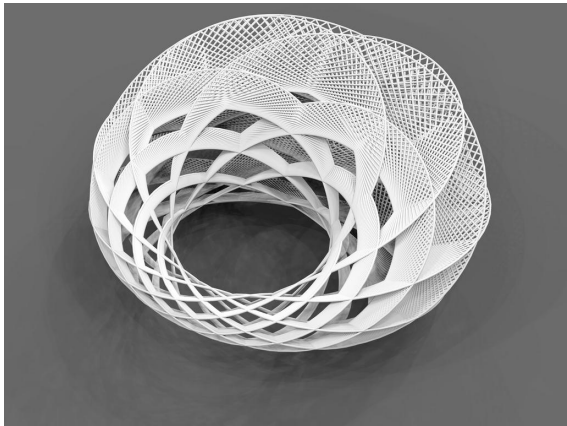
Strings

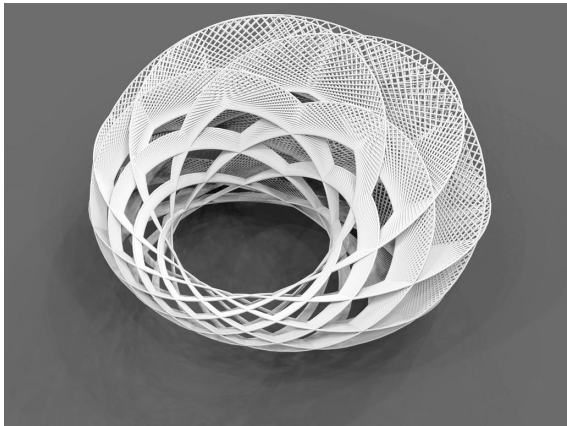
Strings

- Up to this point, we've only looked at individual numbers / characters
  - BUT...arrays can hold multiple independent numbers...
  - Why not multiple chars at once?



Strings

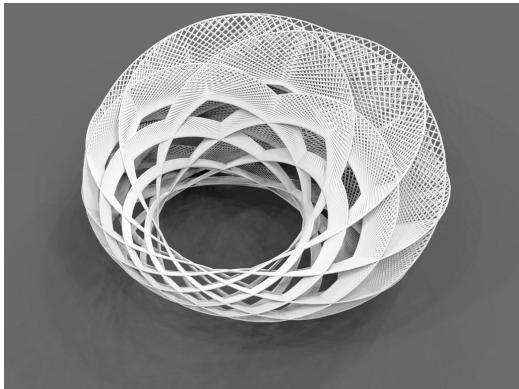
- Strings are char arrays
- These char arrays are treated as a single entity (like words in English)



Strings

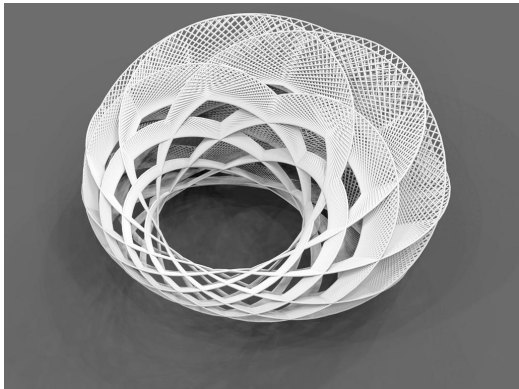
- Formal definition: String = a sequence of zero or more characters

- Remember: Everything is stored as a number within a computer - even strings



Strings vs Chars

- Chars
  - Take up 2 bits of memory
  - Are represented using single quotes: ‘A’
- Strings
  - Take up arbitrary amounts of memory (depending on how long they are)
  - Are represented by double quotes: “Hello World”



Strings vs Chars

- Chars
  - Take up 2 bits of memory
  - Are represented using single quotes: ‘A’
- Strings
  - Can also include entire sentences (remember, spaces/punctuation/numbers are chars too)

Strings vs Chars

- Chars
  - Take up 2 bits of memory
  - Are represented using single quotes: ‘A’
- Strings
  - The name of the string (similar to arrays) is a pointer to the first memory address in the char array

Strings vs Chars

- Chars
  - Take up 2 bits of memory
  - Are represented using single quotes: ‘A’
- Strings
  - Strings also have a special **trailing character → the null character, ‘\0’**

Declaring / Printing Strings

- Strings are declared as char arrays

```
> char array_name[] = “Curley”;
```

- They can also be printed as arrays

```
> int i;
> for (i = 0; i < 7; i++) {
>     printf(“%d - %c\n”, i, array_name[i]);
> }
```

Declaring / Printing Strings

- Null terminator is present as the last char

```
> int i;
> for (i = 0; i < 7; i++) {
>     printf("%d - %d\n", i, array_name[i]);
> }
```

Declaring / Printing Strings

- There are easier ways to print strings than for loops...

```
> char array_name[] = “Curley”;
> printf(“%s\n”, array_name);
```

Declaring / Printing Strings

- There are easier ways to print strings than for loops...

```
> char array_name[] = “Curley”;
> printf(“%s\n”, array_name);
```

- The **%s qualifier** only works with char arrays

Declaring / Printing Strings

- There are easier ways to print strings than for loops...

```
> char array_name[] = “Curley”;
> printf(“%s\n”, array_name);
```

- The printf() statement will keep displaying characters until the first **null character (\0)** is read

Manipulating Null Characters

- The printf() statement will keep displaying characters until the first **null character (\0)** is read
- The string can be manipulated through adding / removing null characters

Try:

> array_name[2] = '\0'

Manipulating Null Characters

- The printf() statement will keep displaying characters until the first **null character (\0)** is read
- The string can be manipulated through adding / removing null characters

Try:

> array_name[6] = ‘!’

- This is called a “buffer overflow”

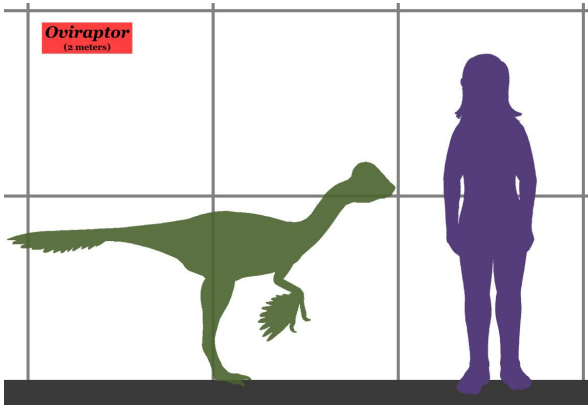
Scanning in strings

- Not as easy as: scanf(“%s”, array_name);
  - Remember, array_name is a pointer...so you would be trying to fit the entire string into a single memory address
  - Called “pointer decay”

Scanning in strings

- To scan in a string: use the function **fgets()**
  - Included in <stdio.h>

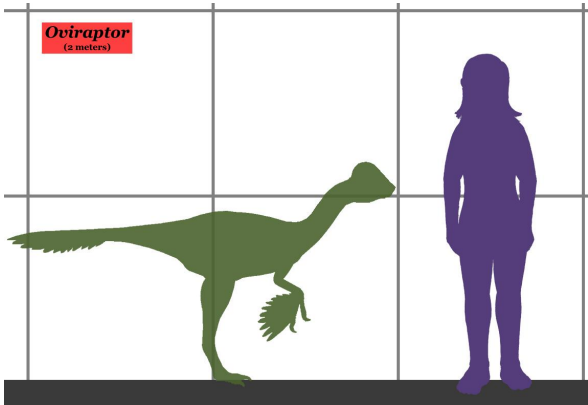
> fgets(array_name, MAX_SIZE, stdin);



Scanning in strings

> fgets(array_name, MAX_SIZE, stdin);

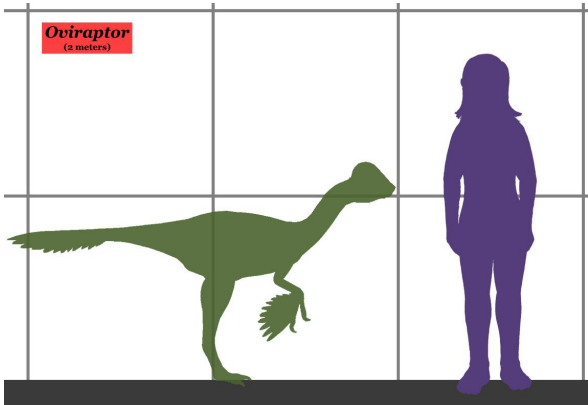
- The *array_name* is simply the name of the array
- *MAX_SIZE* is the number of bytes allocated for the array (should be the same number used when defining the array)
- “stdin” means input is coming from the screen



Scanning in strings

> fgets(array_name, MAX_SIZE, stdin);

- Quirks of fgets()
  - Stores the newline character (‘\n’) as well



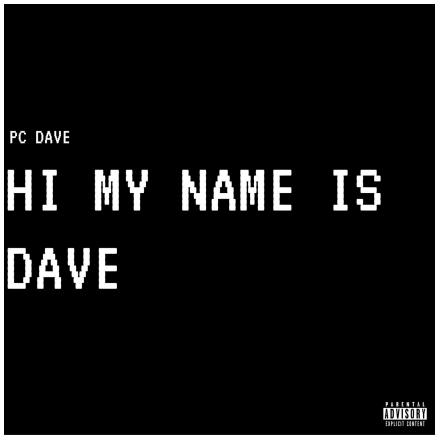
Removing the newline character

- In order to remove this trailing newline, include the following lines of code:

```
int i = 0;
while (array_name[i] != ‘\0’) {
        i++
}
if (i > 0 && array_name[i - 1] == ‘\n’) {
        array_name[i - 1] = ‘\0’;
}
```

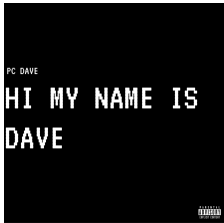
Strings Coding: 1

- Enter in your name as a string (using fgets()) and print it out
- Requirements
  - Have to remove the trailing newline character from fgets



Strings Coding: 2

- Enter in your name as a string (using fgets()) and print it out
- Requirements
  - Have to remove the trailing newline character from fgets
  - Switch the second-to-last character with the second character in your name
    - Ex: “Mr. Malloy” ➔ “Mo. Mallry”



Strings Coding 3: Bioinformatics

- Rosalind: http://rosalind.info/problems/list-view/
  - Bioinformatics platform
  - Many of the early problems deal with strings and can be easily coded
  - Complete at minimum the first one ➜ if you want to go further, go for it
    - First one will be submitted