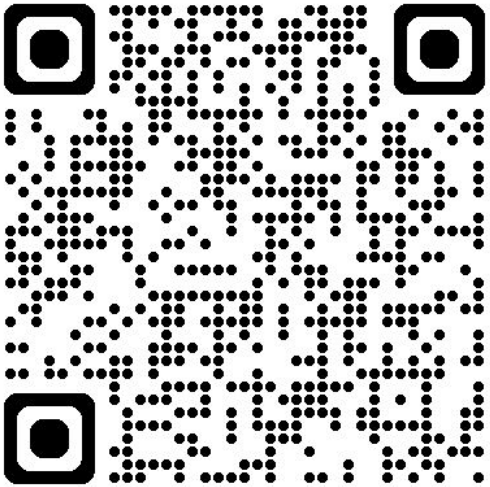
COMP1511/1911 Programming Fundamentals

Week 4 Lecture 2

Strings, Arrays of Strings

Link to Week 4 Live Lecture Code

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



Census Date this wednesday

Term 3, 2025 - Census date (T3)

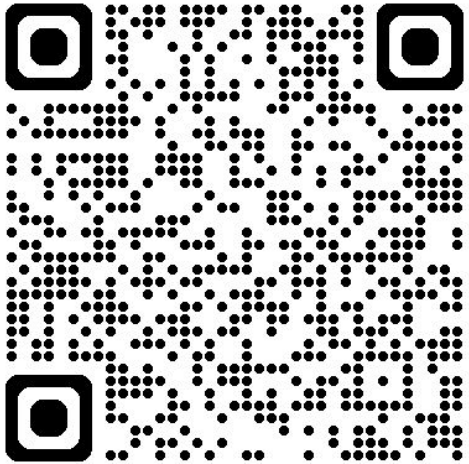
9 Oct 2025, 11:59pm

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

About Census Dates | UNSW Current Students

Public Holiday Classes

- There were no tut/labs yesterday.
- If you are in a monday class usually, don't miss out
  - Sign up for a replacement class for this week:
  - **https://buytickets.at/comp1511unsw/1857310**
  - Access code "COMP1511"
  - **COMP(1511|1911) 25T3 — Course Timetable**



Revision Sessions This Week (Not recorded)

The revision sessions are:

- **Week 4 Wednesday 08/10/2025 2PM-4PM - Lyre Lab (K17 G12)**
- **Week 4 Thursday 09/10/2025 12PM-2PM - Online**
  - **General | COMP1511/1911 Help Sessions + Revision Sessions | T3, 2025 | Microsoft Teams**

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/help-sessions/

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

Last Lecture

- Recap Arrays
- Functions and Arrays
- Arrays of structs
- 2D Arrays

Today’s Lecture

- Assignment 1 intro
- 2D Arrays recap
- Strings
- Arrays of strings

Note: Anything we don’t finish can be covered in the next lecture.

Assignment 1 has been released

- Due: **Mon Wk7 @ 5PM**
- It is an individual assignment
- Aims of the assignment
  - Use arrays and two-dimensional arrays to solve a larger problem
  - Apply the use of functions in code
  - Practice skills in debugging code, and skills in patience as you search for your missing semicolons
  - Practice using good style
    - You will be assessed on style! 20% of your mark

Assignment 1 Stages

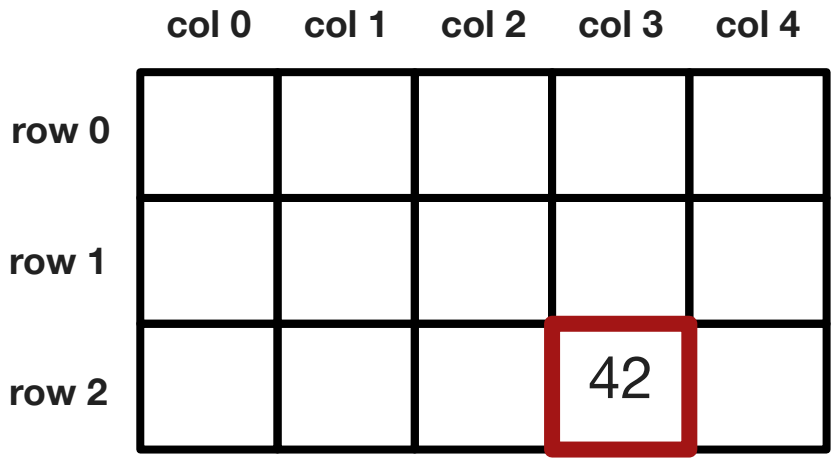
- Broken into stages
  - Stages 1, 2 and 3
- Stage 4:
  - Is a challenge stage.
  - Not everyone will finish stage 4.
  - You can get great marks without completing this stage
  - You are on track and doing well in the course even if you don’t get up to this stage

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
- If stuck, get help from
  - tutors
  - forum
  - help sessions

Recap: 2D Arrays: Accessing Indexes

```c
// 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;
```

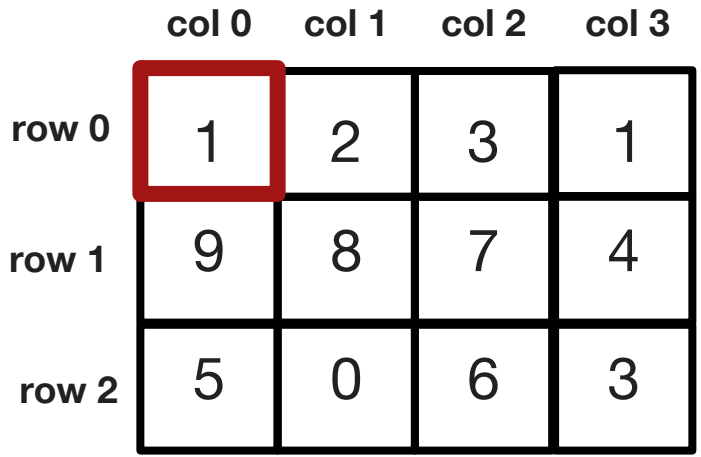


2D Arrays: Traversal

```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

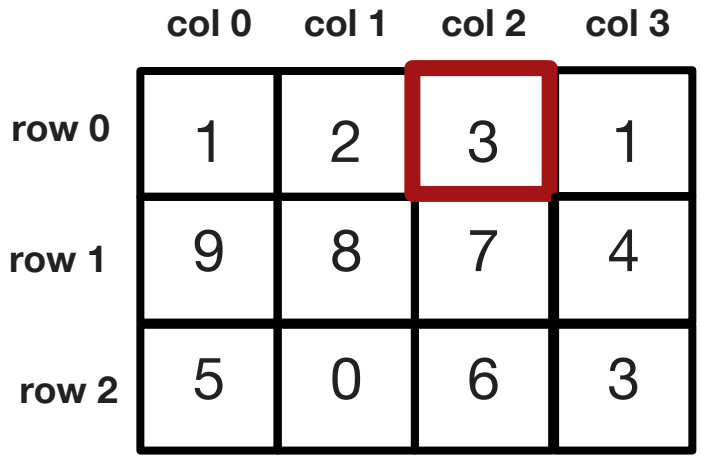
	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

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

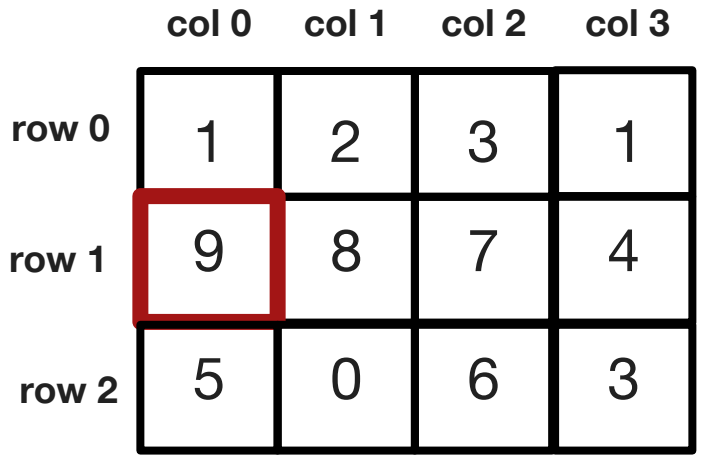
	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

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

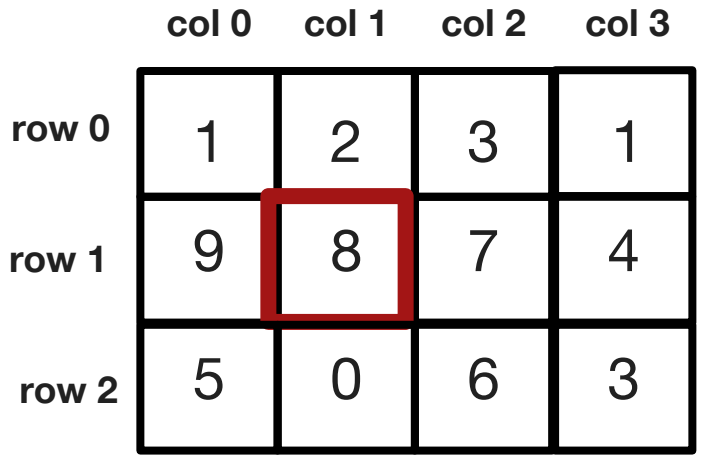


2D Arrays: Traversal

```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



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

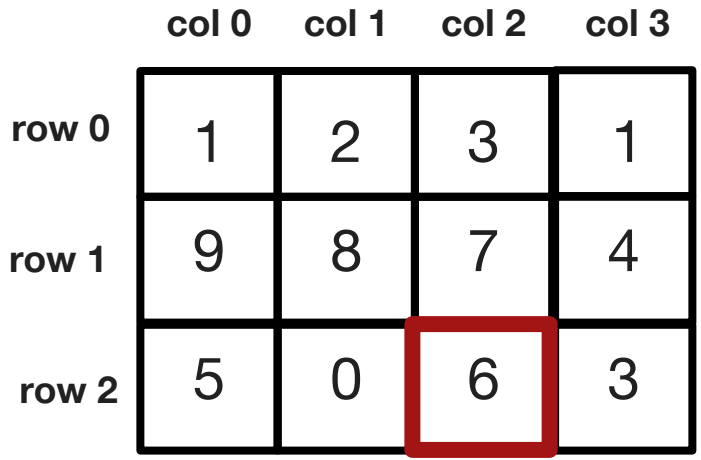
	col 0	col 1	col 2	col 3
row 0	1	2	3	1
row 1	9	8	7	4
row 2	5	0	6	3

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



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

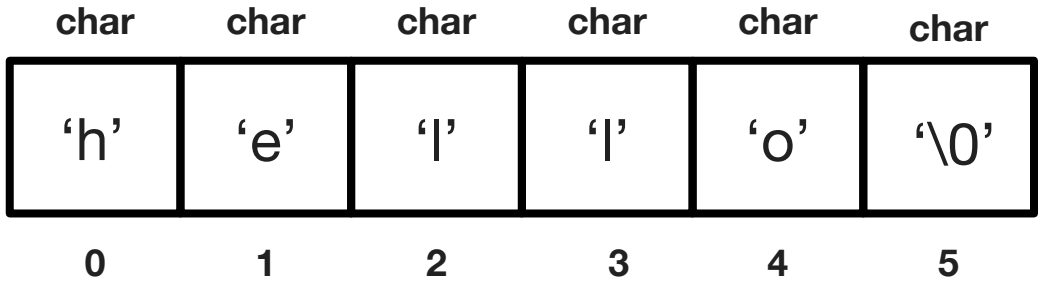
Recap of 2D Arrays Coding

- recap_2D_array.c
  - copy array
- diagonals.c
  - sum_diagonal_top_right

Strings

Strings: What are they?

- Strings are a collection of characters
- In C a string is
  - an array of **`char`**
  - that ends with a special character `'\0'` (null terminator)

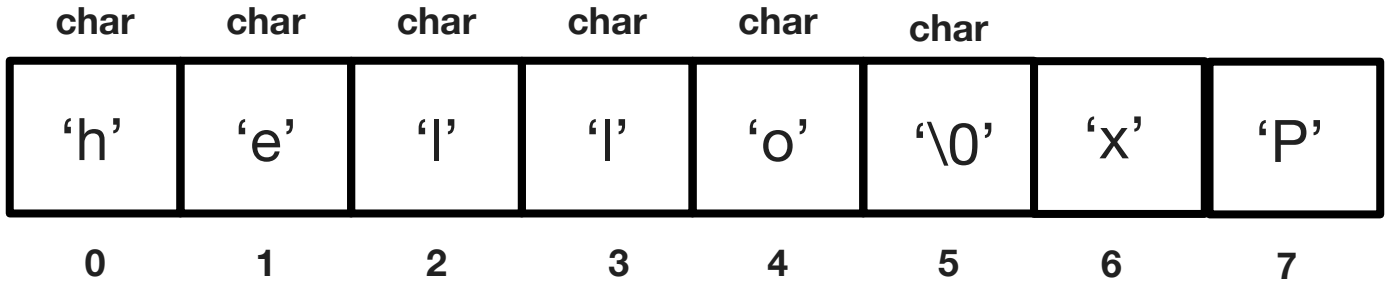


Null Terminator

- The null terminator ‘\0’ must be at the end of every string
  - If it does not have one it is not a string! Just an array of char
- The array must be big enough to store the extra character
- It is not displayed as part of the string
- It is very useful to know when our string has come to an end, when we loop through the array of characters
- Anything in the array after the ‘\0’ is not part of the string

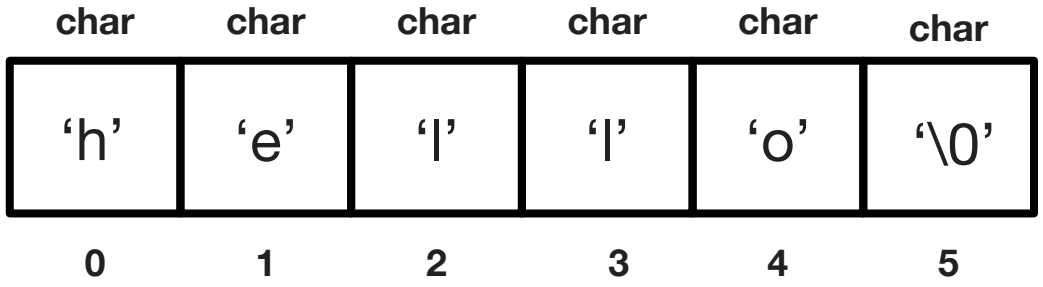
Strings: Null Terminator

- This still represents the string “hello”
- Anything in the array after the first ‘\0’ character is ignored



Strings: How do we initialise them?

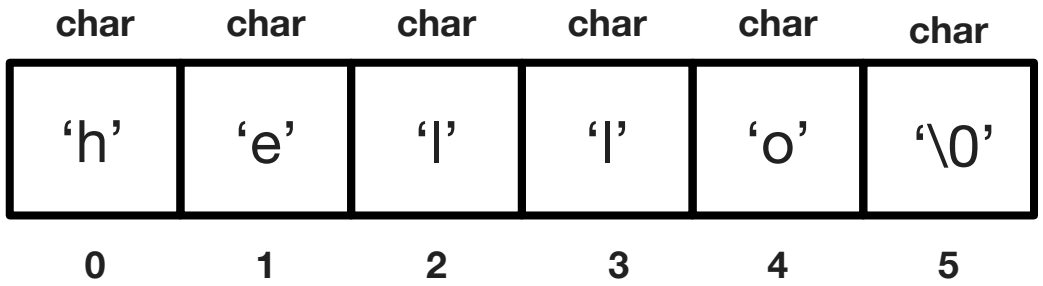
```c
// the painful way
char word[] = {'h','e','l','l','o','\0'};
// the more convenient way which does the same thing
char word[] = "hello";
```



Strings: How do we print them?

```
char word[] = "hello";
int i = 0;
while (word[i] != '\0') {
    printf("%c", word[i]);
    i++;
}
```

```
// the easy way
// using printf with %s
char word[] = "hello";
printf("%s", word);
```



Code Demo

simple_strings.c

- declaring,
- initialising,
- modifying,
- printing strings,
- writing our own printing function

Strings: Can I read them in with scanf %s?

- No. Please don't. It can read strings that are too long to fit in the array
  - Overwrite other memory - **buffer overflow**
  - Security Vulnerability
  - Hackers can exploit this
  - You will see more about this in COMP1521
- It may not do what you expect/want anyway
  - Stops when it encounters whitespace
- It is forbidden in the style guide. You will lose marks for using it

Strings: How do we read them in?

We fgets them: `fgets(array, size, stream);`

`fgets` needs three inputs:

- array - the array that the string will be stored into
- size - the size of the array
  - fgets will only read in and store a max of `size - 1` characters
- stream - this is where this string is coming from
  - For this course it will always be `stdin` (standard input: by default the input will always be from terminal)

Strings: How do we read them in?

One call to fgets will read in characters until

- size - 1 characters are read in
- a newline character is read in
  - this newline character is stored in the array
- we get to the end of file
  - which is Ctrl+D on a line of its own for terminal input

Note: There is a matching function that prints strings out

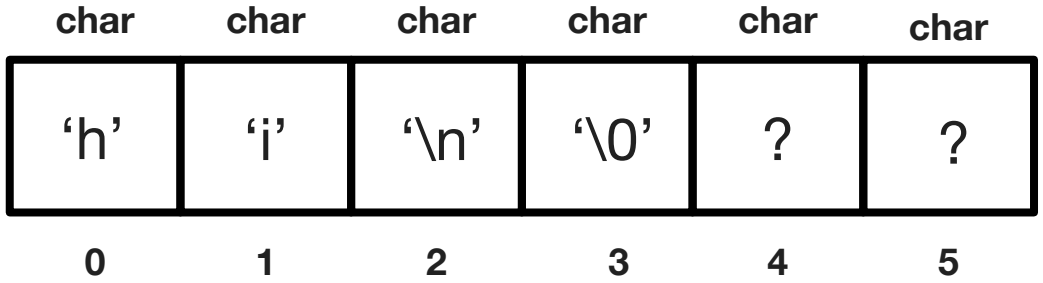
```
fputs(array, stream);
```

- For this course stream will always be `stdout` (terminal)

Strings: How do we read them in?

```
char array[MAX_LENGTH];
// Read in the string into array of length MAX_LENGTH
// from standard input - which by default is the terminal
fgets(array, MAX_LENGTH, stdin);
```

If **MAX_LENGTH** is 6 and the user types in **hi** then presses **enter** we would get an array like:



Strings: How do we read in many of them?

```c
// Declare an array to store your string
char array[MAX_LENGTH];

printf("Type in a string to echo: ");
// Read a string into array again and again
// until Ctrl+D is pressed (indicated by fgets returning NULL)
while (fgets(array, MAX_LENGTH, stdin) != NULL) {
    printf("The string is:\n");
    printf("%s", array);
    printf("Type in a string to echo: ");
}
```

Note: We are only ever storing 1 string at a time with this code

ctype.h library functions

You can use these to make your life easier when working with characters!

`toupper()`	convert a character to uppercase
`tolower()`	convert a character to lowercase
`isupper()`	test whether a character is uppercase
`islower()`	test whether a character is lowercase

Find more here: https://www.tutorialspoint.com/c_standard_library/ctype_h.htm

Code Demo: Reading in Strings

read_strings.c

- Read in a string
- Try reading in a string that is too long
- Repeatedly read in a string
- Convert string to all capitals

string.h library functions

Some other useful functions for strings:

`strlen()`	gives us the length of the string excluding the '\0'
`strcpy()`	copy the contents of one string to another
`strcmp()`	compare two strings
`strcat()`	append one string to the end of another (concatenate)
`strchr()`	find the first occurance of a character in a string

Find more here: https://www.tutorialspoint.com/c_standard_library/string_h.htm

String Functions: strcpy strlen

```c
// Declare an array to store a string
char puppy[MAX_LENGTH] = "Boots";

// Copy the string "Finn" into the word array
// be careful the array is big enough so you do not overflow
// the array
strcpy(puppy, "Finn");
printf("%s\n", puppy);
// Find string length. It does NOT include '\0' in the length
int len = strlen(puppy);
printf("%s has length %d\n", puppy, len);
```

String Functions: strcmp

```c
// Declare an array to store a string
char name[] = "Oscar";

// Use strcmp to compare 2 strings
// It will return 0 if the strings are equal
// A negative number if the first string < second string
// A positive number if the first string > second string
if (strcmp("Oscar", name) == 0) {
    printf("Hello Oscar!\n");
} else {
    printf("You are not Oscar!\n");
}
```

String Functions: fgets and strcmp

```c
// Declare an array to store a string
char name[MAX_LENGTH];
printf("Type in a name: ");

// Read in a string
fgets(name, MAX_LENGTH, stdin);

// Use strcmp to compare 2 strings
if (strcmp("Oscar", name) == 0)
    printf("Hello Oscar!\n");
} else {
    printf("You are not Oscar!\n");
}
```

What issue would we get here?

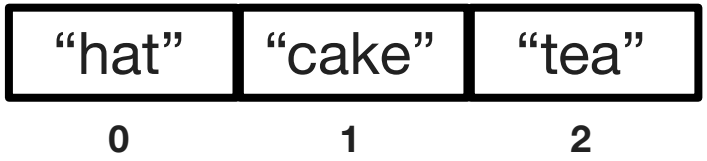
String Functions Demo

string_functions.c

Array of Strings

```
// This array can store 3 strings.
// Each string has max size 5, including '\0'
char words[3][5] = {"hat", "cake", "tea"};
```

- You can have an array of strings!
- You can also think of it as a 2D array of characters

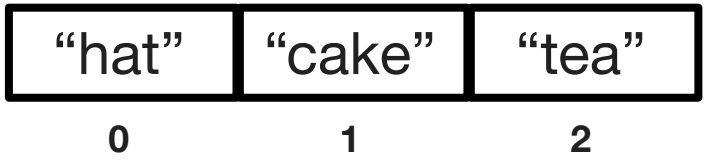


	col 0	col 1	col 2	col 3	col 4
row 0	‘h’	‘a’	‘t’	‘\0’	
row 1	‘c’	‘a’	‘k’	‘e’	‘\0’
row 2	‘t’	‘e’	‘a’	‘\0’	

Array of Strings

```
char words[3][5] = {"hat", "cake", "tea"};
// Using 1 index gives us a row/string
// This would print "cake"
printf("%s\n", words[1]);
```

- You can have an array of strings!
- You can also think of it as a 2D array of characters

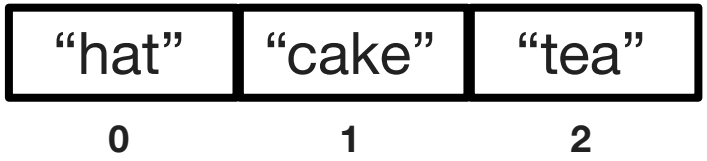


	col 0	col 1	col 2	col 3	col 4
row 0	'h'	'a'	't'	'\0'	
row 1	'c'	'a'	'k'	'e'	'\0'
row 2	't'	'e'	'a'	'\0'	

Array of Strings

```
char words[3][5] = {"hat", "cake", "tea"};
// Using 2 indexes gives us a character
// This would print the 'e' from "tea"
printf("%c\n",words[2][1]);
```

- You can have an array of strings!
- You can also think of it as a 2D array of characters



	col 0	col 1	col 2	col 3	col 4
row 0	'h'	'a'	't'	'\0'	
row 1	'c'	'a'	'k'	'e'	'\0'
row 2	't'	'e'	'a'	'\0'	

Coding Demo: Array of Strings

array_of_strings.c

- initialise data
- print out data

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/EUq6MitZp4

What did we learn today?

- recap 2D arrays
  - 2D_array.c, diagonals.c
- strings
  - simple_strings.c, read_strings.c, string_functions.c
- arrays of strings
  - array_of_strings.c

Next Week

Monday:

- Command Line Arguments
- Putting Everything Together:
  - A larger lecture example with 2D arrays of structs with enums
  - Help with concepts needed in assignment 1

Tuesday:

- Pointers

Reach Out

Content Related Questions:

Forum

Admin related Questions email:

cs1511@unsw.edu.au