comp1511 week 5

from the comfort of your own home :)

announcements

- we’re online!
- no tute/lab next week (flex week)
- assignment due date is now **MONDAY WK 7**
    - late penalty changed

what's happening today?

- **kahoot**: functions and arrays, intro to pointers, some revision
- quick chat about style for assignment 1
- code review
  - I'm going to split you into **breakout rooms** (get your pets and webcams ready)
  - will also come around and say hi to everyone :)
- 2D arrays
- pointers

kahoot

https://play.kahoot.it/v2/?quizId=c2d6fdc5-0e2c-411d-8b7c-189d2a45189e

code review + common style mistakes

https://docs.google.com/document/d/1czqhZLURzJb9JANq_VMK3RIICKzNLSryD6u8Pb3kiqg/edit?usp=sharing

common style mistakes

- **Header comment** - Make sure you follow the style guide on how to produce a good header comment. For this assignment it is important to have a more in-depth description of your program as it has many features compared to a lab exercise.

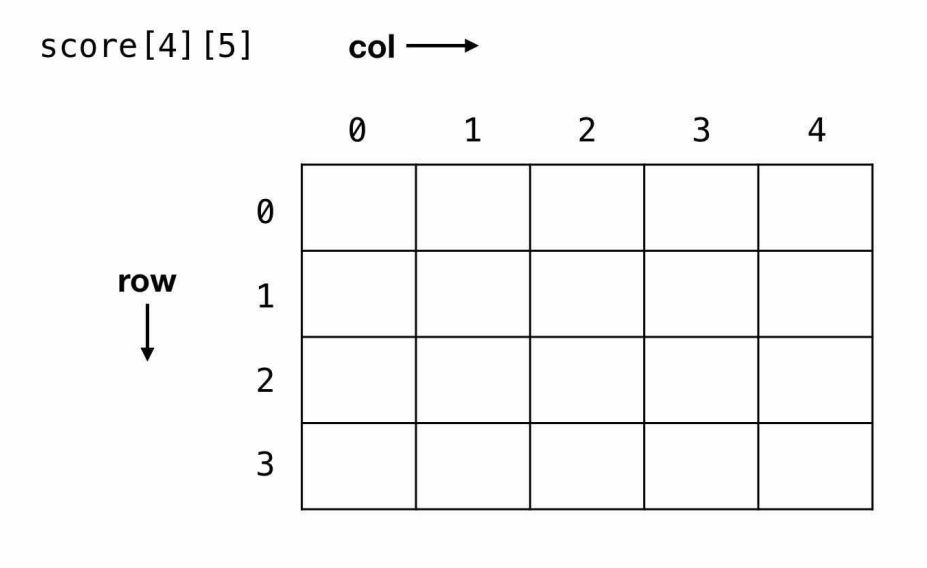
- **Creation and use of #defines** - Make sure you don't have any "magic numbers" in your code. A common issue is not using the "SIZE" #define when looping and checking array elements when needed. You should be able to change the value of "SIZE" and have everything still working (mostly).

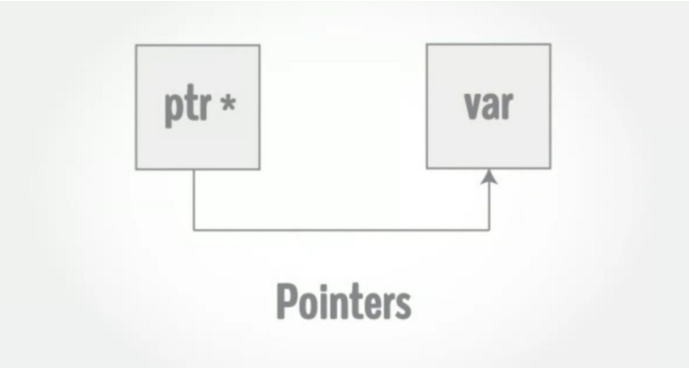
- **Comments** - have comments that explain WHY you have written your code a certain way (eg “check for valid coordinates for the mines”) rather than WHAT your code is doing (eg “increment counter by 1”)

- **Use of functions** - If you're uncomfortable with functions so far and on earlier stages then you shouldn't have to worry too much here, but as you get into later stages, they are very important.

- The above are the main observed issues but make sure you check out the style guide for all our rules.

2D Arrays





Pointers

variables that hold memory addresses
of other variables

what is the point of pointers?????

what is the point of pointers?

You can modify the original variable instead of making a copy

- CONSIDER: if you want to find a specific person in a building
- Passing a pointer around gives the illusion of pass by reference in C
- multiply_by_five.c

You can change more than one variable

- You can’t return more than one variable with functions
- Lab exercises

Basis for data structures like linked lists (coming soon!)

declaring and initialising a pointer

code: pointers.c
common error: null.c

pointer syntax: what happens when each of the following statements are executed in order?

```
int n = 42;
int *p;
int *q;
p = &n;
*p = 5;
*q = 17;
q = p;
*q = 8;
```

why did we have to always include the & symbol in our argument given to scanf?

why did we have to always include the & symbol in our argument given to scanf?

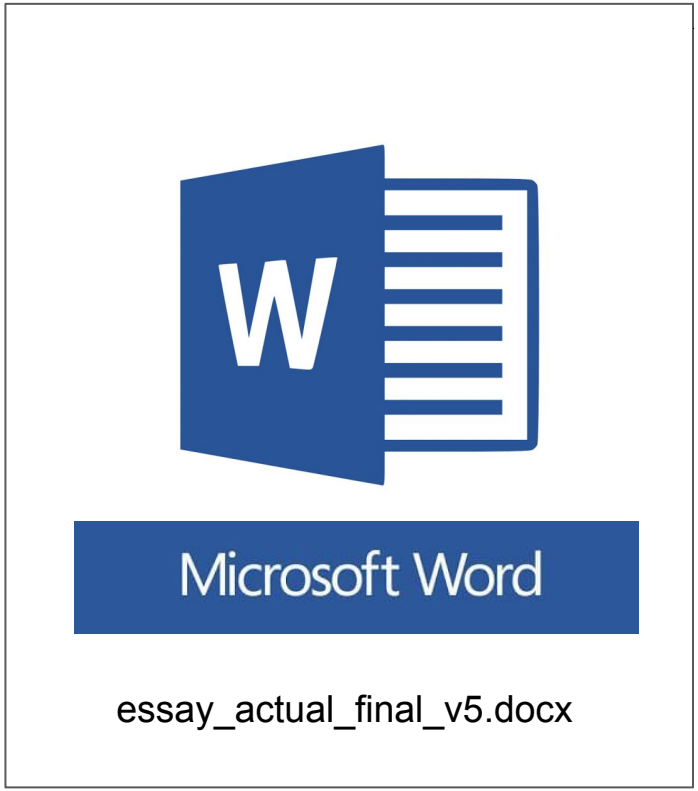
Non-pointer variables in C are **pass by value**

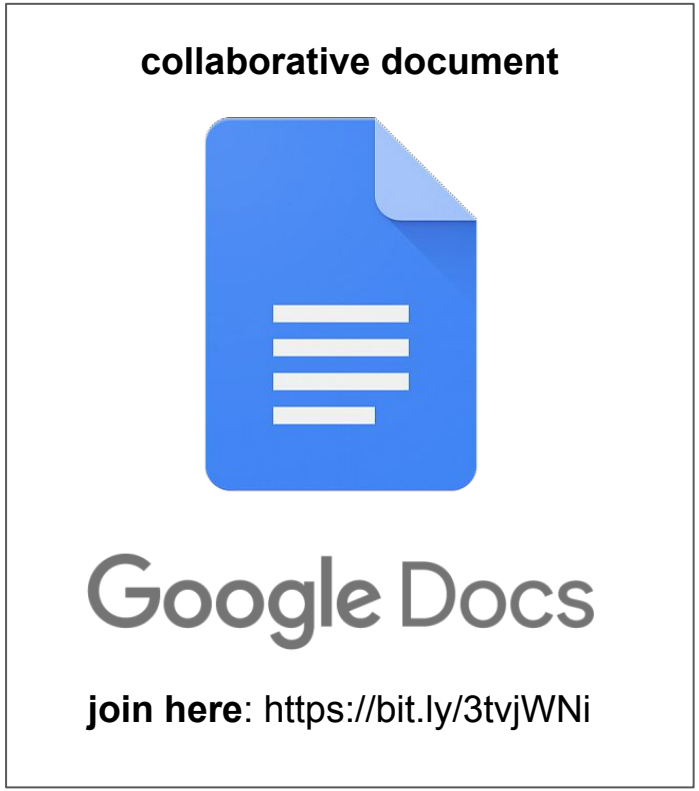
- Eg. Giving a regular variable to scanf without the & symbol
- scanf can't change that value

The & symbol gives the address of the variable instead to scanf

- scanf can directly access that piece of **memory** that the variable occupies and directly modify the variable
- this behaviour is called **pass by reference**

pass by value vs pass by reference





code demos

- changinging the original variable from the function
  - multiply_by_five..c
- dereferencing NULL
  - null.c
- arrays are actually pointers
  - arrays.c

optional:

- another pointer example, mainly to go over syntax
  - max.c

trying to learn pointer syntax:

Which of the following functions are possible to write.

If they are not possible to write, what would you do to make them work?

1. `int array_length(int nums[]);`

   which returns the number of elements in the array `nums`.

2. `int test_all_positive(int nums[]);`

   which returns 1 if all elements of array `nums` are positive, otherwise returns 0.

3. `int test_all_initialized(int length, int nums[]);`

   which returns 1 if all elements of array `nums` are initialized, otherwise returns 0.

4. `int test_all_positive(int length, int nums[]);`

   which returns 1 if all elements of array `nums` are positive, otherwise returns 0.

It is only possible to write the last function in C.

To write the first, or second functions, programmers usually do one of 3 things.

1. Pass the **array length** as another parameter to the function.
2. Use a **special value** in an array element to mark the finish of the array - e.g. 0 if the array need contain only positive `int`s
3. Pass an array of a **specific length** to the function - e.g. always pass arrays of 20 elements

For functions you write in this course, you should opt for option (a), as demonstrated by the fourth function.

Since it is not possible to tell if memory is uninitialised, it is impossible to write the third function. The only defence against uninitialised memory is to ensure you **initialise all variables and arrays.**