COMP1511 PROGRAMMING FUNDAMENTALS

Lecture 12

Linked Lists: inserting into a linked list and traversing a linked list

YESTERDAY...

- Multi-file projects
- A bit more about memory
- Slow intro to linked lists

TODAY...

- Linked lists
  - Adding to a linked list
  - Traversing through a linked list

WHERE IS THE CODE?

LIVE LECTURE CODE CAN BE FOUND HERE:

https://cgi.cse.unsw.edu.au/~cs1511/21T3/live/Week07/

LINKED LISTS

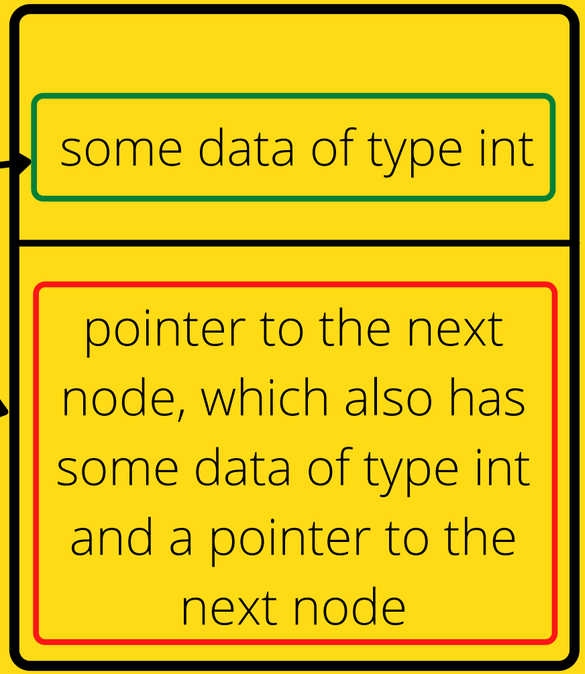
A NODE

- Linked list is made up of many nodes
- Each node has some data and a pointer to the next node, creating a linked structure that forms the list

a node

[contains some data and also a pointer to the next node of the same data type]

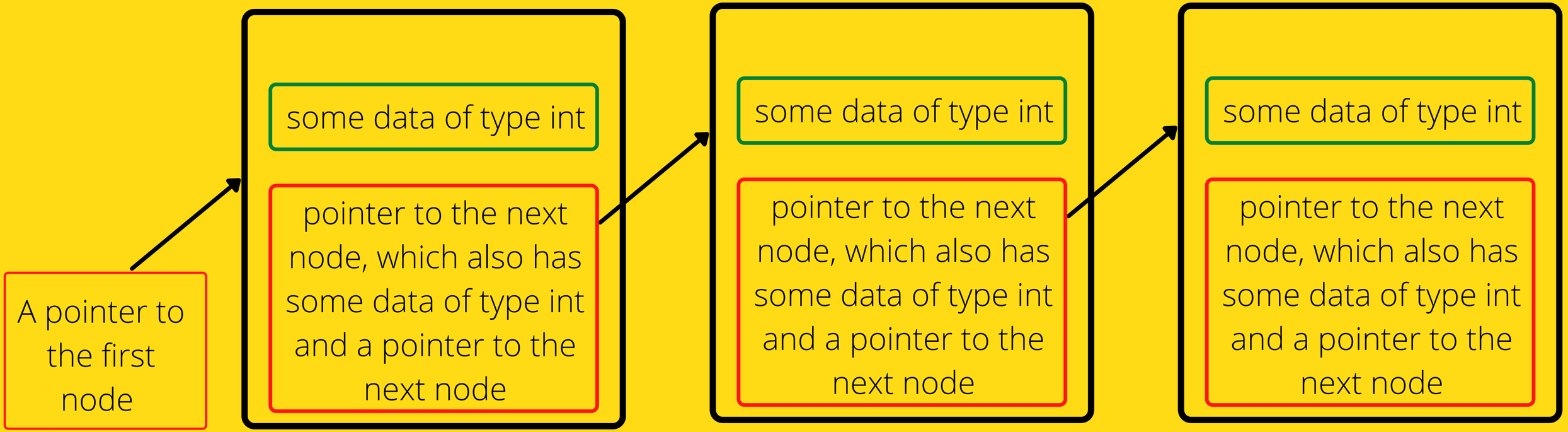
```
struct node {
    int data;
    struct node *next;
}
```



LINKED LISTS

MANY NODES

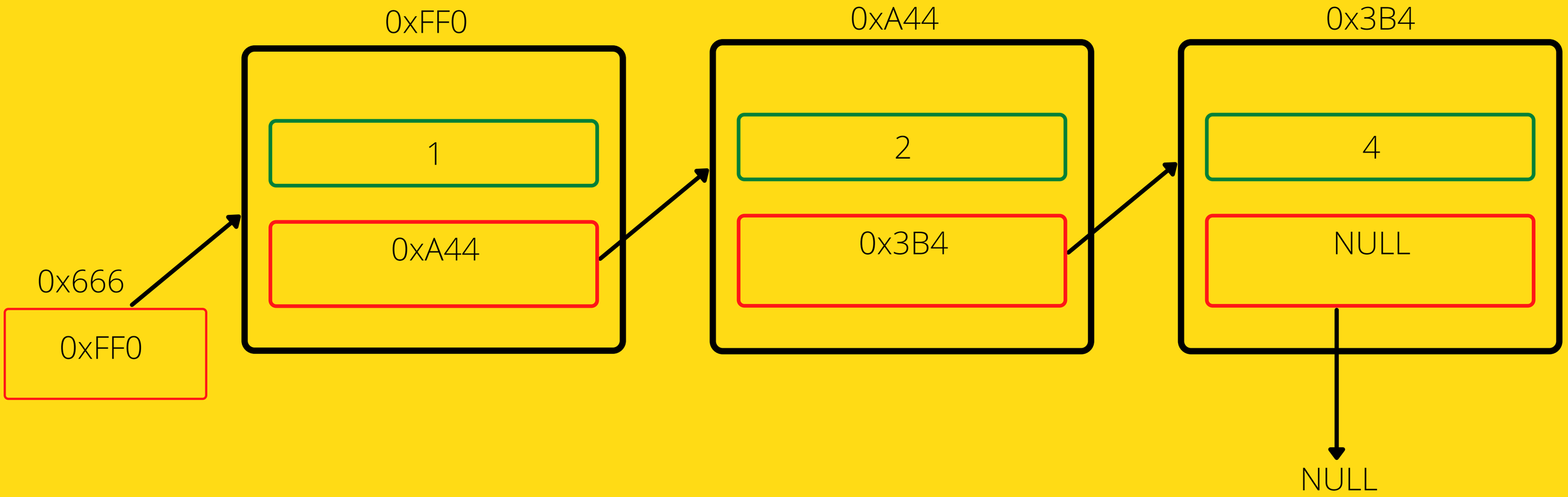
- We can create a linked list, by having many nodes together, with each struct node next pointer giving us the address of the node that follows it



LINKED LISTS

AN EXAMPLE

- Let's say we have a list with numbers:
  1, 2, 4
- How will this look in a linked list structure?



LINKED LISTS

WHY?

- Linked lists are dynamically sized, that means we can grow and shrink them as needed - efficient for memory!
- Elements of a linked list (called nodes) do NOT need to be stored contiguously in memory, like an array.
- We can add or remove nodes as needed anywhere in the list, without worrying about size (unless we run out of memory of course!)
- We can change the order in a linked list, by just changing where the next pointer is pointing to!
- Unlike arrays, linked lists are not random access data structures! You can only access items sequentially, starting from the beginning of the list.

LINKED LISTS

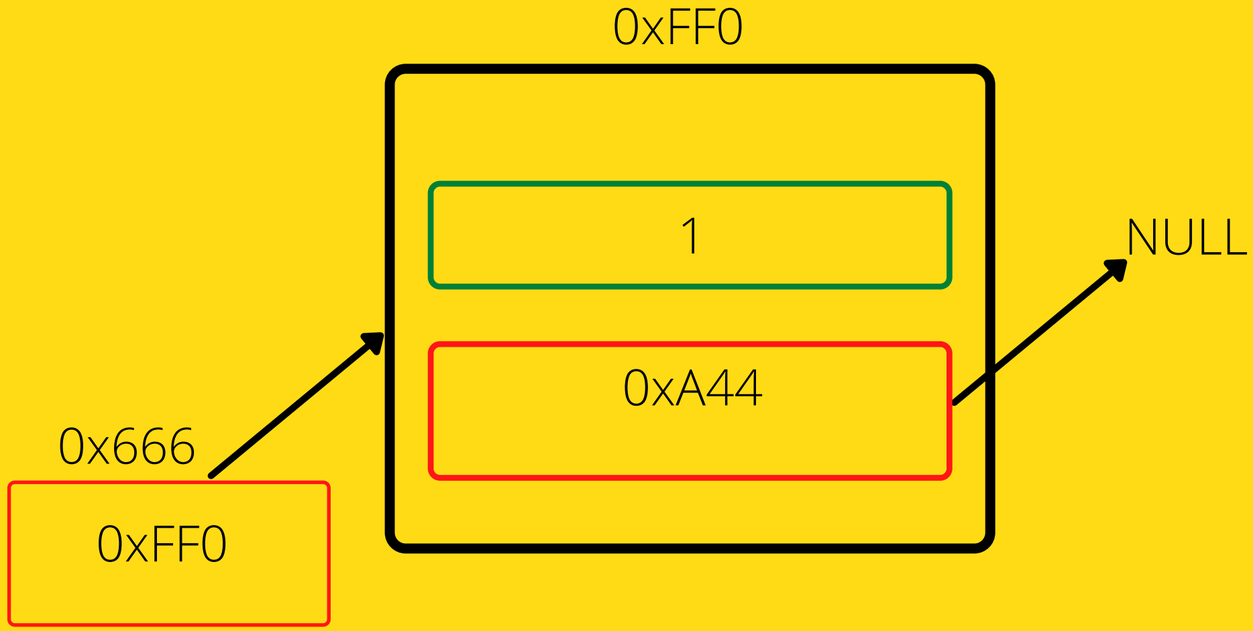
HOW WOULD WE CREATE AND INSERT INTO A LINKED LIST?

- In order to create a linked list, we would need to
  - Define our struct for a node,
  - A pointer to keep track of where the start of the list is and
  - A way to create a node and then connect it into our list...

LINKED LISTS

HOW WOULD WE CREATE AND INSERT INTO A LINKED LIST? LET'S SEE IT VISUALLY FIRST AND THEN TRANSLATE TO CODE

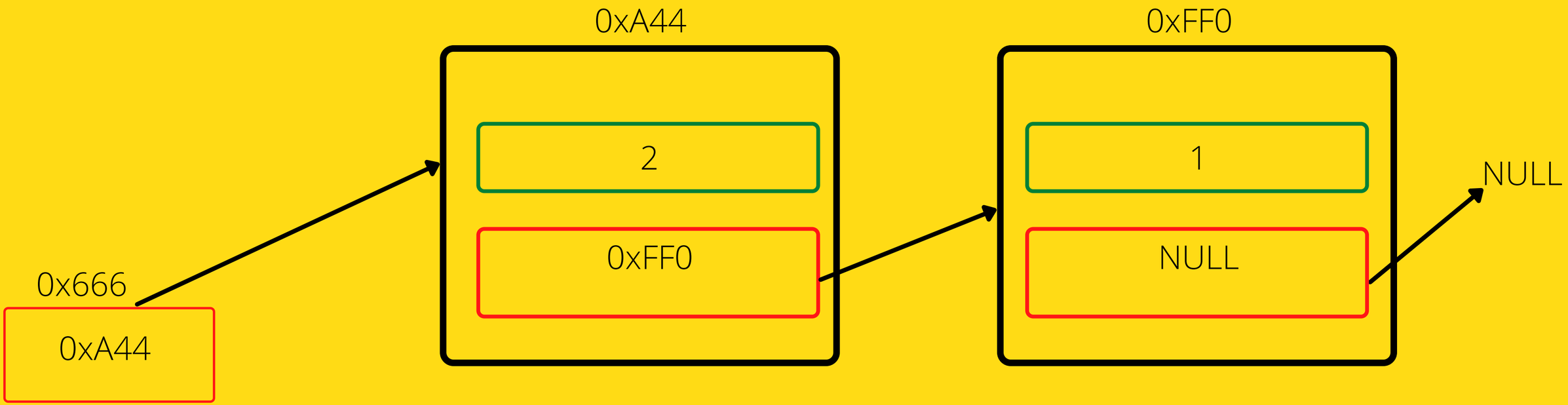
- Let's say we wanted to create a linked list with 1, 2, 4
  - A pointer to keep track of where the start of the list is and by default the first node of the list
  - It will point to NULL as there are no other nodes in this list.



LINKED LISTS

HOW WOULD WE CREATE AND INSERT INTO A LINKED LIST? LET'S SEE IT VISUALLY FIRST AND THEN TRANSLATE TO CODE

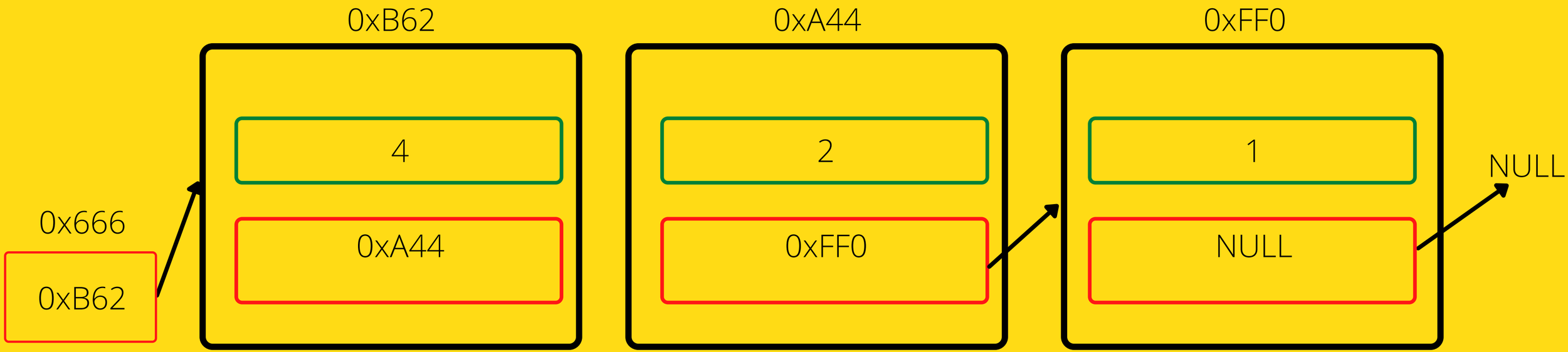
- Create the next node to store 2 into (you need memory)
- Assign 2 to data
- and insert it at the beginning so the head would now point to it and the new node would point to the old head



LINKED LISTS

HOW WOULD WE CREATE AND INSERT INTO A LINKED LIST? LET'S SEE IT VISUALLY FIRST AND THEN TRANSLATE TO CODE

- Create the next node to store 4 into (you need memory)
- Assign 4 to data
- and insert it at the beginning so the head would now point to it and the new node would point to the old head



LINKED LISTS

DEFINE OUR STRUCT FOR A NODE

- Define our struct for a node:

```
struct node {
    int data;
    struct node *next;
}
```

LINKED LISTS

A POINTER TO THE START OF THE LIST

- A pointer to keep track of where the start of the list is:
  - The pointer would be of type struct node, because it is pointing to the first node
  - The first node of the list is often called the 'head' of the list (last element is often called the 'tail')

```
//head will point to the first element of our list
struct node *head =
```

LINKED LISTS

A POINTER TO THE START OF THE LIST

- A way to create a node and then connect it into our list...
  - Create a node by first creating some space for that node (malloc)
  - Initialise the data component on the node
  - Initialise where the node is pointing to

```c
//Function that creates a node
// Inputs: data of type int and a pointer to the next node
// Output: a pointer to a node
struct node *create_node(int data, struct node *next) {
    //create a pointer to a node of type struct node
    //by using malloc to allocate memory
    struct node *n = malloc(sizeof(struct node));
    //Initialise the data by setting it to the given int
    n->data = data;
    //Initialise next pointer to point to the next node given
    n->next = next
    return n;
}
```

LINKED LISTS

PUTTING IT TOGETHER

linked_list.c

```c
//Define a node
struct node {
    int data;
    struct node *next;
};

int main (void) {
    //Create a pointer to the first node of the list, and the first node
    //The first node is pointing to NULL as the next node, since there are
    //no other nodes
    struct node *head = create_node(1, NULL);
    //Create the next node, with data 2 and pointing next to the current head
    //Therefore you are making this new node the head of the list
    head = create_node(2, head);
    //Create the next node, with data 4 and pointing next to the current head
    //Therefore you are making this new node the head of the list
    head = create_node(4, head);

    return 0;
}

//Function that creates a node
// Inputs: data of type int and a pointer to the next node
// Output: a pointer to a node
struct node *create_node(int data, struct node *next) {
    //create a pointer to a node of type struct node
    //by using malloc to allocate memory
    struct node *n = malloc(sizeof(struct node));
    //Initialise the data by setting it to the given int
    n->data = data;
    //Initialise next pointer to point to the next node given
    n->next = next
    return n;
}
```

BREAK TIME (5 MINUTES)

You have five boxes in a row numbered 1 to 5, in one of which, a cat is hiding. Every night he jumps to an adjacent box, and every morning you have one chance to open a box to find him. How do you win this game of hide and seek - what is your strategy? What if there are n boxes?

TRAVERSING A LINKED LIST

MAKING OUR WAY THROUGH THE LIST

```
void print_list(struct node *head) {
    struct node *current = head;
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("X\n");
}
```

- The only way we can make our way through the linked list is like a scavenger hunt, we have to follow the links from node to node (sequentially! we can't skip nodes)
- We have to know where to start, so we need to know the head of the list
- When we reach the NULL pointer, it means we have come to the end of the list.

LET'S MAKE IT A MULTI FILE PROJECT

linked_list.c
linked_list.h
main.c

- Let's see this converted into a multi file project...
- Remember, in the header file (linked_list.h)
  - #defines
  - function prototypes for the functions that will be implemented in the implementation file
  - comments that describe how the functions will be used
- In the implementation file (linked_list.c):
  - implementation of functions defined in the header file
  - #include "linked_list.h"
- In the main.c file (#include "linked_list.h")
  - Call functions that exist in other modules
  - Keep code that actually implements anything to a MINIMUM

PROBLEM TIME

It's a FIFA World Cup year (I wish, but not long to wait!)
To celebrate, I have want to start adding countries to the list as they qualify for the world cup. I will then use the list to keep track of who has been knocked out of the competition and who is still playing.

1. I need to create a list, to which I can add all the participating countries
2. I want to be able to print out this list
3. The countries will start in random order

Next week, we will give our list some order, and look at knocking countries out of the competition

FEEDBACK?

PLEASE LET ME KNOW ANY FEEDBACK FROM TODAY'S LECTURE!

www.menti.com

Code: 6414 7214

WHAT DID WE LEARN TODAY?

INTRO TO LINKED LIST: INSERT AT HEAD AND TRAVERSE

linked_list.c

MAKE IT A MULTI FILE

linked_list.h
linked_list.c
main.c

A HARDER PROBLEM

world_cup_prep.c

ANY QUESTIONS?

DON'T FORGET YOU CAN ALWAYS EMAIL US ON CS1511@CSE.UNSW.EDU.AU FOR ANY ADMIN QUESTIONS

PLEASE ASK IN THE FORUM FOR CONTENT RELATED QUESTIONS