# Data Structures – Stack

Maksim Kalutski

Brno University of Technology, Faculty of Information Technology
Božetěchova 1/2. 612 66 Brno - Královo Pole
xkalut00@stud.fit.vutbr.cz


May 8, 2024

# What is a Stack?

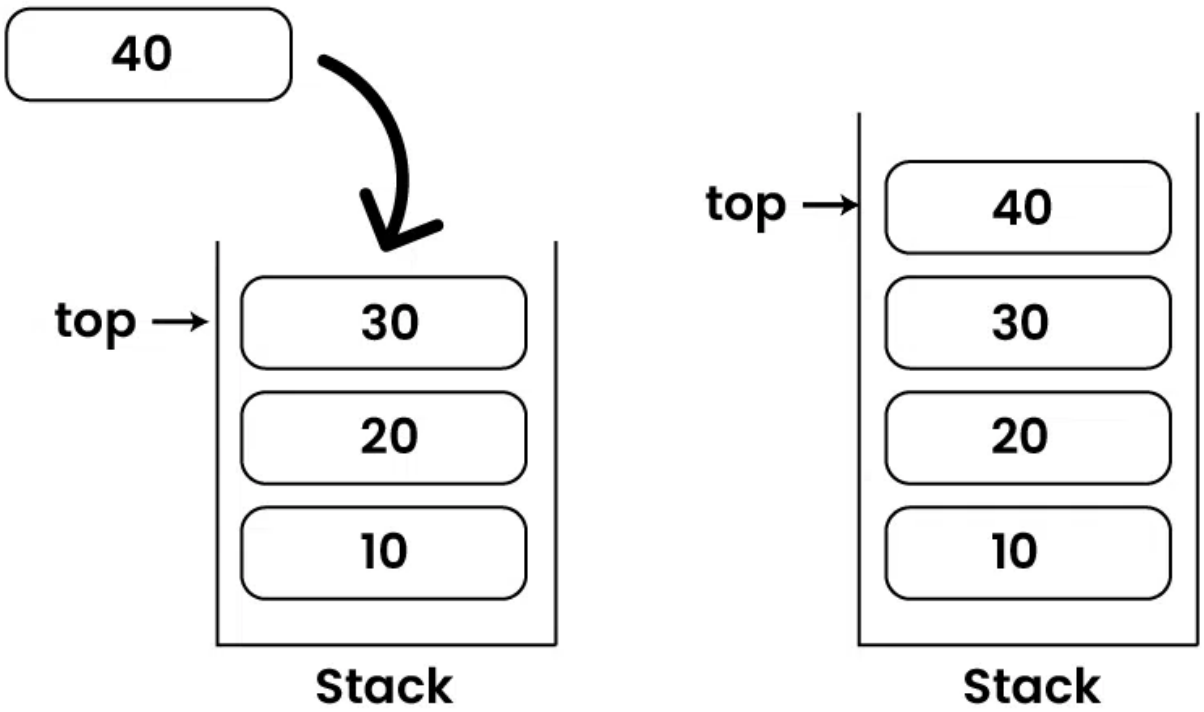
- A linear data structure.
- Follows LIFO (Last In, First Out) principle.
- Elements are added and removed from the top only.

# Operations on a Stack

1. **Push()**
   - **Complexity:** O(1)
2. **Pop()**
   - **Complexity:** O(1)
3. **Top()**
   - **Complexity:** O(1)
4. **IsEmpty()**
   - **Complexity:** O(1)

# Operations on a Stack – Push

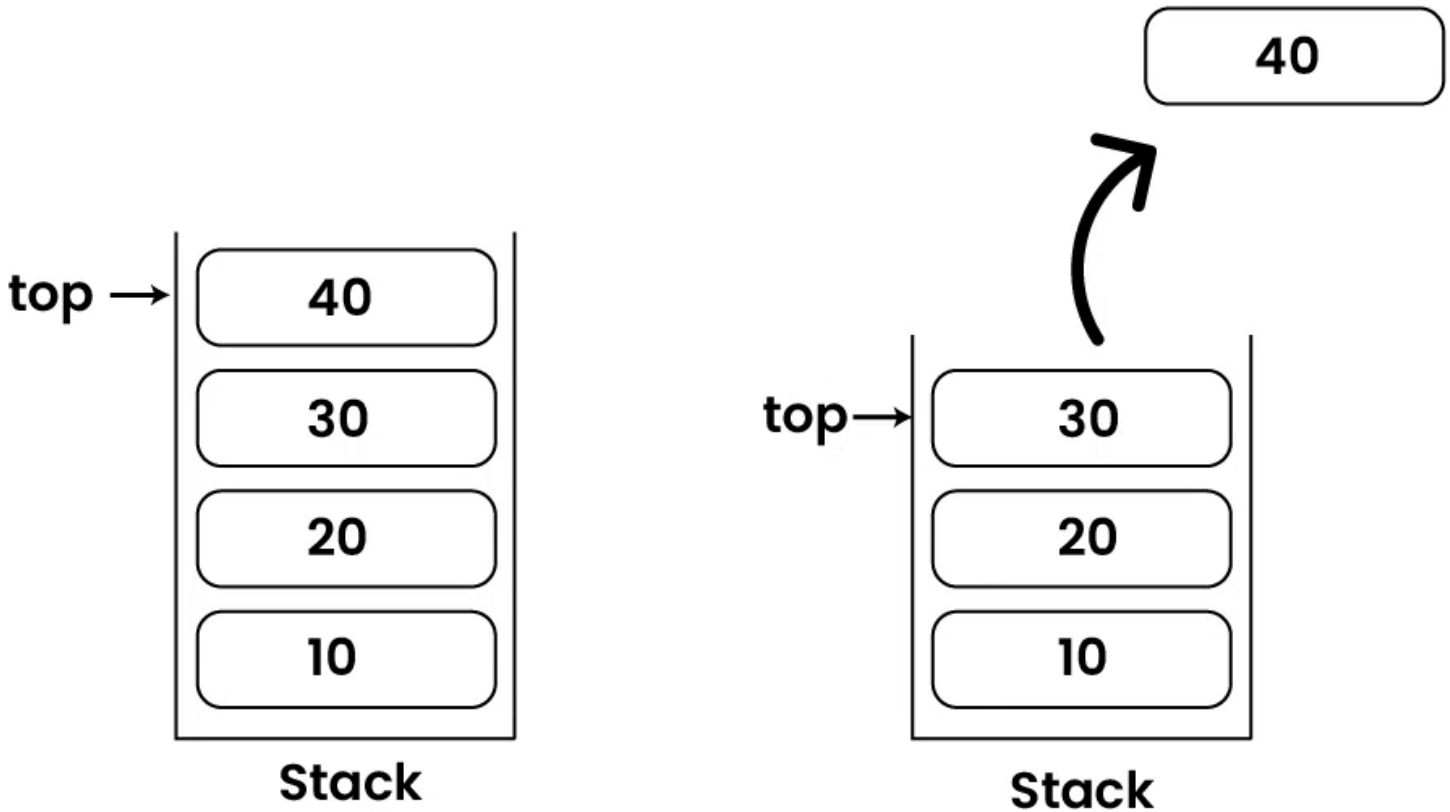
**Push:** Adds an element to the top of the stack.




Push operation on a stack. Source: geeksforgeeks

# Operations on a Stack – Pop

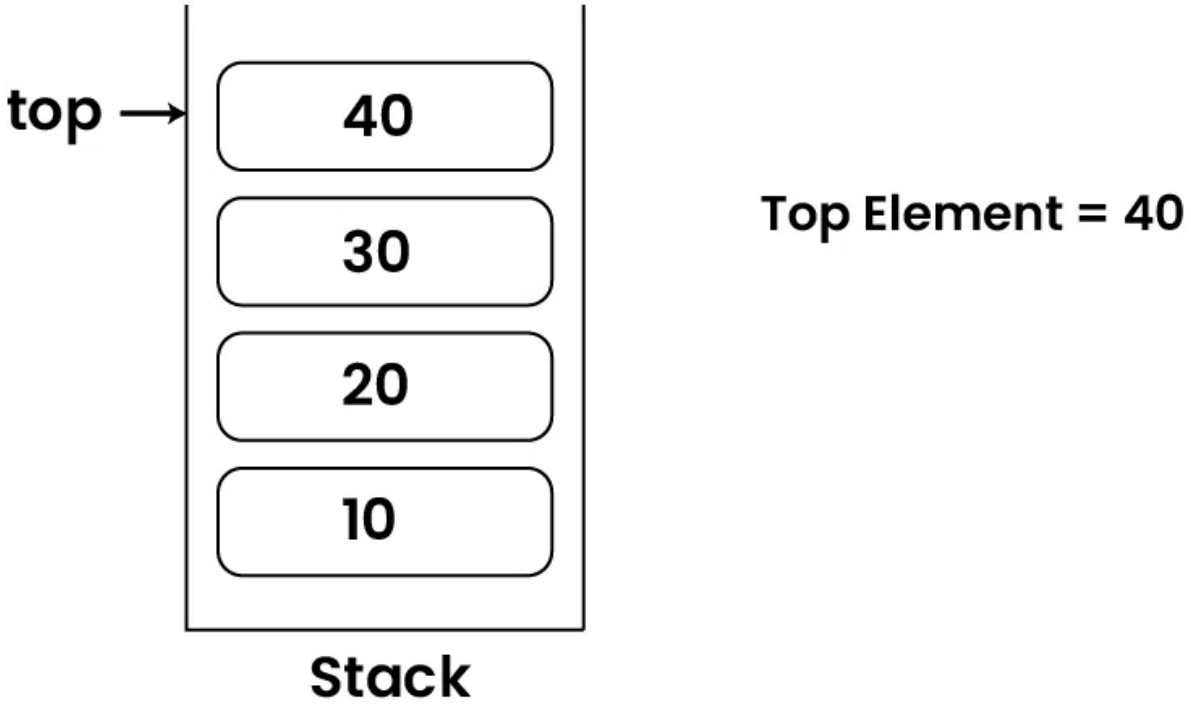
**Pop:** Removes the top element from the stack.




Pop operation on a stack. Source: geeksforgeeks

# Operations on a Stack – Top

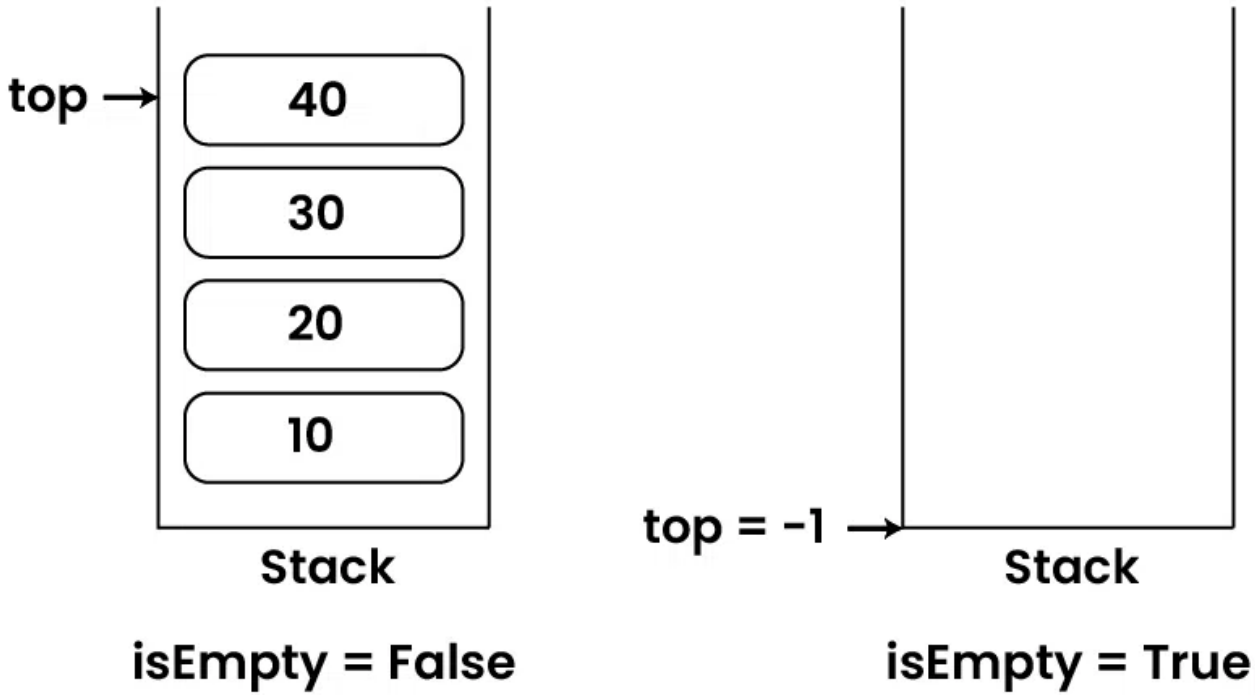
**Top:** Returns the value of the top element without removing it.




Top operation on a stack. Source: geeksforgeeks

# Operations on a Stack – IsEmpty

**IsEmpty:** Checks whether the stack is empty.




IsEmpty operation on a stack. Source: geeksforgeeks

# Implementation of Stack

- Can be implemented using arrays or linked lists
- **Arrays**: Fixed size, efficient for random access
- **Linked Lists**: Dynamic size, no memory overflow

# Implementation of Stack Using Arrays

```
#define MAX_SIZE 100

int stack(MAX_SIZE);
int top = -1;

void push(int data) {
    if (isFull()) {
        printf("Stack Overflow\n");
        return;
    }
    stack(++top) = data;
}
```

# Implementation of Stack Using Arrays

```
int pop() {
    if (isEmpty()) {
        printf("Stack Underflow\n");
        return -1;
    }
    return stack[top--];
}

int peek() {
    if (isEmpty()) {
        printf("Stack is Empty\n");
        return -1;
    }
    return stack[top];
}

bool isEmpty() {
    return top == -1;
}
```

# Implementation of Stack Using Linked Lists

```
struct Node {
    int data;
    struct Node * next;
};

struct Node * top = NULL;

void push(int data) {
    struct Node* newNode = (struct Node*)malloc
                                (sizeof(struct Node));
    newNode->data = data;
    newNode->next = top;
    top = newNode;
}
```

```
int pop() {
    if (top == NULL) {
        printf("Stack is empty\n");
        return -1;
    }
    int data = top->data;
    struct Node * temp = top;
    top = top->next;
    free(temp);
    return data;
}

int peek() {
    if (top == NULL) {
        printf("Stack is empty\n");
        return -1;
    }
    return top->data;
}
```

# Advantages and Disadvantages of Stacks

- **Advantages**:
  - Simple and efficient for LIFO operations
  - Easy to implement
- **Disadvantages**:
  - Limited access (only top element)
  - Can overflow if size limit is reached

# Sources

- https://en.wikipedia.org/wiki/Stack_(abstract_data_type)
- https://www.compnomics.in/post/introduction-to-stacks-understanding-the-lifo-princi
- https://www.geeksforgeeks.org/introduction-to-stack-data-structure-and-algorithm-t
- https://github.com/hardiksachan/CSB102/blob/main/codes/stack_using_ll.c
- https://www.digitalocean.com/community/tutorials/stack-in-c
- https://itexus.com/glossary/lifo-last-in-first-out

Thank You For Your Attention !