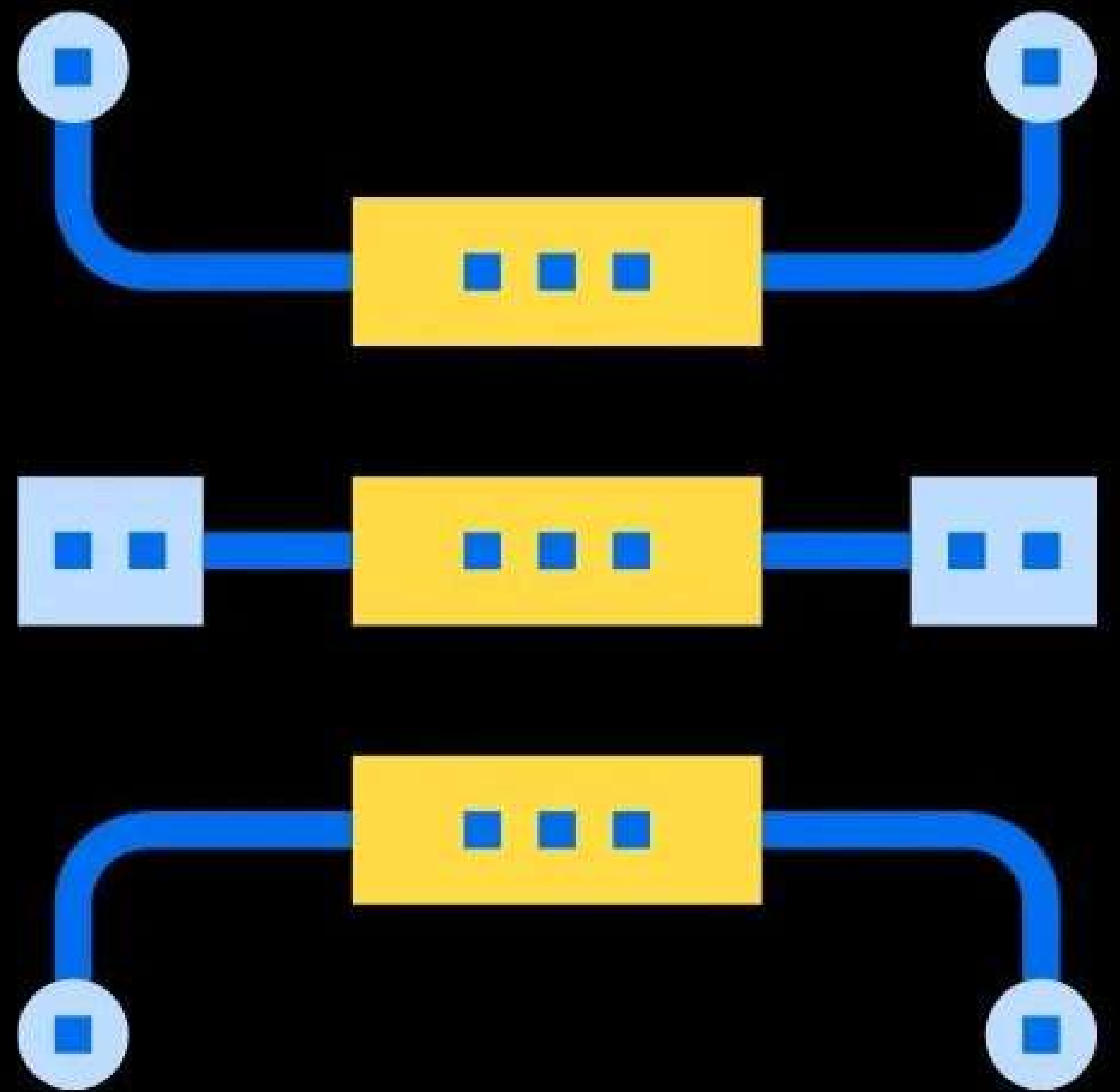




# Why Indexes Start From 0





As you probably already know, **array indexes** start at 0 in almost all major programming languages.



```
int n[] = {25, 50, 75, 100};  
printf("%d", n[0]); // Output: 25  
printf("%d", n[1]); // Output: 50
```



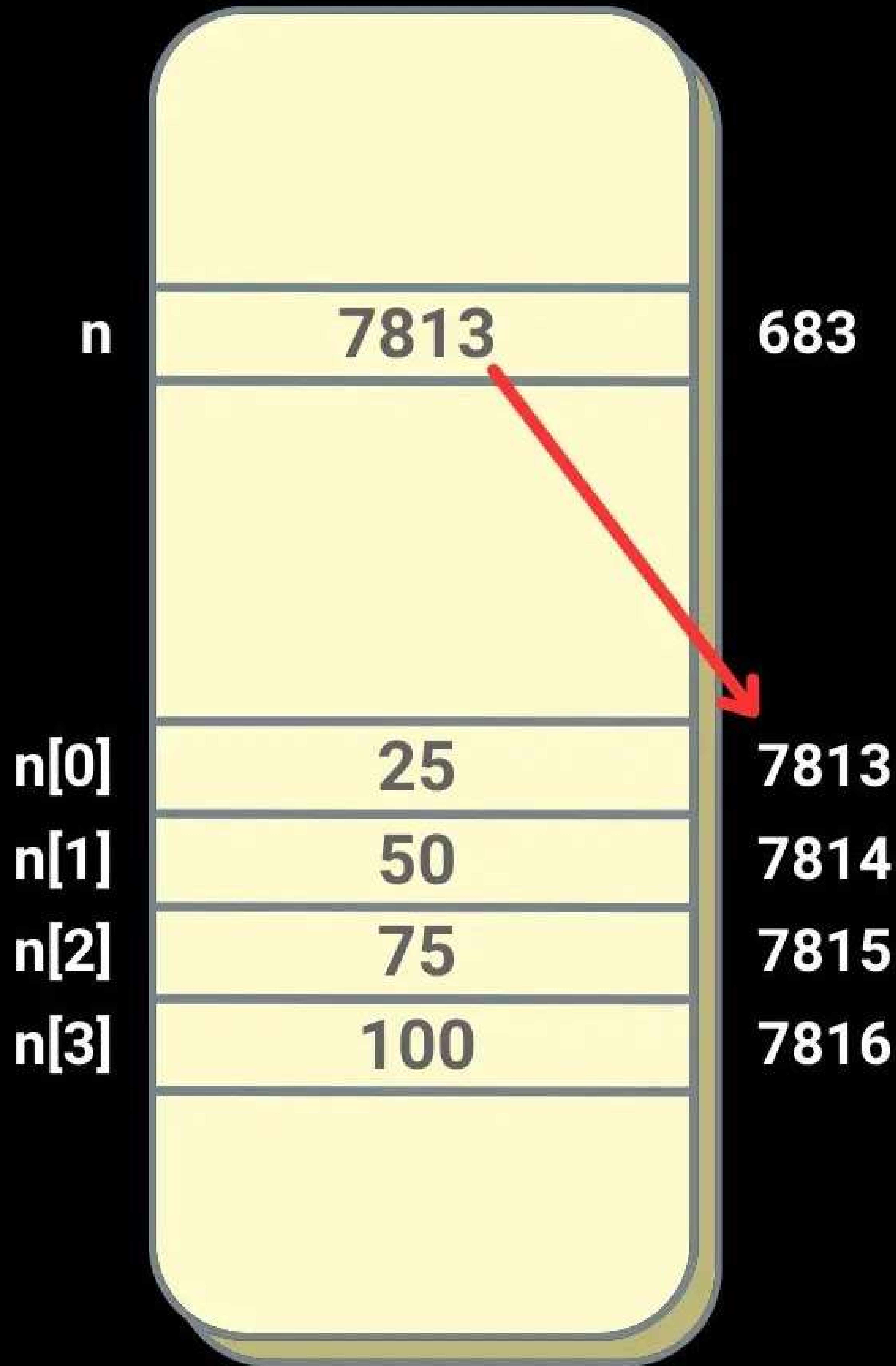
**But do you actually  
know **the reason** why  
it works like that?**



**When an array is used  
as a value, it evaluates  
to a **pointer** to the first  
element of the array...**



# RAM





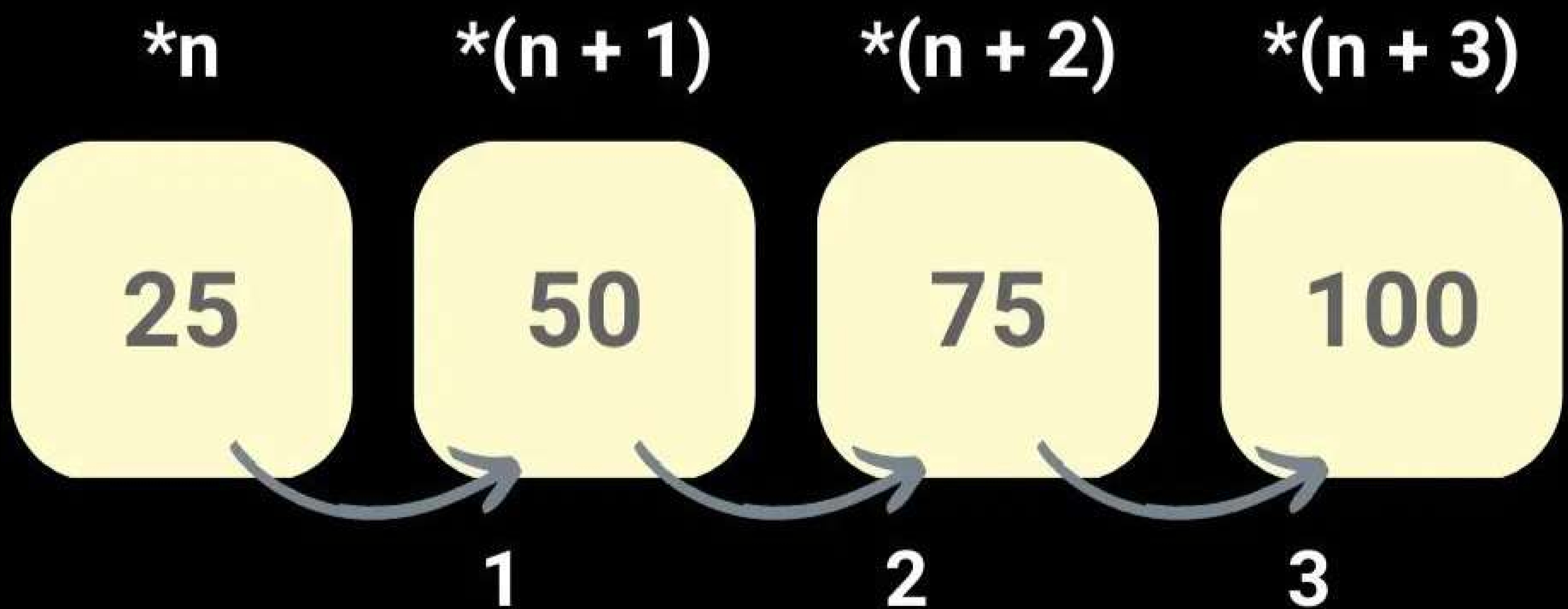
That means the **1st item** is accessed by retrieving the pointer's value.

The **2nd item** is accessed by using the pointer, plus one more memory slot.

The **3rd item** is accessed by using the pointer, plus two more memory slots.



This additional memory slot used to access the items is called **offset**.





In general, we can say that the elements are located at  $*(n + \text{offset})$ .

If the index started at 1, the compiler'd have to use  $*(n + \text{index} - 1)$  to access the elements.





**But if the index starts at 0, it matches the offset.**

**The compiler can then access the elements by using *\*(n + index)* and avoid the additional -1.**