COMP1511 PROGRAMMING FUNDAMENTALS

# LECTURE 18

Revision: Pointers, Strings and Arrays

The final hurrah and big thank you

## LAST TIME...

- Revision of Linked Lists - a few example problems

## TODAY...

- Revision of pointers, arrays, and strings with some example problems
- hopefully no more stories about dead rats

## WHERE IS THE CODE?

**Live lecture code can be found here:**

HTTPS://CGI.CSE.UNSW.EDU.AU/~CS1511/23T1/LIVE/WEEK10/

## COURSE FEEDBACK

Tell us about your experience and shape the future of education at UNSW.

**Click the link in Moodle**

Please be mindful of the UNSW Student Code of Conduct as you provide feedback. At UNSW we aim to provide a respectful community and ask you to be careful to avoid any language that is sexist, racist or likely to be hurtful. You should feel confident that you can provide both positive and negative feedback but please be considerate in how you communicate.

**my Experience surveys**

**http://myexperience.unsw.edu.au/**

# REVISION CLASSES

**PLEASE BOOK NOW!**

Come along and work on revision problems with the support of our lovely tutors:

- FACE TO FACE in Sitar/Kora labs J17:
    - Monday 2-4pm (Sitar) - Anivridh and Gab
- ONLINE:
    - Wednesday 10-12pm - Salina and Liz

Register:

https://www.eventbrite.com.au/e/560086883947

## REVISION TIME!

## ARRAYS

- Let’s see a similar problem to the exam
- This is ●○○

https://cgi.cse.unsw.edu.au/~cs1511/23T1/activity/find_totals

## REVISION TIME!

## ARRAYS

- Let’s see a similar problem to the exam
- This is ●○○

Write a C program indivisible.c, which should print the integers read which are not exactly divisible by any other of the integers read. The reading until EOF is done for you, you only have to implement the divisibility function.

You may assume that the program's input will contain only integers.

You may assume that all integers are >1.

# REVISION TIME!

## ARRAYS

- Let’s see a similar problem to the exam
- This is ●○○

Match the example below EXACTLY.

```
 $ ./indivisible
42
7
6
12 'Ctrl-D'
Indivisible numbers: 7 6
```

# REVISION TIME!

## ARRAYS

- Let’s see a similar problem to the exam
- This is ●○○

https://cgi.cse.unsw.edu.au/~cs1511/23T1/activity/array_clamping_max

## REVISION TIME!

## ARRAYS

- Let’s see a similar problem to the exam
- This is ●●○

Write a C program that reads integers from standard input until it reads a negative integer. It should then print the odd numbers on one line and then print the even numbers on the next line.

You may assume that the program’s input will contain only integers, in other words, you can assume scanf succeeeds.

You can assume a negative integer will always be read.

You can assume a maximum of 1000 integers are read before a negative integer is read.

## REVISION TIME!

## ARRAYS

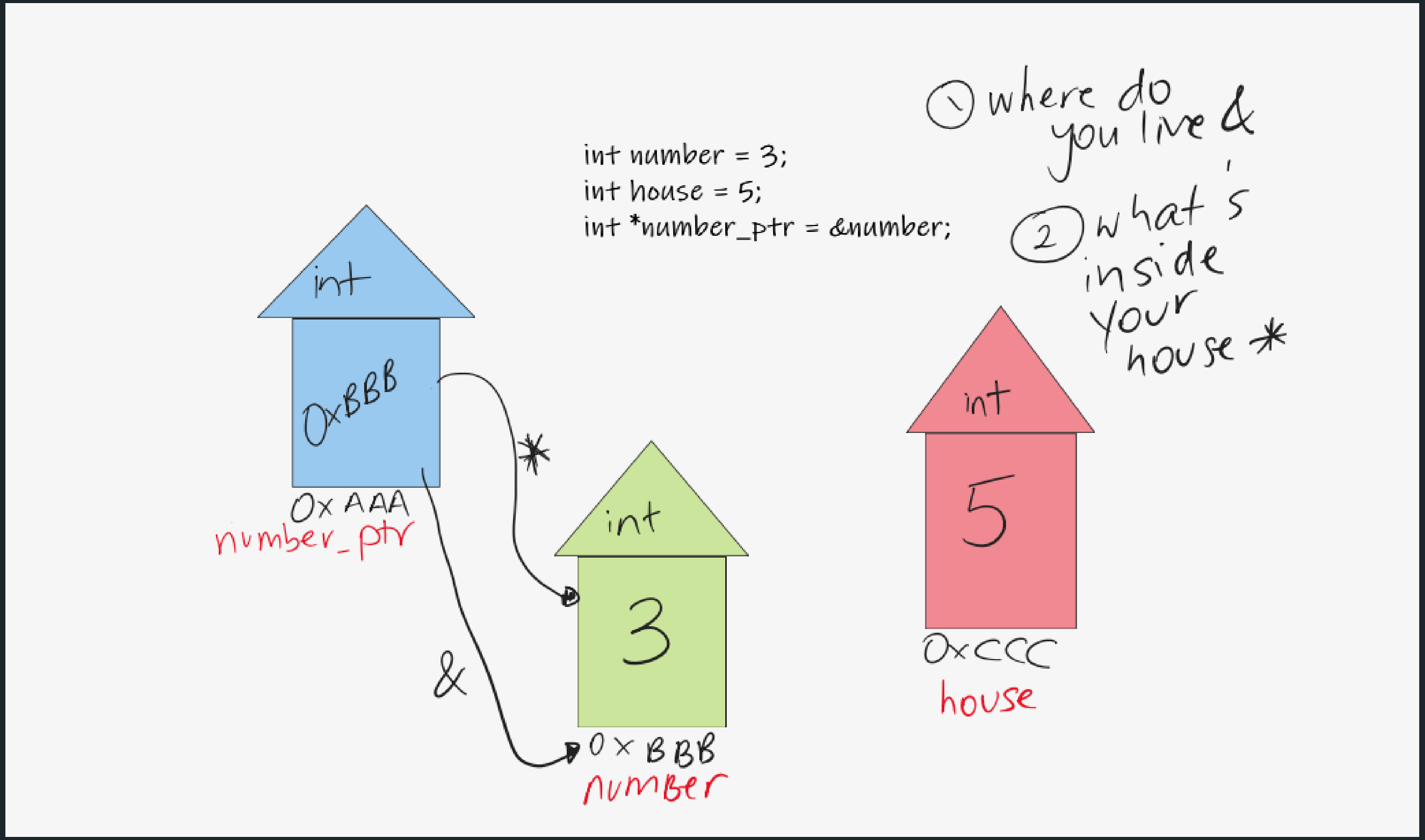
- Let's see a similar problem to the exam
- This is ●●○

```
$ ./even_negative
1
2
3
2
-42
Odd numbers were: 1 3
Even numbers were: 2 2
```

# REVISION TIME!

# POINTERS

- Pointers are another variable type in C
- Pointers store the memory address of another variable
    - & - gives the address of
    - * - dereferences a pointer, so provides the value of stored at the address the pointer is at



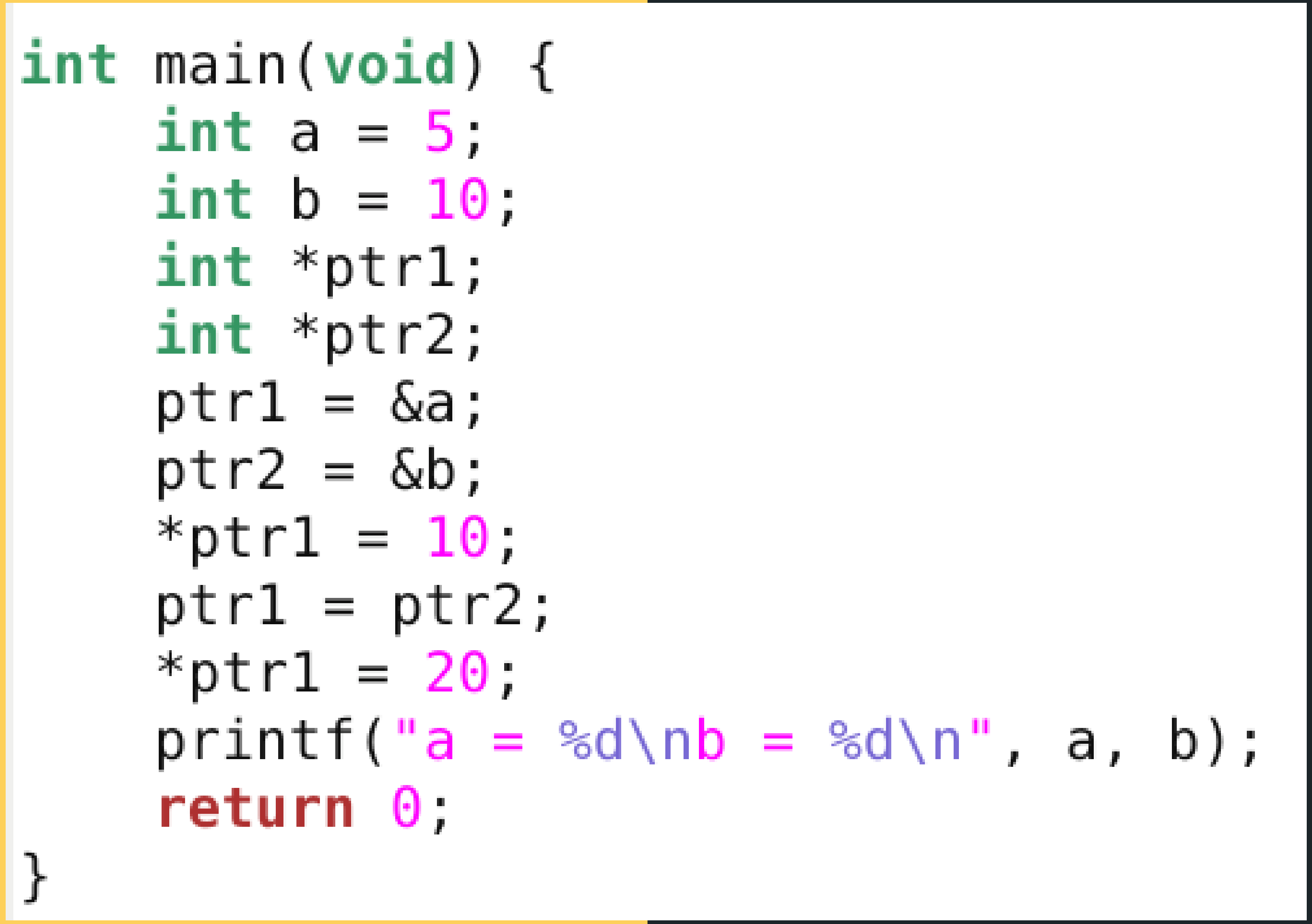
```
int number = 3;
int house = 5;
int *number_ptr = &number;
```

# REVISION TIME!

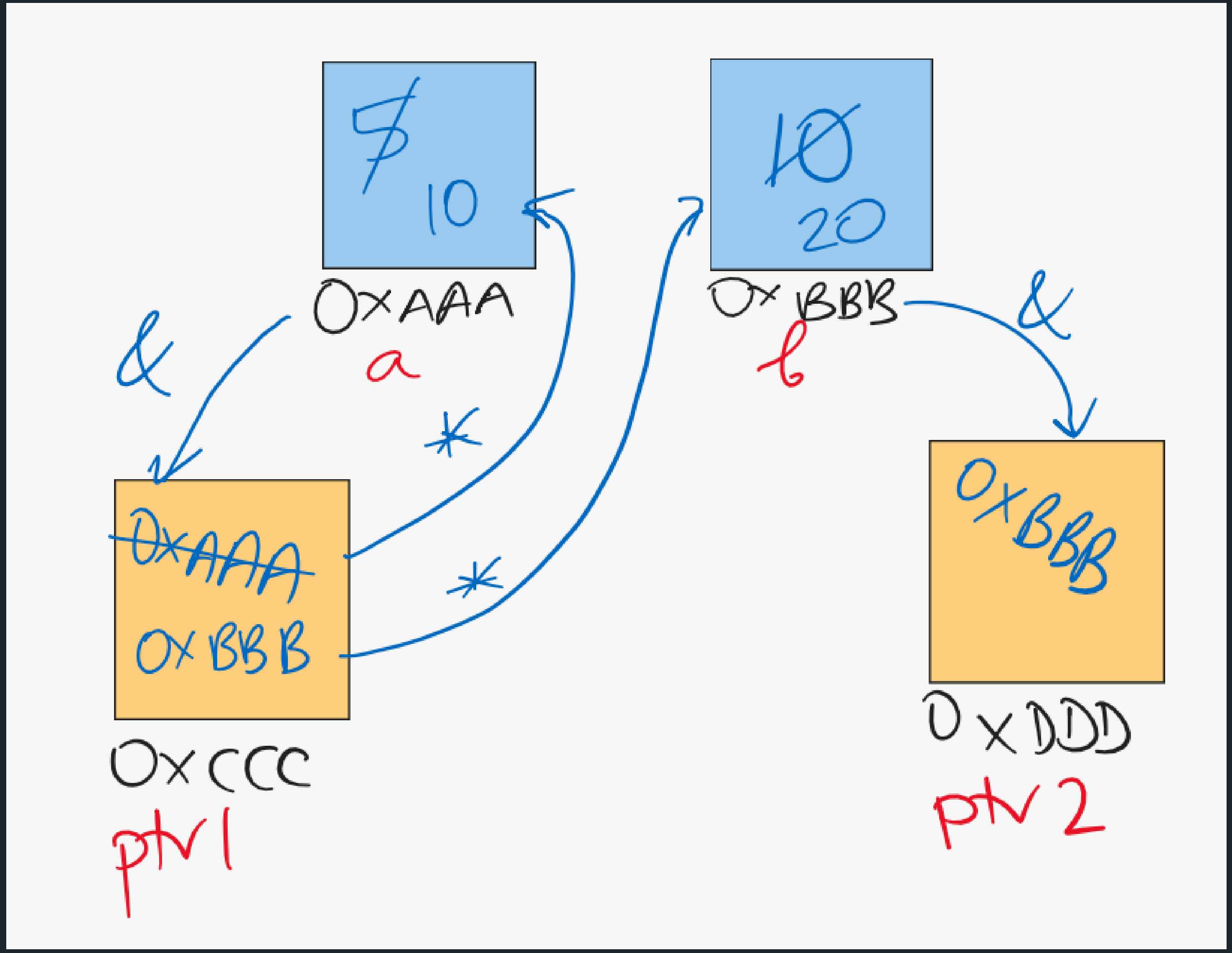
## POINTERS

- Let's see an example:

**`pointer.c`**



```c
int main(void) {
    int a = 5;
    int b = 10;
    int *ptr1;
    int *ptr2;
    ptr1 = &a;
    ptr2 = &b;
    *ptr1 = 10;
    ptr1 = ptr2;
    *ptr1 = 20;
    printf("a = %d\nb = %d\n", a, b);
    return 0;
}
```



## REVISION TIME!

### YOUR TURN FOR POINTERS

- Write some programs using pointers to:
  - Swap two numbers
  - Add two numbers
  - Find the product of two numbers

`pointer2.c`

## REVISION TIME!

### POINTERS

Write a program in C to find the factorial of a given number using pointers.

`pointer_factorial.c`

## REVISION TIME!

## STRINGS

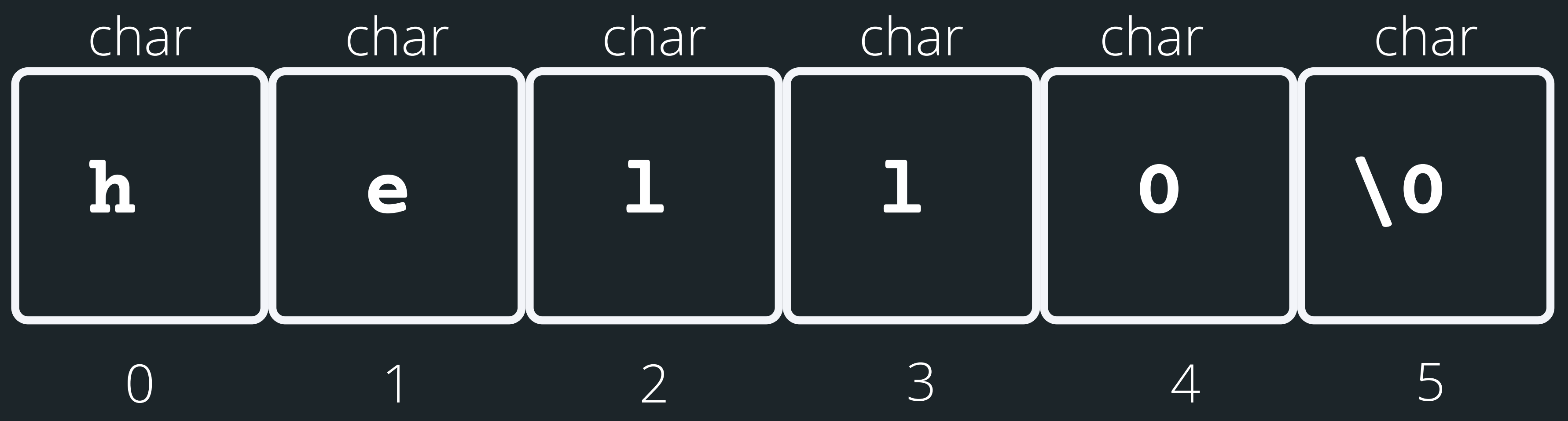
- Strings are a collection of characters that are joined together
  - an array of characters!
- There is one very special thing about strings in C - it is an array of characters that finishes with a
  - This symbol is called a null terminating character
- It is always located at the end of an array, therefore an array has to always be able to accomodate this character
- It is not displayed as part of the string
- It is a placeholder to indicate that this array of characters is a string
- It is very useful to know when our string has come to an end, when we loop through the array of characters

## HOW DO WE DECLARE A STRING?

### WHAT DOES IT LOOK LIKE VISUALLY?

- Because strings are an array of characters, the array type is char.
- To declare and initialise a string, you can use two methods:

```
//the more convenient way
char word[] = "hello";
//this is the same as'\0':
char word[] = {'h','e','l','l','o','\0'};
```



## HELPFUL LIBRARY FUNCTIONS FOR STRINGS

### FGETS()

There is a useful function for reading strings:

**`fgets(array[], length, stream)`**

The function needs three inputs:

- array[] - the array that the string will be stored into
- length - the number of characters that will be read in
- stream - this is where this string is coming from - you don't have to worry about this one, in your case, it will always be stdin (the input will always be from terminal)

```
// Declare an array where you will place the
string that you read from somewhere
char array[MAX_LENGTH];
// Read in the string into array of length
MAX_LENGTH from terminal input
fgets(array, MAX_LENGTH, sdin)
```

## HOW DO I KEEP READING STUFF IN OVER AND OVER AGAIN?

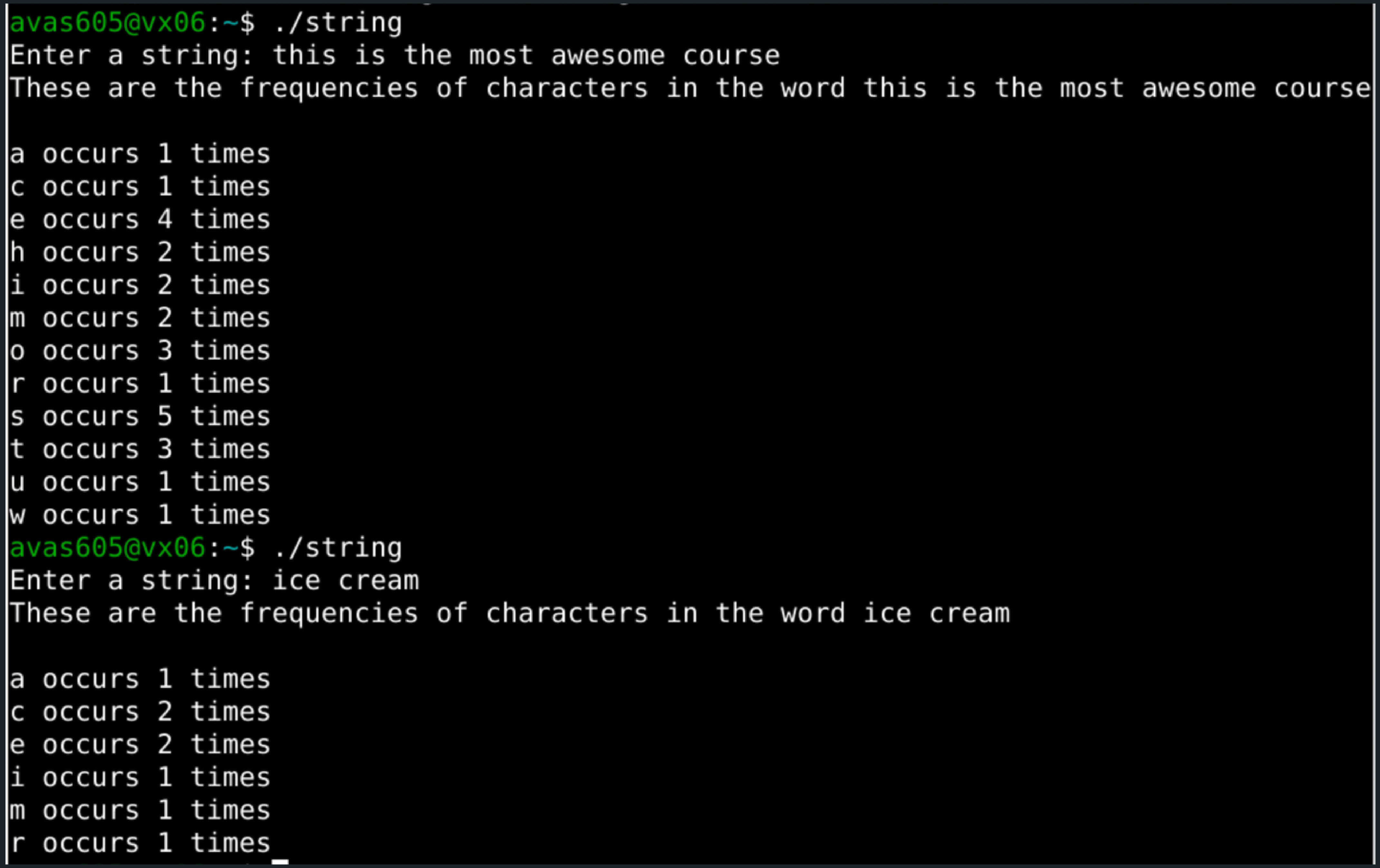
Using the **NULL** keyword, you can continuously get string input from terminal until Ctrl+D is pressed

- fgets() stops reading when either length-1 characters are read, newline character is read or an end of file is reached, whichever comes first

```c
#include <stdio.h>

#define MAX_LENGTH 15

int main(void) {
    // Declare an array where you will place the string
    char array[MAX_LENGTH];

    printf("Type in a string to echo: ");
    // Read in the string into the array until Ctrl+D is
    // pressed, which is indicated by the NULL keyword
    while (fgets(array, MAX_LENGTH, stdin) != NULL) {
        printf("The string is: \n");
        printf("%s", array);
        printf("Type in a string to echo: ");
    }
    return 0;
}
```

## LET'S PLAY!

Write a program that will read in a string from standard input and then count the frequency of each character that is in that string....

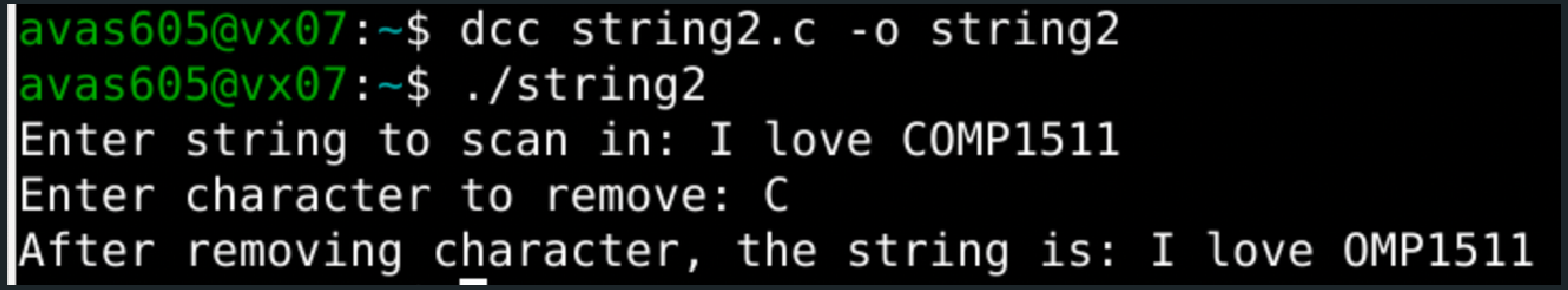


```
avas605@vx06:~$ ./string
Enter a string: this is the most awesome course
These are the frequencies of characters in the word this is the most awesome course

a occurs 1 times
c occurs 1 times
e occurs 4 times
h occurs 2 times
i occurs 2 times
m occurs 2 times
o occurs 3 times
r occurs 1 times
s occurs 5 times
t occurs 3 times
u occurs 1 times
w occurs 1 times
avas605@vx06:~$ ./string
Enter a string: ice cream
These are the frequencies of characters in the word ice cream

a occurs 1 times
c occurs 2 times
e occurs 2 times
i occurs 1 times
m occurs 1 times
r occurs 1 times
```

**string.c**

## YOUR TURN TO PLAY :)

Write a program to take in a string from user and remove the first occurrence of a given character from that string.



```
avas605@vx07:~$ dcc string2.c -o string2
avas605@vx07:~$ ./string2
Enter string to scan in: I love COMP1511
Enter character to remove: C
After removing character, the string is: I love OMP1511
```

**string2.c**

## SOME OTHER INTERESTING STRING FUNCTIONS

### <STRING.H> STANDARD LIBRARY

CHECK OUT THE REST OF THE FUNCTIONS: HTTPS://WWW.TUTORIALSPOINT.COM/C_STANDARD_LIBRARY/STRING_H.HTM

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
- **`strcat()`**attach one string to the end of another (concatenate)
- **`strcmp()`**compare two strings
- **`strchr()`** find the first or last occurance of a character

## THANK YOU

Thank you all so much for tuning in, for learning, for engaging, and I hope that you had an enjoyable intro to programming. Don’t forget that Rome wasn’t built in a day, and becoming a better programmer entails lots of practice!
I really appreciate the engagement that you have shown throughout the lectures, and I wish you all well in the final exam.
Have a wonderful *short* break, I hope you all get some proper down time.

Good Luck in the exam and for your future courses, and I may see some of you again in your later courses :)

## WHAT DID WE LEARN TODAY?

### REVISION: ARRAYS

find_totals.c

array_clamping.c

indivisible.c

### REVISION: POINTERS

pointers.c

### REVISION: STRINGS

string.c

string2.c

## REACH OUT

### CONTENT RELATED QUESTIONS

Check out the forum

### ADMIN QUESTIONS

cs1511@unsw.edu.au