Array

2	23	A	DD	7A	7	31	BB	...	99
0	1	2	3	4	5	6	7		N

Array is by far **the most widely used data structure** in computer science. Why? Because its structure is easy to understand and it comes built-in with most of the programming languages.

What it is? A data structure of multiple elements each defined with an index. Consecutive indexes are placed in a consecutive way in memory. Thus by knowing an index of an element you can access it directly.

Why there are other data structures, than arrays? Yeah, arrays are great, who need something else? Well, arrays are great indeed, but there are cases where they are not optimal. Arrays require certain amount of memory and linked lists could be much more effective for instance.

TIPS

DON'T RELY ONLY ON ARRAYS. Yep, THEY ARE EASY TO USE, BUT WITH THE APPLICATION GROWING WILL GROW ITS MEMORY FINGERPRINT
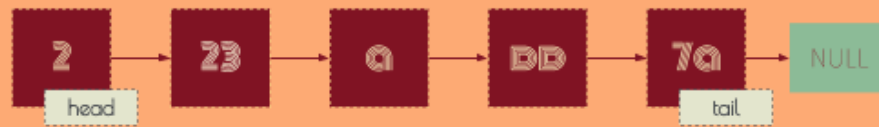
PROS & CONS

PROS: direct access to memory

CONS: Require fixed amount of memory and it's by far the most memory-optimal solution

linked list

In single linked lists (above) the elements are not next to each other in the memory and each item holds a pointer to the next, thus direct access to the elements is not possible. The last (tail) points to NULL to indicate the end of the list. Accessing an element is achieved by walking through the pointers of the elements preceding it.
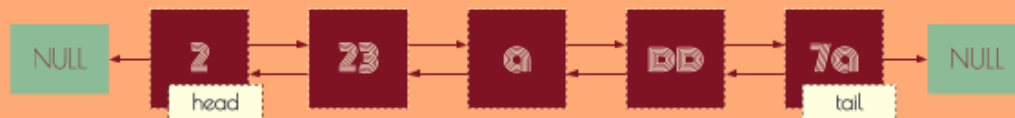
Except single linked lists, there are other implementations as well. The double-linked list (above) is facilitating walk through the elements in both forward and backwards. Linked lists could be very handy for entire class of software problems and are a very basic private case of another widely used data structure: trees!

tips

IF YOU'RE OFTEN INSERTING, DELETING OR REVERSING THE ORDER OF A LIST BETTER USE LINKED LISTS THAN ARRAYS.
WHEN SEARCHING IN A NON-SORTED LIST IS OFTEN PERFORMED - YOUR DATA STRUCTURE OF CHOICE SHOULD BE AN ARRAY!
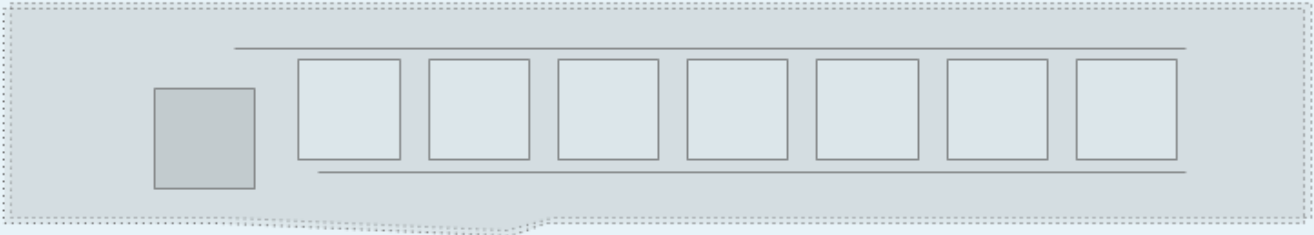
pros & cons

PROS: More memory efficient than arrays and very helpful in certain type of problems i.e. reversing a list or merging two lists.

BAD: Lists lack the big advantage of arrays - the direct access to the elements by a given index, thus searching in a list could be quite time consuming.

stack & queue

The stack data structure models the real-world stack being an abstract implementation and not physical. You can think of it as stack of boxes one above the other. Thus the only way to put another item into the stack is to put it above all other items (on its top). This operation is often called "push". In the other hand taking an item from the stack is called pop, and also only the highest item can be "popped".
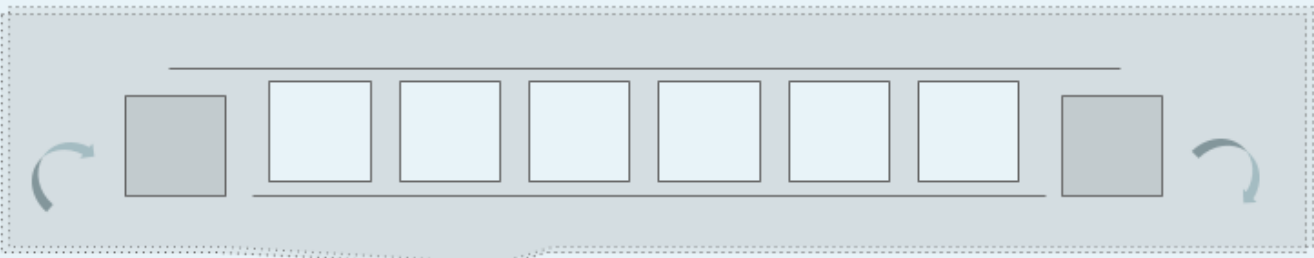
As mentioned above the queue is somehow related to the stack data structure. However it follows a different principle – FIFO (First In First Out), which means that the item that has been in the queue for the longest time is retrieved first.

tips

Queues and stacks are abstract terms, not physical. Thus implementation could be either a linked list or an array. Normally it's implemented as a linked list, because of its memory efficiency. However there is no restriction to implement it as an array where there are already most of the functions like pop or push.

pros & cons

Can't just talk on pros and cons. Specific software cases need a usage of stacks and queues, so … go for it!