COMP1511 PROGRAMMING FUNDAMENTALS

LECTURE 10

struct pointers and Linked Lists

WHERE IS THE CODE?

Live lecture code can be found here:

HTTPS://CGI.CSE.UNSW.EDU.AU/~CS1511/22T2/LIVE/WEEK05/

STRUCTS AND POINTERS

- Remember that when we access members of a struct we use a .

SO FAR...

- We have used variables which store values
- Variables which store a pointer to a value

BUT WHAT IF...

I wanted to store a pointer to a struct?

STRUCTS AND POINTERS

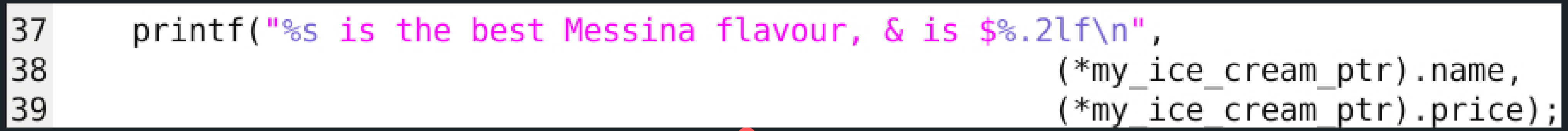
-> VERSUS .

- What happens if we make a pointer of type struct? How do we access it then?

STRUCTS AND POINTERS

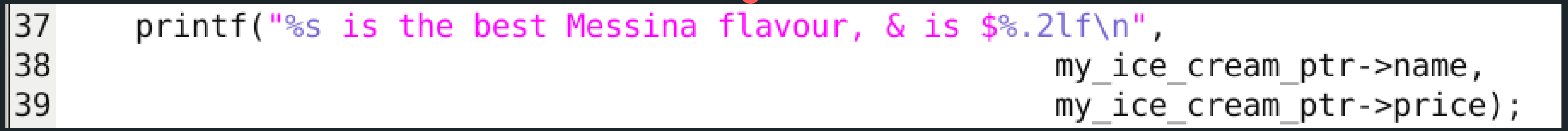
-> VERSUS .

- Those brackets can get quite confusing, so there is a shorthand way to do this with an ->
- There is no need to use (*my_ice_cream_ptr) and instead can just straight my_ice_cream_ptr ->



```
    printf("%s is the best Messina flavour, & is $%.2lf\n",
                                                                (*my_ice_cream_ptr).name,
                                                                (*my_ice_cream_ptr).price);
```

shorthand way - much simpler to use and harder to make mistakes when you use ->



```
    printf("%s is the best Messina flavour, & is $%.2lf\n",
                                                                my_ice_cream_ptr->name,
                                                                my_ice_cream_ptr->price);
```

SO FAR

- 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.

SO FAR

- We have arrays to store multiple values
- A great data structure!
- But they do have a problem... (demo)

BUT WHAT IF

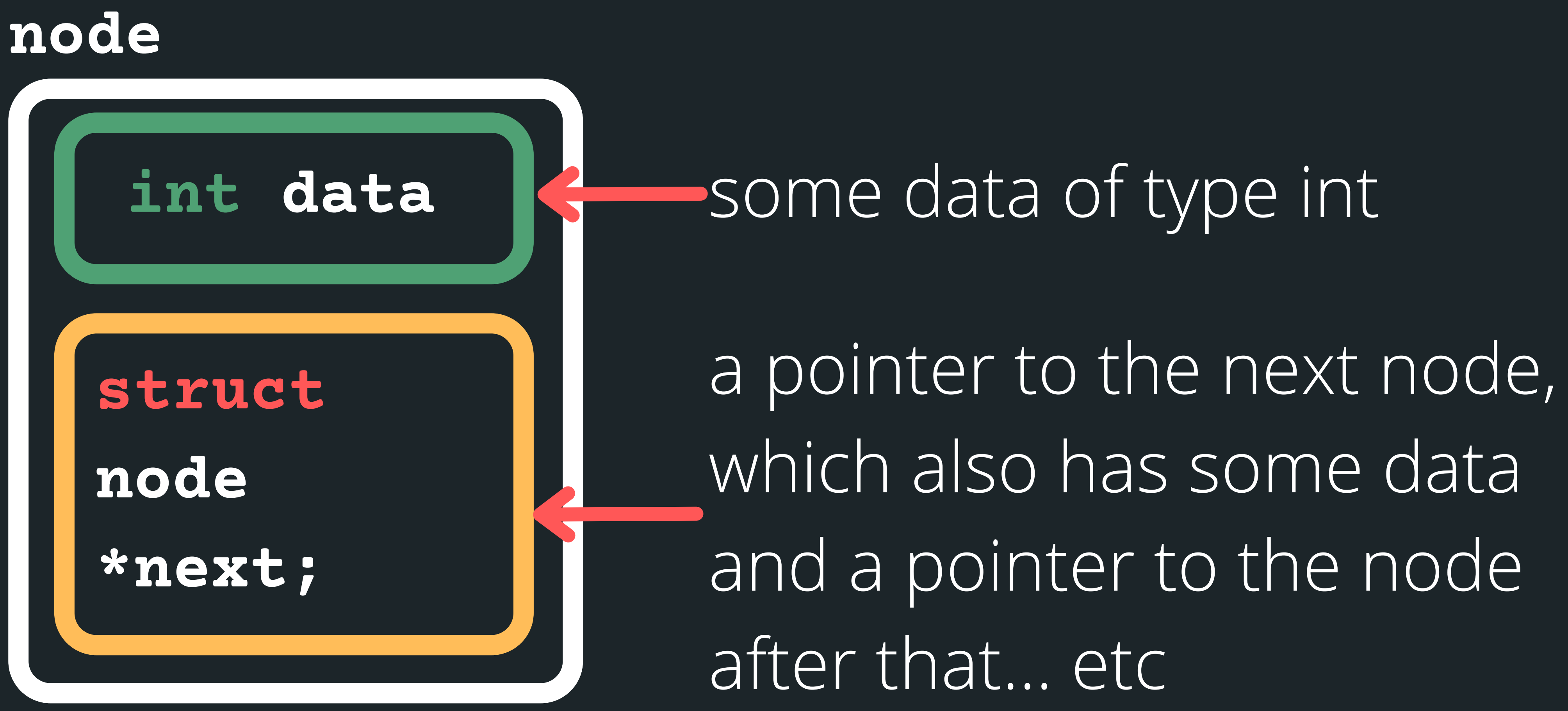
- We wanted more flexibility over the order of our collections?

A LINKED LIST IS MADE UP OF NODES

WHAT IS A NODE?

- Each node has some data and a pointer to the next node (of the same data type), creating a linked structure that forms the list
- Let me propose a node structure like this:

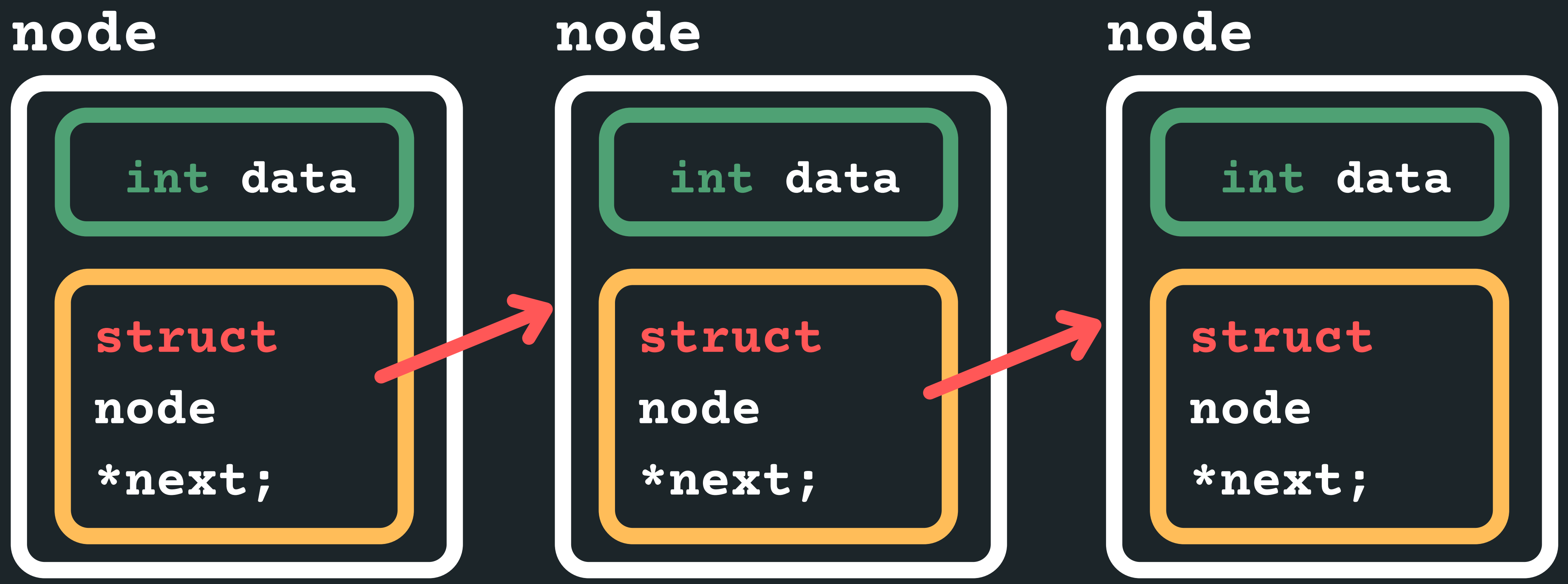
```
struct node {
    int data;
    struct node *next;
};
```


node
int data
struct node *next;
some data of type int
a pointer to the next node, which also has some data and a pointer to the node after that... etc

A LINKED LIST IS MADE UP OF MANY NODES

THE NODES ARE LINKED TOGETHER (A SCAVENGER HUNT OF POINTERS)

- 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

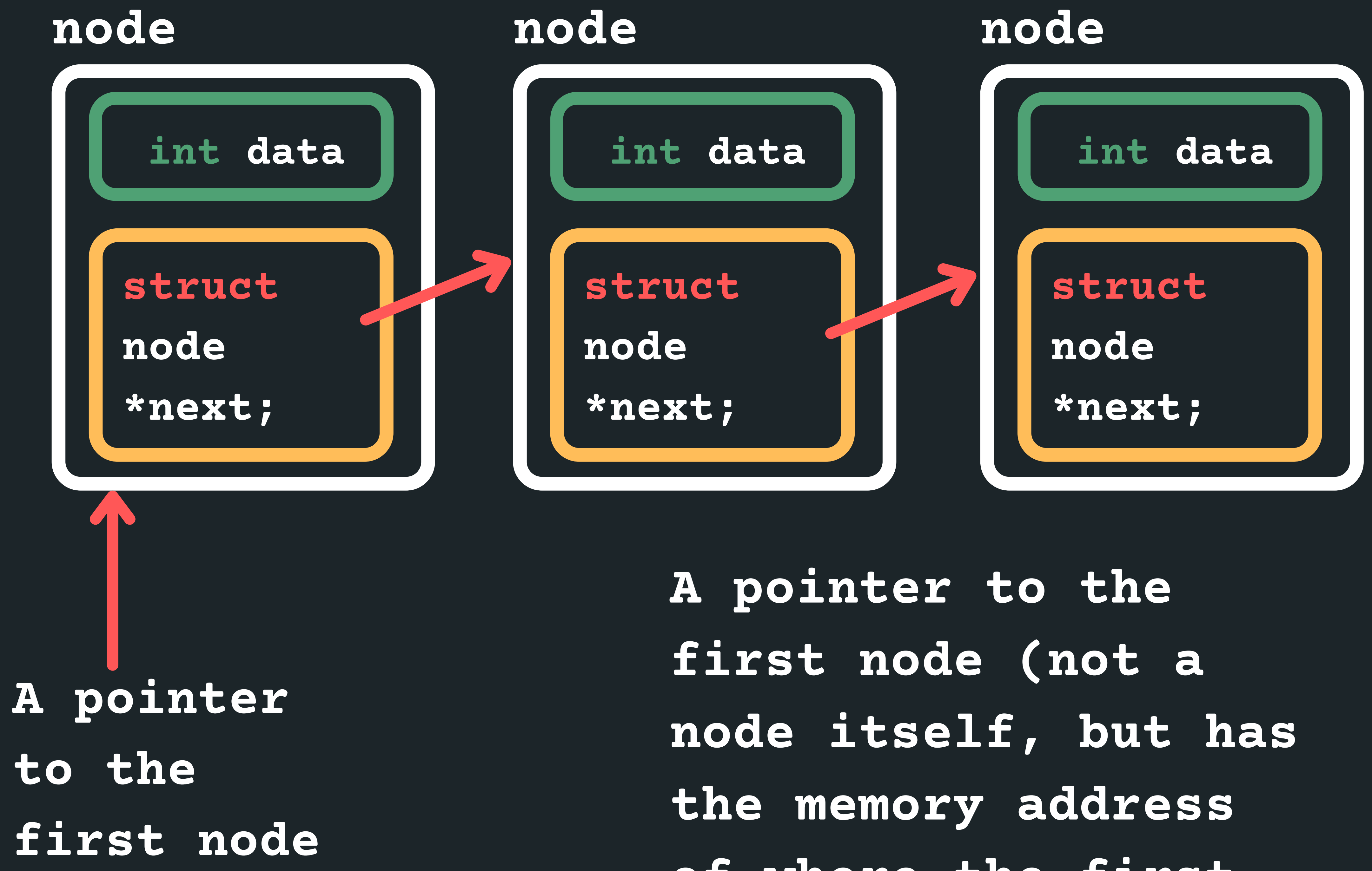


- But how do I know where the linked list starts?

A LINKED LIST IS MADE UP OF MANY NODES

THE NODES ARE LINKED TOGETHER (A SCAVENGER HUNT OF POINTERS)

- What about a pointer to the first node?



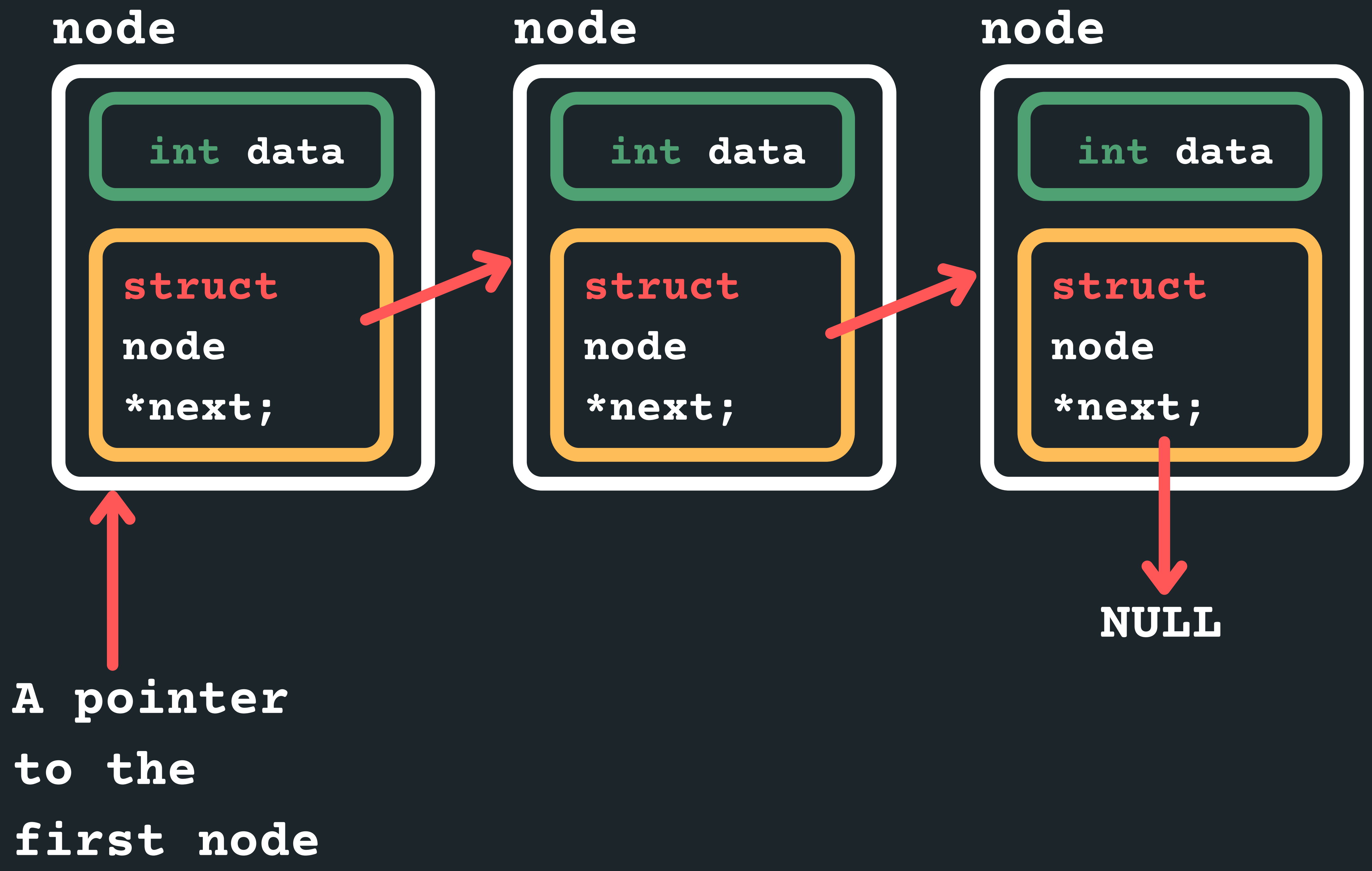
A pointer to the first node (not a node itself, but has the memory address of where the first node is!

- How do I know when my list is finished?

A LINKED LIST IS MADE UP OF MANY NODES

THE NODES ARE LINKED TOGETHER (A SCAVENGER HUNT OF POINTERS)

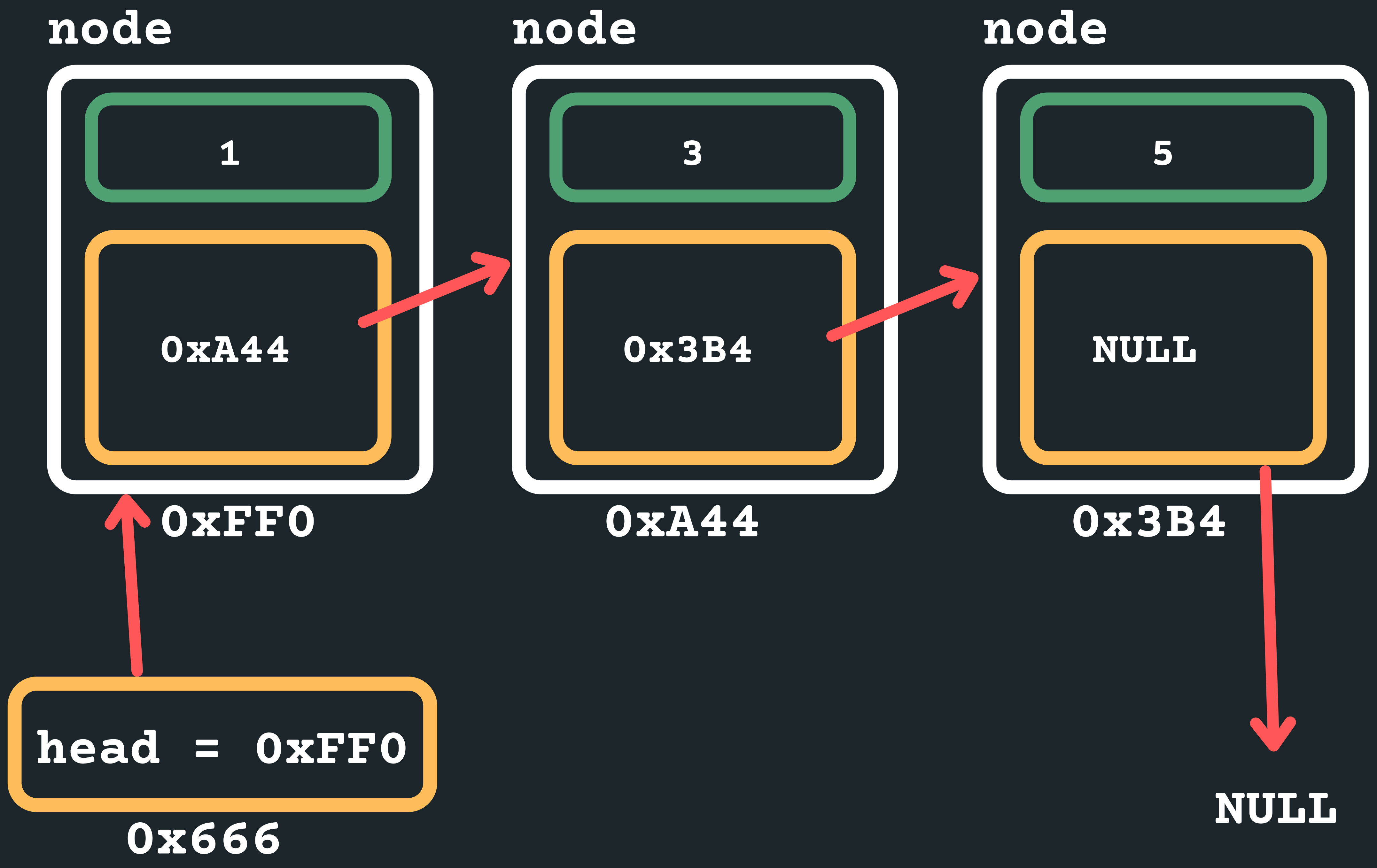
- Pointing to a NULL at the end!



A LINKED LIST IS MADE UP OF MANY NODES

THE NODES ARE LINKED TOGETHER (A SCAVENGER HUNT OF POINTERS)

- For example, a list with: 1, 3, 5



A LINKED LIST

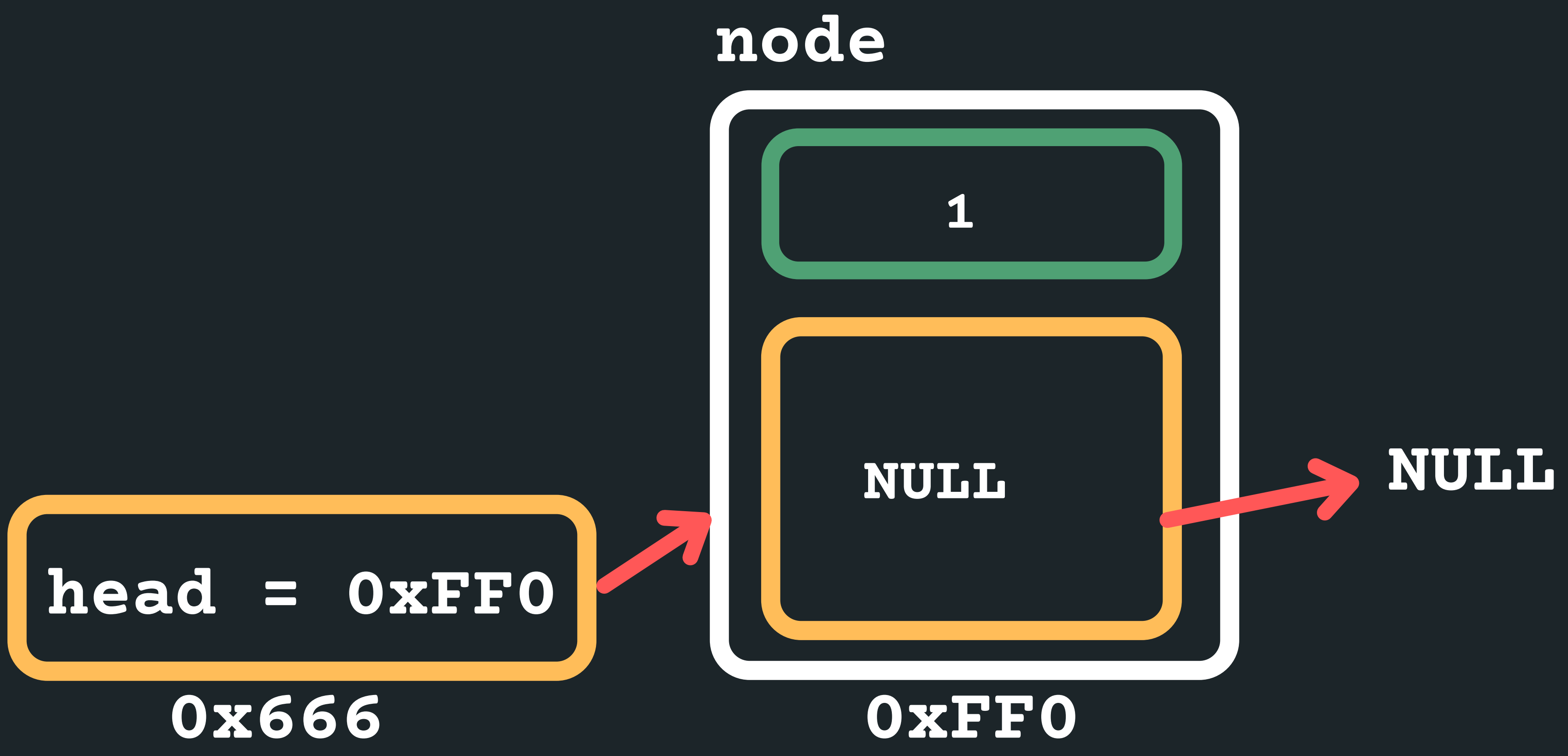
HOW DO WE CREATE ONE AND INSERT

- In order to create a linked list, we would need to
  - Define 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…

A LINKED LIST

HOW DO WE CREATE ONE AND INSERT INTO IT?

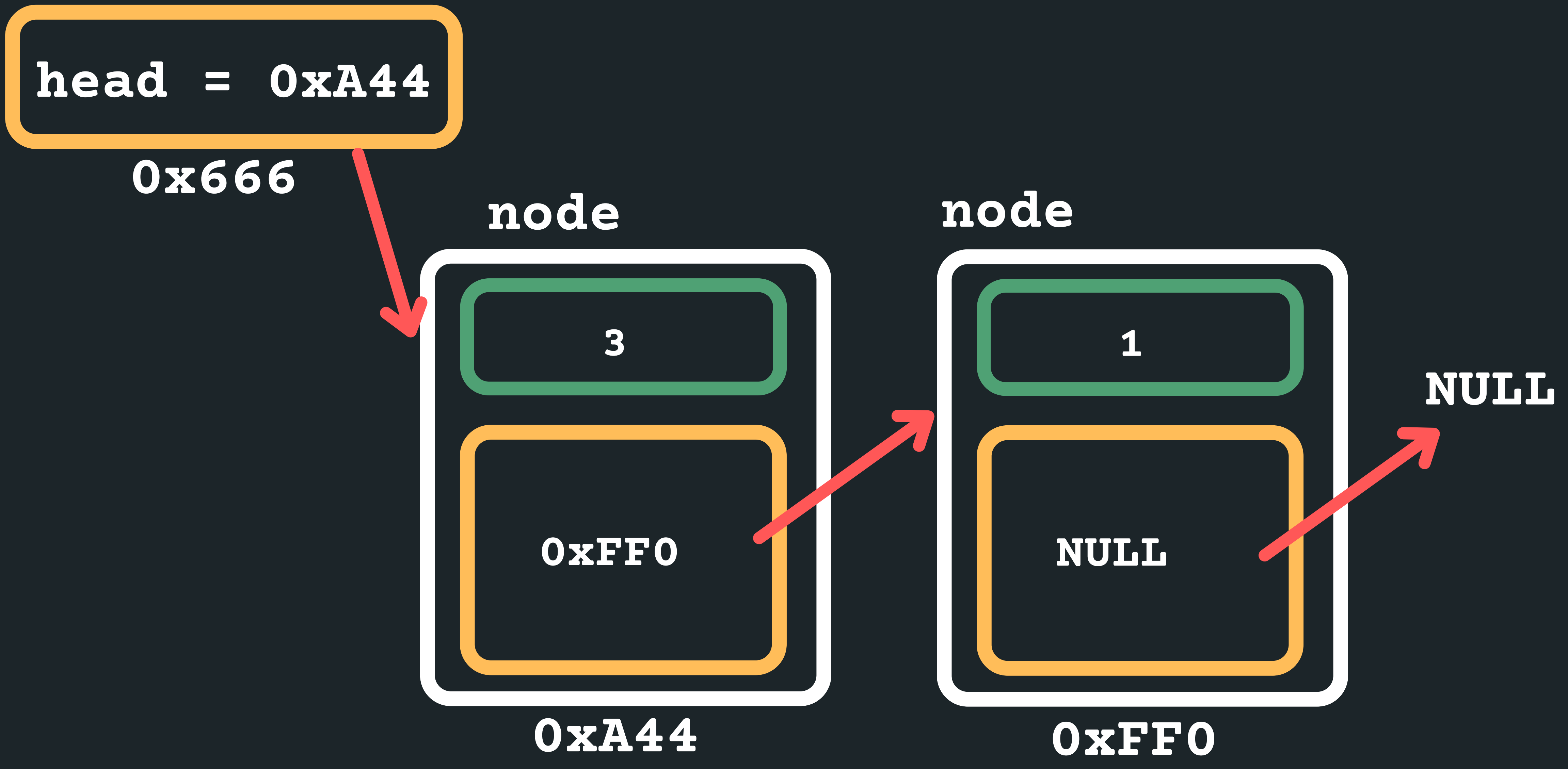
- Let's say we wanted to create a linked list with 5, 3, 1
  - Let's create the first node to start the list!
  - A pointer to keep track of where the start of the list is and by default the first node of the list
  - It will point to NULL as there are no other nodes in this list.



A LINKED LIST

HOW DO WE CREATE ONE AND INSERT INTO IT?

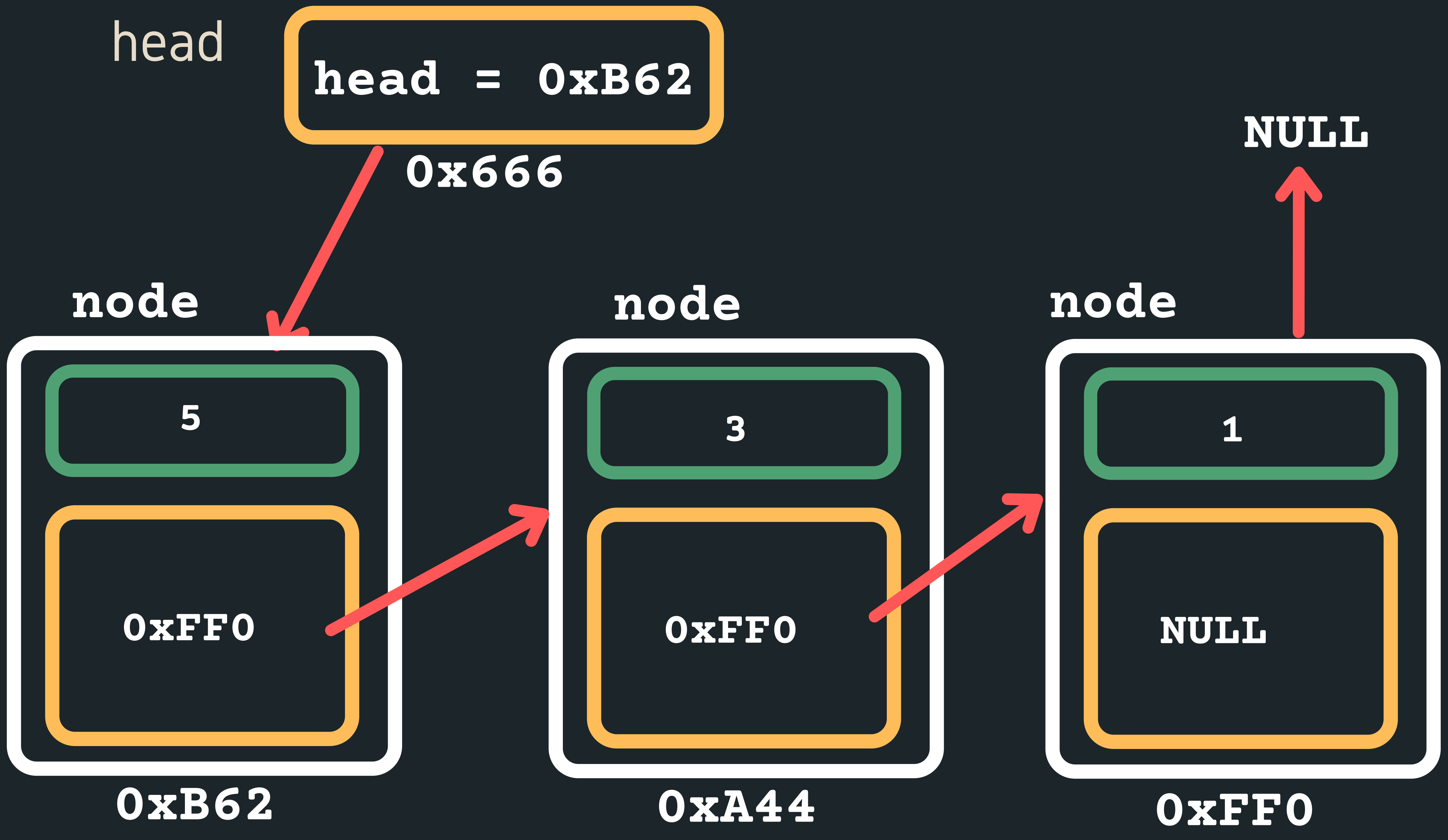
- Create the next node to store 3 into (you need memory)
- Assign 3 to data
- and insert it at the beginning so the head would now point to it and the new node would point to the old head



A LINKED LIST

HOW DO WE CREATE ONE AND INSERT INTO IT?

- Create the next node to store 5 into (you need memory)
- Assign 5 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



A LINKED LIST

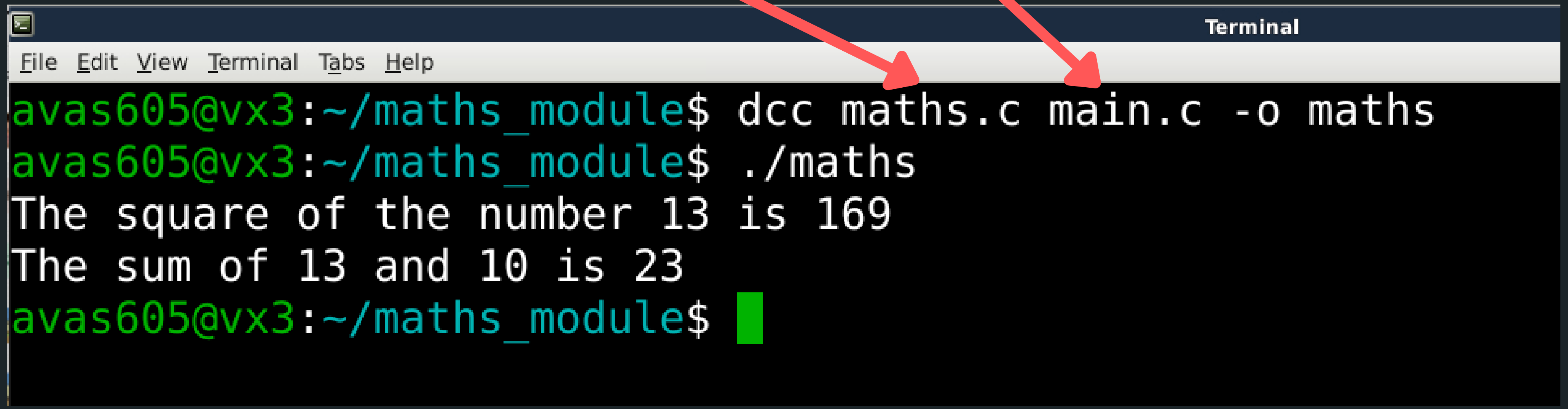
PUTTING IT ALL TOGETHER IN CODE

1. Define our struct for a node
2. 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')
3. 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
4. Make sure last node is pointing to NULL

COMPILING A MULTI FILE

COMPILE ALL C FILES IN THE PROJECT

- To compile a multi file, you basically list any .c files you have in your project
    - In the case of our example, we have a maths.c and a main.c file):



```
avas605@vx3:~/maths_module$ dcc maths.c main.c -o maths
avas605@vx3:~/maths_module$ ./maths
The square of the number 13 is 169
The sum of 13 and 10 is 23
avas605@vx3:~/maths_module$
```

- The program will always enter in main.c, so there should only be one main.c when compiling

Feedback please!

I value your feedback and use it to pace the lectures and improve your overall learning experience. If you have any feedback from today's lecture, please follow the link below. Please remember to keep your feedback constructive, so I can action it and improve the learning experience.

https://forms.microsoft.com/r/dKssTn3AU4

WHAT DID WE LEARN TODAY?

LINKED LISTS

linked_list.c

main.c

linked_list.h

MULTI-FILE PROJECTS

maths.c

main.c

maths.h

REACH OUT

CONTENT RELATED QUESTIONS

Check out the forum

ADMIN QUESTIONS

cs1511@cse.unsw.edu.au