Unicode is the super set of ASCII code.

https://atifishaqgcu.wordpress.com/category/programming-fundamentals/

iOS and android have same code of emoji but some difference in emoji shape.

rgb =72 73 33 =>3 bytes will be allocated

audio store in MIDI format at backend

getch()

method **pauses the Output Console until a key is pressed**. It does not use any buffer to store the input character. The entered character is immediately returned without waiting for the enter key. The entered character does not show up on the console. Using getch() function, we can hide the input character provided by the users in the ATM PIN, password, etc.

This function is used to get a single character input from the user, during execution of program. It also force to wait the output to stay on screen until any key pressed from keyboard or is used to hold the screen so that you are able to see the output.

# What does pragma once do?

#pragma once is a preprocessor directive used to **prevent header files from being included multiple times**. The #pragma once directive, once present in a file, assures that the file will not be included multiple times in the current project.

* What does \* this mean in C++?  **pointer to the object**,

**Stack:**

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO (Last In First Out) or FILO (First In Last Out).

## **Push:**

Adds an item to the stack. If the stack is full, then it is said to be an Overflow condition.

**Algorithm for push:**

begin

if stack is full

return

endif

else

increment top

stack[top] assign value

end else

end procedure

## **Pop:**

Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.

**Algorithm for pop:**

begin

if stack is empty

return

endif

else

store value of stack[top]

decrement top

return value

end else

end procedure

## **Top:**

Returns the top element of the stack.

**Algorithm for Top:**

begin

return stack[top]

end procedure

## **isEmpty:**

Returns true if the stack is empty, else false.

**Algorithm for isEmpty**:

begin

if top < 1

return true

else

return false

end procedure

## **Types of Stacks:**

* **Register Stack:** This type of stack is also a memory element present in the memory unit and can **handle a small amount of data** only. The height of the register stack is **always limited as the size** of the register **stack is very small** compared to the memory.
* **Memory Stack:** This type of stack can handle a large amount of memory data. **The height of the memory stack is flexible** as it occupies a large amount of memory data.

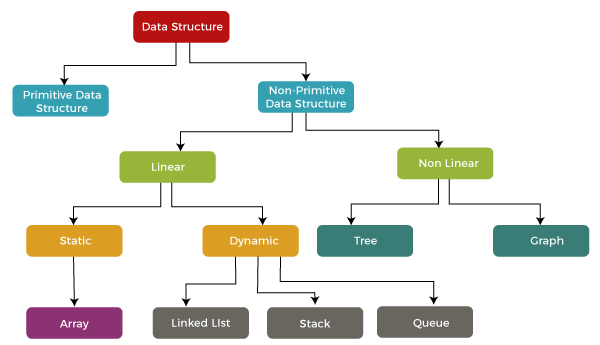
There are two ways to implement a stack

* Using array
* Using linked list

**Time Complexity**

| **Operations** | **Complexity** |
| --- | --- |
| push() | O(1) |
| pop() | O(1) |
| isEmpty() | O(1) |
| size() | O(1) |
| top() | Return the top element |
|  |  |

<https://www.geeksforgeeks.org/introduction-to-stack-data-structure-and-algorithm-tutorials/>



Primitive datatypes are stored in stack and non-primitive are stored in heap.

* **peek():** It returns the element at the given position.
* **count():** It returns the total number of elements available in a stack.
* **change():** It changes the element at the given position.
* **display():** It prints all the elements available in the stack.

When we create an object where it is stored?

The object will be stored **in the stack** when it is created inside the block and when the control exits the block or function then the object is removed or destroyed. In case of dynamically allocated objects (during runtime) the object will be stored on the heap. This is done with the help of new operator

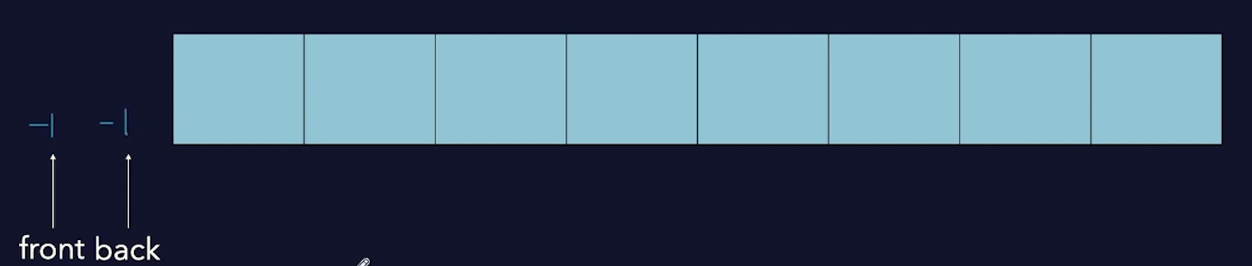
Queue:

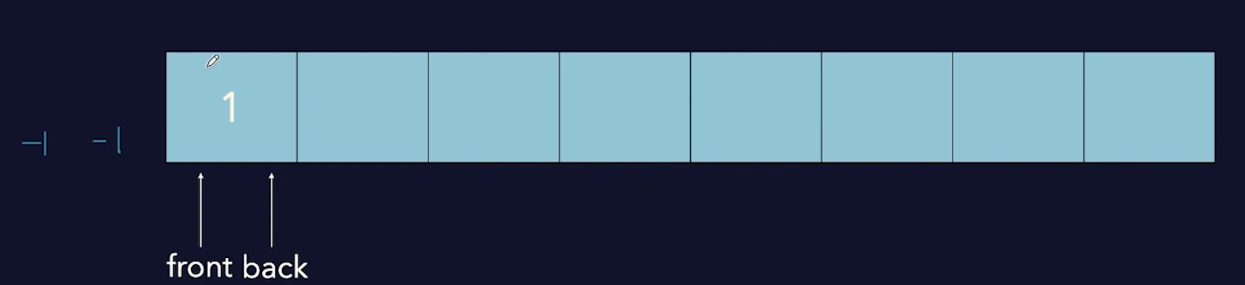
Stores a list of items in which an item can be inserted at one end and removed from the other end only.

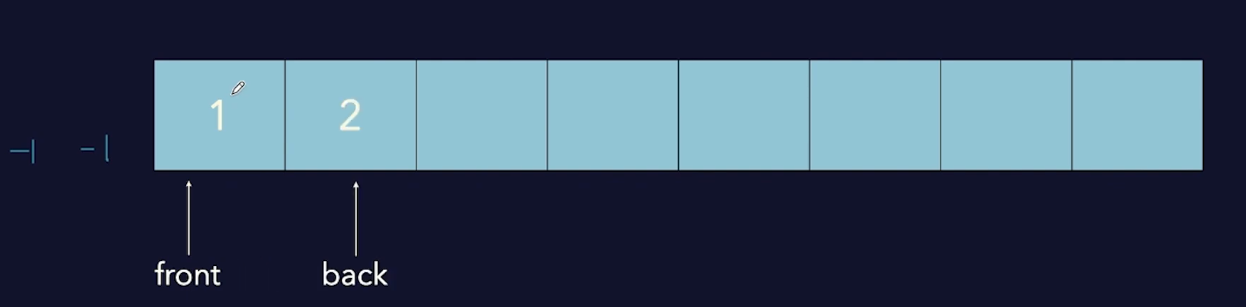
FIFO(First in First out)

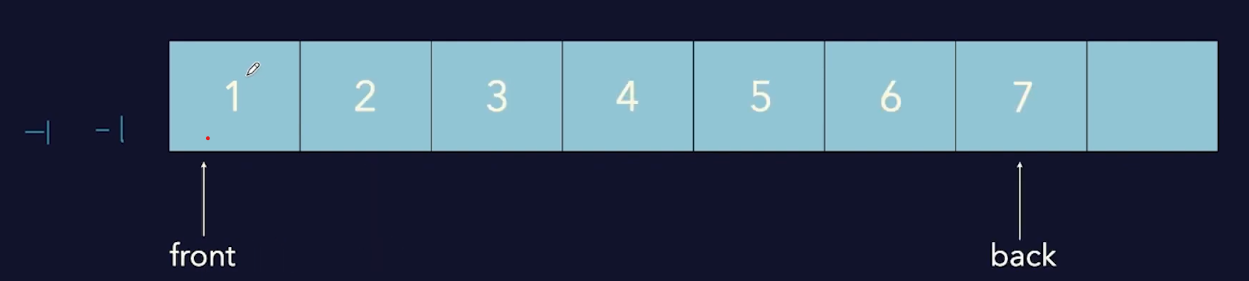
# **Enqueue();**

Added value from **back**





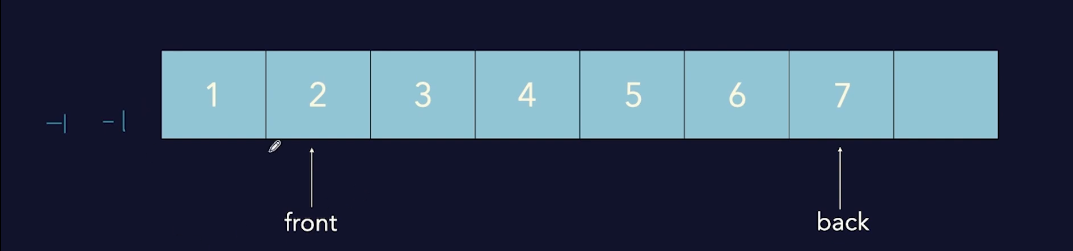




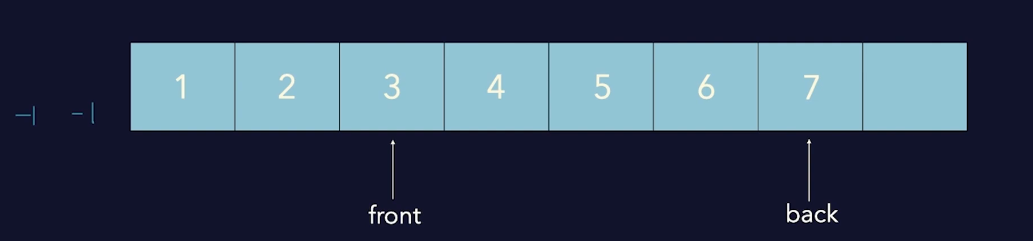
# **Dequeue();**

Remove value from **front**

**front++**



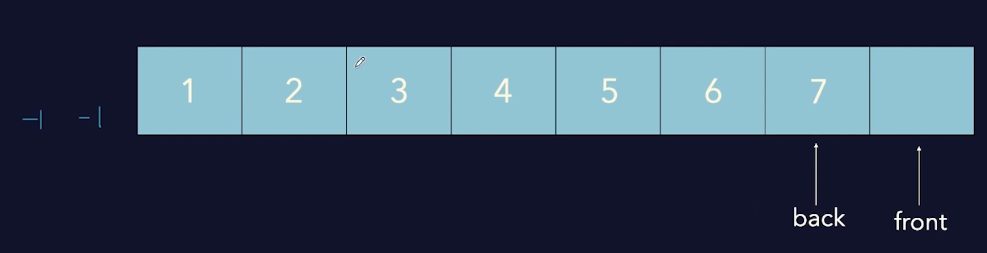
**front++**

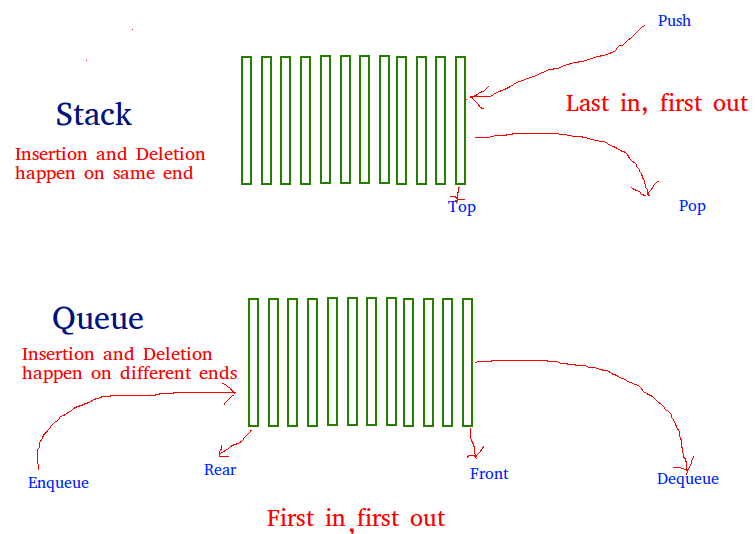


Peek();



**Empty();**

****



Linkedlist:

1. **Singly linkedlist**
2. **Doubly linkedlist**
3. **Circular linkedlist**
4. **Doubly Circular linkedlist**

The size of pointer variable of a node in 32-bit compiler is 4-byte and int 64-bit compiler is 6-byte.

When **insertion** repeated many times then we prefer **linked list** because insert **has time complexity O(1**) in linkedlist .

But when we needed **search** again and again then we **use Array list** and has **time complexity O(1)** in search and in insert it will be O(n) time complexity.

https://www.geeksforgeeks.org/data-structures/linked-list/?ref=shm

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers as shown in the below image:  


## **Advantages of Linked Lists over arrays:**

* Dynamic Array.
* Ease of Insertion/Deletion.
* **Simple/singly Linked List** – In this type of linked list, one can move or traverse the linked list in only one direction
* **Doubly Linked List** – In this type of linked list, one can move or traverse the linked list in both directions (Forward and Backward)
* **Circular Linked List** – In this type of linked list, the last node of the linked list contains the link of the first/head node of the linked list in its next pointer and the first/head node contains the link of the last node of the linked list in its prev pointer

## **Representation of Linked Lists:**

A linked list is represented by a pointer to the first node of the linked list. The first node is called the head of the linked list. If the linked list is empty, then the value of the head points to NULL.

Each node in a list consists of at least two parts:

* A Data Item (we can store integer, strings, or any type of data).
* Pointer (Or Reference) to the next node (connects one node to another) or An address of another node

## **Common singly linked​ list operations**

### 1. **Search for a node in the list**

You can determine and retrieve a specific node from the front, end, or anywhere in the list.

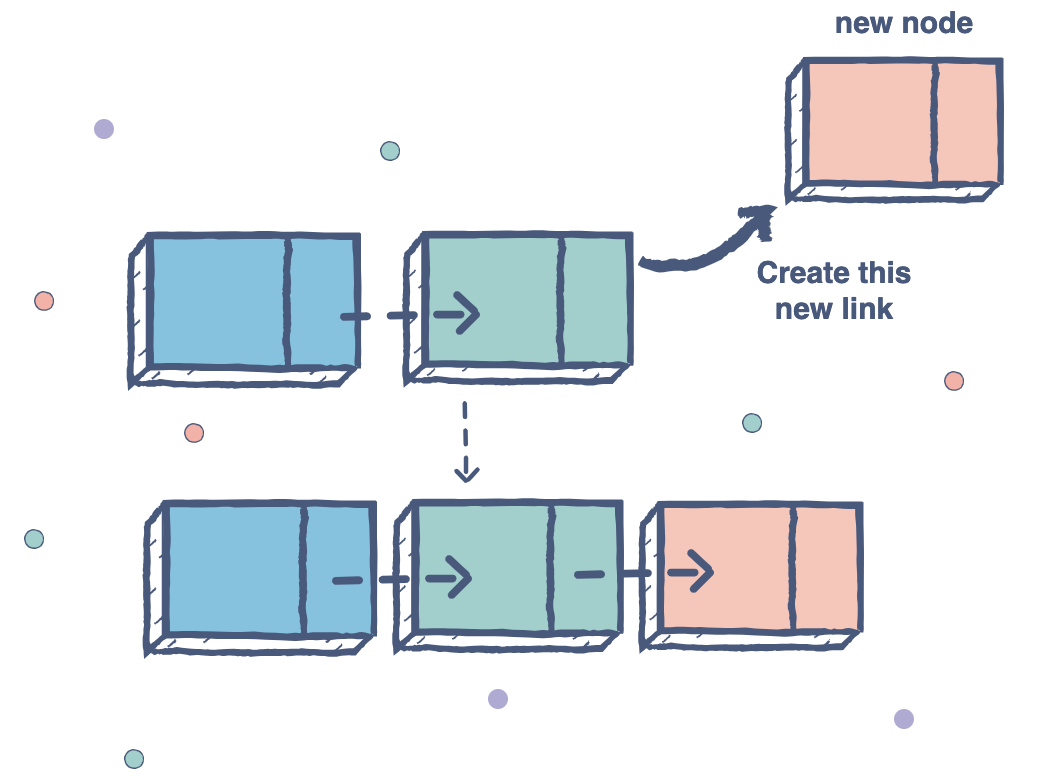
The worst-case​ Time Complexity for retrieving a node from anywhere in the list is O(n).

### 2**. Add a node to the list**

You can add a node at the front, end, or anywhere in the linked list.

The worst-case Time Complexities for performing these operations are:

* Add an item to the front of the list: O(1)
* Add an item to the end of the list: O(n)
* Add an item a​nywhere in the list: O(n)



### 3**. Remove a node from the list**

You can remove a node from the front, end, or anywhere in the list.

The worst-case Time Complexities for performing this operation are:

* Remove an item from the front of the list: O(1)
* Remove an item from the end of the list: O(n)
* Remove an item from anywhere in the list: O(n)