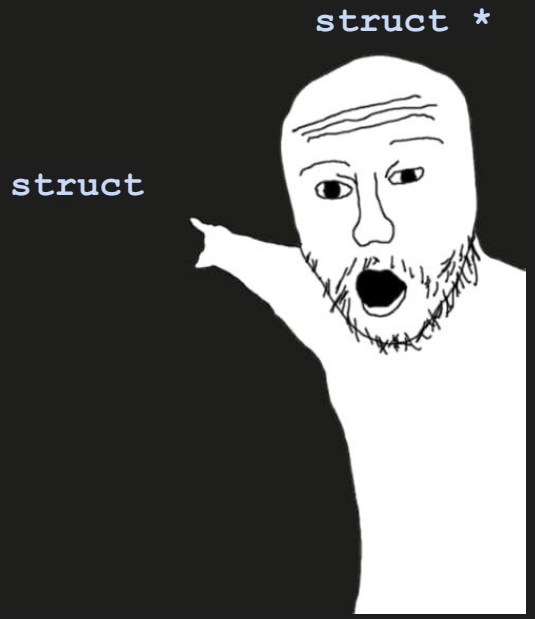


comp1511 week 7

welcome back!

notices

- assignment 0 marks have been released
  - submissions tab -> assignment 0 -> click blue dot for more details
- congratulations for finishing assignment 1!!!
  - will be marked by mid-week 9
  - I'll try to provide general style feedback next week in the tute

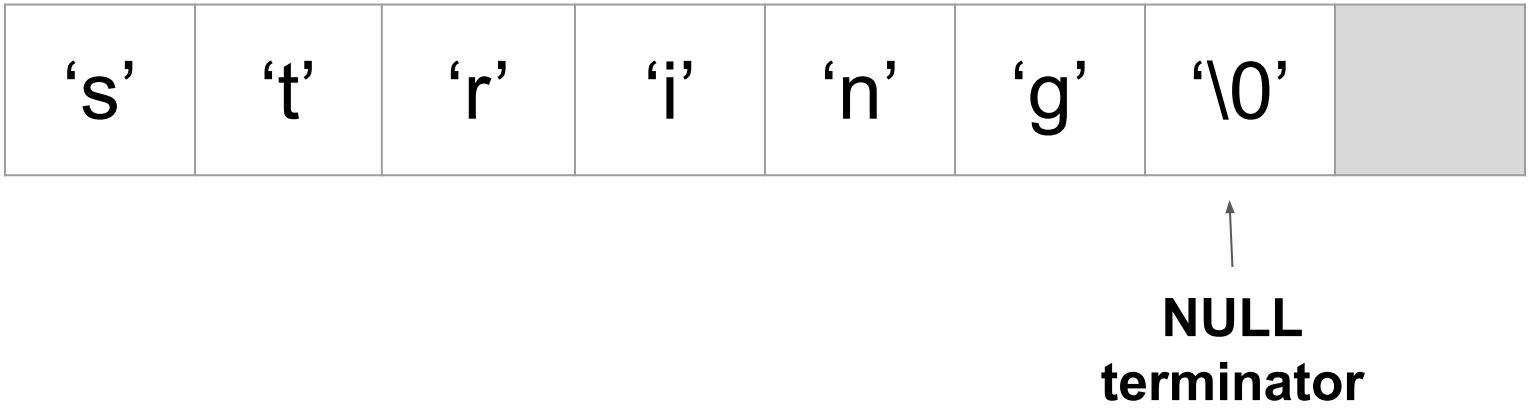
✨reflection time ✨

what have you learnt from assignment 1?
(anything to take into assignment 2?)

today

- strings
- fgets
- struct pointers

what is a “string”



why do we care about strings?

what's the difference about this code vs looping through a regular array?

```c
#include <stdio.h>

// Manually prints out a string, one character at a time.
// Should behave like printf("%s\n");
void print_string(char *string) {
    int i = 0;
    while (string[i] != '\0') {
        printf("%c", string[i]);
        i++;
    }
    printf("\n");
}

int main (void) {
    char my_string[] =  "Many chars";

    print_string(my_string);
    return 0;
}
```

strings practice:

given some functions related to chars, implement one of the following functions related to strings :)

```c
// Functions to implement:

// 1.
// returns the number of lowercase letters in `char *string`
int count_lowercase(char *string);

// 2.
// modifies `char *string` by converting all its vowels to uppercase
void make_vowels_uppercase(char *string);

// 3..
// shortens a string so that it ends after the first word
// e.g. "This is a sentence" should turn into:
//      "This"
//
// (hint. what defines when a string ends?)
void delete_following_words(char *string);
```

helper functions

```c
int is_lowercase(char c);
int is_uppercase(char c);
int is_letter(char c);
char to_lowercase(char c);
char to_uppercase(char c);
int is_vowel(char c);
```

see their implementation:
https://cgi.cse.unsw.edu.au/~cs1511/22T2/tut/07/questions

fgets

another way to read input!
(from STDIN or elsewhere)

is a “safer” function because you specify how much input to read

try: man 3 fgets (in terminal)
code demo: fgets_demo.c

Description

The C library function **char *fgets(char *str, int n, FILE *stream)** reads a line from the specified stream and stores it into the string pointed to by **str**. It stops when either **(n-1)** characters are read, the newline character is read, or the end-of-file is reached, whichever comes first.

Declaration

Following is the declaration for fgets() function.

```
char *fgets(char *str, int n, FILE *stream)
```

Parameters

- **str** − This is the pointer to an array of chars where the string read is stored.
- **n** − This is the maximum number of characters to be read (including the final null-character). Usually, the length of the array passed as str is used.
- **stream** − This is the pointer to a FILE object that identifies the stream where characters are read from.

Return Value

On success, the function returns the same str parameter. If the End-of-File is encountered and no characters have been read, the contents of str remain unchanged and a null pointer is returned.

If an error occurs, a null pointer is returned.

https://www.tutorialspoint.com/c_standard_library/c_function_fgets.htm

implementing a version of fgets ft. scanf(“%c”, &c)

function prototype

char *my_fgets(char *str, int size);

write your implementation with ***pseudocode***

properties of fgets

- Scans characters into str (an array of chars) *until either*
  1. A '\n' is scanned in to the array:
     - str is returned (with the “\n” still at the end of the string).
  2. size - 1 characters are scanned in:
     - str is returned.
  3. CTRL-D is pressed:
     - if > 0 characters have been scanned in:
       - str is returned.
     - if 0 characters have been scanned in:
       - NULL is returned.
- If any characters were scanned in, then a '\0' is added to the array after the last character.

example of pseudocode

```
// C style pseudo-code.

Inputs: array: an array of integers
        size: the size of of array
Output: the sum of the array
int sum_positive_elements(int array[], int size) {

    Initialise variables i and sum to 0
    while (i < size) {
        if (array[i] is positive) {
            add array[i] to sum
        }
        i++
    }

    return sum
}
```

struct pointers

```c
enum weapon { no_weapon, big_sword, little_sword, wand, fish };
enum armor { no_armor, knight_armor, mage_robes, overalls };

struct party_member {
    char character_name[100];

    // Gear:
    enum weapon weapon;
    enum armor armor;
};

// Swaps the weapon and armor of member1 and member2
void swap_gear(struct party_member member1, struct party_member member2) {
    enum armor temp_armor = member2.armor;
    enum weapon temp_weapon = member2.weapon;

    member2.armor = member1.armor;
    member2.weapon = member1.weapon;


    member1.armor = temp_armor;
    member1.weapon = temp_weapon;
}
```

why doesn't swap gear work properly?

let's fix it: struct_pointers.c