

Data Structures

What is a Data Structure?!

Mostafa S. Ibrahim

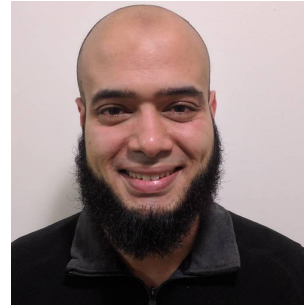
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher

PhD from Simon Fraser University - Canada

Bachelor / Msc from Cairo University - Egypt

Ex-(Software Engineer / ICPC World Finalist)



Recall: 2D Arrays in memory

- What happens when u create such 2D array?
- In **row-major order** style, the **physical memory** will be first data row, then 2nd and so on.
- In **col-major order**, data is ordered column by column
- Observe: the array eventually just *consecutive numbers* in memory

```
int num[3][4] = {  
    {1, 2, 3, 4},  
    {5, 6, 7, 8},  
    {9, 10, 11, 12}  
};
```

	<— row 0 —>				<— row 1 —>				<— row 2 —>			
value	1	2	3	4	5	6	7	8	9	10	11	12
address	1000	1002	1004	1006	1008	1010	1012	1014	1016	1018	1020	1022

Array: A **Physical** Data Structure

- In C++, **array** is a built-in **physical data structure**:
 - **Physical**: data **stored** directly in the **memory**
 - Data structure: **organized** data + **operations** over them
 - What are the allowed arrays operations?
 - 2 simple operations: **Set** and **get** the data using [] operator

```
int numbers[] {1, 6, 10, 5};  
numbers[0] = -3;  
cout<<numbers[2]<<"\n";  
  
float arr[7][10] {};
```

2D array: logical and physical view

- What is 2D array? Just a **table** where we access using `[][]`
 - This is called a **logical** view
 - **No care** how is that exactly in low-level memory
- Physical: How is it organized in memory
 - Eventually, all $N \times M$ data are consecutive in memory
 - But data can be either **row-major** order or **column-major** order
 - Different *math equations* to locate `arr[i][j]`
- 1 logical view, but 2 physical views

Array is limited

- What if I want to **remove** an element from the array?
- What if I want to **add** an element to the array?
- What if I want to support several complex operations?
 - Different sorting styles
 - Searching for elements
 - Comparing 2 arrays
 - Complex operations: intersecting or union of 2 arrays
- We will have to create a **dynamic** array and create several functions
 - Or follow OOP and create a class!

OurArray: A **user defined** data-type

- To solve this problem, we can create a **class: OurArray (aka vector)**
 - Inside it, we have `int*` array to represent our data
 - With some extra data members: E.g. `int size`
 - Then several member functions to operate over this data
- We call OurArray: A **logical data structure**
 - **Data:** `int* p` and `int size`
 - **Operations:** Search, Insert, Remove, Compare
 - **Logical:** it doesn't **directly** correspond to memory
 - But eventually based on primitive data type (`int`, `int[]`, etc) that physically exists
- Overall: to create a data structure
 - Create your own class + define data + operations over them

Built-in Data Structures (DS)

- During our software engineering history, we found some kind of **data+operations** are common and repetitive
 - Some are basic such as what we call now: Vector, Stack and Queue data structures
 - E.g. A queue data structure is useful for queues in *restaurants and hospitals*
 - Some are advanced such as Hash-Table and AVL Trees
 - These are all **logical** data structures!
- It is not wise to keep people re-implementing them
 - Many languages implement them: STL in C++, Collections in Java/C#, etc
- During this course, we will learn the **inner details** of these built-in DS

Normal class vs Data structure

- It seems a data structure is a class with data and member functions!!
- Why do we give it a special name? Because of how we perceive it
- Data structure is very **centered around data** to provide specific functionality
 - It is mainly about the data. Specific functionalities are **driven** to serve the data purpose
 - Every data structure **arranges & stores** data in a specific way to support a specific use case
 - Queue data structure orders that data to follow: **First in First Out order**, like restaurants
- A normal class is **centered around functionalities**. I want a Student class that I can add, mark, print the students' assignments + holds student info
 - Employee, Payroll, Question, Answer, Email, etc. All are classes of **business logic**
- Eventually, both has data + operations, but different view
- After studying basic data structures, you will easily understand the above notes

“Acquire knowledge and impart it to the people.”

“Seek knowledge from the Cradle to the Grave.”