

Name:

Wajiha zahid

Roll no :

SU92-BSSEM–S24-040

Section :

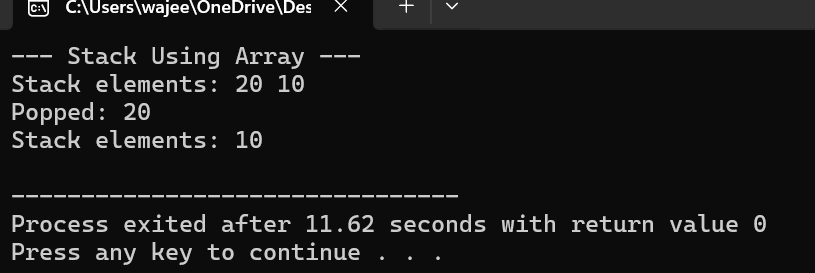
BSSE (3A)

Subject :

DSA(LAB)

(LAB No 10)

STACK USING ARRAY



This code implements a **stack** using an **array** in C++. A **stack** is a linear data structure that follows the **Last In, First Out (LIFO)** principle. That means the last element added (pushed) is the first one to be removed (popped).

Explanation

 Top keeps track of the index of the **top element** in the stack.

 arr[MAX] is the array that holds the elements.

* Initializes the stack by setting top = -1, which means the stack is **empty**

 Checks if the stack is full (top == MAX - 1). If full, it prints **"Stack Overflow"**.

 Otherwise, it increments top and inserts the element x at the new top position.

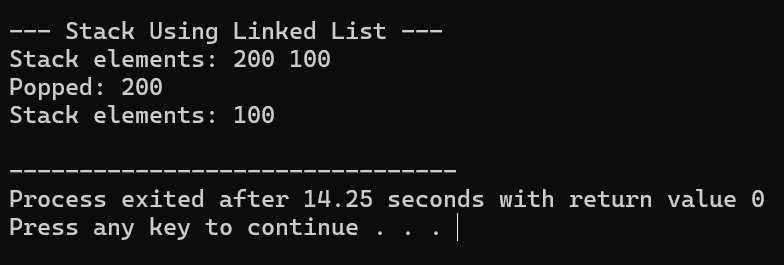
* If top == -1, the stack is empty, so it prints **"Stack Underflow"**.
* Otherwise, it prints the top value and decrements the top index.
* If the stack is empty, it says so.
* Otherwise, it prints the elements from **top to bottom** (since stack is LIFO).
* Creates a stack object sa using the array implementation.  Pushes 10 and 20 onto the stack.

 The stack now looks like this (top → bottom):  
20 (top)  
10

**Summary of What Happens**

1. A stack object is created using an array.
2. Two integers (10 and 20) are pushed onto the stack.
3. The stack is displayed: it shows 20 on top of 10.
4. The top element (20) is popped.
5. The stack is displayed again, showing just 10.

STACK USING LINKED LIST



This code defines a **Stack** data structure using a **singly linked list**. Unlike an array, a linked list allows **dynamic memory allocation**, so you're not limited by a fixed size.

The stack still follows the **LIFO (Last In, First Out)** principle.

Explanation

* This is the building block of your linked list.
* Each Node stores:
  + - An integer data
    - A pointer next to the next node in the stack
* Class: StackLinkedList
* Top is a pointer to the **top node** of the stack.
* This is similar to the top index in array-based stacks, but here it points to a memory location.
* push(int x) – Add an element

 Creates a new node dynamically (new Node()).

 Assigns x to the node’s data.

 Links the new node to the current top.

 Updates top to point to the new node.

 This makes the new node the "top" of the stack.

* **Why?**  
  Because in a stack, the last pushed element should always be the one to be popped first — so we always insert at the **head**.
* PoP ()
* Otherwise, it:
  + Stores the current top in a temp variable.
  + Prints the data being popped.
  + Moves top to the next node.
  + Frees the memory of the popped node.
* 📌 **Why?**  
  To maintain memory efficiency and prevent memory leaks, we use delete.

main() Function

1. Stack is initialized (top = nullptr)
2. sl.push(30)
   * A node with value 30 is added.
   * Stack: 30 (top)
3. sl.push(40)
   * A new node with 40 is added on top.
   * Stack: 40 (top) → 30
4. sl.display()
   * Output: Stack contents: 40 30
5. sl.pop()
   * 40 is removed.
   * Stack becomes: 30
6. sl.display()
   * Output: Stack contents: 30