

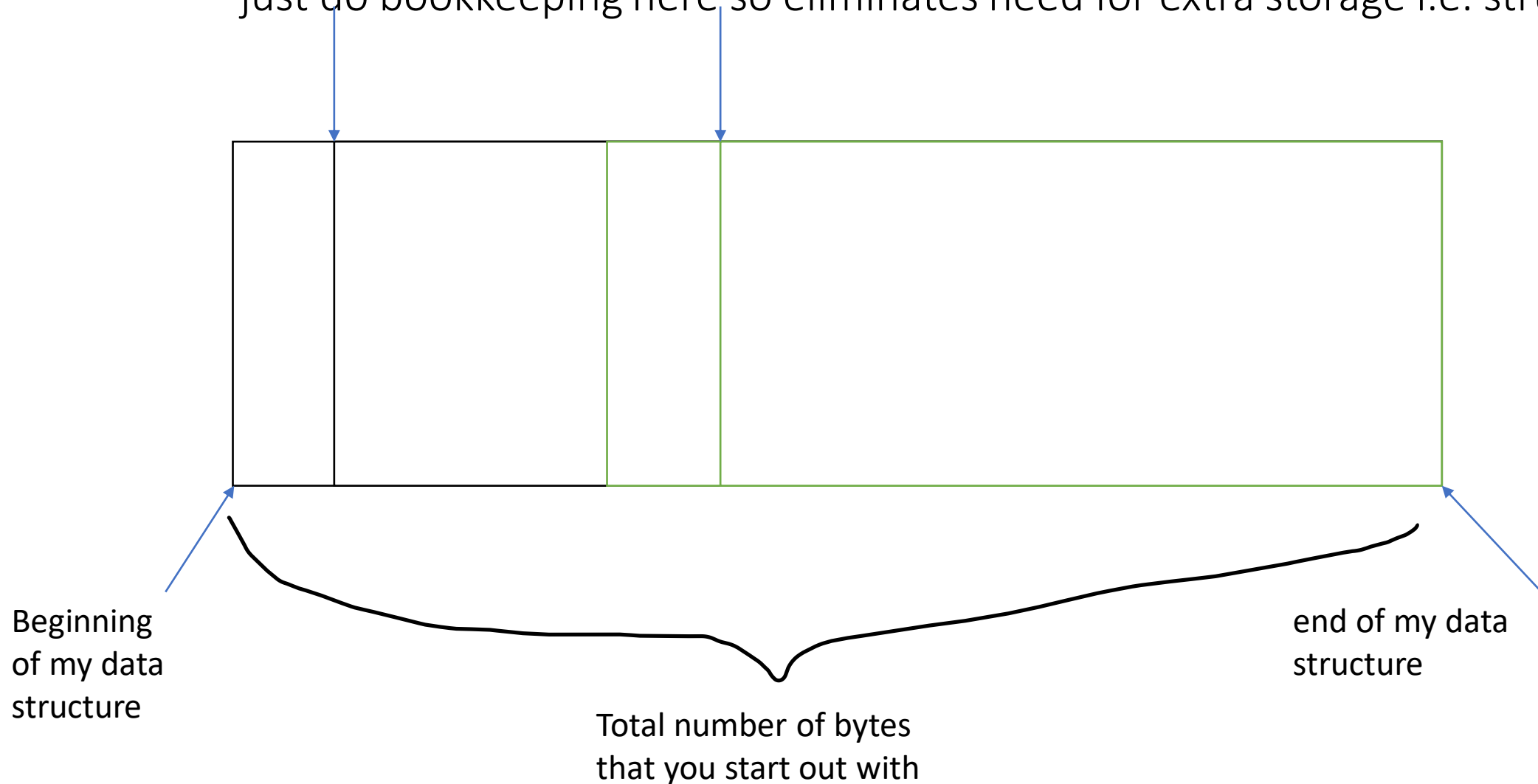
Semester Project
Shira Orlian

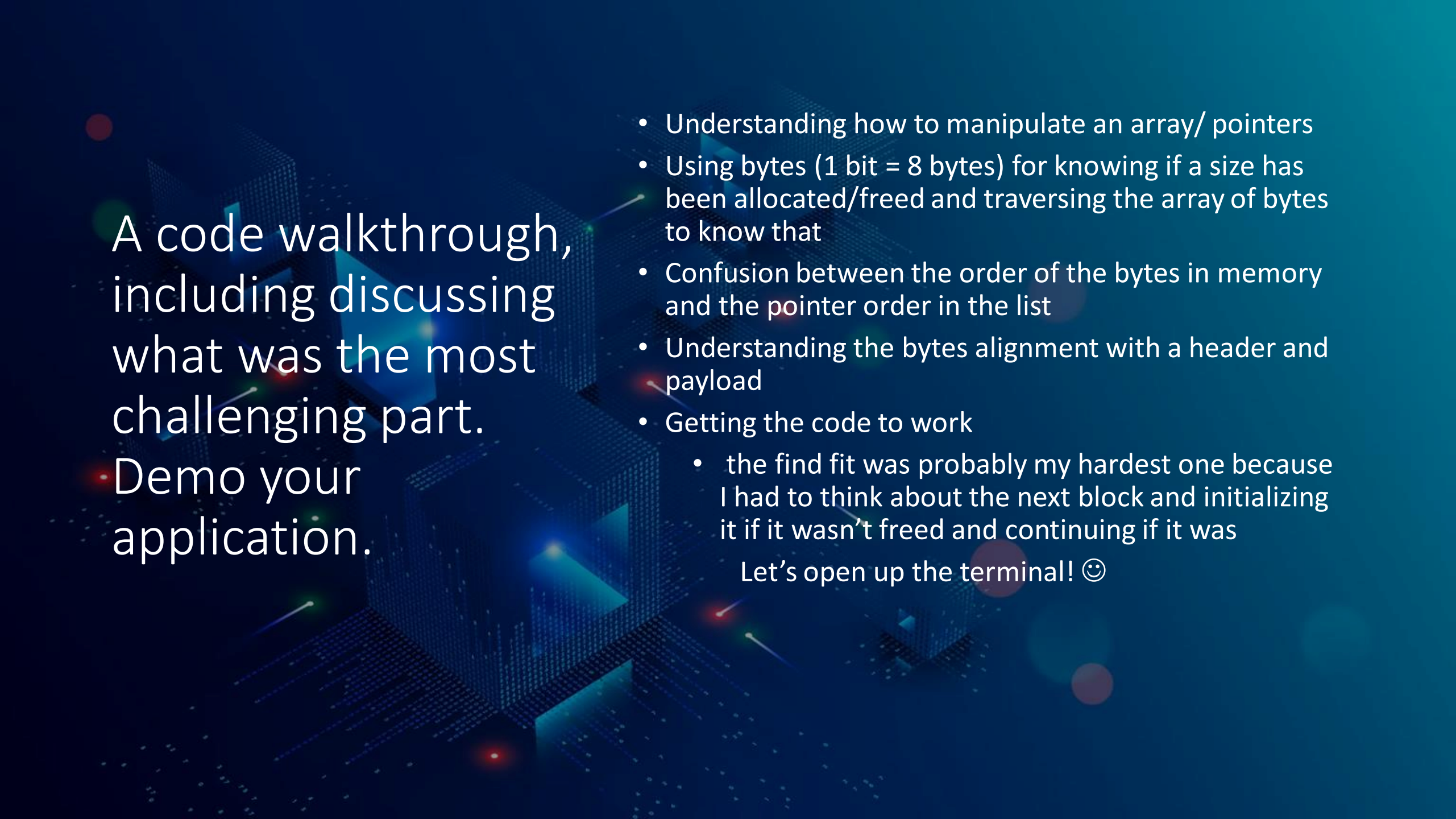
Memory Allocator – Code in C

<https://github.com/whoa-hash/project>

Data Structure: Implicit List with Block – in size bytes

- just do bookkeeping here so eliminates need for extra storage i.e. struct vs byte





A code walkthrough,
including discussing
what was the most
challenging part.

Demo your
application.

- Understanding how to manipulate an array/ pointers
- Using bytes (1 bit = 8 bytes) for knowing if a size has been allocated/freed and traversing the array of bytes to know that
- Confusion between the order of the bytes in memory and the pointer order in the list
- Understanding the bytes alignment with a header and payload
- Getting the code to work
 - the find fit was probably my hardest one because I had to think about the next block and initializing it if it wasn't freed and continuing if it was

Let's open up the terminal! 😊

What did you learn?

You can get very confused because of the different types in C (vs Python) – if you want to loop by 1 byte in an array that's different than looping by 1 int - If you loop with an int vs a char you're looping by int bytes and not 1 byte like you think

Coding in C is tricky with pointers and dereferencing.

What was the most fun part?

Seeing my code run in the terminal. Making up test cases that failed =) appropriately

What was the hardest part?

Understanding the task and coding it when I have barely used C



A performance analysis of the code:

- Malloc
 - Average - without the merging of free blocks $O(n)$ where n is the number of blocks : because it has to search through all the blocks. With the merging, it depends how many freed blocks there are afterwards
 - Worst $O(n)$: going to the last block that I have
 - Best – if it finds a spot right away- which means there's a free spot at the beginning of the heap – we can call that $O(1)$
- Free
 - They're all $O(1)$ because all you do if it exists is set the allocation bit to 0
 - Average
 - Worst
 - best