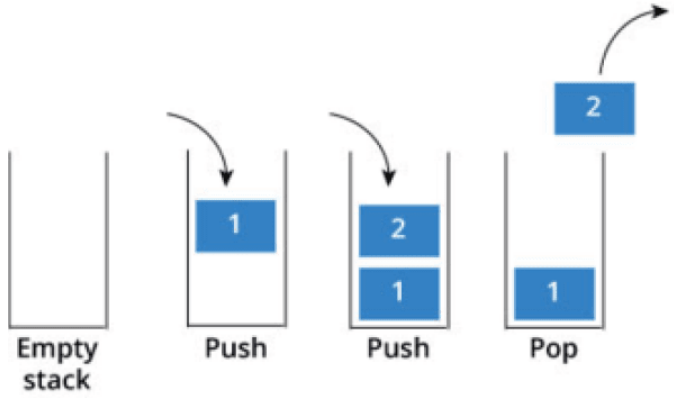
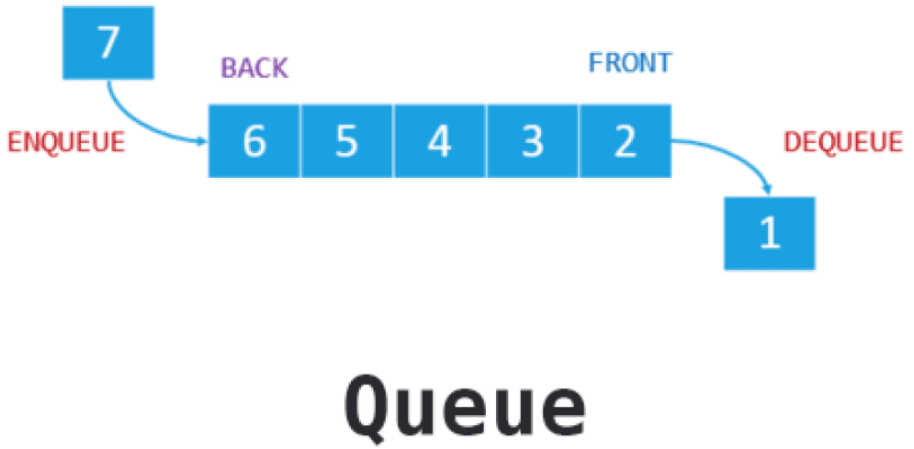
DATA STRUCTURE & PROGRAMMING II

Stack data structure

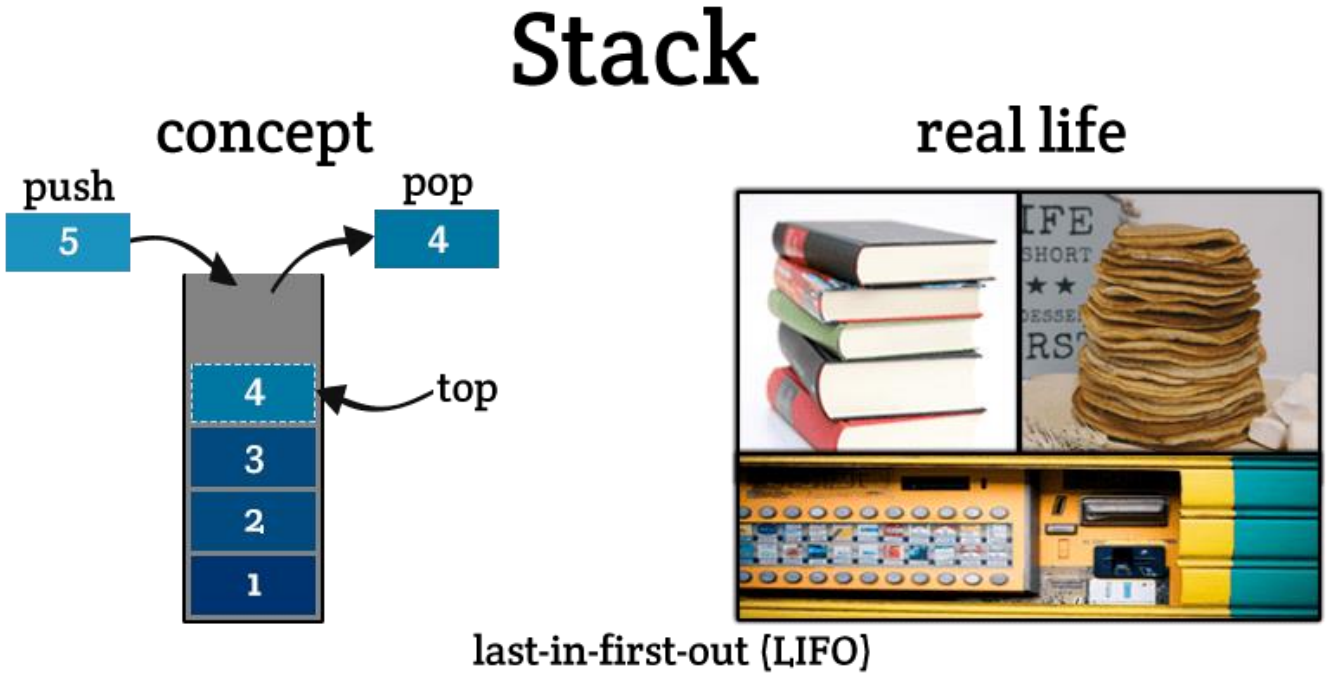




Outline

❑ A Brief of Outline

- What is Stack?
- What are Stack operations?
- How to implement Stack in C++
- Examples



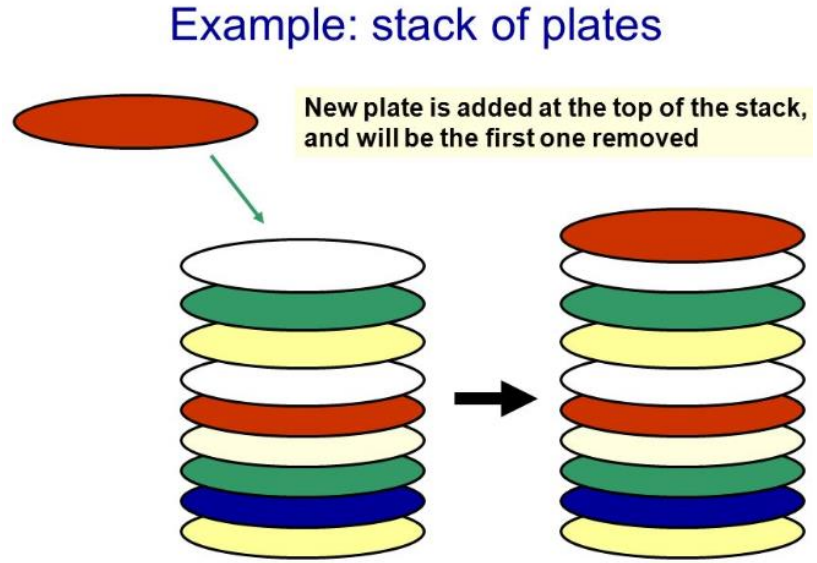
What is Stack?

❑ Definition

- **Stack** is a data structure that stores data in such a way that the element stored last will be retrieved first
- This method is also called **LIFO (Last In First Out)**

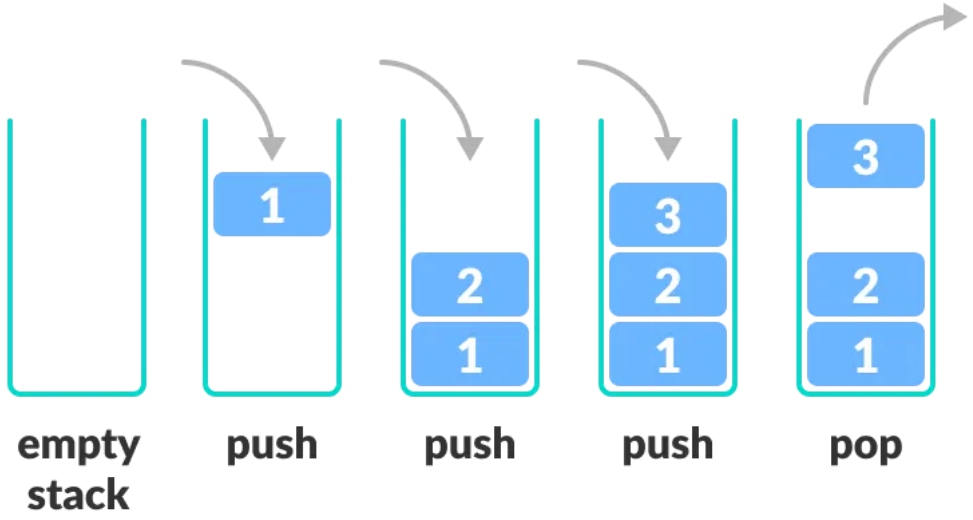
Examples:

- A stack of copies
  - The first copy put in the stack is the last one to be removed
  - Similarly, the last copy put in stack is the first one to be removed
- Stack of plates
- Stack of chairs



Definition

- A **stack** is a linear data structure that follows the **LIFO** (Last In, First Out) principle.
  - LIFO: The last element added to the stack is the first one to be removed, displayed, searched. We do operations with the top element first.
  - You can only ***insert (push)*** or ***remove (pop)*** elements from the top of the stack.



Why using stack data structure?

Stacks are useful for:

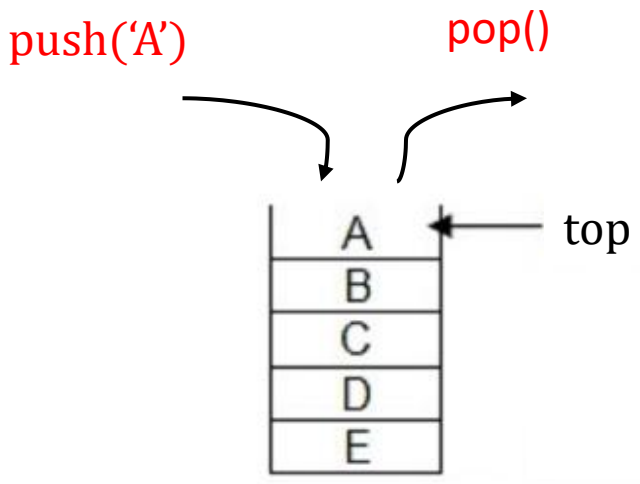
- **Function call** management (call stack)
- **Undo operations** in editors
- **Expression parsing** (e.g., infix to postfix)
- **Backtracking algorithms** (e.g., maze solving, DFS)
- **Managing state/history** in games or applications

Applications of Stack

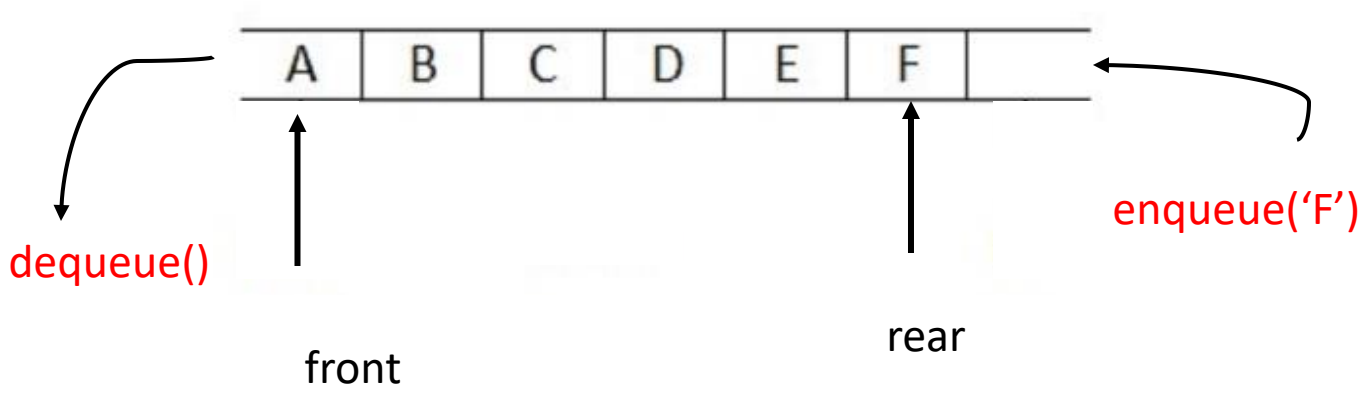
- Undo operations (browser, Ms. Word, …)
- Remembering completed tasks
- Design compilers and interpreters
- etc.

Queue Vs. Stack

- Differences



Stack

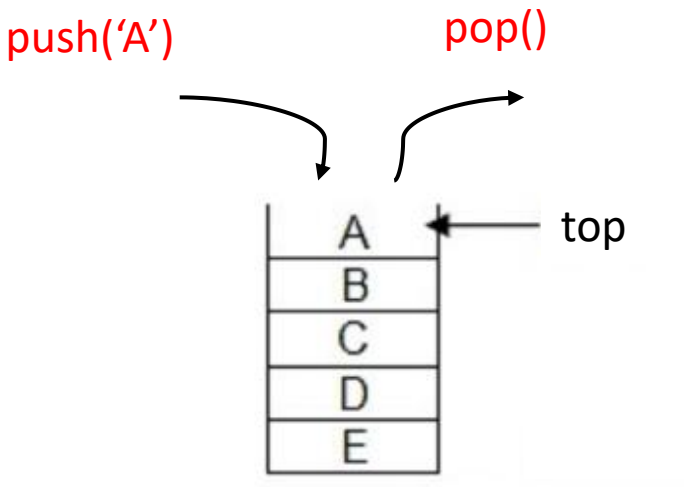


Queue

Stack Operations

❑ Operation

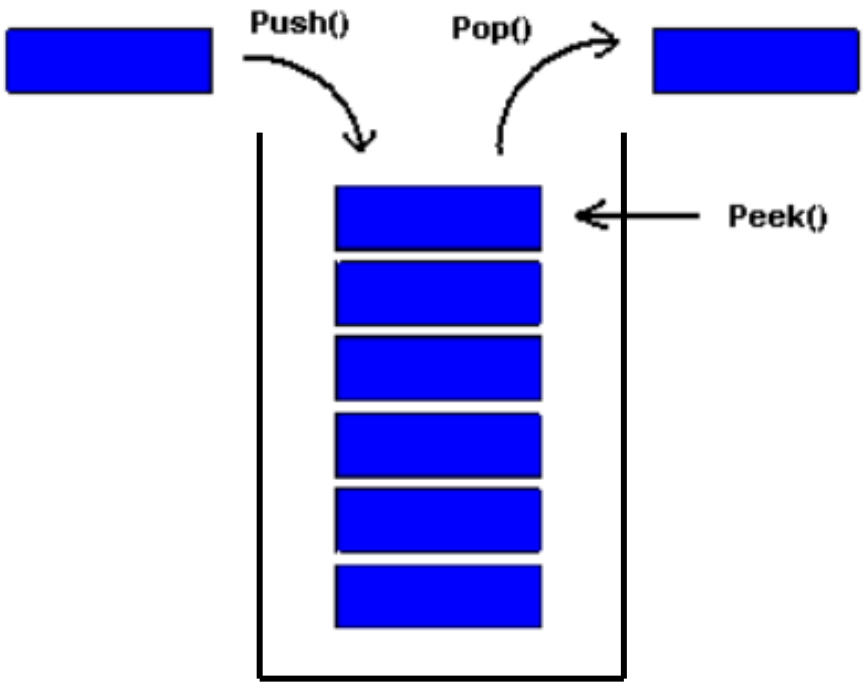
- A **stack** is controlled by two main operations which implement the **LIFO method**
  - ***Insertion***
    - Add element to the stack (add to the top)
    - This method is called ***push***
  - ***Deletion***
    - Remove element from the stack (remove from the top)
    - This method is called ***pop***



- The variable TOP is used to keep track of the top element in the stack

Stack Operations

❑ More operations



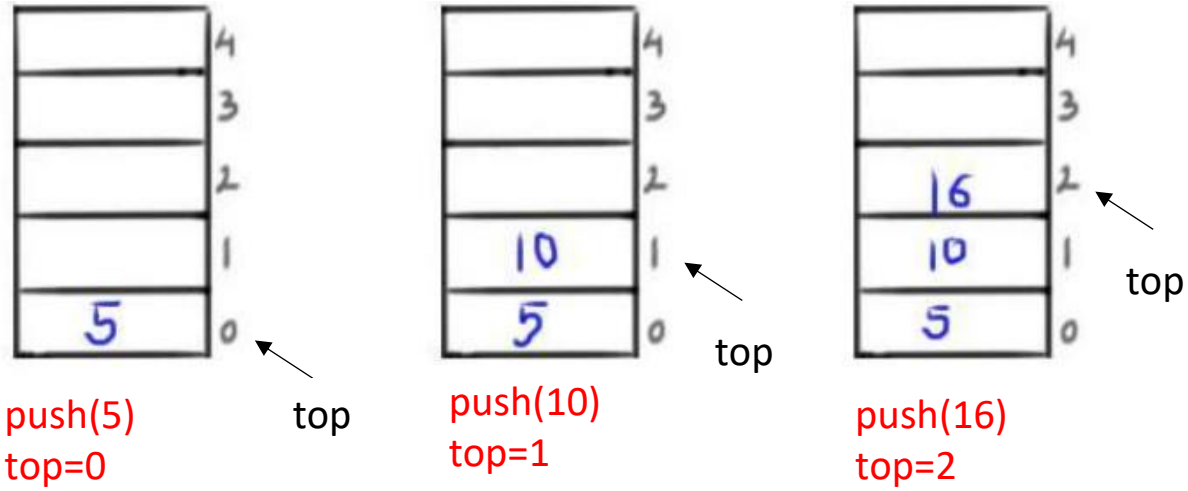
- **`push()`:** Add element to top of stack
- **`pop()`:** Remove element from top of stack
- **`isEmpty()`:** Check if stack is empty
- **`isFull()`:** Check if stack is full
- **`peek()`:** Get the value at the top of stack without removing it

Stack Implementation

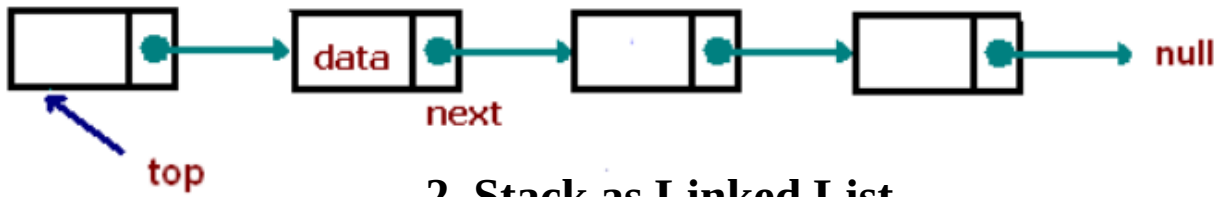
❑ Implementation

➢ Stack can be implemented in two way

1. As an Array
   - An array to store data
   - An integer type variable called TOP which stores the index of the top element of the stack
2. As a Linked List
   - A linked list to store data
   - A pointer variable called TOP which points to the top element of the list



1. Stack as Array



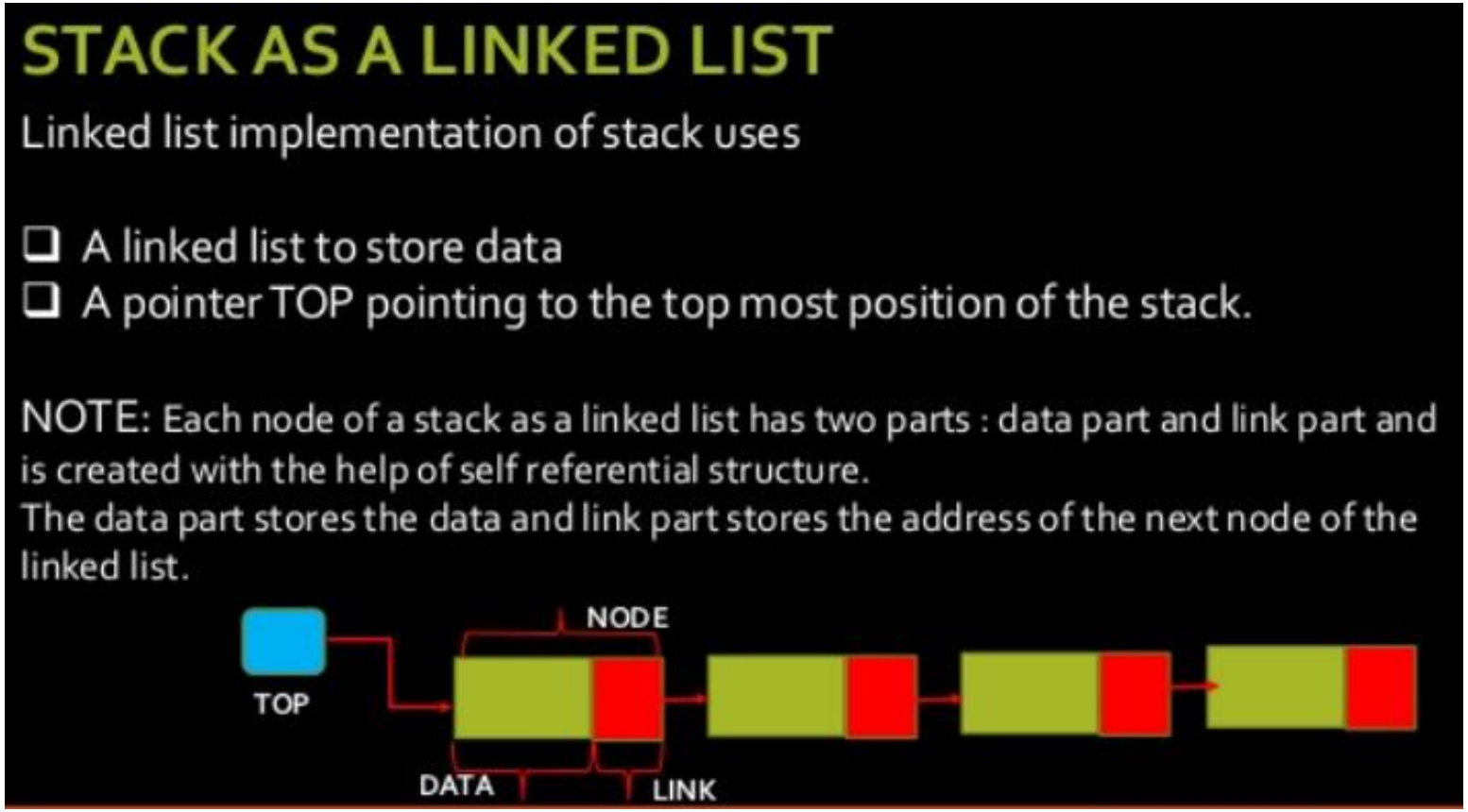
2. Stack as Linked List

Stack as Linked List

- Dynamic
  - It can grow or shrink at runtime

Stack Implementation

- Stack as a Linked List



Stack Implementation

❑ Stack as a Linked List

- Implementing stack as a linked list is just like implementing a linked list with some choices

  Choice 1

  - Element is added to the first of the list **(push operation)**
  - Element can be only removed from the beginning of the list **(pop operation)**

  Choice 2

  - Element is added to the end of the list **(push operation)**
  - Element can be only removed from end of the list **(pop operation)**

Remark: Choice 1 is recommended.

Stack as Array Vs. Stack as Linked list

❑ When to use?

➢ When we use array to implement stack?

- Must know exact size of the stack and it is small

➢ When we use Linked list to implement stack?

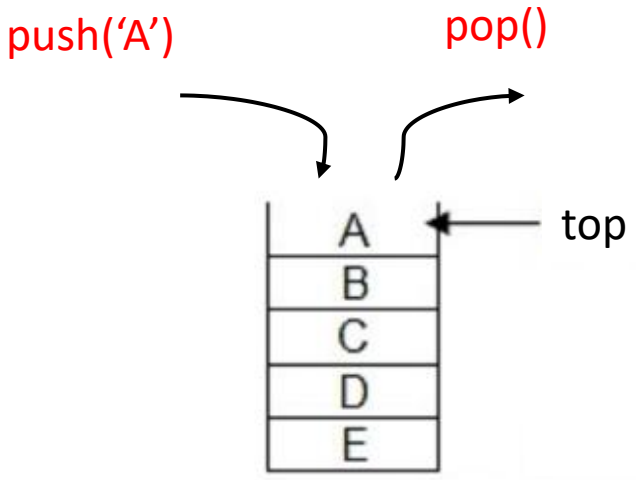
- Do not know the size of the stack.
- If the stack can be bigger and bigger, Linked list will be good to implement it.

Stack Implementation: Examples

❑ Stack as a Linked List

How to implement this Stack?

Demo coding in class



Q and A

Using built-in library in C++

stackbuiltin.cpp

```cpp
#include <iostream>
#include <stack>
using namespace std;
main() {
    stack<int> s;

    s.push(10); // Push elements
    s.push(20);
    s.push(30);

    cout << "Top element: " << s.top() << endl; // 30

    s.pop(); // Remove topendl; // 20
    cout << "Stack size: " << s.size() << endl;

    cout << "Is stack empty? ";
    if(s.empty()){
        cout<<" Yes"<<endl;
    }else{
        cout<<" No"<<endl;
    }
}
```

D:\DSP2-24-25\Stack\stackbuiltin.exe

```
Top element: 30
Stack size: 2
Is stack empty?  No

Process returned 0 (0x0)
```