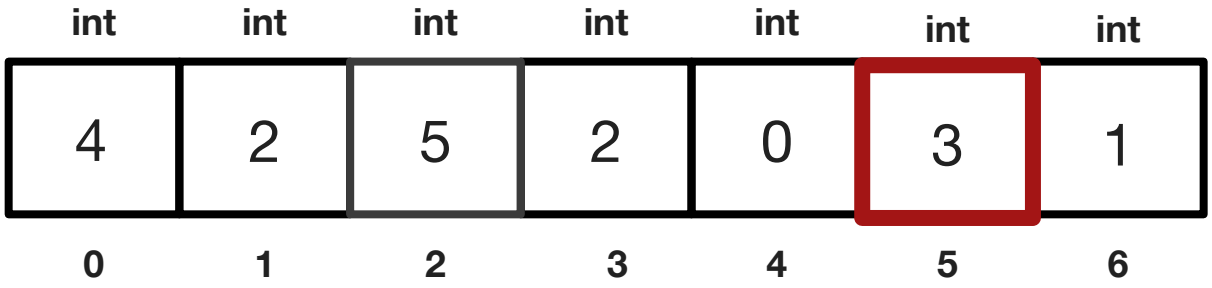


Traversing an Array

```c
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]`

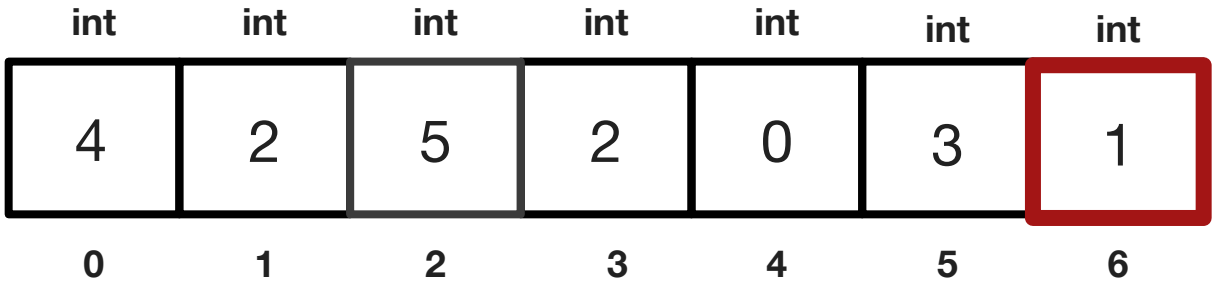


Traversing an Array

```c
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]`



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

```c
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 all 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

```c
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

- number_functions.c
- Write functions to
  - print an array
  - read data into an array
  - print out odd numbers in an array
  - find the maximum of an array

Can you have an array of structs?

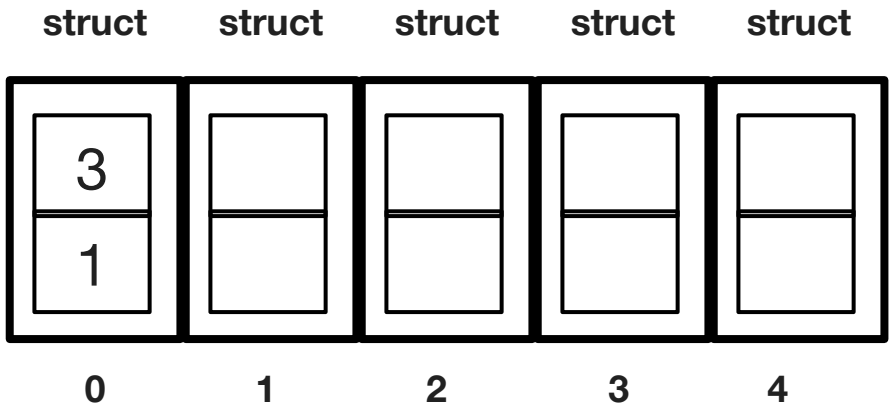
Arrays of structs

- 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 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;
```

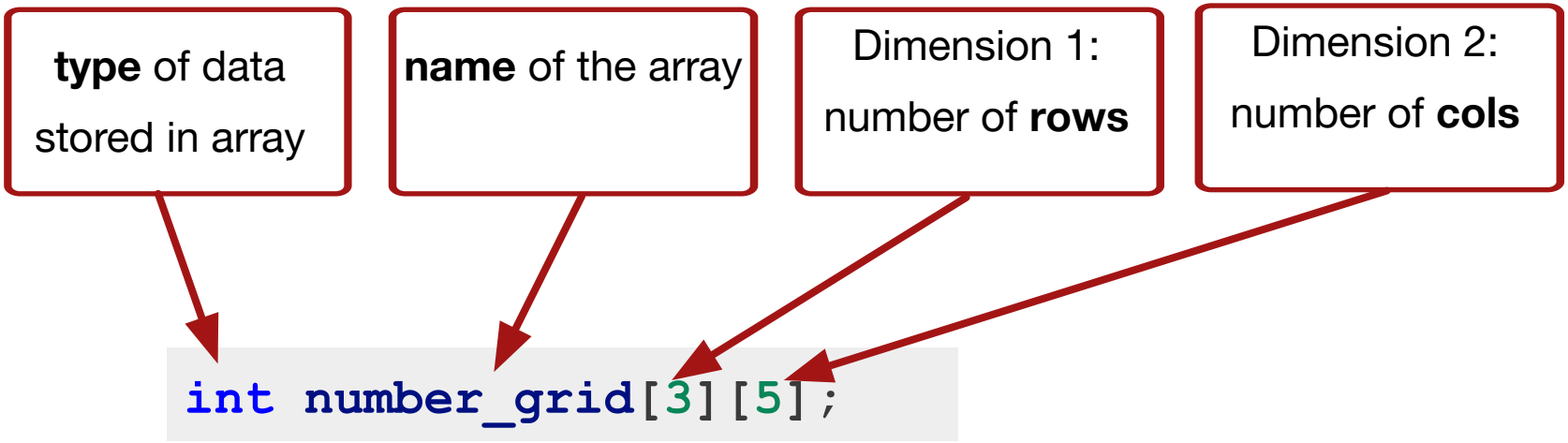


Code Demo of Array of structs

- `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)

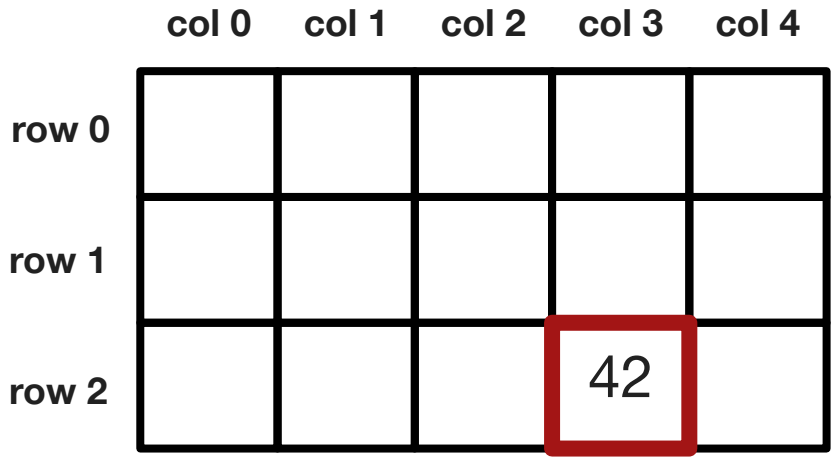
2D Arrays: Declaring



- 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.

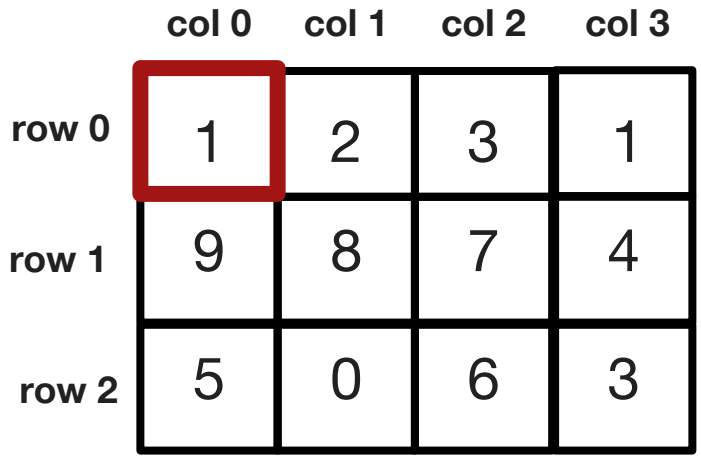
```
int row = 0;
while (row < SIZE) {
    int col = 0;
    while (col < SIZE) {
        printf("%d ", col);
        col++;
    }
    printf("\n");
    row++;
}
```

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

```
// 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

	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

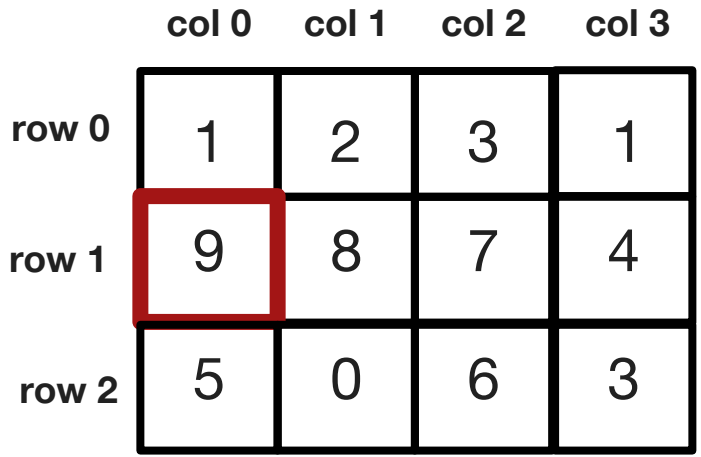
	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

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

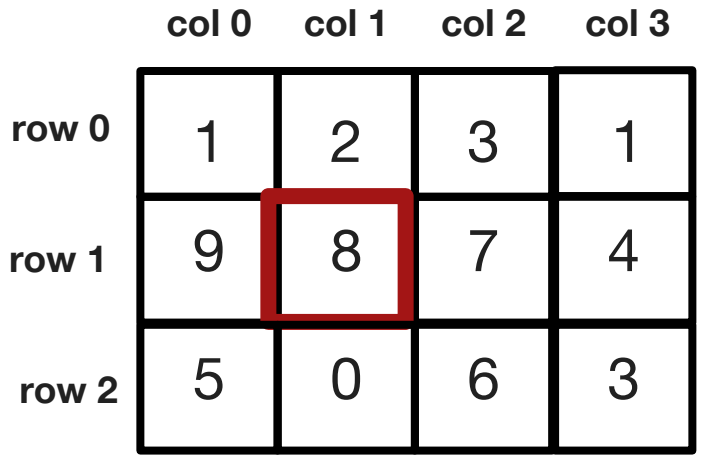


2D Arrays: Traversal

```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

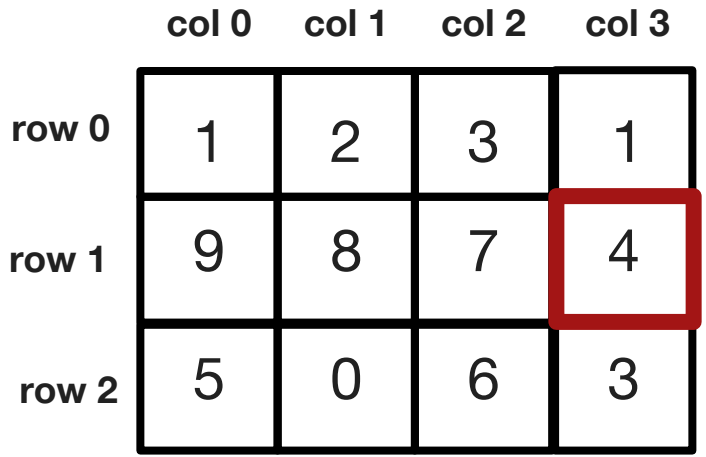
	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

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

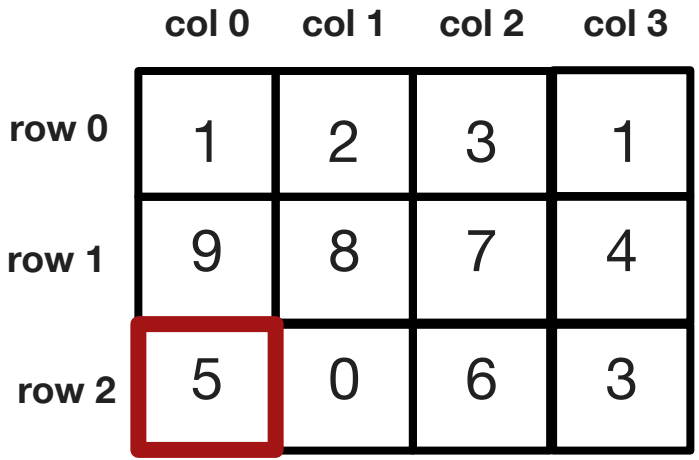


2D Arrays: Traversal

```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



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

	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 = 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 will be released Tuesday morning at 9am!

Next Lecture

- 2D array recap
- 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