

# **2D Arrays**

## **Strings recap**

- An array of chars
- We have a single identifier for the string
- Anything we can do with arrays, applies

```
char[]
```

index:	0	1	2	3	4	5	6	7	8	9	10	11	12	13
values:	J	A	K	E		R	E	N	Z	E	L	L	A	\0

Notice the `\0` at the end! This means that C will know when it reaches the end of the array

Note the # of elements, and don't forget the `\0`

## String literals

`"Jake!"`

- uses double quotes `"` to wrap the string literal
- single quote for characters!
- Used to assign strings to `char[]` easily:

```
char name[] = "Jake Renzella";
```

## Useful string functions

- `fgets()` -> reads a string
- `fputs()` -> prints a string
- `strlen()` -> gives us the length of the string (excluding the `\0`).
- `strcpy()` -> copy the contents of one string to another
- `strcat()` -> join one string to the end of another (concatenate)
- `strcmp()` -> compare two strings
- `strchr()` -> find the first occurrence of a character

note: some of these may require `#include <string.h>`

# Reassigning a string

```
int main(void) {  
    char name[MAX_LEN] = "Jake";  
    strcpy(name, "Mr Otterington");  
}
```

^ Remember we can't reassign like:

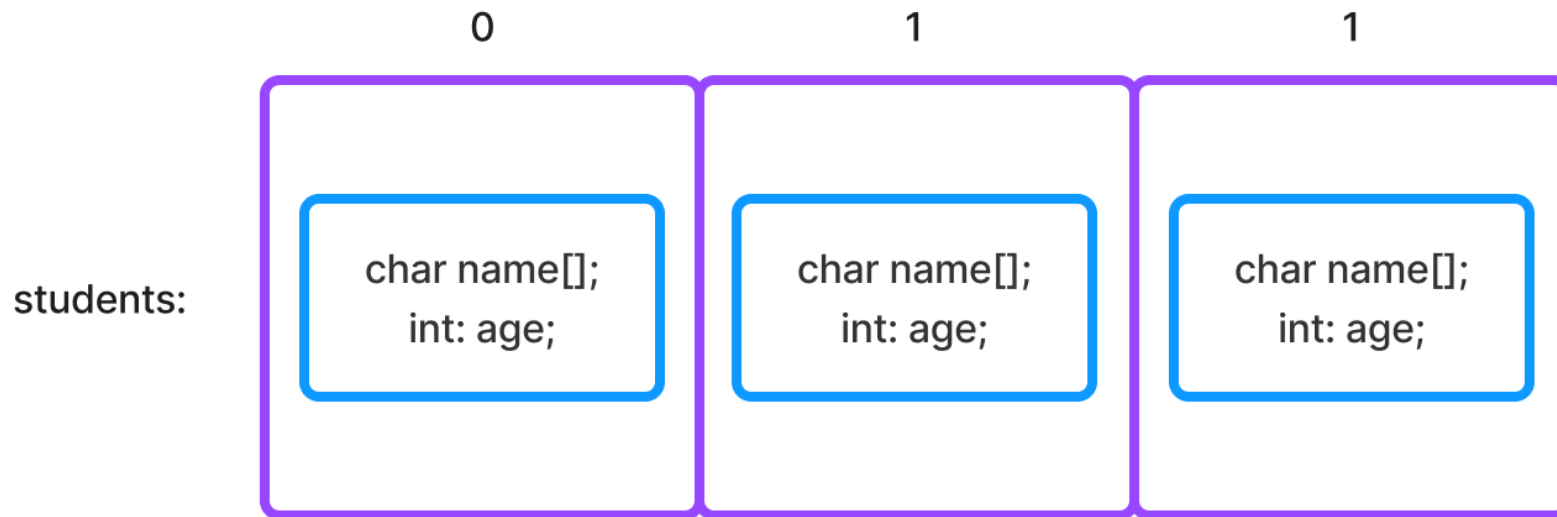
```
name = "Mr Otterington";
```

# **2D arrays**

We can have arrays of type (char, int, struct, enum)

index:	0	1	2	3	4	5	6
values:							

# Array of structs



- Use `students[1].name;` to access element 1's name



# Array of arrays

## 2D arrays

```
<type> <identifier> [<rows>]  
[<cols>];
```

```
int my_grid[5][5];  
my_grid[2][3];
```

2D arrays

cols:

rows:

	0	1	2	3	4
0					
1					
2					
3					
4					

# Visualisation

# **Large demo Program**

- An array of array of structs
- Battleships? Naughts and Crosses?

# Feedback

<https://forms.office.com/r/Ze4admEWnR>

