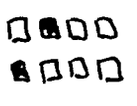
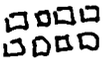
C strings are sequences of char (1 byte each) followed by a ∅ byte, called a null terminator

∅ in decimal
'\∅' as char

C strings are often created as arrays

```
char msg[] = "Hello!";
char msg2[] = {'H', 'e', 'l', 'l', 'o', '!', ∅}
```

	0	1	2	3	4	5	6
	'H'	'e'	'l'	'l'	'o'	'!'	'\∅'
	72	101	108	108	111	33	0
truth				-			

msg[1] evaluates to 'e' 101

msg[4] evaluates to 'o' 111

msg[0] = 'h' changes msg to have 'h' at index 0

#include <string.h>

strlen(s) - returns # of characters up to (not including) null terminator

strlen is for C strings!

Not all arrays are C strings. There's no length function for arrays.

```
#include <stdio.h>
printf("<format string>", val1, val2, ....)
```

Prints the format string with any format specifiers replaced with the corresponding formatted value

%c - characters
%d - decimal numbers
%u - decimal nums unsigned
%s - strings (C-strings)

```
printf("%s has %d characters", "Hi", 2);
```

Hi has 2 characters.

C varargs (if you are curious about printf taking varying args)

Errors:
- mismatched # of format specifiers and values
- no way to format a value with given specifier

```c
#include <stdio.h>

void uppercase(char str[]) {


    for(int i=0; str[i] != '\0'; i+=1) {
        if (str[i]>=97  && str[i] <= 122) {
            str[i] -= 32;
        }
    }
}

void inspect(char str[]) {
    for (int i = 0; str[i] != '\0'; i++) {
        char c = str[i];
        printf("(%c %d) ", c, c);
    }
    printf("\n");
}

int main() {
    char message[] = "Hello!";
    printf("%s\n", message);
    printf("(%c %d) (%c %d) (%c %d) (%c %d) (%c %d) (%c %d) (%c %d)\n",
           message[0], message[0],
           message[1], message[1],
           message[2], message[2],
           message[3], message[3],
           message[4], message[4],
           message[5], message[5],
           message[6], message[6]);
    inspect(message);

    // After this call, message should contain "HELLO!"
    uppercase(message);
    printf("After uppercase: %s\n", message);
}
```

```
$ gcc hello.c -o hello
$ ./hello
Hello!
(H 72) (e 101) (l 108) (l 108) (o 111) (! 33) ( 0)
(H 72) (e 101) (l 108) (l 108) (o 111) (! 33)
After uppercase: HELLO!
```

'a' 'z' 'A' 'Z'

97 122 65 90

```c
if(str[i] >= 'a' && str[i] <= 'z') {
    str[i] -= ('a' - 'A');
}
```