comp1511 week 5

starting ~5 past the hour

announcements

- no tute/lab next week (flex week)
- assignment 1 exists :)

what’s happening today?

- Valid Functions
- 2D arrays
- Pointers

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.

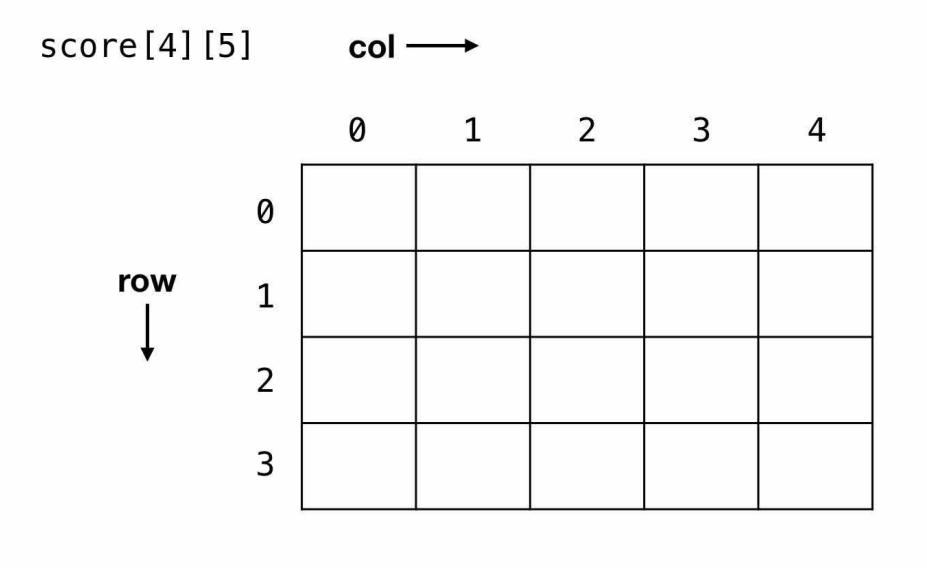
3. `int test_all_initialized(int length, int nums[]);`

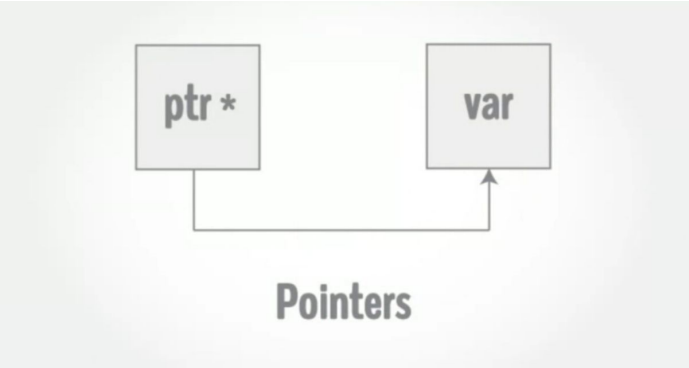
   which returns 1 if all elements of array `nums` are initialized, otherwise returns 0.

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

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

2D Arrays





Pointers

variables that hold memory addresses
of other variables

declaring and initialising a pointer

code: pointers.c
common error: null.c

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?

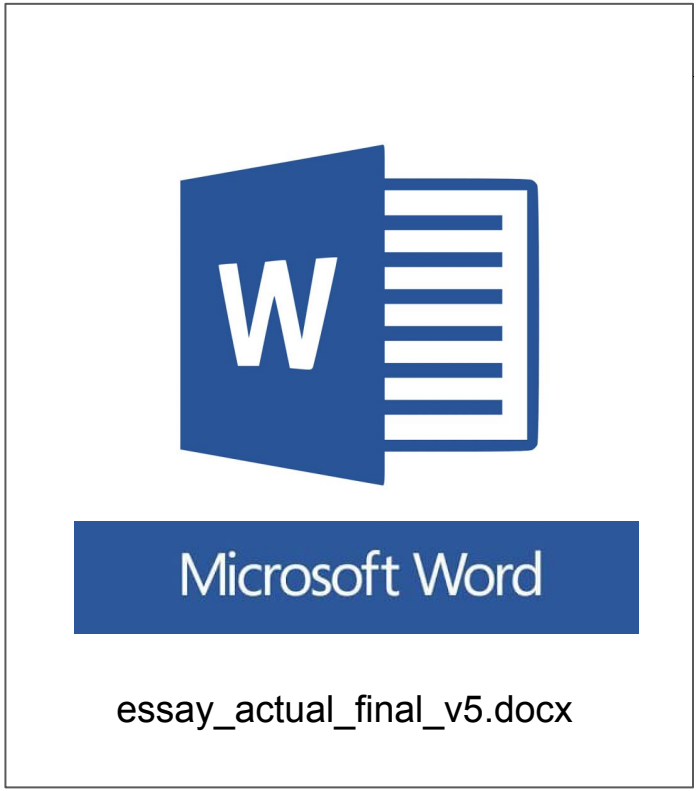
Non-pointer variables in C are **pass by value**

- 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



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

https://cgi.cse.unsw.edu.au/~cs1511/21T3/tut/05/questions

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

code demos

- changing the original variable from the function
  - sum_nums.c
- dereferencing NULL
  - null.c

trying to learn pointer syntax: