\* LMCR2303Competitive Programming

### Data structure Part 1

(lecturer: Nazatul Aini Abd Majid)



### \*Learning outcome

- \*Able to link knowledge of data structures and algorithms in programming.
- \* Able to select the appropriate problem-solving method for computer science problems based on several categories of problems.
- \* Able to write effective programming code in solving computer science problems competitively.



### \*Introduction

→ Program →



Add two numbers: a and b

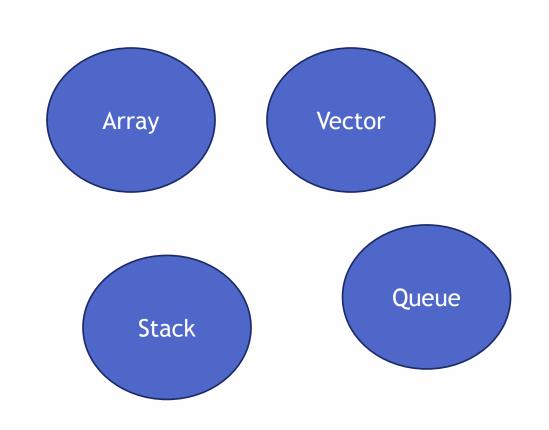


A collection of sequential data to be stored and later accessed using their indices.



**Linear Data Structures** 

## \*Linear Data Structure





- \*Static Array
- \*the array size can be declared to be the maximum input size
- \*int arr[3] = {7,7,7}
- \*int[] arr = new int[] {7,7,7,0,0};



- \*Dynamically-Resizeable Array:
- \*This data structure is similar to the static array, except that it is designed to handle runtime resizing natively.
- \*vector<int> v(5, 5); // initial size (5) and //initial value {5,5,5,5,5}
- \*Vector<Integer> v = new Vector<Integer>(Collections.nCopies(5, 5));

## \*Array and Vector

#### **Tutorial 1**

### **BREAK 10 Minutes**



- \*This data structure is often used as part of algorithms that solve certain problems.
- \*Last In First Out (LIFO) (literal stacks in the real world)
- \*stack<char> s;
- \*Typical C++ STL stack operations
- \*include push()/pop() (insert/remove from top of stack), top() (obtain content from
- \*the top of stack), and empty().



- \*This data structure is used in algorithms like Breadth First Search (BFS).
- \*First In First Out (FIFO), just like actual queues in the real world.
- \*queue<char> q;
- \*Typical C++ STL queue operations include push()/pop() (insert from back/remove from front of queue), front()/back() (obtain content from the front/back of queue), and empty().

## \*Stack and Queue

#### **Tutorial 1**

### **BREAK 10 Minutes**

### \*Sort and Searching

- \*Two operations commonly performed on Arrays
- \*O(n2) comparison-based sorting algorithms: Bubble/Selection/Insertion Sort, etc. These algorithms are (awfully) slow and usually avoided in programming contests, though understanding them might help you solve certain problems.
- \*O(n log n) comparison-based sorting algorithms: Merge/Heap/Quick Sort, etc.
- \*These algorithms are the default choice in programming contests.

## \*Sort and Searching

\*Break 15-20 minutes and Let's do Lab 3 at Makmal Pengajaran 1

# \*Summary



Dr Nazatul Aini Abd Majid nazatulaini@ukm.edu.my