Stack:

A **stack** is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle. All insertion and deletion are permitted only at one end of list. A **stack** is a limited access data structure - elements can be added and removed from the **stack** only at the top.

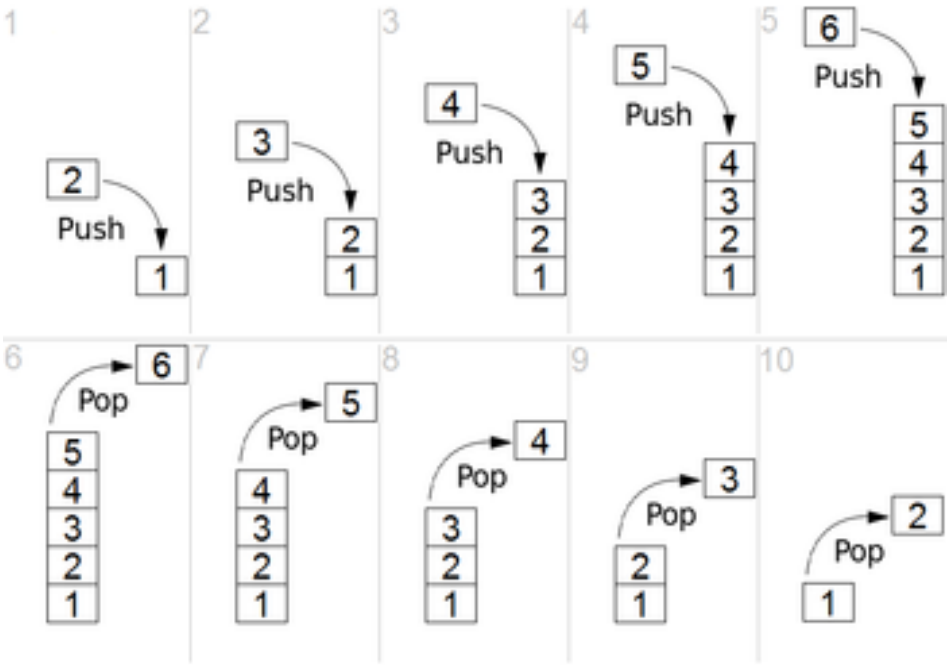
Operations:

Basic operations performed in stack.

- **Push ():**
  Push function is used to add or insert new element in stack.
  Its time complexity is O(1).
- **Pop ():**
  Pop function is used to delete or remove an element from stack.
  Its time complexity is O(1).
- **Peek():**
  Peek function returns the value of the top ("front") of the stack without removing the element from the stack.
  Time complexity of peek function is O(1).
- **IsEmpty():**
  IsEmpty() method is used to check and verify if a Stack is empty or not. It returns True if the Stack is empty else it returns False.
  Parameters: This method does not take any parameter.
  Return Value: This function returns True if the Stack is empty else it returns False.
- **IsFull():**
  IsFull Tests if the stack is full or not.
- **Overflow:**
  Overflow is the state when stack is completely full.
- **Underflow:**
  Underflow is the state when stack is completely empty.

Time Complexity:

Time complexity of stack is O(1).



Real life Example:

A good real-life example of a stack is the pile of dinner plates that you encounter when you eat at the local cafeteria: When you remove a plate from the pile, you take the plate on the top of the pile. But this is exactly the plate that was added ("inserted") most recently to the pile by the dishwasher.

Applications:

Applications of stacks are;

- Expression evaluation(Prefix, Postfix, infix)
- Syntax Parsing
- Parentheses check
- Function call

Implementation:

Link for stack implementation in c++

https://github.com/MahamAwan/11jan2020/blob/master/Stack.cpp